Knowledge-based Action Representations for Metaphor and Aspect (KARMA)

by

Srinivas Sankara Narayanan

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering:
Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

1997

Knowledge-based Action Representations for Metaphor and Aspect (KARMA)

Abstract

Knowledge-based Action Representations for Metaphor and Aspect (KARMA)

by

Srinivas Sankara Narayanan
Doctor of Philosophy in Engineering:
Computer Science

University of California at Berkeley

Professor Jerome A. Feldman, Chair

This report describes a new computational model for verb semantics. A novel feature of the model is an *active* representation of motion and manipulation verb phrases such as walk, push, slide, slip, etc. that can be used for control and monitoring as well as real-time simulative inference in language understanding. Monitoring and control parameters abstracted from the basic model provide semantic grounding for interpreting aspectual expressions and seem to offer elegant ways to solve the vexing linguistic problem of aspectual composition. Our model is able to use metaphoric projections of motion verbs to infer in real-time important features of abstract plans and events; potentially explaining the frequent use of motion and manipulation terms in discourse about abstract plans. These ideas are demonstrated through an implemented model of aspect and metaphoric reasoning about event descriptions that interprets simple discourse segments from newspaper stories in the domain of international economics.

Professor Jerome A. Feldman
Dissertation Committee Chair

Contents

Acknowledgements

I once heard Jerry Feldman say words to the following effect:

Supervising a thesis is like writing a research paper under adverse circumstances.

I can only observe in light of all my adventures over the last two years that he handles adversity extremely well. Jerry Feldman is the ideal advisor. It is rare to find people with his vision and knowledge who also have the patience to understand and deal with a stream of half-baked ideas, and more importantly take the time to sort out the important from the merely cosmetic. His ability and dedication to see through the seduction of techniques and mechanisms to the essence of a problem is truly inspiring, as is his ability to manage multiple projects in different fields and yet be able to dive into detailed technical discussions about any one of them. Every chapter of this thesis has benefited enormously as a result of discussions and ideas from Jerry Feldman. In particular, Chapter 6 is a direct result of his ideas about how to build compositional theories of x-schemas and their link to fine-grained productions.

It is a rare privilege to be able to work with a public intellectual who is widely acknowledged to be amongst the most important and original thinkers in Cognitive Science. Working closely with George Lakoff, I have been fortunate to observe his scholarship and ability to communicate important ideas to members of different disciplines. His enthusiasm, unlimited supply of linguistic examples, and support for this project has been an important motivational factor. To me, his book with Mark Johnson, *Metaphors We Live By* continues to be an exemplary standard to aim for in communicating complex ideas. Somewhat in awe, I note that in the time it took to perform the work and write this thesis, George has written three books (including a 750 page treatise on philosophy!) on subjects as wide ranging as Politics and Morality, Cognitive Mathematics, and Philosophy. I am glad somebody can do it!

I would like to thank Robert Wilensky for his comments on draft versions of this document. His early suggestion that I investigate the encoding of communicative intent in metaphoric usage has proven to be most productive and is an aspect of metaphor that remains incompletely studied and understood both within linguistics and *AI* . Preliminary results from the study reported here show that *NLP* systems that attempt to disambiguate such discourse level phenomena must be able to extract evaluative, intentional, and motivational information from metaphoric usage.

Being part in the L_0 group at ICSI provided me with a great opportunity to work with linguists, computer scientists and psychologists. My special thanks go to fellow traveler and colleague David Bailey for formal and informal sessions thrashing out many of the ideas in this thesis, to Lokendra Shastri for his ideas linking general purpose connectionist reasoning to motor control and x-schemas (much of the second half of Chapter 4 was his work) to Dean Grannes (who implemented the connectionist version of x-schemas in the SHRUTI system) and to Nancy Chang, David Andre, Masha Brodsky, and Jonathan Segal. Thanks to Michael Thielscher for ideas connecting x-schemas to dynamic logic, to Dan Jurafsky (for insights from garden-paths), Terry Regier, Ben Gomes, Collin Baker, Jane Edwards, Sarah Shull, Laura Michaelis, Hubert Dreyfus, Len Talmy, Chuck Fillmore, Eve Sweetser, Stuart Russell, and J. Dasgupta, for critiquing and offering advice that affected various parts of this and other work while at Berkeley.

I owe a great debt to N.S. Sridharan for his encouragement and advice through my years in Silicon Valley. Others who shaped my life in in various ways include Malcolm Acock (for tips on the best hiking trails), Perry Thorndike, Keith Wescourt, Lee Brownston, Jean-Claude Latombe, Chella and Sujatha Rajan, Amy Chang, Kristy Blaise, Vic Shubin, D.D. Sharma, Anil Nair, and Susan Holeridge.

A special thanks to Gurleen Grewal for being who she is. Of course, all that I have accomplished was made possible through the love and unflagging support of my parents. I dedicate this work to them.

Chapter 1

Introduction

This thesis is motivated by the observation that narratives about abstract plans and events are often filled with words and constructions that pertain to spatial motion and manipulation. To illustrate this phenomenon, consider a fairly typical newspaper article that appeared in the New York Times on Sunday, August 1, 1993.[1]

Example 1

> Britain was deep in recession while Germany was flourishing three years ago. France kept moving ahead steadily long after Germany had fallen into recession. But now France is plunging deeper while the German economy continues to struggle. Britain has been taking small steps toward stimulating its economy by cutting interest rates, and has finally started to emerge from recession.

■

Why are words/constructions of motion and manipulation such as *move ahead steadily*, *fall*, *take small steps*, *plunge deeper*, *struggle*, *be deep*, *leave*, *sink*, *start to emerge*, *stimulate*, *cut* etc. used in a narrative about international economics?

This thesis describes a project that provides evidence through computer simulation that a key reason for using motion words and phrases is that it allows for the deep semantics of causal narratives to be *dynamic* and arise from a *continuous interaction* between input

[1]Reading articles on this topic from any newspaper should readily convince the reader of the ubiquity of motion and manipulation terms in newspaper headlines and articles.

and memory. Since knowledge of moving around or manipulating objects is essential for survival, it has to be highly compiled and readily accessible knowledge. Representations meeting these criteria must be context sensitive and allow changing input context to dramatically affect the correlation between input and memory and thereby the set of possible expectations, goals, and inferences. Speakers are able to felicitously exploit this context-sensitivity in specifying important information about abstract actions and plans that take place in complex, uncertain and dynamically changing environments.

In Chapter 2, we place this work in the overall context of the Neural Theory Of Language (*NTL*) project underway at U.C. Berkeley and ICSI. The *NTL* project and this work in particular is inherently inter-disciplinary incorporating results and insights from computer science, structured connectionism, linguistics, cognitive science, and biological control theory. The central hypothesis pursued here is that the meaning of spatial motion terms is grounded in fine-grained, executable action representations that encode key monitoring and control information about the action. Simple abstractions as well as conventionalized metaphors project these features onto abstract domains such as economics enabling linguistic devices to use motion terms to describe features of abstract actions and processes. In this thesis, we emphasize a computer science perspective on the problem and are centrally concerned with issues of knowledge representation and real-time inference. Consequently, the main methodology used is computational modeling and simulation.

A crucial part of the *NTL* hypothesis is that motion and manipulation words (such as *pull*, *push*, *pull*, *walk*, *run*, *stumble*, and *fall*) are grounded in highly compiled parameterized action controllers that encode detailed control and monitoring information. Different action verbs and associated grammatical devices express **parameters** of the underlying action such as *rates*, *manner*, *forces*, and *posture and movement control parameters*, as well as associated **intentional and motivational states** such as *goals*, *resources*, *energy* and *effort*. Thus our theory requires that the representation of the semantics of action words be grounded in **active** structures with a fine granularity that are flexible and adaptive in the way context affects interpretation. Furthermore the representation should support concurrent activities in the memory storage, retrieval, and indexing process.

Chapter 3 describes a computational model that is able to satisfy the representational requirements outlined above. Our representation is partly inspired by results in high-level cortical motor control schemas (Sternberg *et al.* 1978; Bernstein 1967), leading us to refer to our verb model as **x-schemas**. X-schemas are *parameterized* routines with internal state that execute when invoked. [2] Our computational model of x-schemas is an extension to Petri nets (Murata 1989). Our extensions allow run-time binding, hierarchical action sets and stochastic transitions. Chapter 3 states and proves a theorem establishing the formal connection between x-schemas and Petri nets which allows us to tap into the vast literature on Petri nets for specification, design and analysis of x-schemas.

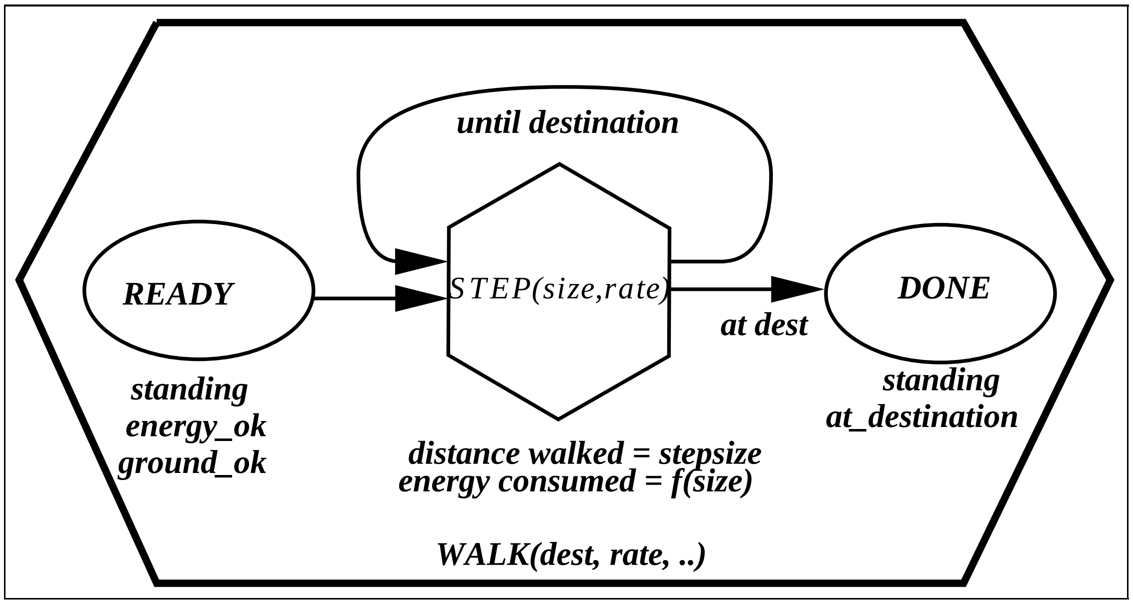


Figure 1.1: A simple x-schema for walk

Figure 1.1 shows an over-simplified version of an x-schema for walking. Later, in Chapter 3 we will describe more realistic versions. The basic walk cycle consists of a taking a series of steps until a destination is reached. STEP is therefore a sub-schema of WALK. STEP itself may be further decomposed into STANCE and SWING sub-schemas. In general, the use of hexagons for x-schemas specifies the presence of sub-schemas. The decomposition process bottoms out in atomic action controllers called *synergies* (sub-cortical parameterized continuous controllers) in Biological control theory. Parameters to the various controllers may be coded in linguistic expressions as *rates*, *forces*, etc. For instance, the WALK schema may take the *destination* (as in "walk to the store") or *rate* (walk quickly, amble) as run-time

[2] The prefix x (stands for executing) is meant to distinguish our representation them from the multitude of other notions of static or declarative schemas that pervade the AI and psychology literature.

bindings and modify its execution behavior appropriately (altering manner, direction, and being *done* when the destination is reached). Similarly the STEP x-schema may take *stepsize* or *durations* as run-time parameters and implement the behavior differently for different parameter settings. Besides being parameterized, to be properly responsive in a complex and dynamic environment x-schemas have to be tightly integrated to perceptual modalities. For instance, x-schemas must be able to respond appropriately to various perceptual and postural conditions and coordinate them to a state of readiness before starting an action. Thus to be *ready* to walk, several posture and resource pre-conditions (such as the fact that the agent must be standing and have enough energy) as well as certain world conditions (such as stable ground ahead) must be met.

The key facts to note here are that x-schemas are executable representations that are different from pre-compiled behaviors in that they allow for flexible execution based on run-time bindings. During execution, x-schemas can model conditional actions and effects (initiate sensing action, branch on results). Their fine-grained structure and asynchronous dynamics allows them to continuously monitor and respond to changes in world state. Their distributed control and ability to control parallel and concurrent actions allows them to model complex, coordinated actions. Our x-schema-based model of motion verbs clearly suggests that the semantics of such verbs is also inherently *dynamic* in this manner.

The interface between schemas and language consists of bi-directional interactions with Feature-structures or *f–structs* (signifying the similarity to traditional feature-value structures in linguistics). Figure 1.2 shows a small part of this interface. Linguistic input codes for features such as *verb label*, *path*, *manner*, and *aspect* which when parsed instantiate the f–struct by setting specific features to specific values. Thus the result of the parsing process provides information and supplies parameters required for the *initiation* and *control* of the underlying x-schema. In this thesis, we *assume* the presence of a parser which is able to instantiate the f–struct to specific values based on the linguistic input. As an example of this process, consider the linguistic input "take small steps" from Example 1. Here the result of the parsing process specifies that the x-schema being referred to is WALK and that the *step size* parameter has value *small*. The input to our model is the linking structure with the parameter values specified as described above. This parameter setting translates to executing the WALK x-schema with a small step size. The interface is *bi-directional* in that x-schema executions can modify the f–struct as well. For instance, executing the WALK x-

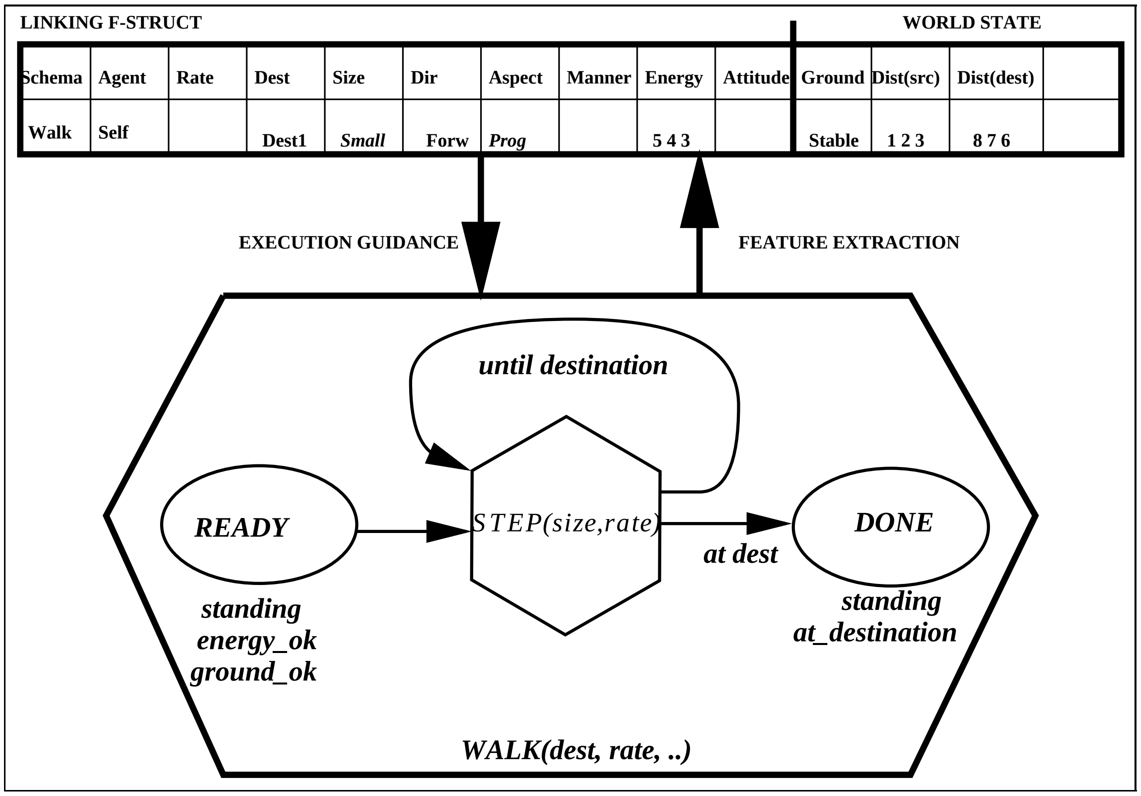


Figure 1.2: Verb phrases supply parameters to x-schemas

schema with a small step size corresponds to less distance traveled per-step, less energy used, etc. Feature values are thus updated depending on the specific execution trajectory of the x-schema. This way x-schema executions are able to add information to the initial parse by setting or modifying feature-values in the f–struct.

In this thesis, we will use the bi-directional interaction described above for real-time context-sensitive simulative inference. The crucial thing to notice is that x-schemas are always active in the context of a world state as well as agent internal state. While linguistic input supplies some parameters, the execution trajectory for a particular episode depends on internal motivational factors (such as the available energy) as well as the perceived world state (such as whether the ground ahead is stable). This allows the x-schema based model to be context-sensitive both to internal states (goals, resources) and perceived world states and evolutions.

While x-schemas provide us with a formal computational approach to investigate the use of dynamic models for grounding verb semantics, the NTL hypothesis further requires us to be able to express such models in terms of possible neural implementations. To this end, Chapter 4 answers the question *How might x-schemas be neurally encoded* ? We look at the question from two different directions, with some rather surprising results. First, we look at a connectionist encoding of x-schemas and show that primitives developed for logical reasoning by Shastri and his students as part of the SHRUTI connectionist reasoning system (Shastri & Ajjanagadde 1993), are sufficient to model the various parameters and control behaviors required for higher level motor control. The second part of Chapter 4 looks at general connectionist primitives and finds that they can be readily and naturally encoded as x-schemas. Since these primitives form the backbone of a connectionist general purpose reasoning system, x-schemas seem to capture some of the basic inferential primitives required for general-purpose reflex reasoning.

Returning to the narrative in Example 1, the reader may already have noticed the frequent reference to internal temporal details and **phases** of events. For instance, constructions like "kept moving ahead", "taking small steps", and "is plunging", "continues to struggle" presuppose that the motion in question was *ongoing* , and entail that the current focus is on the continuation of the motion, while expressions like "had fallen" seem to focus on the consequences of the described event. Similarly expressions like "started to emerge" clearly enable focus on specific stages of an event as it unfolds in time. The study of linguistic devices which profile and focus on the internal temporal character of an event is referred to in linguistic research as the study of "aspect". In conjunction with tense, aspect forms a primary means for "grounding" (back-grounding and fore-grounding) and "relating" events in narrative (Berman & Slobin 1994). Since most narratives are about events and their relations, tense and aspect are essential components of any Natural Language Understanding (NLU) system. Traditionally, it has been especially difficult to come up with a compositional semantics of aspect because in all languages studied, the *natural or inherent* semantics of a situation combine with and modify the *viewpoint* or *perspective* adopted by a speaker. This makes a compositional account of the semantics of aspect difficult giving rise to many paradoxes and problems (Dowty 1979; Moens & Steedman 1988).

Chapter 5 demonstrates through computer simulation that a compositional semantics of aspect can be constructed if we take seriously the representation of action in a neural system. Our solution relies on an abstract x-schema called the CONTROLLER, that captures a control generalization over many individual x-schemas. The controller is itself an x-schema and models important regularities that are relevant in the evolution of processes (enabling, inception, in-process, completion, suspension, resumption, etc.). We propose that such controllers may be directly coded in our neural circuitry and be available to higher level cognitive processes such as language interpretation and problem solving. The semantics of aspect arise from the bi-directional interaction of the generalized controller with the specific underlying x-schema for the verb-phrase in question. This active model of aspect is able to model cross-linguistic variation in aspectual expressions while resolving paradoxes and problems in model-theoretic and other traditional accounts. Chapter 5 provides evidence for our proposition by describing the results of a computer simulation of our dynamic system model of aspect. While our current implementation is unable to model tense, Chapter 5 also describes ongoing efforts to construct a compositional semantics of the tense/aspect combination.

In Figure 1.3, we see the WALK x-schema now recoded as the dynamic interaction of two x-schemas, namely the CONTROLLER x-schema with nodes bound to different phases (stages) of walking. Linguistic devices such as aspect specifiers can activate any of the nodes of the controller. Activation of a controller node specifies the execution state of the underlying x-schema resulting in a modified execution trajectory for different settings. For

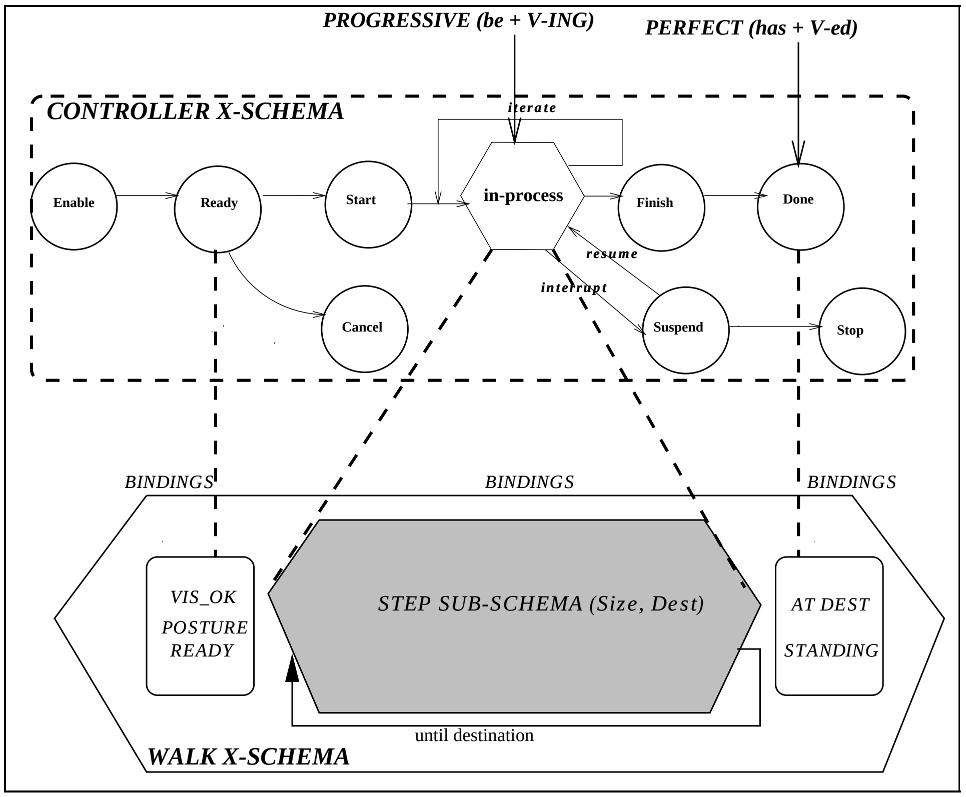


Figure 1.3: The CONTROLLER x-schema models important regularities in process monitoring and control. Grammatical devices such as morphological modifiers refer to specific nodes in the CONTROLLER x-schema, which binds to specific activation states of the verb-phrase x-schema, generating the required interpretation. Shown above are the cases where an *ongoing* activity is specified using the *progressive* construction (English *be V-ing*) or the consequent state is picked out using the *perfect* construction (English *has V-ed*).

instance, the Perfect aspect (represented by the grammatical construction *has V-ed*, as in "*has walked* to the store") activates the consequent state (the *done* node) of the controller, which focuses on the consequences of walking to the store (being there, buying things, etc.). Contrast this to the use of the Progressive construction, (represented by the *be + V-ing* construction in English as in "*is walking* to the store"). Here the *process* node is active, focusing on facts such as that the walker is actively taking steps toward the store, expending energy, moving closer to the store, etc.

Thus, meaning arises from the **dynamic binding** of a specific activation state of the controller x-schema to a specific activation state of the verb-complex x-schema. The structure of the controller and the relevant set of features that characterize the verb-complex jointly control the compositional possibilities. Chapter 5 describes these features and the results of our computational model on various problems and paradoxes in traditional accounts.

While abstraction from motion control and monitoring primitives seems to be a powerful method to address problems in verb semantics, it still doesn't explain the high frequency usage of motion and manipulation terms to describe features of international economics. Here we come to the second part of the thesis, which is based on the observation that systematic metaphors project motion features onto abstract domains such as economics, enabling linguistic devices to use motion terms to describe features of abstract actions and processes.

In our theory, metaphors such as the Event Structure Metaphor (Lakoff 1994) use the dense and familiar causal structure of spatial motions and object manipulations to permit reflex (fast, unconscious) inferences. Conventionalized projections from motion features allows the speaker to use terms that rely on the highly familiar structure of spatial motion and object manipulation to communicate complex scenarios, dynamically changing goals, resources, monitoring conditions, and communicative intent.

Figure 1.4 shows a simple example of this phenomenon. Consider the use of the phrase "Britain has been taking small steps ..." in Example 1. First, the word "small" codes for the *size* of an individual step as *small*. Second, the construction "has been taking" codes for the fact that the steps are *ongoing* and continuing (the agent is continuing to walk). Taking small steps also implies that there is a large distance still to go to reach the destination, and the low step granularity indicates that the walker is able to monitor

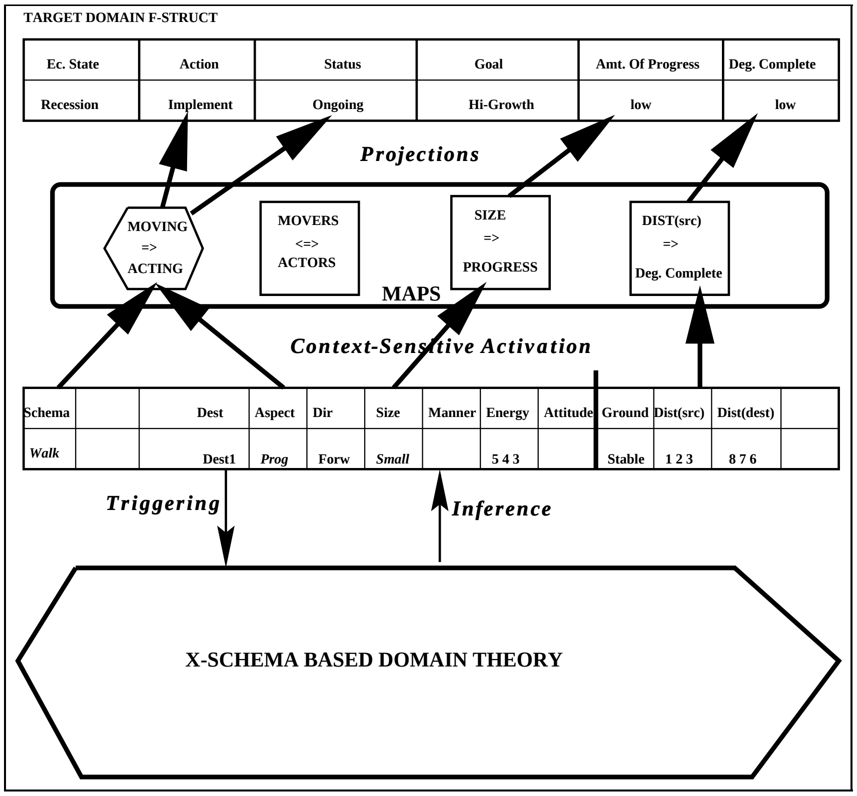


Figure 1.4: Metaphors project motion features onto features of abstract actions and policies. Maps may project objects and roles between domains (shown as rectangles) or actions and events along with their aspect and other parameters (shown as hexagonal maps) from the domain of spatial motion and manipulation to the abstract domain of international economics. Shown are the maps that become active while processing the input "Britain is taking small steps ...". Entries directly affected by linguistic input are shown in italics (Aspect = *Prog*, Schema = *Walk*, Size = *Small*) and the target domain feature Country = Britain (not shown)). Projections of these features and others inferred from x-schema executions results in setting the target feature structure to the values shown.

and be quite responsive to changing world conditions (such as unstable ground ahead, etc.) between steps (responsiveness between steps decreases with larger step sizes or higher rates). Crucially, all the implications above stem from the knowledge of walking and taking steps. Now consider the effect of projecting these features onto the domain of abstract economic policies and actions. The well known cross-linguistic mapping from the domain of spatial movement and manipulation onto the domain of abstract actions is called the Event Structure metaphor (Lakoff 1994). From conventionalized metaphoric mappings such as *Action IS Motion*, *Actors ARE Movers*, *Size of Motion* IS *Granularity of Action*, and *Distance to Destination* IS *Degree of Completion* the hearer is able to conclude that Britain is making *small policy adjustments* to an *ongoing* recovery policy. The recovery is likely in its early stages (with a low degree of completion) and the incremental nature of the adjustments allows the British policy makers to be *responsive* to an unpredictable and changing economic environment. All this information is readily communicated to non-expert economists (who are, however, experts in walking and moving around). Of course, other similar expressions such as *cautious step*, *giant steps*, *measured steps*, *large strides*, *adroit move*, *sidestep* and *rush headlong* are routinely found in discourses about economic policies and readily interpretable in exactly the same manner.

From Figure 1.4, it is clear that in order to investigate the metaphoric projections of familiar motion and manipulation terms through computer modeling, we minimally need the following entities:

1. A representation of the highly familiar and compiled *source* domain of the mapping/projection (the bottom hexagon in Figure 1.4). A crucial requirement for the representation of the source domain is that the x-schema properties of context-sensitivity, high responsiveness, and real-time inference be preserved.

2. A representation of the ontology and relations between concepts present in the international economic discourse (the top rectangle entitled TARGET DOMAIN F–STRUCT in Figure 1.4). This domain is called the *target* domain (the target of the mapping). In this thesis, the target domain is the domain of international economic policies. One crucial requirement for designing a representation of the target domain is the ability to compute the *global* impact of new observations and evidence both from direct linguistic inputs and from projections of inferential products of x-schema executions.

3. A representation of conventionalized metaphor maps (shown in the middle of Figure 1.4) that project features from source to target domains. In our theory, metaphor maps are first-class objects that can be arranged in hierarchies. A requirement of metaphor maps is that they be able to project both *roles* (and objects) from source to target domains (e.g. Actors ARE Movers) as well as *actions* (and events) (e.g. Moving $\Rightarrow$ Acting). One novel feature of our action mapping is that it is of fine enough granularity to project the kinds of fine temporal and aspectual distinctions mentioned above and described in detail in Chapter 5 and Chapter 9.

Chapter 6 addresses the first of the three issues above. Here we come up with a compositional theory of x-schemas. The central idea behind our compositional theory is that x-schema transitions may **set**, **get** and **modify** values of the Agent's state (f–struct). This allows other x-schemas to be activated, inhibited, or their execution modified in some manner. Our model is able to exploit the active and highly responsive nature of x-schemas. The rationale for the highly compiled representation is obvious since quick responsiveness, tight coupling between action and reaction, built-in obstacle avoidance and failure recovery strategies, etc. are required for survival while moving around. This thesis argues that this degree of reactivity is crucial for language processing, and that we can use the same active representation for language understanding relying on metaphoric projections to abstract domains.

Chapter 7 describes our model of the domain of international economics. Our ontology of the domain is fairly simple and consists of entities and relations frequently used in discourse about international economics. Our computational model consists of multi-valued economic policy variables modeled as a Bayes Network (Pearl 1988; Jensen 1996). This allows us to use well understood off–the–shelf algorithms to compute the global impact of network updates from metaphoric projections as well as direct linguistic input. The inherently temporal nature of events (and their descriptions) requires us to store multiple temporal slices of the network. Such a network is called a temporally extended Belief network with temporal links from variables at time-slice t to variables at time-slice $t+1$. The inference algorithms that compute the global impact of new observations on such a network are able to propagate influences both forward and backward in time.

Chapter 8 describes our overall computational model which includes the third important piece, namely **metaphor maps** that project specific features obtained from x-schema executions onto features of the domain of international economics. When active, metaphor maps allow the real-time inferential products of x-schema executions to influence the target domain network by setting evidence for specific features of a given temporal slice of economic policy network to specific values. A novel feature of our model of metaphor maps is their ability to project bindings over multiple time steps (both forward and backward in time), an ability essential for reasoning about event descriptions (ref. Chapter 8 for details). In addition, the projections are context-sensitive in that activating a metaphor map requires that the context (domain of discourse) be identified as being about the target domain, thus shutting off projections otherwise. As mentioned earlier, metaphor maps are first-class objects that can be arranged in hierarchies. The implemented model is capable of projecting events (along with their aspect and other parameters), as well as roles and objects across domains. Chapter 8 also describes the basic computational model for metaphor interpretation which crucially depends on the quick inferences provided by x-schema executions. The implementation is described through a trace of the program on a discourse fragment from a newspaper story in international economics.

To study and model the nature and type of information provided by motion terms in the domain of international economics, we collected a set of around 30 two-to-three-line discourse fragments from standard sources such as The New York Times, Wall Street Journal and The Economist. These "stories" drove the development of our program (the set of x-schemas, the target domain Belief network constructions and the metaphor maps implemented). Once the program was developed, we tested its applicability and performance on a few previously unseen " test stories". Chapter 9 describes results on both sets of discourse fragments. Along the way we try to indicate additions or modifications that were required to process new stories. In general, the results are encouraging and show that our architecture can model important aspects of the usage of motion words in describing policies and events in discourse about international economics. One interesting and potentially important feature that emerged during program development was that the use of embodied terms (with their easily accessible emotional content) often communicates important discourse information, such as the speaker's evaluation of a policy or his attitude towards a specific actor. Even our limited implementation produced some interesting results in this area, and it is clear that further systematic study both from corpus linguistic research as well as from the computational modeling of discourse is likely to be highly productive. We believe that our results show that embodied term usage is often the most felicitous method for communicating information about dynamically changing goals, resources, and intentions in a complex and uncertain environment. Therefore, a dynamic representation of embodied inference seems crucial to pick up these features and project them onto abstract domains like international economics.

Chapter 10 concludes with a summary of the basic results and some ongoing and proposed work. Chapter 11 outlines some of the program parameters and database of stories. Chapter 12 and Chapter 13 show the program behavior for selected examples from the database.

1.1 Contributions

This thesis demonstrates through computer simulation that the high degree of context-sensitivity inherent in a dynamic representation of the semantics of motion and manipulation terms is routinely utilized to *specify* control, monitoring, resource and goal-based information about complex plans and processes that operate in an uncertain and dynamic environment. Our model shows the capability of making these discourse inferences in real-time, consistent with the fact that such information is routinely available as reflex, automatic inference in narrative understanding. Outlined below are the central novel aspects that emerge from the work described in this thesis.

- A dynamic system model of motion and manipulation verbs that is inspired by high-level motor control. A result of our representation of verb-complexes is that the same representation can be used for planning and control and for reasoning about action descriptions.

- A computational model that demonstrates through simulation that a compositional semantics of linguistic aspect can be constructed from schematized generalizations recurring in process monitoring and control. This result is interesting because aspect is a well studied and seemingly intractable problem in linguistics.

- A computational model of metaphoric reasoning about event descriptions. A crucial aspect of the model is its capability to exploit domain knowledge of spatial motion and manipulation for real-time simulative inference. Results of applying our model on discourse fragments from newspaper stories in international economics show that crucial facts about abstract plans, goals, resources and intent are communicated by projections from embodied concepts. Our results also show that the use of motion and manipulation terms provides important discourse-level information about communicative intent.

1.2 Road map

Chapter 2 situates this project within the overall effort to study the link between language and embodiment underway at U.C. Berkeley and ICSI. Chapter 3 and Chapter 6 describe the novel representation developed in this thesis called x-schemas which are used to model highly compiled domain knowledge of motion and manipulation. Chapter 4 shows the equivalence of x-schemas to standard connectionist models of general purpose reasoning. Chapter 5 describes the *controller* abstraction and the results of our computational model in constructing a semantics of aspect. Chapter 7 and Chapter 8 describe the technical details of the metaphor interpretation system. Readers not interested in the implementation details may skip most of Chapter 7 and the latter half of Chapter 8 entirely and go directly to Chapter 9 for the results of the metaphor interpretation system. An overall view of the model without the implementation details can be found in (Narayanan 1996b). Readers not interested in the linguistic issues of aspect may skip much of Chapter 5. A brief paper on the main ideas and the computational model of aspect can be found in (Narayanan 1997).

Chapter 2

Language and Embodiment: A Computational Approach

The Neural Theory Of Language (*NTL*) project underway at UC Berkeley and ICSI (details of the project can be obtained from http://www.icsi.berkeley.edu/LZERO) is based on the premise that known facts on how the brain works can combine with empirically observed linguistic phenomena to provide important constraints in constructing models of language acquisition and use. Outlined below are brief descriptions of relevant facts about brain function and results of recent linguistic research. The exposition is necessarily brief and readers interested in the link between brain and theories of higher-level cognition can refer to (Feldman & Ballard 1982) for foundational work on building computational theories of higher level cognition based on neurobiological constraints. A more recent exposition on the subject can be found in (Ballard 1997). For a novel solution to the problem of how a brain-like mechanism may perform logical reasoning in real-time, the reader is referred to (Shastri & Ajjanagadde 1993). We will be returning to this model in Chapter 4. A good exposition of learning theories for brain-like models can be found in (Valiant 1996).

Readers interested in the link between general purpose language and embodiment are probably familiar with Lakoff and Johnson's pioneering work on the subject in (Lakoff & Johnson 1980). The implications of such theories on Grammar can be found in Langacker's work on Cognitive Grammar (Langacker 1987) and Lakoff's book on Cognitive Semantics (Lakoff 1987). The thesis of embodiment consistent with the effort outlined here can be found in (Johnson 1987). Philosophical implications of embodied semantics are investigated in great detail by Lakoff and Johnson in their forthcoming book (Lakoff and Johnson 1998?).[1]

2.1 Constraints From Brain Function

The capacity of the brain with 10^{11} computing elements and 10^{15} connections points to the awesome potential for information storage, processing and communication. While much is not known or understood, accepted facts about brain function produce surprisingly powerful constraints for modeling higher level cognition. First, neurons are extremely slow computing devices and essentially operate in the *millisecond* range. The numbers above also indicate that there is nowhere near full connectivity (which requires $\approx 10^{24}$ connections) and that the brain is sparsely connected with average fan-in/fan-out of about 10^4 connections. A neuron "fires" by changing chemical state producing an electrical signal out of all their branches. A neuron may have upto thousands of branches connected through tiny gaps called *synapses*. The receiving neuron fires based on a spatio-temporal thresholding function of its state and the excitation and inhibition at its incoming synapses. Furthermore all senses, muscles and the CNS communicate essentially in the same way.

Neural firing rates are typically $\approx$ 100 spikes/sec. In conjunction with the slow operating times of neurons this sets up a low communication bandwidth which in turn means that inter-neuron messages must be very simple. In general, such messages cannot be of sufficient resolution to encode complex symbolic names, pointers or structures. While there is no evidence of central control, there is good evidence of timing mechanisms that operate over multiple time scales (Ballard *et al.* 1997). It is also clear that there is also a great deal of sophisticated brain structure and mental capacity already present at birth. In terms of connectivity, the evidence suggests that there are as many top-down as bottom-up connections. Also, while there may be specialized centers in the brain performing dedicated functions, there is no evidence of isolated brain regions; typically even specialized centers receive inputs from and provide outputs to multiple other regions.

[1]Recently there has been a renewed interest in Embodied Cognition evidenced by the recent AAAI Symposium on Embodied Cognition (MIT 1996). We note that our notion of embodiment differs from that of the Behavior-based Robotics group (Brooks 1986) in that we believe that a critical aspect of embodiment concerns the structure and properties of the human brain.

These elementary facts about the brain have serious implications for language processing. An early example (Feldman & Ballard 1982) of a serious processing constraint is the 100 step rule. A direct consequence of the slow neuron operating times is that typical reaction times are only about about 100 times the average neuron operation time. The 100 step (number of unit operations) constraint implies that much of language processing must be done in a highly parallel fashion, relying on the high connectivity and active nature of neurons. The high reaction time further suggests that the correspondence between brain structure and function must be direct, there is no time to reinterpret structure differently for different functions since any re-interpretation has to be done by the same slow neurons. In summary, uncontroversial (relatively) facts about brain function taken together suggest that a) memory is active (no separate interpretation process) and b) a single single brain structure does representation, inference, action and learning and c) while there is specialization there is no evidence to postulate an autonomous, isolated module in the brain.

Study of the role of brain constraints in questions of higher-level cognition have motivated the field of Structured Connectionism since the early-eighties (Feldman & Ballard 1982). The basic idea was to create modeling frameworks that respected important neural constraints such as the ones mentioned above to investigate computational models of cognitive function. Importantly, structured connectionist models attempt to abstract from the details of brain neuro-chemistry to important known information processing constraints that reflect both *strengths* and *weaknesses* of brain like processing.

In structured connectionist models, a *unit* corresponds to a small number of idealized neurons, a *link* corresponds to an idealized synaptic connection. Computationally, this usually translates to the following computational constraints.

1. Units compute spatio-temporal threshold functions of their inputs. Typical examples are weighted spatial sums (the standard McCullough-Pitts $\sum$ unit) or spatial products (known as π units). Structured connectionist models also use the temporal averaging and summation. Examples include the τ-AND and ρ-btu units discussed in Chapter 4.

2. Units can hold a small, discrete number of states or activation levels.

3. Unit outputs do not have sufficient resolution to encode symbolic pointers or structures. Messages between units are thus extremely simple.

4. There is no central controller instructing individual nodes to perform specific operations at specific times.

A summary of early attempts to use structured connectionist models for language understanding can be found in (Feldman & Waltz, *eds.* 1988). Description of connectionist models of word-sense access and disambiguation can be found in (Cottrell 1985; Lange & Dyer 1989). More recently there have been efforts to model parsing (Henderson 1994), figurative noun-adjective combinations (Weber 1989), and semantic network representation and inference (Shastri & Ajjanagadde 1993). A good source for the more recent work is the edited three volume book (Barnden & Holyoak, *eds.* 1994). An early result of the *NTL* project was Regier's structured connectionist model that was able to model the acquisition of spatial relation terms in different languages (Regier 1996).

2.2 Theories Of Embodiment in Cognitive Semantics

An important reason to search for embodied theories of cognitive phenomena is their explanatory power. An exemplary demonstration comes from research on color terms by (Berlin & Kay 1969; Kay & McDaniel 1978). Berlin and Kay's research showed how the small number of *named* basic colors found in all human languages can be explained by the physiology of the visual system which is able to predict the spectral characteristics of these specific colors. The universal structure of the visual system thus explains the universality of cross-linguistic patterns in basic color terms.

In linguistics and cognitive semantics, there is mounting evidence that many abstract concepts derive their meaning from embodied concepts. Lakoff and Johnson (Lakoff & Johnson 1980) show that many concepts used in everyday discourse are derived from metaphoric mappings of embodied concepts. Talmy suggests (Talmy 1987) that our understanding of the meaning of causal terms can be modeled by schematized primitives of force-dynamic interactions. Eve Sweetser (Sweetser 1980) extends the force-dynamic model and proposes that the semantics of modals can be constructed from metaphoric projections of Talmy like force-dynamic schemas. Here, abstractions from embodied experience of forces and their projections are proposed as an essential component of the meaning of causal terms in language. In this thesis, Chapter 5 argues that the inherent temporal structure of events and their aspectual profiles can be modeled by abstract control schemas.

In addition, recent findings suggest that our conceptualization of the structure of events depends to a large extent on our conceptualization of spatial motion and manipulation. An example is the Event-Structure Metaphor (Lakoff 1994) that has been found to be present in many languages. A few of the significant mappings are reproduced below to give a flavor of the kind of correspondences found.

1. States are conceptualized as locations.
2. Changes are conceptualized as movements into or out of states.
3. Causes are conceptualized as forces.
4. Purposes are conceptualized as destinations.
5. Plans are conceptualized as Paths.
6. Acting is conceptualized as Moving.
7. Difficulties are conceptualized as obstacles to motion.
8. Expected Progress is conceptualized as a travel schedule.
9. Goals-based policies are conceptualized as Journeys.

Clearly, one can easily come up with a wide range of expressions that articulate the mappings above. Take the example of obstacles to motion being projected as plan difficulties. One routinely finds expressions like *faced a brick wall*, *jungle of regulations*, *weighed down*, *uphill*, *find loopholes*, etc. that use this projection to specify aspects of complex plans with embodied terms. Chapter 8 makes use of this and other mappings in our computational model of metaphoric projections of embodied events and actions.

Recent linguistic work by Joe Grady (Grady 1996) suggests that large complex mappings such as the Event Structure map outlined above are best conceptualized as compositions of more primitive maps. Central to his argument is that these primitive maps are embodied or based in direct experiential correlations. Examples of such embodied maps may be Destination $\Rightarrow$ Goal, or More $\Rightarrow$ Up, or Fall $\Rightarrow$ Fail. Grady further argues that a key benefit of viewing complex metaphors as compositions of embodied maps are that some vexing problems regarding incomplete mappings and target domain overrides (certain entities are not mapped because of target domain constraints) go away if the embodiment thesis is taken seriously. This obviously suggests that the set of embodied mappings and the constraints of embodiment are crucial for computational models of metaphor use and extension.

2.3 The *NTL* Hypothesis

With this background, we are ready to state the central hypothesis motivating this thesis. The basic conjecture is that merging results of research in Cognitive Semantics with connectionist representations can yield explanatory theories of language acquisition and use. This conjecture is the *NTL* hypothesis.

Conjecture 1 The *NTL* Hypothesis

1. *Many basic concepts are directly embodied in perception, motor control, emotions, and social cognition.*

2. *Specific perception, action, and force-dynamic abstractions of embodied concepts serve as semantic primitives of abstract concepts. Abstract concepts may also derive their meaning through metaphorical and other mappings from embodied concepts.*

3. *Structured Connectionism provides the right framework to investigate these issues in detail.*

This thesis investigates the overall hypothesis for a small segment of language, namely how *active* representations of motion and manipulation words (developed in Chapter 3 and called x-schemas) based contain features that may directly serve as semantics for describing abstract actions and events. To this end, Chapter 5 shows how the active x-schema representations developed in Chapter 3 can model the semantics of linguistic aspect. Chapter 8 describes a computational model that is designed to study how metaphoric projections of motion and manipulation words may provide important information about international economic discourses. Chapter 9 outlines our model's performance on discourse fragments reflecting motion term usage in newspaper stories on international economics. A parallel dissertation project (Bailey 1997) demonstrates how x-schema like representations provide sufficient inductive bias to make the problem of hand-action verb acquisition tractable.

The NTL hypothesis suggests structured connectionism as the appropriate framework to investigate the role of embodiment in language acquisition and use. However, rather than directly express the language phenomena studied in this thesis (aspect and metaphor) directly in connectionist terms, we found it more convenient to work at a more abstract level, which we will call the *computational level*. We do require models expressed at the computational level to be reduced to the connectionist level. This way, the computational level serves as a convenient language for model development and testing, allowing human modelers to understand and communicate model features and results. Detailed simulations and predictions of our cognitive models may require using the reductions to the connectionist level.

The main mechanisms developed in this thesis, namely x-schemas and metaphor maps are expressed at the computational level. In Chapter 4, we show that the reduction from x-schemas to their connectionist realization. We are also able to model standard connectionist primitives as x-schemas. Given that we have a compositional theory of x-schemas (ref. Chapter 6), we think that x-schemas (which are representatable as Stochastic Petri Nets (ref. Chapter 3)) may be good candidates for formal representations of structured connectionist theories. If this turns out to be true, we can avail of the vast array of available algorithms for structural and dynamic analysis to answer questions of scalability and dynamic behavior of our models.

While x-schemas have been reduced to structured connectionist models, our model of metaphor interpretation (Chapter 8) has not been reduced to the connectionist level. While we can see our way through to an implementation of metaphor maps as connectionist networks, the use of Belief nets to represent knowledge of international economics and the associated inference algorithms to compute the global impact of network updates present a more serious challenge for connectionist implementation. We don't yet know of a good way to accomplish this reduction, and this is a shortcoming of the thesis that we will try to remedy in future work. Chapter 8 (Section 8.6.2) outlines some initial thoughts in this regard.

We are well aware that computational perspectives and information processing abstractions of brain function (such as connectionist models) may *fall far short* of addressing many complex issues in language understanding and cognition in general. In particular, we currently see no way of addressing questions of subjective consciousness or *qualia* (which are clearly present even in our understanding of embodied words like push or walk) within a connectionist (or any) framework. However as the reader correctly suspects, we have a *long way to go* with many open and incompletely understood problems before we get to such obviously unsolvable questions.

Chapter 3

An Executable Model of Action Verbs

Controlling goal-directed behavior in a complex, uncertain and dynamic environment requires an agent to develop **active** representations that tightly couple action and reaction and are highly responsive to environmental as well as intentional and motivational changes (such as new goals or resources). Consider an agent carrying out simple intentional embodied actions such as *pull*, *push*, *crawl*, *grasp* or *suck*. Clearly in each case, the action specified requires controlling and coordinating multiple actions in parallel and in real-time (such as being able to *preshape* the hand while *reaching* for the object to be pushed), being highly reactive to perceptual and motor input in a dynamic and uncertain environment (e.g. object to be pushed moves); and being able to monitor execution and progress as well as recover from minor failures if necessary (e.g. recover from temporary instability while crawling).

But what does all this have to do with verbs and language? This Chapter argues that the semantics of motion and manipulation verbs is grounded in a executable model that encodes key features of the entailed action. Different action verbs and associated grammatical devices code for **parameters** of the underlying action such as *rates*, *manner*, *forces* and *posture and movement control parameters*, as well as associated **intentional and motivational states** such as *goals*, *resources*, *energy* and *effort*. The key observation that motivates our representation of embodied action words is based on a hypothesis that corresponds to the first part of the Neural Theory of Language conjecture (NTL) outlined in Chapter 2. While the NTL conjecture applies to a whole variety of embodied concepts, this Chapter will be concerned with providing evidence for the NTL conjecture as applied to embodied action and event descriptions.

In the case where the word labels an intentional motor action, our representational mechanism is able to model the high-level action monitoring and control required to carry out the action (in a command obeying mode). The representation is partly inspired by results in high-level cortical motor control schemas (Sternberg 1978; Bernstein 1967), leading us to refer to our verb model as executing schemas or **x-schemas**. X-schemas are *parameterized* routines with internal state that execute when invoked. Linguistic input such as the *verb label*, *path*, *manner*, and *aspect* provides information and supplies parameters required for the *initiation* and *control* of the underlying x-schema. Thus x-schemas are *dynamic*, fine-grained, distributed action controllers that tightly couple action and reaction in an uncertain and rapidly changing environment. Our x-schema based model of motion and manipulation verbs clearly suggests that the semantics of such verbs is inherently *dynamic* in this sense.

X-schemas are executing graphical structures that can be used for planning and monitoring an action *or* to provide real-time simulative inference. The version of x-schemas that we are currently using is an extension of a formalism known as Petri nets in computer science (Reisig 1985). Their most important features are the ability to model both events and states in a distributed system, clean ways to capture concurrency and event-based asynchronous control in addition to the control primitives of sequence, conflicts and decisions. Our extensions include the addition of hierarchical control, parameterization and stochasticity. We state and prove a theorem establishing the formal connection between x-schemas and Petri nets, which allows us to tap into the vast literature on Petri nets for specification, design and analysis of x-schemas. We believe that x-schemas provide a good formalism for modeling many important issues in embodied cognition.

The rest of this chapter provides some motivation for our choice of representation both from research in biological control theory as well as from mobile robotics and AI. We then outline some computational requirements of x-schemas and our formal model of x-schemas with examples.

3.1 Motivation for X-schemas

The necessity to act fast in an uncertain and dynamic world requires reactive planning agents (biological or robotic) to develop representations that can tightly couple action, execution monitoring, error-correction and failure recovery. Our inspiration for x-schema design therefore comes from the field of biological control, from research in AI and Robotics as well as previous attempts to use similar representations in modeling high-level cognition.

3.1.1 Biological control theory

The use of the word *schema* to explicitly refer to the mental and neural representation of execution parameters for an action appears to have been coined by Head (Head 1920) to refer to posture maintenance and control. However, the first proposal that *schemas* included storage of movement information including predicted outcomes and error information to drive learning appears to have been made by (Schmidt 1975). Furthermore, ever since Bernstein's (Bernstein 1967) pioneering work, research in biological control theory seems to have accepted the notion of *motor synergies*, which are sub-cortical parameterized continuous feedback controllers for stereotypical motions such as the vestibulo-occular reflex.

While motor synergies control simple reflexes, most complex action control requires the coordination of multiple synergies. Evidence of a precompiled sequential motor plan comes from (Sternberg *et al.* 1978) who focuses on delays in starting or stopping typing sequences depending upon the length of the string to be typed. The work on on grasping and picking up a ball (Jeannerod 1986) demonstrated the need for concurrent activation of multiple synergies (arm movement and preshaping), serial execution, as error-correction based on perceptual input (vision and touch) (Arbib 1994), and timed cerbellar modulation (Bullock 1995).

The temporal details of motor actions are often mediated by afferent feedback. The Hering-Breuer reflex in the mammalian respiratory system is a good example of this phenomena. Here, feedback from the pulmonary stretch receptors activated during lung inflation terminates the respiratory phase. (Feldman 1986) reviews the role of afferent feedback and the effect of deafferentiation in the mammalian respiratory system.

Researchers have also investigated the role of afferent signals in the walking system of the cat, where a sensory signal at the end of the stance phase switches the motor program from stance to swing (Grillner 1985). (Pearson 1993) reviews the various results and hypothesis regarding the role of afferent feedback in vertebrate and invertebrate motor control. He concludes that the basic reason for afferent feedback is to produce effective movements motor output must be synchronized and coordinated with ongoing positions, forces and movements in peripheral structures. Such feedback signals are important for establishing the execution trajectory of motor programs; they may reinforce ongoing motor activity or conversely help switch from one synergy to another.

Tanji and Shima (Tanji and Shima 1994) have found that the cerebral cortex of monkeys contains cells whose activity is exclusively related to a sequence of coordinated multiple movements. They found such activity in the supplementary motor cortex (1, 2) and hypothesize that these cells contribute a signal about the control of multiple anticipated movements for planning several movements ahead. Recent evidence further suggests that some motor areas involved in planning motor sequences are active even when actions are only thought about, including mental imagery or imitative imagination (Gallese 1996; Grafton 1996). This result suggests the possibility of a single mechanism for high-level motor control and action-verb semantics.

All the research above serves as an inspiration for the model described here. It must be noted that the literature on motor control is vast and often controversial since the issues are often extremely delicate and complex. From a biological standpoint, it can be argued that understanding motor control is at least as important as understanding how language works, since it affects all organisms. In this thesis, we argue that postulating an explicit connection between language and certain aspects of high-level motor control and planning allows us to look at converging constraints from both fields (and their numerous associated sub-fields and branches). While the work reported here mostly attempts to argue for the dynamic and active nature of action representations in language understanding, in Chapter 5 we will see how aspectual distinctions made in language point to a richer and more fine-grained representation of events than is normal in classical theories of actions.

3.1.2 AI and Robotics

In AI, formal approaches to model the ability to reason about changing environments have a long tradition. This research area was initiated by McCarthy (McCarthy 1969), who claimed that reasoning about actions plays a fundamental role in common sense. A decisive advantage of deductive approaches to reason about actions is the universality inherent in any logical framework. A purely logical axiomatization which is suitable for temporal prediction, can just as well be employed to answer more general questions such as *postdiction* (i.e., what can be concluded form the current state as to states in the past) and *planning* (i.e., how to act in order that the system evolves into a desired state).

Deductive approaches have suffered from several well known problems, the most famous of which is the *Frame problem* pertains to compactly specifying aspects of world that are unchanged when an action is executed. A number of associated problems and proposed solutions can be found in (Lifschitz 1990). Additionally, formal theories of action and change often make the assumption that (Gelfond & Lifschitz 1993) environmental changes only occur as a result of some agent-initiated action. This assumption is unduly restrictive and renders such systems incapable of dealing with complex environments where state transitions may occur independent of agent-initiated action. In most complex environments, the world continues to evolve between agent actions.

These problems combined with the need to generate real time behavior in complex and dynamic environments renders deliberative reasoning about the effects of low-level action too expensive and thus impractical. Recognition of this fact in AI and Robotics has resulted in various proposals for the representation and use of compiled plans and behaviors (Rosenschein 1985; Brooks 1986; Nilsson 1994; Arkin 1990; Agre & Chapman 1987). The basic idea is that rather than divide overall process of acting in the world into functional components such as planning, execution, and perception, one could instead divide the process into task specific pre-compiled plans for various *behaviors*. Examples of such behaviors may involve reaching, grasping, walking to a destination, stacking blocks, etc. The basic insight here is that the necessity to act fast in an uncertain and dynamic world requires reactive planning agents (biological or robotic) to develop representations that can tightly couple action, execution monitoring, error-correction and failure recovery.

3.1.3 Computational models of cognition

Schema based computational models of higher level cognition have been investigated by Arbib and his students (Arbib 1992) for several years. Our design of x-schemas is inspired by and bears a strong resemblance to the work of Arbib's group. In fact, the example used to illustrate the computational issues involved in our design is taken from (Arbib 1994). However, our focus on language and simulative inference renders our model quite different from those proposed by Arbib's group. (Drescher 1991) argues for the use of schema theory in learning behaviors and develops a computational model of schemas. As we will see in Chapter 6, our compositional theory of x-schemas can be viewed as a more detailed specification of the types of x-schemas proposed by Drescher.

In Natural Language Understanding, the work of Schank and his students was the first serious computational exploration of schema-based interpretation. Early conceptual dependency theory (Rieger1975; Schank & Abelson 1977) used the idea of schemas for inference in discourse interpretation. We share many of the basic premises and guiding intuitions inherent in early work in conceptual dependency theory, including the need for a rich representation of actions, goals and resources (Wilensky 1983). However, our use of a fine-grained, executable model for actions differs somewhat from these proposals. These issues are further discussed in Section 3.8.

3.1.4 Psychology and Cognitive Linguistics

In psychology *schemas* have a long history. The most famous proponent of schema theory was Piaget who argued that much of abstract thought and behavior was grounded in sensory-motor abstractions. Other attempts to use schema theory have included Neisser (Neisser 1971) who investigates schemas in the context of the action/perception cycle, (Norman and Shalice 1980) who have used schemas to describe intentional behavior as opposed to automatic reflex behavior.

Recent work in Cognitive Semantics (Lakoff & Johnson 1980; Lakoff 1994; Talmy 1985; Talmy 1987) suggests that the structure of abstract actions (such as states, causes, purposes, means, etc.) are characterized cognitively by metaphoric mappings to embodied concepts from the domains of force, motion, and space. Such observations have resulted in proposals that abstract reason is grounded in recurring patterns of experience called *Image*

Schemas (Lakoff 1987). Specifically, Mark Johnson argues (Johnson 1987) that *schemas* need to be dynamic patterns rather than fixed or static images, and that schemas are "fluid" and "malleable" structures that are capable of imaginative reasoning rather than frame-like templates with slots needing to be filled in. Thus far, the work in Cognitive Semantics has lacked computational models for such theories, and consequently such ideas cannot currently be used in natural language understanding or problem solving systems. Part of the work outlined in this thesis, including our x-schema based representation, tries to provide a dynamic system framework to investigate issues of cognitive semantics. As we continue, we will make explicit the connection to existing theories in this area.

3.2 Computational Requirements for X-schemas

To develop formal requirements for the representation of x-schemas, let us consider the problem of implementing a high-level controller for the *walking* behavior.[1] Figure 3.1 shows the high-level logic to implement the behavior. While Figure 3.1 is obviously simplified, it will serve to illustrate the central computational requirements.

The basic walk schema remains activated as long as the desired destination is not reached. Once activated, the walk schema controls the execution of the walk behavior. The behavior is decomposable to a temporal arrangement of several sub-actions. Some of these sub-actions may be primitive (corresponding to motor synergies in biological control). Others may be further refined (shown as shaded nodes in Figure 3.1). Figure 3.1 shows the first level refinement of the walk behavior. Note that the behavior can be further decomposed into the constituent synergies that make up a step (the various *swing* and *stance* phases and their temporal arrangement).

The actual execution trajectory is conditional on the results of perceptual tests and/or world inputs. For example, the basic cycle of *ready* $\Rightarrow$ *test footing* $\Rightarrow$ *take step* of the WALK schema may be interrupted if the result of *test footing* fails. Also, if the visual information suggests stable ground ahead, the agent may completely bypass the test. The basic cycle repeats (at a rate specified by the rate input parameter) until the agent receives information (asynchronously from an unscheduled perceptual process) of being at the destination. At this point the walking process is completed.

[1]The high-level controller logic is taken from (Arbib 1994).

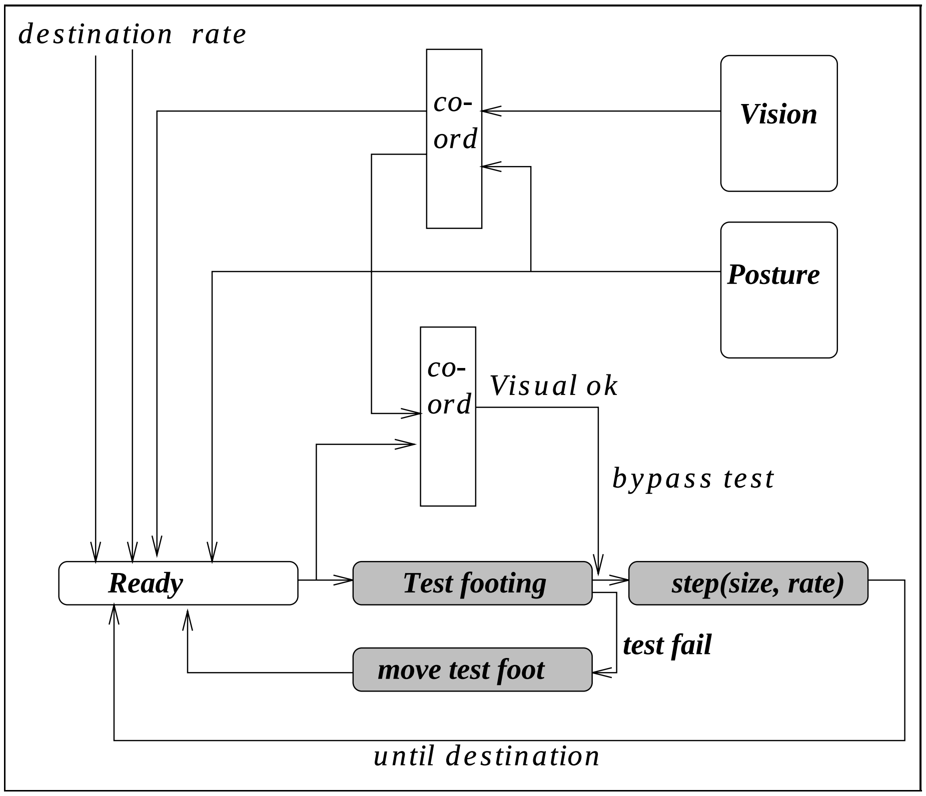


Figure 3.1: High-Level Logic For a Walk Controller

From the discussion above, we conclude that x-schemas must satisfy a number of computational requirements.

- **Flexibility**: Schemas operate in a dynamic environment and have to adapt to different world situations. This requires the ability to **dynamically bind** to different world objects at run time, as well as to be able to execute with different **parameter settings**. For example, in the walk schema, the schema may dynamically bind to different landmarks (as in walk to the store), as well as specific locations or directions (walk to the center of the room, or walk along corridor). Schema execution may be parameterized (modulated by different parameter settings). In Figure 3.1, the x-schema for walk is shown parameterized for a speed parameter. Other relevant parameterizations include force (consider the difference between shoving and pushing something), or direction (push versus pull). Note that the need for dynamic binding and run-time parameterization makes it impossible to view x-schemas merely as pre-compiled programs.
- **Control**: Basic control behaviors should be modeled. This involves the necessity to model sequences of actions, alternative actions, and decomposable actions. Should be capable of executing multiple concurrent and coordinated actions. For example, the WALK controller in Figure 3.1 should be capable of coordinating postural and visual information before deciding to take a step. Second, **enable** and **execute** are distinct. Thus an enabled x-schema may or may not execute. Schema execution proceeds with distributed control. The activation of different schemas is *asynchronous* and the flow of activation within a single schema is locally controlled. This is required to maintain the responsiveness required.
- **Responsiveness**: An x-schema must be highly **responsive** to both internal (perceptually generated) or external events (world generated). For example, the WALK x-schema in Figure 3.1 may forgo testing the footing if perceptual input indicates a stable ground. In general, the x-schema should be capable of event-based interruption.
- **Resources**: Schemas must be able to model the **production** and **consumption** of resources. This includes *measure fluents* such as energy and force. In addition, pre-conditions and post-conditions should be represented.

3.3 Representation Of X-Schemas

The computational model of x-schemas is based on an extension to Petri nets (Murata 1989).

Definition 2 Basic X-schema

An **x-schema** consists of *places*(P) and *Transitions* (T) connected by weighted directed arcs $\mathcal{A}$ ($\mathcal{A} \in (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$). Each arc $a_{ij} \in \mathcal{A}$ has weight w_{ij} ($w_{ij} \in \mathcal{N}$). *Input Arcs* $*\mathcal{T}$ ($*\mathcal{T} \in (\mathcal{P} \times \mathcal{T})$) connect Input Places to Transitions. Output Arcs $\mathcal{T}*$ ($\mathcal{T}* \in (\mathcal{T} \times \mathcal{P})$) connect Transitions to Output Places. Places are typed as *enable* places $\mathcal{E}$, *inhibitory* places $\mathcal{I}$, or *resource* places $\mathcal{R}$.

The underlying model of an x-schema is a weighted, bipartite graph that consists of *Places* and *Transitions. Input Arcs* connect Places to Transitions, *Output Arcs* connect Transitions to Places. The bipartite nature of the x-schema naturally captures the well known **state/event** distinction that pervades linguistic analysis (e.g. (Langacker 1987)). In x-schemas, both states and events are distributed over the entire net. A specific state of the schema corresponds to a *marking*. Formally a marking is a function that assigns either 0 or a positive integer p ($p \in \mathcal{N}, p \leq K$) or a to each place. The state of the x-schema is thus described by an M *-vector*, where the i_{th} element of the vector is $M(i)$, denoting the **number** and **type** of tokens in the i_{th} Place. Clearly, the M -vector ranges over the number of Places in the net.

Definition 3 Execution Semantics

X-schemas have a well specified real-time execution semantics where the **next state** function is specified by the **firing rule**. In order to simulate the dynamic behavior of a system, a marking of the x-schema is changed according to the following **firing rule.**[2]

1. A transition t is said to be enabled if **no** *inhibitory* input place $i \in \mathcal{I}$ of t is **marked and** if each enable place $e \in \mathcal{E}$ of t is **marked** and all other input places $p \in \mathcal{R}$ have at least w_{pt} tokens, where w_{pt} is the weight of the arc from p to t.

[2]All the results for Chapters 3- 6 are from a graphical simulator implemented in C with a *Tcl/Tk* graphical interface. Code for the simulator can be obtained from "snarayan@cs.berkeley.edu" or pointing your browser to "http://www.icsi/berkeley.edu/˜snarayan". To install the simulator you need to have the Tcl/Tk libraries.

2. An instantaneous transition *fires* as soon as it is enabled. A timed transition fires after a fixed delay or at an exponentially distributed rate.
3. The firing of an enabled transition t, removes w_{pt} tokens from each non-inhibitory, non-enabled input place p and places w_{tp} tokens in **each** output place of t. Thus the **next-state** firing rule ($\delta(s_i, t)$) takes the x-schema from a state s_i to the next state s_{i+1} .

The state of an x-schema is defined by its marking. The firing rule produces a change in the state of an x-schema by taking it from one marking to the next. Given an x-schema and a marking, we can execute the x-schema by *successive* transition firings. This can continue as long as there is at least one enabled transition that can fire. The x-schema firing rule semantics allows enabled transitions to fire in a completely distributed manner without any global clocks or central controllers.[3] Execution halts at the state where there is no enabled transition. This naturally allows us to extend the earlier definition to define an **extended next-state function** for x-schemas.

Definition 4 Extended next-state function

The extended next-state function is defined for a state s_i and a sequence of transitions $\rho \in T^*$ as

$$\delta(s_i, t_j\rho) = \delta(\delta(s_i, t_j), \rho)$$
$$\delta(s_i, \lambda) = s_i$$

λ is the *null* transition.

In all further discussion, we will use the extended next-state function to simulate x-schema evolutions. As an example, consider the actual simulation starting from the initial configuration in Figure 3.2 to the final *deadlocked* (no transition is enabled) condition. The basic x-schema shown in Figure 3.2 shows the application of the next-state function to simulate x-schema evolution.

Figure 3.2 shows a very simple x-schema with a single event/action A with duration 1 time unit ($D = 1$), three Input Places, a resource place R , an enable place, E , and

[3]However our sequential simulation adjusts step size to be able to fire multiple enabled transitions in a single step.

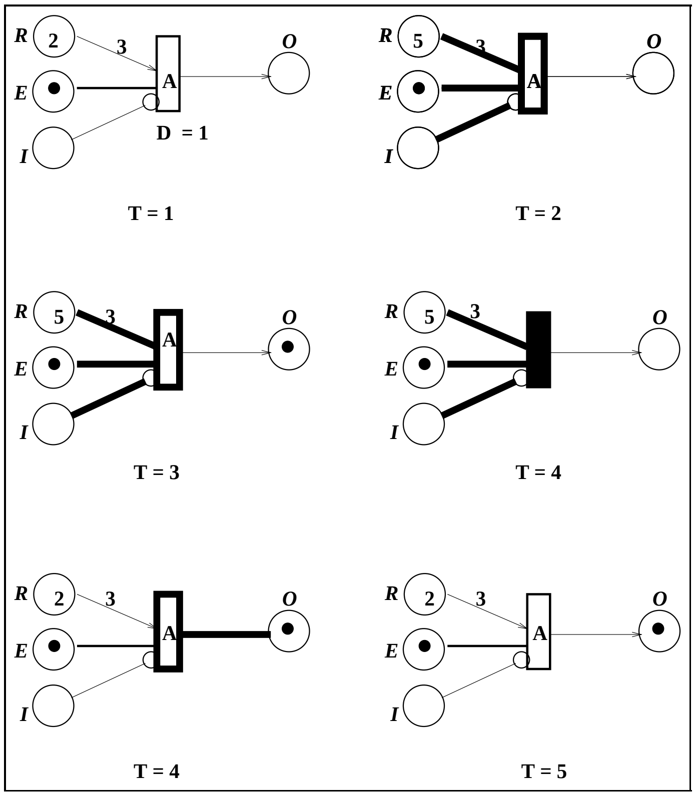


Figure 3.2: Executing a basic x-schema. The bold lines on arcs indicate that token flow is possible on that specific arc. The amount flowing will depend on the weight of the arc. Bold lines on transitions and input arcs (arcs entering transitions) imply that the specific transition is enabled, while bold lines on transitions and outgoing arcs imply token transfer to the output places. The darkened transitions at $T = 4$ corresponds to the firing of A.

an inhibitory place, I. There is single output place, O. The weight of resource input arc is 3, representing the requirement that at least 3 units of the resource be present before the action is enabled.

In the graphical representation of x-schemas, Places are drawn as circles, and Transitions as boxes. If a Place p is marked with the integer k, we say that the "*Place p is marked with k tokens.*" Graphically, this is illustrated by the presence of a black dot with the integer k alongside the relevant Place p. Whenever a specific token type is required by a transition from a specific place, or when a specific token type is present at the input place to a transition, it will be indicated by the appropriate value.

As shown in Figure 3.2, the initial situation is one where there is no inhibitory input ($I = 0$), and the enable input is marked ($E = 1$) and the resource input ($R = 2$). This is insufficient for the action A, so the corresponding transition is not enabled. This is the situation at time $T = 1$. At this point (asynchronously), by some magic three more units of the required resource are produced. This is sufficient to enable the transition at time $T = 2$. The duration of 1 time unit keeps the transition enabled at time $T = 3$ until it fires at time $T = 4$. In the current implementation, firing takes place instantaneously (but one could have enable-to-fire and firing durations as well). Executing the next state function using the firing rule above takes the x-schema to a new state shown in the bottom row of Figure 3.2. The new state at time $T = 4$ shows the depletion of 3 units of resource R, in addition to marking the output O. At this point, the resource R again drops to below the required level, and the transition returns to the initial state of being inactive or unenabled.

Example 2 shows the actual simulation output that results from applying the extended next-state function.

Example 2 Simulating the basic x-schema in Figure 3.2 The basic x-schema shown in Figure 3.2 has 4 places and one transition. The places are labeled R for Resource, E for Enable, I for Inhibit and O for Output. In the x-schema simulation display, each place is mapped to a column the table. This mapping is usually fixed at the beginning of the simulation and is the "x-schema place to column mapping" below. In the simulation of Figure 3.2, column 1 in the table below corresponds to the Resource R, column 2 corresponds to the Enable place E, etc. Each row is labeled by a time and a marking. A marking is a state vector which shows token distribution over the x-schema. For instance the marking M_0 below corresponds to the vector $[2100]^T$, signifying a state where there are 2 units of resource R, the enable input place to the transition E is marked (has a token), and there are no tokens in the inhibitory or output places. Similarly, the marking M_1 corresponds to the vector $[5100]^T$, where the number of resource units has increased to 5 while the rest of the vector is similar to M_0. Once this happens, note that the transition labeled *Event* is Enabled. The simulation output for these and future time steps is shown below.

```
x-schema place to column mapping:
     1 = Resource
     2 = Enable
     3 = Inhibit
     4 = Output
States:
Time Marking    1   2   3   4  | Transitions to...
================================= | -----------------
T=1  M0     |   2   1   0   0  | ( )
T=2  M1     |   5   1   0   0  | ( M3 ) (Label: Event) (Status: Enabled)
T=3  M2     |   5   1   0   0  | ( M3 ) (Label: Event) (Status: Enabled)
T=4  M3     |   2   1   0   1  | ( )
T=5  DEADLOCKED========================
```

■

3.3.1 Extensions to the basic x-schema

The basic x-schema shown in Figure 3.2 can be extended in the following ways to be able to model hierarchical action sets with variables and parameters:

First, tokens carry information (i.e. they are individuated and typed) and transitions are augmented with predicates and actions. This modification allows for predicates to select tokens from input places based on the token type, as well as relate the type of the tokens produced by the firing to the types of tokens removed from the input.

Figure 3.3 shows the effect of parameterizing an x-schema. The networks in Figure 3.3 a) and Figure 3.3 b) show the initial state of the system for an x-schema (without token types). Places A_a and B_b are preconditions (enable places) to the transition T_a and R_a is the output place of the transition(Figure 3.3 a)). Similarly Places A_b and B_a are

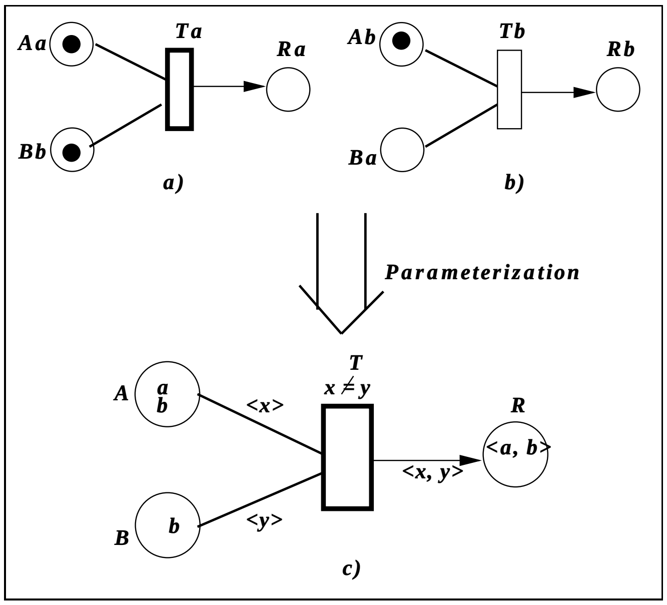


Figure 3.3: X-schema tokens are typed to represent run-time parameters and bindings. This allows a large net to be collapsed into a much smaller net that allows run-time binding.

preconditions to the transition T_b and R_b is the output place of the transition(Figure 3.3 b)). Transition T_a is enabled (in Figure 3.3 a)) since it has tokens in all its input places (no inhibitory place is shown to keep the exposition simple). Firing the transition (the event occurs) implies that the output R_a would be augmented with a token. On the other hand, in Figure 3.3 b) the transition T_b is not enabled, since there is no token in the place entitled B_a . Were it to be enabled then firing it would result in a token as the R_b place. Collapsing these nets into a typed net results in the typed network (called as High Level Net or Colored Net in the Petri net literature) shown in Figure 3.3 c). Here we have tokens that can be of type a , b , or $< a, b >$. The place entitled A has tokens that can be of type a or b as does the place B . The place R has tuples of the form $< x, y >$ (where x, and y are variables). The initial state from Figure 3.3 a) and b) are represented as the token types (shown by their type value) at each of the input places. Firing the transition in this state results in the tuple $< a, b >$ to be added to the place R augmenting the current marking. This is the situation shown in Figure 3.3 c).

In Section 3.10 (Lemma 1), we formally show that High Level nets (such as the one in Figure 3.3 c) do not add any representational power (as long as the number of types are finite) by using a well known result from Petri net theory that shows that any net with a finite number of types can always be *unfolded* into an ordinary net by adding special places for type combinations. Crucially, the extension to individual tokens preserves the *liveness* (if a transition could be enabled in the typed (ordinary) net it would in the equivalent ordinary (typed) network) and *reachability* (any state that can be reached in the typed (ordinary) network can also be reached in the equivalent ordinary (typed) network) properties of the underlying Petri net of the x-schema. So in effect, using token types (or run time parameters) greatly reduces the size of the net required while preserving important dynamic and structural properties of our representation.

The second x-schema extension pertains to the different types of transitions and firing rules allowed. Figure 3.4 shows the three types of x-schema transitions, namely *stochastic* transitions, *durative* transitions, and *instantaneous* transitions.

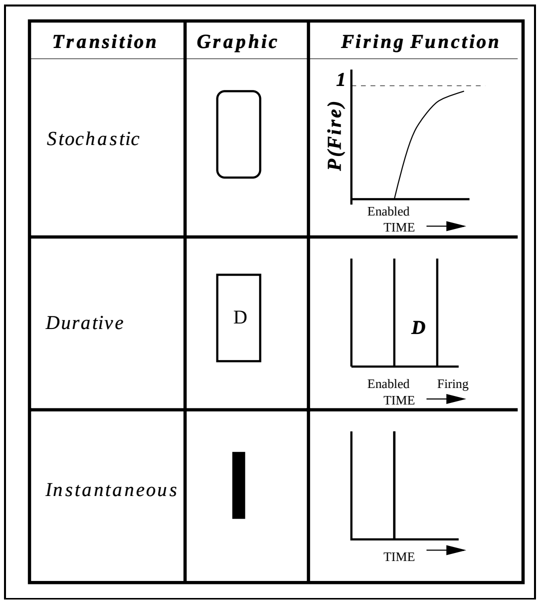


Figure 3.4: Primitive x-schema transition types.

Stochastic Transitions are associated with an exponentially distributed random variable that expresses the delay between enabling and firing. Due to the memoryless feature of the exponential transition, x-schemas with stochastic transitions are isomorphic to semi-Markov processes (by analogy with Petri nets shown in (Molloy 1982)). Durative Transitions fire with a fixed delay (> 0) after being *enabled*. The actual delay is an integer which varies depending on the nature of the transition. Instantaneous transitions fire without any delay after being enabled ($D = 0$). In addition, we will see in Figure 3.5, how we can use these primitive transition types to implement a *hierarchical* transitions that correspond to the activation and execution of a **subnet**.

Third, *input arcs* onto the individual transitions are typed with the following types,

1. *Enable* arcs correspond to *preconditons* of an action. Such arcs have a weight (also called multiplicity) of 1. Executing an action (firing a transition) leaves the marking of an enable arc unchanged. Firings along enable arcs only occur if the output of the transition is unmarked (such firings are called *contact-free* in the Petri net literature (Reisig 1985).)

2. *Consume arcs* correspond to resources that are consumed during execution. The amount consumed corresponds to the multiplicity of the corresponding arc. *Consume* arcs are standard Petri net arcs.

3. With the introduction of *inhibitor* arcs, a transition t is enabled when all of its non-inhibitory inputs I_s are marked with $\#(I_s) \geq w_{st}$ and places with inhibitory arcs onto T are empty. In general, introduction of inhibitory arcs can potentially increase the modeling power of x-schemas to be that of Turing machines with a corresponding decrease in decision power; however, an x-schema with inhibitory arcs can always be transformed into an equivalent ordinary net without inhibitor arcs. This is possible because the underlying Petri net of an x-schema is **bounded**. The reader is referred to Section 3.10 for the proof.

The following theorem establishes the formal equivalence between x-schemas and Petri nets.

Theorem 1 . *An x-schema without stochastic transitions can be converted to a bounded ordinary Petri Net. With stochastic transitions, an x-schema is formally equivalent to a Generalized Stochastic Petri Net (GSPN). The reachability graph of a marked x-schema is isomorphic to a semi-markov process.*

Section 3.10 details the proof of the theorem. The theorem is important to our effort since Petri nets are one of the most popular and well studied computational formalisms for specifying, modeling, and analyzing highly distributed and concurrent systems (Murata 1989). Algorithms and analysis techniques from that literature can directly be brought in to our work. For instance, to model hierarchical action sets with variables and parameters, we extend the basic model to allow tokens to carry variable binding information (i.e. they are individuated and typed). The expressive power remains unchanged since it is well known in Petri net theory that a net with a finite set of colors can be *unfolded* into one with a single color. Furthermore, the bounded nature of the underlying network ensures that the reachability graph of the marked x-schema is finite.

3.4 Modeling with X-schemas

We start by showing how some basic primitives that are required for executing motor programs can be encoded as schematized x-schema structures. We will then describe the WALK x-schema in greater detail. As we outlined in the last section, x-schemas should be capable of modeling basic control primitives such as sequence, concurrency, hierarchy, and conditional execution. Figure 3.5 shows some of these primitives as x-schema implementations. Furthermore, we noted earlier that to be able to control actions in a dynamic and uncertain environment, x-schemas must be capable of goal-based enabling and be responsive to changing resource levels and time constraints. Figure 3.6 describes some of these aspects of x-schemas in greater detail.

3.4.1 Modeling control primitives

Figure 3.5 shows the basic structures of an x-schema. Note that enable arcs are shown as bold and undirected. Inhibitory arcs are shown as undirected arcs with a small circle attached to the end of the arc, specifying that the input place is an inhibitory place to the transition. Enabled transitions are shown with a bold outline.

Primitive actions (synergies, atomic actions, etc.) are modeled as x-schema transitions. A primitive action is enabled when all its preconditions are satisfied , i.e. all *enable* input places (labeled $\mathcal{E}$ in Figure 3.5) are marked with the right type of tokens, the required type and amount of resources are present, i.e. all *resource* input places (labeled $\mathcal{R}$ in Figure 3.5) are marked with the right type and number of tokens, and **no** *inhibitory* input place (labeled $\mathcal{I}$ in Figure 3.5) to the transition is marked. Once the primitive action is

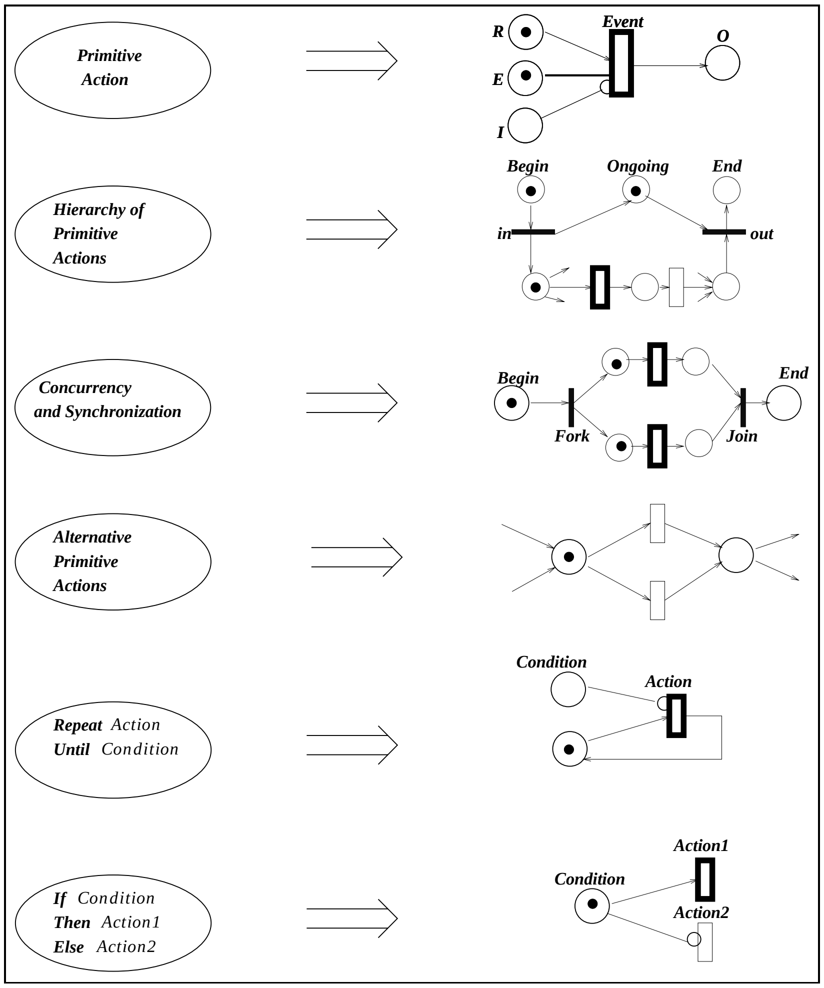


Figure 3.5: Modeling various control primitives

enabled, it can fire leading to new marking where all output places and enable places are marked as a result of the firing of the transition $t \in \mathcal{T}$.

$$M_{i+1} = M_i - \sum_k w_{kt}(R_k) + \sum_j w_{tj}(O_j) \tag{3.1}$$

Notice that the firing of transition t at the *ith* time step changes the marking vector (M) from its old value of M_i to its new value M_{i+1} (the additions and subtractions above are vector operations). For instance, the marking vector in Figure 3.2 (ref. Example 2) showed the effect of firing the transition *Event* with marking vector $M_2 = [5100]^T$, which resulted in the new vector $M_3 = [2101]^T$ in accordance with the rule shown in Equation 3.1.

The enable places remain unchanged from the old marking, while the appropriate amount of resource tokens are consumed by the firing from all the resource input places, and the appropriate amount of output tokens are deposited at all the output places of the transition. As a side note, the mechanism generalizes over the STRIPS action formalism (Fikes & Nilsson 1971), in allowing *measure fluents* such as resources to be modeled. Of course, a central distinction between x-schemas and STRIPS plans are that x-schemas are *dynamic systems* that execute, and while one could run goal regression algorithms and other types of theorem provers over x-schemas using them as declarative structural entities, it is the dynamic properties of x-schemas that concern us in this work.

To model hierarchical actions, we use a pair of instantaneous transitions that correspond to the beginning and end of the subnet. This is the situation shown in the second example of Figure 3.5. Here the presence of a token at the begin input place (signifying the beginning of subnet invocation) gets split into two, one token going to the *ongoing* place signifying that the subnet in question is executing (allowing for time out procedures if needed), and the other goes to the first action/event of the subnet. When the subnet completes execution, the finish transition is able to fire (assuming that the token still exists at the *ongoing* place). Note that this is an example of token splitting(forking) and token merging (join) behavior shown as the third control behavior outlined in Figure 3.5.

Figure 3.5 also shows the representation of alternative actions available to the agent at a given state. The simple template shows that the marking vector changes in an identical fashion if either transition is fired. Of course, there may be other side-effects which may be unique to which action is performed (such as different amount of resources consumed or different world conditions becoming true as a result of choosing a specific alternative), which can be easily modeled as well. In addition, the x-schema model presented here is able to model both **conflicts** and **decisions** (both cases where taking a specific action (firing a specific transition) may lead to widely differing next states).

The last two control behaviors pertain to the standard constructions of *repeating* a behavior *until* a specific condition is reached, and *conditional branching*. In the first case, a transition continues to be enabled until the condition is true, in which case, it is **inhibited** from firing. In the second case, the presence or absence of a condition **activates** or **inhibits** the firing of the appropriate transition.

3.4.2 Modeling resources and goals

Goal directed motion control also requires the modeling and monitoring of resource levels (such as force levels, energy levels) in real time, well as being able to control a specific distribution of action in time. To monitor and control the execution of motor programs and their effects in a dynamic environment, x-schemas need to control both **periodic** and **aperiodic** actions with real **durations** (to monitor, timeout, initiate error-recovery procedures). Our model for both durative and periodic actions is shown in Figure 3.6. The case of the durative action is very simply modeled, since x-schemas are *timed* automata (or more precisely timed *nets*). Hence a transition in our model has a duration which can be a constant ($D \geq 0$) or stochastic (sampled from an exponential distribution). In the bottom right of Figure 3.6 are shown the templates corresponding to an inherently periodic action (like *breathe* or *tap*) and an inherently aperiodic action (like *shove* or *fall*). In the periodic case, our model simply activates the resources and enabling conditions for execution as a result of completing the action (hence the directed link that transfers tokens from the *finish* of an action to the enabling or resource inputs to the *start* of the corresponding action. The non-periodic case could be handled either through explicit **inhibition**, where the *finish* transition results in the marking of an **inhibitory** input place to the *start* transition, or as is shown in the bottom right of Figure 3.6, the finish transition may disable the x-schema from further execution by actively **consuming** a needed resource or pre-condition required to *start* execution again.

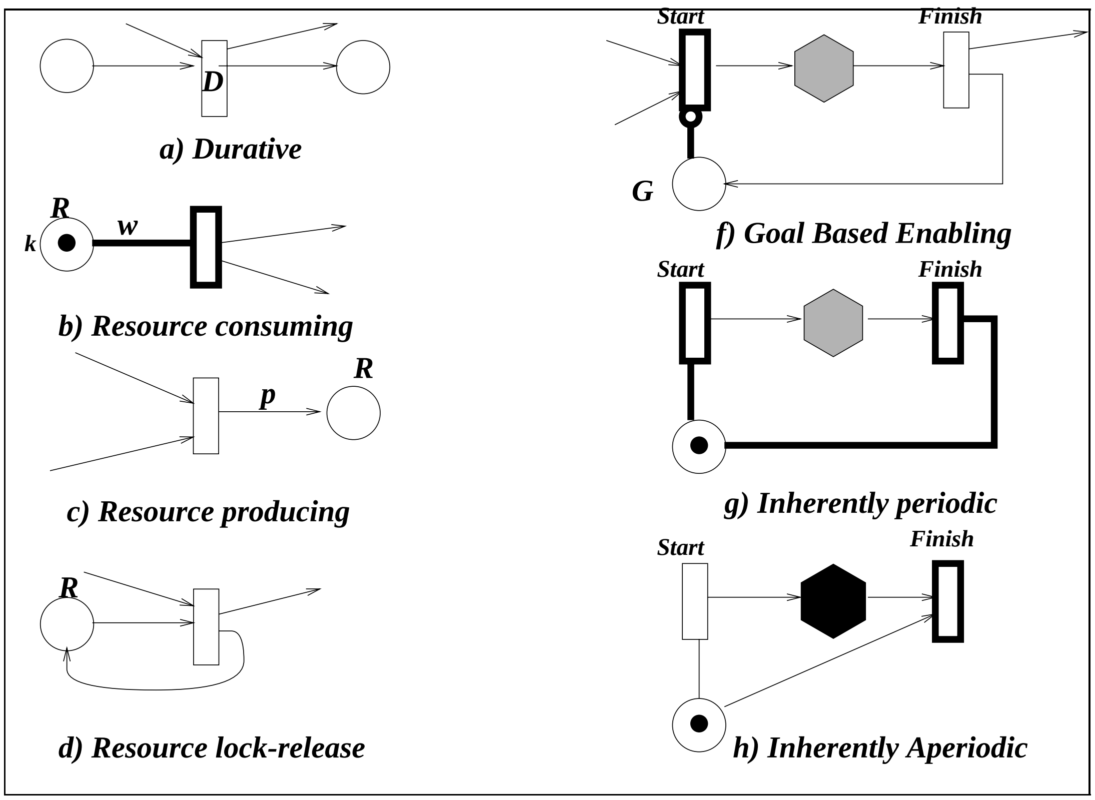


Figure 3.6: Modeling Motion Control Parameters. Here the hexagonal nodes on the right of the figure correspond to sub-schemas (not shown). Bold transitions and input arcs imply that the corresponding transition is enabled. Bold output arcs imply token transfer to the output places of a firing transition (shown in bold).

One interesting thing to note from the discussion above is that from purely sensory-motor arguments we made the case for two different modes of disabling an action, one of explicit **inhibition**, and the other of **consumption** of a needed resource. If we abstract this duality to reasoning in general and in particular to the semantics of **negation**, we note that negation would be *constructive* in an x-schema based reasoning system, where it is either possible to model negation as the explicit **disabling** of the negated entity or as the **consumption** of a needed condition or resource. The implications to reasoning about negation in general are not developed further in the research reported here, but is part of our future work.

The issue of dynamic resource modeling is central for any autonomous agent to function in a dynamic, uncertain environment. Requirements include being able to check and monitor **conditions** and **resource** usage. These include monitoring resource consumption (energy level), as well as respecting mutual exclusion or temporary locking constraints (can't hold two blocks at the same time)), and *enabling* and *disabling* conditions (can't walk if the ground is slippery). Agents should be capable of **goal** based enabling and should be able to monitor and remain active until achievement of the goal. Figure 3.6 shows the x-schema templates for resource consumption, production and locking. Note that amounts of resources required for the action involved is modeled as the weight w ($w \in \mathcal{N}$) on the resource input arc, and in the production case, the output arc weight p ($p \in \mathcal{N}$) represents the amount of resource produced as a result of the action. The top right template in Figure 3.6 shows the model of goal-based enabling. The key thing to note here is that, the relevant x-schema **continues** to be enabled as long as the goal is **not** satisfied (of course if all other pre-conditions are satisfied and resources present). Once the x-schema finishes execution, it marks the goal being satisfied, **disabling** the x-schema from further execution. In future discussion, we refer to the **abstracted** primitives in Figure 3.5 and Figure 3.6 (control templates, duration, periodicity, resources, goals and conditions) as the **process primitives**. In examples on the right hand side of Figure 3.6, we see the use of the *start* and *finish* places signifying control status of the underlying x-schema. The *start* transition signifies that the x-schema in question is active and has execute preparatory steps to a state of *readiness*. The *finish* transition signifies that the completion of x-schema execution leading to the state where the x-schema is *done* . These transitions are part of an underlying motor-control regularity that can be found in all motor programs. The implications of this statement are fully explored in Chapter 5 when we use this regularity in modeling the semantics of aspect.

From the cases above and the discussion to follow, the reader should crucially get the notion of **activation** and **inhibition** as being the central x-schema modeling parameters. Together with another primitive which we will introduce in Chapter 5 (for the curious, the other primitive corresponds to **gating** or execution trajectory modification), we will be in a position to formally define how to construct compositional domain theories using x-schemas in a way that preserves the active and dynamic properties of x-schemas. Such a compositional theory will form the backbone of a simulative inference mechanism, where real-time inferences are made through x-schema executions. But that will have to wait till Chapter 6.

3.5 X-schemas and Verb Semantics

Recall that our proposal is to ground the semantics of intentional action verbs in the x-schema representation. To this end, in our model such verbs code for specific features, parameters, and other **schematized** recurring x-schema patterns. For the moment, let us consider some examples of implemented x-schemas which we will use for language understanding.

Just as an example of the kinds of distinctions made by motion words consider the following words that all pertain to the same x-schema, **walk**, but supply different monitoring, control, intentional and motivational parameters.

walk, walk (slowly/quickly), walk (steady/unsteady), walk (toward/away),
step, measured step, cautious steps, (giant, small, tiny) step,
sidestep, slip, slide, stagger, swagger, kick, dragging ones feet,
dig in, digging your heels, trample, stamp, hoof, perambulate,
ramble, stroll, hike, amble, tread (softly, gently), trudge,
ambulate, foot (it), trundle, pace, traipse, gambol, troop,
circumambulate, , promenade, stroll, tramp,
lumber, plod, slog, stump, trod, leg, race, run, waltz,
stumble, fall, fall in, fall into, limp, hop, bumble, skip,
stride (length), in stride, (giant, small, measured) stride

If one looks at the variations of walk related words and expressions in English, one realizes that words code for fairly detailed dynamics of walk. Parameters include the destination, rate of motion, cautiousness and monitoring, step size, forcefulness of the stance or swing phases, direction of step or walk, purposefulness (or lack) of the walk, energy level and motivatedness, intention, control or loss of control (fall), resistance provided by the environment (such slipperiness of the ground or unexpected encounter of a bump). Many of these primitives are the same as the process control parameters in Figure 3.5 and Figure 3.6. Many of the differences in the words above can therefore be captured in an x-schema based representation of a parameterized walk controller.

3.5.1 Language and x-schema execution parameters

Conceptually, the interface to x-schemas can be considered as a set of feature-value pairs which are labeled the **f–struct** in Figure 3.7. The interaction between the f–struct and x-schemas is **bi-directional**. Specific feature-value settings can can **initiate** and **modify** x-schema executions. Essentially all interactions (including linguistic) to x-schemas are mediated through the f–struct interface by **setting**, **getting** and **modifying** the various feature-value pairs. This results in triggering, inhibiting or modifying x-schema executions in some manner, which in turn may result in affecting the Embodied f–struct. X-schemas can be used for executing actions or interpreting commands in a dynamic and changing world. They can also be used for real-time simulative inference. Here we will address the command-obeying function of x-schemas. The architecture of our x-schema-based inference system is detailed in Chapter 6.

While depicting the interface as a single structure makes it easy to explain, we note that such an abstraction is useful only for pedagogical reasons. In the implementation, the f–struct is distributed over the entire system. As we will see in detail in Chapter 6 the agent state is distributed over all the x-schemas in the system, and can be used to trigger or modify execution in a completely distributed, asynchronous manner.

Linguistic devices may only specify some of the parameters for an x-schema. The subset of motor control parameters coded for by linguistic features is referred to as the **linking features** (signifying the language link). Of course, this is a small part of the required set. X-schemas always execute under specific world conditions. For instance, walking on unstable ground is quite different from walking on firm terrain. The information and world condition parameters that may impact x-schema executions is referred to as the **world state**. The third set of features that modify x-schema executions are intentional and motivational features. In our work, we found goals, energy levels, and affective states (obviously correlated to energy and goals) important execution parameters for x-schemas. Walking when tired is obviously more labored and different than walking with high energy.[4] We will jointly refer to such internal state features as **internal state**. So in effect the f–struct consists of the three kinds of features described above. Conceptually, such a distinction is important since it allows for words and lexical items to be coded using a small set of features under the assumption that the context-sensitive nature of the underlying x-schema representation will take care of the pragmatic issues in interpretation. This can potentially help both in lexical acquisition and lexicon extension (ref. Section 3.6).

Figure 3.7 shows an example of encoding the high level walk controller in Figure 3.1 as an x-schema. Circles in the figure correspond to the various states of the x-schema, hexagons correspond to hierarchical actions as described in Figure 3.5. Rectangles are single actions/events represented as transitions. We assume the *step* to be of duration specified by the rate parameter, and step size to be covering the distance specified by the size parameter. For the simulation shown in Figure 3.8, we set the step duration to be 5 time units and other conditions and tests to be of 1 time unit (including actions such as $test\_footing$ and $move\_test\_foot$).

Figure 3.7 shows the situation where the WALK x-schema is *enabled*. If enabled, the presence of the colored (typed) token *ok* signifies that vision and postural tests return an *ok* status. This fires the transition that coordinates visual and postural information and results in the $vision\_ok$ place being marked with a token. In parallel, firing the transition labeled $co-ord$ on the top of Figure 3.7 places a token at the *ready* place of the WALK x-schema, signifying that the agent is ready to walk, and that the ground ahead is fine. This is the situation shown in Figure 3.7. At this point, the presence of tokens at both input places to the transition enabled $bypass\_ok$ (center of Figure 3.7, and an instantaneous transition) allows the WALK x-schema to bypass the $test\_footing$ test and enable the *step* transition. After a step this cycle is repeated. Note, any time the places entitled $Vision$ or $Posture$ return a not_ok status, the $vision\_ok$ place will no more be enabled, and the $bypass\_ok$

[4]And indeed some words like sluggish or tired may code specifically for these parameters.

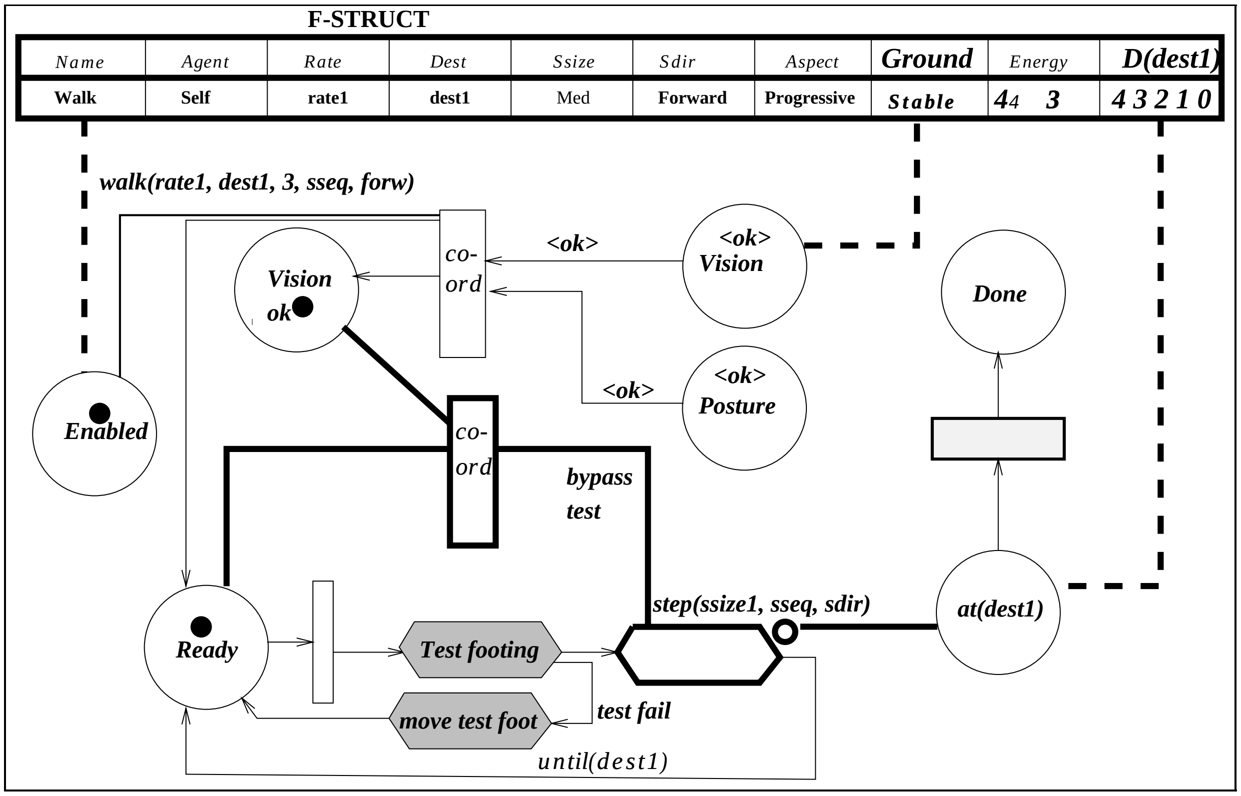


Figure 3.7: The WALK x-schema shown with the motor control parameters, world state parameters and motivational and intentional parameters jointly held by the **f–struct**. Bi-directional interactions (a few shown as dotted lines) **set**, **get** and **modify** f–struct values and x-schema executions. For example, the distance to goal is shown decreasing as a result of x-schema executions, while the world condition pertaining to stability of ground modifies x-schema executions (to test or not to test footing) as does explicit specification of motor control parameters (such as rate, destination, and step size).

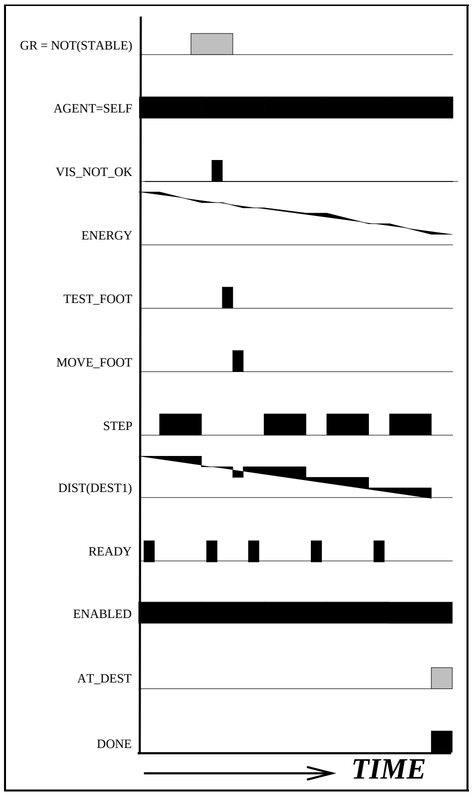


Figure 3.8: Executing the WALK x-schema. The walk x-schema is a dynamic system which is able to respond to world state and linguistic input. Shown in the figure, is the Schema executing with a specific energy level and a specific World state. As long as the visual test is *ok* , the test foot branch is not taken. When in the third cycle (x-axis corresponds to the number of cycles through the **until** loop), test footing is not used. When the ground is not stable ahead, visual test returns $\neg ok$ (shown as dark rectangles) and the test foot branch is taken. Note the dynamically depleting energy resource and the dynamically changing *fluent* representing the distance to goal.

transition will no longer fire, and the *test_footing* test will have to be undergone. Note also that the responsiveness of the system as shown in Figure 3.7 is limited to the granularity of a *step*, since if the vision or postural status changes within a step, it will not affect performance till the next step. For tighter responsiveness, we need more circuitry. For instance, being able to *stabilize* upon encountering an unanticipated *bump* while stepping. Our design of x-schemas allows us to model this degree of responsiveness and failure recovery as well, but we will have to wait till Chapter 6 to see exactly how this is done.

Figure 3.8 shows the changing **marking** as the x-schema executes. Note specifically that states (such as the Agent of the walk being bound to *self*) persist indefinitely. Also world conditions may change during execution, modifying the trajectory appropriately. In Figure 3.8, actions and control and perceptual status are shown as **dark** pulses, while world conditions are shown shaded. As an example, we see how the ground ahead changes from being stable to unstable (depicted as a shaded rectangle for the cycle of interest). This results in the visual test returning a $\neg ok$ status leading to the agent taking the *test_footing* branch, which may return fail status (shown in Figure 3.8) leading the agent to move-foot and repeat the test with a different location. At this point, the world removes the obstacle (asynchronously), and the x-schema is able to respond appropriately by returning visual status *ok* and thus taking the next step. Actions have different durations, which can be controlled by the *rate* parameter (in Figure 3.8 we used a duration of 5 time units for the step and one each for the other actions). We also see the model of different actions **dynamically changing** resource levels, such as energy being consumed, as well as the results of x-schema executions changing world state such as decreasing the distance to destination.

Some readers may already have noticed a certain pattern to the control status information *ready*, *enabled* and *done*. In fact, it turns out that such information is routinely coded using grammatical and lexical devices, allowing a speaker to **focus** on specific sub-phases of an action or process. This phenomenon is widespread across languages and an object of study since Aristotle and is known in linguistics as the problem of aspect. X-schema based dynamic system models seem to provide a novel model of aspectual interpretation that avoids some well known paradoxes in other accounts and appears to provide a compositional semantics of aspect. Chapter 5 explores these issues in detail.

To reiterate, linguistic input only codes for specific features required by the x-schema. In all these cases, we **assume** the existence of a parser, that can provide partial parses by instantiating features of the f–struct to specific values. The assumption holds for the rest of the work reported in this thesis as well. Other parameters may come from perceptual or world state (such as the assertion of a *stable* ground) or from internal motivational and attitudinal states (such as the energy level in Figure 3.7). Thus, the set of features coded for by linguistic input, while constrained by the parameters required for x-schema execution, is still potentially quite large, and can give rise to distinctions in *manner*, *path*, *rates*, *aspect*, and other well known linguistic distinctions. The x-schema model for interpretation always executes the linguistic information provided by lexical items and grammatical devices in the context of the agent's internal state as well as the state of the world. Thus interpretation is always context-sensitive in this manner.

Modifying the parameters to the WALK schema allows the same structure to model different behaviors. Consider the case of the linguistic input *Tiny Step* shown in Figure 3.9. Here, the input label *tiny* codes for the size of the step, while the input *step* codes for the specific x-schema WALK as well as the specific sub-schema STEP . The basic idea is that the input words may refer to a specific x-schema or a sub-schema. Note that the referential possibilities bottom out at the appropriate synergies (or atomic actions), so in this scheme it would be tough to learn or describe in *language* the internals of a reaching behavior (such as specific joint angles or torques), though one could of course learn these internals through practice, demonstration, imitation or a variety of other skill-learning techniques.

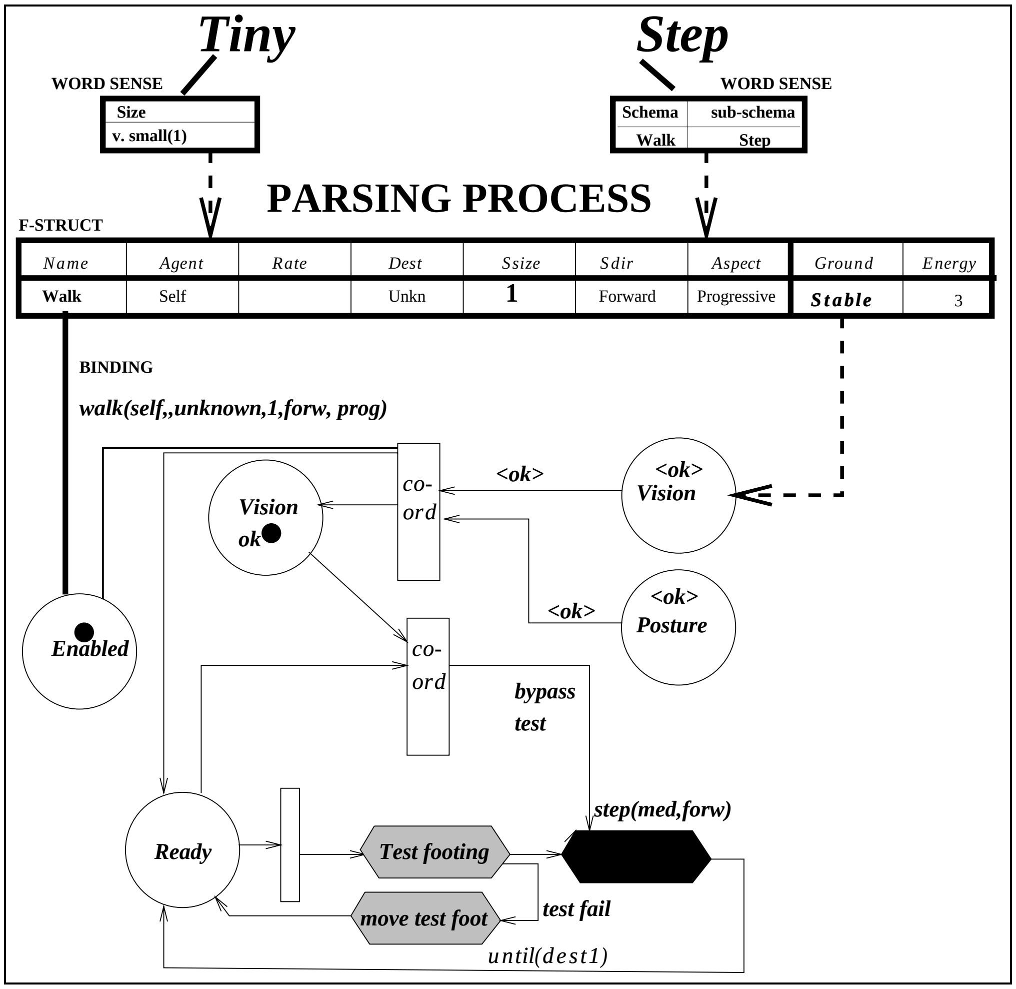


Figure 3.9: Interpreting the input *Take Tiny Steps.* Verb Phrases specify different initiation and execution parameters to x-schemas. In this case, the word label *tiny* is translated to executing the WALK x-schema with a tiny step size. Note that some of the parameters are left unspecified, and others take on default values. For instance, the rate parameter is unspecified at the input, but will be interpreted as a result of x-schema execution. The default destination is unknown, and the other default values are as shown in the figure.

Example 3 Taking tiny steps

Here we describe the execution of the WALK x-schema with step parameter set to 1 (corresponding to tiny). As usual, a row in the table below corresponds to the system state (technically specified by a marking vector). A column in the table below corresponds to a specific place in the x-schema. A cell in the table thus corresponds to the number of tokens present in a specific x-schema place for a specific state (given by the marking). Thus for example, in the initial state (Marking M_0), the agent has 8 units of energy (place 8 has value 8 in M_0), and the WALK x-schema is enabled (token at place 2), and the visual tests return *ok* status (token at place 6). As the execution (simulated) proceeds, energy decreases by an amount correlated to step size and the distance walked increases by the appropriate step size after every step. In accordance with the inherently parallel nature of x-schema executions, multiple tokens may change in one step and mulitiple transitions may be simultaneously active.

```
x-schema place to column mapping:
     1 = started
     2 = Enabled
     3 = ready
     4 = ongoing
     5 = done
     6 = VISUAL_OK
     7 = BYPASS?
     8 = Energy
     9 = Dist(walked)
Simulation:

Marking    1   2   3   4   5   6   7   8   9  | Transitions to...
============================================= | -----------------
 M0     |  0   1   0   0   0   1   0   8   0  | ( M1 )
 M1     |  0   0   1   0   0   1   0   8   0  | ( M2 )
 M2     |  1   0   0   0   0   1   0   8   0  | ( M3 )
 M3     |  0   0   0   1   0   1   1   8   0  | ( M4 )
 M4     |  0   0   1   1   0   1   0   7   1  | ( M5 )
 M5     |  1   0   0   1   0   1   0   7   1  | ( M6 )
 M6     |  0   0   0   2   0   1   1   7   1  | ( M7 M8 )
 M7     |  0   0   0   0   1   1   1   7   1  | ( M9 )
 M8     |  0   0   1   2   0   1   0   6   2  | ( M10 M9 )
 M9     |  0   0   1   0   1   1   0   6   2  | ( M11 )
 M10    |  1   0   0   2   0   1   0   6   2  | ( M12 M11 )
 M11    |  1   0   0   0   1   1   0   6   2  | ( M13 )
```

```
M12   |  0   0   0   3   0   1   1   6   2  | ( M13 M14 )
M13   |  0   0   0   1   1   1   1   6   2  | ( M15 )
M14   |  0   0   1   3   0   1   0   5   3  | ( M16 M15 )
      =============================================
         1   2   3   4   5   6   7   8   9
```

■

Example 3 shows the actual simulation for the tiny steps input shown in Figure 3.9. The crucial thing to notice in the simulation is the dynamic nature of the interpretation, where the actual walk parameters are changing as the simulation proceeds. The simulation is obviously highly-responsive to changing world and motivational state context, exhibiting the context-sensitivity required for command interpretation in a dynamic environment.

While *tiny* obviously codes for a small sized step, Example 4 shows the interpretation for the input *Giant Steps*. Here the key thing to note is that since the size of the step is very large, the **energy** required per-step is also extremely high, a fact reflected in the simulation, which consumes 7 units of energy per-step in this case, as opposed to the 1 unit consumed in the earlier case. In this way **correlations** between features in the f–struct are captured in the relevant x-schemas as compiled knowledge used in execution. In Chapter 7, we will see another method for capturing correlations is used when the knowledge is not compiled directly.

Example 4 Taking giant steps

Here we describe the execution of the WALK x-schema with step parameter set to 7 (corresponding to Giant). As usual, a row in the table below corresponds to the system state (technically specified by a marking vector). A column in the table below corresponds to a specific place in the x-schema. A cell in the table thus corresponds to the number of tokens present in a specific x-schema place for a specific state (given by the marking). The interesting thing to note here is that **both** the amount of distance walked per step (7 in this case) as well as the amount of energy consumed per step (8 in this case) are different from the "tiny step" case. As the execution (simulated) proceeds, both energy consumed and distance walked are correlated to the step size parameter.

```
x-schema place to column mapping:
     1 = started
     2 = Enabled
     3 = ready
     4 = ongoing
     5 = done
     6 = VISUAL_OK
     7 = BYPASS?
     8 = Energy
     9 = Dist(walked)
Simulation:

Marking    1   2   3   4   5   6   7   8   9  | Transitions to...
============================================= | -----------------
 M0     |  0   1   0   0   0   1   0  48   0  | ( M1 )
 M1     |  0   0   1   0   0   1   0  48   0  | ( M2 )
 M2     |  1   0   0   0   0   1   0  48   0  | ( M3 )
 M3     |  0   0   0   1   0   1   1  48   0  | ( M4 )
 M4     |  0   0   1   1   0   1   0  40   7  | ( M5 )
 M5     |  1   0   0   1   0   1   0  40   7  | ( M6 )
 M6     |  0   0   0   2   0   1   1  40   7  | ( M7 M8 )
 M7     |  0   0   0   0   1   1   1  40   7  | ( M9 )
 M8     |  0   0   1   2   0   1   0  32  14  | ( M10 M9 )
 M9     |  0   0   1   0   1   1   0  32  14  | ( M11 )
 M10    |  1   0   0   2   0   1   0  32  14  | ( M12 M11 )
 M11    |  1   0   0   0   1   1   0  32  14  | ( M13 )
 M12    |  0   0   0   3   0   1   1  32  14  | ( M13 M14 )
 M13    |  0   0   0   1   1   1   1  32  14  | ( M15 )
 M14    |  0   0   1   3   0   1   0  24  21  | ( M16 M15 )
      =============================================
1   2   3   4   5   6   7   8   9
```

■

As another example, consider the case where the step size is specified as *large* , corresponding to a large step. This would correspond to taking a large step, covering a greater than normal distance per step and requiring fewer iterations to reach the destination.[5] Similarly the same x-schema is capable of exhibiting different types of *steps* (including *sidestep* , where the direction of the step away orthogonal to the destination), cautious steps (where there is a test footing before every step), etc.

By now it it should be clear how our model is capable of handling adverbs like *quickly* or *slowly* as well as other lexical variants such as *amble* or *saunter* (coding for direction (or lack thereof), rate and other manner distinctions). Destinations and paths are coded through the prepositional attachments to the phrase as shown in Figure 3.10.

Figure 3.10 shows the case where the same x-schema as before is used to model walking to the store. The individual word senses are not shown partly for ease of exposition and partly because the parsing process which results in instantiating the f–struct is not modeled anyway. In this case, the run-time binding *destination* gets bound to *store* at runtime, and the condition check at the bottom loop changes from $until(dest1)$ to $until(at(store))$. Figure 3.5 showed how the

```
repeat <action> until <condition>
```

control works. Of course, if the **destination** specification is not in the input utterance, or **unknown** (as was the case with *Tiny Step*), the $dest1$ is not specified, at *at_dest* (ref. Figure 3.7) is never marked, and the x-schema continues execution forever, unless some other parameter like number of steps (as in *Take* 3 *tiny steps*) specifies the end state **or** *energy* runs out, **or** the WALK x-schema is otherwise disabled. We will see these types of cases in Chapter 6.

As an example of how x-schemas could be used for simulative inference just as readily as command-obeying, consider Figure 3.10 which shows the WALK schema interpreting the sentence *John is walking quickly toward the store.* Here, the individual words

[5] In the implementation as it stands, the size parameter is restricted to be one of *seven* values 7 indicating a *giant* sized step, 1 indicating a very small sized step.

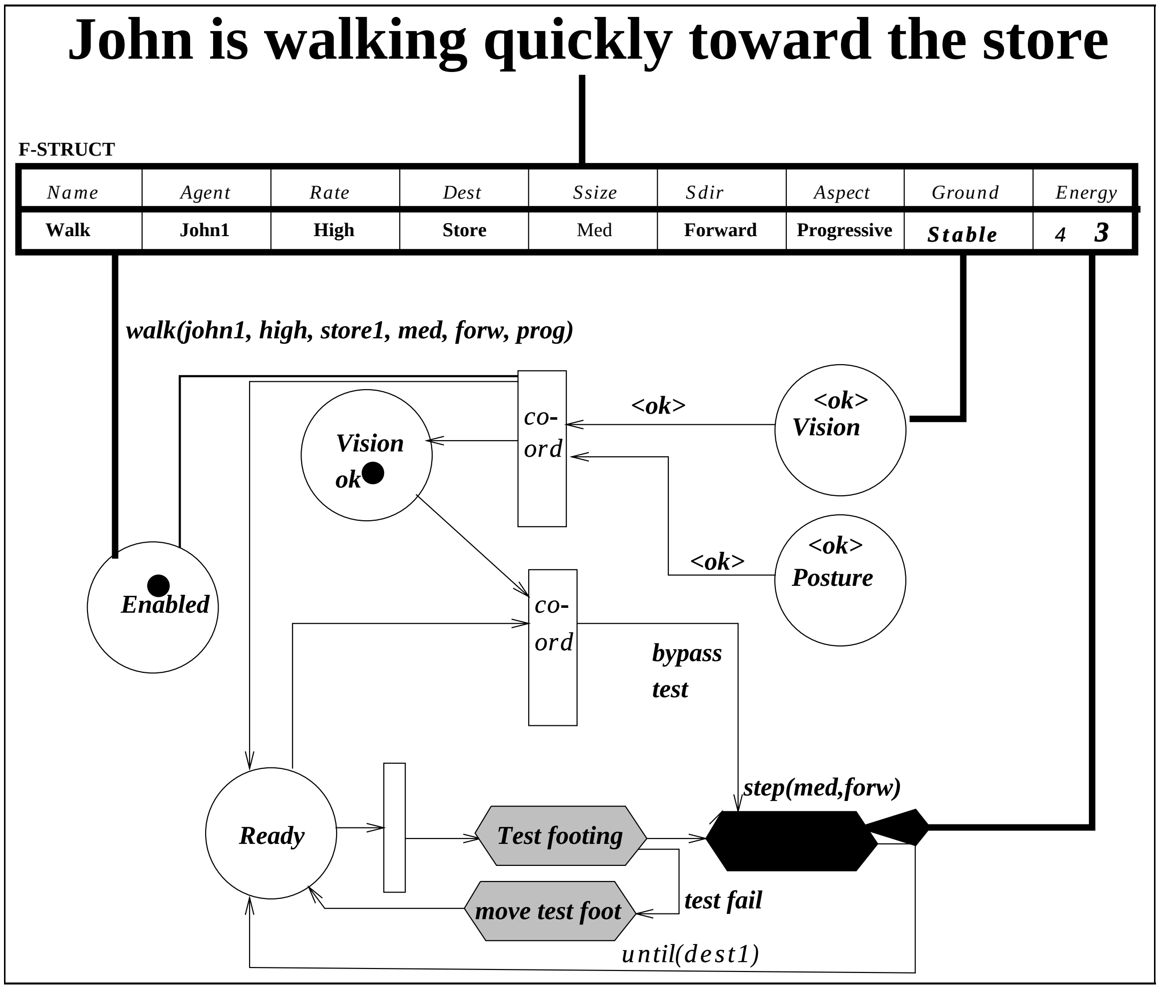


Figure 3.10: Interpreting the input *John is walking quickly toward the store.* Verb Phrases specify different initiation and execution parameters to x-schemas. In this case, the word label *quickly* is translated to executing the WALK x-schema with high *rate* parameter, etc. Note how different components of the input utterance supply different execution parameters, such as the prepositional phrase supplying the destination, the adverbial modifier the rate, etc.

in the sub-categorization frame of walk code for *schema name* (walk), *agent* (John), *rate* (quickly), *destination* (store), *direction = forward* (toward destination), and *aspect* (present progressive) (from the *be + V-ing* construction). While, we have seen how the other parameters are modeled, we still have to see how the progressive specification of *is walking* is handled by our model. That is the subject of Chapter 5.

Other expressions and words that use the WALK schema setting different parameters (including some affective and motivational states) include *amble*, *saunter*, *giant steps*, *cautious steps*, *measured steps*, *crawl*, *amble*, *trundle*, *trudge*, and *hike*. In Chapter 8 and Chapter 9 we will argue that some of the distinctions between these and other embodied words point to crucial semantic features that are projected metaphorically onto the domain of abstract plans, events and actions such as discourses about international economic policies.

3.6 Learning Action Verbs

Assume that all the relevant x-schemas are already present. Then the problem of learning embodied verbs becomes the problem of associating specific parameter values to specific verb-labels. The distinction between different action verbs therefore may be that they refer to different x-schemas **or** that they refer to the same x-schema but with different parameter settings. For instance, the difference between *walk* and *push* is that they refer to completely different synergies or muscle-group control systems, hence to different x-schemas. On the other hand, both *push* and *shove* refer to the same x-schema (PUSH) but with different parameter settings (such as different force values and perhaps different postures). Similarly *amble* and *walk quickly* refer to the same underlying x-schema (WALK) but with different **rate** parameter settings. Using x-schemas to represent verb phrases makes the problem of associating verb phrase labels to specific parameter settings bi-directional. The learning algorithm can thus be used to **label** x-schema executions with the appropriate verb-phrase. In the labeling phase, action word senses are clustered based on the specific parameter settings of the f–struct. Once learned, the inverse mapping (from word senses to f–structs) can be used to set x-schema execution parameters (given a world state) for **obeying**. Such a learning and execution scheme for the restricted domain of hand action verbs is the dissertation project of David Bailey (Bailey 1997).

3.7 Plan analysis with x-schema representations

In most cases, we will use x-schemas in a simulative mode, generating inferences through execution in real-time. Chapter 6 goes through the details of our architectural design. However, in Chapter 5, we use x-schema analysis techniques to provide a formal algorithm to resolve some problems and paradoxes in model-theoretic accounts of verbal semantics. Here will will describe the two relevant algorithms in anticipation of their use in Chapter 5.

The central features of x-schemas, namely that they are *executing* and *action-based* inherently allows x-schemas to obey the Frame axioms. Specifically, the action-based **executable** formalism allows frame axioms to be structurally encoded in the topology of the net and the firing rules. The execution semantics (based on the *firing rule* and *next-state* function) for a transition/action changes markings from the context in which the action takes place to a completely local change to the state that impacts only the direct effects of the action. Thus the execution of an action automatically respects the frame axioms, which can be read off from the topology of the network.

3.7.1 Temporal Projection

In the temporal projection problem, one is interested in the result of executing a specific sequence of actions starting in a specific system state. In the standard case, the problem consists computing a final state given an initial state s_0 along with a sequence of sets of actions $\mathcal{S} = [a_1, \ldots a_n]$.

Temporal projection can be solved very efficiently by x-schemas. The following algorithm is a *massively-parallel* solution to the problem of temporal projection. Each atomic action is a transition, each hierarchical action is composed of the hierarchical network shown in Figure 3.5. The initial state is set to be the initial marking on the x-schema. Enabled transitions fire as a result of the initial marking taking the x-schema to new marking. The new marking in combination with the input marking enabling the next action in the set $\mathcal{S}$ is then the new system state, which allows for further evolution.

```
$PROJECT(M, s_0, \mathcal{S})$
  **Input** : X-schema $M$ , and initial state $s_0$ :
  A sequence of action sets, $\mathcal{S} = [a_1, \ldots a_n]$ .
  **Output**: A state vector that results from executing the sequence
  of action set $\mathcal{S}$ , given $M$ and $s_0$ .
  **begin**:
    Set initial marking $M_0$ . $\forall p \in s_0 : M_0(p) = 1$ ,
    $\forall p \notin s_0 : M_0(p) = 0$ (where $p \in P$ ).
    **while** $1 \leq i \leq n$ , **do**:
      Fire enabled transitions $T_{e_{i-1}}$ with marking $M_{i-1}$ .
      (The next state function described earlier takes the x-schema
      to a new marking $M_i$ , based on the transitions
      enabled by $M_{i-1}$ .)
      Set $M_i = *a_i \bigcup M_i$
    **endwhile**
  **end**
```

Essentially, the projection algorithm is the application of the extended state function to the action set in $\mathcal{S}$. Note that the model is easily able to handle both the projection of explicit actions, as well as world evolutions (including ramifications) which may occur at any time during projection, giving rise to a new marking vector during execution. The algorithm is linear in the number of actions in $\mathcal{S}$ and can be executed in $O(\mathcal{S})$ time. Note that at each application of the next-state function **multiple**, **concurrent** transitions may be enabled and fired. This allows for a natural extension into modeling the behavior of multiple agents in cooperative action and planning scenarios. However, this possibility will not be explored further in the work reported here.

3.7.2 Reachability

Reachability is a fundamental problem of dynamic systems. In terms of x-schemas, the problem is stated as follows.

Given that the state space of an x-schema evolves through execution of the x-schema, we can define the following entities that correspond to the **reachable** states of an x-schema, given an initial marking.[6]

[6]The reachability definitions are equivalent to the Petri net reachability, so the reader knowledgeable

Definition 5 Immediately reachable states

For an x-schema $\mathcal{S}$, with a state s_i, state s_j is **immediately reachable** if there exists a transition $t_k \in T$ such that $\delta(s_i, t_k) = s_j$.

Extending this concept, we can define the set of reachable markings for a given x-schema in some initial state. Basically, if s_j is immediately reachable from s_i, and s_k is immediately reachable from s_j, then s_k is in the reachability set of s_i. Thus the **reachability** relationship is the reflexive transitive closure of the immediately reachable relationship.

Definition 6 Reachability set

The reachability set $\mathcal{R}(\mathcal{S}, s_0)$ for an x-schema $\mathcal{S}$ with state s_0 is the smallest set of markings defined by a) $s_0 \in \mathcal{R}(\mathcal{S}, s_0)$, and b) If $s_j \in \mathcal{R}(\mathcal{S}, s_0)$ and $s_k = \delta(s_j, t_l)$ for some $t_l \in \mathcal{T}$, then $s_k \in \mathcal{R}(\mathcal{S}, s_0)$.

Definition 7 Reachability Given an x-schema $\mathcal{S}$ with an initial state s_0 and a final state s_f, is $s_f \in \mathcal{R}(\mathcal{S}, s_0)$?

To answer this question, we will simulate the evolution of the x-schema from the state s_0 and construct a reachability tree of states. To construct the reachability tree, for each immediately reachable next-state s_{i+1} we will draw a directed arc from s_i to s_{i+1}. The simulation halts with a **no** answer either if there is no enabled transition left. The tree is truncated at the leaf where a previously encountered state is repeated. Return with a **yes** answer if s_f is a node in the reachability tree. Since most of the x-schemas considered in this thesis are bounded, we will use the reachability tree method.

To construct the reachability tree, we define nodes as frontier nodes (nodes yet to be processed), duplicate nodes (nodes that have been seen before), terminal nodes (leaves of the reachability tree), or an internal node (a non-leaf node). Frontier nodes are converted to one of the other three types. As long as there are frontier nodes, the algorithm processes them. Initially, the initial marking is defined as the *root* of the tree, and the first frontier node to be processed. The set X in the algorithm below holds the current set of frontier nodes.

about Petri nets can skip the rest of this section and read the Appendix where the exact equivalence between Petri nets and x-schemas is shown.

```
$REACHABLE?(M, s_0, s_f)$
  **Input** : X-schema $M$ , and initial state $s_0$ and final state $s_f$
  **Output**: **true** if $s_f$ is **reachable** from $s_0$ ( $s_f \in \mathcal{R}(M, s_0)$ ); **false** otherwise.
  **begin**:
    **loop**
      If $X$ is empty **return false**
      Choose $x \in X$ (a frontier node from the set $X$ )
      If $s_f$ is the same as the marking for node $x$ , **return true**.
      Label $x$ as a *duplicate node* if the same marking $s_x$
        exists in another non-frontier node $k$ ( $s_k = s_x$ )
      Label $x$ as a *terminal node*
        if there is no enabled transition for that marking
      **for** each enabled transition $t \in s_x$
        Fire $t$
        Create a new tree node $z$ with an arc from $x$ to $z$ labeled $t$
        Set the state of a new node ( $z$ ) to $s_z = \delta(s_x, t)$
        Label $z$ as a *frontier node* (add to $X$ )
        Label $x$ as an *interior node* (remove from $X$ )
      **end for**
    **end loop**
  **end**
```

For general purpose nets the reachability algorithm has been shown to be decidable, but to be $PSPACE$-complete. However, in the cases we are interested in (outlined in Chapter 5), the occurrence sequence over a few time steps is sufficient.

3.8 Comparison to Other Models

3.8.1 Schemas in natural language understanding

In Natural Language Understanding, the work of Schank and his students is clearly connected to the work here. Early conceptual dependency theory (Rieger1975; Schank & Abelson 1977) used the idea of schemas to represent world knowledge in discourse interpretation. In his seminal work, Rieger implemented a system where sentences were mapped to a semantic representation that generated inferences. All possible inferences that could be drawn were drawn. In the presence of ambiguity, the interpretation that generated the most number of inferences was selected. While his proposal showed the close connection between domain knowledge and interpretation, the uncontrolled nature of forward inferences led to severe intractability and counter-intuitive results (Charniak 1976).

Later, Schank's group scaled back the effort and used the idea of Scripts, where inferences were limited to narrow domains and text interpretation relied exclusively on stereotyped scenarios. This limitation proved too severe, and led to extensions that pertained to recognizing the importance of goals and their complex interactions in understanding stories (Wilensky 1983), and the need for dynamic memory assemblages (Kolodner 1984).

We share the basic premise inherent in conceptual dependency theory that sentences are ultimately mapped to a semantic representation that generates inferences. We also share many of the goals and guiding intuitions from the early work in conceptual dependency, including the idea that a theory of actions should be useful both for planning and language understanding. Such considerations naturally led us to a rich representation of actions that is able to model the effects of dynamic enabling and disabling of goals, and the effects of resource production and consumption. As (Wilensky 1983) convincingly demonstrates, these are among the essential primitives for building story-understanding systems. One distinguishing feature of our x-schema-based representation of events is that they are more fine-grained than other proposals we have seen. This finer granularity resulted from our interest in modeling aspectual distinctions made by different languages (ref. Chapter 5). We also believe that the finer-grained nature of our representation of aspect presents one natural way to constrain automatic inference in reasoning about event descriptions. Additional evidence comes from the observation that metaphoric projection of events across different domains appears to respect the temporal and aspectual distinctions made by our representation. Our model is thus able to exploit these characteristics in metaphor interpretation as shown in Chapter 9.

Another distinguishing feature of our representation (see Chapter 6 Section 6.3 for further details) is that x-schemas are executable graph-structures (bi-partite, cyclic graphs). The graph-based representation allows us to formally state and reason about inter-schema relations declaratively while using their real-time execution capability for inference. This allows our representation to be used to declaratively to specification and design or procedurally for projection and automatic inference. We believe this property to be essential for representations that are to be used both for acting and for reasoning about action descriptions.

Connected to the question of general purpose inference in language understanding, computational models of semantic interpretation often rely on a form of inference called marker passing (Norvig 1989; Wu 1992). One possibility that is not explored in this thesis but is part of ongoing research is that x-schemas may provide formal models of massively-parallel marker passing. Chapter 10 touches on ongoing work exploring this connection.

3.8.2 Action representations for planning

X-schemas are partly motivated by previous AI models of planning and acting. As was pointed out earlier, when used purely declaratively x-schemas could be considered fine-grained models of action that can be used to model partially ordered, hierarchical plans with variables, loops and bounded recursion. Their ability to model true concurrency (as opposed to interleaving) as well as stochastic and durative actions renders them more expressive than traditional partial order planning models such as (Fikes & Nilsson 1971; Sacerdoti 1975; Chapman 1987; Barrett & Weld 1994). Of course, this is not always good news since optimal planning with such expressive representations is likely to be harder as well.[7] Some consolation comes from recent work on GRAPHPLAN (Blum & Furst 1995) which shows how formal bi-partite graph structures very similar to x-schemas can aid in constructing efficient planning algorithms in real domains (by annotating, analyzing, and decorating the graph structure in specific ways).

We believe that it would be quite productive to investigate x-schema structures for efficient planning. For the interested reader, the connection between x-schemas and Petri nets can be exploited for structural and dynamic analysis as well as for synthesis of x-schema based plans. In particular, the results of (Murata 1991) can be used for *planning* with x-schemas, the results of (Portinale 1994) on a Petri net based abduction framework can be used to solve the *postdiction* problem in restricted domains using *T-invariants* (a polynomial time operation) of x-schemas. Furthermore, we have ongoing work investigating a connection to resource-based logics such as linear logic (Girard 1987) and to dynamic versions of situation calculus such as the fluent calculus (Thielscher 1995). In this connection, we also note the work of (Tollu & Masseron 1993) who show the equivalence of deduction in linear logic to planning for dynamic systems. This is further evidence that efficient x-schema

[7]Note that STRIPS-based planning is already in PSPACE so worst-case complexity comparisons are unlikely to be illuminating.

analysis techniques may lead to better search-control techniques in linear deduction.

3.8.3 Plan executives and production systems

While the declarative structure of x-schemas can be analyzed by graph-theoretic algorithms, their real-time execution semantics gives them the ability to respond asynchronously to changes in perceived (or projected) world states as well as dynamic changes in resource levels, and to goal assertion or retraction. This gives them different dynamics than the traditional execution agents which essentially use an "eyes closed" strategy. For example, their ability to model conditional plans allows x-schemas to check world state through scheduled sensing actions and branch on the results. The perceptual tests and the *test_foot* branch of the WALK x-schema are examples of this phenomenon. In this respect, x-schemas are comparable to reactive plans. This connection is explored in section 3.8.4. Furthermore, x-schemas may automatically get triggered and modify execution trajectory asynchronously based on the perceived world state. In this respect x-schemas are similar to production systems (see Chapter 6). One central difference between x-schemas and other production systems or compiled plans is that their continuous time distributed and asynchronous control allows x-schemas to update and respond continuously to changes in world or internal state. As an example of this phenomenon, Chapter 6 shows how the presence of an unexpected *bump* that interrupts an ongoing step immediately triggers recovery strategies to stabilize and reestablish control. Such high responsiveness is a typical side-effect of the tight action-reaction coupling in x-schemas.

An important feature of x-schemas is their ability to model truly parallel execution. The underlying Petri-net based semantics allows x-schemas to model *true concurrency*. While most action representations are able to model partially ordered plans, the execution agent must have the distributed control circuitry to monitor them. Note that transforming concurrent actions into nondeterministic action selection from a linear order ($conc(a, b) = a.b \vee b.a$) loses this parallelism. Their true-concurrency semantics allows x-schemas to use capacities to perform more than one action at once. This means that x-schemas have a natural execution semantics to model the execution of multi-agent plans or multiple interacting plans for a single agent.

3.8.4 Behavior controllers and reactive planners

The high-responsiveness, distributed control and situated nature of x-schemas invites comparison to other proposals in mobile robotics and reactive planning.

In robotics, we note the similarities between x-schemas and *behaviors* used by the behavior-based Robotics group (Brooks 1986). It should be obvious the explicit equivalence of x-schemas to $GSPN'S$ (ref. Section 3.10) allows them to model all the control aspects of finite state machine (FSM) type behavior compilers employed in (Connell 89) ($GSPN \supseteq SPN \supset PN \supset FSM$ (Murata 1989)). The extension to continuous time allows our model to exhibit the high responsiveness to dynamic and uncertain environments that characterizes behavior-based models. In addition, our model of x-schemas can model in detail issues of synchronization, forking, coordination, and stochastic priorities, which are likely to be difficult for FSM based controllers. One fundamental difference is that our model can be driven by environment features (as in the case of *behaviors*) **or** through linguistic input (unlike *behaviors*). Furthermore, the same x-schema model is capable of fast response to internally generated triggers and agent state conditions (such as assertion or retraction of goals, resources, enabling conditions, etc.) rendering it capable of simulative reasoning and inferences (details of the implemented x-schema based inference system can be found in Chapter 6). X-schemas also differ from compiled (reactive) plans such as Situated Control Rules (Drummond 1989) in that x-schemas are able to take run-time bindings and parameters, thereby allowing flexible execution with a smaller circuit size. In the other cases of compiled plans, large amount of circuitry is required to anticipate all contingencies.

Among reactive planners, the RAP (Reaction Action Package) (Firby 1989) is notable in that it uses a fairly high-level specification language and world model. In Firby's system, a RAP consists of a *task* to be achieved, a *success* condition (to verify that a goal has been achieved), and a set of *methods* for carrying out the task. A method consists of a *context* which is a world state that must hold for the method to be applicable, and a *task network* that consists of subtasks and primitive actions with constraints on their performance. The RAP system maintains an agenda of tasks to be performed (initialized by top level goals), and a world model which can be updated by perceptual tasks or through side-effects of actions. The robot repeatedly chooses a task from the agenda (based on some scheduling criteria) and if there is a primitive method for the task it is dispatched as an effector command. If the method is not primitive, the robot chooses one of the methods for the task whose context is currently satisfied and adds that task network to the agenda. When all subtasks in a specific task network are satisfied, the success criterion is checked. If *true*, the task is done and the robot moves on to a new item on the agenda. If a task fails, the method in which it was a subtask is removed completely, and the task remains on the agenda.

Chapter 6 describes our compositional theory of x-schemas which bears some strong resemblances to RAP-based systems. Individual x-schemas are similar to RAPS. The triggering conditions for x-schema initiation and execution are similar to RAP contexts. The goal-based enabling and disabling mechanisms allow the x-schema based system described in Chapter 6 to avoid sequential search through an agenda to evaluate success conditions and remove completed tasks. All implemented RAP systems we have seen work on one task at a time.[8] In contrast, the x-schema composition machinery allows for multiple parallel tasks to be simultaneously active and their results and effects on the world state create new goals and contexts. The price we pay is of course that analyzing x-schema plans is likely to be harder than analyzing RAP plans. This is where the graph-based representation and connection to Petri nets should help. Another difference is that our finer grained action representation allows for a fairly rich set of inter-schema control relations to be declaratively specified and modeled. The reader is referred to Chapter 6, Section 6.3 for a detailed description of the set of inter-schema control relations available. In contrast, while the central components of a RAP based model are easy to specify declaratively, detailed control is often hidden inside a method and we suspect would be fairly hard to specify formally within the language itself. As (Wilensky 1983) points out, a theory of plans that can be used in language understanding should be able to reason about fairly complicated resource and goal interactions. We take that to mean that such detailed information should be available for theory specification and analysis, rather than hidden inside some procedure. We believe the model in Chapter 6 to be consistent with this view.

In this connection, we believe that x-schemas provide a formal model of reactive plans and behaviors. The resulting active model has a well specified real-time execution

[8]This is not an inherent limitation of RAP, since the responsibility of tracking inter-task dependence and status is placed on the scheduler. But putting this burden on a scheduler may not be the best overall design since it will potentially slow down the responsiveness of the system while the scheduler checks if all dependencies are maintained before searching the agenda for new tasks.

semantics, and is capable of modeling hierarchical and concurrent actions and of *action and execution monitoring.* Furthermore x-schemas can take into account serendipitous effects, provide limited flexibility in the face of exogenous events, and handle event-based interrupts. X-schemas are capable of modeling atomic and durative actions, as well as uncertainty in world transitions and action effects. Finally, as Chapter 4 shows, x-schemas have deep connections to connectionist models of reasoning, allowing for a single representation to be viewed as a plausible computational model of high-level, coordination-oriented control of synergies, as well as a candidate representation for task-oriented robotic behaviors.

3.8.5 Teleo-reactive programs

Recently, (Nilsson 1994) proposed the notion of *Teleo-reactive* control programs ($T-R$ programs) which were developed to address the need to formalize a representation to compute and organize actions for autonomous agents operating in dynamic environments. Like x-schemas, $T-R$ programs are parameterized routines with run-time variable binding. Like x-schemas, and unlike *behaviors*, these representations are designed to work with stored environment models as well as direct perceptual and environment input.

On the one hand, $T-R$ representations appear to be more expressive than x-schemas in that they allow continuous variables (and thus continuous state) at certain levels of their representation, while in our representation *time* is the only continuous variable allowed at all levels. All other continuously varying entities are at the level of *synergies*, whose internal state is un-inspectable at the x-schema level. Hence our model employs a **continuous time**, **discrete state** representation which is typical of structured connectionist models. Also, we note that while $T-R$ trees have addressed the issue of learning $T-R$ programs, x-schema learning remains a future project.

On the other hand, $T-R$ programs appear not to have paid much attention to modeling multiple concurrent actions, or to explicit representation of synchronization and other temporal issues pertaining to coordination of multiple actions. Neither does the $T-R$ model have the notion of stochastic events and transitions. We found the issues of coordinated control to be of crucial importance in reasoning about action and event descriptions. Furthermore, being closely related to well-studied models of complex, distributed stochastic systems, our representation allows us to readily apply well known structural and dynamic analysis and specification algorithms to x-schema design and analysis. Preliminary efforts in comparing the two representations leads us to believe that a detailed analysis of the relationship between $T - R$ programs and x-schemas will be highly productive.

3.9 Conclusion

X-schemas are a graphical and mathematical representation with fine-enough granularity to model asynchronous and concurrent action systems with interacting plans and goals. The active nature of x-schemas allows them to be highly reactive and respond asynchronously to changes in perceived or projected world state. The connection to Petri nets allows the single system to simulate dynamic, stochastic, and concurrent activities as well as allow for the setting up of state equations and other mathematical models to theoretically study its structural and dynamic behavior. X-schema executions respect resource constraints and directly model the production and consumption of resources. Furthermore, x-schemas have a truly concurrent semantics which allows them to be natural candidates for modeling multi-agent systems.

The thesis that motion and manipulation words refer to parameters that **initiate** and supply **control and monitoring** parameters to the underlying action x-schema allows us to use a uniform mechanism for action and interpretation. Our hypothesis naturally results in a *dynamic* semantics for action verb phrases. While we are obviously partial to our specific x-schema model and feel that the extended Petri net representation can capture much of the needed functionality, our allegiance is ultimately to a larger class of models that can exhibit the high-responsiveness and dynamic system characteristics involved and that can be encoded in a structured connectionist framework. To this end, the next chapter outlines our mapping from x-schemas to a popular and widely used structured connectionist model. In the rest of the work reported in this thesis, we attempt to argue that an x-schema like dynamic model is not only useful for interpreting expressions about motion and manipulation words but is essential for context-sensitive interpretation and to generate pragmatic discourse-level inferences about abstract event and action descriptions.

3.10 Appendix A

Here we prove the equivalence between x-schemas and Petri nets. The first part is concerned about the non-timed case, while the second is concerned about the timed and stochastic case. We begin with the following definitions.

We represent x-schemas as a modified class of high-level Petri Nets (Reisig 1985), where the modifications allow for stochastic transitions, as well as special compositional states and transitions (decomposable to a an ordered collection of primitive states and transitions), and run-time parameter binding. We will show that in the case where the set of different objects to be bound to is finite (including finite parameter values), x-schemas are equivalent to bounded ordinary Petri nets. We begin by defining the basic Petri net.

Definition 8 Basic Petri Net
A Petri Net (PN) is a 5-tuple $PN = (P, T, I, O, M_0)$ where

- $P = \{p_1, \ldots p_n\}$ is a finite and non empty set of places,
- $T = \{t_1 \ldots t_m\}$ is a finite and non-empty set of transitions,
- $P \bigcap T = 0$,
- $I : T \rightarrow P^\infty$ is the *input* function, a mapping from a transition to a bag of places,
- $O : T \rightarrow P^\infty$ is the *output* function, a mapping from a transition to a bag of places,
- M_0 is the initial net marking, a function from the set of places to the non-negative integers $\mathcal{N}$, $M_0 : P \rightarrow \mathcal{N}$.

Definition 9 Enabled Transition
A transition $t_j \in T$ in a Petri Net $N = (P, T, I, O, M)$ is *enabled* if $\forall p_i \in P$, $M(p_i) \geq \#(p_i, I(t_j))$ where $\#(p_i, I(t_j))$ represents the multiplicity of place P_i in bag $I(t_j)$.

Definition 10 Dynamic Behavior
The next-state function $\delta : Z_+^n \times T \rightarrow \delta : Z_+^n \times T$ for a Petri net $N = (P, T, I, O, M)$, $|P| = n$, with transition $t_j \in T$ is defined if t_j is enabled. The next state is equal to M' where:

$$M'(P_i) = M(P_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j)) \forall p_i \in P \tag{3.2}$$

This can be extended to a sequence of transitions as follows

$$\delta(M, t_j\sigma) = \delta(\delta(M, t_j), \sigma), \quad (3.3)$$
$$\delta(M, \lambda) = M$$

where λ represents the null sequence.

Definition 11 High-Level Petri Net ($HLPN$)

A $HLPN$ is a 6-tuple $HLPN = (P, T, C, I, O, M_0)$ where

- P, T are the places and transitions of an ordinary PN net defined earlier.
- C is a type (color) function defined from $P \bigcup T$ into finite sets.
- I and O are forward and backward incidence functions defined on $P \times T$ st.
- $I, O \in [C(t) \rightarrow C(p)_{MS}], \forall (p,t) \in P \times T$
- M_0 is a marking function defined on P describing the initial marking such that $M_0(p) \in C(p)_{MS}, \forall p \in P$.

Definition 12 Dynamic behavior of HLPN's

- A transition $t \in T$ is *enabled* in a marking M w.r.t. a type $c^{'} \in C(t)$ (denoted by $M(t, c^{'}) >$), **iff**

$$M(p)(c) \geq (p,t)(c')(c), \forall p \in P, c \in C(p) \quad (3.4)$$

- An enabled transition may *fire* in a marking M w.r.t. type $c^{'} : c^{'} \in C$ yielding a new marking M' (denoted as $M \rightarrow M'$) or $M(t, c') > M'$ with

$$M'(p)(c) = M(p)(c) + O(p,t)(c')(c) - I(p,t)(c')(c)(\forall p \in P, c \in C(p)) \quad (3.5)$$

- The various properties defined for PN can also be defined for a $HLPN$. The reachability set of a $HLPN$ $R(HLPN) := R(HLPN, M_0) := \{M | M_0 \Rightarrow^* M\}$ where $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

Definition 13 Bounded-net

Given a marked PN $\Sigma = (P, T, C, I, O, M_0)$, a place $p \in P$ is *k-bound* iff $\forall M_i \in R(N, M_0)(\#p \leq k)$. Σ is k-bounded iff $\forall p_i \in P$ (p_i is k_i bounded) and $K = max_i[k_i]$.

Lemma 1 . *HLPN's with finite number of types can be* unfolded *into a unique ordinary net PN. Also Bounded HLPN $\Rightarrow$ Bounded PN.*

Proof: A $HLPN = (P, T, C, I, O, M_0)$ can be *unfolded* into an ordinary Petri net PN in the following way:

1. $\forall p \in P, c \in C(p)$, we create a place (p, c) in PN.
2. $\forall t \in T, c^{'} \in C(t)$, we create a transition $(t, c^{'})$ in PN.
3. We define the input (I) and output (O) function on PN as

$$I(p, c)(t, c^{'}) := I(p, t)(c^{'})(c) \tag{3.6}$$
$$O(p, c)(t, c^{'}) := O(p, t)(c^{'})(c) \tag{3.7}$$

4. The initial marking of PN is

$$M_0(p, c) := M_0(p)(c), \forall p \in P, c \in C(p) \tag{3.8}$$

The unfolded net PN is thus given by:

$$PN = (\bigcup_{p \in P} \bigcup_{c \in C(p)} (p, c), \bigcup_{t \in T} \bigcup_{c^{'} \in C(t)} (t, c^{'}), I, O, M_0) \tag{3.9}$$

■

Lemma 2 . *X-schemas (with finite parameter/values) can be converted to bounded HLPN's with finite number of types.*

Proof: The proof proceeds in several steps.

1. *enable* conditions (preconditions/etc.) consist of tokens that are not consumed at the firing of a transition. The incidence function is $m_{p_o} - m_{p_i} + m_{p_{ei}}$. This can be modeled by a Petri net place that is both *input* as well as *output* to the relevant transition. This construction preserves the boundedness of the original net since it does not produce new tokens. In fact, preconditions and post-conditions are 1-bounded.

2. *consume* arcs correspond to the standard PN input arc. Here a specific amount of a resource is consumed by the firing of a transition.
3. Consumed resources are bounded from their initial values. In a finite number of transitions the net reaches the final marking. Since in each step the amount of produced resource is finite ($\leq w_{max}(T*)$), the total number is finite as well.
4. Parameters and dynamic objects are representatable as token types. In this thesis, we assume that the world consists of a finite number of types, and parameters consist of a finite number of values (integer parameters are bounded). As long as this assumption is valid, the number of token types is finite, and Lemma 1 allows us to reduce the $HLPN$ to an ordinary PN . Note that if the $HLPN$ is bounded (as in the x-schema case), the constructed PN is bounded as well.
5. From the discussion above, we conclude that without inhibitory arcs, the x-schemas are bounded ordinary Petri Nets. With the introduction of inhibitor arcs, a transition is enabled when all of its non-inhibitory inputs I_e are marked with $(I_e) \geq w_{et}$ and places with inhibitory arcs onto t are empty. In general, introduction of inhibitory arcs can increase the modeling power of Petri Nets to that of Turing machines; however in the case of bounded ($k - safe$) nets, a net with inhibitor arcs can always be transformed into an equivalent net without inhibitor arcs (Peterson 1981). Thus as long as x-schemas are bounded, the addition of inhibitory arcs does not increase their representational power to a Turing machine (rendering most analysis algorithms undecidable).

From the discussion above, we conclude that x-schemas can be encoded as Petri nets.

■

From the discussion above we conclude that x-schemas are equivalent to ordinary Petri nets without stochastic transitions.

3.10.1 The Stochastic Case

The continuous time stochastic Petri net was introduced by Molloy (Molloy 1982). As described above, the firing time is governed by an exponentially distributed random variable x_i . The firing time of transition t_i is given by

$$\mathcal{F}_{x_i} = 1 - e^{-\lambda_i x} \tag{3.10}$$

The negative exponential distribution renders the reachability graph of the SPN isomorphic to a continuous time Markov chain. The Markov chain MC can be obtained from the reachability graph as follows: The MC state space is the reachability set $R(PN)$ of the marked SPN . In MC , the transition rate from M_i to M_j is given by $q_{ij} = \lambda_k$, corresponding to the firing rate of the transition t_k from M_i to M_j . If several transitions lead from M_i to M_j , then q_{ij} is the sum of the rates of these transitions. If there is no link from M_i to M_j in $R(PN)$ then $q_{ij} = 0$ in MC .

The steady state distribution π of the MC is obtained by solving the linear equations:

$$\pi Q = 0 \tag{3.11}$$

$$\sum_{1=1}^{s} \pi_j = 1 \tag{3.12}$$

From the vector $\pi = (\pi_1, \ldots, \pi_s)$ we can compute the following measures:

- **Probability of being in a set of states**: Let $\mathcal{B} \subseteq R(PN)$ constitute the states of interest in a given SPN . Then the probability of being in a state of the corresponding SPN is

$$P(\mathcal{B}) = \sum_{M_i \in \mathcal{B}} \pi_i \tag{3.13}$$

- **Probability of taking a transition** t_j : Let EN_j be the subset of R(PN) in which the transition t_j is enabled. Then the probability that an observer looking randomly into the next sees t_j firing next (p_j) is given by

$$p_j = \sum_{M_i \in EN_j} \pi_i \frac{\lambda_j}{-q_{ii}} \tag{3.14}$$

where q_{ii} is the sum of the transition rates out of M_i .

- The throughput of a transition is the mean number of firings at steady state

$$d_j = \sum_{M_i \in EN_j} \pi_i \lambda_j \tag{3.15}$$

Given the definitions of $SPN'S$, we define a colored SPN as follows.

Definition 14 $HLSPN$
A $HLSPN$ is a $3-tuple$ HLSPN = (HLPN, T, W) *where*

$HLPN = (P, T, C, I, O, M_0)$ is the underlying marked high-level Petri net.

T is a set of timed transitions.

$W = (w_1 \ldots w_{|T|})(w_i \in \mathcal{R})$ is an array whose entries are negative exponential distributions specifying the firing delay. This rate could be marking dependent.

Definition 15 $HLGSPN$
A $HLGSPN$ is a $4-tuple$ $HLGSPN = (HLPN, T_1, T_2, W)$ where

- $HLPN = (P, T, C, I, O, M_0)$ is the underlying marked High-Level Petri net.
- $T_1 \subseteq T$ is a set of timed transitions $T_1 \neq 0$.
- $T_2 \subset T$ is a set of immediate transitions.
- $T_1 \bigcup T_2 = T$ and $T_1 \bigcap T_2 = 0$.
- $W = (w_1 \ldots w_{|T|})(w_i \in \mathcal{R})$ is an array whose entries
  1. Could be a negative exponential distribution specifying the firing delay when $t \in T_1$. For such a timed transition, the rate could be marking dependent.
  2. Is a firing weight for the immediate transition t, when $t \in T_2$. This weight is also possibly marking dependent.

If $T_2 = 0$, then the $GSPN$ defined above essentially reduces to an SPN. Hence $SPN \subset GSPN$. The presence of immediate transitions makes the reachability graph of a marked $GSPN$ non-markovian since immediate transitions fire in zero time. Markings where only immediate transitions are enabled are called **vanishing** since an external observer will never see these states, even though the stochastic process sometimes visits them. For timed transitions, the stochastic process sojourns for an exponentially distributed amount of time, and states resulting from markings enabling timed transitions are likely to be seen by an external observer. These states or markings are called **tangible** markings.

Lemma 3 . *HLGSPN* **to** *GSPN*

High-Level Generalized Stochastic Petri Nets (HLGSPN) can be converted to Generalized Stochastic Petri Nets (GSPN).

> **Proof (sketch):** The proof is exactly similar to the non-stochastic case, since the typing of tokens does not depend on the types of transitions. ■

Lemma 4 . **X-schema to GSPN** *X-schemas with stochastic, hierarchical, instantaneous and durative transitions are isomorphic to GSPNs.*

> **Proof (sketch):** The proof is simple, since the stochastic and immediate transitions correspond to the *GSPN* transitions. As was shown in Figure 3.5, we can model hierarchical transitions using a pair of instantaneous transtions (with perhaps a stochastic timout transition). ■

From Lemma 2 and Lemma 4 we conclude that x-schemas can be encoded as Generalized Stochastic Petri Nets (*GSPN*), thereby proving Theorem 1.

Chapter 4

Connectionist Inference and X-schemas

How might x-schemas be neurally encoded? This chapter answers the question from two different directions, with some rather surprising results. We first look at a connectionist encoding of x-schemas. Here we find that primitives developed for logical reasoning by Shastri and his students as part of the SHRUTI connectionist reasoning system are sufficient to model the various parameters and control behaviors required for higher level motor control. Section 4.1 presents this reduction and is joint work with Lokendra Shastri, Dean Grannes and Jerry Feldman. We then look at some general connectionist primitives and find that they can be readily and naturally encoded as x-schemas. Since these primitives form the backbone of a connectionist general purpose reasoning system, x-schemas seem to capture some of the basic inferential primitives required for general-purpose reflex reasoning. All this suggests a rather close connection between sensory-motor control representations and general purpose reasoning. A more practical by-product is of course that x-schema based models can be used for computer simulation and analysis of complex connectionist systems. But this remains future work.

4.1 Connectionist Encoding of X-schemas

This section reports joint work with Lokendra Shastri, Dean Grannes, and Jerry Feldman. Detailed information including a SHRUTI based implementation of x-schemas is described in (Shastri *et al.* 1997). Details can also be found from http://www.icsi.berkeley.edu/˜grannes/shrutihome.html.

We describe the encoding with the help of an example from our familiar notation for x-schemas developed in Chapter 3. Figure 4.1 shows the implemented PUSH x-schema in the formalism of Chapter 3.

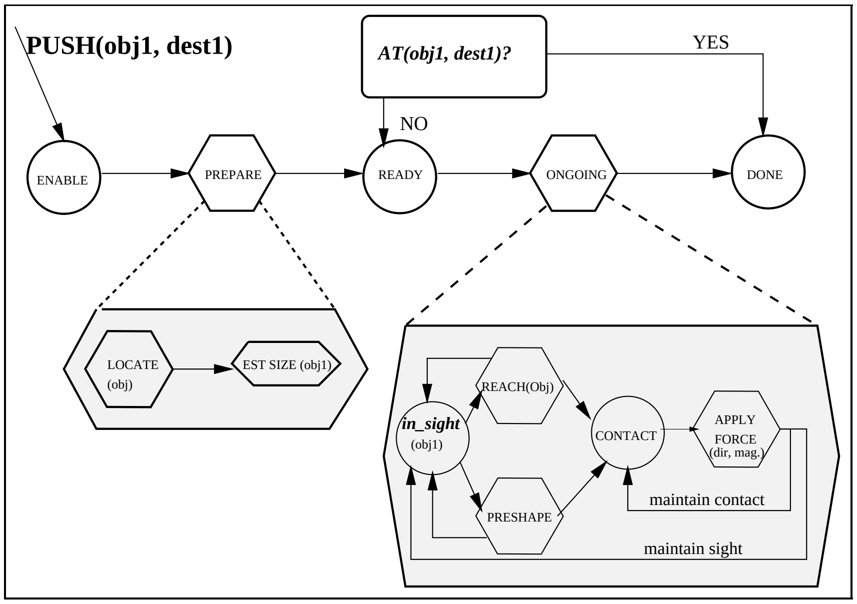


Figure 4.1: The Push x-schema in the formalism developed in Chapter 3.

The PUSH x-schema takes parameters corresponding to the object to be pushed (labeled obj_1 in Figure 4.1) and the destination where the object needs to be pushed to (optional). As usual, the hexagonal nodes show decomposition into sub-schemas (shown as shaded hexagons). For instance, the preparatory steps involve locating the object and estimating its size and mass. This will determine the amount of force as well as the kind of posture required. Once this is done the object is located and the agent is *ready* to push the object. Pushing involves reaching for the object while concurrently preshaping the palm/hand for the push motion (we assume that the object is kept track off during these operations by vision). Once contact is made with the right palm posture, force can be applied in the direction computed from the destination or directly specified as a parameter (as in "push left"). When the object is at the desired destination, the pushing is *done* . We use this x-schema as a representative example to illustrate our connectionist encoding.

Our connectionist encoding of x-schemas involves a number of ideas. One key is the expression and propagation of dynamic bindings via temporal synchrony. Another is the use of *focal clusters* and feedback loops to control a distributed process without a central controller. A third important feature is the uniform mechanism for interaction between schemas, low level somatosensory and proprioceptive processes, and high-level reasoning and memory processes. Many of these mechanisms have been drawn from SHRUTI (Shastri & Ajjanagadde 1993).

The network structure enclosed within the dotted hexagon in Figure 4.2 depicts the partial connectionist encoding of the schema shown in Figure 4.1. This schema is simplified to keep the exposition simple and focus on illustrating how the encoding addresses the computational requirements discussed in Chapter 3.

The PUSH schema is an interconnected network of focal clusters (depicted as ovals). Each focal cluster embodies a locus of control and coordination as well as a mechanism for exchanging information between schema components. These components can be specific to the schema or can be generic. For example, the PUSH schema invokes generic schemas such as LOCATE-OBJECT, ESTIMATE-MASS, and MOVE-HAND (depicted as hexagons outside the PUSH schema) that find the location of an object matching a given description, estimate the size of an object at a given location, and move the hand to a specified location, respectively. The issue of how the connections between the appropriate focal clusters are formed during the learning of the relevant behavior is not addressed in this thesis.

Other schemas can invoke the PUSH schema by activating the ENABLE cluster near the bottom of and communicating the appropriate parameter values and role bindings to this cluster (we will see how this is done, shortly). The first step in the schema execution involves locating the object to be pushed using the LOCATE-OBJECT schema (invoked by the LO cluster). We will explain the control in detail below, but notice that activation can flow from the ? node in the ENABLE cluster to the ? node in the LO cluster and then to the LOCATE-OBJECT schema. The completion of the LOCATE-OBJECTschema is followed by the invocation of the ESTIMATE-MASS schema (invoked by the EM cluster) for estimating the mass of the object to be pushed. At the end of this step, the agent is ready to carry out the requisite motor action. This would be indicated by the activation of the READY cluster.

The next stage of execution consists of two concurrent steps: reaching for the object and orienting the hand in a manner appropriate for pushing it. These steps are

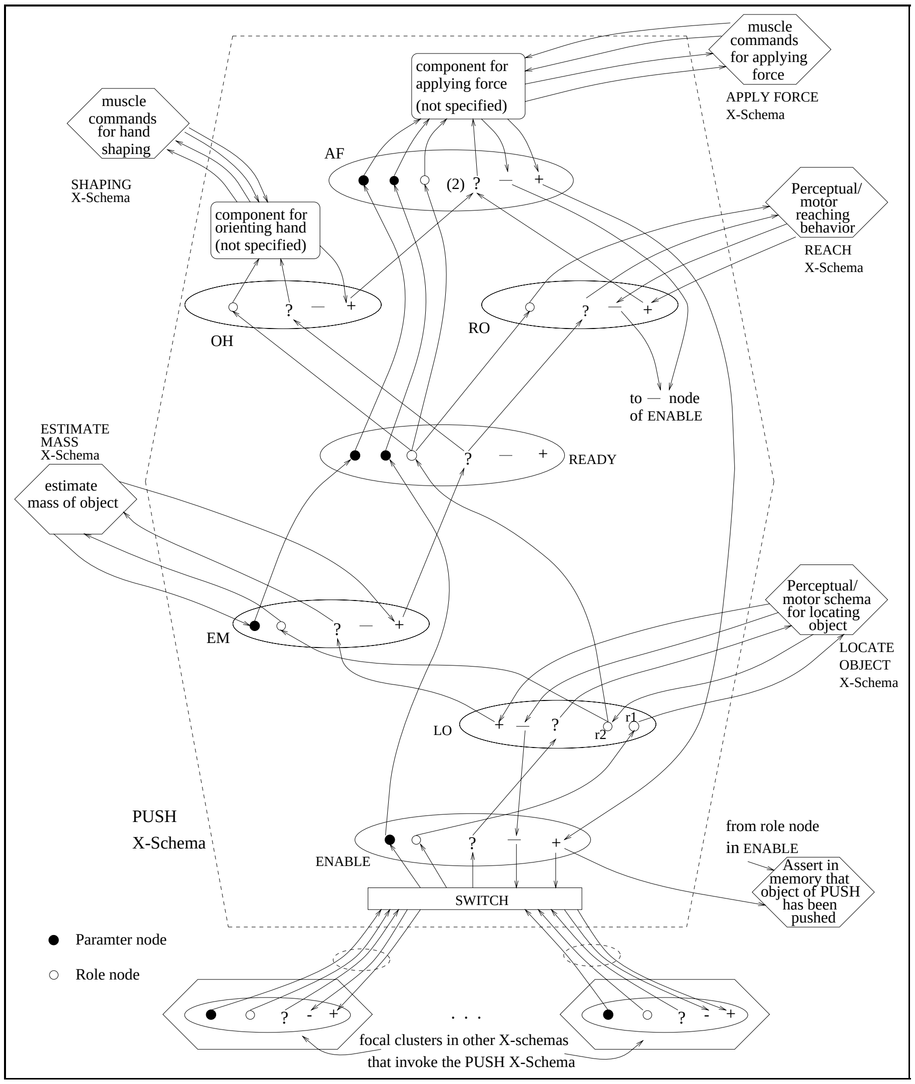

muscle commands for applying force
APPLY FORCE X-Schema
component for applying force (not specified)
muscle commands for hand shaping
SHAPING X-Schema
AF
(2) ?
component for orienting hand (not specified)
Perceptual/ motor reaching behavior
REACH X-Schema
OH
RO
to — node of ENABLE
ESTIMATE MASS X-Schema
estimate mass of object
READY
Perceptual/ motor schema for locating object
LOCATE OBJECT X-Schema
EM
LO
r1
r2
PUSH X-Schema
ENABLE
from role node in ENABLE
Assert in memory that object of PUSH has been pushed
SWITCH
● Paramter node
○ Role node
focal clusters in other X-schemas that invoke the PUSH X-Schema

Figure 4.2: A Connectionist version of the PUSH x-schema

executed by the REACH schema and a (not shown) component that eventually invokes the appropriate low level motor commands to orient the hand. Once the hand is properly oriented *and* in place to apply the force, the appropriate motor commands for applying force are executed.

4.1.1 Focal Clusters

The focal clusters (depicted by ovals in Figure 4.2) represent the core functionality underlying our model and implementation. Each focal cluster consists of a small number (possibly zero) of parameter nodes (depicted as filled circles), a small number (possibly zero) of role nodes (depicted as open circles), and a small number of control nodes depicted as ?, +, and − in Figure 4.2. [1] The structure and representational significance of focal clusters parallels that of a relation focal cluster used in SHRUTI (Shastri & Grannes 1996) and may relate to the biological notion of "convergence zones" (Damasio, 1989).

The function of the various nodes in a focal cluster is as follows: the *parameter* nodes set values for continuous parameters in the invoked schema. In the PUSH x-schema, the firing rate of the single parameter node specifies the relative force to be applied during the push action. Parameter nodes can also encode discrete parameter values such as "direction = left". In this case, the supra-threshold firing of the parameter node would indicate that the action is to be performed in the left direction.

Role nodes provide the mechanism for dynamically binding a schema-role with a filler at the time of schema invocation. The PUSH x-schema has one role denoting the intended object of the push action(we leave out the *dest* role to keep the exposition simple). A dynamic binding between a role and its filler is expressed in the system by the synchronous firing of the role node and the focal node representing the filler of this role. Such bindings are propagated through the role nodes by the synchronization of the activity of connected role nodes. In Figure 2, the role node of the ENABLE cluster is synchronized with the r1 role of the LO (locate) cluster and this in turn is synchronized with the LOCATE OBJECT schema to the right of the PUSH x-schema. The location of the object is returned as a binding for the r2 role in the LO cluster and this is propagated to the READY cluster, etc. The idea of encoding and propagating dynamic bindings among roles using synchronization

[1] In more detailed modeling, other control nodes can occur. Also, as explained in (Shastri & Dean 1996), each node in the cluster is can be thought of as a small cluster of cells.

was introduced in the SHRUTI inference engine (Shastri & Ajjanagadde 1993) and has since been adopted to a wide range of problems (Hummel & Biederman 1993; Valiant 1996). The fact that the same dynamic binding mechanisms can also apply to active schemas is potentially of great importance. Neurophysiological evidence suggests that such propagation of synchronous activity is neurally plausible (Singer 1993) and our latest results suggest additional functions of the brain where it might be employed.

The ?, +, and − nodes in each cluster serve control and coordination functions. The ? node can be viewed as an *initiate* or *query* node. A process initiates an action (or poses a query for information) by activating the ? node of the appropriate focal cluster. Each ? node has a threshold (default = 1) which determines the number of active inputs it must receive in order to fire. The + and − nodes indicate the outcome of schema execution. The activation of the + (−) node of a focal cluster by a schema indicates the successful (unsuccessful) *completion* of the schema. This provides the conditional computation required for complex schemas and is also used for error recovery. Observe that recurrent connections are a central aspect of schemas. In particular, every cluster that initiates a schema also receives a signal when the schema is completed.

Each x-schema has a head focal cluster – the ENABLE cluster – that serves as the point of initiation. The ENABLE cluster of a shared schema has a *switch* which controls the flow of signals into the schema. All schemas that invoke this shared schema have their appropriate focal clusters connected to this switch (see bottom of Figure 4.2). We assume that these inter-schema connections are learned as part of learning an automatic behavior or a reactive plan.[2] The switch behaves like a bidirectional k to 1 switch, where k is the number of external schemas that are linked to this one. It ensures that at any given time, signals from at most one schema can propagate into the ENABLE cluster of the (invoked) schema. This switch state is maintained until the invoked schema finishes execution (which is signaled by the activation of the + or − node in ENABLE).[3] If several schemas try to activate one schema simultaneously, the switching mechanism selects signals from one of these schemas. The switch also channels the output of the + and − nodes from ENABLE to the appropriate links feeding back to the invoking schema.

Let us walk through the PUSH schema (refer to Figure 4.2). Imagine that our

[2]Connections between schemas have to be created dynamically during deliberative (reflective) planning. We are not concerned with such reflective processes in this paper.

[3]Preemption by a high priority schema is possible but not discussed here.

agent has the goal of vigorously pushing a red wagon. This leads to one of the x-schemas connected to the PUSH schema invoking it by activating the ? node in its ENABLE cluster, activating the parameter node in the cluster with a high firing rate (to indicate a high force value), and synchronizing the firing of the role node in the cluster with the appropriate role node of the invoking schema. Since the object of the action is the "red wagon", the this role would be firing in synchrony with the focal nodes of the concepts "red" and "wagon" and thereby be dynamically bound to the object description "red wagon". As a result, the PUSH x-schema is enabled with its force parameter set to high and its object role bound to the description "red wagon".

The activity in ENABLE of Figure 4.2 leads to the activation of the focal cluster LO wherein the ? node becomes active and the role *r1* synchronizes with the role in ENABLE. Consequently, the role *r1* starts firing in synchrony with "red wagon" and hence, gets bound to that description. LO in turn invokes the LOCATE-OBJECT x-schema with the binding "red wagon". Upon successful execution, LOCATE-OBJECT activates the + node in LO and binds the role *r2* with the location where the object matching the description "red wagon" is located. This binding is expressed by the role node firing in synchrony with the appropriate "location" node in an egocentric spatial map. The activation of the + node in LO leads to the activation of the cluster EM with its role bound to the location returned by LO . EM estimates the mass of "red wagon" nd upon completion activates the + node of LO and fires the parameter node of LO with a rate indicative of the mass of the "red wagon".

The activation of the + and the parameter nodes in EM leads to the activation of the ? node in the READY cluster and the setting of the parameter representing the estimated mass of the object to be pushed. In parallel, the activity of the second role of READY is set by the parameter node in the ENABLE cluster to indicate the desired relative force, and the role of READY is bound to the location of "red wagon" by the role *r2* in the LO cluster as described earlier.

The activation of READY marks a significant state in the execution of the PUSH. At this point the agent has carried out the necessary preparatory steps but has not executed any actions that affect the environment. Thus READY provides a locus for high-level executive processes to exercise inhibitory control over the actual execution of the PUSH. It turns out that these stages are extremely important for language understanding and as we will see the use of such important control stages in our model of linguistic aspect, detailed in Chapter 5.

READY activates clusters OH and RO concurrently and binds their roles to the location of "red wagon". This leads to the concurrent activation of (i) the REACH x-Schema that brings the agent's hand to the location of "red wagon" and (ii) the component that shapes and orients the hand in a manner appropriate for carrying out the push action. Since this component receives a binding specifying the location of the object to be pushed, it can extract the shape of this object and determine the appropriate shaping of the hand.

REACH and the hand orienting components execute independently and signal their completion (or failure) by activating the + (−) node in OH and RO respectively. The successful termination of these two components leads to the activation of the AF cluster. Since the ? node of AF has a threshold of 2, it becomes active only upon receiving a completion signal from both OH and RO. By this time AF also has its parameters set with the appropriate values of estimated mass and desired relative force and its role bound to the location of "red wagon". Now the final component for applying the force is initiated which initiates low-level motor schemas for issuing muscle commands to apply force on the "red wagon". Once this component executes successfully it activates the + node of the AF cluster which activates the + node of ENABLE and signals the successful completion of PUSH . Finally the activity of the + node in the enable cluster is propagated via the switch to the + node of the appropriate focal cluster in the schema that invoked PUSH .

In addition to successful completion, failure is also communicated within the schema. For example, the failure to locate the object, reach the object, or apply force to the object can lead to the activation of the − nodes in LO, RO, and AF, respectively. This would lead to the activation of the − node in ENABLE and signal a failure of PUSH . Alternately, the failure of a step can trigger alternate plans or remedial strategies (not shown). For example, the failure of REACHcan trigger schemas for walking around an obstacle or using an implement to extend the agent's reach.

The completion of a schema can also trigger an update of the agents memory. Thus the completion of PUSH can lead to the agent's episodic memory recording the fact the agent has pushed the "red wagon". It can also lead to the remembering of the position of the "red wagon" at the end of PUSH . In the proposal, a memory lookup is treated as being similar to a perceptual "look up". Similarly, the updating of episodic or working memory is treated as being similar to an external motor action.

The encoding of x-schemas described above has been implemented in SHRUTI (by

L.Shastri and Dean Grannes) and tested on examples such as the PUSH schema shown in Figure 4.2. Details of the model and the implementation can be obtained by pointing your web browser to "http://www.icsi.berkeley.edu/˜grannes/shrutihome.html/".

From the discussion above, we conclude that general purpose structured connectionist binding and inference systems such as SHRUTI can model x-schemas or parameterized action controllers.

4.2 X-schema encoding of general-purpose structured connectionist primitives

As mentioned in Chapter 2, neuron outputs are a spatio-temporal threshold function of their inputs. The popular *linear threshold* and *sigmoid* functions used in neural network implementations only perform spatial summation. For instance the McCulloch-Pitts Neuron performs spatial summation of the input signal.

$$y = f(w^t x - \theta) \tag{4.1}$$

Here, w and x are the weight and input vectors respectively. f is the *step* function and θ is a threshold value. The output y is binary. The weighted sum is represented as an inner product between the weight vector w and the input x .

While it is easy to construct x-schemas to model the $\sum$ and π functions that perform purely spatial thresholding, we believe that that the fact that a biological neuron may perform both spatial and temporal summation over its inputs is crucially important in modeling cognitive functions. This property is best exploited in structured connectionist models such as SHRUTI (Shastri & Ajjanagadde 1993). Therefore, we require our x-schemas to be able to model these more complex unit functions.

To explore the power of x-schemas to model such connectionist primitives, we will look at the set of primitives common to most models of structured connectionism. In particular, the primitives chosen correspond to the most advanced model of structured connectionist networks, namely SHRUTI. SHRUTI is a general purpose structured connectionist reasoning system that is capable of modeling a class of deductive inferences in a connectionist framework. The central idea exploited by SHRUTI is the detection and usage of dynamic

synchrony in making and propagating **run time** bindings. In this section, we show how the x-schema model is able to model the set of basic primitives used. We start off with the x-schema implementation of a key node that appears in many connectionist models called the 2/3 binder node. We then go on to the more complex spatio-temporal nodes such as the τ-and and ρ-btu nodes used in SHRUTI.

4.2.1 Implementing Binder units as x-schemas

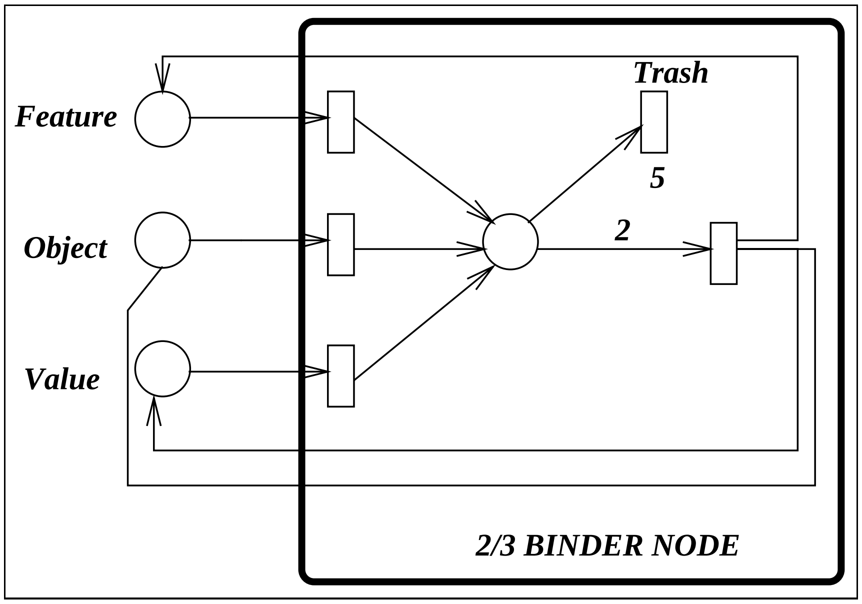


Figure 4.3: Modeling a **2/3 binder node** as an x-schema

The 2/3 binder node is a key node that has been used in many structured connectionist models (Feldman and Shastri 1984; Grossberg 1985). The basic idea is to bind $<$ *object, attribute, value* $>$ triplets in long-term memory. Consider the object *apple*. We would like to store that the attribute *color* of the apple is *red*, the attribute *shape* of the apple is *round*, etc. The crucial fact is that we would like to be able to retrieve the third feature given the other two. For instance, given apple (object) and color (attribute), we would like to activate red (value). Similarly, given the object (apple) and value (red), we would like to activate the relation color (attribute). This is done through the use of a binder node, which activates the third of its inputs, given the other two. The reader may already have noted that *binder* nodes are a connectionist implementation of the feature structures or “f–structs” described in Chapter 3. Recall that f–structs supply parameters and get updated as a result of x-schema executions. To specify a scheme where there is mutual exclusivity between different values in a multi-valued “f–struct” use a winner-take-all network that can be found in detail in (Feldman & Ballard 1982; Feldman & Shastri 1984).

Figure 4.3 shows the x-schema model of a **2/3 binder node**. The model is fairly straightforward. The basic operation is that if two of the inputs to the node become active (are marked), the node becomes active and marks all (three) of its inputs. We need the node labeled TRASH to flush out excess tokens since the structure is always active. In the current design, this happens every-time 5 tokens accumulate, but the parameter values can vary.

4.2.2 Implementing τ-and units as x-schemas

In SHRUTI, a τ-and node becomes active after receiving an uninterrupted pulse train, where there is a pulse at every time step. On becoming active, the τ-and node produces an output pulse that mimics the input pulse by outputting a pulse at every phase of the next cycle. Thus the τ-and node is like the temporal and node. Figure 4.4 shows how this functionality can be achieved with an x-schema based model.

In Figure 4.4, we see that only the presence of activity at every phase can trigger the node for the next cycle. Once active, the τ-and node is able to deliver activation for every phase of the next cycle, thus achieving the desired functionality. The circuit acts as a ripple counter with delay 1. Thus at the end of a cycle, in order for the instantaneous transition (shown darkened, as usual) there should be a token in each of the input places which would happen only if there was activation in each phase. If this is true, the network becomes *active* with n (where n is number of phases) tokens at the active place. These tokens are drained one at a time by the transition labeled $Output$ with delay 1. There is thus one token emitted per phase of the next cycle, achieving the τ-and functionality.

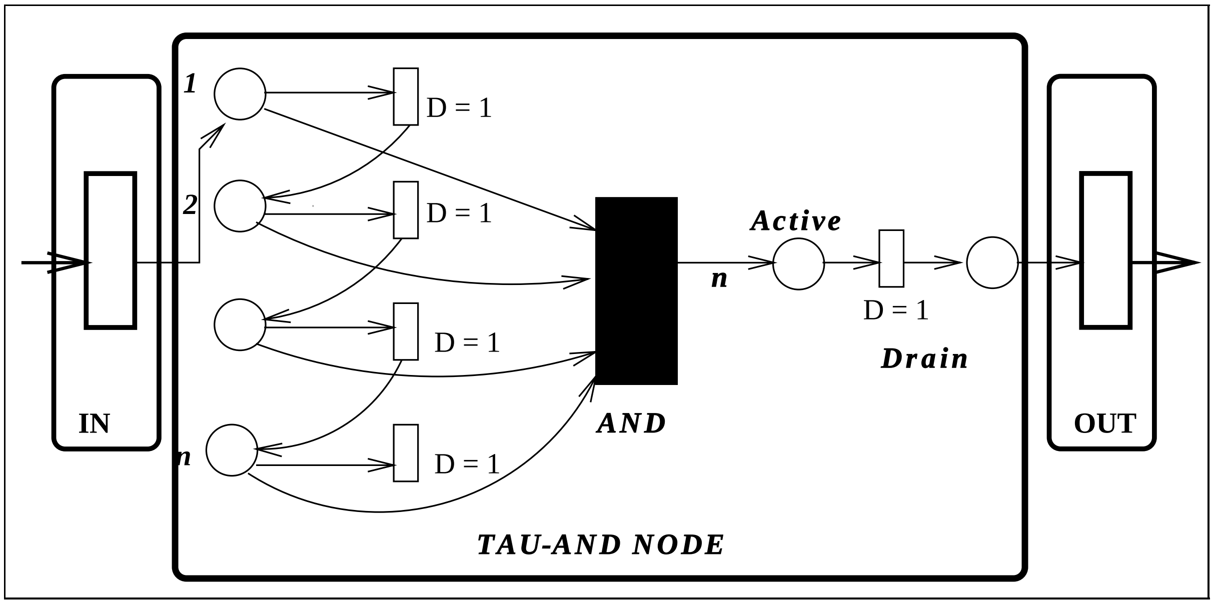


Figure 4.4: Modeling a τ-AND node as an X-schema

4.2.3 Implementing ρ-btu nodes as x-schemas

In SHRUTI, arguments to predicates and concepts are encoded using ρ-btu nodes. The basic function of these nodes is to propagate synchronous activity to the connected node in a manner that the phase of the incoming activity is maintained. So if a node A is connected to another node B, then if node A fires with a period p, then node B will also fire **in-phase** with the same period p. Figure 4.5 shows the x-schema encoding of the ρ-btu node.

In Figure 4.5, the input and output are transitions. The firing of the input transition marks the input place of the ρ-btu unit. This fires the instantaneous transition **iff** it has not already been fired in the current clock cycle. If it has been fired in the current clock cycle, it becomes disabled until the next cycle due to the inhibitory input from the place labeled *inhib*. The *inhib* place becomes marked with the first token recd. per cycle, and gets drained at the beginning of every cycle. The draining operation is accomplished by the CLK_RESET hexagon. We could make do with just a transition firing once at the beginning of each cycle consuming the token at the *inhib* place, but there has to be a more elaborate local clock circuit which keeps track of the phase that could double for this

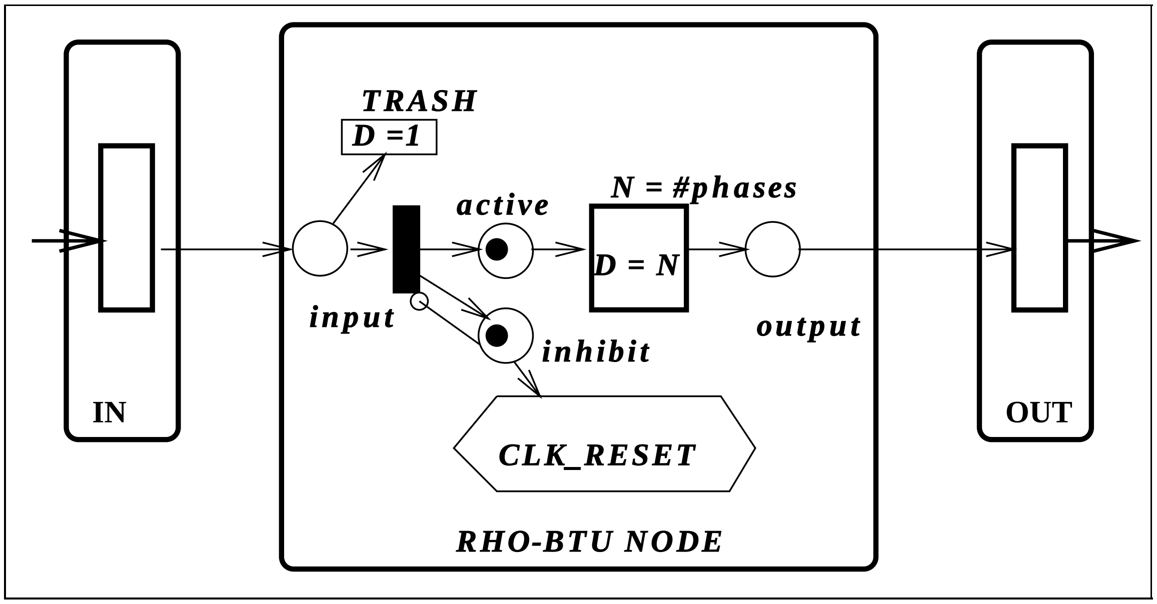


Figure 4.5: Modeling a ρ-btu node as an x-schema

purpose. Hence the hexagon. If multiple tokens appear within a single cycle, the first one is activated, others are discarded in one time step to **trash**. The phase at which the **active** place gets marked is the phase at which the output transition will fire in the next cycle. The durative transition labeled $D = N$ has a delay that fires marks the place labeled *output* at the same phase in the next transition. Thus if the input is periodic, the output will fire with the same period. If the input is noisy and non-synchronized, the output will be noisy as well.

Using n ρ-btu nodes, and two τ-and nodes is sufficient the SHRUTI system is able to encode a predicate with n arguments. A long term fact is encoded in SHRUTI using a τ-and node with inhibitory links from the argument nodes and the appropriate entities. Thus with these nodes, the basic functionality of SHRUTI can be achieved.

From the discussion above, we conclude that x-schemas are able to model general purpose structured connectionist reasoning systems.

4.3 Discussion

The two different parts of the system modeled above suggest that a close and deep connection between high-level sensory motor control and general purpose inference. In this context, it is interesting that the same mechanisms seem sufficient for reflexive inference, perceptual motor schemas and the modeling of aspect. Actually this is not surprising if one takes seriously the notion that language and conceptual structures are grounded in our body and shaped by our motor and perceptual systems. In this thesis, we will see how x-schema like models are useful of modeling linguistic aspect (Chapter 5) and for metaphoric inference (Chapter 8). The use of such representations for hand-action verb acquisition is explored in (Bailey 1997).

Chapter 5

A Computational Model of Verbal Aspect

The study of aspect pertains to the study of linguistic devices that enable a speaker to direct the hearer's attention to the internal temporal character of a situation. Aspects differ from tenses in that whereas tenses deal with the temporal relations between situations (such as past, present and future), aspects enable focus on the compositional attributes of a situation. In this chapter, we describe a model of aspect that is based on the x-schema-based representation of actions and events outlined in Chapter 3. The model described does not currently handle tense.

Many languages have grammatical aspectual modifiers such as the English *progressive* construction (*be* + *V-ing*) which enable a speaker to focus on the ongoing nature of an underlying process while allowing for inferences that the process has started and that it has not yet completed. Similarly, one use of the *perfect* construction (*has V-ed*) allows a speaker to specify that some *consequences* of the described situation hold. Languages also have a variety of other means to express aspect including aspectual verbs like *start*, *end*, *cease*, *continue*, and *stop* and related grammatical forms.

The frequency with which languages refer to events and the universality of such expressions has made aspect an object of study since Aristotle. Aspect is a particularly interesting phenomenon because in all languages studied, the *natural or inherent* verb phrase semantics combines with and modifies the interpretations and entailments of the grammatical marker system. This makes a compositional account of the semantics of aspect difficult, giving rise to many paradoxes and problems (Dowty 1979; Moens & Steedman 1988). This chapter demonstrates that a compositional semantics of aspect can be constructed if we take the embodiment of action in a neural system seriously.

5.1 Basic Result

This chapter describes a computer simulation that provides evidence to support the following proposition.

Proposition 1 . *Aspectual expressions are linguistic devices referring to* **schematized** *generalizations that recur in process monitoring and control (such as inception, interruption termination, iteration, enabling, completion, force, and effort).*

In Chapter 3, we described a computational model of actions called x-schemas which was then used to model the semantics of action verb phrases. The model was inspired by knowledge of what is known about the cortical representation of high-level motor control and satisfied general computational constraints on modeling neural activity. In the implemented model, the semantics of individual verbs are *active* and primarily involve salient features of behavior control. In this chapter, we show that the process primitives developed in Chapter 3 are sufficient to model the inherent aspect (the aspect that is inherent in the basic meaning of the verb-complex) of abstract events and actions. Furthermore, inherent aspect naturally falls out of the x-schema representation of verbal semantics.

We refine our model of verb phrase semantics further by proposing a particular active x-schema called the CONTROLLER, that captures a control generalization over many individual x-schemas. The controller is itself an x-schema and models important regularities that are relevant in the evolution of processes (enabling, inception, in-process, completion, suspension, resumption, etc.). Throughout the rest of the thesis, we provide evidence that the controller abstraction seems to capture the basic temporal structure of our conceptualization of events.

The semantics of aspect arise from the bi-directional interaction of the generalized controller with the specific underlying x-schema for the verb phrase in question. This active model of aspect grounded in sensory-motor primitives is able to model cross-linguistic variation in aspectual expressions while avoiding paradoxes and problems in model-theoretic and other traditional accounts.

We begin by briefly reviewing the linguistic theory on aspect. After describing the relevant action parameters and the controller abstraction, we describe our model of aspect and the results obtained. The chapter concludes by describing some more complicated cases of aspect which involve tense-aspect combinations that are outside the scope of the current model, and some initial steps extending the model to model tense-aspect interactions.

5.2 A Brief Note on Linguistic Theory

The subject is vast and I will only be touching on a small set of work directly relevant to this thesis. The interested reader is referred to (Vendler 1967; Comrie 1976; Dowty 1979) for foundational work including a *Montague Semantics* for aspect; to (Moens & Steedman 1988; Nakhimovsky 1988; Steedman 1995; Steedman 1996) for a computational linguistics perspective that uses an influential *tripartite* process logic model, to (Bennett 1986; Dorr 1993) for applications to machine translation and cross-linguistic issues; to (Ter-Muelen 1995) for an innovative linguistic model that bears some resemblances to the work reported here; to (Michaelis 1992) for a construction-grammar approach to aspect and for arguments about the fundamental nature of the event/state distinction, (Langacker 1987) for the Cognitive Grammar approach, and to (Berman & Slobin 1994) for a developmental and cross-linguistic perspective.

The fundamental problem in formulating a semantics of aspect lies in the interaction of the **inherent** aspectual class of situations (called Aktionsart) with the different linguistic devices that allow the speaker to adopt different **viewpoints** with respect to a situation or refer to different **phases** of a situation.

A case of adopting different viewpoints on a situation comes from the possibility of viewing a situation from the inside (without focus on the endpoints) versus from the outside (where the focus is exclusively on the endpoints as opposed to the situation's internal temporal structure). Such a difference is coded in many languages as the imperfective/perfective distinction. This distinction is often referred to as the **viewpoint** aspect.

Perfective aspect is a viewpoint that refers to a situation as a complete whole and is defined by its *lack* of reference to the internal temporal details of a situation. Russian has special affixes (such as *pos*) to denote the Perfective aspect. English does not have a special construction to mark the Perfective, and the Perfective/Simple Past distinction in English is context dependent. (Langacker 1987) defines the perfective aspect as a *focus* on the end-points of the described activity. In contrast, the **Imperfective aspect** refers to a the internal state of a situation; namely viewing a situation from within. Here in the focus is *not* on the end-points but on the internal structure of a process. The term viewpoint is meant to connote that what is important is the speaker's subjective position with respect to the described situation.

Linguistic devices can also refer to the internal temporal details of a single situation by focusing on different logical stages **phases** of a situation. Such devices constitute the **Phasal** aspect of an utterance.

Example 5 John is playing tennis. ■

For instance, Example 5 is an instance of the **Progressive** aspect (grammaticalized in English by the *be + V-ing* construction), and refers to the *ongoing* process of playing tennis (a *phase* of the *play_tennis* situation).

In contrast, Example 6 is one case of the **Perfect** aspect (grammaticalized in English by the *has + V-ed construction*), indicates that some *consequences* of the described situation hold at the time of description. Other cases are described in Section 5.6.1.

Example 6 John has lost his keys. ■

without goals (*energea*) or goal-directed activities (*kinesis*). Vendler and others extended the basic classification, and most modern systems are based on some variant of the *VDT* (Vendler-Dowty-Taylor) system. We discuss this system in Section 5.4.3. (Talmy 1985) suggests a finer grained distinction that comes closer to our model. Classifying inherent aspect as a distribution of action over time, he notes that many verb-complexes inherently entail specific action patterns. For instance, *walking*, *tapping* and *breathing* are inherently periodic activities, while *shove* or *hit* are not. On the other hand *dying* or *falling* are inherently aperiodic.[1]

[1]Prof. Fillmore points out the contradiction between this assertion and my Hindu background.

One central question in the analysis of aspect is how these three subsystems (the **viewpoint**, **phasal** and **inherent aspect**) interact in interpretation. In other words, given that different parts of a sentence or different linguistic devices may code for different subsets of the features above, how does one construct an interpretation of the entire expression in *real-time*?

5.3 Relevant Process Parameters

Recall from Chapter 3, Figure 3.5 described some of the common control patterns that can be modeled by x-schemas and Figure 3.6 showed other important x-schema parameters and patterns useful to describe the semantics of motion and manipulation verbs. We will now argue that some of the same features and primitives are useful in moving toward a compositional semantics of aspect. In particular, we will show how the same parameters can form a fine-grained description of **inherent** aspect that is a) motivated from sensory-motor considerations, b) can easily derive other schemes, and c) allows us to make finer distinctions that are often metaphorically projected.

In chapter 3 we argued that in order to monitor and control the execution of motor programs and their effects in a dynamic environment, x-schemas need to control both *periodic* and *aperiodic* actions with real *durations* (to monitor, timeout, initiate error-recovery procedures) and to be able to check and monitor *conditions* and *resource* usage. These include monitoring resource consumption (energy level), as well as respecting mutual exclusion constraints (can't hold two blocks at the same time)), and *enabling* and *disabling* conditions (can't walk if the ground is slippery) They should be capable of *goal* based enabling and should be able to monitor and remain active until achievement of the goal. Together, we referred to these *abstracted* primitives (duration, periodicity, resources, goals and conditions) as the **process primitives**. Section 5.4.1 describes in detail the connection between our x-schema model of the process primitives developed for grounding the semantics of embodied verb phrases and their use in modeling the inherent aspect of general events and actions. But before that we introduce the other crucial piece for modeling aspect, namely the CONTROLLER x-schema.

5.3.1 The Schema controller

The CONTROLLER is a control generalization over different x-schemas. Thus whether our simulated robot controller controls the execution of a PUSH or WALK schema, it monitors the *enabling*, *inception*, or *ongoing* and *termination* of the motor program. It should also be capable of coordinating sensory and motor inputs to a state of *readiness* as well as be able to know about the successful *completion* of a program.

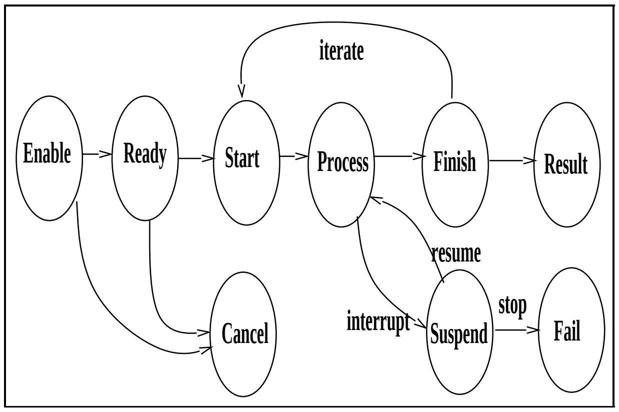


Figure 5.1: The Schema controller is a generic structure that captures relevant features of behavior control.

Crucially, the generic controller is itself an active structure or x-schema. The controller sends signals to individual motor schemas and may transition based on signals from the underlying schema. Thus nodes in the controller bind to process states in the underlying x-schema. Directed arcs constrain behavior execution trajectories. The controller is *stateful* and the control graph encodes possible process evolution trajectories. Thus if the ongoing node of the controller is *active*, the activity in question has already started and that the next interesting transition is to the termination, suspension, or iteration of the underlying activity. While it is quite easy to argue that the CONTROLLER abstraction is a useful one for process control and planning, [2] it provides an elegant and motivated basis to ground aspectual interpretation.

Figure 5.2 shows the x-schema version of the CONTROLLER, that corresponds to the depiction in Figure 5.3. Note that the hexagons indicating different transitions indicate

[2]In fact one of the reviewers of (Narayanan 1997) informed us that the controller was very similar to structure used in Behavior-based Mobile Robotics.

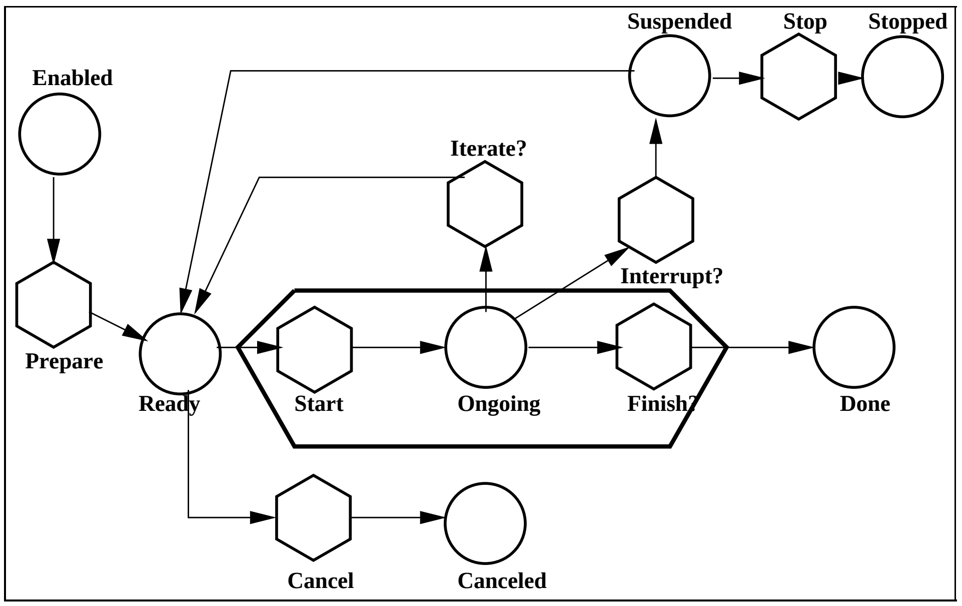


Figure 5.2: The Controller X-schema

that any of these can be composed of sub-schemas (limited depth). In general, however it is usually only the start-ongoing-finish? process (shown by the enclosing hexagon) that is decomposed further. But of course, this is not always the case, since we can have preparatory processes, starting processes, ending processes, stopping processes, etc. One thing to note in Figure 5.2 is that whether an ongoing process is finished or interrupted or iterated depends on other parameters supplied by the **process** features of the individual x-schema in question or by specific world state or linguistic input.[3]

5.4 Links to Verb Semantics

We hypothesize that such sensory-motor controllers may be directly coded in our neural circuitry and be available to other cognitive processes such as language interpretation, and more relevantly may ground the semantic and grammatical structure of the well known linguistic notions mentioned above.

When composed with the **process primitives**, the CONTROLLER forms the basis

[3]In future discussion, whenever a depiction similar to Figure 5.1 is used, it is to simplify exposition, and the reader is to assume that the computational model is as shown in Figure 5.2.

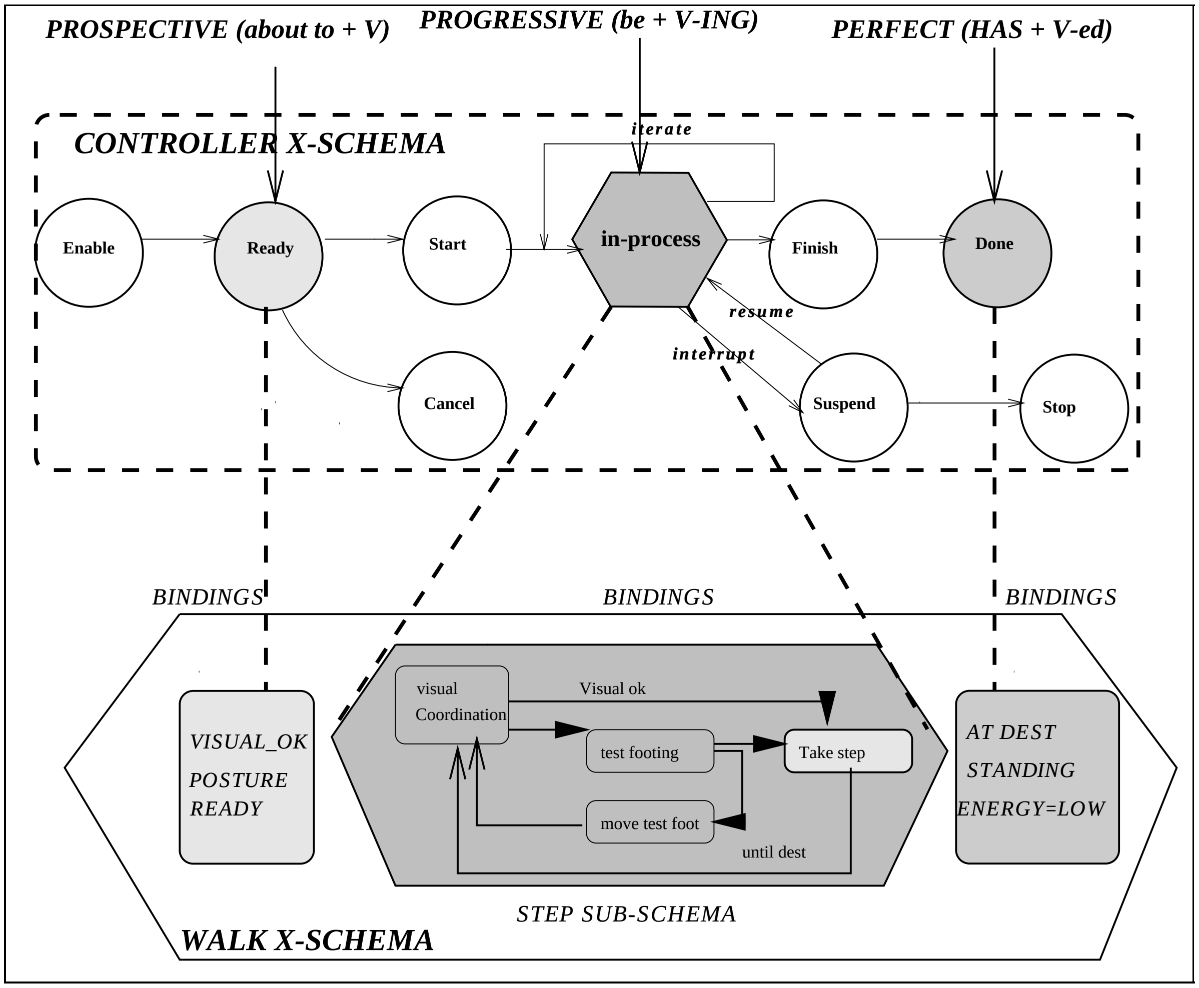


Figure 5.3: The CONTROLLER x-schema models important regularities in sensory-motor control. Grammatical devices such as morphological modifiers refer to specific nodes in the CONTROLLER x-schema, which binds to specific activation states of the verb phrase x-schema, generating the required interpretation. Shown above are the cases where an *ongoing* activity is specified using the *progressive* construction (English *be V-ing*) or the *done* state is picked out using the *perfect* construction (English *has V-ed*) or the *ready* state is picked out using a prospective construction such as *about to V*.

for grounding verbal aspect. The process primitives characterize the *inherent* semantics of the verb. Any individual verb may have some or all of these parameters set to specific values. These parameters inspired by sensory motor primitives generalize Talmy's aspectual primitives (Talmy 1985) and can easily derive the Vendler-Taylor-Dowty classification VDT (Dowty 1979). For the curious, peeking ahead a little bit, it should be fairly obvious that the VDT event/state distinction as well as the punctual/durative distinction are fairly easy to capture in x-schemas, as is the notion of goal, and more unique to our model the notion of **dynamic** resources. This allows us to model VDT **accomplishments** as durative x-schemas with goals *or* some other well defined resource constraint that governs the firing the *finish* transition. Achievements are goal-based punctual x-schemas. The important thing to note is that both the controller and the process primitives that characterize the underlying verb are x-schemas. In this way, the semantics of the verb is *grounded* in the execution of the action itself. Figure 5.3 shows the same schema as the one in Figure 3.7 but now redrawn to include the controller abstraction.

Linguistic devices specifying **phasal aspect** (lexical items, morphological modifiers and other grammatical devices) are like knobs which when set activate the corresponding CONTROLLER node, sanctioning which inferences can be made by the hearer, given the same underlying schema (verb phrase). Languages may differ in which knob settings they allow, and hence may vary which aspects and how much bandwidth they allow the speaker.

In Figure 5.3, we see how the English *Perfect* construction activates the result or consequent state of the underlying verb phrase (x-schema). The x-schema is the familiar WALK x-schema now recoded as the dynamic interaction of two x-schemas, namely the local CONTROLLER abstraction and the underlying process of walking. Thus different walking stages are now represented as nodes in the controller graph, and linguistic devices such as phasal aspect specifiers can activate any of these nodes. Node activation the changes the execution state of the underlying x-schema resulting in a modified execution trajectory. For instance, in the *Perfect* aspect in the example above, what is relevant are the characteristics of the process that bind to the result stage of the controller. These bindings are the consequences of the action/event. Propagation of these bindings and the general inferential mechanism is described in Chapter 6. In the case of *Jack has walked to the store*, this implies that the speaker directs the hearer's attention to the fact that Jack is possibly at the store, a little tired, ready to do his shopping. We note that use as a phasal aspect is just one way in which the Perfect is used in English. In general, Perfect can be used in conjunction with tense to specify the relation between a past and current situation. The work described here does not model tense, and is unable to handle these uses of Perfect. For ongoing work that addresses this state of affairs, refer to Section 5.6.1 and Section 5.6.2.

In contrast to the Perfect, the simple past *Jack walked to the store* does not say anything about whether any of the consequences of being at the store still hold. All it says was that at some time in the past, Jack was at the store. *Jack is walking to the store*, on the other hand, activates the *process* node of the controller thus calling the hearer's attention to facts such as Jack is not at the store, but actively walking towards it, expending energy, etc. Constructions such as the English *about to V* or *ready to V* or the Hindi *thayar* pick out the state of *readiness* to engage in an activity. In our model this is the controller state *ready* , which in the WALK x-schema corresponds to the visual test having returned *ok* (recall from Chapter 3 and the posture in a state of readiness (standing $\wedge$ stable).

Thus, meaning arises from the **dynamic binding** of a specific activation state of the controller x-schema to a specific activation state of the verb-complex x-schema. The structure of the controller and the relevant set of features that characterize the verb-complex jointly control the compositional possibilities.

5.4.1 X-schema parameters and Inherent Aspect

Figure 5.4 shows our x-schema based computational model of the important features of inherent aspect. Many of these features are repeated from Figure 3.6 where they played a role in modeling verb semantics based sensory-motor control. Here we argue that the very same features are sufficient to characterize what is traditionally called Aktionsart or inherent aspect. Section 5.5 shows how our model based on motor-control features is able to easily derive the canonical linguistic models of inherent aspect. Section 5.5.2 describes some additional linguistic implications of our model.

- **Duration** (Hobbs 1985; Nakhimovsky 1988) and others note that situations referred to by verb phrases have durations that fall into specific **time scales**. The table below shows some typical situations along with the associated time scales.

Situation	**Duration**
walk	minutes
run	minutes
cough	second
read a book	hours
read *War and Peace*	days
sleep	hours

Note that the precision with which we reason about the durations of activities seems to cut the continuous time line into discrete time scales. Note also that the default time scale can be overridden by specific information. For instance, the default duration of reading a book can be overridden if we know the book to be of a specific size.

In our case, the x-schema model of durations in Figure 5.4 can certainly be modified to allow only discrete time scales. However the tricky part comes from the fact that the inherent time scale of an activity is **not** a fixed, static entity. In fact we will soon see how linguistic devices explicitly allow perspectival shifts by dynamic transformation of time scales. We know of no computational model of aspect that is able to model this phenomena. Our model is outlined in Section 5.4.2.

- **Periodicity**

  Talmy (Talmy 1985) characterizes verbal aspect as the pattern of distribution of action through time. His characterization classifies verbs based on the distributivity properties of the semantics of verbs. The following classification scheme is a generalization of the scheme of Talmy. The set of verbs can be classified in the following categories.

  1. Certain predicates refer to processes that are inherently *Aperiodic*. Such processes may be due to some irreversible terminal condition obtaining (as in the case of the verb *die*), or may be due to some world state that has to be reversed before the process can be initiated again (as in the case of *fall*).
  2. Certain predicates refer to processes that are inherently *Periodic*. At the end of each period, the conditions that obtain in the world are such that the process is re-enabled. The default periodicity of the processes may be 1 (as in the case of *flash* or *hit*) or > 1 (as in *breathe*).
  3. Certain predicates refer to processes that are composed of specific *sequence* of a other activities. Consider, for instance the predicate *waltz*. This minimally

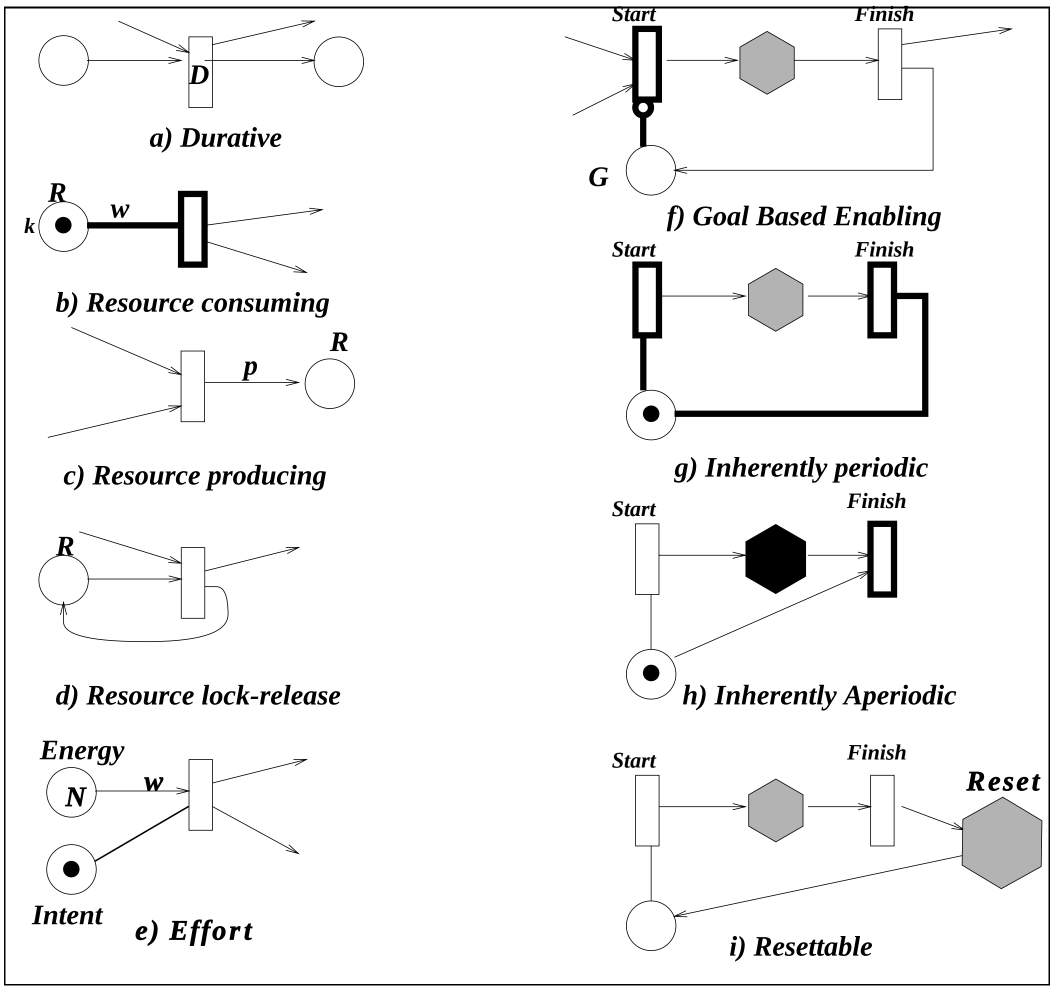


Figure 5.4: Process Features directly model Inherent Aspect or Aktionsart

requires a sequence of three patterns of motion (specific length and direction of steps). The overall activity is periodic in that by default it is iterated multiple times. Figure 3.5 showed how such sequence and other control compositions can be handled by the x-schema model.

Modeling Periodicity is shown in Figure 5.4 (g)- (i). Again the technique is fairly straightforward, repeatability with single, multiple and sequential situations shown in the figure. Note there may be other interactions (such as resource or effort) that may ultimately decide if the situation is actually repeated or not in any particular **episode**. Figure 5.4 also shows how distinction between periodic, resettable (aperiodic like fall but can be re-enabled again by getting up), and unresettable (aperiodic and one-time like **die**) activities are easily modeled in the x-schema framework.

- **Effort**

  Effort is the expending of energy to maintain an activity. A well-motivated distinction that can be made between different situations is between those that require the use of *energy* and those that don't. Our experience of activities such as *running*, *walking*, etc. makes the distinction an extremely salient one, since if we don't expend energy in these activities they terminate. In other cases, such as *knowing something* (or *believing something*), the situation continues despite the absence of continuous effort.

  Modeling effort is fairly straightforward. Figure 5.4 e) shows the model graphically. The two input places correspond to the the expenditure of energy (labeled $Intent$). The amount of energy available is the value N shown. The amount of energy required for the situation is w . If $N \gg w$, the situation can be repeated several times.

- **Resources**

  Generalizing from the idea of effort, one could obtain the more general notion of a *resource requirement* in situation. Consider the situation referred to by the sentence *John is taking the test.* Apart from the effort required (attention, scanning, etc.) there is also an inherent resource consumption where time is the resource that is being consumed until it runs out and the exam ends. So the activity described obtains till time *runs out*.

  Resources include **world conditions** that hold in a situation. This includes the classic STRIPS like *preconditions* as well as the locking and subsequent release of shared resources. Resources such as energy are routinely consumed in the performance of an action. While **energy** is a special type of resource that falls into this category, there may be other consumable resources that characterize a situation (for instance *food* in an *eating* situation). Resources and world conditions may be *created* or *destroyed* in the situation. Often the situation is described as **telic** when a special object is created at the end of the situation. Situations like *make a chair* and *demolish the building* require the explicit creation or destruction of an object.

In our model, resources can be consumed, created, held and released or destroyed. Figure 5.4 (b)- (d) graphically depict the model of the different types of situations. The consumable resource has same form in our model as the *Effortful* situation. The distinguishing feature in the two cases is the intentional input which is a requirement for the *Effortful* situation, but not a requirement for the consumable resource. Creation of a new resource is modeled by the presence of an Output Place that models the created resource. Consider the following example. Resources can be locked (if shared) and then released at the end of some situation. Computation resources, such as *attention* can be modeled this way.

While we argue that resources are important features of the aspectual class of a situation, (Wilensky 1983) contains a detailed description of how resource contention and limitations are often key features in understanding stories about conflicting or competing goals. The central point he makes is that a theory of plans that is to be used in story understanding must be able to reason about negative goal interactions (such as goal conflicts and competition) as a result of resource limitations and conflicts. (Wilensky 1983) also has a detailed discussion of the knowledge required by such story understanding systems to deal with situations involving goal conflicts and goal competitions. Our x-schema-based model is completely consistent with this view and naturally models resource levels and changes in resource. However, the impact of these on recognizing goal-interactions and possible resolution strategies of the sort described in (Wilensky 1983) is outside the scope of the system described here.

Before proceeding further, we need one abstract parameter that is derivable from the parameters mentioned earlier and the context. This parameter pertains to the relationship of the time scale of an activity to the time scale of reasoning.

5.4.2 Reasoning in multiple time scales

In this section, we will formally define a method of abstraction which models contextual interaction in defining Verb Classes. The principle is quite simple and is stated below:

Definition 16 The Multiple Time Scale Hypothesis

The interaction between a temporal granularity of interest (called the Reference Time Scale) and the temporal granularity of a described situation (called the Situation Time Scale) is used to create a *Temporal Abstraction Hierarchy*, where levels in the hierarchy control the precision with which the given situation is represented.

Several attempts at modeling commonsense knowledge have led AI researchers to propose that human perception/reasoning of activities deals with specific time scales (such as *Lifetimes* (Mcdermott 1982) or Grain (Hobbs 1985). Nakhimovsky (Nakhimovsky 1988) points out that such concepts have a natural ways of cutting the continuous time parameter space into discrete units. He proposes that the temporal relations **much greater than** and **greater-than** be used in combination with natural units (seconds, minutes hours, days, weeks, months, years and multiples of 5 for each of these units) to control the granularity of reasoning. For instance, he points out the difference in the precision of reasoning in the two cases shown below

1. I'll be back in five minutes.
2. I'll be back in six minutes.

In the first case, a much larger deviation from 5 minutes (a minute) can be tolerated than in the second (a minute cannot be tolerated). Time scales set up a way to partition verbs into aspectual classes. Using natural time scales and the the relation **much greater than** ($\gg$), I propose an automatic method to construct a temporal abstraction hierarchy to control the precision of reasoning. Section 5.8 specifies our formal model of reasoning in multiple time scales.

The temporal abstraction hierarchy deals with both **periodic** and **aperiodic** processes. Agents may reason at different levels of the temporal hierarchy and consequently arrive at different conclusions about the same situation. Our shared experience of the world allows the speaker to assume that the hearer will associate default values for the time scale of reasoning based on **natural** units of the hierarchy.

The basic idea is if we are interested at a certain time scale s_i, we consider as **Processes**, activities that happen on that time scale ($p_{s_i}(\theta)$). Activities on much smaller time scales are assumed to be instantaneous and qualify as **Events**. Activities on much larger scales are assumed to be unchanging over the episode of reasoning and can be referred to as **Unchanging States**. To discuss how the scales of reasoning affect the interpretation, consider the following examples

1. This morning, I was playing tennis.
2. In 1993, I was playing tennis.

Clearly in (1), most people would interpret the situation as referring to a single instance of playing tennis, whereas in (2) the interpretation would be one which refers to several instances of playing tennis (similar to the habitual interpretation *I used to play tennis*). Why are the two interpretations different?

The only difference between (1) and (2) is in the reference time scale. The activity is the same, the Aspect Marker (the Progressive *-ing*) is present in both cases. The time scale of the activity of playing tennis (s_i) is **minutes**. The reference time scale in (1) (s_{R1}) is **hours**, and in (2) s_{R2} is **years**. Hence $s_{R2} \gg s_i$, and $s_{R1} \sim s_i$. Hence, Equations 5.1 - 5.6 (Section 5.8) suggest that (1) be interpreted as a single process, and (2) as an extended sequence of several individual tennis playing events.[4]

Reasoning in multiple time scales is a very useful abstraction to make since it cuts down unnecessary cognitive processing. It also turns out to explain cross-linguistic data. In fact, in languages such as Tamil and Spanish, there is no *Habitual* aspect, and so disambiguation of sentences such as the ones above can only be done using a scheme like the one suggested here.

[4]This is true even in combination with temporal adverbials. For example, compare the following two sentences *This morning, when McEnroe was playing Tennis, he sprained his ankle* and *In 1991, when McEnroe was playing Tennis, he sprained his ankle.*

5.4.3 Traditional Aktionsart classes

In this section, we derive the canonical verb classification schemes that are a result of previous work in Formal Lexical Semantics and Most previous work on aspectual classification of predicates (Vendler 1967; Comrie 1976; Dowty 1979; Moens & Steedman 1988; Dorr 1993; Talmy 1985). are an extension of the *VDT* system that postulates the following situational distinctions.

1. States.
2. Punctual Activities.
3. Durative Activities.
4. Accomplishments.
5. Achievements.

Though the exact definitions vary across the different proposals, the main distinctions between these classes is made in terms of the following three parameters

1. Telic/Atelic [+t/ -t].
2. Punctual (Atomic)/Durative [+a / -a].
3. Dynamic/Stative [+ d / - d].

The table below shows how the various parameter settings constitute the various classes.

Aspectual Class	**Parameter Setting**
States	[-d]
Punctual Activities	[+a, -t, +d]
Durative Activities	[-a, -t, +d]
Accomplishments	[-a, +t, +d]
Achievements	[+a, +t, +d]

Henceforth, the various classes mentioned above as well as the associated parameter settings will be referred to as the Vendler-Dowty-Taylor (VDT) classification scheme.

5.5 Results Of The Aspect Model

5.5.1 Deriving the VDT scheme

The VDT type classes can be systematically derived from sensory-motor parameters shown in Figure 5.4 and Figure 3.5. Table 5.1 shows the mapping. An explanation of the mapping scheme follows.

Table 5.1: Mapping VDT class to x-schemas

VDT Class	**VDT Parameters**	**x-schema features**
States	[-d]	x-schema marking
Punctual Activities	[+a, -t, +d]	Transition(with $D = 0$)
Durative Activities	[-a, -t, +d]	x-schema or Transition (with $D \geq 0$)
Accomplishments	[-a, +t, +d]	Goal-enabled x-schema
Achievements	[+a, +t, +d]	Goal-enabled Transition

VDT based classification schemes are a *subset* of the process parameters that are used in our more fine-grained representation. The distinction between states and events is ubiquitous in linguistics (Langacker 1987) and (Michaelis 1992) argues persuasively for its being a basic distinction respected by all languages. The **bi-partite** nature of the x-schema verb model inherently captures this linguistic distinction in a direct and natural manner. The issue of dividing activities into durative and punctual is also directly modeled in the x-schema framework that can represent transitions with real durations. The distinction between achievements and accomplishments is usually modeled as telic dynamic processes with well defined end point. The difference is that *achievements* are conceptualized in the VDT schema as atomic actions (without duration or internal structure), whereas *accomplishments* are telic, durative processes. In the x-schema based model, achievements are goal-enabled **transitions** whereas accomplishments are goal-enabled processes. Goal-enabling is shown in Figure 3.6 in Chapter 3.

While showing that we can derive the VDT scheme from sensory-motor abstractions points toward the epistemological adequacy of our specific verb classification scheme, there are several interesting issues that arise in from our x-schema based semantics that have not been properly explained by other theories. In particular, we feel that the independent motivation for our model as a representation of actions useful for monitoring and control endows it with explanatory power, that a descriptive listing of linguistic regularities lacks.

- **States** in the VDT system correspond to marking vectors of x-schemas. Recall that a marking vector is a vector of the number and type of tokens in the various x-schema places. Note that the x-schema state is distributed over the entire net. A given state is thus a set (technically a multi-set) of marked places. In the simplest case, a state is a single place. Entering the state involves firing some input transition (whose output is the place in question). Thus firing the transition will mark the place (deposit a token) signifying that the state holds. Once a state holds, the only way to leave the state is to fire an output transition (one of whose inputs is the place in question). Firing an output transition consumes the token from place is not marked, and the state ceases to hold. The downward sub-interval property of states (White 1991) (which states (sic.) that if a state holds for a certain time interval T then it holds for all sub-intervals ($\forall t_i \subseteq T$)) follows trivially from the fact that markings (states) persist until drained by a transition, so if marking has been persistent for an interval T it has been persistent $\forall \; t_i \subseteq T$.

- **Durative Activities** in the VDT sense can be modeled by x-schemas with durative transitions. But as we suggested earlier the whether an activity is durative depends on **two parameters**, one of which is the intrinsic duration (or time scale), the other being the *reference* time scale s_R. Thus activities can be made durative if their intrinsic time scale (s_i) is on the order of the reference time scale (s_R), ($s_R \sim s_i$). An example is the well known slow motion movie example of coughing which changes a normally punctual activity into a durative one. A more interesting prediction of the model (which needs further confirmation) is that if we increase the time scale sufficiently, we may be able to view some slowly changing processes (treated canonically as states) into durative activities as well. An example of this phenomenon is that in suitably large time-scales, California cannot be *at* a location since it is drifting toward Alaska. But under normal time-scales of reasoning, California is *at* some fixed longitude and latitude.

- **Punctual Activities** in the VDT system correspond to instantaneous x-schema **transitions**. Again, as in the durative case, activities can be made punctual by changing the time scale of reference s_R to be much greater than the intrinsic time scale of a durative process (s_i) ($s_R \gg s_i$) (as was shown in the playing tennis example in Section 5.4.2).

- **Accomplishments** in the VDT system correspond to x-schemas that are either goal-enabled or have certain resource or world conditions that can fire the *finish* transition of the CONTROLLER. This allows for the solution of various problems that arise from the $[+d, +t, -a]$ classification of the VDT system. Specifically, the imperfective paradox ceases to be a problem with our representation. Section 5.5.5 discusses this issue in detail.

5.5.2 Some additional implications

Our model can quite easily explain the observation (Dowty 1979) that accomplishments (x-schemas with durations and goals) can serve as complements of the verb *stop* (as in *Lucy stopped fixing the car*) but that achievements (transitions to goal) cannot (as in **John stopped falling off the chair*). Simply, transitions cannot be interrupted but x-schemas can.

In our model, events require at least one transition, while states are bags (multi-sets) of places. This directly explains the **pseudo-cleft** constructions where you can say *What John did was run* but not *What John did was love Mary. Doing* in our model requires firing a transition. The only dynamics allowed for places are when they are **initially** marked (signifying **inception** of the state) and when their marking is actively consumed by a transition (corresponding to that state **ceasing** to hold). Thus our model predicts that the the only way to use the pseudo-cleft construction is to specify the inception or cessation of the state as in *What John did was fall in love* or in *What John did was stop loving Mary* (both perfectly felicitous utterances).

Furthermore, it is well known (Micahelis 1992) that in languages where the perfective/imperfective distinction is present (such as French), the perfective can be used to specify the inception of a state. Our model is completely consistent with this evidence, since as will see later, the perfective perspective corresponds to an x-schema **transition**, and from the arguments above, the only natural transitions of a state are its inception and termination. So a perfective description of a currently holding state can only be talking about its inception event.

Finally, in our model, we are able to define telicity with fine-grained primitives that helps avoid some paradoxes with earlier accounts and explain some compositional profiles.

Definition 17 Modified Telic

A situation can be considered telic if it requires a resource. In this case the final event of the characterizing the situation is the complete consumption of the resource or of the creation of a resource. An example is *eating the sandwich*, where the situation ends when the sandwich is done. Any intentional situation can be Telic. In this case the final event signifying completion can be a durative adverbial (as in *He ran until 5 P.M.*), a specific world state (as in *draw a circle*), a specific effort (as in *run till tired*), or a specific *affective* state (*sleep till restored*).

An important thing to note is that our characterization of **inherent** aspect using sensory-motor parameters crucially retains the **active** and **dynamic** nature of the representation and the corresponding entailments of **fast**, **real-time** inference) allows us to solve well known problems in aspectual classification (such as indefinite plurals, the imperfective paradox, and some unclassifiable words (such as “die”)).

5.5.3 Modeling Phasal Aspect

Aspectual modifiers such as (*-ing, has X-ed*), and other techniques (such as verbs and adverbs like *start, stop* etc.) provide an *initial marking* to the CONTROLLER schema. The initial marking consists of placing tokens in selected place(s) preferentially selecting one or more transitions. The *inherent semantics* of a verb form interacts with the controller by enabling or disabling specific transitions. Our model allows any of the controller nodes to be decomposed further. For instance the beginning of an activity may involve a limited number of sub-processes (the starting process). The *process* transition is usually decomposed to an appropriate subnet.

Figure 5.5 shows how **phasal aspect** (Michaelis 1992) maps onto the controller in a fairly straightforward way. While the standard literature in the logical semantics, cognitive semantics and the computational semantics community has focused on the phases shown in Figure 5.5, we note that the the interruption and suspension of processes/actions/events (either voluntary because some goal is satisfied) or involuntary (because resource levels are low, or preconditions are not satisfied) are equally important in interpreting aspectual

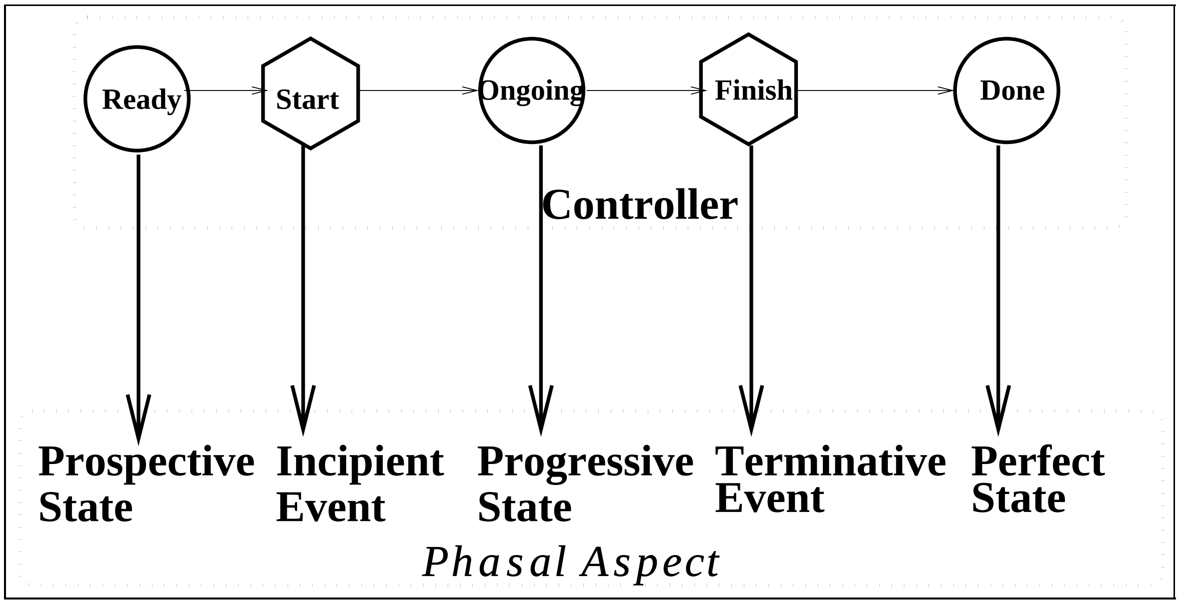


Figure 5.5: Representing Phasal Aspects in the Controller

expressions. This is something we believe our model is unique in modeling.

In addition to the standard categories shown above, our model is able to model the **imminent** aspect (*ready* but not started yet) and separate it from the **prospective** (*enabled* but $\neg$ *ready*) and **inceptive**. This will prove to be fairly important since the distinction is respected by metaphoric projections (ref. Chapter 9). Additionally, our controller is able to elegantly model of the **cessive** and **interruptive** phases as well.

While most languages have grammatically obligated structures for expressing aspect, there are many optional devices used as well. In the model proposed here, these correspond to activating specific states (trajectories) of the controller. Figure 5.6 shows some examples of this phenomenon. One interesting result of our design is that many lexical items code for specific types of interrupts in the controller (for example *stumble* codes interrupting a WALK schema.

Unlike all previous theories we are aware of, our model does **not** require an online transformation (or type-coercion) operation to model phasal aspect. An example of a type-coercion theory is Moens and Steedman's meta-rules for corecion (Moens & Steedman 1988). On the contrary, we argue that the **dynamic** binding of a specific controller state

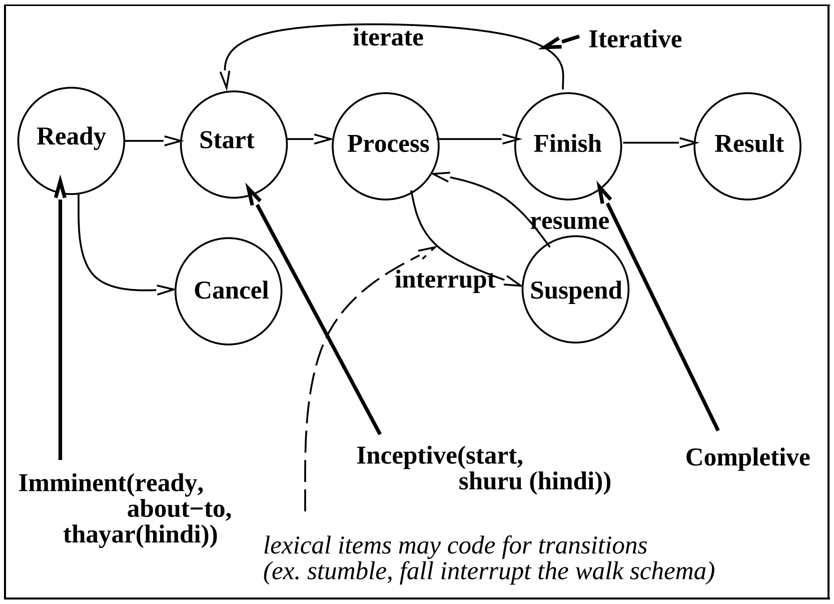

iterate
Iterative
Ready
Start
Process
Finish
Result
resume
Cancel
interrupt
Suspend
Imminent(ready, about−to, thayar(hindi))
Inceptive(start, shuru (hindi))
Completive
lexical items may code for transitions (ex. stumble, fall interrupt the walk schema)

Figure 5.6: Other ways of Expressing aspect

with the underlying x-schema is sufficient to provide the required interpretation in real-time. This becomes critical in modeling the interaction of phasal and inherent aspect. The next few sections flesh out this assertion further.

5.5.4 Interaction of Phasal and Inherent Aspect

Figure 5.3 shows the implemented computational model that shows the interaction between phasal aspect and the inherent semantics captured *directly* by our verb phrase representation. The top half of Figure 5.3 (enclosed by the broken line rectangle) corresponds to now familiar controller abstraction. In the current model, every verb has and interacts with an instance of the CONTROLLER x-schema. The specific interactions come from the inherent semantics of the verb through its *process features*. For example, walking involves specific *enabling conditions* (such as a proper upright posture, a visual test indicating a steady ground, etc.) specific *resources* like energy, and may have a specific *goal* such as being *at the store*. These features interact with the controller preferentially enabling or disabling transitions.

Interactions also come from the grammatical devices, which typically supply an initial activation (marking) to the controller. For instance, consider the situation faced by our interpreter upon hearing the utterance *Jack has walked into the store.* Figure 5.3 shows this situation graphically.[5]

An example is shown in Figure 5.3. The *Perfect* perspective that results from the *has walked to the store* results in a specific activation to the *result* stage of the CONTROLLER x-schema resulting in a specific *binding* to the walk situation. Here the hearer is sanctioned the inference that Jack is at the store and possibly tired. More importantly, using the Perfect aspect the speaker sets the **context** for future discourse that the world conditions and agent state at the time of description are the consequent state of having walked to the store. The architecture to model such inferences is described in Chapter 6. Note that this use of the Perfect is only one of several possible uses. To properly model other cases requires the ability to model tense-aspect interactions, a subject not dealt with in this work. Some idea of the issues involved can be found in Section 5.6.1.

Contrast this to the case of **is walking to the store**. A few steps of the simulation output is shown in Example 7.

Example 7 Interpreting "is walking to the store"

The simulation below shows the WALK x-schema interpreting "is walking to the store". The destination parameter becomes bound to the location of the store, and being at the At(store) is required for the x-schema to successfully complete execution. Hence in the table below, we see that *energy* is consumed and *Dist(store)* decreasing since the process of walking is *ongoing* . The interpretation of the phrase is thus the result of executing a mental model of walking to the store with the appropriate parameters.

```
x-schema place to column mapping:

     1 = started
     2 = activate
     3 = ready
     4 = ongoing
     5 = done
     6 = VISUAL_OK
     7 = STEP_READY
     8 = VISUAL_NOTOK
```

[5]For exposition purposes, the *test-foot* branch is not shown.

```
     9 = BYPASS?
    10 = Energy
    11 = Dist(Store)
    12 = At(Store)

Simulation:

Marking    1   2   3   4   5   6   7   8   9  10  11  12  | Transitions to
========================================================= | --------------
 M0     |  0   1   0   0   0   1   0   0   0  14  10   0  | ( M1 )
 M1     |  0   0   1   0   0   1   0   0   0  14  10   0  | ( M2 )
 M2     |  1   0   0   0   0   1   0   0   0  14  10   0  | ( M3 )
 M3     |  0   0   0   1   0   1   0   0   1  14  10   0  | ( M4 )
 M4     |  0   0   0   1   0   1   1   0   0  14  10   0  | ( M5 )
 M5     |  0   0   1   1   0   1   0   0   0  13   9   0  | ( M6 )
 M6     |  1   0   0   1   0   1   0   0   0  13   9   0  | ( M7 )
 M7     |  0   0   0   2   0   1   0   0   1  13   9   0  | ( M8 )
 M8     |  0   0   0   2   0   1   1   0   0  13   9   0  | ( M9 )
 M9     |  0   0   1   2   0   1   0   0   0  12   8   0  | ( M10 )
 M10    |  1   0   0   2   0   1   0   0   0  12   8   0  | ( M11 )
========================================================= | -------
           1   2   3   4   5   6   7   8   9  10  11  12
```

■

Example 7 shows the walk-controller simulating walking to the store. Crucially, note that the progressive aspect *binds* to a dynamic process that corresponds to Jack expending energy and moving closer to the store. Contrast this to the **Perfect** where entirely different inferences are available to the interpreter.

5.5.5 Imperfective paradox

The Imperfective Paradox (Dowty 1979) comes from trying to separate verbs of accomplishment from verbs of activity. One diagnostic test comes from the different entailments of the two verb classes when used in the Progressive Aspect.

Consider the difference between *Jack was walking* and *Jack was walking to the store*. The first sentence sanctions the inference that *Jack walked*, whereas the second does not sanction the inference that *Jack walked to the store*. This creates what is referred to as the imperfective paradox, and model theoretic accounts are forced to invent new unanalyzed

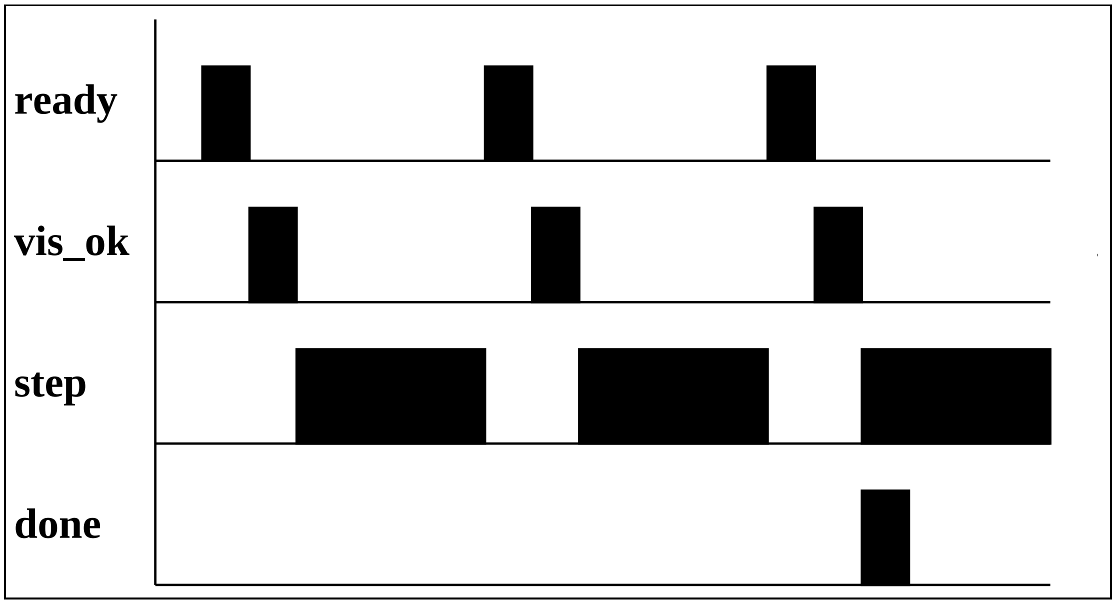


Figure 5.7: Interpretation of *walking* as a trace of a portion of the **Marking** vector

primitives such as the inertial world primitive **Inr** (Dowty 1979) (set of predictable world futures) to establish truth conditions that satisfy this test.

Figure 5.8 and Figure 5.7 graphically depict the relevant portion of the Marking vector for the situations described by the two sentences. In our action-based x-schema model, the difference comes from the constraint that in one case the *finish* transition can fire or equivalently the *done* state is **reachable iff** only if the *goal* (reaching the store) obtains. It is important to note that goal attainment can be asynchronously asserted by a scheduled or unscheduled perceptual process (or can be time based (I saw Jack walking to the store yesterday)), and the x-schema reacts appropriately. In the case of *walking*, no such constraint exists and the result obtains after every two steps (taken from Dowty's definition (Dowty 1979).

Technically the resolution of the paradox relies on whether the **reachability** graph of the active x-schema contains the *done* state as a sub-marking, a question that can be answered by activation propagation over the x-schema as shown in Chapter 3. Example 9 shows the actual results clearly showing the case where the *done* state is readily obtained for the *walking* scenario and and results shown from the simulation in Example 8 show that

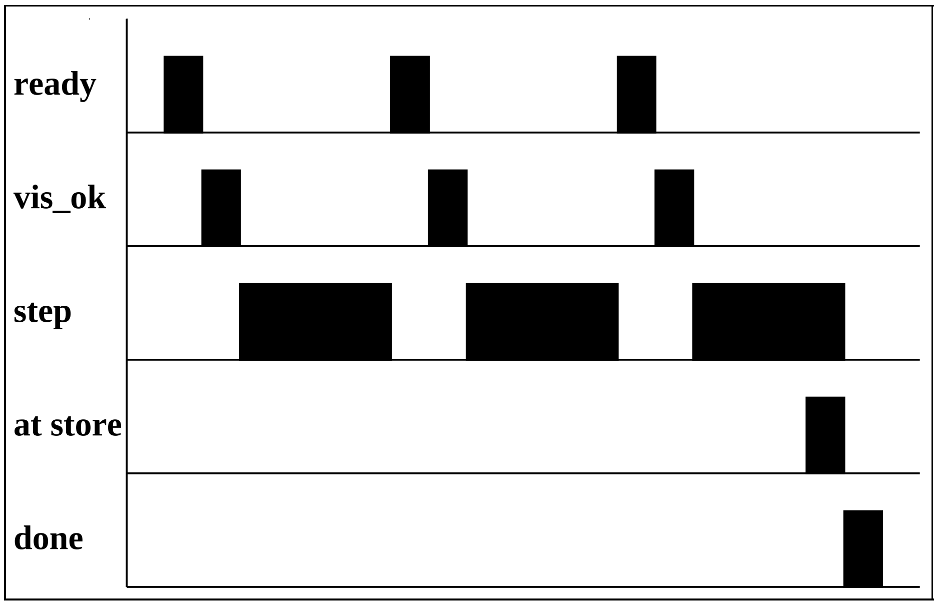


Figure 5.8: Interpretation of *walking to the store* as a trace of the relevant portion of the **Marking** vector

---

the *done* state is not reachable from the initial marking for the *walking to the store* scenario.

Example 8 Simulating Walking to the store

In this example note that walking to the store is only *done* when the walker is at the store. Until that event occurs, the walking continues (in the absence of an external interrupt). Given parameters such as distance and rate, etc. the simulation shows decreasing distance to the store (column 11) and descreasing energy (column 10). When distance to store is down to 0 , the *finish* transition is enabled leading to the marking of the *done* state indicating a completed action (happens in cycle 42 of the simulation shown below).

```
x-schema place to column mapping:

     1 = started
     2 = activate
     3 = ready
     4 = ongoing
     5 = done
     6 = VISUAL_OK
     7 = STEP_READY
     8 = VISUAL_NOTOK
     9 = BYPASS?
    10 = Energy
    11 = Dist(Store)
    12 = At(Store)

Reachability Simulation:

Marking     1   2   3   4   5   6   7   8   9  10  11  12  | Transitions to...
=========================================================  | -----------------
 M0     |   0   1   0   0   0   1   0   0   0  14  10   0  | ( M1 )
 M1     |   0   0   1   0   0   1   0   0   0  14  10   0  | ( M2 )
 M2     |   1   0   0   0   0   1   0   0   0  14  10   0  | ( M3 )
 M3     |   0   0   0   1   0   1   0   0   1  14  10   0  | ( M4 )
 M4     |   0   0   0   1   0   1   1   0   0  14  10   0  | ( M5 )
 M5     |   0   0   1   1   0   1   0   0   0  13   9   0  | ( M6 )
 M6     |   1   0   0   1   0   1   0   0   0  13   9   0  | ( M7 )
=========================================================
 M40    |   0   0   0  10   0   1   1   0   0   5   1   0  | ( M41 )
 M41    |   0   0   1  10   0   1   0   0   0   4   0   0  | ( M42 )
 M42    |   0   0   1  10   0   1   0   0   0   4   0   1  | ( M42 )
 M43    |   0   0   0  0    1   1   0   0   1   4   0   1  | ( M44 )
 M44    |   0   0   0  0    1   1   1   0   0   4   0   1  | ( )
 DEADLOCKED=========================================================
            1   2   3   4   5   6   7   8   9  10  11  12
```

■

Example 8 shows the reachable states for *is walking to the store.* The key here is that the **done** state is never marked **until** the destination is reached (time 40) in the simulation[6](depends on distance, rate, etc.). More importantly, while the destination is still to be reached, the process may be interrupted, stopped or resumed. For instance, the narrative *John was walking toward the store when he met his friend*, signifies process interruption with an indeterminate future action progress. We believe that an x-schema based dynamic model is essential to capture the right inferences of such an utterance.[7]

Contrast the simulation in Example 8 the that in Example 9 which shows the system interpreting *is walking* . Here after an initial setup (to coordinate, initialize perceptual tests, etc.), the done status can fire after every 2 steps (as defined). Thus the done state is reachable after every two steps.

[6]Note that the line in the middle of Example 8 signifies that the simulation results for the intervening period are not shown.

[7]An interesting side-effect of the simulation is that the ongoing place is able to *count* steps (something that is not essential to the model, of course).

Example 9 Walking

Under the definition of walking where two steps constitutes a walk (from Dowty 1979), we can see how our simulation shows that the *done* state is reachable after every two steps, thus allowing the agent to infer that “is walking” implies “walked”.

```
x-schema place to column mapping

     1 = started
     2 = activate
     3 = ready
     4 = ongoing
     5 = done
     6 = VISUAL_OK
     7 = STEP_READY
     8 = VISUAL_NOTOK
     9 = BYPASS?
    10 = Energy

Reachability Simulation:

Marking    1   2   3   4   5   6   7   8   9  10  | Transitions to...
================================================= | -----------------
 M0     |  0   1   0   0   0   1   0   0   0  14  | ( M1 )
 M1     |  0   0   1   0   0   1   0   0   0  14  | ( M2 )
 M2     |  1   0   0   0   0   1   0   0   0  14  | ( M3 )
 M3     |  0   0   0   1   0   1   0   0   1  14  | ( M4 )
 M4     |  0   0   0   1   0   1   1   0   0  14  | ( M5 )
 M5     |  0   0   1   1   0   1   0   0   0  13  | ( M6 )
 M6     |  1   0   0   1   0   1   0   0   0  13  | ( M7 )
 M7     |  0   0   0   2   0   1   0   0   1  13  | ( M8 M9 )
 M8     |  0   0   0   0   1   1   0   0   1  13  | ( M10 )
 M9     |  0   0   0   2   0   1   1   0   0  13  | ( M10 M11 )
 M10    |  0   0   0   0   1   1   1   0   0  13  | ( M12 )
 M11    |  0   0   1   2   0   1   0   0   0  12  | ( M13 M12 )
 M12    |  0   0   1   0   1   1   0   0   0  12  | ( M14 )
 M13    |  1   0   0   2   0   1   0   0   0  12  | ( M15 M14 )
 M14    |  1   0   0   0   1   1   0   0   0  12  | ( M16 )
================================================= | -----------------
```

■

From the discussion above, we conclude this section with the following points to note with respect to our representation and its avoidance of the imperfective paradox are the following.

1. The imperfective paradox is a result of using a time based representation, where the only primitives are world evolutions over temporal intervals or points. This gives actions and agent-initiated (directed) world evolutions and other external events and occurrences equal status, making it impossible to model the attendant issues of intention, resources, and control. In this case, we are in concord with (Steedman 1995) in noting that an action-based representation is an essential first step to avoid the imperfective paradox.

2. The ability of our model to *dynamically* model the effects of resources and voluntary or involuntary suspensions seems to be essential to capture the issues involved. For instance, reachability is adversely affected in walking to the store not **just** in the case when the store is not reached, but also as a result of an involuntary suspension (became too tired or fell on the way) or a voluntary one (the required items had been obtained in some other way, retracting the goal of being at the store). These kinds of dynamic action-features are naturally modeled in a highly-responsive, active, asynchronous action controller, aka x-schemas.

Importantly, our model has explanatory power which allows us to conclude that an activity with a specific goal or telic constraint (ref. to our modified definition of telicity in Section 5.5.2) may not finish because of lack of energy or resources, or because of suspension or stoppage due to an external or internally generated interrupt (voluntary or involuntary suspension). An activity without such constraints may *finish* when a specific control sequence is completed. This explanatory power can be brought to bear on the *acceptability* of certain **but** sentences.[8]. For instance *I was walking to the store but became tired* is acceptable but (sic.) *I was walking to the store but the Mars Rover landed perfectly* is unacceptable. One of the well known uses of the lexical item "but" is to provide information that contradicts the expected future trajectory of an action. So by asserting resource unavailability (energy), *tiredness* contradicts the expected future trajectory of the walking action and is acceptable. Mars Rover landings (in most contexts) do not, and the

[8]Thanks to George Lakoff for pointing this out

second sentence is thus unacceptable. Note that a rich action representation which models resources, preconditions, goals, and interrupts is key to making judgments of acceptability.

In summary, our dynamic action model grounded in sensory-motor primitives avoids/solves the Imperfective Paradox implicitly without resort to any unanalyzed primitives such as *Inertia Worlds*. The important thing to note is that unlike previous efforts, we needed no special language-specific meta-rules (such as type coercion networks). Our solution is part of the normal execution of the model itself and a direct result of language being grounded in action.

5.5.6 Problems with Telicity

Telic situations are distinguished in standard theories as an **Activity** with a well defined **Achievement**. Hence telic situations are a combination of a **Process** followed by an **Event** which is the culmination of the **Process**.

This notion of telicity creates problems. For instance, take the verb *die*. Vendler (Vendler 1967) classifies it as an *Achievement*, which implies a *punctual* characteristic. But the fact that you can say *Harry is dying* suggests that the situation is *durative* and *telic*. So Vendler's classification is clearly problematic.

On the other hand, if we suggest that *dying* is telic, composed of a *process* and then an *event*, we should be able to say * *Harry was dying, but recovered*. Since this is quite a strange sentence for many speakers, (Comrie 1976) suggests a new class of situations where *dying* is a *special* telic predicate, one which does not allow its end-event to be prevented, once the associated process starts.

Thus traditional accounts are forced to abandon compositionality (Comrie 1976) and invent a new class of situations, which once *started* cannot be *prevented*. In our model (assuming Comrie's reading) situations that cannot be interrupted is modeled by inhibiting the interrupt transition in the controller. This implies that the activity in question cannot be interrupted. The only possible evolutions are then to completion or for the activity to be iterated. If we further require that the activity not be iterated, we inhibit that controller transition as well. Now the only trajectory of the ongoing ("dying") process is to completion. This is the situation for the CONTROLLER shown in Figure 5.9. In general, these inhibitions are of different strengths modeled by *stochastic* transitions with different

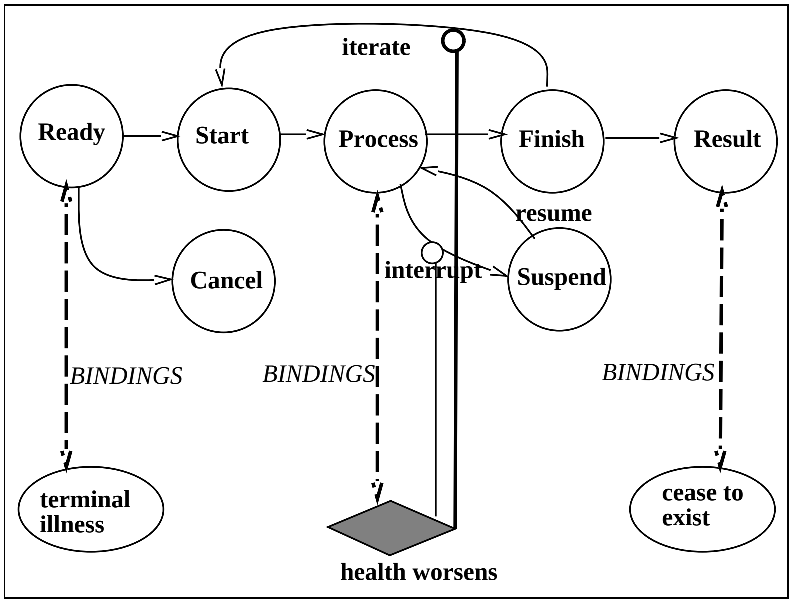


Figure 5.9: Inherent Semantics may constrain possible evolutions

rates (λ)(ref. Chapter 3). While Figure 5.9 shows how we can model Comrie's reading of the verb *die* without introducing new primitive Aktionsart classes, we can safely predict that the work described in this thesis will not for-close discussion on the meaning of death. [9]

5.5.7 Iterative readings of Progressives

Often, the inherent semantics can make generating the meaning of aspectual expressions non-compositional. For example in the cases *He is rubbing ointment* or *He is coughing*, the normal reading is *inherently* iterative while as we have seen the normal reading of the progressive construction used (*be + V-ing*) is that it is an *ongoing* process.

One could suggest that such normally iterative reading is linked solely to the punctual nature of the underlying activity, which makes iteration the only method to sustain the activity. This works for *is coughing* but does not for *rubbing ointment* since a single

[9] A view that many found attractive in George Lakoff's graduate Cognitive Linguistics class was that "die" should really be viewed as taking the interrupt-abort trajectory of a top level STAY ALIVE schema in the same way that the concepts "Trip" and "fall" refer to the interrupt-abort trajectory of the WALK x-schema.

rub is certainly not punctual.

Type-coercion theories (Moens & Steedman 1988) require that iteration coerce a durative activity into a punctual one, and once this coercion is made, an iterated reading of a progressive construction is possible. This again has problems. Apart from being an ad-hoc meta-rule (why should the type coercion be enabled by some activities and not others?), the explanation is still unsatisfying because it seems that in the *is rubbing* case, we don't actually collapse a single rub into a point.

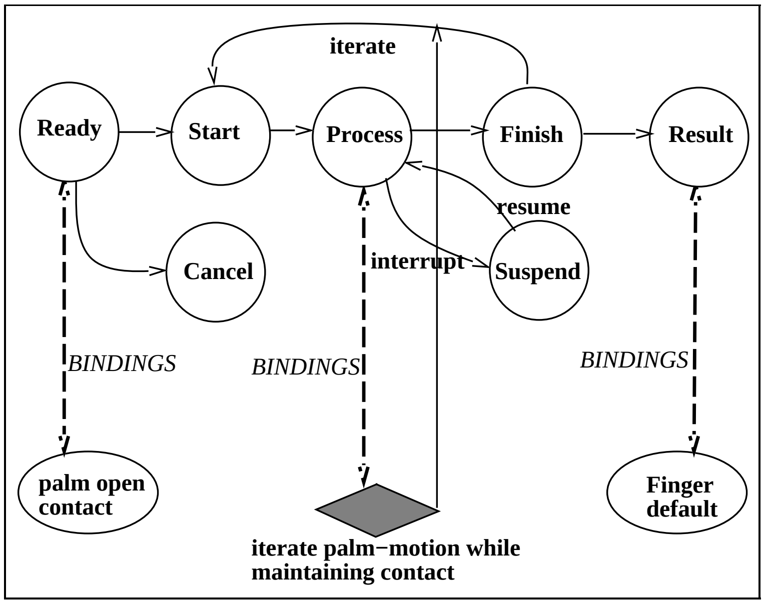


Figure 5.10: Inherent Semantics may enable specific evolutions

---

In our model, certain activities (those that are inherently iterative) preferentially enable the iterate transition, rather than the finish transition. Thus in Figure 5.10, this corresponds to executing the *rub* several times before finishing (how many may depend on other resources such as energy, or goal being satisfied).

5.5.8 Dynamic time scale shifting

To illustrate the point, consider the famous example below.

Example 10 Bill is coughing. ■

The standard account of the meaning of this sentence is that coughing is a *punctual* verb, and the only way it can appear in a Imperfective perspective (since the Imperfective perspective requires a *Process verb*) is if the process is *iterated.* This is a result of the *inherent* punctuality of cough.

The argument stated above does not work. For instance, if I change the time scale of reasoning by showing a slow motion movie of a cough and saying *Look he is coughing*, I could be pointing to the process of a *single* cough.

Instead of the kind of argument made above, we propose that the punctuality of a verb is a derived entity that depends on the interaction between a Time Scale of reasoning and the time scale of an activity. In most contexts, since the time scale of a cough is much smaller than the reference time scale, a cough counts as an event and needs to be iterated to form a process. However, upon changing time scales of interest, a cough can become a process.

Thus while certain verbs encode the underlying action as punctual or durative, specific context including a reference time-scale can explore the actual process or treat the activity as punctual. Our experience of most activities seem to fall into the **minutes** time scale, and so the *default* interpretation would classify activities in the **second** time scale (like *cough*) as a punctual verb.

The Multiple Scale Hypothesis can also elegantly explain when a durative verb can be recast as a punctual verb. We saw an example of this modification in the earlier section with the *play tennis* situation where a shift in the level of the temporal resolution resulted in the shift in the Aspectual Class of the situation.

5.5.9 Perfectivizing operators

Perfective aspect involves viewing a situation as a single whole, without explicit reference to the individual constituent phases. English does not have a Perfective/Imperfective morphological distinction, but several languages such as Russian, Mandarin, Spanish, etc. do. (Comrie 1976).

In our model, the perfective perspective involves applying the network transformation operation shown in Figure 5.12. This operation corresponds to not-monitoring the execution of a subnet. This leads to the situation shown in Figure 5.11. Figure 5.11 shows the two transformations involved. Being a graphical model, we can specify both information losing and non-information losing net transformations structurally.

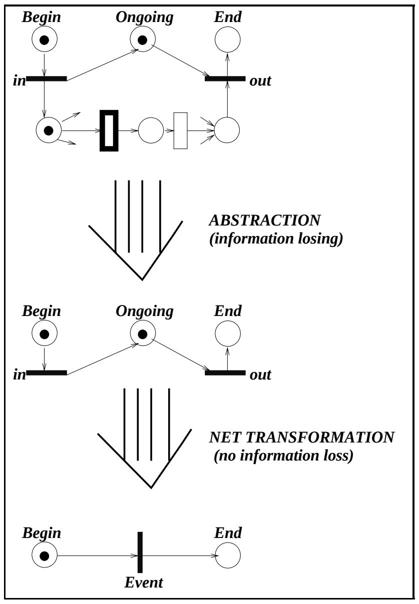


Figure 5.11: Not monitoring/paying attention to subschema-execution is like making an abstraction with loss of information.

The lack of subnet monitoring involves the **abstraction** transformation (we don't monitor the detailed subschema execution anymore). Obviously this results in information loss compared to the earlier situation (depicted on top of Figure 5.11). The second transformation is possible because, the marking at the *ongoing* node can never be visible since it instantaneously appears at the output. Hence no information is lost in the transformation from the middle to the bottom of Figure 5.11.

This situation is depicted in the middle of Figure 5.11. When applied to the CONTROLLER, the result is shown in Figure 5.12.

Figure 5.12 shows one possible abstraction from the **controller** where the process is not monitored, only starts and finishes are. In this case, through an information loosing net transformation (Figure 5.11), we get a a simplified controller that corresponds to the the **perfective** perspective present in many languages (Langacker 1987).

Once again, the our process-based model has explanatory power. For instance, the model predicts that a perfective perspective may allow iteration but not interruption

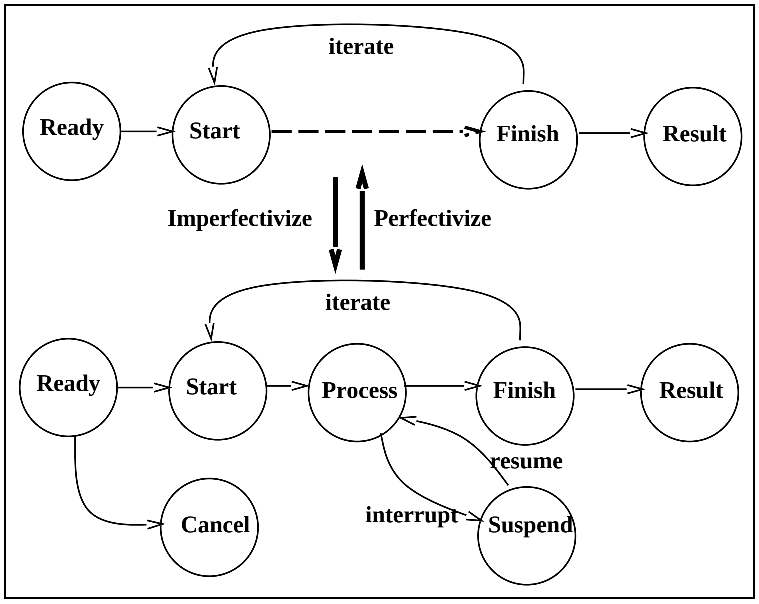


Figure 5.12: Perfectivizing and Imperfectivizing Operators

or cessation (since transitions are not interruptible). So in our model, the sentence *John swung at the ball but stopped* would be marked, but *John was swinging at the ball but stopped* would be fine. Similarly the iterated perspective *John swung at the ball three times* is acceptable.

Note an interesting transformation occurs in going from the imperfective to the perfective viewpoint. In the imperfective case, what started out as focus on a **state** (marking the **ongoing** controller state) after the network transformations in Figure 5.11 ends up as an **event** (**transition**) with a before state and an after state. This account is consistent with and provides a computational model for the intuitions of (Michaelis 1992).

5.6 Current Work and Extensions

5.6.1 Perfect revisited

So far, we discussed one of the meanings of the Perfect aspect, called the *perfect of result* (Comrie 1976), which indicates that some consequence of the event in question holds at the time of description. A confirmatory test of the difference between the Perfect (as in *I* *have lost my keys*) and the simple past (as in *I lost my keys*) comes from the fact that in the Perfect case one can't follow the sentence with an explicit denial of the consequences (as in *I have found them again*). The simple past makes no such claims. An interesting example suggesting that the Perfect is not merely temporal comes from the example *I have bathed*[10]. In normal contexts this implies that the consequences of taking a bath (I am clean, don't immediately need another bath) holds. Note that the relevant time intervals where such a statement can be made are entirely context dependent, since for me the interval would span a day, while for a medieval monk, the interval might well be a year.

The perfect of result is only one of the uses of the Perfect. Other uses pertain to the relationship of the current situation to some past situation. Consider cases like *We have lived here for ten years*. Here, what seems to be asserted is that a situation that started in the past continues to hold currently. The pluperfect (or past perfect) as in *John had lost his keys* shifts the reference point back to the past, while the sentence still asserts that the consequences of the described event (the keys were lost) held at the past reference time. In both cases above the model described here has to be modified to include a representation of tense. Initial steps in this direction look promising and are described in Section 5.6.2. However, the general subject of tense is not dealt with in any depth in the current model.

The meaning of Perfect in some languages is complicated further since the specific kind of temporal relation asserted varies. For instance, in English, use of the Perfect construction can signal that the described event has occurred at least once in the past. Such uses of the Perfect are called *existential perfect* (Comrie 1976)[11]. An example of this is *John has been to India*, where the sentence asserts that John was in India at least once (possibly more than once). Contrast this with *John has gone to India* which uses the perfect of result described earlier. Only in the latter case do the consequences of going to India (being there, vacationing) hold at the time of description. However, even in the case of the existential perfect, *contradicting* the consequences seems to be disallowed. A classic case of this phenomenon is McCawley's example (McCawley 1971) **Einstein has been to Princeton* which seems marked (unless the context is very special, like in a play about Einstein). Here, we argue that the consequent state of the subject (Einstein) being in Princeton is contradicted by our knowledge that Einstein is dead. This direct contradiction

[10]I am assuming a definition of bath that includes taking a shower (bath $=$ soaking in tub $\vee$ showering).

[11]Many other languages (such as Chinese, Hindi) have different constructions for the existential perfect and the perfect of result and make grammatical distinctions between the two types, while English doesn't.

renders the use of the Perfect infelicitous unless supported by a special context. Note that our analysis would make no such prediction about *Princeton has been visited by Einstein*, since Princeton as a subject still exists. In addition, in the case of the existential perfect, one can specify a time-interval as in *I have been to Germany after the Berlin wall came down*, or an exact number of times as in *I have been to Africa three times*. In all these cases, exactly which consequences are *relevant* and cannot be contradicted by the use of existential perfect seems highly context dependent and the work described here offers no general theory to solve that problem.

The Perfect aspect is also used to signal "temporal closeness" of an event to the time of description. For instance, in English, I can say *I have been to the dentist this morning* while it is still morning. Exactly what interval sanctions the use of the Perfect is language dependent. For example in English one doesn't say **I have been to the dentist this morning*, in the afternoon (since the morning is over) while Spanish allows the equivalent expression. Note that while temporal closeness may allow the use of the Perfect, it is clearly not a requirement as evidenced by the *I have bathed* example.

In both the existential perfect and the temporal closeness use of the Perfect, we suspect that the inherent time-scales of activities, their periodicity, and the existence of clear and enduring consequences are important determinants of when the Perfect can be used and its meaning. However, working out a theory of these uses of Perfect awaits further research.

5.6.2 What about tense?

Tenses deal with relations between situations. The basic insight on representing tense comes from Reichenbach (Reichenbach 1947). Most implemented computational systems use some extension of the basic proposal. Our exposition follows that of (Moens & Steedman 1988) who use an extended version.

Reichenbach's key idea for representing tense was to use *three* pointers to map the temporal relation expressed onto a timeline. The three pointers are called S (for speech time), R (for reference time) and E (for event time). In the simple past tense, the reference R is to a past event E, so both R and E are in the past, while S is the current speech time. Similarly in the case of the present tense all three pointers coincide at the present time.

Future tense is modeled by both R and E referring to a later time than S.

Reichenbach's original proposal does not deal with the combination of tense and aspect for complicated events with internal structure of the kind we have been concerned with here. The key question for us is whether our representation of aspect composes with tense to correctly model the combined tense-aspect specification. This will allow us to correctly model and distinguish past perfect (as in *John had lost his keys*), from present perfect (as in *John has lost his keys*), from future perfect (as in *John will have lost his keys*). The same distinctions are obviously true of other aspectual categories.

Figure 5.13 shows some initial work that shows how the aspect model outlined in this chapter may compose with the tense model based on extending Reichenbach's proposal to complex events.

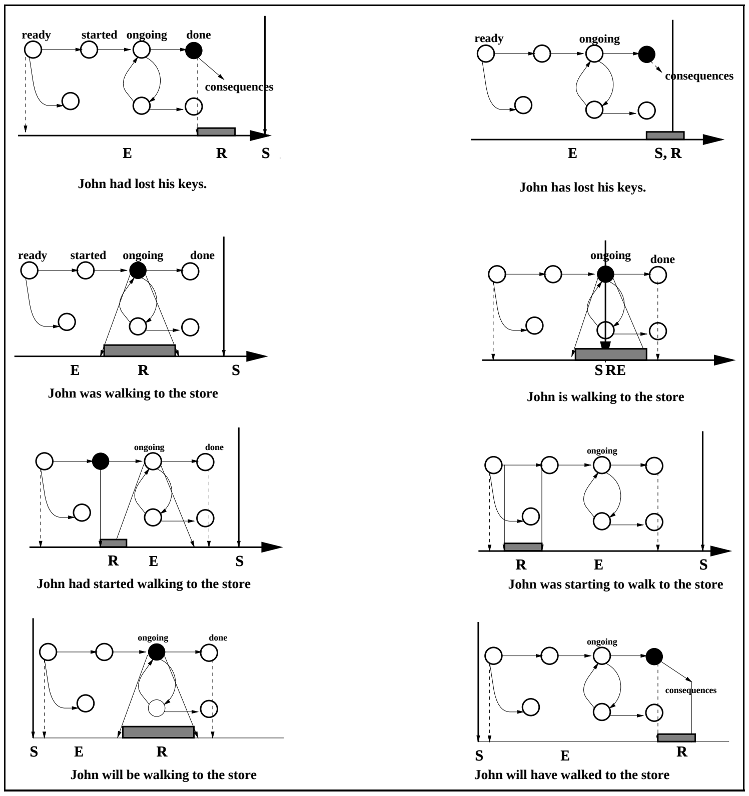


Figure 5.13: Tense and Aspect

The cases shown in Figure 5.13 pertain to the use of the Perfect and Progressive aspects in conjunction with different tenses. The representation of tense is done using the three pointers R, S, and E while the representation of aspect uses the model developed in this chapter. Consider the case of the past perfect as in *John had lost his keys* shown in the top-left corner of Figure 5.13. In all cases the dark downward arrow points to the speech time S. Here the event refers to the entire composite event of losing keys. In all cases, E refers to the projection onto the time-line of the entire CONTROLLER graph. In this case, the event is entirely in the past and E labels the composite event. The reference point R on the other hand, refers to some time where the consequences of the completed event (keys are lost) still holds. This point is also in the past, indicated by R in the figure. The projection of the reference pointer R onto the time line is shown by the shaded rectangle (R spans the shaded interval) in the various situations depicted in Figure 5.13. The result node is shown shaded to indicate that the inferences and bindings made from this stage of the process hold for the time indicated by R. In contrast, the use of the present perfect shown on the top-right hand corner of the figure, is different in that the consequences of the *lose_key* situation hold at speech time. Hence R spans the speech time S. The rest is the same as before.

The progressive construction refers to the *ongoing* node of the CONTROLLER. Thus the reference pointer, R is placed somewhere on the projection onto the time line of the *ongoing* process. The past-progressive shifts the event into the past, thus E is in the past, and R refers to the *ongoing* phase of the past event. This is the situation shown in the middle left figure that models the tense-aspect specification for the sentence *John was walking to the store*. In contrast, the sentence *John is walking to the store* has the speech and reference time referring to the *ongoing* state of the WALK(to store) process. The aspectual inferences are exactly the same, except the reference and speech times are different in the two cases.

The third row in Figure 5.13 shows how our proposal can model some more complicated interactions between tense and aspect. For instance, different nodes in the controller graph can compose with other tense-aspect combinations. For instance, past perfect can combine with the "starting" process allowing the system to model *John had started walking to the store*, shown in the left column of the third row. Note that in this case, the start process has been completed and the consequences hold (during the reference interval, John is actively walking toward the store, hasn't yet reached, etc). The reference point (R) is to that state of the underlying x-schema. As in the earlier cases, the event and reference point are in the past compared to the speech time (the past tense). In contrast, the situation on the right column of the third row in Figure 5.13 models the sentence *John is starting to walk toward the store*, which places the reference point on the *ongoing* node of the START sub-schema (not shown) covering the time span shown shaded in the figure. Note other tense-aspect combinations such as *John was ready to walk*, or *John was finishing his homework* can be similarly modeled.

Finally, the future progressive shown in the bottom-left of Figure 5.13 shows the a future E and an R that refers to an ongoing process. The future perfect, on the other hand refers to the consequences of a completed event in the future. This is the situation shown in the bottom right corner of Figure 5.13.

Thus the basic cases of Perfect and the phasal aspects seem to compose nicely with tense in the manner indicated in Figure 5.13. However, the simple case of using three pointers to model tense leaves much unspecified, since in general discourse the different temporal relations are often nested, as in *He said that he had arrived in India three weeks earlier*, where there is a current speech time (S), the speech time of the speaker in the narrative (S'), the speech event of his saying (E), the event of his arrival (E'), etc. In an ongoing project, Dan Gildea [12] is attempting to use a recursive structure called tense trees (Hwang & Schubert 1994) (based on an extension to Reichenbach's system) in conjunction with the aspect model described here to capture some of the more complex embeddings in a computational model.

5.6.3 Metaphoric extensions

We note that metaphors from different source domains *map onto* the controller (ex. *set out* (journey metaphor) and *enter* (container) map to the **start** state). These provide independent evidence that the controller abstraction seems to capture the inherent temporal structure of events. An x-schema based model that can interpret metaphoric expressions about events is described in Chapter 8. Some results of applying the model to simple newspaper stories can be found in Chapter 9. Here we show that the ability of our system in making aspectual distinctions of the kind described here is absolutely crucial to be able to interpret narratives about abstract events and plans.

[12]This is being done as a class project for George Lakoff's graduate linguistics seminar (Ling 205).

5.6.4 Other dimensions of aspectual composition

In the L_0 project, we hypothesize that "much" of what is grammaticalized in a language is grounded in patterns generated by our sensory and motor systems as we interact in the world. We conjecture that perceptual and motor control generalizations similar to and interacting with the CONTROLLER can model tense, conditionals, and modals. To this end, we have extended our framework to be able to model the composition of multiple interacting dynamic systems. Our architecture is described in the next chapter (Chapter 6). We believe with this extension, we can begin to seriously investigate long-standing linguistic issues in aspectual composition. Nancy Chang (Chang 1997) has already made some progress in this regard. A brief description of this work can be found in Chapter 10. In general, while preliminary work in this regard has been promising, much remains to be done.

5.7 Conclusion

The main focus of this chapter has been to provide evidence for the proposition that the semantics of verbal aspect is grounded in primitives of sensory- motor control. To this end, we outlined a novel computational representation and simulation, inspired by well known perceptuo-motor process features, that captures interesting distinctions made by aspectual expressions while avoiding some paradoxes in standard accounts. Recent work has shown that the representation can deal with particle constructions (such as *eat up, back off*) and temporal adverbials (Narayanan 1997; Chang 1997).

The active, dynamic, highly-responsive nature of x-schemas enables them to model real-time, defeasible inferences. This novel feature of our model distinguishes it from previous attempts to model aspect (Moens & Steedman 1988; Steedman 1995), and allows for a natural solution to the attendant issues of context-sensitivity and inference. We note that (Steedman 1995) proposes the use of dynamic logic to represent the semantics of tense and aspect. In this context, we are exploring the connection between x-schemas and a dynamic version of situation calculus. Other related work shows an equivalence between the multiplicative fragment of linear logic (Girard 1987) and x-schemas. Chapter 10 details this connection further.

5.8 Appendix 5A: Using the Multiple Time Scale Hypothesis For Abstraction

Equations 5.1 - 5.6 below set up this hierarchy formally for different types of situations.

- **Aperiodic Processes**

  The basic idea is if we are interested at a certain time scale s_i, we consider as **Processes**, activities that happen on that time scale ($p_{s_i}(\theta)$). Activities on much smaller time scales are assumed to be instantaneous and qualify as **Events**. Activities on much larger scales are assumed to be unchanging over the episode of reasoning and can be referred to as **Unchanging States**.

  Let $p(\theta)$ be an Aperiodic process whose time scale (start to finish) is s_i . and s_R is a reference time scale of reasoning. The following equations set up the natural hierarchy for aperiodic processes.

$$R : \forall p_{s_i}(\theta) : s_R \gg s_i \Rightarrow \frac{\delta p(\theta)}{\delta t} = \infty. \tag{5.1}$$

$$R : \forall p_{s_i}(\theta) : (s_R \not\gg s_i) \wedge (s_i \not\gg s_R) \Rightarrow -\infty < \frac{\delta p(\theta)}{\delta t} < \infty. \tag{5.2}$$

$$R : \forall p_{s_i}(\theta) : s_i \gg s_R \Rightarrow \frac{\delta p(\theta)}{\delta t} = 0. \tag{5.3}$$

- **Periodic Processes**

  In the case of periodic process, our clustering will have to be done on two axes, namely the frequency and the duration axes. If the Reference time scale of interest is s_R, and the time scale of the process $p_{s_i}(\theta)$ is s_i, and the periodicity parameter is f ($f \in [single, multiple]$); we have the following possible cases:

  1. Short Duration, Multiple Process.
  2. Short Duration, Single Process.
  3. Long Duration Process.

Long duration processes satisfy $s_i \gg s_R$, and **Short** duration processes satisfy $s_R \gg s_i$. Processes that are not **Short** or **Long** duration can be considered aperiodic for the purposes of this discussion, and satisfy Equation 5.1. The interesting case for a periodic process is when the described situation is of a short duration, but can be repeated multiple times. Equation 5.4 shows how this case gets transformed into a continuous process resulting from the *integration* of the multiple occurrences. On the other hand, if the periodic process is of a very long duration, the process gets transformed to a *state* described in Equation 5.6.

1. *Short Duration, Multiple* Processes.

$$R : \forall p_{s_i}(\theta) : (s_R \gg s_i) \Rightarrow -\infty < \frac{\delta \int p(\theta)\delta\theta}{\delta t} < \infty. \tag{5.4}$$

2. *Short Duration, single* Processes.

$$R : \forall p_{s_i}(\theta) : (s_R \gg s_i) \Rightarrow \frac{\delta p(\theta)}{\delta t} = \infty. \tag{5.5}$$

3. *Long Duration* Processes.

$$R : \forall p_{s_i}(\theta) : (s_i \gg s_R) \Rightarrow \frac{\delta p(\theta)}{\delta t} = 0. \tag{5.6}$$

Chapter 6

A Compositional Theory of X-schemas

So far we have looked at single verb schemas and their interaction with CONTROLLER x-schema. In this chapter, we are concerned about extending the representation to be able to model entire domain theories. To this end, we have extended the representation to allow for inter-x-schema *activation* and *inhibition*. We have found the reified CONTROLLER discussed in Chapter 5 useful for this purpose as well, since it effectively decreases the combinatorial coupling possibilities quite drastically, and allows for an elegant theory of x-schema composition.

The central idea behind our compositional theory is that transitions in the controller graph **set**, **get** and **modify** values of the Agent's state (the Agent state f–struct). This allows other x-schemas to be activated, inhibited, or their execution modified in some manner. The model is able to exploit the dynamic, active, and highly-responsive nature of x-schemas. The rationale for the highly compiled representation is obvious since quick responsiveness, tight-coupling between action and reaction, built-in obstacle avoidance and failure recovery strategies, etc. are required for survival while moving around. Our claim is that this degree of reactivity is crucial for language processing, and that we can use the same active sensory-motor representation for language understanding. The next chapter describes our computational model for metaphoric interpretation, which crucially depends on the quick inferences provided by x-schema execution.

The controller graph allows us to distinguish between cases of *sequential* and *concurrent* x-schema triggering or inhibition. Additionally, we are able to model the case where the execution of an x-schema is able to *interrupt* , *terminate* , or otherwise modify the execution trajectory of another x-schema. Interestingly, it appears our design could eventually lead us to new, massively-parallel, fine-grained asynchronous production system implementation.[1] Here an individual x-schema (which we have used so far for representation of actions and concepts labeled by verb phrases) now becomes a fine-grained *production* with internal state (corresponding to the marking of the controller graph), which gets triggered if the Agent State f–struct matches its triggering conditions. Whenever the executing x-schema makes a control transition, it potentially modifies state, leading to asynchronous and parallel triggering of other x-schemas. We believe such a system design supports a broad notion of action, in that the same active representation can be used for motor control and inference. As was explained in Chapter 3 our notion of state is completely distributed over the entire network, so the *working memory* of an x-schema based production system is distributed over the entire set of productions. However, detailed implementation, analysis and conflict resolution in a general purpose x-schema based production system is definitely future work and will not be pursued further here.

An important and novel aspect of our representation is that the same system is able to respond to either direct sensory-motor input **or** other ways of setting the agent state (such as linguistic devices). This allows for the same mechanism to perform simulative reasoning and generate inferences from linguistic input as well as be used for high-level control and reactive planning. We believe that this is an important aspect of embodiment allowing the same mechanisms to reason as well as react. As we show in Section 6.3, the graphical nature of our representation (x-schemas are bi-partite, cyclic graphs) allows us to formally state and reason about inter-schema relations declaratively while using their real-time execution capability for inference. This is a key property of our representation (that it can be viewed as procedural or declarative) that makes it quite different from traditional production systems. We believe this property to be essential for representations that are to be used for both planning and reasoning about plan descriptions.

To motivate and illustrate our representation, we will use an example of embodied

[1]As in many of these instances, the connection between x-schema composition through f–struct modification and production systems is something that Jerry Feldman recognized a-priori.

motion described in Section 6.1. This example is sufficiently interesting and representative to allow us to walk through the central concepts and mechanisms of our compositional theory. We will then formally define the representational primitives of our theory and compare it to related models that have appeared in the Robotics and Cognitive Science literature.

6.1 Motivating Example

Consider the following example, which we will use to illustrate the central concepts underlying our compositional theory.

Example 11 *During* a WALK, and *while taking* a STEP, if you encounter an unanticipated **bump**, or if the ground is **slippery**, you become *unsteady* which *leads to* you to TRIP. This *may lead* to a FALL unless you are able to *simultaneously expend energy* and STABILIZE, in which case you may *resume* the *interrupted* STEP . If you are unable to STABILIZE, and thus FALL, you will be **supine** and **hurt**. In order to *start walking* again you will have to GET UP and **be standing** again. ■

This example will be used in detail throughout the chapter to illustrate the issues involved. To encode the example above, we need to be able to model each of the *italicized* inter-schema (x-schemas are in UPPERCASE) relations which are mediated by the state variables in **boldface** above. In general, we need to be able to link individual verb x-schemas together to yield a compiled x-schema representation of the embodied domain. We first exhibit the mechanism for inter-schema relations by going through our model of this example in detail. We will then be in a position to generalize and formally define the various inter-schema interactions involved.

6.2 Using X-schemas For Embodied Domain Inferences

Figure 6.1 shows the initial state of the x-schemas that should be seen as depicting just those parts of the knowledge base needed to describe the example. The Agent State f–struct is a vector of feature-value pairs. The state is distributed over the entire network. A feature takes on a specific value with the presence of a token of a specific color. For simplicity, the example shown in Figure 6.1-Figure 6.11 uses only binary valued world-state variables and integer valued resource variables (like energy). In the next chapter, we will see how the use of typed tokens carries variable binding information about specific objects and entities and allows us to distinguish different walking agents, etc. In Figure 6.1, the presence of a token indicates that the relevant condition is true, a number inside the place signifies an integer value of the number of tokens at the relevant place. Note that in the example shown, there are four schemas, namely WALK (with a STEP subschema), FALL , STABILIZE , and GETUP). Note also that the TRIP x-schema is modeled as a simple transition.

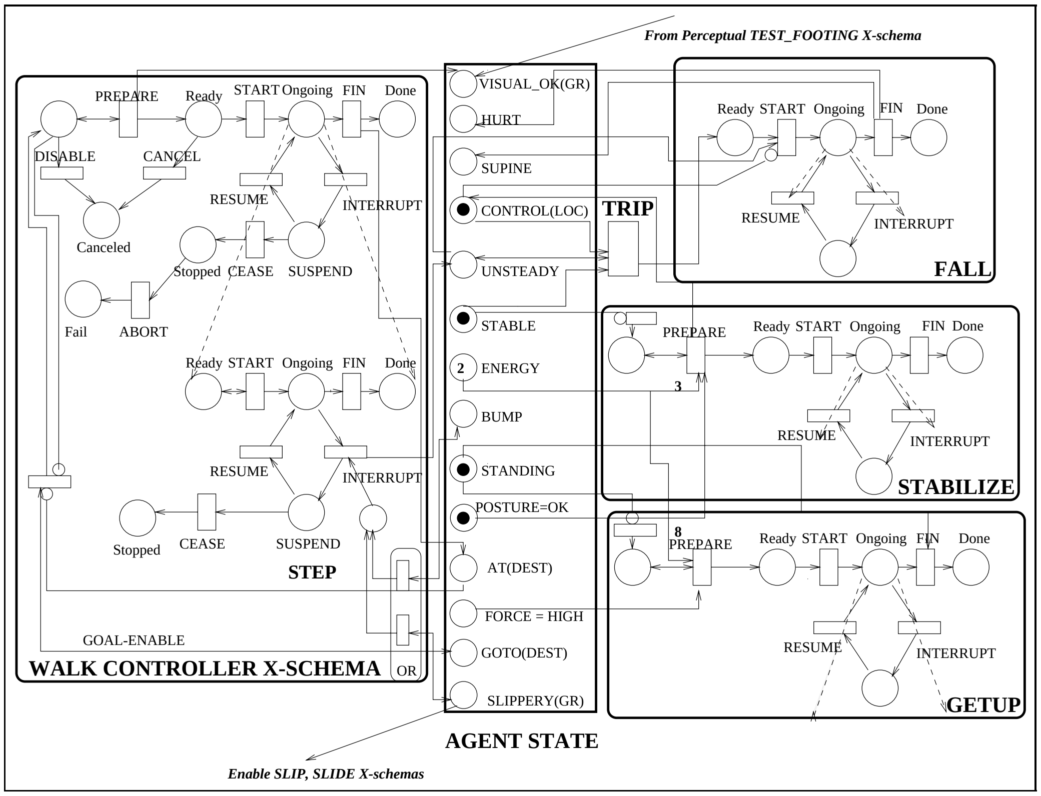


Figure 6.1: The initial state and x-schemas modeling the domain theory shown in the example.

Figure 6.1 shows the wiring structure and initial state. As explained in the last section, specific resources, world-state and agent force-dynamic and emotional state may trigger, inhibit, modify, or be produced as a result of a transition in the CONTROLLER x-schema. For instance, the TRIP transition results in loss of control of one's location (signified by the consumption of a token from the *control*(*loc*) state variable), and occurs as a result of some instability. Such instability may be produced by the presence of a **bump** causing an ongoing STEP to be interrupted. Note that the agent has some reserves of energy; in this case, 2 units, signified by the integer **2** at the appropriate state variable. Initially, the agent is assumed to be in control of his location, indicated by the presence of a token at the *control*(*loc*) state variable, standing (token at *standing*), and stable (token at *stable*).

Figure 6.1 also shows the expansion of the *ongoing* place of the WALK controller to the associated subnet as was discussed in Chapter 3. Remember that the notation of having an *ongoing* place is realized by hierarchical transitions representing the invocation of a sub-schema.

The agent has a *goal* to be at a location *DEST* , and when that location is reached, there will be a token at the state variable *AT*(*DEST*) . The *absence* of such a token, i.e. not being at a destination, is sufficient to enable methods of getting to that destination. This is an instance of *goal-based* enabling, which basically allows for goals to enable x-schemas if they are in the Agent's state. The link from the *finish* transition of a walk automatically sets the goal state of being at the destination, and the walk x-schema (and other x-schemas whose sole enabler was the *AT*(*DEST*) variable) are disabled.

Figure 6.2 shows the WALK x-schema enabled as a result of the goal not being satisfied. We adopt a convention enabled transitions and input (output) arcs are shown in heavy **boldface**.

In the presence of a *goal* to *GOTO* (*destination*), if the WALK x-schema is not already *enabled* *and* if the agent is not already *AT*(*destination*) , then the WALK x-schema becomes *enabled* . Once the destination is reached, i.e. a token is present at the *AT*(*DEST*) node in the Agent's State, all x-schemas that became enabled due to the *GOTO*(*DEST*) goal become disabled. Any residual token at an *enable* place can *timeout* through the *disable* transition. The relevant x-schema will not be enabled again until the goal of reaching some destination is asserted and the agent is not at that destination.

In general, *goals* are special type of state variables. Goal assertion is a method to *enable* the relevant schemas. However, they are also *resources* in that the successful completion of the relevant schema consumes a token from the goal variable, thus retracting the requirement to satisfy the goal. Thus, there is a *resource* link from the goal to the *finish*

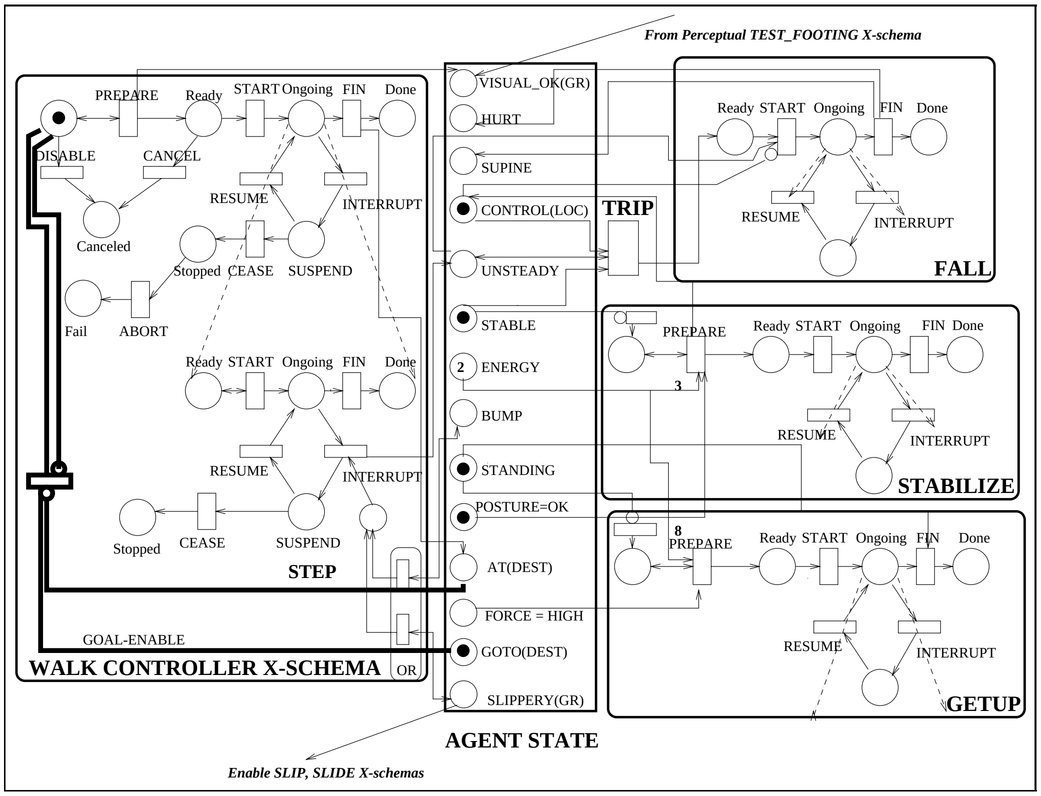


Figure 6.2: Not being at the destination enables an unenabled walk x-schema.

transition of the relevant enabled x-schemas (omitted from the figure for the WALK x-schema for legibility).

Goal-enabling transitions are instantaneous and so transition enabling and token transfer take place in a single step of the simulation. The **bold** lines show the enabled transition, its input and output links and the enabling of the WALK x-schema that continues to be active until goal attainment.

An important aspect to note is that *either* goal retraction *or* goal attainment (being at the destination) could be done by any specific enabled x-schema (or asynchronously asserted by some higher level process such as specific linguistic input), and the system will react appropriately by disabling all the associated x-schemas.

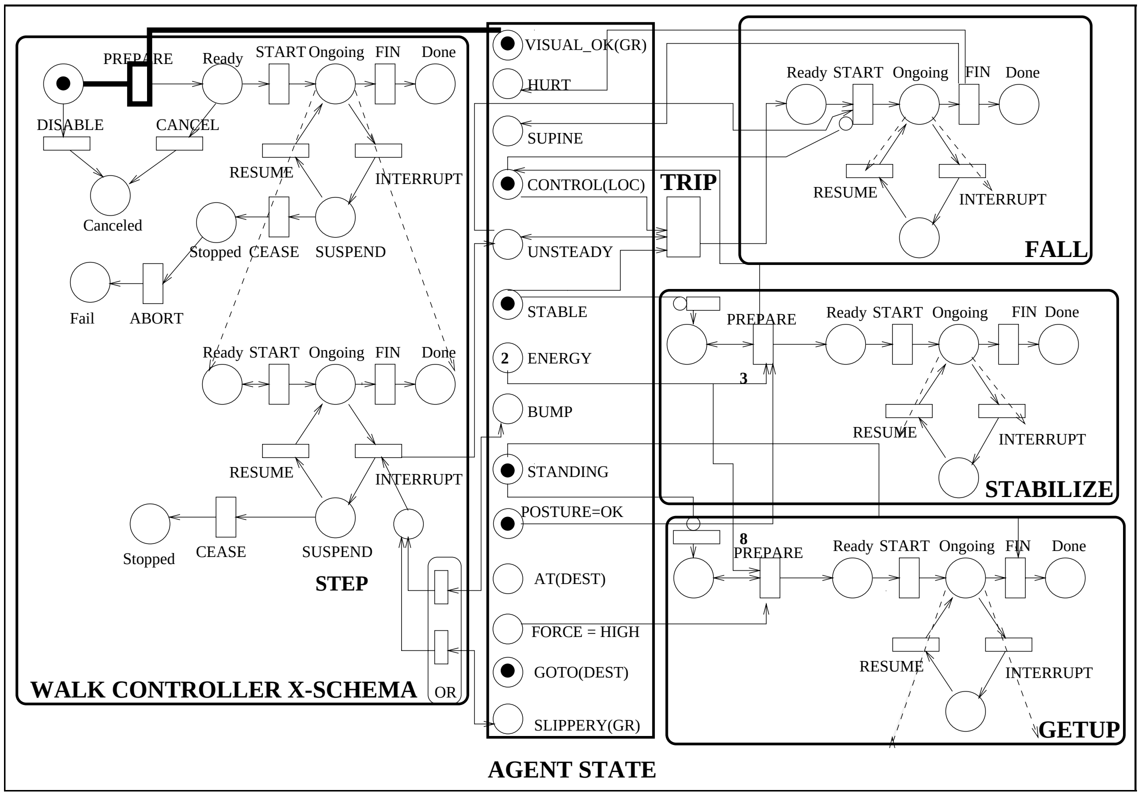


Figure 6.3: The WALK schema is enabled, perceptual test returns *ok* and the WALK is *ready* .

Figure 6.3 shows the WALK controller executing the preparatory steps for a walk. The **bold** links specify either an input to a control transition that has the required number and type of tokens, or an output link of an **enabled** transition which has received a token of the appropriate type. So, in Figure 6.3, the $VISUAL\_OK$ has returned an *ok* status thus the link from that place to the PREPARE transition is shown in boldface. Also, since all the preparatory steps are completed, the prepare transition is ready to execute the *next_step* function described in chapter 2.

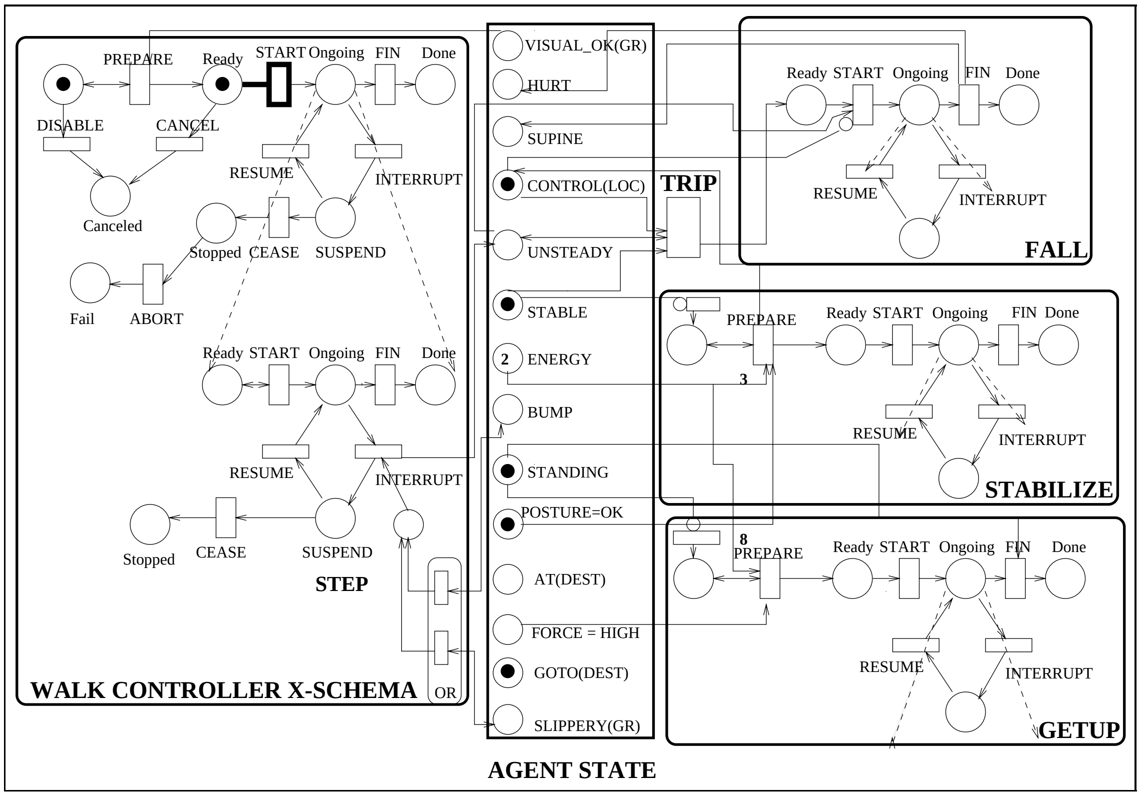


Figure 6.4: Starting to Walk

Once the preparatory stage has been completed, the **walking** starts. The START transition signifies this change. Figure 6.4 shows the start of walking. Note that there is still a token at the *enable* place (signifying that the WALK x-schema is enabled) and the *start* transition is enabled, indicating the start of walking event. Recall that usually the *start* transition is by itself an atomic event, but not always, since it may also have sub-schemas indicating a starting subsequence. When the starting process is complete, the token is transferred to the place labeled *ongoing* .

Once the walk transition is *ongoing* , individual steps are taken. The mechanism is exactly as described in chapter 3 and Chapter 5 . The presence of a token in the *ongoing* state of the WALK remains until the destination is reached (or the walk stops for some other reason). The token in the STEP sub-schema circulates and iterates every step. This is the situation shown in Figure 6.5 and Figure 6.6.

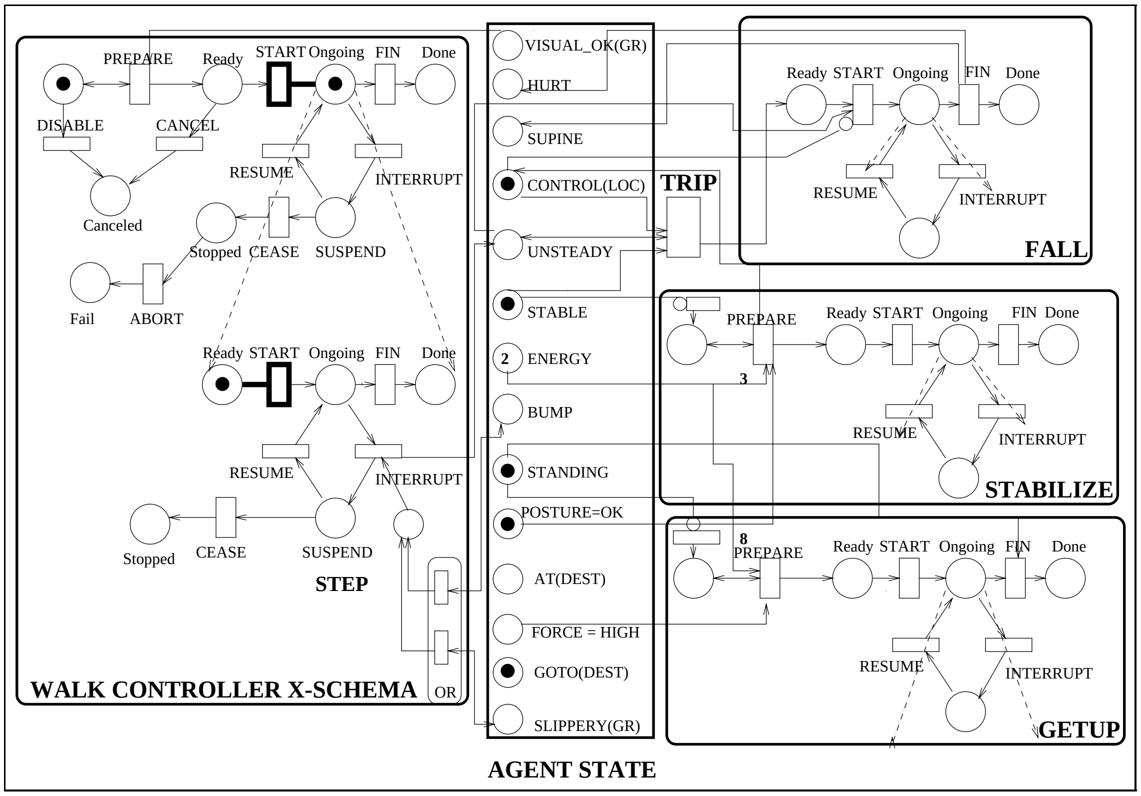


Figure 6.5: An *ongoing* walk calls the *sub-schema* corresponding to a STEP .

At some point, suppose a bump is encountered. Then the presence of an *ongoing* STEP combined with the condition signifying the presence of a bump leads to the situation in Figure 6.7.

The situation in Figure 6.7 pertains to the agent taking a step, shown by the token in the ongoing controller node of the WALK x-schema and of the subschema STEP . Note that the presence of a **bump** enables the interrupt transition of the STEP , which translates to interrupting the WALK x-schema as well. The situation depicted shows an the presence of a bump during an ongoing step. Note that bump is a world condition that has bi-directional arrow to the transition that leads to interrupting a walk (and may

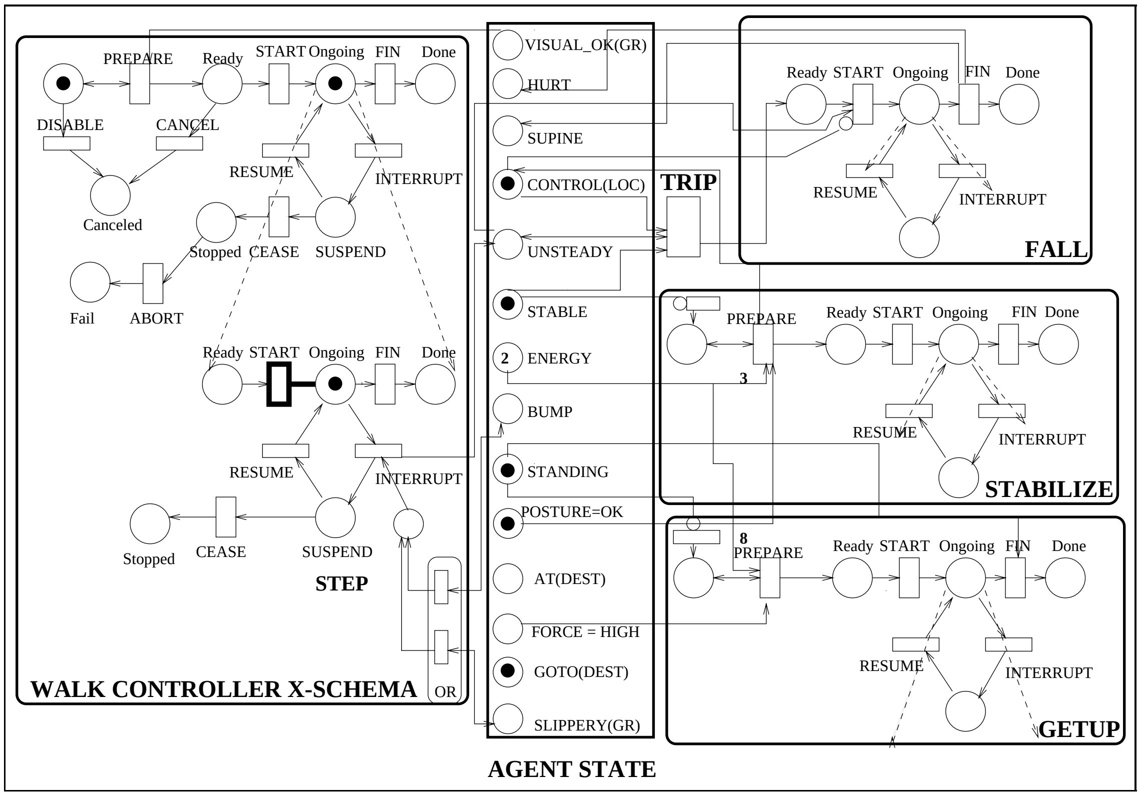


Figure 6.6: STEP is now *ongoing* as is WALK .

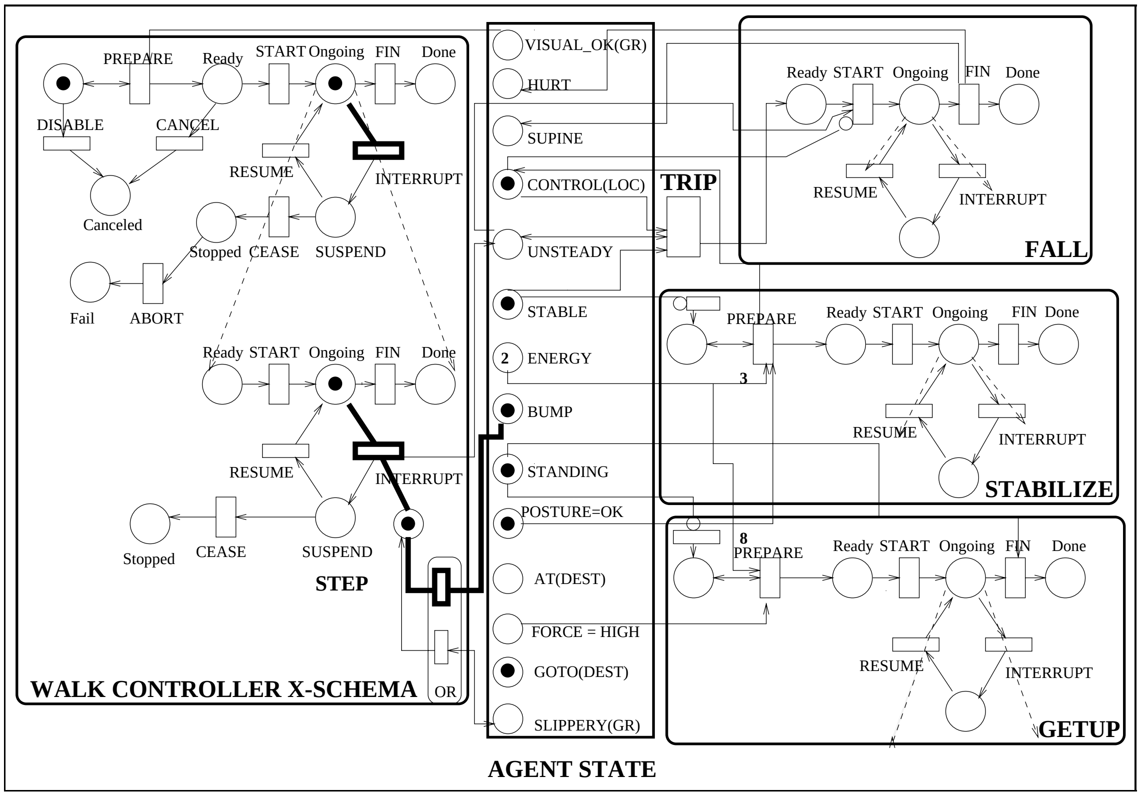


Figure 6.7: Simulating a bump while executing a step

have other effects as well not shown in the example), implying that it is an pre-condition to the transition. There may be many such world and agent conditions that may result in interrupting the WALK x-schema, including the ground being slippery, etc. Thus the rounded edge box implements the OR of its inputs. Also, we note that the transition to interrupt the walk is instantaneous (others have a default duration of 1, unless otherwise specified). Thus the transition enabling and token appearance is done in a single step of the simulation.

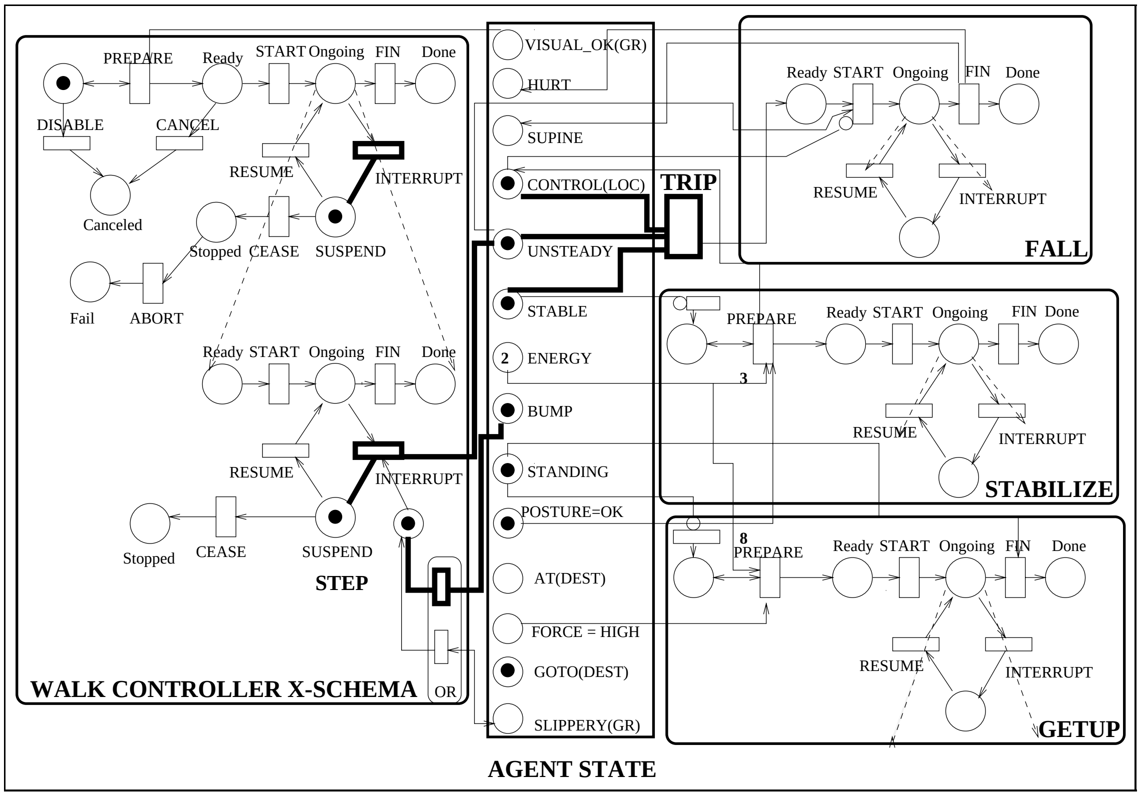


Figure 6.8: The resulting instability and possible tripping.

Once the interrupt transition fires, the agent state is now changed, to reflect the resulting instability. This is the situation depicted in Figure 6.8. The TRIP x-schema now becomes enabled (which is shown as just a transition, with no internal structure to show the different possibilities). The firing of this x-schema results in consuming a token from the agent state corresponding to controlling one's location, indicating that the agent is no more in control of his location. Also, we note that the agent continues to be unstable after tripping.

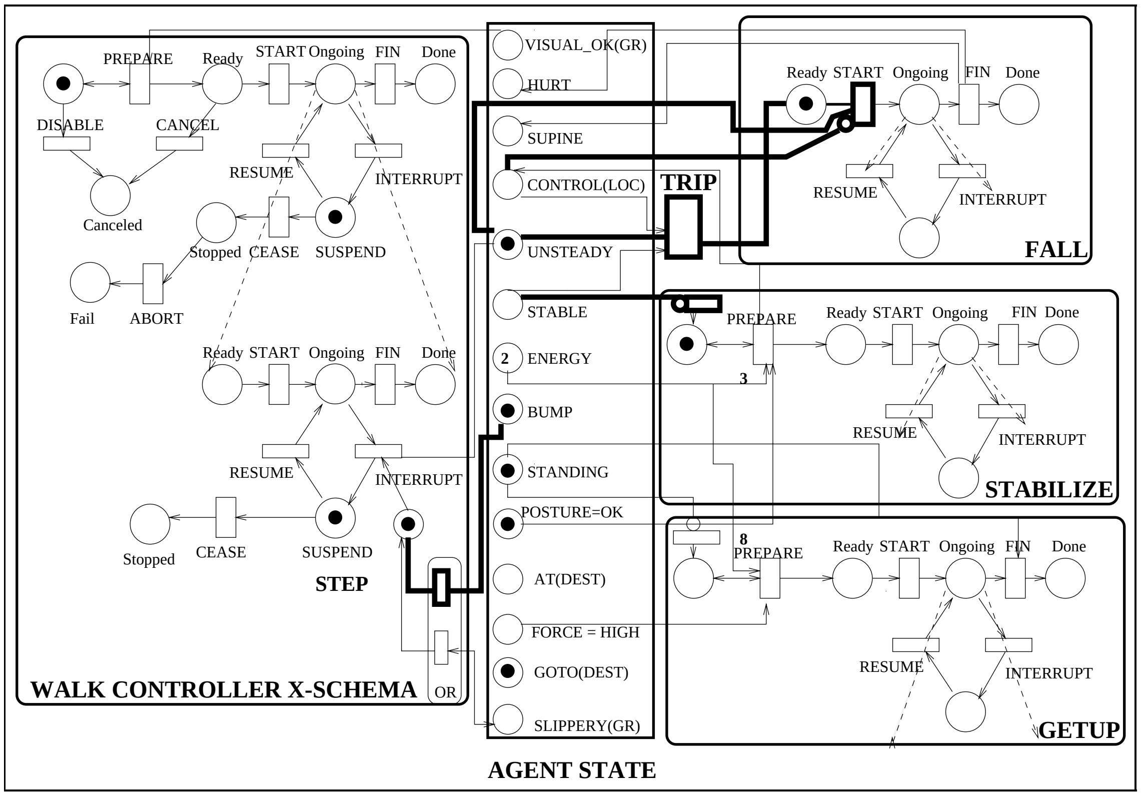


Figure 6.9: To STABILIZE or FALL ?

Tripping leaves the agent unstable, and acts as a goal to enable the STABILIZE x-schema. Intentional schemas can be enabled by asserting the need to satisfy a goal. In this case, removing stability results in the continuous enablement of the STABILIZE x-schema (of course, if there are multiple ways to satisfy a goal, the corresponding x-schemas become enabled as well).[2]

An important thing to note about our model of goal-enablement is that if some other process retracts the goal, the corresponding x-schemas cease to be enabled, and may

[2]In current model, the default conflict resolution strategy is **free choice**(unbiased coin-toss). At the neural/connectionist level, the response is inherently graded and conflict resolution works by enabling x-schemas to different degrees of activation and using winner-take-all networks to resolve multiple active schemas. At the computational level, we hope to use the stochastic nature of the x-schema transitions to implement conflict resolution strategies where the rate of transition firings is determined by sampling from an exponential distribution. This way we can set stochastic priorities to transition firings. The implementation details and analysis of such a scheme is not addressed in this thesis.

timeout based on the desired responsiveness and the details of the design.

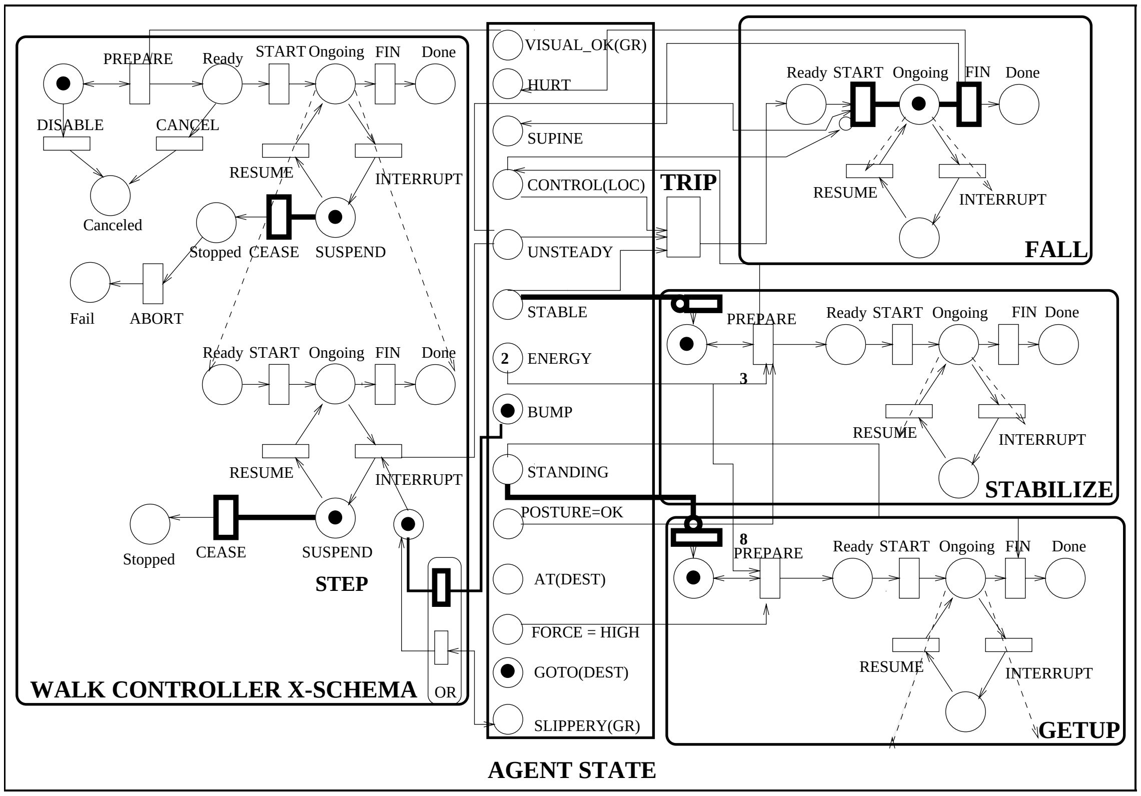


Figure 6.10: Not enough energy or control to stabilize

The enabled STABILIZE schema is unable to execute, since the agent does not have sufficient energy or postural control, hence the FALL schema starts executing; the agent loses control and in the absence of any other interrupting source, becomes **supine** and **hurt** as a result of the fall. This is the situation depicted in Figure 6.10.

As a result of the fall shown in Figure 6.10, the agent is not standing ($\neg standing$) anymore which enables the GETUP schema, which cannot execute currently, since there are insufficient energy resources. At this point some other energy producing source may be triggered using the same goal-based enabling mechanisms discussed earlier, to be able to execute the GETUP schema. But this is not modeled in the simple illustrative example.

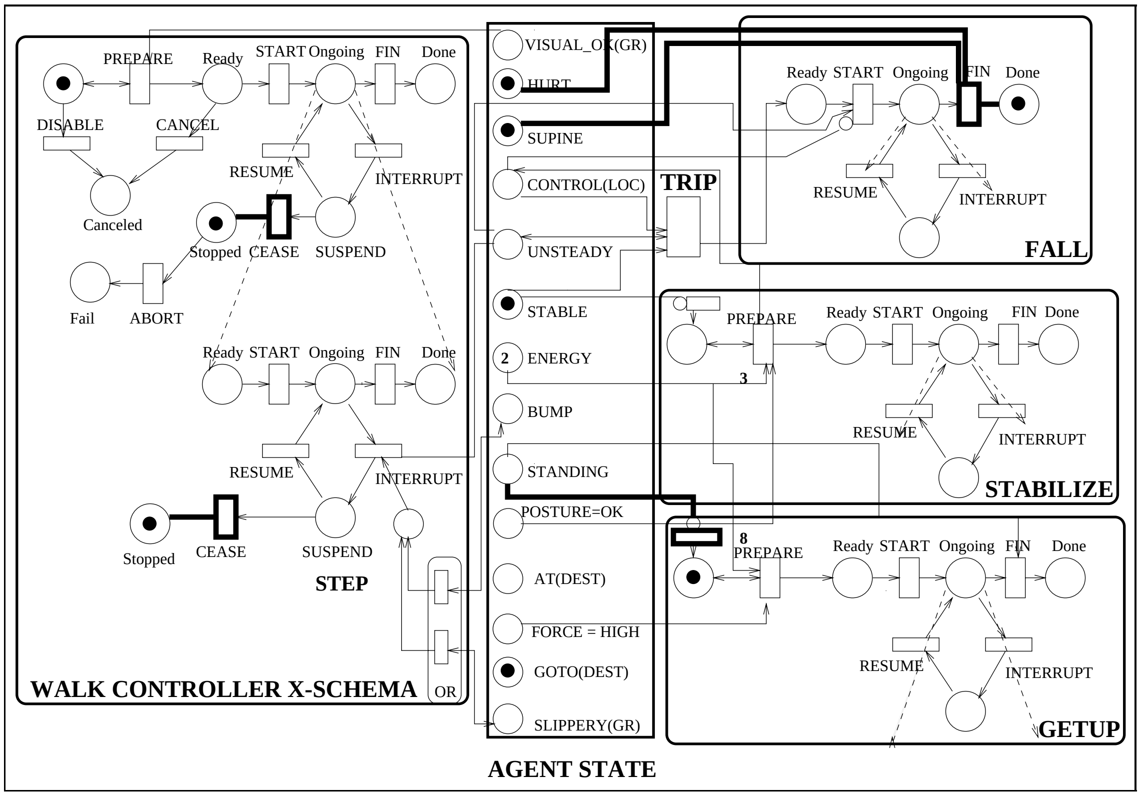


Figure 6.11: After the Fall one must Get Up.

6.3 Generalization and Formal Definitions of Inter X-schema Relations

We have seen in detail how x-schemas are able to be triggered, inhibited and have their execution state modified as a result of control transitions of other x-schemas. We are now in a position to formally catalog the various inter-schema relationships in a general way that allows us to model knowledge of the embodied domain.

Importantly, the fact that x-schemas are graphical structures allows us to formalize the various inter-schema relations and come up with a compositional theory. We strongly believe that such a structured representational system is essential for planning as well as reasoning about plan descriptions. Completely procedural representations such as production systems often lose this ability since they do not allow inspection or manipulation of internal state. In contrast, as we will see in this section, the graphical nature of x-schemas (they are bi-partite, cyclic graphs) allows us to state and reason about interesting inter–schema interactions and their compositions.

As we have seen in the last section, x-schemas are connected to each other through three broad sets of relations, namely **Activating**, **Inhibiting** and **Modifying** relations. Since any control transition of one x-schema can potentially change the Agent's state f–struct in a way that may cause a change in the control status of some other x-schema, there are potentially N^2 possible relations between any two x-schemas, where N represents the kinds of control transitions ($N \approx 10$, so the number of relations between any two schemas is $N^2 \approx 100$). Of course this is quite small compared to the combinatorial possibilities without the *controller* abstraction, which is potentially M^2 , where M is the number of states of an individual x-schema.

In this section, we will catalog the general definitions of these relations and define some of the more important connections.

Important **Activating** relations include
Hierarchical Enabling, Sequential Enabling , Concurrent Enabling
Mutually Enabling, Inherent Periodicity, Resource generating.
Important **Inhibitory** relations include
Sequential, Concurrent Disabling, Mutually Disabling
Inherently Aperiodicity, Mutual Exclusivity, Resource consuming.
Important **Modifying** relations include
Interruption, Prevention, Termination, and Stoppage.

Activation can be partial in that some of the preconditions of the activated schema are **set** as a result of executing the activating x-schema. If all the requisite preconditions and resources of the activated x-schema are **set**, we say that executing the first x-schema *enables* the second. Note that in contrast to other theories, we are able to distinguish *concurrent* from *sequential*n activation.

Inhibiting links prevent execution of the inhibited x-schema by **setting** an inhibitory input place. Again, our model is able to distinguish between concurrent and sequential inhibition as well as be able to model mutual inhibition and aperiodicity.

Modifying relationships between x-schemas occur when the execution of the modifying x-schema results in setting the Agent State in such a way the the currently active modified x-schema undergoes a **controller** state transition. For instance the execution of the modifying x-schema could result in the interruption, termination, resumption of the modified x-schema.

Example 12 Some examples of inter x-schema links

Figure 6.12 shows some implemented inter x-schema links. The first three examples are *activation* links, the following two examples *inhibitory* links and the final two examples are *modifying* links between the x-schemas shown. In each case, the required resource or enabling Agent state f–struct value is shown immediately below the link itself. For example, we have seen how Tripping results in the agent losing control of his own location ($\neg in\_control(loc)$), which leads to enabling a possible $fall$. This is the first example below. All the other examples are self-explanatory and should be fairly obvious to the reader. The interesting examples are the case of concurrent enabling and disabling and **dynamic** resource based enabling as in the case where the production of a specific amount

TRIP *sequentially enables* FALL and STABILIZE .
  $\neg$*in_control(loc) enables* FALL
  $\neg$*stable goal-enables* STABILIZE
GRASPING(X) *concurrently enables* HOLDING (X).
  in_hand(x) enables HOLD(X)
PRODUCE(ENERGY, X) is a *resource-producer* for WALK.
  Energy(x) is a *resource* for WALK
HOLDING(X) *concurrently disables* GRASP(Y)
  $\neg free(hand) \wedge in\_hand(x)$ *disables* GRASP(Y)
WALK is *mutually-exclusive* to RUN.
STABILIZE *sequentially disables* FALL.
  in_control(loc) disables **Fall**
GETUP *resumes* WALK.
  standing enables WALK
REACH(X) *terminates* WALK(X).
  at(x) disables WALK(X)

Figure 6.12: Some implemented inter x-schema links

of energy ($Produce(energy, x)$) enables the **Walk** x-schema (which requires the specified amount to be able to start).

■

Definition 18 Activation

X **activates** Y ($(X, Y) : X, Y \in \mathcal{SS}$) if executing X marks some subset p of either *pre-condition* or *resource* places to the *prepare* transition ($T(Y_e^+)$ of Y ($p \subseteq *T(Y_e^+)$). If the *result* state $P(X_r)$ of X marks p ($p \subseteq P(X_r)$), then we say X **sequentially activates** Y . If the *process* stage of X activates Y ($p \subseteq P(X_p)$), we say X **concurrently activates** Y . X **enables** Y if ($p \supseteq *T(Y_e^+)$). Both **activates** and **enables** are *non-transitive*, i.e. $activates(X, Y) \wedge activates(Y, Z) \not\Rightarrow activates(X, Z)$ (since Y may never execute, or never complete execution). If $activates(X, Y) \wedge activates(Y, X)$, then we say X and Y are **Mutually enabling**. For **inherently iterative** activities $*T(X_e^+) \subseteq P(X_r)$.

A special class of activation is those relationships that explicitly result in tokens being placed at the **enable** input places to control transitions (compared to adding specific resources). Figure 6.13 depicts some of the more important relationships defined below.

Definitions of some important **Activation** relationships between x-schemas.
Refer to Figure 6.13 for a Graphical depiction.

$$\mathbf{seq-enables}(X,Y) : done(X) \wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_e^+))(Figure\ 6.13a))$$
$$\mathbf{conc-enables}(X,Y) : ongoing(X) \wedge (p \in P(X_p)) \wedge (p \subseteq *T(Y_e^+))(Figure\ 6.13b))$$
$$\mathbf{inh-periodic}(X) : \mathbf{seq-enables}(X,X)(Figure\ 6.13c))$$
$$\mathbf{mut-enables}(X,Y) : \mathbf{seq-enables}(X,Y) \wedge \mathbf{seq-enables}(Y,X)(Figure\ 6.13d))$$
$$\mathbf{res-producer}(X,Y) : done(X) \wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_r^+))$$

Definition 19 Inhibition

Just as with activating relationships, the presence of a token in a specific inhibition place can inhibit the triggering of an x-schema. Here we will be concerned mainly with explicit inhibition through activation the some *inhibitory* place to an x-schema. Such relationships will be called **disabling** relationships.

X **disables** Y ($(X,Y) : X,Y \in \mathcal{SS}$) if executing X marks some subset p of *inhibitory* places to the *Enable* transition ($T(Y_e^-)$) of Y ($p \subseteq *T(Y_e^-)$). If the *result* state $P(X_r)$ of X marks p ($p \subseteq P(X_r)$), then we say X **sequentially disables** Y. If the *process* stage of X disables Y ($p \subseteq P(X_p)$), we say X **concurrently disables** Y. If If $disables(X,Y) \wedge disables(Y,X)$, then we say X and Y are **Mutually disabling**. For **inherently aperiodic** activities (**?**) $*T(X_e^-) \subseteq P(X_r)$. **Disables** is *non-transitive*,i.e.

$$disables(X,Y) \wedge disables(Y,Z) \not\Rightarrow disables(X,Z) \tag{6.1}$$

Since Y cannot execute, X actually weakly *activates* Z.

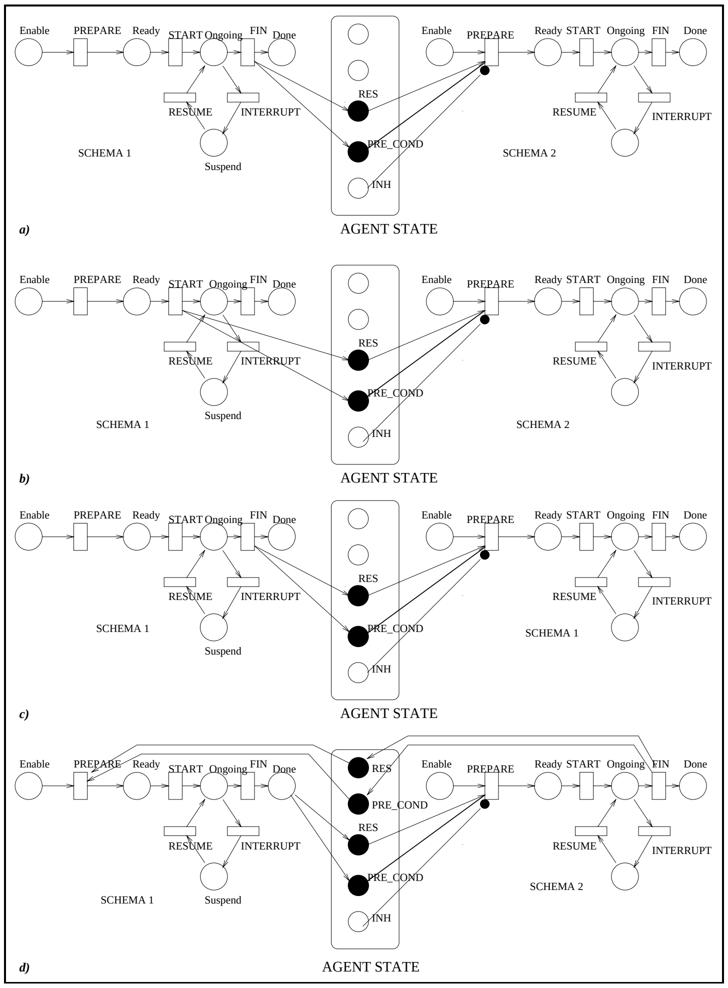


Figure 6.13: A sample of the various types of activation relationships between x-schemas. In the concurrent activation case, the link between schemas is an enable link. In this case, once Schema 2 is enabled, the execution of Schemas 1 and 2 proceeds independently. Note that in all case **only** the controller state of the individual schemas are shown, the reader is to assume that each of the schemas has internal structure not shown in the figure.

Definitions of some important **Inhibitory** relationships between x-schemas
Refer to Figure 6.14 for a Graphical depiction.

$$\begin{aligned}
\mathbf{seq-disables}(X,Y) &: done(X) \wedge (p \in P(X_r)) \wedge (p \subseteq *T(Y_e^-))(Figure\ 6.14a))\\
\mathbf{conc-disables}(X,Y) &: ongoing(X) \wedge (p \in P(X_p)) \wedge (p \subseteq *T(Y_e^-))(Figure\ 6.14b))\\
\mathbf{inh-aperiodic}(X) &: \mathbf{seq-disables}(X,X)(Figure\ 6.14c))\\
\mathbf{mut-disables}(X,Y) &: \mathbf{seq-disables}(X,Y) \wedge \mathbf{seq-disables}(Y,X)(Figure\ 6.14d)\\
\mathbf{mut-exclusive}(X,Y) &: \mathbf{conc-disables}(X,Y) \wedge \mathbf{conc-disables}(Y,X)
\end{aligned}$$

Definition 20 Modification

The following relations pertain to schema X modifying the execution of schema Y. While in the implemented model, only the *result* state of Schema X ($P(X_r)$) is responsible for the modification, obviously the links could be from other stages as well (for instance, starting X could interrupt and ongoing Y, etc.).

Definitions of some important **Modifying** relationships between x-schemas.
Refer to Figure 6.15 for a Graphical depiction.

$$\begin{aligned}
\mathbf{interrupts}(X,Y) &: ongoing(Y) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_{int}^+))(Figure\ 6.15a))\\
\mathbf{prevents}(X,Y) &: enabled(Y) \wedge \neg start(Y) \wedge (p \in P(X_r)) \wedge (p \in *T(Y_s^-))(Figure\ 6.15b)\\
\mathbf{terminates}(X,Y) &: ongoing(Y) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_f^+))(Figure\ 6.15c)\\
\mathbf{resumes}(X,Y) &: suspended(Y) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_{int}^+))(Figure\ 6.15d))\\
\mathbf{stops}(X,Y) &: (suspended(Y) \vee ongoing(Y)) \wedge (p \in P(X_r)) \wedge (p \supseteq *T(Y_{cease}^+))
\end{aligned}$$

Theorem 21 Composition

X-schemas are closed under a finite number of compositions using the relations defined earlier.

Proof (sketch): We use the methods of converting x-schemas to Petri nets in Chapter 3 and the following well known result from Petri net theory (Peterson

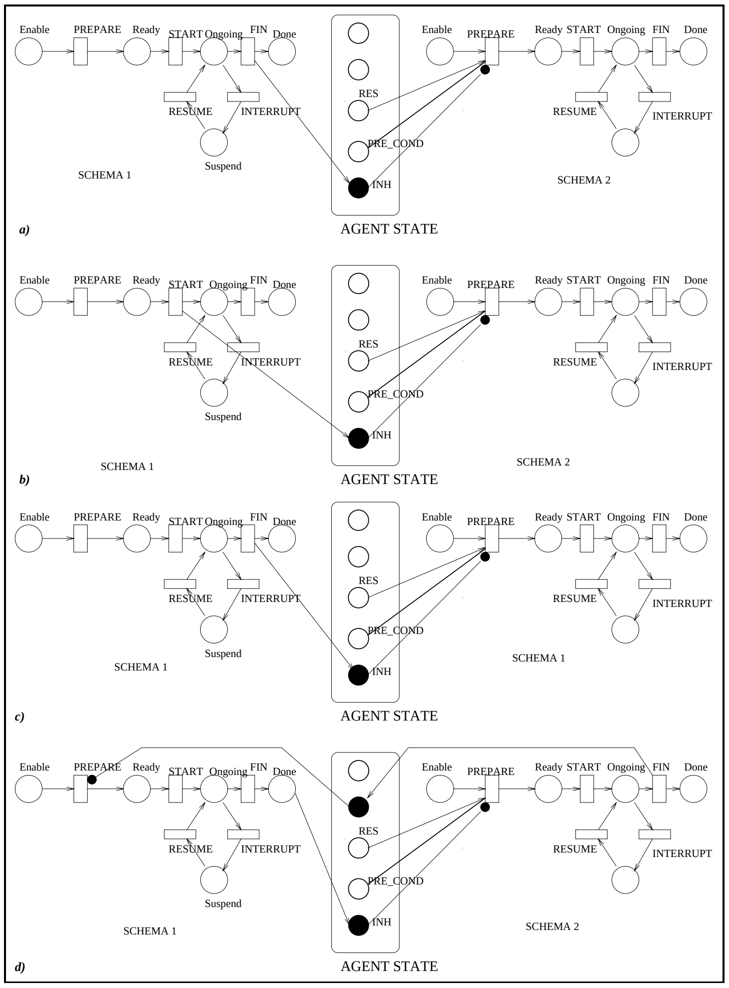


Figure 6.14: A sample of the various types of inhibitory relationships between x-schemas. In contrast to the case of activation, note that if Schema 2 is disabled during execution of Schema 1 , it continues to be disabled until Schema 1 has completed execution or stopped due to some other interrupt.

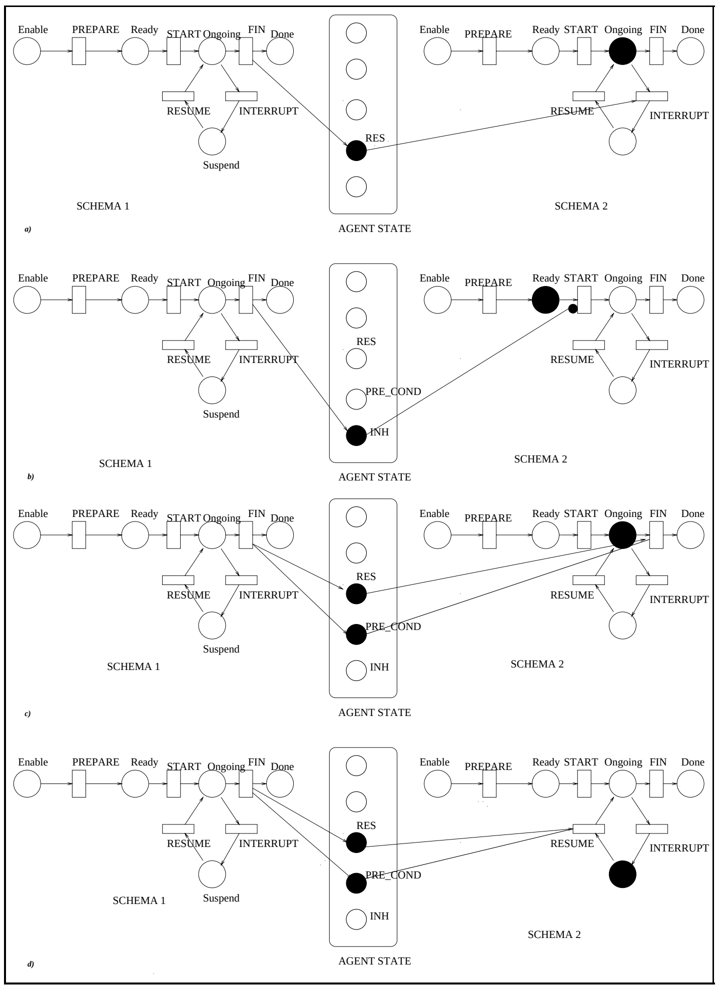


Figure 6.15: Relationships between executing x-schemas The relationship asserted depends both on the execution state (marking vector) of Schema 2 as well as the structural linking property. Only the sequential case is shown, where the the modification occurs as a result of execution the modifying schema to completion. Of course, for all the relationships shown, one could model the concurrent modification case as well, where the once the modifying schema executes, it enables or disables specific transitions of the modified schema by setting the appropriate state variables.

1981, Crespi-Mandzioli 1989) to conclude that under finite number of repeated compositions, we still get a bounded x-schema. In conjunction with Section 3.10, the following result does the rest. ∎

Lemma 22 *(Peterson 1981) Petri net languages are closed under a finite number of applications, in any order, of the operations of union, intersection, reversal, concurrency, and concatenation.*

6.4 Conclusion

In this chapter we set up a computational framework for implementing embodied theories. Our representation has two important and novel features. First, our compositional theory retains the highly compiled, dynamic and active nature of x-schemas to implement an active model of embodied and familiar domains. Second, we showed how the **controller** x-schema abstraction, developed in Chapter 3 and used for interpreting expressions of linguistic aspect allows us to construct a general purpose theory of x-schema composition. Both these features allow us to use a uniform mechanism for sensory-motor control and for real-time simulative inference.

Of course, it remains to be seen why such a reactive, dynamic system representation is crucial for metaphoric reasoning about international economic policies. But before getting there, we need to explain the ontology and our representation of the abstract domain of international economics since the implemented program processes simple stories from this domain. The next chapter deals with this issue. Chapter 8 discusses the implemented model and goes through the working of the program in detail. Both these chapters are fairly technical and serve to document details of the implemented program. Readers who are more interested in what the model can do and the cognitive relevance of the enterprise described in this thesis can read the first three sections of Chapter 8 and directly jump to Chapter 9, where we lay out the actual performance of the program on our database of stories and discuss the wider relevance of our model.

Chapter 7

A Belief Net Model of International Economics

Recall that our system is to interpret simple newspaper stories that use embodied terms to describe features of international economics including economic policies and the result of their implementation on the impacted entity. To this end, we need a representation that can capture the ontology and structure of the domain of international economics at a level comparable to a non-expert newspaper reader.

The structure of the abstract domain (the domain of international economic policies) encodes causal domain knowledge about Economic Policies. The crucial requirement here is that our representation be capable of modeling causal knowledge and be able to support both belief **updates** and **revisions** in computing the global impact of new observations and evidence both from direct linguistic inputs and from inferential products of x-schema execution. Our implementation of the agent's conceptual structure of the causal theories of the target domain uses Probabilistic Independence Networks (PIN) (Pearl 1994; Darwiche 1994; Jensen 1996).

A Probabilistic Independence Network is a convenient data structure to encode causal domain knowledge. The basic algorithms that operate on the probabilistic network data structure deal both with new observations and database updates due to external intervention such as actions and random disturbances. We also note that a major advantage of choosing probabilistic networks to represent knowledge about Economic Policies is that there is a long well-known history of integrating such networks with decision nodes to produce decision networks and influence diagrams. This allows our architecture to seamlessly integrate with current computational models in Economic Policy design and analysis, and an interesting future study would involve seriously considering this possibility.

We start by describing the theory of *PIN* 's and the basic algorithms that operate of such structures to propagate evidence and return the most probable explanation for the observed data. This section is mostly for completeness and readers who already know about Probabilistic Independence Networks can safely skip the section and go directly to the section on the ontology of the target domain.

7.1 Probabilistic Independence Networks

Probabilistic Independence Networks consist of a set of variables and a set of directed links (though undirected networks are also used, our exposition is limited to the directed case) between variables. The structure of a network is that of a *DAG* (Directed Acyclic Graph). In describing the structure of such networks, it is customary to use the language of family relations such that if there is a link from variable A to variable B in the network, B is referred to as the child of A, and A as the parent of B. Similarly, children of one parent are called siblings of each other.

PIN's have become the mainstay in AI systems that represent and reason with uncertain knowledge, and there is a large literature on the subject. Also, it is now generally expected that most undergraduate courses in AI devote a sizable amount of time to the algorithms and technology of PIN's. The exposition here is therefore quite brief, and is intended to provide readers with just enough knowledge to be able to understand the operation of the Metaphor Reasoning System. In general, I follow the excellent books of (Jensen 1996; Pearl 1988), and the interested reader is referred to these texts for further information and pointers into this important area of research.

Variables in a PIN are events or propositions that can take a number of states. For example, a variable in a PIN could be *disease* (taking on values influenza, bronchitis, tuberculosis, cancer), or more relevantly for us a country's specific *economic–policy* (taking on values Autarky, Import Substitution, Protectionist, Liberal, Export Push). Thus a variable in a PIN represents a state of affairs. A variable is in exactly one of its possible states, thus the values of a variable are mutually exclusive, but which state may be unknown to us.

The most important aspect of PIN's is their ability to exploit the independence relationships between variables. This is a structural property of such networks and is thus not dependent on the specific quantitative uncertainty combination method used. The key structural property is called the property of *d-separation*. In this exposition I follow the recent text by Jensen (Jensen 1996) closely in describing this property.

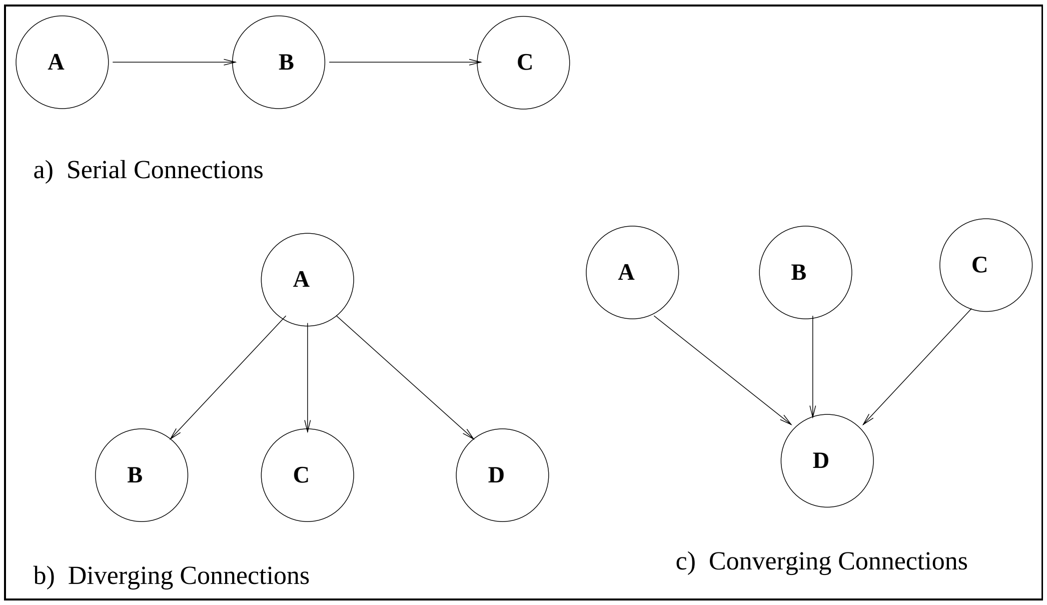


Figure 7.1: The three types of inter-variable connections. In a) if the variable B is instantiated (when the state of B is known through observation as evidence, or otherwise clamped) A and C become independent. In b) (the divergent case), if A is instantiated it blocks communication between its children (B , C and D) and this B , C and D become independent if the state of A is known. In c), if nothing is known about D , except what is known about D 's parents A , B and C , then the parents are independent. However if there is some evidence that sets the state of D (or any of D 's descendents) then the parents are no more independent.

Figure 7.1 illustrate the different kinds of structural relationships between connected variables in a DAG . In general, the structural properties allow us to conclude the following.

1. Evidence transmission is *blocked* in a serial chain when the state of the intermedi-

ate variable is known. The variables on either side of the clamped variable become *independent* when this happens.

2. Evidence transmission is *blocked* in a divergent connection when the parent variable's state is known. The various sibling variables become *independent* when this happens.

3. Evidence transmission is *blocked* in a convergent connection **until** the state of the connecting variable or that of any of its descendents is known. Until this situation happens, the various parent variables are *independent.*

Definition 23 d-separation(Jensen 1996) Two variables A and B are d-separated if for all paths between A and B, there is an intermediate variable V such that

- the connection is serial or diverging and the state of V is known or
- the connection is converging and neither the state of V or that of any of V's descendents are known.

7.2 Bayesian Networks and Causal Theories

Reasoning about uncertainty in PIN's also requires a quantitative calculus to calculate and combine uncertainty measures. In our work, we use a Bayesian calculus which comes from classical probability theory. PIN's that use Bayesian calculus to reason about variable values are called Bayesian Networks.

Definition 24 Bayesian Network A Bayesian Network consists a set of variables and directed edges between variables. Each variable has a finite set of mutually exclusive states (Bayesian Networks can admit continuous valued Gaussian variables as well, but we will not be concerned with them here). The variables together form a DAG (Directed Acyclic Graph). To each variable A with parents $B_1 \ldots B_n$ there is attached a conditional probability table $P(A|B_1, \ldots B_n)$.

Note that if A has no parents, then the table reduces to the unconditional prior probabilities $P(A)$. One of the main advantages of Bayesian Networks is that they admit d-separation; thus is A and B are d-separated in a Bayesian Network with evidence e

entered, then $P(A|B,e) = P(A|e)$. This allows us to use *d-separation* to read off conditional independencies. This is used without further proof.

If $U = (A_1 \dots A_n)$ is a universe of variables. Knowing the joint probability table $P(U) = P(A_1 \dots A_n)$ allows us to calculate $P(A_i)$ as well as $P(A_i|e)$, where e is the evidence. However, the table $P(U)$ grows exponentially with the number of variables. A Bayesian Network B_u can be viewed as a *compact* representation of the probability table, and if the conditional independencies in B_u hold for U, then the following theorem allows us to calculate $P(U)$ from the conditional probabilities in the network. For the proof, the reader is referred to (Jensen 1996).

Theorem 25 The Chain Rule*(Jensen 1996) Let B_u be a Bayesian Network over $U = (A_1 \dots A_n)$. Then the joint probability distribution $P(U)$ is the product of all conditional probabilities specified in B_u.*

$$P(U) = \prod_i P(A_i|pa(A_i)) \tag{7.1}$$

where $pa(A_i)$ is the parent set of A_i

Example 13 Exploiting *d-separation* in Bayesian Trees (Pearl 1988)

In cases where the Bayesian Network is a tree, exploiting the d-separation property results in a local message passing scheme for evidence propagation. This algorithm does not work for the more general multiply connected DAG. For these cases, our implementation uses the JLO algorithm (Jensen 1996; Lauritzen & Spiegelhalter 1988) explained in the next section.

Figure Figure 7.2 shows an internal node X of a Bayes tree network and its neighbors, adopting notation from (Pearl 1988). U is the parent node, Y, Z the child nodes (we use two children for concreteness, but everything is easily generalized to nodes with arbitrary number of children). V is a generic sibling node to X (either to the left or to the right). $\mathfrak{e}^+$ denotes all the conditioning evidence reaching X through its parent, whereas $\mathfrak{e}^-$ represents ell evidence affecting X through its children. The arrows indicate that the local dependencies are given through conditional distributions $P(X|U)$, $P(Y|X)$ and $P(Z|X)$. In the exposition to follow, uppercase letters denote random variables ($U, V, X, Y, \dots$), lowercase letters are used for instantiations (particular values) of variables ($u, v, x, y, \dots$).

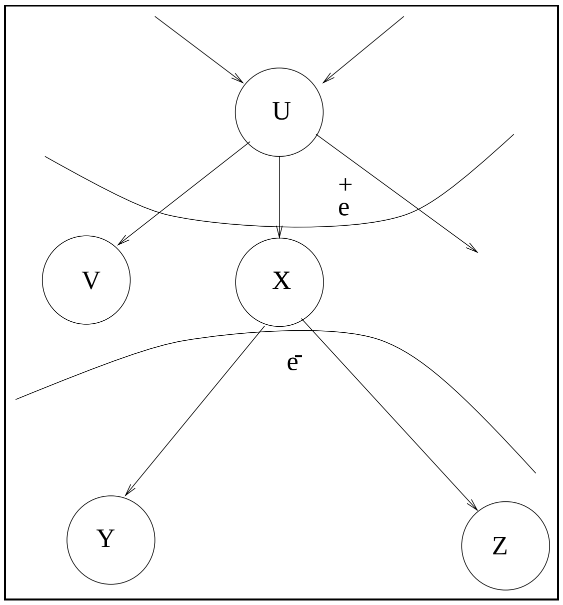


Figure 7.2: Internal nodes of a Causal Tree.

The central idea in belief propagation is to split the conditioning evidence in the two parts $\mathfrak{e}^+$ and $\mathfrak{e}^-$ and compute the influence of each separately. Using Bayes' rule we get

$$\begin{aligned} P(x|\mathfrak{e}) &= P(x|e^+, e^-) \\ &= \frac{P(x, e^-|e^+)}{P(e^-|e^+)} \\ &= \frac{P(x|e^+)P(e^-|x, e^+)}{P(e^-|e^+)} \\ &= \frac{P(x|e^+)P(e^-|x)}{P(e^-|e^+)} \end{aligned}$$

The last simplification is possible since e^+ can affect e^- only through X .

The final expression can be factored into a 'prior' term $\pi(x)$ and a 'likelihood' term $\lambda(x)$ (analogous to Bayes' rule) where

$$\pi(x) = P(x|\mathfrak{e}^+) \tag{7.2}$$

$$\lambda(x) = P(\mathfrak{e}^-|x). \tag{7.3}$$

π and λ are also known as *causal* and *diagnostic support*, respectively (Pearl 1988). The conditional belief then becomes

$$P(x|e) = \alpha\pi(x)\lambda(x), \tag{7.4}$$

where α is the normalizing constant $\frac{1}{P(e^-|e^+)} = \frac{P(e^+)}{P(e)}$ (which need not be computed explicitly). Note that α (and therefore $P(x|e)$) is only defined if $P(e) \neq 0$, as expected.

The λ 's and π 's can be computed by passing messages between parent and child nodes. This is because of the following recursive relationships:

$$\lambda(x) = \sum_y \lambda(y)P(y|x)\sum_z \lambda(z)P(z|x) \tag{7.5}$$

$$\pi(x) = \alpha\sum_u \pi(u)P(x|u)\sum_v \lambda(v)P(v|u) \tag{7.6}$$

λ 's on leaf nodes are initialized as follows: if a node X is instantiated (has known value) x , then $\lambda(x) = 1$ and 0 for all other values. If no value is given for X , $\lambda(x) = 1$ for all x . The root node is instantiated π 's representing the prior distribution on its variable. (Evidence on non-leaf nodes can be incorporated by creating an extra leaf child node on the non-leaf.) ■

7.2.1 The JLO algorithm

While the example above illustrated how local message massing could work for causal trees, the implementation actually uses a data-structure called a **junction tree** and an updating algorithm called the **JLO** Algorithm (Jensen 1996) which works for multiply connected networks where there may be more that one path between two variables. The basic structure and propagation scheme are briefly outlined here for completeness. The interested reader is referred to the citations above for more details. Implementation of the Multiply-connected DAG to Junction trees and the JLO algorithms were adopted from the public domain IDEAL package (Srinivas & Breese 1990).

Definition 26 **Junction Trees** (Jensen 1996) A *junction tree* J_{BN} over the Bayesian network BN is a tree whose nodes are clusters of variables from BN , which are *cliques* in BN . Each link between any two variables U and V in J_{BN} are labeled with a *separator* which is the is the intersection of the adjacent cliques. Each clique and separator holds a real-numbered table over the configurations of its variable set.

The **junction tree property** is said to hold if for each pair of variables U, V that represent nodes in J_{BN}, all paths between nodes U and V contain the intersection $V \cap U$.

A junction tree is said to **represent** the Bayesian network BN if

1. For each variable A, there is a clique containing $pa(A) \cup \{A\}$.

2. $P(U)$ is the product of all clique tables divided by all separator tables.

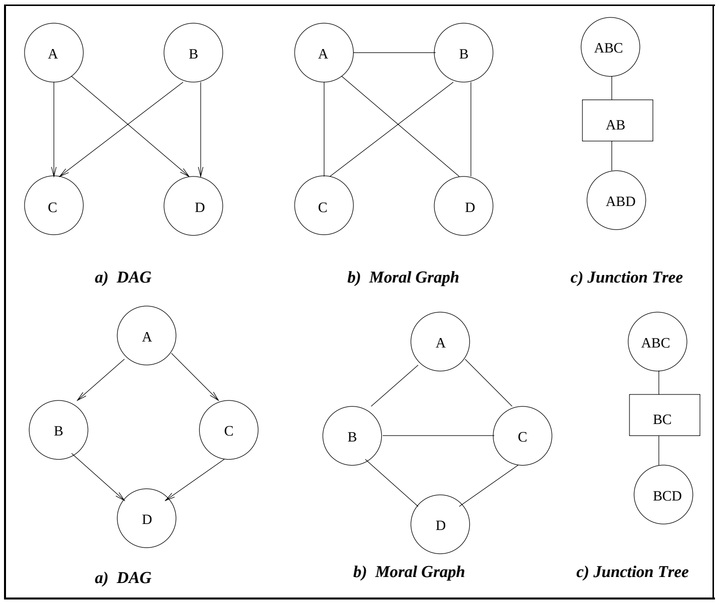


Figure 7.3: Construction of a junction tree from a Multiply Connected DAG

Algorithm 1 Constructing a junction-tree

Let BN be a Bayesian Network over the variables U.

1. Construct the Moral graph: the undirected graph with a link between all variables in $pa(A) \cup \{A\}$ for all A. This step results in transforming the DAG's in Figure 7.3 a) to the one in Figure 7.3 b).

2. Triangulate Moral graph: add links until all cycles consisting of more than three links have a chord. Both moralized graphs in Figure 7.3 are already triangulated.

3. The nodes of the junction tree are cliques in the graph.

4. Connect the cliques of the triangulated graph with links such that a junction tree is constructed. The junction tree for the two graphs are shown in Figure 7.3 c).

5. Initialize the cliques and separators with a table consisting of only **1's**.. Then for each variable, A find a clique containing $pa(A) \cup A$, and multiply $P(A|pa(A))$ on its table. The resulting junction tree represents BN.

■

Algorithm 2 Propagation in Junction Trees

The basic $BELIEF\_UPDATE$ algorithm shown below starts with a junction tree with the various node clusters and the separators. One clique (recall that a node in the junction tree is a cluster of nodes forming a clique in the original Belief network) is chosen as the root. Starting from this cluster the function $collect\_evidence$ asks all its neighbors to collect evidence as they proceed down the tree recursively. When the neighbors are done, the root cluster *absorbs* form them. The absorption operation is described below. After the absorption, messages are sent in the other direction, where neighbors are made to *absorb* from the root cluster/node and then do the same to their neighbors recursively down the tree. This is the function $distribute\_evidence$. The propagation takes time proportional to the size ($variables \times values$) of the tree (which can be exponential in the size of the largest clique in the original Belief network).

```
BELIEF_UPDATE(J_tree)
  begin:
    An arbitrary clique $R_t$ from J_Tree is chosen as root.
    collect_evidence($R_t$)
      collect_evidence from neighbors proceeding down the tree recursively.
      When all the called neighbors are done, $R_t$ absorbs from them.
    distribute_evidence($R_t$) .
      make all its neighbors absorb from $R_t$ , and afterwards recursively
      distribute_evidence to all its neighbors (except $R_t$ ).
  end
```

■

BELIEF_UPDATE thus performs a series of *absorptions* to update the probabilities of various clusters to compute the *global* impact of new evidence over the entire network. The absorption operation results in rearranging the information stored in the various nodes of the tree (the Conditional Probability Tables). The basic idea of absorption is that information in common between two adjacent nodes U and W is held in the separator between these nodes (S) and the operation. The *absorption* operation between U and W are the result of the following operations at the end of which W has absorbed from U .

1. Calculate $\mathbf{t}^*_\mathbf{S} = \sum_{U \setminus S} \mathbf{t_U}$
2. Send S the table $\mathbf{t}^*_\mathbf{S}$
3. Send W the table $\mathbf{t}^*_\mathbf{W} = \mathbf{t_W} \frac{\mathbf{t}^*_\mathbf{S}}{\mathbf{t_S}}$

7.2.2 Belief revision and MPE Estimation

In Belief revision, we are interested in the Most Probable Explanation (MPE) for the observed data. Here we are concerned with assigning a *value* to all the variables in the Belief net such that the overall joint posterior probability is maximized. Thus the MPE algorithm returns one *state* for each variable such that the overall probability of that state assignment (called a configuration) is maximal (better that any other configuration). For obvious reasons the MPE algorithm is also known as the Most Probable Configuration (MPC) algorithm.

Definition 27 Most Probable Explanation (MPE)

Let V be a set of variables. A *configuration* v^* of V is a set of states $\{a \ldots, b\}$ which contains exactly one state for each Variable of V.

Let W be a subset of V, and let w^* be a *configuration* of W. Then $V * w^*$ is the set of configurations of V containing w^*:

$V * w^* = \{v^* | v^*$ is a *configuration* of V, $w^* \subseteq v^*\}$.

Let V stand for the set of all variables considered, including those in the evidence $\mathbf{e}$. The MPE problem is then to find a *configuration* $\mathbf{w}^*$ that maximizes the conditional probability $P(\mathbf{w}|\mathbf{e})$. In other words, $\mathbf{W} = \mathbf{w}^*$ is the MPE (also called maximal configuration) of the evidence $\mathbf{e}$ if

$$P(w^*|e) = \max_w P(w|e) \tag{7.7}$$

With a slight modification, the JLO propagation algorithm described earlier can be used to compute the MPE for the evidence at hand through local message passing on the *junction tree*. The basic modification is that in the absorption phase, the $\sum$ is replaced by a **max** operation. The new equation is

$$t_s^* = \max_{W \backslash S} \mathbf{t_W}$$
$$t_v^* = t_v \frac{t_s^*}{t_s}$$

The propagation method to compute the maximum probability configuration, is very similar to the absorption operation, except the $\sum$ is replaced by a **max** operation. The two functions *max_collect* and *max_distribute* are exactly similar to the functions *collect_evidence* and *distribute_evidence* in the $BELIEF\_UPDATE$ algorithm, except that they call Equation 7.8 above instead of the absorption equations described earlier.

7.2.3 Stochastic Simulation

The other algorithm used in our simulations is the stochastic simulation algorithm with Likelihood Weighting(Fung 1990). In stochastic simulation, we run repeated simulations of the world described by the belief networks. The basic algorithm begins by choosing a values for the root variables (the prior slice) based on the prior values. We continue to chose values to child variables based on the conditional probability distribution, till we hit an evidence node. We count the number of times both the variable of interest and the evidence node were of the instantiated value and divide this by the number of times that the evidence node was incorrectly sampled. In the limit, this number approaches the true probability of the node of interest given the evidence.

The MPE and Stochastic Simulation algorithm was used to update the temporal belief net corresponding to the reader's knowledge of international economics. The *MPE* and *BELIEF_UPDATE* algorithms used in this thesis were adapted from the public domain Belief Network package IDEAL (Srinivas & Breese 1990).

With this background, we are ready to see how the knowledge about international economic discourse is encoded in a Belief network framework. But before describing the encoding, we briefly we need to know in a bit more detail what we are encoding, namely the domain of international economics.

7.3 Target Domain Ontology

Let us begin by examining the various terms that comprise the target domain of international trade policy. The first cut distinction is between *Open* and *Closed* economies. These terms pertain to the importance of trade in an economy. As (Bradford 1991; Bhagwati 1987) and others point out, these concepts are graded and that one could place policies as being partially open or closed.

The table below is based on data from (Bradford 1991) and describes the various ontological primitives in the discourse of trade policy.

State	**Description**
Autarky	No trade, Isolationism, self-reliant
Import Substitution	Discriminate against imports
Liberalization	trade based, internal liberalization
Export push	Subsidize exports

In a *closed* economy, trade plays a relatively small part in the overall GDP. A closed economy may result from deliberate policies or from size, or from abundance of natural resources.

In an *open* economy, trade plays an important role in contributing to the overall GDP. An open economy can be achieved through the process of *internal liberalization.* Internal liberalization is the removal of any tariffs, import controls, or other trade restrictions. Open economies are also commonly referred to as *Trade* economies for the reason cited above. Open economies may result from external demand driving up internal production or may be may be a result of increased supply.

The terms closed and open economies describe a state of affairs. Various *development strategies* or policies may influence these states. In general, economic policies may be *inward oriented* or *outward oriented*(Bradford 1991). Inward oriented policies favor the domestic market whereas outward oriented policies favor trade and exports.

The extreme version of inward orientation postulates *delinking* or *isolationism* leading to a completely closed economy. The extreme version of outward orientation results in subsidizing export industries leading to an *export push.* In an *export push* economy, the state may interfere to set *domestic prices* and *exchange rates* to provide more incentives for exports rather than imports or domestic consumption, in effect subsidizing exports.

Import substitution refers to the situation when a state interferes to discriminate against imports to favor domestic production. Import substitution policies may be implemented using multiple exchange rates, tariffs, and other control measures. The bias against imports may vary from complete *protectionist* to more *selective* policies.

One intermediate form of outward orientation is internal liberalization that leads to an open economy. In an open economy the "incentives to import as as great as the incentives to export" (Bradford 1991). In such an economy the import and export exchange rates are likely to be the same.

At any given time, a country would fall into one of the aforementioned clusters. Adopting a specific policy, a country may transition from one cluster to another.

Current	**Next**	**Transition**
Isolationist	Import Substitution	Ease export controls
Import Substitution	Open	Ease import controls
Open	Export Push	Intervene to increase export share of GDP
Open	Import Substitution	Impose restrictions on selected items
Import Substitution	Closed	Impose restrictions on all items
Export push	Open	Stop intervention

Controls and restrictions refer to exchange rates, tariffs, taxes, duties and other special licensing fees. Export controls impact the type and amount of goods exported whereas import controls impact the type and amount of goods imported.

A country's economic strategy may be to start by selectively restricting imports through import substitution and then when the domestic industry has reached a certain maturity, remove restrictions through internal liberalization and become an open economy. Clearly, the entire transition table can be generated by concatenating actions to go between non-contiguous states and by reversing actions to move in the opposite direction. Examples may involve removing import and export restrictions *in order* to go from an isolationist to an open economy or imposing export restrictions to move from an import substitution to an isolationist economy. However, to capture order dependent effects additional intermediate states may be required.

7.4 Encoding Economic Policy Knowledge in A Bayesian Network

A Bayesian network is a convenient data structure to encode the domain knowledge about international economic discourse described in the last section. The basic algorithms that operate on the network data structure deal both with belief revisions in the face of new observations (from x-schema executions) and database updates due to external intervention such as linguistic input.

The target domain representation models the non-professional reader's knowledge of the domain of Economic Policies. Since the discourse fragments we are interested may refer to multiple stages of events by specifying different *phasal aspects*, we need to be able to capture some of the temporal structure of events. Furthermore, we would like our Belief network model to encode the future implications of specific assertions; for instance if an utterance asserts a policy change (say from a protectionist to a liberal economy), then the corresponding implications (goals change from restricting trade to free-trade, etc.) should be available for the next utterance. Furthermore, as we will see in Chapter 9, we may often have to propagate the influence of specific assertions back as well as forward in time to construct an overall explanation of the discourse fragment in question. For all these reasons, we chose to use a temporally extended Belief network (Dean & Wellman 1991) with one network copy representing the result of processing each input utterance (event description). Figure 7.4 shows the implemented temporal belief network.

The structure of the target domain for three temporal slices of the Belief network is shown in the top part of Figure 7.4. We assume that the Belief network nodes are discrete multi-valued variables. There is one copy of the target domain net for each relevant time step (each input utterance). The dashed link that connects the $t-1$ copy to the t th copy of the target domain belief net tends to keep the value of the variable unchanged in the absence of other intervening factors and is called the **persistence** link.

Of course, in any real discourse processing, earlier slices would be reclaimed and only a small set of copies (a window size) would be kept. The actual number would be determined by working memory capacity limitations. However, since our discourse fragments encompass at most 3 event descriptions (implying at most 5 temporal slices, one representing the prior (background) knowledge, and one for projecting into the future from the discourse fragment).

The Bayesian network that encodes the target domain has variables for the different policies and transitions discussed in the last section. For instance the policy variable has the five options discussed earlier (Ec. Policy = (autarky, isolation, import substitution, liberalization, export push)) and the change variable *Do_Change* has the 6 transitions discussed earlier. Besides there are variables that represent the Actor/Agent of the policy in question (values include US Gov., Indian Gov. Business, Economists, and International Banks (IMF and WB)). Besides there are variables that correspond to the *outcome* (success or failure) of a given policy, whether there is any *difficulty* in plan implementation and whether the policy leads to *free-trade* and *deregulation* or trade restrictions.

In Figure 7.4, the shaded nodes for the target domain belief network at time $T = 0$, represent prior values for the relevant economic policy variables. For instance, the prior expectation is that the actor in question is the US Government ($USGov.$ in Figure 7.4)

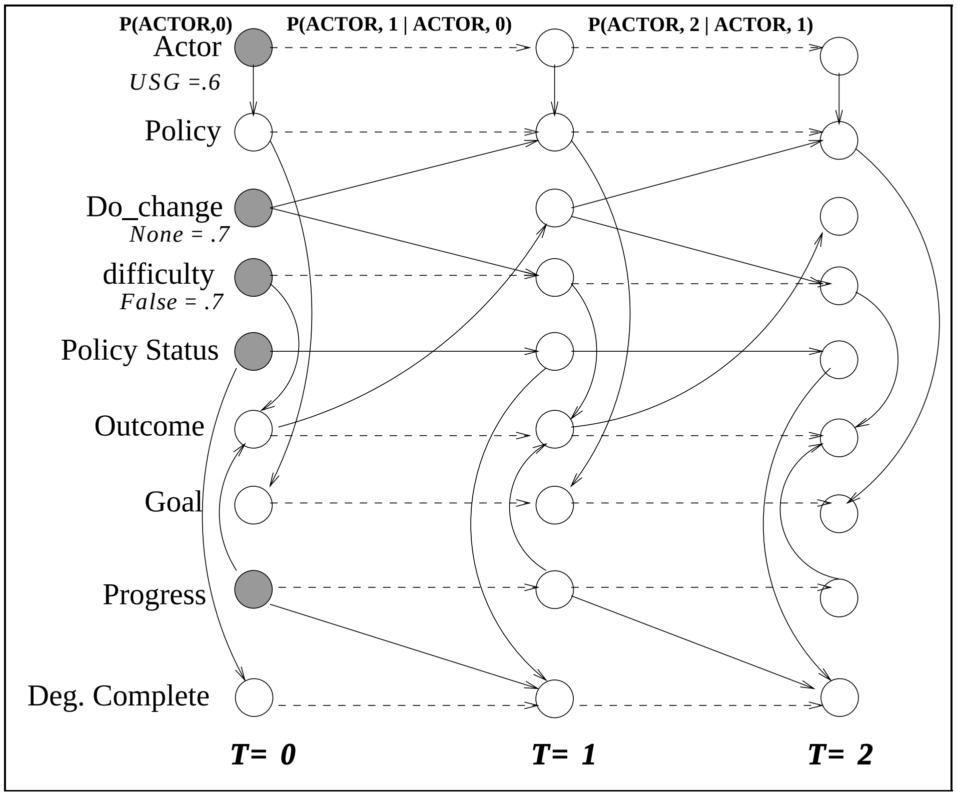


Figure 7.4: A temporally extended Bayesian network that represents knowledge about international policy discourse outlined in the last section. As mentioned earlier, there is one network slice for each input utterance. Since the network has multiple temporal slices, it is a temporally extended network. The figure shows a network with three temporal slices. The slice labeled $T = 0$ is the network *prior*. Links shown in dotted lines represent persistence links, i.e. the chance the variable of interest (say the $ACTOR$ variable) takes on a specific value (say US_Gov) at time t (say $t = 1$) given that it takes on a specific value at time $t - 1$ ($t = 0$). Nodes in the network represent discrete and *multi-valued* variables. For instance, the variable entitled $Policy$ in the figure takes on one of the 4 values [autarky, protectionist, Liberlization, and Export push] described in the text. Links between variables in a single temporal slice represents the causal dependence of the state of one variable on another. For instance the variable $Goal$ is dependent on the type of policy followed (for example free-trade and deregulation for liberalization policy).

and that the the no change in policy in asserted ($Do\_Change = None$) in Figure 7.4. This may be overridden by an input f–struct asserting "Actor = Indian Government", in which case the Actor node is clamped to the value IG .

The persistence links encode the conditional probability of a variable's value at time t , given it's value at $t-1$. One interesting point is that the *aspect* graph transitions are implicitly coded in the target domain as the conditional probability of being in a specific control status at time t given the control status at time $t-1$. For instance let us look at one variable, namely the probability that the status of a schema is *ongoing* at time t , given the status at time $t-1$.

$$\begin{aligned} P(ongoing(t)|started(t-1) &= .8) \\ P(ongoing(t)|enabled(t-1) &= .2) \\ P(ongoing(t)|suspended(t-1) &= .1) \\ P(ongoing(t)|ongoing(t-1) &= .6) \end{aligned}$$

All other values, such as $P(ongoing(t)|done(t-1))$ are negligibly small (≈ 0). These values are default values and are often overridden by specific assertions as we will soon see in detail with the case of "stumbling". Also, the inferential machinery described in the last chapter can go backward and forward in time to answer questions such as "If the negotiations were *ongoing* at utterance time t , what is the probability they were *ongoing* at time $t-1$, or that they are *done* at utterance time $t+1$?"

Figure 7.5 shows the impact of asserting evidence of the Belief network. The first thing to notice is that evidence can be asserted by clamping any node (shown in black) at any of the time steps. This will be crucial in Chapter 8 since a single embodied lexical item (say *stumble*) is able to assert that the previous state corresponded to an *ongoing* WALK , while the current state is an *interrupted* WALK . As we will seen in Chapter 8, this gets projected onto two slices of the target network, one that asserts an *ongoing* status (Figure 7.5) at the previous copy, and an assertion of an *interrupted* status at the current copy. Figure 7.5 shows this situation.

In order to completely specify the network shown in Figure 7.4, we need to specify

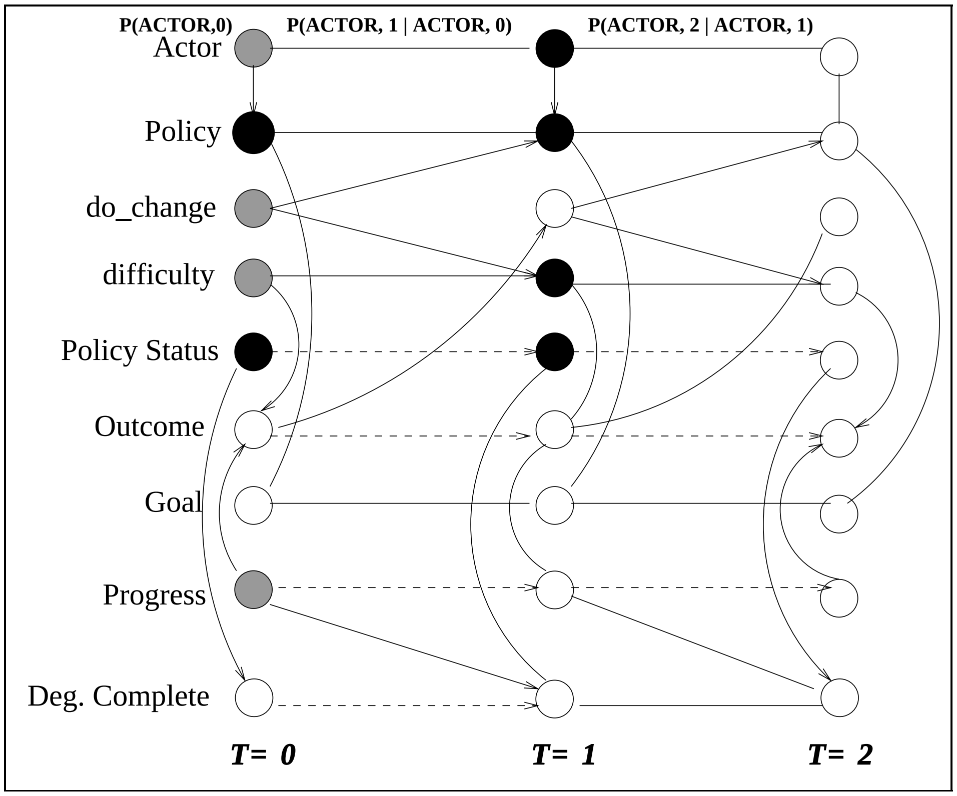


Figure 7.5: Asserting evidence on the target domain network and subsequent updates. First point to note is that a single network intervention (from linguistic input in our case) can set evidence (shown as dark nodes in the network in more than one time slice). In the example shown, evidence has been set for the presence of a difficulty at time $T = 1$, the Actor has been instantiated to IG (Indian Government), and the Policy Status is set as *ongoing*.

the *prior* probabilities at time 0 for root nodes in the Bayesian network, and the various conditional probabilities of both the persistence and inter-state correlation links. This is shown in Chapter 11, Section 11.3. Once we specify the various network parameters, the $BELIEF\_UPDATE$ and MPE algorithms can compute the impact of any new intervention in the network, such as new evidence on all variables in both in the current state as well as previous and future states. The use of this method to retrieve explanation for narratives is discussed in the next chapter.

Chapter 8

A Computational Model of Metaphoric Reasoning

The central hypothesis pursued here is that the meaning of motion and manipulation words is grounded in fine-grained, dynamic representations of the action described. Systematic metaphors project features of these representations onto abstract domains such as economics enabling linguistic devices to use embodied causal terms to describe features of abstract actions and processes. In this chapter and in Chapter 9, we attempt to provide evidence for the following proposition.

Proposition 2 . The role of Embodied Metaphors
Embodied metaphors project features of spatial motion and manipulation onto abstract plans and processes. This allows event descriptions to exploit the highly compiled and real-time aspects of x-schema representations to express **complex**, **uncertain**, *and* **evaluative** *knowledge about abstract domains such as international economic policies.*

In our theory, embodied metaphors such as the Event Structure Metaphor (Lakoff 1994) use the dense and familiar causal structure of spatial motions to permit reflex (fast, unconscious) inferences. In Chapter 6, we saw how our x-schema based representation was able to encode theories of embodied and highly familiar domains in manner that retains the *dynamic* and *highly responsive* real-time nature of x-schemas and their ability to model *reflex* inferences. In this Chapter and in Chapter 9 we will detail how projection of x-schema based inferences can add valuable information about abstract plans and events in real-time.

Recall that we are concerned about modeling information provided by motion and manipulation terms used in ordinary discourse (such as newspaper stories) about the domain of international economics. To this end, Chapter 7 laid down our ontology and simple model of the target domain of international economic policies. Our computational model consisted of multi-valued economic policy variables modeled as a Bayesian network. We argued that the inherently temporal nature of events necessitated a temporally extended network with temporal links that predicted the value of a variable at time t, given its value at time $t-1$.

This chapter describes our overall computational model which includes a third important piece, namely **metaphor maps** that project specific features obtained from x-schema executions onto features of the domain of international economics. When active, metaphor maps allow the real-time inferential products of x-schema executions to influence the target domain network by **setting** evidence for specific features of a given temporal slice of the economic policy network to specific values. As we will see in this chapter, activating a metaphor map requires that the context (domain of discourse) be identified as being about the target domain; thus shutting off projections for irrelevant contexts.

In our model, embodied metaphor maps are **hybrid** structures connecting the x-schema based sensory motor representations to the target domain belief network that represents knowledge about international economics. Such maps project specific results of x-schema executions; adding new information about the target by asserting *evidence* on the temporally extended Bayes net. The $BELIEF\_UPDATE$ algorithms described in Chapter 7 are then able to propagate the influence of this *new evidence* forward and backward in the temporally extended network to arrive at new *posterior* values for other relevant economic features at the same and different time steps.

Our model currently includes three different types of embodied maps. One type of map corresponds to **ontological** maps (Lakoff & Johnson 1980) which map entities and objects between embodied and abstract domains. Such maps are called $OMAPS$. As an example of a commonly used $OMAP$ consider the mapping $Mover \Rightarrow Actor$ which maps the the object filling the role of *mover* onto the abstract domain as the entity filling the role of *actor*. In general, one central function of $OMAPS$ is to map the fillers of various case-roles of an *event phrase* across domains.

A second type of map projects *events*, *actions*, and *processes* from embodied to abstract domains. In keeping with our representation, we will call such maps Schema maps or $SMAPS$. An important function of $SMAP$ projection is to **invariantly** map the *aspect* of the embodied domain event onto the target domain. This is done by mapping the CONTROLLER status of the motion event to create evidence for the $status$ variable of the target domain network at the appropriate time slice. A common example of an $SMAP$ is the well known event structure (Lakoff 1994) mapping $Motion \Rightarrow Action$. This map projects self-propelled spatial motion onto the domain of plans as action/plan/policy implementation. Crucially, the aspectual phase of motion ($ready$, $ongoing$, $done$, $suspend$, etc.) is mapped as the status of the abstract action described. Recall from Chapter 7, how the use of a temporally extended network allowed us to store aspectual state information about the abstract event through the $status$ variable and the possible transitions of the CONTROLLER as conditional probabilities of the temporal links (links between time slices) connecting the $status$ variable. $SMAPS$ use this feature to project **events** and their **aspect** from familiar to abstract domains.

As we described in Chapter 3, a central component of x-schemas is that they are *parameterized* routines with run-time parameter and dynamic variable binding. We found specific x-schema parameters to be routinely projected to abstract domains. In our model, these projections are accomplished by x-schema parameter maps ($PMAPS$). Common examples of such projections include the map $Rate\_of\_motion \Rightarrow Rate\_of\_progress$, $Distance\_traveled \Rightarrow Degree\_of\_Completion$, and $Size\_of\_step \Rightarrow Amount\_of\_Progress$. The first $PMAP$ above projects rates of motion onto the abstract domain as the rate of progress made, the second one projects distance traveled onto the abstract domain as degree of completion of a plan, and the third projects size of a step to be the amount of progress made. All these maps are frequently used in discourse about economics (for ex. *crawling* is projected as a low rate of progress, *giant steps* is projected as a *good* amount of progress in plan implementation, *setting out* refers to the *start* of a plan, etc).

8.1 Computational Architecture Of the Metaphor Reasoning System

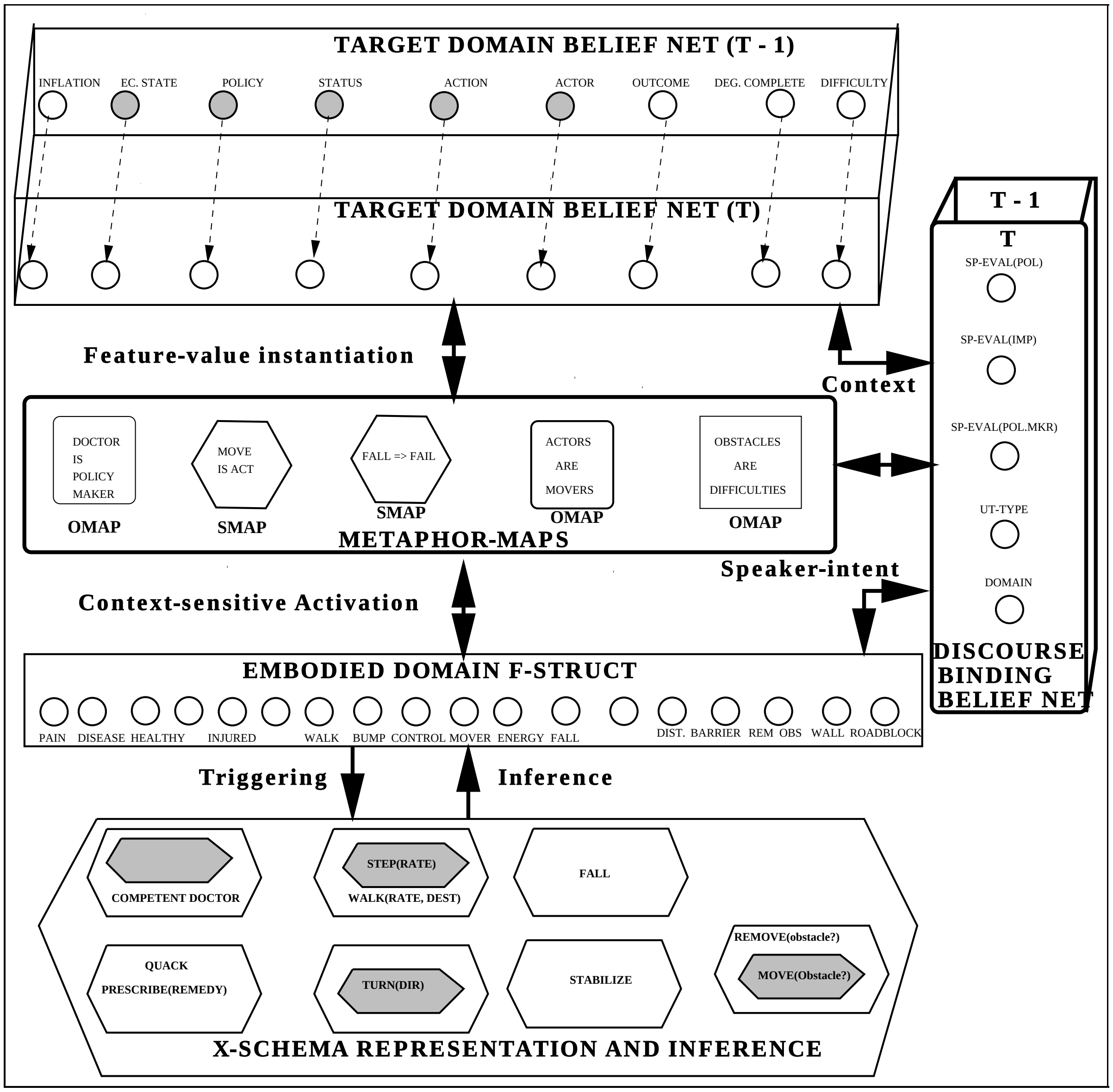


Figure 8.1: Metaphors capture systematic correlations between features of different domains. Shown in the figure is a small fragment of the x-schema-based source domains, a fragment of Bayesian network modeling the target domain and different maps projecting from source features to target features by setting evidence for specific nodes. Note the special variables for the discourse binding. In the figure, the bottom half corresponds to the x-schema inference model described in Chapter 6 . The top part of the figure refers to the temporally extended network representing the system's knowledge of the domain of international economic policies. (ref. Chapter 7). The middle part of the figure is new (shown in boldface) and contains the various types of metaphoric maps that project familiar source domain events and actions onto abstract events and plans.To the right of the figure is another new temporally extended belief net entitled Discourse Binding Belief Net. Nodes in this net code for various communicative intent and evaluatory information often derivable from the choice of embodied terms.

Figure 8.1 shows the basic computational architecture of the implemented system. As shown in the figure the system has four main components.

1. The top part of Figure 8.1 shows the temporally extended Belief network representation of the reader's knowledge of international economic policies. The different temporal slices have default **persistence** links between them which can be overridden by new evidence. The Belief network at time t_0 corresponds to the prior or background knowledge of the reader.

2. The bottom part of Figure 8.1 shows a set of x-schemas and the corresponding embodied domain f–struct, whose feature values both **trigger** x-schema execution and are set or modified as a result **inference** from x-schema execution.

3. The middle part of Figure 8.1 (shown in bold) shows the different kinds of **hybrid** metaphor maps which when **activated** based on the discourse context map embodied feature-values as evidence for specific feature-values in the economic policy Belief net. Such static maps may map objects and roles (called $OMAPS$ in Figure 8.1), activation and execution status of x-schemas (labeled $SMAPS$), or parameter-values (labeled $PMAPS$) from one domain onto another. The maps are context-sensitive in that they only project values if the domain of discourse is known to be about the target (economic policies in this case).[1]

4. On the right of Figure 8.1 is shown a set of multi-valued features labeled DISCOURSE BINDINGS. These nodes are also modeled as probabilistic nodes in a temporally extended Belief net. One variable pertains to the domain of discourse, while the others exploit an interesting feature of embodied terms; the felicity with which they encode speaker evaluation and intent which are integral to discourse comprehension, and absent from any previous model on metaphor that we are aware of.

We have seen in detail how the embodied domain inferences work through x-schema execution and simulation. We have also seen how target domain knowledge is represented as a Belief net and the various algorithms for belief update and revision that we use in the work described here. In this Chapter, we describe how the various pieces fit together through

[1]In the discussion to follow we will use the convention of depicting $OMAPS$ (corresponding to role-filler bindings) as rounded rectangles, $PMAPS$ (mapping x-schema parameter values) as rectangles and $SMAPS$ (mapping the aspectual component of events across domains) as hexagons.

projection structures (metaphor maps) that are able to mediate between the abstract and embodied domains. We will then be in a position to describe in detail the use of the overall model is used for on-line interpretation.

The target domain representation models the non-economist reader's knowledge of the domain of Economic Policies. The structure of the target domain for two temporal slices of the Belief network is shown in the top part of Figure 8.1. There is one copy of the target domain net for each input utterance. We will assume that the narrative has the same temporal order as the event occurrence order. The representation of tense and narrative time is not addressed in this thesis. The link entitled **persistence** that connects the $t-1$ copy to the t th copy of the target domain Belief net by default leaves the values of the policy variables unchanged unless a) new evidence is set to change a value as a result of metaphoric inference at the current time step, or b) a policy change was asserted directly or through metaphoric inference at the last time step.

In Figure 8.1, the shaded nodes for the target domain belief network at time $t-1$, represent prior values for the relevant economic policy variables. For instance, the prior expectation (before any linguistic input) that the actor in question is the US Government (USG in figure) and that the economy is experiencing moderate growth ($EC.STATE = mod.growth$) is encoded by setting these nodes to these value distributions in the slice corresponding to $t = 0$. As we will see, once linguistic input comes in some or all of these values may be modified, resulting in a different *posterior* distribution for each of the relevant variables.

The persistence links encode the conditional probability of a variable's value at time t, given it's value at $t-1$. These values are default values and are often overridden by specific assertions as we will soon see in detail with the case of "stumbling". Also, the inferential machinery described in the last chapter can go backward and forward in time to answer questions such as "If the negotiations were *ongoing* at utterance time t, what is the probability they were *ongoing* at time $t-1$, or that they are *done* at utterance time $t+1$?"

As an example of linguistic input overriding the prior value of a variable, consider the case where an input asserts "Actor = Indian Government" at time $t = 0$. As a result of processing the input, the model is able to *clamp* the Actor variable to the value IG for the slice of the network $t = 1$. In the belief update phase (using the algorithms described in

Chapter 7), the influence of this evidence at time $t = 1$, propagates backward to the slice at time $t = 0$, and results in *setting* the Actor variable to the value IG, overriding the network prior value of USG explained above. While linguistic input often results in changes to priors (after all that is what new information routinely does), it is important to note that our system is able to model cases where **prior** information can modify the interpretation of a specific utterance. Chapter 9 has interesting examples of this phenomenon, where prior information plays a role in which x-schema runs, and thus which inferences are generated and projected as updates on the target domain network.

Comprehending a story corresponds to finding the set of trajectories (high probability feature-values for missing features *implied* by the input) that satisfy the constraints of the input and are consistent with the domain knowledge (encoded both as inter-variable links and temporal links in the target net). This may involve *filling in* missing values for the target domain features, as well as inferring values for unmentioned target features *implied* by the story. The most probable trajectory can then be retrieved as the most likely explanation of the story. Features with highly selective posterior distributions are likely to be present in the recall of the story. A complete trajectory infers values for features comprising the reader's state taking as evidence nodes at specific times that are clamped due to x-schema inference or linguistic input (or both), and re-estimating values for these features at other times. We use the belief update and revision algorithms described in Chapter 7 for this purpose. In Figure 8.1, we see that the two domains of relevance, namely the target domain of Economic Policies and the embodied domains of health and spatial motions have their own domain theories. For instance, in the domain of spatial motion, we may expect that the presence of a specific type of obstacle while executing a step may cause the mover to stumble. Thus the concept of *stumbling* is encoded by activating the the x-schema for **walk** with $status = ongoing$ at time $t - 1$, and asserting a $bump$ at time t.

In all further discussion, we assume the presence of a systematic correspondence across conceptual domains, that is characterized by a fixed system of conceptual metaphors (Lakoff & Johnson 1980; Lakoff 1994) which govern how abstract concepts and domains are understood in terms of the more concrete and "experiential" domains. In our model, such metaphors are first class objects that are nodes capturing inter-domain constraints and correlations. As the figure shows, the different types of embodied metaphor maps are **hybrid** structures linking concrete domain features and values and target domain features and values. One one side of the map lies the x-schema based perceptuo-motor representations, which interact with the embodied f–structure, performing rapid, reflex reasoning by getting, setting and modifying the embodied f–struct values while executing a mental simulation of the event in question. On the other side is the Target Domain Belief Net, which captures the naive reader's knowledge of the domain of international economic policies. The various maps interact with this structure setting evidence based on inferential products from x-schemas, as well as direct input from the parser.

Recall that our model allows three different types of maps. An example of a *schema map* (or $SMAP$) is the Event Structure projection $Moving \Rightarrow Acting$ (shown as the hexagonal node labeled Moving IS Acting in Figure 8.1). A novel feature of $SMAPS$ are that they are able to naturally map aspectual inferences from embodied to abstract domains. Other types of maps modeled include ontological maps or $OMPAS$ (object-to-object or role to role maps) such as the map $Actors \Rightarrow Movers$, as well as a new type of map that projects specific x-schema parameters between the concrete and abstract domains (such as $distances$ traveled in the spatial motion domain onto *degree of completion* of an economic policy). Such maps are called parameter maps or $PMAP$s in our model. We note that earlier models of metaphor have either focussed on event-to-event mappings(including sub-categorization frames) (Martin 1990; Carbonell 1982) or on object-to-object maps (Sun 1994). Our model allows both types of maps to be modeled. In additional, the capability of projecting aspectual inferences through $SMAPS$ and projecting sensory-motor parameters to abstract events through $PMAPS$ are unique to our model. Each of the maps described above has its own execution logic which is detailed in Section 8.2.

The box on the right in Figure 8.1 labeled DISCOURSE BINDING BELIEF NET contains the variable that controls projections from concrete to abstract domains based on the domain of discourse. This is represented by the Belief net variable labeled $domain$. In general, domains are arranged in a type hierarchy, and identification of the domain is part of the parsing process. We will mostly assume prior knowledge that the domain of discourse is one of four possible values (namely international trade policy, economic state description, or the concrete domains of health and spatial motion). While we allow the parser to be ambiguous about which value is actually instantiated, we have not implemented any type inference over domain hierarchies, and for our implementation we assume a single multi-valued variable $domain$ to represent the result of the domain identification process.

Interestingly, as the program was being developed and trained on the database of stories, it became quite apparent that one central reason to use embodied terms is to compactly code communicative intent and evaluatory information (opinions, propaganda, judgments). The two variables entitled $Ut - Type$ (with four possible values $description$, $opinion$, $propaganda$, and $assertion$) and Speaker Evaluation of the specific *policy*, or *actors* in a specified event description (with values $neg{+}{+}$, neg, $neutral$, pos, $poss{+}{+}$) are meant to capture some of this aspect of embodied term usage. There are links labeled *Speaker intent* which directly set evidence for the $Ut - type$ and $Speaker - Eval$ variable based on specific values of the embodied domain f–struct. Thus x-schema executions are able to directly influence and set evidence for various values of this variable. Also, there is a Belief net link from $Speaker - Eval$ to $Ut - type$ captures the dependency of the utterance type on the extent to which the speakers evaluation of the situation is present in the utterance. For instance, the fact that the speaker evaluates a situation as highly negative (positive) influences that utterance type variable in that the utterance is more likely to be an opinion or propaganda than just a description. Obviously, the coding of intent in metaphoric usage is a much deeper issue, and results both from the training set and the test set indicate that further investigations in this area are likely to be highly productive. Chapter 9 outlines some interesting results that can be obtained even from this limited implementation. Chapter 10 outlines possible applications of our approach for speech act recognition.

Figure 8.2 shows the portion of the implemented network in action while interpreting the input utterance *Indian Government stumbling in implementing Liberalization Policy.* The bold entries correspond to direct intervention from the input as well as from x-schema inferences. Note that the $OMAP$ Actors ARE Movers is active with value IG, resulting in a token of **color** $< IG, t >$ in the mover place. Note also, that we assume that the domain variable has been instantiated to the value Economic Policy, thus setting the context for activating various maps.

We saw in detail in Chapter 6 how the lexical item *stumble* codes for the presence of a **bump** that *interrupts* an *ongoing* WALK x-schema. We saw how the x-schema based simulative inferences then result in bindings that specify an *ongoing* STEP being *interrupted*, leading to the agent *tripping* over the bump. Subsequently, the agent becomes unstable, and may be able to STABILIZE , depending on the available *energy* and other resources, or

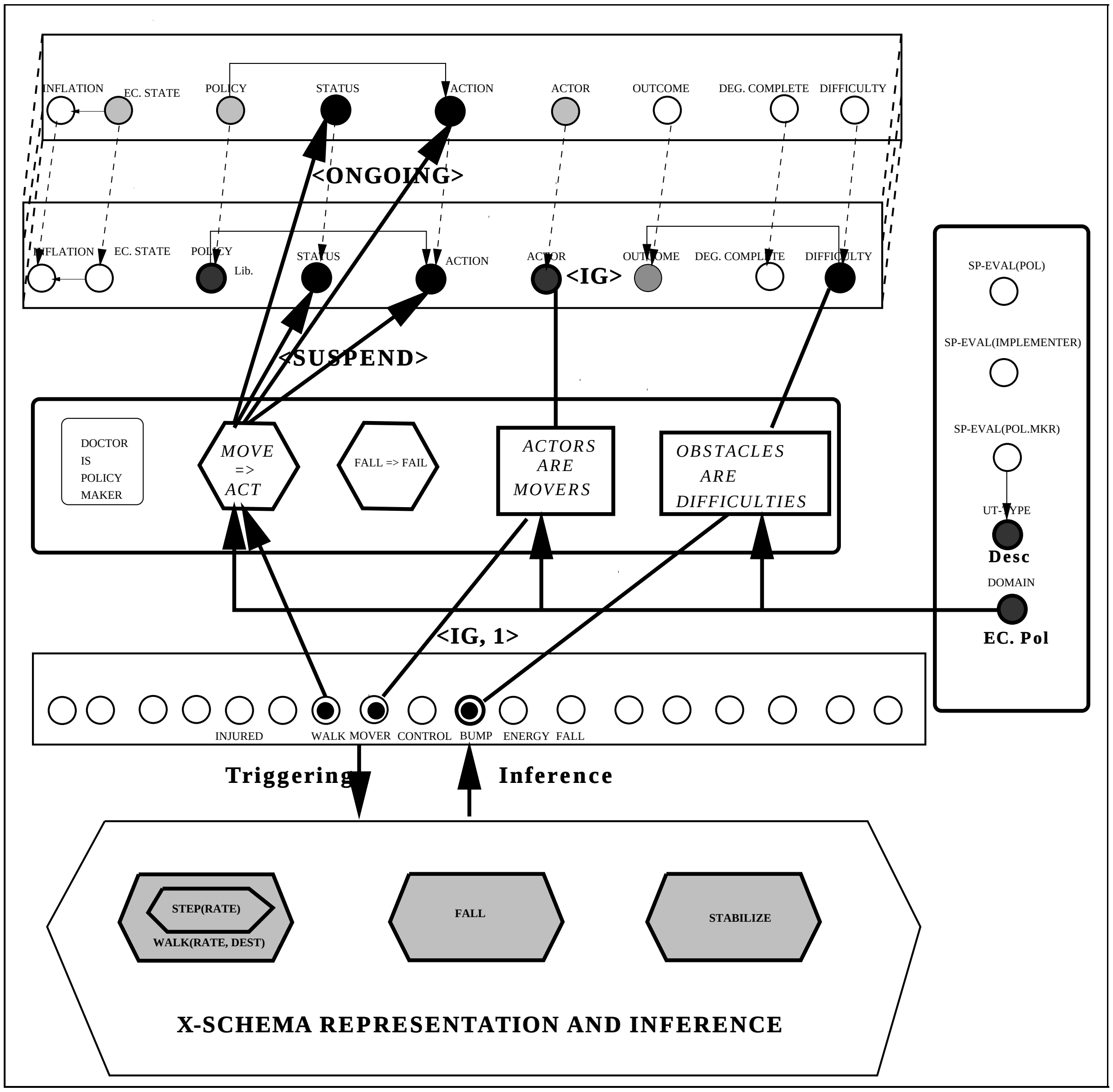


Figure 8.2: The system is shown interpreting the sentence *Indian Government stumbling in implementing Liberalization Policy* The entries in bold on the Belief net are evidence that has been directly asserted as part of the input (for instance the domain node is instantiated to the value "Economic Policy") as a result of the parsed input. Active Metaphors and inferential projection result in some other nodes being instantiated, for example, the suspension of walking due to the presence of a bump sets evidence for status of the policy to be one of suspension, and the presence of a bump activates the $PMAP$, $Obstacles \Rightarrow Difficulty$ thereby asserting the presence of a difficulty in the planning process.

may *lose_control* and enable the FALL x-schema.

An interesting thing to note about the lexical item *stumble* is that it codes for and *ongoing* WALK at a previous time step, which is *interrupted* as a result of encountering a bump. In other words, stumbling can occur only in the context of an ongoing walk. *Crucially*, this **framing** knowledge about stumbling is *projected* onto the abstract domain as well in that an ongoing plan is temporarily interrupted.

In our model, this kind of dynamic information is naturally modeled. As shown in Figure 8.2, the lexical entry for *stumble* binds to the walk schema at the previous time step, indicated by the marking of the *ongoing* node of the **walk** x-schema at time $t-1$. The result of asserting the *bump* at t, then activates the interrupt transition as described in Chapter 6 leading to the inference that the walk is suspended at time t. In the figure this is indicated by the two status tokens for the **walk** x-schema $< ONGOING, t-1 >$ and $< SUSPEND, t >$. As shown in Figure 8.2, the $SMAP$ functions then create evidence for two slices of the target domain network, one that corresponds to asserting an *ongoing* policy status at the previous time slice $t-1$, followed by a *suspend* status at the current time slice t. We know of no other implemented model of metaphor that is capable of making these temporal inferences. It also points out the need for a **dynamic** semantics in modeling even simple descriptions of events.

Figure 8.2 also shows the metaphors $Obstacle \Rightarrow difficulty$ and $Fall \Rightarrow Fail$. For example, the token at the **bump** node activates the $PMAP$ $Obstacle \Rightarrow Difficulty$ which in turn will project onto the target as the concept *difficulty.* Setting evidence for the presence of a difficulty at time t automatically allows the Belief update algorithm to infer greater posterior chance of failure due to the conditional dependence of the *outcome* variable on the presence of a *difficulty.*[2]

We also note that the utterance type is instantiated to the value *description*, which propagates back to the speaker evaluation node to set the value to *neutral*, except the presence of a token at the f–struct asserting instability of the Indian Government's efforts changes the value to be *neg* (negative) with a reasonably high probability. More such examples can be found in Chapter 9. A detailed description of the implemented system processing this example can be found in the next section. But before details of the overall

[2]Alternatively, in many cases the simulation carried forward an additional time step could execute the FALL x-schema which gets projected as failure.

program, the execution logic of the various types of metaphor-maps is described.

8.2 Representing Metaphor Maps

In our theory, Metaphor Maps are **active**, **hybrid** structures that mediate between highly compiled sensory-motor representations and more abstract feature structures.

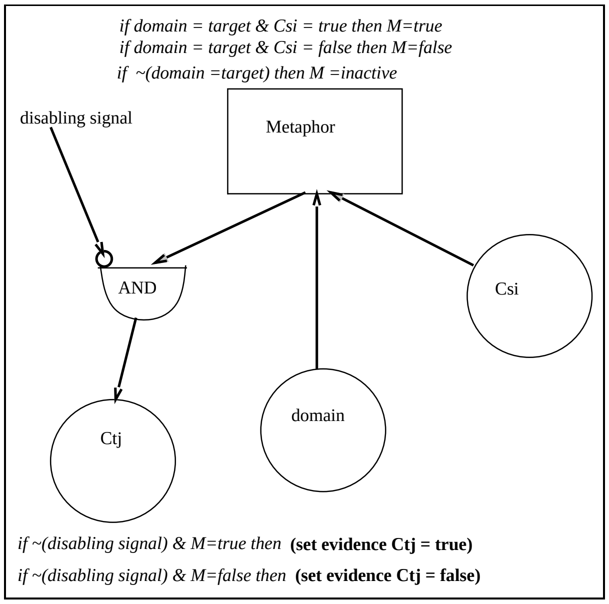

if domain = target & Csi = true then M=true
if domain = target & Csi = false then M=false
if ~(domain =target) then M =inactive
disabling signal
Metaphor
AND
Csi
domain
Ctj
if ~(disabling signal) & M=true then (set evidence Ctj = true)
if ~(disabling signal) & M=false then (set evidence Ctj = false)

Figure 8.3: Representing a metaphor. When active, the metaphor can potentially project the value of a source concept to the corresponding value of the target domain concept. Transfer can be blocked if the target domain knowledge is incompatible with the projected value.

Figure 8.3 shows the general structure of a metaphor node. It constrains the projectability of source domain concept C_{si} onto a target domain concept C_{tj} . In this case shown in Figure 8.3 both source and target domain concepts are binary but they can be multi-valued. Multiple values are represented as colors on tokens on the source side and multi-valued random variables in the target Belief net. Thus in the absence of specific disabling signals, conventional metaphors project feature-values from source to target concepts by setting the relevant nodes in the target domain Belief net to specific values. Projection does not change the instantiated value in the target domain. As was noted in the example shown in Figure 8.2, the maps are able to assert evidence to different temporal copies of the target Belief net.

In general all maps have inputs from the *domain* variable and are activated only in the target context. Instead of direct input from the *domain* variable, maps may be arranged in dominance hierarchies and instead may have only indirect input from the target context, in that they may be activated by a subsuming map that has direct input from the domain variable. If active, they have procedures that **set** and **get** *typed tokens* from the connected source domain node, or **set** and **get** evidence from the connected target domain node. Additionally, maps may be arranged in hierarchies with sub-mappings.

Before detailing the specific map functions, we look at the standard access functions used by all maps. These functions are shown in Figure 8.4. The access functions are fairly straightforward and either get or set the value of the instantiated target variable (Belief net node) or get or set the value of the source domain f–struct with the appropriate token type.

The first three functions in Figure 8.4 pertain to the Belief net side of things, and allow the map to set, get and check the value of the connected target node at a particular time slice. Recall that a map has access to values of a Belief net node at all time slices. The last three functions are specific to the different types of implemented maps. The other functions pertain to getting, setting and creating tokens to by placed at the source domain f–struct location that may specify a specific variable to be set to a value or set of get information about the status of a specific schema. Given these functions, we are ready to look at the implementation of specific maps.

8.2.1 Object Maps

OMAPS represent projections of individual case role-fillers between source and target domains. An example the the mapping between the energetic mover in the domain of motions and the active agent in the domain of abstract actions. Figure 8.5 shows the representation of OMAPS.

The function *EXECUTE_OMAP_FUNCTIONS* describes in pseudocode the logic of the *OMAP* projection functions. Figure 8.5 shows the specific case of economic actors (countries, business, governments, economists, international banks) being mapped onto the concrete domain as moving entities.

Function	**Description**
$CHECK\_CONTEXT(t)$	Return the specific instantiated value (if any) of the target node *target* at time t or *nil* (if none)
$GET\_EVIDENCE(target, t)$	Return $TARGET$ if the domain node is instantiated to Economics or if there is an active connected map
$SET\_EVIDENCE(target, x, t)$	sets the node *target* (time-slice t) to value x
$MARK\_SOURCE(< x, t >)$	mark the source input place with a token of type $< x, t >$.
$CREATE\_TOKEN(< bindname, timestamp >)$	create a token of type specified by the tuple $< bindname, timestamp >$.
$GET\_TOKEN(< t >)$	returns the token value from the source node of the map at time t.
$GET\_STATUS\_TOKEN(source - schema, t)$	Gets the source x-schemas controller state
$GET\_STATUS(map)$	Get the specific map's (any kind) status. Returns *active* or *nil*
$SET\_STATUS(map)$	Set the status of the map to *active* or *nil*.
$EXECUTE\_OMAP\_FUNCTIONS(omap, t)$	Binds Target Roles/Objects to Source Roles/Objects
$EXECUTE\_SMAP\_FUNCTIONS(smap, t)$	Projects schema names and aspect from source to target
$EXECUTE\_PMAP\_FUNCTIONS(pmap, t)$	Projects parameter-values from source to target

Figure 8.4: Common Access Functions for Maps

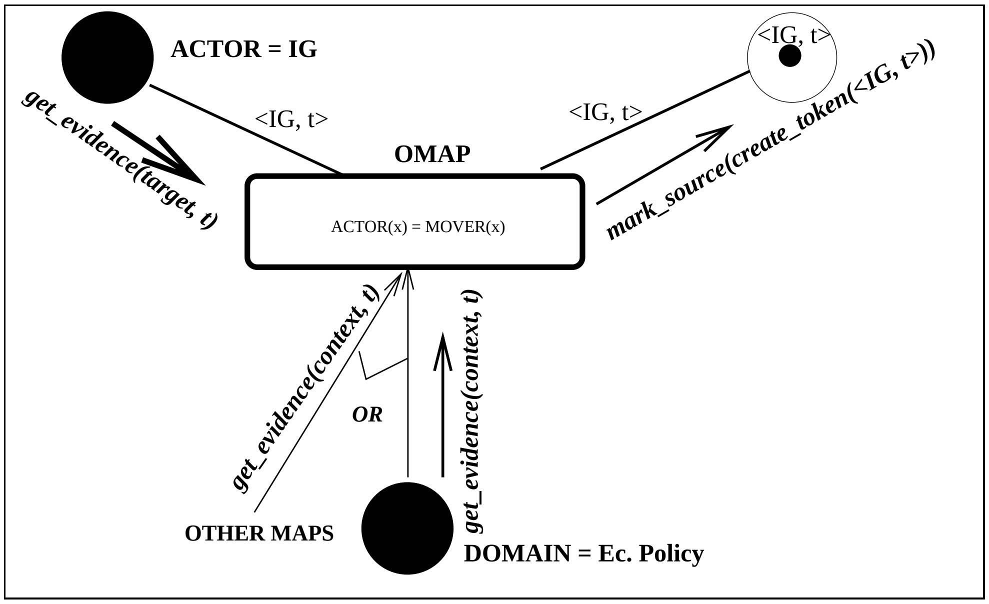


Figure 8.5: Mapping entities across domains.

All maps become active **only** in the context where domain of discourse is about Economics (note that *OMAPS* have an input from the model variable Belief Net node *domain*, which is checked for the instantiated value), if the target node is instantiated (from the input utterance), the binding is propagated to the source node by generating a **unique** token that corresponds to the specific binding in question.

```
EXECUTE_OMAP_FUNCTIONS( omap , t )
 begin
   if CHECK_CONTEXT(t) returns TARGET
   (if domain = Econ, or map is a sub-mapping of an active map)
     set_status(omap) = active
   endif
   if get_status(omap) = active
     if GET_EVIDENCE(target, t ) returns x
     (target domain node is instantiated to value x )
     ThenUnless source is marked with a token of color x
       MARK_SOURCE(CREATE_TOKEN( x , t ))
     endif
     if GET_EVIDENCE(target, t) returns nil
     And there is a token with color x in the source
       Then SET_EVIDENCE(target, x , t )
     endif
   endif
 end
```

The pseudocode above implements a fairly straightforward logic. Basically in the context where the *domain* node is directly input to the map (as was shown in Figure 8.2) and it has been identified to be about economics, **or** if the *OMAP* is called by an already active map (note that metaphor maps can be arranged in hierarchies, where a map can activate sub-maps), it's status variable is set to *Active*.[3] On the *target* side, The functions *GET_EVIDENCE* and *SET_EVIDENCE* either **get** the **instantiated value** (or nil) of the relevant connected target node, or **set** it to a specific value at a specific temporal slice. On the *source* side the functions *MARK_SOURCE* marks the source node of the map with an appropriate token (creating one if necessary).

One interesting thing to note is that these maps are bi-directional, bindings rather than projections. So in the context of abstract actions, the OMAP shown in Figure 8.5 gets activated either if there is a token of the right color in Mover place of the source domain net, or if there is specific evidence from the linguistic input that results in the node *Actor* being clamped to the value x (in this case x is an Economic Actor. Once activated, the

[3]In general, maps may decay when they are not continually activated, so an active map will continue to have residual activity for the next time step. This allows for projection even in cases where the parser has been unable to establish the target context.

Metaphor produces *sets_evidence* to the Actor node, or consumes and produces a token of color x at every step of the simulation.

Of course, to avoid generating redundant tokens every time this function is called, the source node (f–struct place) is checked for the presence of a token of the specified color. If there exists such a token, no new one is generated. The color check is required since the system allows more that one type of agent to be simultaneously referred to in the domain of discourse, thus the **color** carries variable binding information in the x-schema.

Example 14 Brazil fell into recession. ■

In the example below, the OMAP $Mover \Rightarrow Actor$ allows the faller to be bound to the abstract agent (Brazil the Country). Similarly the $States \Rightarrow Locations$ and the sub-mapping $Holes \Rightarrow Negative - Economic - State$ allows the location of the faller (a hole) to be mapped onto the economic state of *recession* .

The implemented system currently has 15 *OMAPS* . Besides the ones mentioned above there are maps that relate economic actors to counter-forces and creating obstacles to movement in the domain of spatial motion (usually governments with restrictive policies). The other set of *OMAPS* pertain to the mapping between Economies as Patients, Economists as doctors (of various sorts from competent ones to downright dangerous quacks), and various undesirable economic processes (such as inflation) as diseases, which are cured by following the Economists' *prescription* of proper policy options.

8.2.2 Parameter Maps

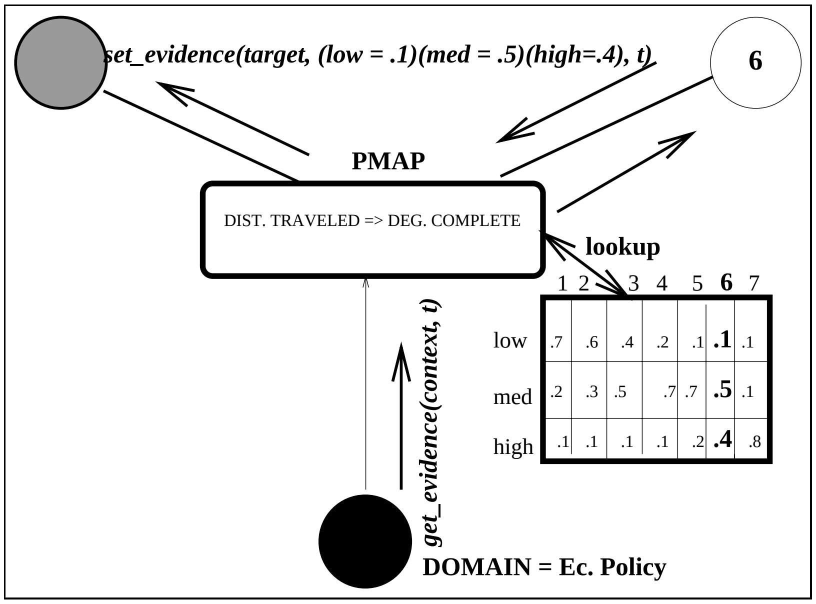
PMAPS map individual x-schema parameter-values across domains. *PMAP* s are also used to map invariant parameters such as rates and forces across domains (ex. rate of motion maps to rate of progress). Figure 8.6 shows the representation of PMAPS and the main *PMAP* execution function. The new function $LOOKUP\_PMAP\_TABLE(key)$ retrieves from an $m \times n$ hash-table (m = number of target values, n = number of source feature values) the appropriate values for source given target and vice-versa.

```
EXECUTE_PMAP_FUNCTIONS( $pmap$ , $t$ )
  BEGIN
    if $CHECK\_CONTEXT(t)$ returns $TARGET$
    (if domain = Econ, or map is a sub-mapping of an active map)
       $set\_status(pmap) = active$
    endif
    if $get\_status(pmap) = active$
      if GET_EVIDENCE(target, $t$ ) returns $x$
      (target domain node is instantiated to value $x$ )
      then LOOKUP_PMAP_TABLE(key:target= $x$ )
      (returns the corresponding src. value $s$ )
      and unless there is a token of color $s$ in the source
        MARK_SOURCE(CREATE_TOKEN( $s$ , $t$ ))
        (in some cases mark with integer number $s$ to source place)
      elseif GET_EVIDENCE(target, $t$ ) returns $nil$
      if there is a token with color $x$ in the source
        Then SET_EVIDENCE(target, LOOKUP_PMAP_TABLE(key:src= $x$ ), $t$ )
      endif
    endif
  END
```

We note that an interesting reason for such hybrid parameter maps is that they are able to change the granularity of distinctions, so an integer number of tokens in the source side may be mapped to a much lower resolution target value. Typical examples are cases where numerical measures such as rates, or distances, or energy levels are mapped onto the Economic Policy domain as different probabilities of *low* , *med. high* (or a 5 point scale with *low−* and *high+*). For instance, the $PMAP$ mapping distance from source (ref. Figure 8.1) to degree of completion, maps distances (values from 1 to 7 , modeled as number of tokens) to degree of completion (three valued Belief net node) was sufficient to make the appropriate distinctions for the examples we looked at.

8.2.3 Schema Maps

Schema maps or $SMAPS$ project events from the source domain onto their images in the target domain. For instance the $SMAP$ *Motion IS Action* shown in Figure 8.1 maps motion events onto action events. Shown below is the pseudocode for the main loop controlling $SMAP$ execution $EXECUTE\_SMAP\_FUNCTIONS$ $(smap, t)$. The


set_evidence(target, (low = .1)(med = .5)(high=.4), t)
6
PMAP
DIST. TRAVELED => DEG. COMPLETE
lookup
get_evidence(context, t)
DOMAIN = Ec. Policy

	1	2	3	4	5	**6**	7
low	.7	.6	.4	.2	.1	**.1**	.1
med	.2	.3	.5	.7	.7	**.5**	.1
high	.1	.1	.1	.1	.2	**.4**	.8

Figure 8.6: Mapping feature-values across domains.

interesting thing to note is that this is a projective mapping where the control status of the source x-schema is invariantly mapped to the status value for the action. The rich inferences that result from changes in agent state as a result of control transitions in perceptual and motor actions, reflect in activating many metaphor maps and thereby allow for the new evidence and bindings to be entered about the target domain regarding goals, outcomes, resources, and difficulties in plan implementation.

```
EXECUTE_SMAP_FUNCTIONS( $smap$ , $t$ )
  begin
    if $CHECK\_CONTEXT(t)$ returns $TARGET$
    (if domain = Econ, or map is a sub-mapping of an active map)
      $set\_status(smap) = active$
    endif
    if $get\_status(smap) = active$
      AND there is a token marking the source schema node
    Then GET_STATUS_TOKEN (source)
      (returns a token typed $enable$ , $ready$ , $ongoing$ , $done$ , etc.)
      and SET_EVIDENCE(target, $status = token - color$ , $t$ )
      and if target node is uninstantiated
      SET_EVIDENCE ( $target$ , $x$ , $t$ )
    Else do nothing
  end
```

The example in Figure 8.7 allows an interpreter to infer that moving forward in the context of an economic policy corresponds to making progress in the policy implementation. Of course, the **aspect** of the underlying schema is projected as well. Thus in the example above the difference in *is dealing with the obstacle* is mapped invariantly onto the target domain as monitoring some difficulty and taking corrective action. versus *has dealt with the obstacle* is mapped as the setting evidence of $difficulty = false$.

The combination of $SMAPS$ and $PMAPS$ are able to map parameterized events. For instance the combination of the $PMAP$ *forward IS progress* and the $SMAP$ above, $move(forward) \Rightarrow make(progress)$, $move(back) \Rightarrow regress$, etc. In combination with $OMAPS$, $SMAPS$ are able to map entire verb-sub-categorization frames and relational structures from one domain to another, such as in the example shown in Figure 8.1 we are able to map energetic movers (the agent of Moving) to wilful actors (the agents of acting). Similarly in the example described in the last section the combination of the $SMAP$ $fall \Rightarrow fail$ and the $OMAP$ s $Actor \Rightarrow Agent$, $States \Rightarrow Locations$, and $Recessions \Rightarrow Holes$, the system is able to map the relational structure $fall - into(Brazil, Recession)$ to $fall - into(Br : Agent, recession : hole)$.

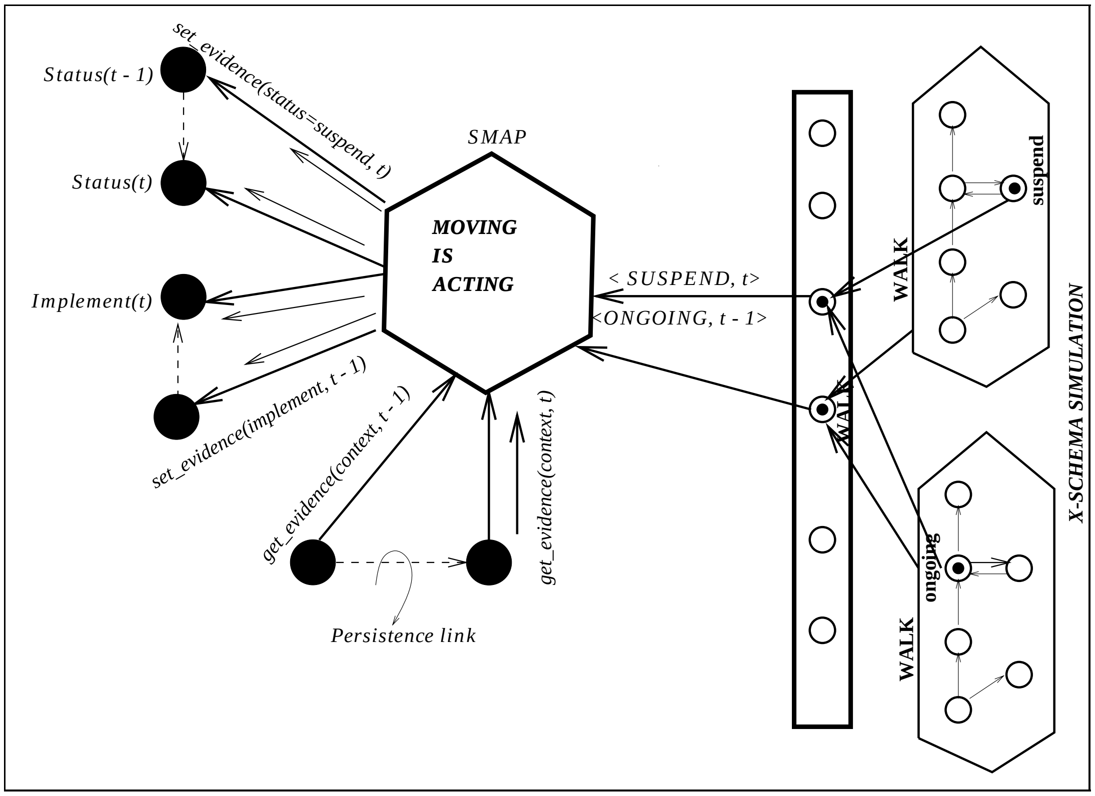


Figure 8.7: Mapping events across domains over multiple time steps. The figure shows invoking the $EXECUTE\_SMAP\_FUNCTIONS$ function for the $SMAP$ $Moving \Rightarrow Acting$ twice; once with the time argument at the previous time step $t-1$, and once in the current time step t. In the first case, the *ongoing* aspect of motion at $t-1$ and the *suspend* aspect at time t is projected as a policy implementation that is ongoing at $t-1$, and suspended at time t respectively. The broken lines between the status and policy nodes (to the left of the figure) represent the temporal persistence links between network copies (their influence is overridden through direct evidence being asserted at both time steps).

8.3 Using The Model For Interpretation

Figure 8.2 showed the product of interpreting the input utterance "Indian Government Stumbling in Implementing Liberalization Plan". To model causal narrative interpretation, we assume that the reader's epistemic state consists of a set of features each of which can take on one of a discrete set of values with varying degrees of belief. Thus, this model, a story represents the specification of a *partial* trajectory over epistemic states. The remaining features are estimated using known target domain inter-feature correlations as well as from metaphoric projections from the highly compiled domain knowledge (x-schemas). Metaphoric projections of x-schema executions creates new evidence to the Belief net by clamping some of the target features to specific values. In this section, we outline the algorithmic details of our system, by using the input above as an example. First, we will describe in detail the I/O behavior of our system for this example, since we will use many of the same display techniques to describe results of running our program on a variety of examples (ref. Chapter 9). Then we will go through the specific control and algorithmic details of the implemented program, illustrating some of the more trickier issues using a program trace of the system on the example above.

8.3.1 I/O Behavior

Table 8.1 illustrates the I/0 behavior of the implemented system interpreting the newspaper headline *Indian Government Stumbling in Implementing Liberalization Plan.* The input to the system is a set of feature-value pairs (called "f–structs") resulting from a partial parse. Table 8.1 shows the input to the system as occurring at time $t = 1$. Columns in the table show the value of various features at different time steps. The first set of feature values represent relevant prior values for variables in the target domain network. For instance, as was discussed earlier, the value for the *Actor* variable has a high prior probability of being the US Government.($P(Actor = USG, 0) = .7$). This is reflected in the value of the variable at time $t = 0$.[4]

[4]In future I/O displays, we only display prior values if they are relevant to the processing of the input utterance. Otherwise, the reader is to assume that the economic variables have the same prior values as shown in Chapter 11, Section 11.3.

Table 8.1: I/O Behavior for the input *Indian Gov. Stumbling in implementing Liberalization plan.* The values are shown for the relevant priors, input, output, and for active metaphor maps.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$
PRIOR			
Actor = *USG*	t(.7)		
Domain =Ec. Policy	t (.7)		
INPUT FEATURES			
Event = stumble		t	
Domain =Ec. Policy		t	
Policy = Liberalization		t	
Actor = *IG*		t	
Aspect = Present-Prog		t	
Ut-type = Description		t	
ACTIVE MAPS			
$M_o(Mover \Rightarrow Actor))$		A	A
$M_s(Walk \Rightarrow implement(policy))$		A	A
$M_s(Walk(ongoing) \Rightarrow Status = ongoing)$	A		
$M_s(Walk(suspend) \Rightarrow Status = suspend)$		A	A
$M_p(Obstacle \Rightarrow Difficulty)$		A	A
TARGET(OUTPUT) FEATURES			
Event = stumble		t	
Domain = Ec. Policy	**t(.8)**	t	**t (.8)**
Ec. Policy =Liberalization	**t(.8)**	t	**t (.8)**
Aspect = Present-Prog		t	
Actor = *IG*	t(.8)	t	t(.8)
Status(Pol) = *ongoing*	**t**	**f**	**t(.3)**
Status(Pol) = *suspended*	**f**	**t**	**t(.7)**
Difficulty	t(.7)	**t**	**t(.7)**
Outcome = succeed	**t(.6)**	**t(.4)**	**t (.5)**
Goal = free-trade ∧ deregulation	**t**	**t**	**t**

A note about the display method. In cases where I use a table to display the results of the program on a specific example, both input and output f–structs are shown in a tabular form. In most tables, I have changed multi-valued variables to the value of the instantiated (or high probability) state to label the appropriate row of the table, so that cell entries only convey the probability of that specific state (as true t or false f). The individual columns of the table represent the time slices. Cells in the tables indicate the value of the relevant feature at that time slice. When appropriate the actual *degree_of_belief* for the specific value at the specific time slice is shown is brackets as well. Unless otherwise specified target values whose posterior probability falls below the threshold value are not shown (threshold was .6 unless otherwise specified). Cells that represent the cases where metaphors are active, are labeled with A. Cells with no entries represent inestimable or irrelevant values. The specific target domain feature-values that became active as result of metaphoric inference and associated propagation in the temporal Belief net are shown in **bold**. In cases where the input contained the value of a feature that had to be re-estimated for other times, only the re-estimated values are shown in boldface. An example of such a feature is $Domain = Ec.Policy$, where the input f–struct supplied this information at time $t = 1$, which had to be reestimated at time $t = 0$, etc. Only the re-estimated values are in boldface. The conventions above are followed in Chapter 9 as well.

The pre-parsed input is shown as setting the values of relevant features at time $t = 1$. These values appear in Table 8.1 under the heading *INPUT FEATURES.* Since the input is available at time $t = 1$, there are entries only for that time slot. Notice that the input essentially instantiates specific target features (such as $Actor = IG$) and also contains aspectual and source event information. This information is used to set the controller status and the relevant x-schema parameters, possibly triggering execution.

The result of processing the input in Table 8.1 is a set of new bindings asserted in the target domain resulting in an updated posterior for other variables. As part of the processing of the input various x-schemas may execute and various **metaphor maps** may become active transferring bindings and projecting x-schema inferential products. For example Table 8.1 shows the $OMAP \;\; M_o(Mover \Rightarrow Actor))$, and the $M_s(Walk \Rightarrow implement(policy))$ along with the aspectual maps $M_s(Walk(ongoing) \Rightarrow Status = ongoing)$ which maps the ongoing status of the walking event at time $t = 0$ onto the target by setting evidence for an ongoing policy implementation at time $t = 0$, and the $M_s(Walk(suspend) \Rightarrow Status = suspend)$ aspectual map that projects the suspension of the walk process at time $t = 1$ onto the economic domain as a suspended policy implementation. Also active is the $PMAP \;\; M_p(Obstacle \Rightarrow Difficulty)$ which asserts a plan difficulty. All these projections are available in real-time as x-schema inferences.

Once x-schema inferential products are projected, the $BELIEF\_UPDATE$ algorithm propagates the effect of the new evidence throughout the temporally extended network. **Bold** entries under the heading *TARGET (OUTPUT) FEATURES* in Table 8.1

above correspond to cases where the value of the relevant variable was a result of projecting x-schema inferences. Of particular interest is the *context setting inference* which projects the embodied knowledge that stumble occurs as a result of an obstacle while executing a step (causing an interruption to forward motion) to the target as plan difficulty (causing a temporary suspension). Another interesting binding occurs as a result of the source domain knowledge that stumble **may** lead to a *fall* which is mapped onto the target as an enhanced likelihood of *plan failure*. Thus we note that while *stumble* is **not directly** mapped in our system as a meaningful concept in the domain of Economics, through **inferential projection** from maps such as *Falling MAPS TO Plan Failure* and *Obstacle MAPS TO Plan Difficulty* the system is able to assert a target context where an ongoing plan is experiencing difficulty increasing the chance of failure as the outcome. This kind of real-time defeasable inferential projection is a novel feature of our model and we believe a crucial requirement for any model of metaphor interpretation.

Of course, many possible x-schema bindings, especially those that don't activate any conventional metaphor are invalid are thus have no impact on the agent's epistemic state (for example the source inference stumble $\Rightarrow$ losing balance). Thus the inferences that are actually made are context-sensitive and depend on the target domain and the associated set of metaphoric maps.

The resultant target network state shown in Table 8.1 is now a prior for processing the next input at stage $t = 2$. Background knowledge is encoded as the network state at $t = 0$. Potentially target inferences can go forward and backward in time in the estimation of the most probable explanation of the input story.

8.4 Algorithmic Details

The main loop consists of waiting for input f–structs, or queries to return the Maximum Probable Explanation (MPE) of the variables in $TARGET\_NET$. The interpreter has access to the target domain Bayes net $TARGET\_NET$, the Discourse Bayes net $DISCOURSE\_NET$, the various Metaphor maps M, and the concrete domain x-schemas X, the f–struct F, and keeps a local variable corresponding to the last step index $t-1$

(current evidence is at time t).[5] The $TOP\_LEVEL\_CONTROL$ function is waiting for one of three kinds of input. One is a request to process a new input f–struct, the other two input pertain to making updates on the target domain networks and returning the highest posterior trajectory (from algorithms described in Chapter 7).

```
TOP_LEVEL_CONTROL()
  static:
    Target ( $TARGET\_NET$ ), Discourse ( $DISCOURSE\_NET$ ),
    Metaphor-Maps( $M$ ), x-schemas ( $X$ ), f–struct ( $F$ )
  local variables:
    $last\_time\_index$  $(t - 1)$
  loop
    (wait for input and process the appropriate cases)
    if $INPUT(I)$ is read
      (if you see an input f–struct)
    then $INTERPRET\_INPUT(I)$
      (then process it)
    if $GET\_MPE(network)$ is read
      (if an explanation is requested)
    then $MPE(network)$
      (return MPE Assignment to the Belief net $network$ )
    if $GET\_THRESHOLD(Network, Maxt, Thresh)$ is read
      (if an thresholded update for a few time steps is requested)
    then $THRESHOLD(Network, Maxt, Thresh)$
      (return all values above Thresh in $Network$ upto time $maxt$ )
```

$INTERPRET\_INPUT(I)$ processes input f–structs and is responsible for entering evidence, binding to x-schemas and running the map functions to create evidence in the $TARGET\_NET$ and the $DISCOURSE\_NET$. $MPE(network)$, and $THRESHOLD(Network, Maxt, Thresh)$ run the MPE and $BELIEF\_UPDATE$ algorithms on the appropriate Belief net. These functions are adapted from IDEAL.

The main function of interest is $INTERPRET\_INPUT$. The best way to illustrate its operation is through a trace, and here we use an familiar example, namely the input 'Indian Gov. Stumbling in implementing Liberalization Plan".

[5]All algorithms in Chapter 8 were implemented in CMU Common Lisp, a public domain implementation of Common Lisp and CLOS. The Belief Update and MPE algorithms were adapted from the public domain Belief network package IDEAL (Srinivas & Breese 1990).

```
$INTERPRET\_INPUT(I)$
  **input**: $I$ is the input f–struct
  **Begin**
     $INCORPORATE\_INPUT(I, t)$
       Enter new evidence in target nets and bind to x-schemas
     $EXECUTE\_BOUND\_SCHEMAS(t)$
       Apply the Next Step Firing function from Chapter 3
     $BELIEF\_UPDATE(TARGET\_NET, t)$
       Update the target network
     $BELIEF\_UPDATE(DISCOURSE\_NET, t)$
       Update the discourse network
       Both the above functions are from Chapter 7
  **End**
```

As shown above interpretation of an input f–struct consists of several steps. First the input example is *incorporated* into the existing structure. This is done with the $INCORPORATE\_INPUT$ function shown below.

```
$INCORPORATE\_INPUT(I, t)$
  **input**:
     $I$ is the input f–struct
     $t$ is the current time-slice
  **Begin**
     $UPDATE\_TARGET\_NET(I, t)$
       Enter new evidence in $TARGET\_NET$
     $UPDATE\_DISCOURSE\_NET(I, t)$
       Enter new evidence in $DISCOURSE\_NET$
     $UPDATE\_MMAPS(I, t)$
       Execute map functions
     $UPDATE\_SCHEMAS(t)$
       Call schema update functions
  **End**
```

First, the target domain Belief net $TARGET\_NET$ is instantiated, which includes setting evidence for specific nodes. For example, the input f–struct that results from a partial parse of the input sentence.

```
((context Economic Policy)(type Liberalization)(actor IG)(event Stumble)
(aspect Progressive)(ut-type Description))
```

Most lexical entries are fairly straightforward, in that they either *set_evidence* for specific nodes in the Belief net. For instance, the input above is parsed as

```
set_evidence(TARGET_NET, Domain= Ec. Policy, t = 1)
set_evidence(TARGET_NET, Ec. Policy=Liberalization, t = 1)
set_evidence(TARGET_NET, Actor=IG, t = 1)
```

Events are a bit more complicated, since they are more dynamic, bind to x-schemas, or set the Agent state f–struct which may result and may set evidence to multiple time steps.

```
(defevent (stumble, t)
    (bind_to_schema ('#n(ongoing(walk), t-1))
    (create_fs_marking('#n(bump), t)))
(defevent (slide, t)
     (bind_to_schema('#n (ongoing(walk), t-1)))
     (create_fs_marking('#n(slippery(gr), t))))
```

The function *bind_to_schema* marks the *enable* place of the schema in question (here WALK) and sets the controller to a specific state (such as *ongoing*). The function *create_fs_marking* marks specific places of the Agent state f–struct with tokens. The $'\#n$ macro translates names to the corresponding objects. The code above results in setting the target domain Belief Net variable *Domain* to the state *Economic_Policy* , the target domain Belief Net variable *Policy* to the value *Liberalization* , and the target domain Belief Net variable *Actor* to the value *IG* .

The *event* functions $defevent(x)$ (in this case $x = stumble$) encode the specific method by which a linguistic item interacts with the x-schemas. For instance, the linguistic input corresponding to *stumble* , when asserted at time t , binds to the WALK x-schema at time $t-1$, and creates a token corresponding to a **bump** at time t .

At this point, $UPDATE\_DISCOURSE\_NET(I, 1)$ is called, resulting in the

```
set_evidence(DISCOURSE_NET, Ut-Type=Description, t = 1)
```

Now all the direct Belief net interventions from the linguistic input is complete, and $UPDATE\_MMAPS$ is called, where the metaphor maps are updated, as a result of the new evidence entered. Here, mostly object maps become active and transfer bindings from the appropriate target domain node to create a token in the source domain node. The token is a tuple $< bindname, timestamp >$, where *bindname* is the appropriate binding and *timestamp* is the input utterance time $(0-4)$. These two steps are part of the $INCORPORATE\_INPUT$ algorithm shown below. A trace of this step of the algorithm for the input in question is shown below.

```
ttyp3 tiramisu:~/thesis/code/interpreter> (load "istruct-stumble")
(Input ((Context EconomicPolicy)(Type Liberalization)(Agent IG)
(Event stumble)(Aspect Progressive)))
*****INTERPRET_INPUT(<#I223>)**
*******INCORPORATE_INPUT(<#I223>)***
*********UPDATE_BN(<#I223>, <TARGET_NET02>, t = 1 )***
***parsing <#I223>***
set_evidence(TARGET_NET, Domain=Economic Policy, t = 1)
******* done ****
set_evidence(TARGET_NET, Policy = Liberalization, t =1)
****done *****
set_evidence(TARGET_NET, Actor = IG, t= 1)
****done *****
*********done UPDATE_BN(<#I223>,<#TARGET_NET22>, t = 1))***
*********UPDATE_BN(<#I223>,
set_evidence(DISCOURSE_NET, Ut-Type=Description, t = 1)
*********done UPDATE_BN(<#I223>, <#TARGET_NET22>, t = 1)***
```

At this point the metaphor maps are updated. Mostly $OMAPS$ get activated since the target domain roles are mapped into source domain roles. In this case the new evidence asserted activates the $Actors \Rightarrow Movers\ \ OMAP$.

```
*********UPDATE_MMAPS(<#I223>, t = 1)***
******Checking enabled Maps********
(M_o3) Active with target node ``Actor= IG''
****** Executing OMAP Maps *****
Marking embodied node P44(P44.Name = Mover),
Token T112 #(<IG>, t1) generated
**No more active maps
*********done UPDATE_MMAPS(<#I223>..)***
```

The result is a token of type $< IG, t_1 >$ being deposited in the f–struct places corresponding to the Mover of the spatial motion. We are now ready to run the $UPDATE\_SCHEMA$ function.

```
*********UPDATE_SCHEMAS(<#I223>)***
**** parsing-event(Event stumble) ***
Binding to schema <#S12> (S12.Name = ``Walk'')
Bound E121 ``Enable input place''P44(P44.Name = T112)
Marking place P123(P123.Name =``Bump'')
Token T135 #<nil, t1> generated
***done
****parsing (Aspect Progressive)*****
Attempting to mark <# S12> controller <ongoing>
Mark <# S12>.controller <ongoing>
Token T234 #<S12.A148>, t0> generated
Marking <# S12>.controller <ongoing> with T234
****done
*********done UPDATE_SCHEMAS(<#I223>)***
*******INCORPORATE_INPUT(<#I223>)n***
```

At this point, we are done incorporating the new example. We have entered new evidence to the Target Belief networks, and have successfully enabled the WALK schema with an agent that corresponds to the Indian Government (through the $Mover \Rightarrow Actor$ $OMAP$. The interesting thing to note is that the token that gets deposited in the ongoing node of the WALK schema $T234$ is timestamped t_0 signifying prior context for the interpretation. This comes from the lexical entry for the *stumble* event explained earlier. Thus the situation at the end of this phase, is that the WALK schema is active and *ongoing* and a *bump* has been asserted (Figure 6.8).

Now we are ready to execute the bound schema walk in the presence of a bump. This is exactly the situation shown in Chapter 6 (Figure 6.8).
The $EXECUTE\_BOUND\_SCHEMAS$ function is shown below.

```
EXECUTE_BOUND_SCHEMAS($t$)
  input: current time slice $t$.
  Begin
    loop until SMAP_project?
      continue until a new event projection is made
       PERFORM_ONE_STEP
        execute next step on active x-schemas
       EXECUTE_SMAP_FUNCTIONS ( $\mathbf{t_i < t}$ )
        project existing events (with aspect) to target
       EXECUTE_PMAP_FUNCTIONS ($\mathbf{t}$)
        project schema parameters
    end loop
    EXECUTE_SMAP_FUNCTIONS($t$)
      project new events (with aspect) to target
  end
```

The key variable to notice here is the binary valued function *SMAP_project?*, which is called after every step of the x-schema simulation. This function returns *true* if any new *SMAP* is activated at time t, and the inner *loop* is exited. Schema executions for times $t_i < t$ are not sufficient to get out of the loop (this is required since some lexical items refer set the context as well as specify what the current event is). For example, the lexical item *stumble* simultaneously sets the context to be an ongoing walk which encounters a bump while taking a step. This was outlined in detail in Chapter 4, in Figure 6.8 - Figure 6.11. Upon exiting the loop, the x-schema state is preserved, but since we have asserted something new about the target inference propagation stops till the next input. Thus the loop is exited as soon as there is some *SMAP* that can be activated from x-schema executions. The problem of there not being any *SMAP* that can be activated was never encountered, partly because embodied words usually refer directly to x-schemas that have projections onto the target (such as FALL or MOVE, or REMOVE-OBSTACLE), but sometimes execution of x-schemas may asynchronously trigger long inferential chains leading to many possible target domain inferences. The function *SMAP_project?* is meant to constrain such long chains, since the loop is exited after one SMAP execution. A rationale for this choice is outlined in Section 8.5.

Going on with our program trace, we are still in the *INTERPRET_INPUT* procedure, and are in the *EXECUTE_BOUND_SCHEMAS* routine.

```
**** EXECUTE_BOUND_SCHEMAS***
*******START LOOP
**** PERFORM_ONE_STEP ******
***** Checking Bound Schemas***
(<#S12> (S12.Name = ``Walk''))
Token E121 ``Enable input place''P44
Token T234 #<S12.A148>, t0> at <# S12>.controller <ongoing>
No more bound schemas
**** EXECUTE_SMAP_FUNCTIONS(ti < t) ******
((SMAP MOVE_399 active
       (MOVE_399.name = MOVE=>IMPLEMENT POLICY)
(MOVE_399.status = ONGOING)
(TIME = 0))
****Executing SMAPS********
set_evidence (TARGET_NET, IMPLEMENT POLICY, t = 0)
set_evidence (TARGET_NET, POLICY STATUS = ONGOING, t = 0)
*** done SMAPS (t =0)
```

At this point we have found one schema that was active at time $t = 0$, hence satisfies $t_i < t$, and we are execute the $SMAP$ which asserts an *ongoing* plan at time $t = 0$ on by setting evidence on the $t = 0$ copy of $TARGET\_NET$. But we are still inside the loop, and hence we execute bound schemas to advance the simulation to $t = 1$, the current time.

```
**** EXECUTE_BOUND_SCHEMAS***
*******START LOOP
**** PERFORM_ONE_STEP ******
Executing S12 (S12.Name = ``Walk'')
***** Checking Enabled Transitions**
Token T112 #(<IG>, t1) (Actor)
Token T135 #<nil, t1> at P123(P123.Name =``BUMP'')
Token T234 #<S12.A148>, t0> at <# S12>.controller <ongoing>
(S323 (S323.Name = ``Interrupt''))
*****Executing Next-Step Function *****
Firing SS323(S323.Name = ``Interrupt''))
Token T234 #<S12.A148>, t1> at <# S12>.controller <suspend>
Token T187 #<IG>, t1> generated
Token T187 #<IG>, t1> at P63(P63.name = ``UNSTEADY'')
```

The above trace corresponds exactly to the step function taking the system from the state in Figure 6.8 to the state in Figure 6.9. At this point, we are ready to $EXECUTE\_PMAP\_FUNCTIONS$.

```
**** EXECUTE_PMAP_FUNCTIONS ******
(PMAP P_932 active)(P_932.name = BUMP => DIFFICULTY)
set_evidence (TARGET_NET, DIFFICULTY = true, t = 1))
>>>>>>>>>SMAP_PROJECT? Returned nil
LOOP
```

While the $PMAP$ projection asserts evidence of a difficulty in the target domain Belief net at time $t = 1$, still no $SMAP$ projection has occurred at time $t = 1$, hence the $SMAP\_PROJECT?$ function returns nil, and we repeat the loop again.

```
**** PERFORM_ONE_STEP ******
***** Checking Bound Schemas***
((<#T53> (T52.Name = ``Trip''))
Token T112 #(<IG>, t1) (Agent)
Token T004 #<nil, t1> at P86(P86.name = ``CONTROL(LOC)'')
Token T187 #<<IG, t1> at P63(P63.name = ``UNSTEADY'')
Token T008 #<nil, t1> at P63(P63.name = ``STABLE'')
***** Checking Enabled Transitions**
((<#T53> (T52.Name = ``Trip''))
*****Executing Next-Step Function *****
Consuming Token T004 #<nil, t1> at P86(P86.name = ``CONTROL(LOC)'')
Consuming Token T187 #<<IG, t1> at P63(P63.name = ``UNSTEADY'')
Consuming Token T008 #<nil, t1> at P99(P99.name = ``STABLE'')
Adding Token T187 #<<IG, t1> at P63(P63.name = ``UNSTEADY'')
Adding Token T303 #<IG, t1> at F331.A24(F331.A24.name = ``Fall'
(status = ready))
******** Goal-based Enabling detected ***********
****Instantaneous transition fired******
**** New bound schema
((S17 (S17.Name = ``Stabilize''))
***************
Resource = Energy (# 2)
Inh = stable
******************
>>>>>>>>>SMAP_PROJECT? Returned t
(SMAP FFAIL_121 active)(status = ready)
(SMAP STCO_133 active)(status = enabled)
(FFAIL_121.name   = FALL=>OUTCOME=FAIL) Active(status = ready)
****************Exit Loop***************
****** DONE PERFORM_ONE_STEP
```

Continuing with the x-schema simulations, we now are in the situation depicted in Figure 6.9 - Figure 6.10. The TRIP x-schema fires, and the mover has lost control of his location, and become unstable, which enables the x-schema STABILIZE , requiring energy and resources insufficient to transition to the *ready* stage. At this point we have two possible x-schemas (FALL and STABILIZE) enabled (FALL *ready* to go while STABILIZE just *enabled*), and the one corresponding $SMAP$ enabled namely $Fall \Rightarrow Fail$. [6]

At this point, we have completed the source domain inferences, and have found a map from falling to failing, or ready-to-fall to ready-to-fail. At this point, the next step is to $EXECUTE\_SMAP\_FUNCTIONS$ and set new evidence in the target Belief Net.

```
*******EXECUTE_SMAP_FUNCTIONS
******* Timestamp t=1
*** Checking active SMAPS*********
((SMAP FFAIL_121 active
       (FFAIL_121.name   = FALL=>OUTCOME=FAIL)
(FFAIL_121.status = READY))
****************************
****Executing SMAPS********
set_evidence (TARGET_NET, OUTCOME = FAIL(0.6), t = 1)
********** done*****************
*******DONE EXECUTE_SMAP_FUNCTIONS*********
*******DONE EXECUTE_BOUND_SCHEMAS*******
********BELIEF_UPDATE(TARGET_NET, 1) ****
*******checking t =0 evidence******
*******checking t =1 evidence******
******done BELIEF_UPDATE(TARGET_NET,1) ****
**************BELIEF_UPDATE(DISCOURSE_NET, 1) ****
*******checking t =0 evidence******
*******checking t =1 evidence******
************** done BELIEF_UPDATE(DISCOURSE_NET, 1) ****
ttyp3 tiramisu:~/thesis/code/interpreter>
```

At this point we are ready to perform belief updates on $TARGET\_NET$ and $DISCOURSE\_NET$ using the update algorithms described in the last chapter. The resultant belief state is thresholded (the current value is .6) and the output nodes and values above threshold in shown in Table 8.1.

```
(return-threshold TARGET_NET, 1, .6)
```

[6]Note there is another $SMAP$ $Stabilize \Rightarrow Status(Pol) = resume$ but to keep the exposition simple we omit this possibility. Note also that in general many x-schamas in the source domain such as TRIP will **not** activate any $SMAP$ and hence will only have impact on the processing if they lead to an $SMAP$ activation, as in this case with $Fall \Rightarrow Fail$.

```
((context Economic Policy)(type Liberalization)(actor IG)
(event Stumble)(aspect Progressive)(status (ongoing 0 1)
(suspend 1 1))(difficulty (t 0 .7)(t 1 1))(outcome (fail 1 .6))
(goal (free-trade  0 1)(dereg 0 1)(free-trade 1 1)(dereg 1 1)))
ttyp3 tiramisu:~/thesis/code/interpreter>
```

The function $RETURN\_THRESHOLD(BN, Maxt, Thresh)$ is called, where $Maxt$ is the maximum time step for which values need be returned (0 returns prior values, 1 returns prior and updated values at $t = 1$, etc.), and $Thresh$ is the threshold value, only variables with posterior values above threshold are returned. The format is $(var_j(val_1, time_0, Prob.), \ldots (val_i, time_{maxt}, Prob.))$ where i is the number of values above threshold for variable j . At this point further update operations with $Maxt$ set to 2 will propagate the influence of the evidence over the entire network to fill in the other time slot values in an analogous manner. Note that in presenting the results (as in the tabulated output shown in Table' 8.1 or the results in Chapter 9) these values appear in **boldface** and indicate the result of metaphoric reasoning about the specific event descriptions.

8.5 Comparison with Other Models

It is now generally accepted that metaphor interpretation requires the ability to explicitly represent the source and target domains as well as the metaphor maps themselves. This was best illustrated in the work of (Martin 1990) which showed how explicitly representing conventional metaphors as structured associations between source and target concepts is useful both in interpretation and for biasing search during generalization to new mappings. Martin was also one of the early modelers to take seriously the evidence that metaphors and other non-literal constructions are accessed without additional cost (compared to literal constructions) fairly early in interpretation. Both these facts have been accepted by later modelers, resulting in knowledge-rich approaches becoming the primary method of choice for several implemented metaphor interpretation systems (Martin 1990; Barnden *et al.* 1994; Indurkhya 1992). [7] The approaches above share many goals and have informed the work presented here. For instance, as in Martin's work, we treat metaphor maps as first–class objects that can be arranged in hierarchies. We also agree that the

[7]The reader is also referred to (Martin 1990) for other convincing arguments against knowledge-deficient approaches to metaphor interpretation.

metaphorical use of words that share a subsumptive relationship in the target (called "core" relations in (Martin 1990)) results in mapping to words with similar subsumptive relations in the source domain. In addition, we believe that metaphor mappings are sensitive to rather subtle semantic distinctions between concepts. Our results (ref. Chapter 9) suggest the need for detailed semantic representation of source and target domains to properly investigate the role of metaphoric mappings in interpretation.

In influential early work on metaphor, (Carbonell 1982) looked at how metaphors are able to invariantly map plans and strategies across domains. The work presented here is consistent with Carbonell's observations that metaphors transfer planning knowledge across domains. However, in contrast to Carbonell's work, which was concerned with modeling long metaphoric inferential chains where entire plans and problem solving strategies are transferred across domains, the data we looked at suggests fairly short source domain inferential chains where key dynamic information (such as changes in rates, resources, dynamic thwarting or enabling of goals) is transferred. Of course, one could potentially create long source domain inferential chains by continually using a related set of metaphors from single source domain, but this kind of sustained and consistent mapping was rare in the types of narratives we looked at. In connection with Carbonell's observation that certain distinctions are always *invariantly* projected across domains, our results (ref. Chapter 9) suggest that aspectual distinctions are fall into this category and that aspect is an inherent characteristic of events that is invariantly projected across domains.

Our system also accords with observations (Indurkhya 1992; Barnden *et al.* 1994) that metaphoric projection often occurs from densely structured (familiar) domains to sparsely structured (unfamiliar) domains. In our model, this simply translates to the fact that familiar domains have many more x-schemas and inter-schema connections (more *dynamic* knowledge) that can be exploited by metaphors.

While our metaphor reasoning system is quite similar to other knowledge-based approaches, there are a few additional features of our model that basically stem from paying greater attention to fine-grained dynamic characteristics of events and their role in metaphoric projection.

1. First, our representation of actions and events with durations is more fine-grained than other systems we are aware of. Specifically, we believe our system to be novel

in being able to model temporal and and aspectual inferences across domains. As we will see in Chapter 9, both these features are routinely exploited by newspaper narratives of abstract events and policies.

2. Our use of a temporally extended Belief network to represent target domain knowledge allows us to uniformly combine direct linguistic input and prior knowledge with results of metaphoric projections in a single normative framework. Chapter 9 contains examples where the system is able to generate different interpretations for the same discourse segment when the interpreter has different priors or specific biases.

3. Projections about entities in the target can be made over multiple time steps (as in the case of stumble). The influence of metaphoric projections is able to assert information about past states as well as bias future expectations in the temporally extended Belief network. Chapter 9 shows how this enables us to interpret expressions such as *back on track*.

4. We believe that important information about communicative intent and speaker evaluation of different situations or policies is often metaphorically communicated and Chapter 9 shows some examples of our systems' capability of making these inferences. We feel this avenue can be productively explored using the mechanisms in our system.

5. While most approaches require extra resources to process novel expressions (requiring some form of metaphor extension), our approach is able to explain why **some** novel expressions can be processed without additional cost through inferential projection from the source domain (as in the case of the "stumble" example).

In short, we believe using an x-schema-based dynamic semantics allows us to model a set of interesting distinctions made by metaphoric expressions. On the flip side, our focus on narrow set of expressions, our impoverished target domain and the inability of our system to extend existing metaphors are important shortcomings. Some of these issues are discussed in detail in Chapter 10.

8.6 Issues with the Current Implementation

The system presented in this chapter was implemented to investigate how metaphoric projections of motion and manipulation words could make use of the context sensitive nature of the x-schema based representation to specify features of abstract plans and actions. The performance of our system on discourse fragments from standard newspaper sources of articles on international economics is the subject of the next chapter (Chapter 9). However, the implementation of our metaphor interpretation system brought up some important issues. First, the set of maps that are implemented was based on an intuitive judgment based on our story database (ref. Chapter 11, Section 11.1) combined with some known Event structure metaphors obtained from linguistic research (which are less fine-grained than the maps we seem to need). We would like a more systematic method to obtain these maps. Second, we need an answer to the question *how much source domain inference is appropriate?*. Conceivably, inference could continue in the source domain until a *deadlock* is reached and no transition is enabled. While this might be appropriate in some instances, our results suggest otherwise. Section 8.6.1 discusses this issue in greater detail. Finally, in Chapter 2 we outlined the overall philosophy that motivates this work which requires that we be able to express our models at the connectionist level. Until Chapter 6, we were successful, but as the reader has obviously noticed, the metaphor model presented here is not connectionist. Some of these issues are discussed further.

8.6.1 When does propagation stop?

The first thing we noticed about our database was that typical newspaper articles do not require extended propagation of source domain inferences. In contrast, they seem to pick up some key features of the motion and manipulation terms that exploit the real-time and context-sensitive nature of the representation (such as aspect, resources, goal disabling/enabling, etc.). While it is certainly true that one can construct narratives where extended source domain inferences are required, most cases relied on "quick and dirty" inferences provided by embodied terms.

This is related to the issue brought out by the $SMAP\_PROJECT?$ function in the design of our Metaphor Reasoning System, namely how much concrete domain inference is felicitous given the input utterance. Our current answer to this question comes from the following observation consistent with Gricean maxims of relevance and with the observed data.

> Make as many source domain automatic inferences as required to infer a **new** event about the domain of discourse. In our system, this observation translates into a rather simple rule. Continue simulating the concrete domain x-schemas until an $SMAP$ is activated at the current time step, i.e. a *new* event projection is made onto the target domain.

While the actual validity of the above claim is yet to be rigorously tested, we have found this to be a intuitively satisfying assumption to work with. In our system one role of metaphor maps is to actually **constrain** source domain inferences. In other words, new $SMAPS$ curtail propagation. It easily follows from our design that in cases where much of the source is mapped directly onto the target there is no advantage to going back to the source domain for inference, and reifying concepts in the target domain would be a good idea. However, in cases with a rich source domain and relatively sparse target domain (as is the case with embodied metaphor maps) the the ratio $\frac{SMAP}{EmbodiedConcepts} \ll 1$ and thus it makes sense to make source domain inferences. So from our hypothesis, it follows that the sparser the target domain knowledge, the more likely are long source domain inferential chains.

8.6.2 But the implementation is not connectionist

A personal disappointment was that our implemented version of the metaphor interpretation system is incompletely mapped to the connectionist level. Specifically, we have no connectionist theory of Belief net inference. The reader may be aware that for polytrees Pearl's algorithms (Pearl 1988) essentially allow us to use local message passing and a structured connectionist style of updating. However the JLO clustering algorithm (described in Chapter 7) used in this thesis requires a significant pre-processing step to convert the input DAG to a junction tree. We don't know how to make this algorithm connectionist. Stochastic simulation may offer a cleaner mapping to connectionist computation, and Radford Neal (Neal 1994) shows that Boltzman machine updating algorithms are a subset of Belief network updates with stochastic simulation. Besides, the reader may already be aware that the link between Neural network and Belief network models are a currently active area of research (evidenced by the large attendance and interest in the 1995

NIPS workshop on the topic). But exact mappings await further research.

A possible alternate implementation approach that is being explored may offer a better chance for the reduction to a connectionist level. The idea is to use an x-schema representation for the target domain as well. There is currently an implementation of the system that replaces the target domain belief net with an x-schema based inference system similar to the one described in Chapter 6. In this system, target domain theories are modeled as x-schemas as are metaphor maps. The difference between the two domain theories is that the target domain is sparse while the source domain is densely connected. The advantages are a uniform processing mechanism and translatability to a connectionist implementation (ref. Chapter 4). The big disadvantage is that we don't yet know how to compute the global impact of target domain f–struct updates (which is the beauty of the Belief update algorithms).

Chapter 9

Results of the Metaphor Reasoning System

The program was developed using a database of concepts and stories collected from various on-line sources.[1] The Metaphor Reasoning System is designed to work as a component of an incremental interpreter, accepting partial interpretations (as input f–structs) as input and returning more informative ones (as output f–structs) as output. In its current form, the Metaphor Reasoning System should be viewed as a stage in the interpretation process, after certain parts of the input (namely syntactic, surface semantic, and morphological structures) have been identified. In terms of standard pyscholinguistic theories (MacDonald, Tannenhaus, Truesman) it should be viewed as an essential part of the *integration* phase of the on-line interpreter.

Crucial to building an on-line interpreter is the requirement that the interaction with the system described here be **bi-directional**. Our database contains examples where metaphoric projections of x-schema executions are required even for early semantic disambiguation decisions, such as identifying the referent of the *subject* of a sentence. Hence we view the tight, bi-directional coupling between an on-line parser/interpreter and the metaphor reasoning system described here to be a first order design requirement. Being composed of feature-value pairs, our system's I/O was partly designed to reduce future

[1]Thanks to the UC Berkeley Law Library and Deborah Kearney for allowing me access to the amazing LEXIS/NEXIS news resource. Jane Edwards was kind enough to provide some of the Wall-Street Journal data, and some of the data on inflation and illness came from the British National Corpus and FrameNet project, courtesy Collin Baker and Prof. Fillmore.

interfacing problems to such a interpreter.

In related work not reported here, we have spent some time designing and implementing the right kind of incremental construction-grammar parser (Fillmore 1988; Goldberg 1995; Jurafsky 1996; Narayanan & Jurafsky 1997) that could integrate with the metaphor reasoning component. However, this work is still ongoing and the parser has not been integrated with the Metaphor Reasoning System. For the examples shown here, all the input f–structs shown were manually generated.

The implemented system is sensitive to both target and source domain context. Using a temporally extended Bayesian network allows us to store the state of the results of previous processing and propagate influences backward and forward in time. As we have seen in Chapter 6 and Chapter 8, the x-schema representation of the concrete domain is inherently **stateful**, so the current execution state of the x-schema influences future evolutions making the x-schema based inference and the overall system completely context sensitive (both to the previous parse and world states).

Currently our embodied domain theory has about 50 linked x-schemas, while the abstract domain theory is relatively sparse with a belief net of 12 multi-valued variables with at most 5 temporal stages. We have also encoded about 50 metaphor maps from the domains of *health* and *spatial motion*. These were developed using a database of around 30 $2-3$ line newspaper stories all of which have been successfully interpreted by the program. The database of stories used during program development is shown in Chapter 11, Section 11.1. The set of implemented source domain schemas is described in Chapter 11, Section 11.2. The belief net priors are described in Chapter 11, Section 11.3. Full I/O behavior of the implemented system for some of the stories in Chapter 12.

When the system could cover the initial example set in Chapter 11, Section 11.1, we tested the system on a previously unseen set of stories from the various on-line corpora in the domain of international economics. Section 9.2 discusses the results obtained when the system was presented with these never before seen stories. Chapter 13, the full I/O behavior of the program on specific new stories.

We were somewhat surprised by the number, variety, and subtlety of roles played by sensory-motor features in processing narratives about abstract plans and actions. Throughout the description to follow, while presenting results of the implemented model on specific examples, we hope to give the reader an idea of the scope and range of information that relies on the dynamic, compiled, and highly-responsive nature of x-schemas to provide crucial inferences. Our results show the capability of our model to make these inferences in real-time. While the results pertain to how information from embodied terms can be useful in interpreting descriptions of events in the domain of international economics, we hope that the generalization to abstract plans, goals, and intentions in any domain is obvious to the reader.

Results are broken into two main sections. The first section deals with results of running our program on the development set, which were the set of stories that drove development of the program. Once we were satisfied that our program to handle the type of event descriptions present in development set, we decided to test the program on the test set. Results on this test set can be found in the next Section.

9.1 Results on the Development Set

As explained in Chapter 7, the input and output of the implemented program are f–structs. The result of running the program on the input f–struct generates additional bindings resulting from x-schema inferences and belief updates on the target domain Bayesian network. The use of embodied terms and projections from embodied metaphors such as *OMAPS* (object maps), *PMAPS* (parameter maps), and *SMAPS* (schema maps) appear to have a variety of uses in adding valuable information about the domain of discourse that is readily available from the use of the embodied term. These aspects and our program's performance is elaborated below.

A note about the tagging convention. Wherever expressions are from the development database in Appendix A, a reference is made by indicating the story number in brackets. If the I/O behavior for the specific example is also shown, a reference is made to the appropriate Table or Graph as well. Since the stories are of at most three lines, we expect the reader to find their usage in the actual story without further tagging.

9.1.1 X-schema Parameters

Distances, *speeds* , *force-values*, *sizes* and *energy-levels* are obviously important perceptual and motor control parameters, but with $PMAP$ projections, each of these perceptual and motor features can also become important descriptive features of events in abstract domains. For instance, in Chapter 7, we already saw the use of the $PMAP$ which transfered distance from the source of travel ($Dist(Src)$) onto the domain of plans and policies as indicating the degree of completion of a plan (Doc). In our example, we used a seven point distance scale and translated it to a probability distribution over a Bayes net variable with three values. In general, these projections often *scalarize* target feature-values with decreased target domain resolution.

Other examples of motion parameters being projected onto the abstract domain of policies include the implemented $PMAPS$ which project *speeds* and *rates* onto *progress made* in plan implementation. For instance, proceeding at the *desired speed* setting in the domain of spatial motion is projected as making progress *on-schedule* in the target domain. Similarly *high speed* of motion is projected as making *good progress* (better than expected) in plan implementation, and *moving slowly* is projected as make *little progress* (less than expected) in plan implementation. Moving *backward* is similarly projected as *undoing* progress, and different *paths* to a *destination* are projected as different *plans* with the same *goal* .

In our examples, we were able to use $PMAP$ s to map the following kinds of expressions and modifiers that corresponded to the x-schema parameters mentioned above. Size parameters were routinely found in expressions like *giant steps* (Story 21)(ref. Table 9.1), *large step*, *small steps* (Story 21), *great leap forward* (including the Chinese Economic Reform), and the interesting *inches and feet not miles* comment (Story 5). Rate of motion was also a frequently used feature with expressions like *slow progress* (Story 5), *slowed down* (Story 4), *sprint* (Story 7), *jog* (Story 7), and *long, painful slide* (story 6). Other high-frequency lexicalized items coding for rate and manner included *crawl*, *leap*, *trod*, *plod*, *slog* (Story 21), and *lurch* (story 10), *slither* (story 7), etc. Examples of distance related parameters include expressions like *almost there*, *long way to go*, *halfway there* (Story 21), and *a little further* (which were handled by the $PMAPS$ $dist(dest) \Rightarrow AmountLeft$, and $dist(src) \Rightarrow DOC$). Force magnitudes and durations are also routinely projected as in *grip* (Story 2),*tear down* (Story 4), *hold back* (Story 3). Specific counter-forces may cause injury or harm to the traveler including *robbing* (Story 4). Force amounts are projected through $PMAPS$ linking them to the amount of influence or intensity, determination or duration of a specific plan or action. The area of force-dynamics and its projection onto abstract domains is an ongoing area of research and current efforts and extensions are detailed in the next section.

As far as we know ours is the only computational implementation capable of modeling the transfer of such sensory-motor parameters from embodied to abstract domains. Table 9.1 shows the program I/O for the frequently occurring phrase, *Giant Steps* (Story 21).

Example 15 Giant Steps

Table 9.1: Processing the Input *Giant Steps*. The values are shown for the input and relevant target features (thresholded at $P \geq 0.6$). This example shows the role of $PMAPS$ in interpreting event descriptions.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$
INPUT FEATURES			
Domain=Economic Policy	t	t	t
Event= step		t	
step.size=large		t	
ACTIVE MAPS			
$M_s(step) \Rightarrow implement(policy))$		A	A
$M_p(step.size = large \Rightarrow progress = good)$		A	A
TARGET(OUTPUT) FEATURES			
Action = implement(policy)	**t(.6)**	**t**	**t(.8)**
Status = ongoing	**t(.8)**	**t**	**t(.8)**
Progress = good	**t(.6)**	**t**	**t(.8)**

■

In the **Giant Steps** example, the input appears at time $t = 1$, and the partial interpretation (assumed) results in setting the feature values $Domain = EconomicPolicy$, $Event = step$ and the step parameter $step(size) = large$. Note that all these values are shown for time $t = 1$.

The input **marks** the STEP x-schema (and the corresponding WALK x-schema) to be *ongoing* and the step-size parameter to be *large* ($step(size = large)$). The WALK x-schema then executes in a manner described in Chapters 6 with CONTROLLER status *ongoing*. This activates the $SMAP$ described in Chapter 8 $Move(forw) \Rightarrow Impl(pol)$ with $Status = ongoing$. The corresponding $SMAP$ projects an *ongoing* step to *set_evidence* for an *ongoing* policy implementation. The *size* parameter also activates the $PMAP$ $M_p(step.size = large \Rightarrow progress = good)$ (better than expected) which is thus able to assert that *good* progress is being made in the ongoing policy implementation. Thus the f–struct in Table 9.1 shows these active metaphor maps described above.

The projections made by the metaphor maps onto the domain of economic policies is thus able to *set_evidence* that at time $t = 1$, an ongoing policy is making good progress. During the subsequent $BELIEF\_UPDATE$, the *persistence* links propagate this information (absent other information to the contrary) backwards and forwards in the Bayesian network setting the context at the last time step and the future time step to be about economic policies (hence the values t for this variable at times $t = 0$ and $t = 2$ in Table 9.1). Persistence links carry the inference forward using the conditional probability link $P((X, t+1)|(X, t))$, and backward using Bayes rule. Recall that applying Bayes rule allows to compute the posterior value of the variable of interest at time $t - 1$, given its value at time t (as shown in Chapter 7). In all cases, we set the prior that progress was good at time $t = 0$ to be .4 .($P(Progress = good, 0) = 0.4$), and the conditional link $P((Progress = good, 1)|(Progress = good, 0)) = 0.8$. With these values and Bayes rule, we get the values of the target variables (shown in **boldface** for $t = 0$, and $t = 2$) in Table 9.1. As a result of the operations described above, the system is able to infer that the input *giant steps*, when uttered in the context of economic policies indicates better than expected progress in the implementation of an ongoing policy.

9.1.2 Aspectual Inferences

Chapters 5 and 6 outlined the compositional model of x-schemas which allowed us to make inferences about *aspect* using the **controller** or **aspect graph**. It turns out, that aspect is an *inherent property* of events and thus is *invariantly projected* across domains. We also saw it ubiquitously used in describing events, both concrete and abstract, validating our explicit representation and supporting one of our claims about the CONTROLLER abstraction.

The following examples from our database were successfully processed by the Metaphor Reasoning Program and serve to substantiate the claims above. We already saw in detail how the concept *is stumbling* is projected onto the domain of international economic policies as an *interruption* to an *ongoing* policy. Other easily modeled cases include the focus on the consequent state that is signaled by the use of the *perfect* aspect (Chapter 5) such as *have robbed* (story 4), *has been lurching forward* (story 10), *has walked* (story 11),*has sidestepped* (story 14) *has been now taken ill* (story 11). We could also nicely model several other high frequency aspectual expressions such as *start to pullout* (story 12), *on the verge of* (story 10), *still trying to climb out* (story 15), and metaphoric expressions of aspect such as *set out* (story 2), (ref. to Appendix B for the full I/O behavior) *remain stuck* (story 12), *on-track* (ref. Table 9.2) and the interesting phrase *back-on-track* (Story 22) (Table 9.3). Another interesting case was the phrase *stumble to the brink* (Story 17), where the CONTROLLER graph distinction between the state of a schema being *ready* to execute versus actually *start* to execute was crucial to make the right inferences. In summary, almost every event description had a aspectual component, and so we believe attention to the details of the semantics of Verbal Aspect is essential even to interpret the simplest of event phrases and distinctions. We believe our model is unique in integrating the semantics of aspect with metaphoric interpretation.

While we have seen many examples of how our model interprets aspectual expressions, consider the interesting metaphoric case of *on-track* (from *Liberalization Policy on-track*). Table 9.2 shows the results of the metaphor reasoning system on this input. Here, the event $on - track$ is coded as a traveler moving forward (toward destination) at the specified (desired) rate (rate = $setting$). A familiar $SMAP$ projects *moving* onto the domain of Economic Policies as *implementing* the policy, with the status as *ongoing* (since motion is ongoing in the source). More interestingly note the $PMAP$ $M_p(rate = setting \Rightarrow Progress = on - schedule)$, which together allows for the interpretation of "on-track" as an $ongoing$ policy being implemented $on - schedule$.

For an even more striking argument for the x-schema based architecture, consider the semantics of *Policy back on track.* Table 9.3 shows the result of the system processing this input. The interesting fact is that here one has to assert two previous states i.e. if the input was at time t , the traveler was not moving forward at the desired rate (not on-track) at the previous relevant stage ($t - 1$), but was moving forward at the desired rate (on-track) at the stage previous to that one ($t - 2$). Projecting this onto the target, using the maps described earlier, the system is able to infer that if the input is at time t , then at a time immediately prior ($t - 1$), the $policy$ was not $on - schedule$, but at a time prior to this ($t - 2$) the policy $status$ was $ongoing$ and the $policy$ was $on - schedule$. We

Table 9.2: Processing the Example *on track*. Relevant *target* variables with Posterior Probabilities above Threshold (.6) are shown.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$
INPUT FEATURES			
Domain=Economic Policy	t	**t**	t
Event= on-track		t	
ACTIVE MAPS			
$M_s(move(forw) \Rightarrow Implement - policy)$		A	A
$M_p(rate = setting \Rightarrow Progress = on - schedule)$		A	A
TARGET(OUTPUT) FEATURES			
Action = Implement(Policy)	**t(.8)**	**t**	**t(.8)**
Policy Status = Ongoing	**t(.7)**	**t**	**t(.8)**
Outcome =success	**t(.7)**	**t**	**t(.9)**
Progress = on-schedule	**t(.7)**	**t**	**t(.9)**

believe that the **dynamic semantics** provided by x-schemas was absolutely crucial to be able to successfully model this example.[2] As a final example of how Aspectual inferences are absolutely crucial to interpret even such simple narratives, consider the phrase *on the verge of falling back into recession* (Story 10), which nicely exercised the temporal aspect of *back* discussed above, as well the **controller** distinction between *enabled*, *ready*, and *start* required for *on the verge of.*

9.1.3 Goals, Resources

It is well known that narratives are generally about *goals* (their accomplishment, abandonment, etc.) and *resources* (their presence, absence, levels, etc.) (Wilensky 1983; Schank & Abelson 1977; Carbonell 1982). However, somewhat serendipitously (at least to this author), we found that embodied terms may in fact be *compactly coding* for these features as well. Note that this assertion is quite different from (Carbonell 1982) who hypothesized that entire strategies and plans (proof trees) ware invariantly transferred. In our theory, it is the *key-event* such as the thwarting of a goal, or the absence or production of a needed resource, etc. that is asserted using embodied terms. Complex planning, new-

[2]Notice that the standard Markov assumption inherent in Hidden Markov Models or two-slice Dynamic Belief Nets would have problems with this example because the current state is allowed direct influence to only one state in the past or future (time being symmetric in stationary models). That is one reason, we played it safe by saving all time slices temporally extended nets in the target domain rather than the more efficient two-slice Dynamic Belief networks.

Table 9.3: Results of processing *Back On Track*. Note that there are at least three *time* slices needed to model this utterance.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$	$V(t_2)$
INPUT FEATURES				
Domain=Economic Policy	t	t	**t**	t
Event= back(on-track)			t	
ACTIVE MAPS				
$M_s(move(forw) \Rightarrow Implement(policy))$			A	A
$M_p(rate = setting \Rightarrow Progress = on - schedule)$	A	A	A	A
TARGET(OUTPUT) FEATURES				
Action= implement(policy)	**t**	**f(.9)**	**t**	**t(.8)**
Policy Status = ongoing	**t**	**f(.9)**	**t**	**t(.8)**
Outcome =success	**t(.8)**	**f(.9)**	**t**	**t(.9)**
Progress = on-schedule	**t(.8)**	**f(.9)**	**t**	**t(.9)**

strategy development and problem solving may well be common across different domains, but they are not reflex, real-time processes, whereas we found that statements about changing goals, resources and intentions often are available in real-time, especially when they are asserted using embodied terms.

In Chapter 6, we saw how out x-schema executions can be enabled, disabled, terminated, or otherwise modified by the assertions and retraction of goals and energy levels. Moreover in Chapter 3, we showed how sensory-motor representations must inherently be highly responsive to changing goal and resource needs for survival.

With embodied metaphor maps, narratives are able to exploit this feature of sensory-motor representations to assert changing goals and resources. Consider the stumble example (from Story 1) painfully detailed in Chapter 8. What stumble codes for is the presence of a difficulty interrupting a plan, leading perhaps to the thwarting of a policy goal. Similarly while we saw items coding for rate of progress or regress in the plan, embodied verbs and event descriptions also compactly code the *energy* levels as a resource For instance *slog* (BNC story 5), *anemic* (Story 10) (ref. Table 9.6), *sluggish* (Story 19) or *stagger to their feet*, *battered and bloodied*(Story 16) crucially communicates the inherent lack of *stimulating conditions*, *resources*, *level of interest*, *attention* or *motivation*. Similarly *tearing barriers* (Story 4) or *lightening burdens* (Story 5) are able to assert conditions where an impediment to goal achievement has now been removed. Compare this to the expression *go around* or *sidestep* where the difficulty of potential failure is still present. Similarly *slippery slopes*, *slipperiest stones*(Story 19), *slide into recessions* (Story 21), get projected through $SMAPS$ as the possible thwarting of goals due to unanticipated circumstances. *Falling* is crucially important in this regard. In all the cases where a country was described as *falling* into recession, we never saw a case in which the country's administration was directly blamed as being able to control the downturn, a fact directly project-able from the fact that falling is not controllable (an obvious and easy inference about fall). This is shown in the example below.

Example 16 *Brazil fell into recession.* (ref. # Fin. Times, 1992) ■

The input f–struct for this story is

```
(Input ((Context EconomicState)(Type recession)(Agent Brazil)(Event fall-into)
(Aspect Perfective)(Ut-type description)))
```

For the output (thresholded at $P \geq 0.6$) refer to Figure 9.1. Note especially the projection from the **uncontrollable** and **unintentional** nature of *falling* onto the domain of Economic states as the recession being cased by external factors beyond direct control or influence of the specific policy being pursued ($\neg control(loc) \Rightarrow \neg control(Ec.State)$). Of course, this kind of information is often useful to assign responsibility or blame to various agents in the described scenario. For instance, with *fall into recession*, the speaker is inherently indicating that he does not hold any specific administration or policy to blame (probably a normal business cycle recession or some external causative factor). Contrast this to the choice of *walk into recession*, where the speaker is quite likely assigning some responsibility to a failed policy. In fact, quite often intentional aspects of embodied terms get transferred onto more abstract domains as causative factors. While the explicit casual connection is not modeled, even our prototype is able to model some issues of controllability as shown in Figure 9.1. [3]

Thus, while we certainly do not claim that our model is able to account for the rich range and extent to which narratives are about goals changes and dynamic resource levels, we note that in many cases, use of embodied terms and relying on metaphoric projection

[3]In cases where the results are displayed as graphs, the x-axis represents the time step of interest (0 is the prior, 1 - 3 are the the result of processing the input at that time step, and 4 is the predicted future state of variable in question). In the case of a graph the shading of the relevant cell indicates the degree to which the variable is believed to be $true$, the darker the shading the higher the $degree\_of\_belief$.

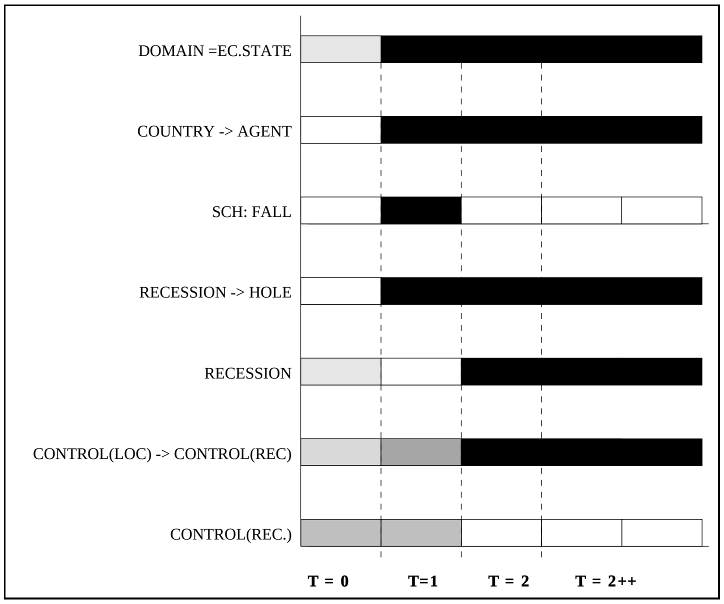


Figure 9.1: Simple inferences are transfered from embodied to abstract domains for the input "Brazil fell into recession". Of specific interest is the spatial inference that falling into a hole results in being there; this results in setting evidence for the target domain node that Brazil is in recession at the end of the period specified by the input. Also interesting is the inference that the recession was probably unanticipated and uncontrollable, an inference from falling. No such inference is intended or available from processing *Germany has walked into recession (Story 11).*

of the quick, compiled simulative reasoning products of x-schema execution may be the best way to specify these complex and abstract notions of goals, resources, and controlling and causative factors. In any case, we fail to see how a semantics that is not inherently **dynamic** could account for this range of phenomena.

9.1.4 Control/Monitoring Inferences

As was seen in the case of the *fall* story, the uncontrollable nature of a fall gets projected onto the target. We found stories in the abstract domain to often be about the complex notion of controllability, monitoring problems, and policy adjustments. Again, monitoring, changing directions, rates, etc. are obviously common in sensory-motor activity, and so again using these features and appropriate projections allows the speaker to communicate monitoring and control problems in abstract plans. For instance consider the expression *taking a cautious step in the right direction* (Story 13) or the beautiful example (Story 19) *Economic reform is like crossing a river by feeling for the stones*, including the concept of *inflation is the slipperiest stone.* Of course, we have already seen how the notion of a slippery ground can cause our x-schema model to assert the possibility of tripping and falling which with the $SMAP\ \ Fall \Rightarrow Fail$ (described in Chapter 8) can readily generate the inference of possible *reform failure.* This is done in a manner completely analogous to the example described in Chapter 6 including the inference that frequent monitoring (remember $test\_footing$ in the WALK x-schema) would have detected the *inflationary* possibility and led to *tinkering* with the policy variables. This frequent monitoring and subsequent adjustment is also often communicated using an embodied term such as *reorient.* Table 9.4 shows the performance of the system on the input *reorient policy* (story 21).

Table 9.4 shows how common embodied concepts can easily encode complex monitoring and control conditions in the domain of spatial motion which can be projected using $PMAPS$ and $SMAPS$ onto the domain of abstract plans and actions. Here *reorient* encodes the monitoring of distance, rate and direction to the destination, and subsequent **adjustments** to correct any discrepancies. Table 9.4 shows the results of the metaphor reasoning system on this input. The metaphor maps shown in the example $M_s(check(dist\_to\_goal) \Rightarrow monitor(DOC)$, $M_s(check(rate) \Rightarrow monitor(progress)$, $\wedge$ $M_s(adjust(motion) \Rightarrow adjust(policy))$ allow the interpreter to conclude that the the policy was monitored and examined for performance and accordingly adjusted. The new policy

Table 9.4: The Metaphor Reasoning System's response to the input *reorient policy*. Embodied terms often convey information about monitoring and control of complex plans.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$
INPUT FEATURES			
Domain=Economic Policy	t	t	t
Event=reorient		t	
ACTIVE MAPS			
$M_s(check(dist\_to\_goal) \Rightarrow monitor(DOC)$		A	
$M_s(check(rate) \Rightarrow monitor(progress)$		A	
$M_s(adjust(motion) \Rightarrow adjust(policy)$		A	
$M_p(rate = slow) \Rightarrow (progress = low)$		A	
TARGET (OUTPUT) FEATURES			
Monitor(DOC)	**t(.8)**	**t**	**t(.8)**
Monitor(progress)	**t(.8)**	**t**	**t(.8)**
Adjust(policy)		**t**	
Status(new policy) = READY		**t**	**t(.3)**
Status(new policy) = START		**f**	**t(.7)**

is ready (a result of reorienting) and will most likely be implemented soon (shown as a predicted *start* at the next time step (the next temporal slice of the target network)).

9.1.5 Mapping Attitudes and Affects

We found attitudes to be essential ways of encoding anticipatory conditions, motivation and determination of agents involved. We have implemented some of this in the prototype system. For instance *bold* (Story 3) encodes determined implementation as well as anticipated difficulty ahead. In the current model this is directly encoded as the semantics of *bold* in the context of the embodied domain (anticipating some counter-forces (future time step), and showing a high degree of determination. As usual, counterforce at the next time step gets translated to anticipated difficulty at the $t+1$ temporal slice. Determination gets translated as a reduced *prior* chance of policy change. The following output shows the effect of the use of *bold* in story 3 (the second line starting with *boldly set out*). The point to note here is that the embodied term *bold* codes for possible future obstacles, and the readiness to deal with them. This projects onto the target as the possibility of future difficulty and the map $M_{s_5} : deal - with - obstacle \Rightarrow remove - diff$ asserts the status of removing difficulty to be $READY$ $(Remove - diff(status) = READY)$, indicating the readiness to remove such a future difficulty. The complete I/O for this story can be found in Appendix B.[4]

Table 9.5: System's response to *Story*3 (ref. Appendix B) showing showing the impact of **bold** in the phrase *boldly set out*. For full I/O behavior on this story, please see .

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$	$V(t_{3+})$
$M_p : obstacle \Rightarrow difficulty$			A	A	
$M_{s_5} : deal - with - obstacle \Rightarrow remove - diff$		A	A	A	
Remove-diff(status) = READY		**t**	**t**	**t**	**t**
Goal = FT $\wedge$ Dereg			**t**	**t**	**t**
Difficulty				**t**	**t(.7)**

9.1.6 Communicative Intent and Metaphor

One of the important aspects of communication involves specifying evaluative judgments of situations to communicate speaker intentions and attitudes. We hypothesize that the cross-linguistic prevalence of the use of embodied notions of force and motion to communicate aspects of situations and events is linked to the ease with which evaluative aspects can be communicated in experiential terms. Consider the difference in the two sentences below.

Example 17

1. Government loosened strangle-hold on business.

2. Government deregulated business.

■

Both sentences communicate the same fact in the domain of economics, namely the the situation corresponding to business deregulation. Our implemented system is able to conclude that only in the first case does the speaker intend to communicate the threatening nature of Government intervention, its highly systemic and deleterious consequences, as well as his evaluation of the inherent cruelty of high regulation, something very aptly encoded

[4]Another case (the last example with reorient was yet another) where linguistic devices are able to exploit the CONTROLLER distinctions between *ready* and *start* , a fine-grained control distinction that is useful for motor control but proving quite indispensable for language.

by the embodied concept *strangle*. We know of no other implemented model of metaphor understanding that can reason about these phenomena. In our model, the $speaker - eval$ variable is highly negative($neg + +$) for the situation of strangling prior to the loosening ($t_i < t$). Of course the $Ut - type$ variable also changes correspondingly from $description$ to $opinion$ as explained in Chapter 8. Chapter 12 has the full I/O behavior for this story.

9.1.7 Novel expressions

As (Lakoff 1987; Gibbs 1994) and other researchers point out, a variety of novel expressions in ordinary discourse as well as in poetry make use of highly conventionalized mappings such as the ones described here. In fact, the implemented system is able to interpret novel expressions which it has never seen in the context of abstract actions and plans.

Table 9.6: I/O for the *anemic recovery* example. Input is shown in the first part ($t = 1$), and output variables (shown in **bold**) correspond show the output values for the relevant $target$ variables.

FEATURE	$V(t_0)$	$V(t_1)$	$V(t_{1+})$
INPUT FEATURES			
Domain=Economic State	t	t	t
Event= Recover		t	
Recovery.type= anemic		t	
ACTIVE MAPS			
$M_s(Recover \Rightarrow Change\_state(neg, pos))$		A	A
$M_p(energy = low \Rightarrow \neg stable)$		A	A
$M_p(energy = low \Rightarrow growth\_rate = low)$		A	A
TARGET(OUTPUT) FEATURES			
Ec.state = neg	**t(.6)**	**t(.1)**	**t(.2)**
Ec.state = pos	**t(.1)**	**t(.5)**	**t(.3)**
Ec.state = pos++	**f(1)**	**t(.1)**	**t(.1)**
Ec.state = neg++	**t(.1)**	**t(.1)**	**t(.1)**
Ec.state = flat	**t(.1)**	**t(.2)**	**t(.3)**
$Growth\_rate$ = **low+**	**t(.1)**	**t(.9)**	**t(.9)**
$Growth\_rate$ = **neg**	**t(.9)**	**t(.1)**	**t(.1)**
Stable(Economy)	**t(.2)**	**t(.3)**	**t(.2)**

As an example of a novel expression successfully processed by our system, consider the expression *anemic recovery* (Story 10). The result of processing this example is shown in Table 9.6. The concrete domain meaning of *anemic* in this example was set to be a low energy recovery ($energy = low$)[5] The $SMAP$ $M_s(Recover \Rightarrow Change\_state(neg, pos))$ maps the process of recovery onto the domain of economics as the change of state from a negative to a positive value. Similarly the $PMAP$s $M_p(energy = low \Rightarrow \neg stable)$ and $M_p(energy = low \Rightarrow growth\_rate = low)$ map the non-energetic nature of the recovery to assert an unstable economy with a low $growth\_rate$. With these maps and the input shown, we get the target domain f–struct shown in the Table 9.6. In the example, we note that the instability of the economy keeps the predicted future values of the Economic state variable unchanged from the current one, showing unpredictability of the current situation.[6]

The system was also tested with the following novel example.

Example 18 Government at crossroads. ■

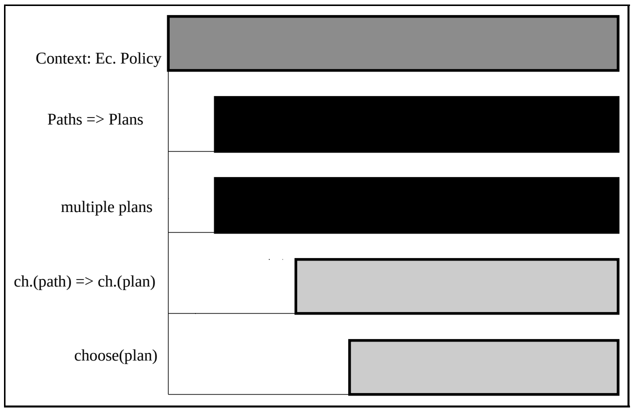


Figure 9.2: Novel expressions may use conventional mappings allowing the interpreting agent to correctly infer the intended meaning in contexts where it has never been encountered before. Here the concrete domain meaning of crossroads (multiple paths) maps to the abstract domain of economic policy through the event structure metaphor.

The challenge is to be able to interpret *crossroads* in the context of economic plans and policies. Figure 9.2 shows how knowing the concrete domain meaning of crossroads

[5]The actual grounded map from low RBC counts to low energy state was not modeled, under the assumption that the low energy (or *weak*) meaning of anemic is sufficiently grounded.

[6]Variables below threshold were included to show that default persistence links were overridden by the assertion that the economy is unstable, so effectively the system makes no prediction, resulting in a redistribution of the probability mass to be further down the scale (here we just added .1 the two closest items lower on the scale).Something more clever can be done in the real situation where there is more explicit target knowledge but that is outside the scope of this system.

(multiple possible paths) and the event structure metaphor mapping conceptual features from the domain of motions to abstract actions allows for a reasonable interpretation.

The relevant results of processing this example is shown in Figure 9.2. Crucially, given the stored maps ($Paths \Rightarrow Plans$ and the $Choose(path) \Rightarrow Choose(Plan)$) concrete domain meaning of crossroads (multiple paths to choose from) the system is able to conclude:

1. The sentence is uttered in the context of an ongoing plan.
2. There is a choice of possible next actions, one of which will be chosen by the agent.

Thus the interpreting agent correctly infers the intended meaning of a source domain concept in contexts where it has never been encountered before. Here the concrete domain meaning of crossroads (multiple paths) maps to the abstract domain of economic policy through the event structure metaphor. Note that the source domain feature remain activated until a metaphoric inference can be made. The target markings remain persistently active.

Other examples of novel expressions correctly interpreted by our program include *roadblocks* (barrier to progress was correctly interpreted as difficulty through $Obstacles \Rightarrow Difficulty$) (story 3), *anemic recovery* (Story 10) (ref. to Table 9.6), *lurching forward* (story 10), *long, painful slide* (story 7), and *treading on toes* (story 7), and the beautiful *stumble over rocky relationship* (Story 17).

9.1.8 Inferring Target Roles and Objects

Example 19 European Economic Giant falls sick. (from Story 11). ■

This is an interesting case in that the target context alone is insufficient to disambiguate the country in question. The lexical item $Giant$ in fact cannot activate any node in the target Bayesian network. However the concept maps directly to the concrete domain as a $Person$ with $size = large++$. Given this and the context (we are talking about an Economy), the $OMAP$ $EconomiesAREPersons$ gets activated [7] (in this case activation is from the $source$, hence the reason for $OMAPS$ to be bi-directional explained

[7] of course this is through the metonymy Countries STAND FOR Economies, which is assumed since our model only deals with economies or administration's policies, but in a more general system we would need many such metonymic maps between concepts in the target domain as well.

in Chapter 7), as does the corresponding $PMAP$ ($Person.size \Rightarrow Economy.GDP$), thus asserting evidence for a high GDP European Economy as the referent of the *subject* of the sentence. From the $BELIEF\_UPDATE$ operation of the target Bayesian network, the Actor node now gets instantiated to $Germany$, thus inferring the referent of the sentence subject.The event part is then fairly trivial, and the inference of a negative German economic state is made. [8] The key issue here is of course that inference about the referent of the subject is unavailable until *metaphoric reasoning* suggesting that metaphors (especially embodied ones) are indispensable even for simple semantic parsing. The output for this sentence is shown in Figure 9.3.

9.1.9 Multiple Source Domains

Multiple source domains pose no problem for the system, as long as they are interpretable and coherent in the *target*.

Example 20 Stocks down. Healthy again. ■

The serial use of metaphors from the domain of *vertical motions* and the domain of *health* is coherent in the target domain as shown in Figure 9.4. We assume the presence of the conventionalized mappings between the *verticality* and *quantity* domains (these include *More IS Up*, *Less IS Down*, *Healthy is More Unhealthy IS Less.*

Processing the first sentence leads to activation of the *Less IS Down* metaphor, propagating over to the target (since the context is set by the input **Stocks**) to mark the **Down** feature in the domain of stock prices. The next input activates the *Healthy IS more* metaphor, and since the context is still stock prices, the target feature more becomes marked, **disabling** the earlier asserted feature **less**.

Other examples of our system being able to process serially mixed metaphors from multiple source domains includes *sidestepped* the *crippling debt crises* (Story 14), *lurching forward* and *anemic recovery* (Story 10) and *flourishing* and *taken ill* followed by *walked into recession* (Story 11).

[8]Here we had to add additional knowledge about relative $GDP's$ of European countries. The $PMAP$ that GDP parameter of a country maps invariantly to the size of the person does the rest.

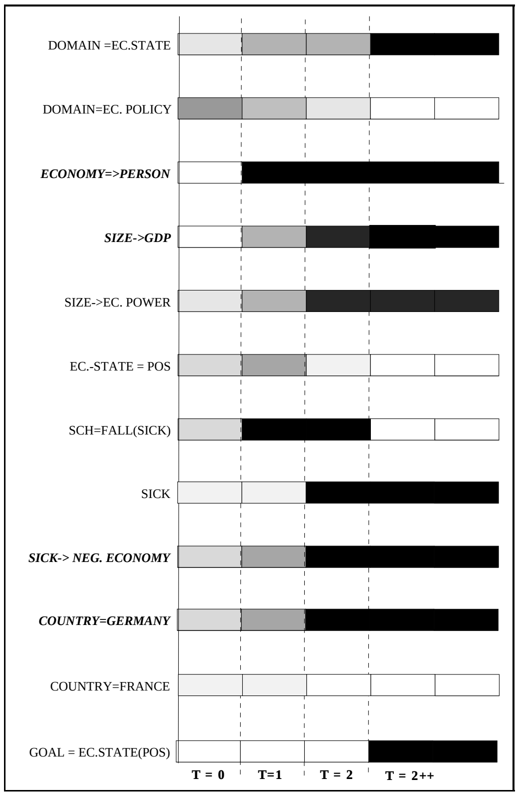


Figure 9.3: Processing the input European Giant falls sick. Note that the *OMAP Economic Actors* ⇒ *People* coupled with the *PMAP* that projects size onto *GDP* in the Economic context, results in asserting evidence in the target that identifies Germany as the country with the largest *GDP* and thus as the referent of the *subject* of the input sentence. Note also that France has some posterior probability of being the referent as well.

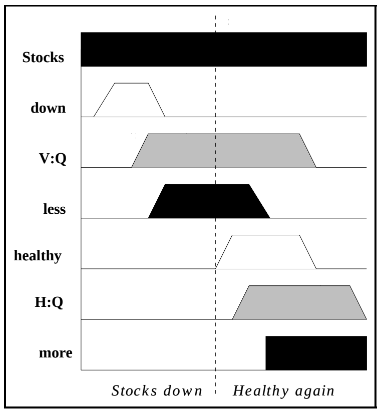


Figure 9.4: Multiple source domains can be serially used as long as they are coherent in the target domain. Here the first input *Stocks down* activates the *Less IS Down* metaphor, while the second input *Healthy again* activates the *More IS Healthy* metaphor.

9.1.10 Effect of Background Knowledge

Consider the example:

Example 21 *World Bank prescribed Structural Adjustment Program (SAP) bleeding Indian Economy.* (ref. # Report of Int. Womens Conf. Beijing, 1995) (Story 2) ■

Here the input f–struct was[9]

```
(Input ((Context EconomicPolicy)(Policy SAP)(Policy-Maker WB)
(Policy-Implementer IG)(Policy-Applied-to IEC)
(Event Cause_to_bleed(causer = WB, cause = SAP, bleeder = IEC))
(Aspect Progressive)(Ut-type description)))
```

Here is a case where we looked at the effect of background knowledge on inferences made.

In one case, we set the $Sp - eval(WB)$ variable to be *neutral* with respect to World Bank (signifying no *prior* knowledge about the speaker's evaluation) and this triggers the *OMAPS* $Policy\_Maker \Rightarrow Doctor$, $Policy\_Imp \Rightarrow Patient$, $Policy \Rightarrow$

[9]Note the new policy-type variable *SAP* (was added to the domain of the Policy variable discussed in Chapter 7) as well as the *IEC* (Indian Economy) variable as the the segment impacted by the policy. The target domain knowledge is also assumed to be sufficient to determine that the policy maker is *IG* (Indian Government). We also assume that the *transitive* sense of bleed has been parsed as the event *Cause_to_bleed* by the parser.

$Prescription$, and the $SMAP$ $Cure \Rightarrow Policy$ making the World Bank appear as a *doctor* and the *Indian Economy* as a *patient*, and the SAP program as a $treatment$. The result is one of *mistaken therapy* shown in Figure 9.5. A well-intentioned $cure$ is seen as $failing$ (hence the $outcome$ variable is $failure$). The expectation is that the good doctor (World Bank) would remedy the situation upon receiving this information and discontinue the policy (this is not modeled).

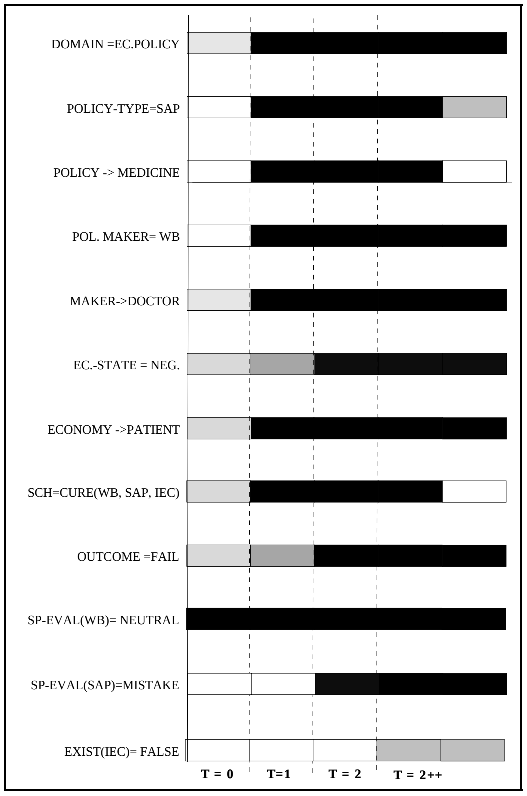


Figure 9.5: The results of interpreting the input corresponding to the sentence "World Bank SAP bleeding Indian Economy". Here, the prior belief of the interpreter activates the **cure** x-schema. Here, the target domain inferences is one of *mistaken therapy*, where the cure doesn't work. Also note, that in the domain of health and well being, the bleeding activated slow dying. Note that the future continuation of the policy is uncertain as is the fate of the Indian economy.

In one case, we set the $Sp - eval(WB)$ variable to be $neg++$ with respect to World Bank (WB) (representing $prior$ knowledge that the speaker was very critical toward World Bank in general). This triggers the $OMAPS$ $Policy\_Maker \Rightarrow Harmer$, $Policy\_Imp \Rightarrow Patient$, $Policy\_ \Rightarrow Prescription$, and the $SMAP$ $Bad\_Medicine \Rightarrow$

Policy . This favors viewing World Bank as the *harmful* doctor rather than as a *competent* doctor. This triggers the $bad-medicine$ x-schema as opposed to the *cure* x-schema. So in this case, World Bank is the harmful doctor, Indian Economy is the bleeding *patient* , and the *SAP* prescription is a harsh and *harmful* prescription of the World Bank. The results of this setting can be found in Figure 9.6. Compare this to the earlier case of evaluating the policy as a genuine mistake and benign-ness of the *WorldBank* .

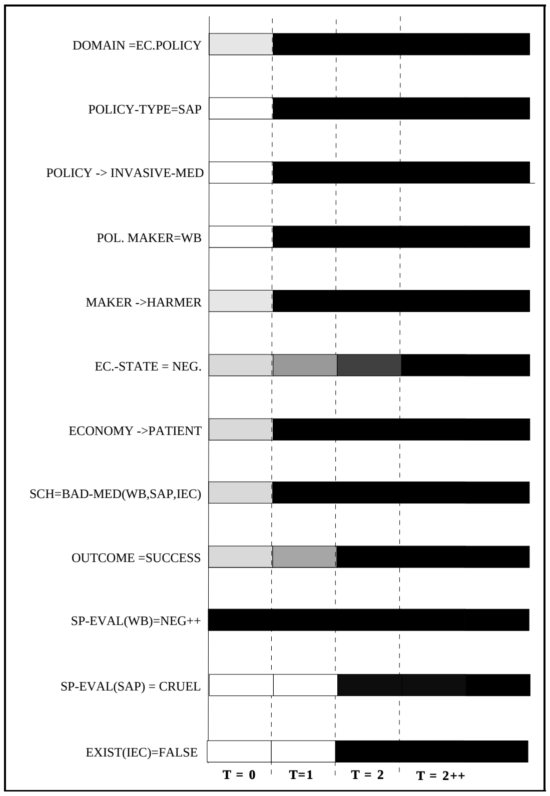


Figure 9.6: Processing the input “World Bank SAP bleeding Indian Economy” with a different prior. Here, the prior belief of the interpreter activates the **bad-medicine** schema. Here, the target domain inferences is one of harmful neglect, with **bad medicine**. Also note, that in the domain of health and well being, the bleeding *enabled* dying, which translates to the increased possibility *extremely deleterious consequences* (non-existence) to the Indian Economy. Also note that the Policy Maker is viewed as a cruel blood-letting *harmfuldoctor* and the policy is viewed as *invasive* .

In the third case, we set the $Sp-eval(WB)$ variable to be $pos++$ with respect to World Bank (signifying *prior* knowledge that the speaker was positive about World Bank intervention) and this does makes the World Bank appear as a *doctor* and the *Indian*

Economy as a *patient*, and the *SAP* program as a *treatment* . The result is one where *partial blood letting* is part of the curing process leading to a *partial cure* shown in Figure 9.7. Notice the absence of any malicious intent being ascribed to the World Bank in this case, and the evaluation of the policy as *required* and also the existence of the Indian Economy as never threatened by the policy.

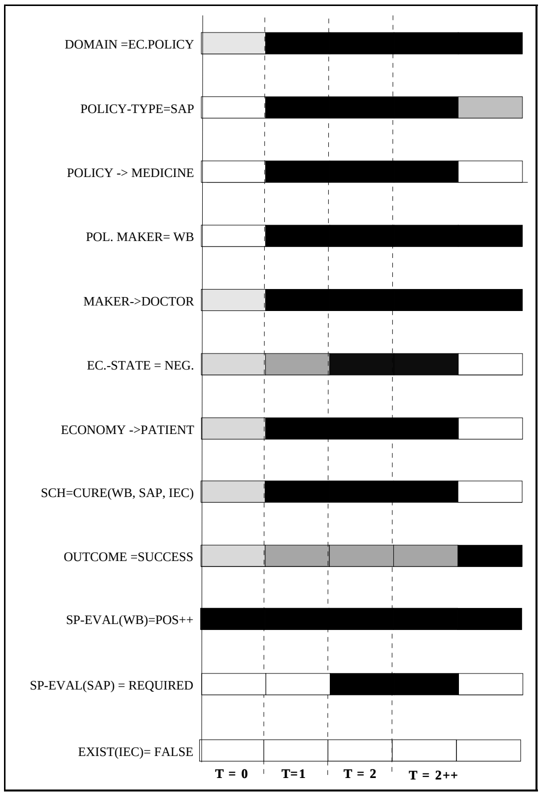


Figure 9.7: The results of interpreting the input corresponding to the sentence "World Bank SAP bleeding Indian Economy". Here, the prior belief of the interpreter activates the **cure** x-schema. Here, the target domain inferences is one of *partial cure*, where the cure **does work** and is **required**. Note that in this case, the speaker expects the outcome of the curing to be successful, the cure to be ongoing, and no indication of deleterious consequences for the Indian Economy, unlike the other cases.

One crucial difference in the three cases is in the first case, the the *outcome* of a *mistakenpolicy* is asserted as *unsucessful* , in the second case the *outcome* of a *cruel* and *uncaring* policy is asserted as *successful* , in the third case an *ongoing* policy is asserted as partially successful leading to future *success* .

Thus as the three cases, above show we are able to model how changes in prior evaluation of a situation can be used to compute what the *meaning* of an utterance is. Crucially, as in most of these cases, the difference seems to be in which **source domain** schema gets invoked, and the resulting inferences. Of course, in all cases described above, the $Ut - type$ variable is changed from description to opinion or propaganda (cases a and c) suggesting that this sentence is highly judgemental and belongs in an editorial or syndicated column as opposed to the reported news.

It is somewhat interesting that even our simplistic model is able to detect these rather subtle differences in speaker intent and communicative goals. We believe the choice of the embodied x-schema is often a compact and efficient way to encode such information. Conversely, the unconscious choice by a speaker of an embodied term can give the hearer significant clues as to the prior belief and intent of the speaker, obviously something that needs far more exploration.

9.2 Test Results

After development on the database shown in Appendix A, we tested the system on the stories shown in Appendix C. In general, test stories continued to exercise many of the same representational features and abilities of our system that were critical to process the event descriptions in our development database in Appendix A. While our initial set of results were somewhat surprising since we had underestimated the range and extent to which embodied knowledge is projected onto abstract domains, results on the test stories were less dramatic, partly because confirmatory examples are everywhere. While the description below pertains to a few stories, we continue to find numerous relevant examples in every kind of story about international economics, from newspaper articles to magazines and even some journal articles. If anything, the evidence is fairly convincing that certain subtle aspects of communicative intent, evaluatory information, control and monitoring information, dynamic resource production/disappearance, goal threats, plan interrupts, etc. are **mostly** communicated using embodied terms relying on embodied metaphor-maps such as the ones described earlier.

Test stories are presented with a brief synopsis of the performance of our system on these previously unseen stories. In some instances we had to add some new knowledge to the lexicon (such as the name of a new *actor* in the domain of international policies). All such additions are documented in the synopsis as under the title **Additions** which appears immediately below the story in question. For cases where no modification was necessary, this entry is left blank. In the body of the brief synopsis, we attempt to briefly describe the inferences obtained through metaphoric projection of the embodied terms. Whenever there was a significant difficulty due to the syntactic or semantic structure of the narrative in question, we describe any input simplifications made. Following the synopsis, just as in the case of the development database, we show the complete I/O behavior for a few representative stories (stories 1 , 2 , and 3). The complete I/O behavior for these stories can be found in the Appendix (ref. Chapter 13).

Some test stories pointed to specific deficiencies in the implementation, while remaining compatible with the overall model. These mostly included new aspects of source domains or new kinds of metaphors that are possible extensions to the system described here. Section 9.2 describes some of these required extensions in greater detail.

9.2.1 Synopsis Of Test Results

Stories 1 - 8 below were processed without major problems. The concept of *slippery slope* in Story 8 is of-course an entire theory, and it is heartening that despite having never encountered the concept before, the right inferences were generated. Story 9 presents an interesting problem in that it has a *modal* should, and while the content of the story including the source domain x-schema of are relatively straightforward, the modal implication is **ignored** by the system. Story 10 is quite straightforward but requires knowledge source domain knowledge about entering, and target domain knowledge about the *EEC* . We believe this is a relatively minor addition. Several stories required additional knowledge of body posture and metaphor maps from postural conditions. While this further confirms our hypothesis of Embodied Metaphoric maps, further implementation is required before these stories can be processed. This and other minor extensions are discussed in the next section.

1. **Story**:

> Zaire on the brink of falling(H). Government teeters as rebels advance. (Oakland Tribune, May 14 , Front Page Headline) [10]

Additions: Added that Zaire was a country and the Actor was the Government of Zaire. Also target variables $in\_power(gov)$.

Inferences: Full I/O behavior is shown in Table 13.1 (ref. Appendix C). The main inferences here are aspectual; *on the brink* suggests that the Government is about to fail ($Fall \Rightarrow Fail$, $Status = READY$) but has not yet failed. Our **controller** abstraction and our hypothesis of aspectual inferences being invariantly projected was once again supported by this example.

2. **Story**:

> Protection is a *mistaken therapy* of *prescribing palliatives* to the economy in response to *painful change* and the *perception of injury.*(CNN transcripts, 1995).

Additions:

We translated the enabling condition from the *perception* of injury to asserting *injury* since our model has no way to account for beliefs about other people's beliefs etc. The palliative medicine x-schema already existed, so the rest of the example was relatively easy to process. Note that in response to palliative medicine, the true symptoms remain while the pain may be alleviated for a short while. [11]

Inferences: Figure 13.1(Chapter 11) shows the output for this test example. Here the inferences were mainly from the health and well being domain, where the x-schema corresponding to the *mistaken therapy*, and prescription of palliatives was able to conclude that while the economy looked good temporarily, the underlying symptoms continued to exist after the palliative was prescribed, and that the net effect was detrimental in the long run, including the speakers negative evaluation of Protectionist policies.

3. **Story**:

[10] Thanks to Collin Baker.

[11] The input to the system was the f–struct (Input (Context EconomicPolicy)(Policy Protection) (Event Palliative(Prot, EC)) (ENABLE(Pain, Event1)) (ENABLE(Perception(Injury)), Event1) (Ut-type Description))

> The U.S. economy *continues to struggle along* showing fresh signs of *weakness* almost daily.(BNC Corpus)

Additions:

Inferences: Table 13.2 (Chapter 13) shows the full I/O behavior for this example. The basic inferences include the aspectual one of *ongoing* weakness and struggle, and the current and anticipation of *difficulty.*

4. **Story**:

   > HEADLINE: THE U.S. ECONOMY *BEGINS TO CRAWL FROM THE MIRE* (National Review, 3/21/94).

   Additions:

   Inferences: Here the inferences were mainly aspectual, allowing the system to assert a *slow* economic *recovery* whose status was *ongoing* , and one that had a low degree of completion ($DOC = low+$) at the current time. Also, interestingly, begin allows the system to make the inference that at the last time step ($t = 0$), the US Economy was *ready* to emerge from recession, but the Economic State was still negative.

5. **Story**:

   > Economy *moving along* at the *pace of a Clinton jog.* (WSJ, May 2, 1997)

   Additions: Yes, we were forced to put a value to the pace of a Clinton jog (was rate $= 2$) (on a seven point scale).

   Inferences: Once we were presumptuous enough to code the slow rate of a Clinton jog, the rest of the inferential process was easy for the system, asserting an economy making very slow progress. Of course the inferences were completely humor-less, a deficiency we suspect will not be addressed in the foreseeable future.

6. **Story**:

   > HEADLINE: Indian Government "turns it back" on Protectionism (The New York Times, July 11, 1993)

   Additions:

Inferences: Turning your back involves facing the other way, resulting in a reversal of direction of motion. This changing direction corresponds to changing the policy in question. This is done in our system by activating the familiar $SMAP$ $Change\_direction \Rightarrow Change\_policy$. In the target domain we have the knowledge that Protectionism in an *inward oriented* policy. Changing the policy from being protectionist to non-protectionist changes this variable from $inward\_oriented$ to $outward\_oriented$. This, in turn changes the posterior distribution of *policy* variable at the current time step to allow all the outward oriented values (liberalization, Export-push, etc.) to have more of the probability mass. ($P(policy = Lib., 0)|(outward\_orientation, 0) \gg P(policy = Lib., 0)|(inward\_orientation, 0)$).

Of course, the economic knowledge that Liberalization and Protectionism are linked through inward and outward orientation. Hence the economy now becomes outward oriented, and all outward oriented policies become active.

7. **Story**:

   HEADLINE: BAD MEDICINE. Nafta took some time swallowing and has failed to cure the Economic ills of Mexico. (LA Times, May 1996)

   Addition: Had to add NAFTA as an Economic Policy variable (Policy Maker US), affecting Mexico.

   Inferences: Interestingly the headline named the appropriate x-schema directly! Inferences that the policy implementation took time and had difficulty in passing, and that the current outcome of the policy is one of failure.

8. **Story**:

   HEADLINE: Protectionism Is Our Most Important Product The United States plunged down the slippery slope of free trade. (The New York Times, July 11, 1993)

   Additions:

   Inferences: The system was able to infer that the US Government was implementing a Liberalization Policy which in the speaker's evaluation was highly negative ($Sp - Eval(Pol) = neg++$), and that the Economic State of the country was changing rapidly to a highly negative state (from plunging down at $SMAP$ $move(down) \Rightarrow$

$change(neg)$ and the $PMAP$ $move(rate = high) \Rightarrow change(rate = high)$), and that the future *stability* of the Economy was highly uncertain, and the the once the Government embarked on this *open* policy, they would *lose_control* (from slippery slope) of the process.

9. **Story**:

   > HEADLINE: US should *leap not crawl* in E.European policy. (Walesa (Agence France Press 1994)).

   Additions: The system currently has no way to interpret the **modal** *should*, so the input given was that $Sp - Eval(US)$ was positive in the case of $leap$, and negative in the case of $crawl$. This resulted in two inputs one corresponding to US leaps with $Sp - Eval$ positive and the other to $crawl$ with $Sp - eval$ negative.

   Inferences Generated: suggested that in the opinion of Speaker (Walesa), US should progress substantially in ($progress = good$) in implementing its $ongoing$ policy. rather than the East European policy. ($implement(policy)$, $Status = ongoing$, $progress(t_0) = slow$).

10. **Story**:

   > HEADLINE: New EU Members Cool Heels As Union Debates Voting BYLINE: Howard LaFranchi, Staff writer of The Christian Science Monitor. After negotiating to enter the EU, Austria, Finland, Norway, and Sweden have arrived on the doorstep only to find that those who asked them in are embroiled in a quarrel over the conditions they have to settle before they can open the door.
   > Now to the guests' further dismay, the arguing has degenerated into a "crisis" grave enough to leave the four standing on the doorstep for some time to come. In fact, the whole issue would have been put off until then if the EU were not forced to make a decision by the knocking at the door of the first four.

   Additions: Needs knowledge about entering, Not implemented.

   Inferences:

In summary, our results suggest that a large proportion of commonplace descriptions of abstract events seem to project familiar motion and manipulation concepts onto more abstract domains such as economics and politics. This allows non-experts to comprehend and reason about such abstract policies and actions in terms of more universal and commonplace concepts. Our results also provide evidence for our hypothesis that familiar and essential domain of spatial motion is encoded as highly accessible **compiled** knowledge required both for action monitoring and failure recovery but also used for fast, parallel, real-time reflex inference in interpretation. While we believe that our results confirm that we have taken a step in the right direction, there are several challenges to the system as is stands. Some of these issues are discussed in Chapter 10.

Chapter 10

Conclusion

This thesis was an exercise in theoretical cognitive science, where we attempted to demonstrate through computer modeling the plausibility that the semantics of motion and manipulation phrases are grounded in the fine-grained, active representations. We tried to provide evidence for this hypothesis by showing that that the dynamicity and reactivity provided by such representations is often exploited by the use of motion terms and expressions in discourse about abstract plans and processes.

Specifically, we took a small but interesting segment of ordinary discourse, namely the use of familiar motion and manipulation terms in complex event and policy descriptions. We built a computational model that demonstrates how the high degree of context-sensitivity inherent in the representation is routinely utilized to specify control, monitoring, resource and goal-based information about complex plans and processes that operate in an uncertain and dynamic environment. Our model shows the capability of making these discourse inferences in real-time, consistent with the fact that such information is available as reflex, automatic inference in narrative understanding.

While these initial results are encouraging, full validation awaits demonstration of the approach on a wider range of language problems. Particularly pressing are issues pertaining to integration with parsing, modeling image-schemas and image-schema transformations, and the ability to learn and extend metaphoric mappings. We are making progress on some of these issues and a summary of ongoing efforts is described below.

10.1 Parsing and Interpretation

Much of the difficulty of interpreting natural language, whether from speech or text input, comes from noise and ambiguity inherent to language. For example the problem of identifying phonological structures such as phonemes or syllables in acoustic input requires dealing with the massive ambiguity of acoustic and spectral features, a problem further compounded by noise. Allophonic or dialect variation as well as the lack of explicit segments in the signal makes the problem of identifying words given a phoneme stream similarly ambiguous. Lexical or morphological ambiguity (homophany, homography) and syntactic ambiguity cause identical difficulties in building a syntactic or semantic interpretation.

In order to deal with these problems, cognitive models of human language processing at every level have turned more and more to parallel architectures in which multiple representations are built and ranked by some metric. Many of these metrics are frequency-based or probabilistic; for example this is particularly true for speech processing, where the problem of identifying phones or phonemes in acoustics has long been addressed using statistically trained Gaussian models and probabilistic neural-networks to phonetically label a section of speech by choosing the phone which maximizes the likelihood of the acoustic data. But recent models of human lexical, syntactic, and semantic processing are probabilistic as well. Recent probabilistic models of syntax (Resnik 1993; Resnik 1992; Jurafsky 1996; Stolcke 1995) augment syntactic rules with various kinds of conditional probabilities, for use in choosing a preferred parse for syntactic ambiguities. Some models of semantics use probabilities of certain semantic combinations to compute the maximum-likelihood interpretation of an ambiguous utterance (Charniak & Goldman 1988; Wu 1992; Hobbs & Bear 1990). Furthermore, recent psycholinguistic models of syntactic and lexical disambiguation (MacDonald 1993; MacDonald *et al.* 1994; Trueswell & Tanenhaus 1991; Trueswell & Tanenhaus 1994) show that certain processing difficulties can be modeled by considering appropriate lexical frequencies.

But while preliminary probabilistic models exist for processing unique linguistic levels, the problem of combining these models and their probabilities, especially the association of probabilities with sophisticated linguistic structural representations has remained unadressed. Consider the cases in Chapter 9 which clearly shows that to obtain the kind of information that is required for understanding even simple discourse segments, a parser/interpreter must be capable of representing and reasoning with metaphorical mappings and other non-literal constructions.

In joint work with Dan Jurafsky, we are trying to address this state of affairs by asking the following questions. Can a coherent probabilistic interpretation be given for the problem of language interpretation at different levels? What kinds of conditional independence assumptions can we make in combining knowledge, and how can we represent these assumptions? How can sophisticated linguistic structural knowledge be combined with probabilistic augmentations?

In this regard, some early results (Jurafsky 1996; Narayanan & Jurafsky 1997) obtained using *Probabilistic Independence Networks* or *PIN* s (described in Chapter 7) demonstrate how individual factors contributing to language interpretation such a) Syntactic factors, b) Lexical and thematic factors, and c) morphological factors can be modeled in a uniform computational formalism that exploits conditional independence for compact representation and efficient inference. The early results suggest that simple combination rules using the same underlying formalism may be able to capture the interaction between various information sources/ factors in input disambiguation. Furthermore, the model is consistent with known experimental results of previous psycholinguistic experiments.

The work described above is completely consistent with our prototype implementation of the metaphor reasoning system described in Chapter 8. Our results (ref. Chapter 9) show the use of metaphoric knowledge quite early in the interpretation process and tighter integration with parsing is likely to be highly productive for access and disambiguation. In this connection, we are working toward integrating our probabilistic parser with the metaphor reasoning system. Since both the parser and the target domain network are based on PINs, we believe our x-schema/Belief network interface can be used with minor modifications to include parsing knowledge. But this is yet to be demonstrated.

While evidence combination from multiple sources is an important problem and building normative frameworks is useful, the Belief network inference algorithms do get intractable quite fast for multiply connected networks. Second, the representational power of such networks is currently limited to propositional knowledge, and for our purpose, this implies lack of ways to encode relational, dynamic and situated knowledge about actions, resources, etc. (also the temporally extended net cannot scale up at all). Such constraints required most previous efforts (notably (Charniak & Goldman 1988)) to rely on additional inferential machinery to dynamically construct propositional Belief networks from a first-order logic database (encoded as frames) while processing linguistic input. Besides being unable to exploit a key aspect of PINs to encode relevance efficiently through conditional independence, such an architecture is obviously cognitively implausible (consider arguments about reinterpretation from Chapter 2). So the question is very much open, *what might a cognitive interpreter look like?*

10.2 Extensions To the Aspect Model

In recent work, Nancy Chang (Chang 1997) has investigated the use of the aspect model described in Chapter 5 to handle general aspectual composition. She has focussed on the combination of verbs, arguments and temporal modifiers in terms of the conceptual features they impose or contribute. She proposes that the semantics of sentential aspect arises out of the combination of motor-schematic and image-schematic specifications.

Crucially, she argues that the active x-schema representation employed provides a way to relate both motor-schematic and image-schematic verbal specifications, since particular image schemas can enable or activate the dynamic, motor-schematic components. Thus, for example, *leave* may be *enabled* if some image corresponding to a notion of *in* is present, activating some motor process which in turn cannot *finish* until some image corresponding to *out* is satisfied. The representation also illuminates the interaction between event structure and nominals, which can instantiate pieces of the image-schematic representation (e.g. by serving as a goal or landmark, as in *walk to the park* or *leave the office*) or otherwise constrain possible interpretations (e.g. by forcing a stative reading of *The moat surrounds the castle* or a processual reading of *Soldiers are surrounding the castle*). Modification by *in-* and *for-* adverbials and appearance in progressive tense is also dependent on the features of the event structure: *in-* adverbials require some clear goal state; *for-*adverbials require a relation (either state or process) to either halt (without attaining any goal) or reverse. Progressive form may require a similar notion of temporariness and the (haltable) expenditure of effort. Nancy's work thus addresses at a level of detail amenable to computational modeling the issue of precisely how aspectual composition takes place, and accounts in a unified and cognitively motivated way for a number of well-known aspectual phenomena.

10.3 Extensions To the Metaphor Model

The metaphor model has many shortcomings, some of which were discussed in Chapter 8. One central issue with any project of this size is of course that the implementation described here is only a prototype model, and to be practically useful, we would need significantly more knowledge engineering both at the source and target domain level as well as in the number and types of mappings implemented.

Another shortcoming of the metaphor model (and the rest of the thesis) is the lack of an explicit model of *image-schemas*. In work so far, we have compiled away much of image-schematic reasoning into relationships between f–structs. However, we know that several metaphors are inherently imagistic and we currently have no way to handle such image metaphors. We pointed out some early work to address this issue in Nancy Chang's extension of the aspect model. The interesting claim she makes is that key features in image-schematic reasoning can be modeled as enabling or disabling x-schema transitions. This suggests a possible method to integrate image-schemas of the type modeled by Terry Regier (Regier 1996) with the x-schema based representations described here. Details await further research and implementation.

One issue brought up by our model is that embodied terms are often efficient means to encode communicative intent and evaluatory information. While we have identified the issue, the implemented model is very ad-hoc and does not do justice to the importance of this issue in discourse processing. For instance, previous research has stressed the importance of goal-interactions (Wilensky 1983) and intentions and attentional focus (Grosz & Sidner 1986) in narrative understanding. We do believe that active, context-sensitive representations such as x-schemas should prove a good framework to investigate such issues in detail, but we don't have much by way of results yet. In general, the move toward complex goal interactions, intentions and speech-act processing seem natural and promising areas to extend this work.

A significant long-term issue left undressed in this thesis is the learning and extension of metaphors. Work by Martin (Martin 1990) demonstrated that a central use of conventionalized metaphors was that they could be systematically extended to create new mappings. Thus the system of known and conventionalized metaphors provide the basis for learning novel mappings. We believe that this insight may provide us methods to extend the set of embodied mappings discussed in this thesis.

There are some bright sides, however. Recent linguistic work by Joe Grady and Chris Johnson (Grady 1996) provides evidence that to properly understand metaphor usage and extension, the constraints of embodiment are critical. The argument basically goes that complex mappings and extensions are essentially compositional if you get the primitives right, the *crux* being that primitive maps are direct experiential correlations.

10.4 X-schemas and Marker Passing

Marker Passing has been applied in *deductive*, *abductive* and *predictive* reasoning in text-inferencing systems (Norvig 1989; Wu 1992). Symbolic approaches as well as connectionist systems can be viewed as marker-passing systems. Traditionally, it has been argued that symbolic systems have the advantage that nodes have a well defined semantics and markers can carry variable binding information. However, symbolic approaches are less accurate, and are not able to use search control information as naturally as connectionist systems.

Marker passing seems to have a straightforward mapping to x-schema executions. In marker passing systems, the underlying representation is some sort of semantic network, with nodes representing predicates or constants and links representing various relations (subsumptive, part-off, etc.). The *occurrence* of a specific predicate-binding in the input results in that predicate being **marked** with the appropriate variable-bindings. In x-schemas, places represent predicates and the typed tokens represent variable bindings. Movement of tokens corresponds to marker passing.

In ongoing work, we are attempting to formalize marker-passing based on x-schemas. We believe such a model could be used to simulate and evaluate various marker passing proposals. This approach has several advantages over previous efforts. It provides a method to formally translate marker propagation and intersection hypotheses into analysis of the reachability graph of the Generalized Stochastic Petri Net ($GSPN$) that underlies our x-schema representation. The reachability graph of a $GSPN$ is isomorphic to a semi-markov process, so we can potentially bring in the various *lumping* theorems and the other exact and approximate estimation techniques for semi-markov processes into the analysis of marker passing algorithms.

10.5 X-schemas and Dynamic Logic

In recent years, resource-based logics such as Linear Logic (Girard 1987) have become popular as possible formalisms to represent and reason about action systems (Masseron *et al.* 1990). In (Narayanan & Thielscher 1997), we show a sound and complete translation between the multiplicative fragment of Linear Logic and non-durative x-schemas. This allows us to use linear deduction methods for x-schema composition in deliberative planning, and may provide some insights into the interleaved planning and execution. Furthermore, x-schemas and their connectionist realizations may prove to be good models for linear logic. The increased representational power of x-schemas, including their ability to model time as well as stochastic actions and world transitions may give us some clues to extend linear logic into the stochastic and temporal domains.

In related work with Michael Thielscher, we are looking at the possible connection between x-schemas and a dynamic version of situation calculus called *fluent calculus* (Thielscher 1995). So far, we have been able to come up with a sound encoding of non-stochastic x-schemas in a fluent calculus, which allows us to preserve the active nature of the x-schemas for execution control and monitoring while allowing provably correct x-schemas to be generated if required for exploration or in the case of catastrophic failures. However, we have still to investigate in detail the planning/execution interface x-schemas and the Fluent-calculus. Furthermore, the enhanced representational power of x-schemas, specifically the properties of time and stochasticity make coming up with a sound and complete translation difficult. While we have some preliminary results these questions are largely unanswered.

10.6 Epilogue

This thesis is a small step along a long path that attempts to construct theories of language rooted in perception, action and in the computational properties of the brain. While there is obviously a long way to go, and the terrain ahead is uncertain with many stumbling blocks, I defer any questions about the direction to Alan Turing's intuitions on the matter (thanks to Jerry Feldman for the quote).

Of (. . . many) possible fields the learning of languages would be the most impressive, since it is the most human of these activities. This field seems however to depend rather too much on sense organs and locomotion to be feasible.

—- Alan Turing, 1948.

Chapter 11

Appendix A: Databases and Program Parameters

11.1 Database Of Stories Used For Program Development

1. We *suffer* from one of the longest and worst sustained inflations in our national history. The *ills we suffer* have come upon us over several decades of *neglect.* This adminstration's objective will be a *healthy, vigorous economy.* (Reagan acceptance speech, 1980-81)

2. World Bank prescribed Structural Adjustment Programs (SAP) is bleeding the Indian Economy. (Report of World Women's conf in Beijing, 1995, pp.26)

3. In response to World bank *pressure* to *open up* its economy, the India Government *boldly set out on a path of liberalization.* It *loosened stranglehold* on business and *slashed import tariffs.* However, now the Government *is stumbling in its efforts* to implement the liberalization plan. (Economist, Asian Ed. 1994)

4. Taxes and inflation *have robbed us* of our savings. The great industrial machine *slowed down.* The time has come for the *economic emancipation, to be freed from the Government's grip.* We will lead a national fight to *tear down economic barriers* and liberate the spirit ..(Reagan 1980-81 Acceptance speech)

5. In the days ahead, I will propose *removing roadblocks that have slowed our economy* and reduced productivity. *Progress will be slow – measured in inches and feet not miles* – but we will progress. It is is time to *reawaken*

the industrial giant, to get Government back within its means and *lighten our punitive tax burden.* (Reagan 1984 acceptance speech)

6. Japan began its *long, painful slide* towards recession.

7. While others *sprint* or *jog* to the information super-highway, Europe is *slithering.* Governments must *step out of the way*, or intervene by *removing obstacles* ; not *impede progress* by attempting to *build it.* (Economist, 1995)

8. There is a price to pay for *changing course*; but it is less than the price of *plunging ahead carelessly.*(Michael Mendelbaum /PBSMcNeil Lehrer Dec. 11, 96).

9. HEADLINE: Vienna *stumbles* in waltz to join European Union

   AUSTRIA, once expected to *waltz smoothly into the European Union*, is *elbowing its partners*, *treading on toes and pogo-dancing* in a most un-Viennese manner.

10. HEADLINE: U.S. ECONOMY MAY TIP BACK INTO RECESSION

    The U.S. economy may be *on the verge of falling back into recession* after more than a year of half-hearted recovery. It *has been lurching forward but the anemic recovery..* (press association newsfile, Dec. 17, 1992)

11. HEADLINE: news analysis: european economic giant falls sick

    BYLINE: by xia zhimian

    DATELINE: bonn, december 17; ITEM NO: 1217071

    After 10 years of *steady flourishing, the european economic giant has now been taken ill.* Some say that germany *has walked into recession*, and others argue that it has not yet entered that unwelcome state.

12. ... major continental European economies *start to pull out* of recession, academic economists claimed today. However *Germany will remain stuck in recession* leading to a "two speed" recovery in Europe.

13. France has not yet signed the treaty (NPT), but at least they are *taking a cautious step in the right direction* toward world peace, safety and sanity.(Houston Chronicle, 4/17/92)

14. Buoyed by hefty doses of foreign investment and driven by reform-minded governments, Asia has *sidestepped* the *crippling debt crisis* that cost Latin America and Africa at least a decade of lost development. (by Ramon Isberto DATELINE: MANILA, Sept. 3,1993)

15. The defense buildup, along with the deep tax cuts in '81, *helped dig the fiscal hole* that the United States is *still trying to climb out of* - LA Times Editorial(Aug. 19, 1993).

16. As the United States and the UK *stagger to their feet*, continental Europe and Japan are *still battered and bloodied by recession*, forcing tens of thousands of redundancies, layoffs and production ... (Copyright 1993 Information Access Company; Copyright National Review Inc. 1993 National Review)

17. HEADLINE: ONCE MORE U.S. STUMBLES TO THE BRINK WITH NORTH KOREA (C. Krauthammer) HEADLINE: UNITED STATES, UNITED NATIONS *STUMBLE OVER ROCKY* RELATIONSHIP;

18. HEADLINE: "Reluctant Giant": Germany "Balks" At "Leading" Europeans To Victory.

    SECOND HEADLINE: An Economic "Colossus" Seems Politically "Impotent"

    Germany's shoulders, it turns out are *not very broad.*

19. Economic reform in China is like "Crossing a river by feeling for the stones" (described by an official) Among the *firm footings* the economy has *found*, the *slipperiest stone* was inflation. (nyt 02/08/93)

20. *Flinching* over Washington's *Hand in state wallets* (H) Government's *habit of striking its long arms* into the state's wallets bound to *squeeze* state and local budgets and only intensify the strains that have developed between Washington and the local governments.(nyt02/08/93)

21. The following expressions were all found in a single sunday issue of the New York Times (02/08/93).

```
Shoulder (responsibilities, burden)
Toe the line
(Cautious, small, large)steps
Measured steps
Economy floundering
US turning its back (on Bosnia)
US reorients policy
Uphold
Upright image
Veered in different directions
Close ties
Blind spots
Aging party
Far closer to the goal
Laid the groundwork
Stable and as nurturing Government
Speed up the negotiations
Slow down the talks
```

```
Fleeting second
Stand-off
Moved toward (recovery, adjournment, recess)
UN would support steps to ...
Like-minded countries
Political will
Political muscle
Germany throws its weight
Cripple food production
Washington is determined to press ahead
..Proposing force to prevent STRANGULATION of Sarajevo
Difficulties lie ahead
Economy continues to slog along
```

22. World Bank's Structural Adjustment Program's (SAP) is *forcing* developing countries to make huge payments to the World Bank. This is *bleeding* them of much needed resources. Recently, the "massive haermorrhage" has intensified. (Executive summary of World Women's conf in Beijing, 1995)

11.2 Implemented Source Domain x-schemas

Shown below are two examples of domain theories, one from the embodied domain of motion, the other from the familiar domain of health and well being. Note that the *agent* = *self* parameter is left unspecified but is present in all the embodied schemas.

Example 22 Domain Theory of Motion and Manipulation Shown below is a representation of a portion of the domain theory for embodied motion. Shown below are the set of parameterized x-schemas that constitute the domain. Each parameter has a finite set of values, including a default. In cases, where there is an explicit and important connection between two schemas (such as one schema interrupts another), the connected schema is mentioned alongside the dependent schema.

```
PARAMETERS:
rate = [low, normal, high(default = normal)]
step_size =[(1 - 7) default  = 3]
dir = [forw,back,side (default = forward)]
duration = [fast, default = med. long]
dist = [1 -7]
hole_depth = [low, med, high]
Energy Level = [low, med, high]
DOMAIN OBJECTS:
Agent = [Mover, Injurer]
Ground_type =[bump, slippery, stable]
Locations = [hole, ground, hill]
Obstacle_type = [barrier, counterforce, burden]
SCHEMAS:
============== Motion and Manipulation =====================
walk x-schema
walk slowly/fast (walk x-schema)
measured/giant/tiny/first/cautious/adroit steps
sidestep giant/great leap amble
step(sideways, away), tread,
saunter, swagger,
  slip (x-schema interrupts walk) reorient,
painful_slide (x-schema interrupts walk),
stumble (x-schema interrupts walk),
Fall Stabilize Trip Foot-Dragging Reorient.
slide x-schema
push(toward(x)/away), pull.
cut x-schema
```

```
slash, cut.
Grasp x-schema
grab, grip, pinch, flinch, squeeze
dig(hole) x-schema
Move x-schema
slow down
Change(dir)
turn(around)
==========Energy=============================
Replenish Stimulate Drain
============ Monitoring=======================
detect_obstacle(obstacle)
check(dist) check(rate)Monitor(dir)
Test_footing Test_ground Monitor_posture
=============Obstacle Avoidance =============
Deal_with_obstacle Approach_obstacle(rate, dist)
Apply_force
Smash
Go_around
Jump over
=========Force Dynamics=====================
Fight/resist(counterforce or injurer)
Shed/remove/lighten(burden)
Apply_force(dir), Apply_pressure(dir)
Remove_obstacle Impose_obstacle
Reduce_obstacle Injure Strangle (injure x-schema)
Strangle
Loosen Strangle-hold (injure x-schema)
```

■

The other set of schemas pertain to the Health and Well Being Domain. The implemented set of entities are presented after the general schema is presented.

Example 23

In the presence of symptoms, including weakness or pain or varying magnitude and duration, the patient may take part in the following four independent scenarios.

1. Patient goes to a *competent doctor* who diagnoses the illness, and prescribes a cure. Upon taking the prescribed medicine, the patient is cured and the symptoms disappear, leaving the patient healthy as before.

2. Patient goes to an ineffective doctor who prescribed palliatives which may kill the pain for a short while but leave the patient no healthier than before. This *ineffective doctor* scenario may lead to the *neglect* scenario.

3. Patient goes to a quack who may prescribe medicines and therapies that are potentially dangerous and may increase the illness rather than cure it. This situation will be referred to as the *quack* scenario.

4. Patient does nothing or *neglects* the illness, which may leads to increased diseased condition.

```
PARAMETERS:
Duration = [low, normal, high(default = normal)]
Magnitude =[(1 - 7) default  = 3]
Expense = [(1 -5) default = 3]
Pain = [(1 -7) default = 3]
DOMAIN OBJECTS/ROLES:
Doctor type = [Good Doctor, Surgeon, Incompetent Doctor, Quack,
      Harmer]
Patient type = [ideal, problem]
Illness type = [wound, systemic, cancer, psychological]
Medicine type = [mild, placebo, restorative, invasive]
Therapy type = [drugs, surgery]
DOMAIN SCHEMAS:
Cure, Diagnose Prescribe
Correct diagnosis (cure x-schema)
Efficient cure (cure x-schema)
Partial cure (cure x-schema)
Mistaken cure
Mistaken Diagnosis (mistaken cure x-schema)
Mistaken Therapy (mistaken cure x-schema)
Harm
Cause to Bleed (harm x-schema)
Neglect
```

■

11.3 Network Priors

Presented below are the network priors used for the *TARGET_NET* as well as the *DISCOURSE_NET*. For some variables, the probability mass is uniformly assigned to the variables, indicated by the use of the term Uniform. In all cases the *persistence* links (from copy at $t-1$ to copy at t) of variables (except for the *status* variable (ref. Chapter 7) was .9.

```
Actor/Implementor [USG (.6), IG(.1),uniform for others]
Policy Maker [WB (.5), USG(.2), IG(.2), IB(.1)]
New_policy [Lib (.6), Prot(.3), Uniform for others]
Do_Change [None (.7), Change (.3)]
Difficulty [t (.3), f (.7) ]
Policy Status [Uniform over all values]
Outcome [succeed (.6), fail(.4)]
Goal [lib(.5) Prot (.5)]
Progress [Uniform over values]
Deg. Complete [Uniform over values]
State [Low (.2) Med(.2), Hi(.3) Uniform for other values]
Inflation [t (.3), f(.7)]
```

The *DISCOURSE_NET* priors are shown below. Note that in some cases, (ref. Chapter 9) we changed the priors to see show how background evaluatory information could affect the processing of the input.

```
Ut-Type = [description(.5), assertion (.3), opinion (.1), propaganda(.1)]
Domain = [Ec-Policy (.3), Ec-State(.3), Health(.2), Embodied(.2)]
Sp-Eval(WB) = [neg+ (.1), neg (.2), neutral (.3), pos(.3), pos+(.1)]
Sp-Eval(IG) = [neg+ (.1), neg+ (.3), neutral (.2), pos(.3), pos+(.1)]
Sp-Eval(France) = [neg+ (.1), neg+ (.3), neutral (.2), pos(.3), pos+(.1)]
Sp-Eval(US_GOV) = [neg+ (.1), neg (.1), neutral (.3), pos(.3), pos+(.2)]
Sp-Eval(US_BUS) = [neg+(.01), neg(.1), neutral (.2),  pos(.39), pos+(.3)
Sp-Eval(Other_BUS) = [neg+(.01), neg(.1), neutral (.2),  pos(.39), pos+(.3)
Sp-Eval(Other_Players) = [neg+(.1), neg(.2), neutral (.3),  pos(.2), pos+(.2)
Sp-Eval(Liberalization) = [neg+(.01), neg(.1), neutral (.2),
pos(.39), pos+(.3)]
Sp-Eval(Protectionist) =[neg+(.3), neg(.39), neutral (.2),
pos(.1), pos+(.01)]
```

11.4 Implemented Maps

Shown below are some implemented metaphor maps mapping schemas, parameters and objects from the spatial motion and health domains onto the domain of international economics. Maps between specific instantiations of $OMAPS$ and $PMAPS$ such as the different types of obstacles to difficulty (the maps are many-to-many) are shown only if there is something interesting about the instantiation. Same is the case for the mapping from the CONTROLLER state to the target Belief net status variable.

Example 24 Motion To Events

$$\begin{aligned}
DOMAIN - MAPS(DMAPS) : \\
\mathcal{S} \Rightarrow \mathcal{T} \\
\mathcal{S} = [Motions, Actions] \\
\mathcal{T} = [AbstractEvents, Actions, EconomicPolicy] \\
OBJECT - MAPS(OMAPS) : \\
M_{o1} :< mover > \Rightarrow < agent > \\
M_{o2} :< location > \Rightarrow < state > \\
M_{o3} :< location = hole > \Rightarrow < state = recession. \\
M_{o4} :< obstacle > \Rightarrow < difficulty > \\
SCHEMA - MAPS(SMAPS) : \\
M_{s1} :< fall > \Rightarrow < fail > \\
M_{s2} :< setout > \Rightarrow < start > \\
M_{s3} :< step > \Rightarrow < implement\_policy > \\
M_{s4} :< detectobstacle > \Rightarrow < monitor\_difficulty > \\
M_{s5} :< dealwithobstacle > \Rightarrow < plan\_with\_difficulty > \\
M_{s6} :< turnaround) > \Rightarrow < undo\_plan > \\
M_{s7} :< goaround > \Rightarrow < choose\_alternate(action) > \\
M_{s8} :< gothrough > \Rightarrow < choose\_alternate(action) > \\
M_{s9} :< check(dist\_to\_goal > \Rightarrow < monitor(DOC) > \\
M_{s10} :< check(rate) > \Rightarrow < monitor(progress) > \\
M_{s11} :< adjust(motion) > \Rightarrow < adjust(policy) > \\
M_{s12} :< applyforce(mag) > \Rightarrow < regulate(mag) > \\
M_{s14} :< loosen(force) > \Rightarrow < reduce(reg) > \\
M_{s15} :< applypressure(dir) > \Rightarrow < result\_enable(change\_policy) > \\
M_{s16} :< Disable(alive) > \Rightarrow < Stop(exist) > \\
M_{s17} :< remove(obstacle) > \Rightarrow < deal\_with(diff) > \\
M_{s18} : Recover \Rightarrow Change\_state(neg, pos))
\end{aligned} \tag{11.1}$$

$$M_{s19} : Change\_direction \Rightarrow Change\_Policy$$
$$PARAMETER-MAPS(PMAPS):$$
$$M_{p1} :< disttogoal(1-7)) \Rightarrow < degreesinceinception(1-3) >$$
$$M_{p2} :< distfromsource(1-7) > \Rightarrow < degreeofcompletion(1-3) >$$
$$M_{p3} :< dir.ofmotion = forward > \Rightarrow < progress >$$
$$M_{p3} :< dir.ofmotion = backward > \Rightarrow < regress >$$
$$M_{p4} :< rateofmotion > \Rightarrow < progressrate >$$
$$M_{p5} :< rate(1-7) > \Rightarrow < Progress(low, on-schedule, hi) >$$
$$M_{p6} :< step.size(1-7) > \Rightarrow < progress(low - \ldots hi+) >$$
$$M_{p7} :< energy = low > \Rightarrow < \neg stable >$$
$$M_{p8} :< energy = low > \Rightarrow < growth\_rate = low >$$

■

Example 25 Health and Well-being To Economic Policies

$$DOMAIN-MAPS(DMAPS): \tag{11.2}$$
$$\mathcal{S}_H \Rightarrow \mathcal{T}$$
$$\mathcal{S} = [Health]$$
$$\mathcal{T} = [AbstractEvents, Actions, EconomicPolicy]$$
$$OBJECT-MAPS(OMAPS):$$
$$M_{o1} :< Patient > \Rightarrow < Business > | < Government > | < Country >$$
$$M_{o2} :< Doctor > \Rightarrow < WorldBank/IMF > | < Economy > | < Business >$$
$$M_{o3} :< Disease > \Rightarrow < Inflation > | < Recession > | < Growth- >$$
$$M_{o4} :< IncompetentDoctor > \Rightarrow < Government > |$$
$$M_{o4} :< Harmer > \Rightarrow < Government > | < WorldBank >$$
$$M_{o4} :< Blood > \Rightarrow < Resource(exist) >$$
$$M_{o4} :< Blood > \Rightarrow < Money >$$
$$SCHEMA-MAPS(SMAPS):$$
$$M_{s1} :< diagnose > \Rightarrow < diagnose >$$
$$M_{s2} :< prescribe(treatment) > \Rightarrow < suggest(corrective\_policy) >$$
$$M_{s2} :< prescribe(palliative) > \Rightarrow < suggest(ineffective\_policy) >$$
$$M_{s2} :< treat > \Rightarrow < Implement(corrective\_policy) >$$
$$M_{s3} :< cure > \Rightarrow <>$$
$$< succeed(implement(corrective\_policy) >$$
$$M_{s4} :< neglect(disease) > \Rightarrow < inaction >$$
$$M_{s5} :< harm > \Rightarrow < economicdownturn ++ >$$
$$M_{s6} :< behealthy > \Rightarrow < grow >$$

$$M_{s7} :< bevigourous > \Rightarrow < grow(high) > \wedge stable(economy)$$

$$M_{s8} :< drain(blood) > \Rightarrow < extort(money) >$$

$$PARAMETER - MAPS(PMAPS) :$$

$$M_{p1} :< Healthstatus > \Rightarrow < Economicstatus >$$

$$M_{p2} :< Pain > \Rightarrow < Unemployment > | < Businessclosure >$$

$$M_{p3} :< TerminalIllness > \Rightarrow < Uncorrectabledownturn >$$

$$M_{p4} :< Illness = systemic > \Rightarrow < StructuralProblem >$$

$$M_{p5} :< Illness = local > \Rightarrow < CorrectableProblem >$$

$$M_{p6} :< Prescription > \Rightarrow < plan >$$

$$M_{p7} :< Disease.mag \Rightarrow Infl.mag >$$

$$M_{p8} :< Disease.dur \Rightarrow Infl.dur >$$

$$M_{p9} :< Blood.amt \Rightarrow Money.amount >$$

■

Chapter 12

Appendix B: I/O Behaviour on Selected Stories

Table 12.1: $Input_1$ F-struct for Reagan Acceptance Story (Story 1) in database

Feature	$V(t_1)$
Event	suffer(US, Inflation)
Event.mag	high++
Event.duration	high++
Domain	Ec. State
State	Inflation
Ut-Type	description

Table 12.2: $Output_1$ F-struct for Reagan Acceptance Story

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$
Event = suffer(US, Inf)	t	t	t(.8)
Domain=EC. State	t	t	t
Ec.State=Infl	t	t	t
Ut-Type=Opinion	t	t	t(.6)
Ut-Type=Propaganda	f	f	t(.4)
$M_s : D \Rightarrow Infl$		A	A
$M_p : D.mag \Rightarrow Infl.mag$		A	A
$M_p : D.dur \Rightarrow Infl.dur$		A	A
Infl.Status = ongoing	t	t	t(.8)
Infl.mag = high++	t	t	t(.8)
Speaker-eval=neg++	t	t	t

Table 12.3: $Input_2$ F-struct for Reagan Acceptance Story

Feature	$V(t_2)$
Health State(US)	disease
Event	neglect(illness)
Event.dur	high++
Domain	Disease (.5)
Ut-Type	description

Table 12.4: $Output_2$ F-struct for Reagan Acceptance Story

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$
Event = suffer(US, Inf)	t	t	t(.8)	
Domain=EC. State	t	t	t	t
Ec.State=Infl	t	t	t	t
Ut-Type=Opinion	t	t	t(.4)	
Ut-Type=Propaganda	f	f	t(.6)	
$M_s : D \Rightarrow Infl$		A	A	A
$M_p : D.mag \Rightarrow Infl.mag$		A	A	
$M_p : D.dur \Rightarrow Infl.dur$		A	A	A
$M_s : Neglect \Rightarrow Inaction$			A	A
Infl.Status= ongoing	t	t	t	t(.8)
Infl.mag = high++	t	t	t	t(.8)
Inf. Cause =Inaction	t	t	t	t(.8)
Sp-eval(Pol)=change policy	t	t	t	t
Sp-eval=neg++	t	t		

Table 12.5: $Input_3$ F-struct for Reagan Acceptance Story

Feature	$V(t_3)$
Ut-Type	assertion
Domain	Ec.State
Goal(USG)	healthy(economy) $\wedge$ vigorous(economy)

Table 12.6: $Output_3$ F-struct for Reagan Acceptance Story

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$
Ut-Type =assertion				t
Event = suffer(US, Inf)		t	t	t
Domain=EC. State	t	t	t	t
Ec.State=Infl	t	t	t	t
Ut-Type=Opinion	t	t	f	
Ut-Type=Propaganda	f	f	f	
$M_s : D \Rightarrow Infl$		A	A	A
$M_p : D.mag \Rightarrow Infl.mag$		A	A	
$M_p : D.dur \Rightarrow Infl.dur$		A	A	A
Infl.Status= ongoing	t	t	t	t(.8)
Infl.mag = high++	t	t	t	t(.8)
$M_s : Neglect \Rightarrow Inaction$			A	A
Inf. Cause =Inaction	t	t	t	t(.8)
Sp-eval(Pol)=change policy	t	t	t	t
Sp-eval=neg++	t	t		
$M_s : Health \Rightarrow Growth$				A
$M_s : Vigourous \Rightarrow HighGrowth$				A
Objective(USG) = Growth				t
Objective(USG) = High Growth				t

Table 12.7: Input F-struct for "WB bleeds" story (story 22) in database

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$	$V(t_{3+})$
Domain=Int. Lending		t	t	t	
Ut-Type = Desc		t	t	t	
Event=force(WB, DEV, SAP, make-payments)		t			
Event=make-payments(DEV, WB, size=high++)		t			
Event=bleed(DEV, MONEY)			t		
Event =hemorrhage(DEV)				t	

Table 12.8: Output F-struct for “WB bleeds” story

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$	$V(t_{3+})$
Domain=International Lending		t	t	t	
Ut-Type = Desc		t	f	f	
Ut-Type = Opinion		f	t	t	
force(WB, DEV, SAP, payments)	f	t	t	t	t
can-pay(DEV, payments)	f	f	f	f	f
achieve (WB, payments)	t	t	t	t	t
make-payments(DEV, WB)	t	t	t	t	t
$M_s : F \Rightarrow C$	A	A	A	A	A
cause(SAP, make-payments(DEV, WB))	t	t	t	t	t
$M_o : Blood \Rightarrow Resource$			A	A	A
$M_o : Money \Rightarrow Resource$			A	A	A
$M_s : Bleed \Rightarrow Pay$			A	A	A
Sp-eval(SAP) = cruel	t	t	t	t	t
Continuous-loss(DEV, Money)	t	t	t	t	t
Systemic-Effect(DEV)	t	t	t	t	t
Hemorhage(DEV)t(.5)	t(.7)	t(.7)	t	t	t
Uncontrollable(DEV, bleeding)	t	t	t	t	t
Bleeding-amount(massive)	t	t	t	t	t
Exist(DEV) = False	t(.5)	t(.6)	t(.6)	t(.7)	t

Table 12.9: *Input* F-struct for “WB and Lib” story (Story 3) in database

Feature	$V(t_1)$	$V(t_2)$	$V(t_3)$	$V(t_{3+})$
$Event_1$ = pressure(World Bank, India)	t			
Event = respond(Ind, $Event_1$, Lib)		t		
Attitude(IG) = bold		t		
Event = set-out(India, Liberalization)			t	
Event= loosen-stranglehold(IG, Bus)				t
Domain = Economic Policy	t	t	t	t
Ut-Type = description	t	t	t	t

Table 12.10: *Output* F-struct for "WB and Lib" story

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$	$V(t_{3+})$
Event =pressure(WB, IG)		t			
Event = set-out(IG, LIB)			t		
Event = loosen-stranglehold(IG, I_Bus)				t	
Domain = Ec. Policy		t	t	t	t
Ut-Type = description		t(.6)	t(.5)	t(.4)	t(.4)
Ut-Type = opinion		t(.4)	t(.5)	t(.6)	t(.6)
$M_s : Pressure \Rightarrow Cause(ImpPol)$		A	A		
$M_p : P.goal \Rightarrow Pol.Goal$		A	A		
$M_p : P.dir \Rightarrow Pol.Plan$		A	A		
$M_s : set - out \Rightarrow start$		A	A		
$M_p : obstacle \Rightarrow difficulty$		A	A	A	
$M_{s_5} : deal - with - obstacle \Rightarrow remove - diff$		A	A	A	
Remove-diff(status) = READY		t	t	t	t
Goal = FT $\wedge$ Dereg			t	t	t
Difficulty				t	t(.7)
$M_s : force \Rightarrow Regulation$			A	A	
$M_p : force.mag \Rightarrow Reg.mag$			A	A	A
$M_p : loosen(force) \Rightarrow Reduce(Reg)$			A	A	A
$M_s : Disable(Alive) \Rightarrow Stop(Exist)$			A	A	A
Policy=Prot	t	t	t(.5)		
Policy=Lib			t(.5)	t	t
Status(Prot) = ongoing	t	t	t(.5)	f	f
Change-policy = reduce restrictions				t	t(.8)
Status(Lib) = f	f	t(.5)	t	t	t
Sp-eval(Lib.)=bad	t	t	t	t(.5)	f
Sp-eval(Lib.)=good	f	f	f	t(.5)	t

Chapter 13

Appendix C: Specific Test Results in Detail

Table 13.1: I/O Behavior for *Zaire on Brink Of Falling* (Story 1) in The Test Results Section

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_3)$
INPUT FEATURES				
Domain=Government Status		t	t	
Government = Zaire	t	t	t	t
Ut-Type = Desc		t	t	
Event= brink-of(falling)		t		
Event= teeters(Gov.)			t	
ACTIVE MAPS				
OUTPUT FEATURES				
Fail(Gov)	t(.6)	t(.7)	t(.8)	t(.8)
Fail.Status = ready	t(.6)	t	t	
In-Control(Gov)	t	f(.7)	f(.8)	f(.8)
In-Power(Gov)	t	t	t(.6)	t(.4)

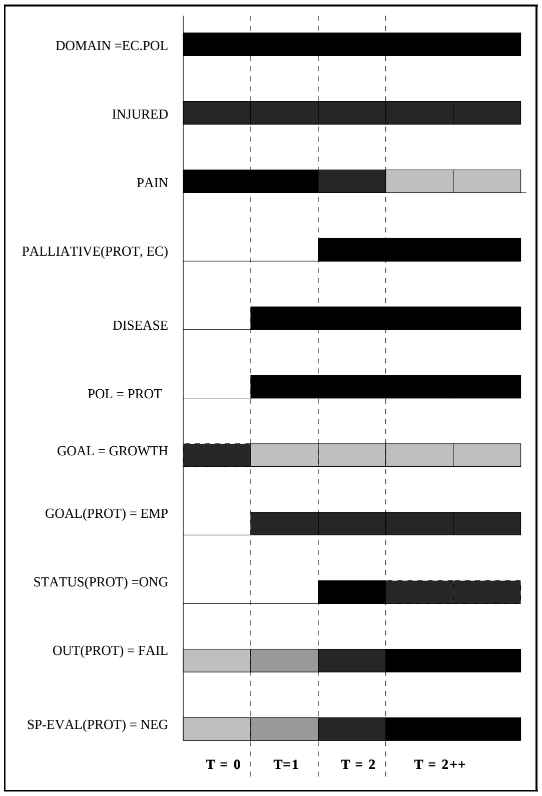


Figure 13.1: An example of the health-well being source domain and its projection onto the domain of plans. Corresponds to Story 2 in the *Test Results* section. Note that the disease remains after the palliative medicine is applied, hence the Negative economic state persists after the protectionist policy is in force.

Table 13.2: System Behavior For US Economy Struggles (story 3) in test results

Feature	$V(t_0)$	$V(t_1)$	$V(t_2)$	$V(t_{2+})$
INPUT FEATURES				
Event = CONTINUE(struggle)	t	t	t	t(.7)
Event = inc.(weakness)			t	t
Domain = Economic State	t	t	t	t
Actor = U.S.Economy	t	t	t	t
OUTPUT FEATURES				
Ec.State	neg	neg	neg+	recession(.8)
Sp-Eval(Pol) = neg+	t(.7)	t(.7)	t(.8)	t

Bibliography

Agre, P., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proc. AAAI-87*, 268–272, Seattle, WA.

Connell, J. (1989). *A Colony of Robots.* Ph.D. Thesis, MIT, AI-Lab, 1989.

Arbib, M.A., (1992). Schema Theory. In the *Encyclopedia Of Artificial Intelligence*, 2nd. Edition, edited by Stuart Shapiro, 2:1427-1443, Wiley, 1992.

Arbib, M.A. (1994). *The Metaphorical Brain: 2.* Wiley Interscience, 1994.

Arkin, R.C. (1990). Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation. *Robotics and Autonomous Systems* 6 (1990) 105-122.

Bailey, D. (1997). *A Computational Model of Embodiment in the Acquisition of Action Verbs.* Ph.D. Thesis, University of California Berkeley (to appear).

——, Feldman, J.A., Narayanan, S., & Lakoff, G. (1997). Modelling Embodied Lexical Development. *Proc. 19th Conference of the Cognitive Science Society*, 1997.

Ballard, D.H., Hayhoe, M.M., Pook, P.K., & Rao, R.P.N. (1997) Diectic codes for the embodiment of cognition. *Behavioral and Brain Sciences*, in press.

Barnden, J. Helmreich, L. Iverson, E. Stein, G.C. (1994). An integrated implementation of simulative, uncertain, and metaphorical reasoning about mental states. *Prinicples of KR and Reasoning: Proceedings of the Fourth International Conference* (Bonn, Germany, 24-27 May 1994), San Mateo, CA: Morgan Kaufmann.

—— and Holyoak, K. eds. (1994). *Advances in Connectionist and Neural Computation Theory.* Volumes 1-3, Ablex Publishing Corp. New Jersey ISBN: 1-56750-101-X.

Barrett & Weld (1994) Partial-order Planning: Evaluating Possible Efficiency Gains. *Artificial Intelligence*, 67: 71-112, 1994. Lawrence Earlbaum, 1982.

Bennett, P.A. et al, 1986. *Multilingual Aspects Of Information Technology.* Gower, Brookfield, VT, 1986.

Berlin, B., and Kay, P. (1969). *Basic Color Terms: Their Universality and Evolution.* Berkeley, CA: University of California Press.

Berman, R., and Slobin, D., 1994. *Relating Events in Narrative: A Crosslinguistic Developmental Study.* LEA Press, 1994.

Bernstein, N. A., (1967). *The Co-ordination and Regulation of Movement.* New York: Pergamon Press.

Bhagwati, J. (1987). Rethinking Trade Strategy. *Development Strategy Reconsidered*, Transaction Books, N.J.

Blum A. and Furst M. (1995). Fast Planning through Planning Graph Analysis. *Proc. of IJCAI95*, 1636-1642, Montreal 1995.

Borchardt, G.C. (1994). *Thinking Between The Lines: Computers and Comprehension of Causal Descriptions.* MIT Press 1994.

Bradford, C. (1991). Policy Interventions and Markets: Development Strategy Typologies and Policy Options. *Manufacturing Miracles.* Princeton University Press, 1991. pp. 32-51.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14-23.

Bullock, D. (1995). Motorneuron Recruitment. *Handbook Of Brain Theory*, MIT Press 594-97.

Cacciari, C. and Tabossi, P. (1988). The comprehension of idioms. *Journal of Memory and Language*, 27, 668–683.

Carbonell, J. (1982) Metaphor Comprehension. *Strategies for Natural Language Processing*, 413–433, Lawrence Earlbaum, 1982.

Carlson, G. N. and Tanenhaus, M. K. (1987). Thematic roles and language comprehension. In W. Wilkins (Ed.), *Thematic relations.* Academic Press, San Diego.

Chang, N. (1997). A Cognitive Approach to Aspectual Composition. Cognitive Linguistics Class Term Paper. ICSI Technical Report (in preparation), August 1997.

Chapman, D. (1987). Planning for Conjunctive Goals. *Artificial intelligence*, 32:333-377 1987.

Charniak, E. 1976. Inference and Knowledge. *Computational Semantics*, Charniak and Wilks (eds.), North-Holland Publishing Company pp.1-22 (1976).

—— and Goldman, R. (1988). A logic for semantic interpretation. In *Proceedings of the 26th ACL*, Buffalo, NY.

Comrie, B. 1976. *Aspect.* Cambridge University Press.

Cottrell, G., (1985). *A Connectionist Approach to Word Sense Disambiguation* (PhD thesis). University of Rochester, Morgan Kaufmann, CA 1985.

Darwiche A. and Pearl J. (1994) Symbolic Causal Networks. *Proc. Of the Twelfth National Conference on AI*, 1994.

d'Arcais, G. B. F. (1993). The comprehension and semantic interpretation of idioms. In C. Cacciari and P. Tabossi (Eds.), *Idioms: Processing, Structure, and Interpretation.* Lawrence Erlbaum Associates, New Jersey.

Dean, Tom and Wellman, Michael, (1991). *Planning and Control.* Morgan Kaufman Series in Representation and Reasoning, 1991.

Damasio, A. R. , (1989). Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33, 25-62.

Dorr, Drew, (1993). Interlingual Machine Translation. In *AI Journal, V63 (1993) : pp. 429-492* .

Dowty, D. 1979. *Word Meaning and Montague Grammar.* Synthese Language Library, Reidell, New York.

Drescher, G.L. (1991). *Made-up minds: A constructivist approach to AI.* MIT Press, 1991.

Feldman J. A., & Ballard, L. (1982). Connectionist Models and Their Properties. *Cognitive Science* No. 6, 205-254 (1982)

—— & Shastri, L. (1984). Evidential Inference in Activation Networks. *Proc. 6th Conference of the Cognitive Science Society*, 1984. pp. 156-160.

—— and Waltz, D., eds., (1988) *Connectionist Models and Their Applications.* Ablex Publishing Company, 1988.

—— 1989. Neural Representation Of Conceptual Knowledge. *Neural Connections, Mental Computation.* MIT Press 1989.

——, Lakoff G., Bailey D.A., Narayanan S., Regier T., & Stolcke, A. (1996). L_0—the first five years of an automated language acquisition project. *AI Review* 8.

Feldman, J. L. (1986). Neurophysiology of respiration in mammals. *Handbook of Neurophysiology: Section 1, The Nervous System.* ed. F. Bloom, 4:463-524, Bethesda, Md: Am. Phisiol. Soc.

Bernstein, N. A. (1967). *The Co-ordination and Regulation of Movement.* New York: Pergamon Press.

Fillmore, C. J, (1988). The mechanisms of "Construction Grammar". In *Proceedings of BLS 14*, pages 35–55, Berkeley, CA.

Fikes, R. & Nilsson (1971). STRIPS: A new approach of theorem proving to problem solving, *Artificial Intelligence*, No. 2 189-208.

Firby, J. (1989). Adaptive Execution in Complex Dynamic Worlds. *Reasearch Report No. 672*, Yale University.

Fung R. & Chang, K.C. (1990). Weighting and integrating evidence for stochastic simulation in Bayes networks. *Proc. UAI 5*, 209-219, Elsevier, Amsterdam, 1990.

Gallese, V., Fadiga V., Fogassi L., & Rizzolatti G. (1996). Action recognition in the premotor cortex. *Brain* 119(2), 593–609.

Gelfond, M. & Lifschitz, V. (1993). Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17:301-322, 1993.

Grafton, Scott T., Arbib, M. A., Fadiga L., and Rizzolatti G. (1996). Localization of grasp representations in humans by PET: 2. Observation compared with imagination. *Experimental Brain Research* (112), 103–111.

Gibbs, R. Jr. (1994). *The Poetics Of Mind.* Cambridge University Press, 1994.

Gary Gareffi, 1991. Paths Of Industrialization: An overview. *Manufacturing Miracles*: Princeton University Press, 1991. pp. 32-51.

Girard, J. (1987). Linear Logic. *Theoretical Computer Science* 50 (1987) 1- 102.

Goldberg, A. (1995). *Constructions.* UC Press.

Goldszmidt, M. & Pearl, J. (1992). Rank-based systems: A simple approach to belief revision, belief update and reasoning about evidence and actions. *Proc. Of the Third Conference on Principles of KR and Reasoning*, 1992: 661-672, Morgan Kaufman, Inc. 1992.

Grady J. (1996). A Compositional Theory of Metaphor. Presented at CSDL-II, Buffalo NY. Proceedings (to appear) CSLI/Cambridge University Press, 1998.

Grosz, B. and Sidner C. (1986). Attentions, Intentions, and the Structure of Discourse. In *Computational Linguistics, 12(3), pp. 175-204* .

Head, H. (1920). *Studies in Neurology.* Hodder and Stroughton, London, 1920.

Henderson, J. (1994). Connectionist Syntactic Parsing Using Temporal Variable Binding. *Journal of Psycholinguistic Research*, 23(5):353-379.

Hobbs, J. R. and Moore, B. (1985). *Formal Theories Of The Commonsense World.* Ablex Publishing Corporation, NJ.

—— and Bear, J. (1990). Two principles of parse preference. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, pages 162–167, Helsinki.

Hummel, J.E. & Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review* 99:480-517.

Hwang, C. H. & Schubert, L. (1994). Interpreting Tense, Aspect, and Time Adverbials: A Compositional, Unified Approach. *Proceedings of the First International Conference on Temporal Logic (ICTL 94)*, July 1994, Bonn. Germany, 1-27.

Indurkhya, B. (1992). *Metaphor and Cognition.* Kluwer Academic Publishers.

Jeannerod, M. (1986). The formation of finger grip during prehension. A cortically mediated visuo-motor pattern. *Beh. Brain Res.*, 19:99-116.

Jelinek, F. and Lafferty, J. D. (1991). Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17, 315–323.

Jensen, F. (1996). *An Introduction to Bayesian Networks.* Springer-Verlag ISBN 0-387-91502-8.

Johnson, M. (1987). *The Body In The Mind: The Bodily Basis of Meaning, Imagination, and Reason.* University Of Chicago Press, ISBN 0-226-40318-1.

Jurafsky, D. (1996). A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science* 20, 137–194.

Kay, P. and McDaniel C. K.(1978). The linguistic significance of the meaning of basic color terms. *Language* 54(3), 610–646.

Kawamoto, A. H. (1993). Nonlinear dynamics in the resolution of lexical ambiguity. *Journal of Memory and Language*, 32, 474–516.

Kjaerulff, U. (1992). A computational scheme for reasoning in dynamic probabilistic networks. *Proc. UAI-92*, 121-129,Stanford.

Kolodner, J. (1984). *Conceptual Memory: A Computational Perspective.* Lawrence Earlbaum, Hillsdale, NJ.

Langacker, R. (1987). *Foundations of Cognitive Grammar I: Theoretical Prerequisites.* Stanford University Press, Stanford.

Lange, T. and Dyer, M. (1989) High-level Inferencing in a Connectionist Network. *Connection Science*, 1 (2), pgs. 181-217, 1989.

Lakoff, G. and Johnson, M. (1980). *Metaphors we live by.* University Of Chicago Press.

—— (1987). *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press.

—— (1994). What is Metaphor?. *Advances in Connectionist Theory. V3 : Analogical Connections*, V3,1994.

Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society* B, 50:127–224.

Lifschitz, V. (1990). Frames in the Space of Situations. *Artificial Intelligence* 46:365-376, 1990.

Luce, P. A., Pisoni, D. B., and Goldfinger, S. D. (1990). Similarity neighborhoods of spoken words. In G. T. M. Altmann (Ed.), *Cognitive Models of Speech Processing.* MIT Press, Cambridge, MA.

MacDonald, M. C. (1993). The interaction of lexical and syntactic ambiguity. *Journal of Memory and Language*, 32, 692–715.

MacDonald, M. C., Pearlmutter, N. J., and Seidenberg, M. S. (1994). Syntactic ambiguity resolution as lexical ambiguity resolution. In *Perspectives on Sentence Processing.* Erlbaum, Hillsdale, NJ.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The Penn treebank. *Computational Linguistics*, 19, 313–330.

Marslen-Wilson, W. (1990). Activation, competition, and frequency in lexical access. In G. T. M. Altmann (Ed.), *Cognitive Models of Speech Processing.* MIT Press, Cambridge, MA.

Marslen-Wilson, W., Brown, C. M., and Tyler, L. K. (1988). Lexical representations in spoken language comprehension. *Language and Cognitive Processes*, 3, 1–16.

Martin, J. (1990). *A Computational Model of Metaphor Interpretation.* Academic Press, NY, 1990.

Masseron, M. and Tollu, C., and Vauzeilles, J (1990). Generating Plans in Linear Logic. *Foundations of Software Technology and Theoretical Computer Science*, vol 472 of *LNCS*, p 63-70, Springer 1990.

McCarthy, J., Hayes, P. (1969). Some Philosophical Problems From the Standpoint of Artificial Intelligence. *Machine Intelligence* 4 (1969) 463-502.

McCawley, J. D. (1971). Tense and time reference in English *Studies in Linguistic Semantics*, New York: Holt, Reinhart and Winston, pp. 96-113.

McClelland, J. L., St. John, M., and Taraban, R. (1989). Sentence comprehension: A parallel distributed processing approach. *Language and Cognitive Processes*, 4, 123–154.

Mcdermott, D. (1982). A Temporal Logic For Reasoning About Processes and Plans. In *Cognitive Science 6: pp. 101-155* .

Miller, G. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicaography*, 3(4)(Special Issue).

Moens, M. & Steedman, M. (1988). Temporal Ontology and Temporal Reference. In *Proc. ACL–88, V4, Number 2, June 1988, pp. 15-29*.

Molloy, M.K., et al, (1982). Performance Analysis Using Stochastic Petri Nets. *IEEE Transactions on Computers, C-31, No.9, pp.913-917* .

Murata, T. (1989). Petri Nets: Properties, Analysis, and Applications. In *Proc. IEEE–89, V77, Number 4, April 1989, pp. 541-576*.

—— (1991). Planning with Petri Nets. In *Information and Control*, 1991.

Nakhimovsky, A. (1988). Aspect, Aspectual Class, and the Temporal Structure Of Narrative. In *Proc. ACL–88, V4, Number 2, June 1988, pp. 29-44*.

Narayanan, S. (1996). Embodiment in Language Understanding: Modeling the Semantics of Causal Narratives. *AAAI Symposium on Embodied Cognition and Action*, AAAI Press TR:FS-96-02.

—— (1997). Walking the Walk is Like Talking The Talk: A Computational Model of Verbal Aspect. *Proceedings of the Nineteenth Annual Conference on Cognitive Science Society* (to appear) 1997. A longer and more linguistically oriented paper was Presented at CSDL-II, Buffalo, NY. Proceedings (to appear) CSLI/Cambridge University Press, 1998.

—— & Jurafsky, D. (1997). Exploiting Conditional Independence in Language Understanding ICSI TR-97-14, July 1997. Also submitted to *Computational Linguistics.*

—— & Thielscher, M. (1997). A connectionist model for dynamic resource-based logics. Working Paper. Available from http://www.icsi.berkeley.edu/ snarayan.

Nilsson, N. J. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research* 1, 139–158.

Norman, D.A. & Shalice, T (1980). Attention to Action: willed and automatic behavior. *Human Information Processing, Tech. Report* No. 99, UC San Diego.

Norvig, P. (1989). Marker Passing as a Weak Method for Text Inferencing. *Cognitive Science* No. 113: 569-620.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, San Mateo, Ca.

—— (1994). A probabilistic calculus of actions. *Proc. UAI-94*, 454-462,Stanford.

Pearson K.G. (1993). Common Principles of Motor Control in Vertebrates and Invertebrates. *Ann. Review Of Neuroscience*, 1993, 16:265-97.

Portinale, L. (1994). *A Petri net Model of Abduction.* Ph.D. Dissertation, University of Torino, 1994.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago.

Reichenbach, H. (1947). *Elements Of Symbolic Logic.* Berkeley, CA, University Of California Press.

Reisig, W. (1985). *Petri Nets.* Springer Verlag.

Regier, T. (1996). *The Human Semantic Potential: Spatial Language and Constrained Connectionism.* Cambridge, MA: MIT Press.

Resnik, P. (1992). Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the 14th International Conference on Computational Linguistics*, pages 418–424, Nantes, France.

Resnik, P. (1993). *Selection and Information: A Class-Based Approach to Lexical Relationships.* PhD thesis, University of Pennsylvania. (Institute for Research in Cognitive Science report IRCS-93-42).

Rieger C. (1975). Language Comprehension. *Readings in KR*, Morgan-Kaufman, 1975.

Rosenschein, J.S. (1985). Formal theories of knowledge in AI and Robotics. *New Generation Computing* 3(4):345-357.

Sacerdoti, Earl D. (1975). The nonlinear nature of plans. In *Proc. IJCAI–75*, 206–214.

Sag, I. A., Kaplan, R., Karttunen, L., Kay, M., Pollard, C., Shieber, S., and Zaenen, A. (1985). Unification and grammatical theory. In *Proceedings of the Fifth West Coast Conference on Formal Linguistics.*

Salasoo, A. and Pisoni, D. B. (1985). Interaction of knowledge sources in spoken word identification. *Journal of Memory and Language*, 24, 210–231.

Schank, R.C. & and Abelson, R.P. 1977. *Scripts, Plans, Goals, and Understanding: An inquiry into human knowledge structures.* Hillsdale, NJ:Erlbaum 1977.

Schmidt, R.A. (1975). A Schema Theory of Discrete Motor Learning. *Psychol. Rev.*, 82, 225-60.

Shastri, L. & Ajjanagadde, V. (1993). From simple associations to systematic reasoning. *Behavioral and Brain Sciences*, 16:3, 417-494.

—— & Grannes, D.J. (1996). A Connectionist Treatment of Negation and Inconsistency. *Proc. 18th Conference of the Cognitive Science Society*, 1996. pp. 142-147.

——, Grannes, D.J. Narayanan, S. and Feldman.J A. (1997). Connectionist parameterized routines. Submitted to Neural Information Processing Systems 10. Also presented as a poster at the 19th Cognitive Science Society Conference.

Simpson, G. B. and Burgess, C. (1985). Activation and selection processes in the recognition of ambiguous words. *Journal of Experimental Psychology: Human Perception and Performance*, 11, 28–39.

Singer, W. (1993). Synchronization of cortical activity and its putative role in information processing and learning. *Annual Review of Physiology* 55: 349-74

Siskind, Jeffrey Mark, (1995). A computational study of lexical acquisition. *Cognition* 50, 1–33.

Srinivas, S. and Breese, J.S. (1990). IDEAL: A software package for the analysis of belief networks. In *Proceedings of Sixth Workshop on Uncertainty in AI*, Cambridge, Mass. 1990.

Steedman, M. (1995). Dynamic Semantics for Tense and Aspect. *Proc. IJCAI 1995*, pp. 1292-98 1995.

—— (1996). Temporality. *Situation Calculus and its Applications* CSLI Press, Stanford 1996.

Sternberg, S. Monsell, R. Knoll, R.L. & Wright C.E. (1978). The latency and duration of rapid movement sequences: comparisons of speech and typewriting, *Information Processing in Motor Control and Learning*, G. E. Stelmach, Academic , New York: 117-152, 1978.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21, 165–202.

Sweetser, E. (1990). *From Etymology to Pragmatics: The mind-as-body metaphor in semantic structure and semantic change.* Cambridge, UK: Cambridge University Press, 1990.

Swinney, D. A. and Cutler, A. (1979). The access and processing of idiomatic expressions. *Journal of Verbal Learning and Verbal Behavior*, 18, 523–534.

Talmy, L. (1985). *Lexicalization Patterns, Semantic Structures in Lexical Form.* in Language Typology And Symbolic Description, v3: Cambridge University Press, 1985.

—— (1987) Force Dynamics in Language. Tech Report. Institute For Cognitive Science, UC Berkeley, 1987.

Tanenhaus, M. K. and Lucas, M. M. (1987). Context effects in lexical processing. *Cognition*, 25, 213–234.

Tanji J. and Shima S. (1994) The supplementary motor area in the cerebral cortex. *Nature*, vol. 371, issue 6496, (SEP 29, 1994) : pp. 413-416.

Thielscher, M. (1995). The Logic of Dynamic Systems. *Proceedings of IJCAI*, Vol2, pp. 1956-62 1995.

Tollu, M and Masseron, M. K. (1993). Deductive Planning and Linear Logic. *Theoretical Computer Science*, 234-243, 1993.

Trueswell, J. C. and Tanenhaus, M. K. (1991). Tense, temporal context and syntactic ambiguity resolution. *Language and Cognitive Processes*, 6, 303–338.

Trueswell, J. C. and Tanenhaus, M. K. (1994). Toward a lexicalist framework for constraint-based syntactic ambiguity resolution. In *Perspectives on Sentence Processing.* Erlbaum, Hillsdale, NJ.

Trueswell, J. C., Tanenhaus, M.K., and Garnsey, S. M. (1994). Semantic influences on parsing: Use of thematic role information in parsing. *Journal of Memory and Language*, 1194, 285–317.

Tyler, L. K. (1984). The structure of the initial cohort: Evidence from gating. *Perception and Psychophysics*, 36, 417–427.

Vendler, Z. (1967). *Linguistics in Philosophy.* Cornell University Press, Ithica, New York.

Weber, S. (1989) Figurative Adjective-Noun Interpretation in a Structured Connectionist Network. *Proceedings of the 11th Annual Meeting of the Cognitive Science Society*, pgs. 204-211, 1989.

Webber, B. (1988). Tense as Discourse Anaphor. In *Proc. ACL–88*, V4, Number 2, June 1988, pp. 61-74.

Wilensky, R. (1983). *Planning and Natural Language Understanding.* Addison Wesley, 1983.

—— (1991). Extending the lexicon by exploiting subregularities. Technical Report TR-91-618, University of California at Berkeley Computer Science Division.

Wu, D. (1992). *Automatic inference: A probabilistic basis for natural language interpretation.* PhD thesis, University of California, Berkeley. Computer Science Department Tech. report UCB/CSD 92/692.

Zwitserlood, P. (1989). The locus of the effects of sentential-semantic context in spoken-word processing. *Cognition*, 32, 25–64.