Structural Classification

A Preliminary Report

Jana Koehler* Kilian Stoffel†

James A. Hendler‡

TR-96-023

July 1996

Abstract

A new type of classification algorithm is introduced that works on the folded representation of concepts. The algorithm comprises two phases: a preprocessing phase working on the normal-form representation of concepts to test for unsatisfiability and tautology, and a structural classifier that generates predecessors and successors of concepts by exploiting new optimization techniques not available to standard classifiers.

Working on the folded terminology instead of its expanded and normalized representation allows to significantly reduce the number of subsumptions tests that are necessary to correctly classify a concept. We describe the algorithm, and prove it sound and complete for two different languages. It can be extended to more expressive languages when combined with a new method for reasoning about number restrictions over role hierarchies based on diophantine equations.

The algorithm is very fast and very well parallelizable taking less than 4 hours for the classification of a terminology of 100,000 concepts on an SP2.

*On leave from German Research Center for AI (DFKI), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, e-mail: koehler@dfki.uni-sb.de

†University of Maryland Department of Computer Science College Park, MD 20742, USA, e-mail: stoffel@cs.umd.edu

‡University of Maryland Department of Computer Science College Park, MD 20742, USA, e-mail: hendler@cs.umd.edu

Contents

1 Introduction

The demands of modern information technology applications, such as search engines for large heterogeneous knowledge sources like the World Wide Web or the mining of knowledge from databases, place new requirements on the development of the technology of knowledge bases. These systems must now scale to extremely large sizes while still being able to deliver the benefits of knowledge based inferencing in terms of expressivity and powerful inferential capabilities. Work in the PARKA project has been focusing on the application of high performance computing technologies to AI knowledge bases with a specific focus on developing knowledge representation languages that can support applications with millions of assertions.
In particular, our work focuses on developing both an implementation and underlying computational science of the development of very large knowledge bases. We have designed and implemented several versions of the KR language PARKA, which have been supported on various supercomputers (including a recent implementation in C which runs on a wide variety of machines including IBM SP-2, CRAY T3D, and Paragon platforms). In addition, PARKA is being used to support applications including case-based planning, defense transport logistics, and medical informatics – supporting reasoning in knowledge bases ranging from tens of thousands to millions of logical assertions [14, 13, 15].
However, although previous work has focused on developing a powerful language including exception handling during inheritance, a graphical query interface, and tools for browsing and editing very large knowledge bases, until recently little work has gone into examining the underlying formal properties of the system. Thus, while PARKA can handle complex "structure-matching" queries for a two million assertion knowledge base in a few hundred milliseconds on a single processor Sparc, we have not clarified the underlying descriptive reasoner or related the work to ongoing research in the area of description logics.
In this paper, we take a step at removing this deficiency by relating the underlying algorithm for some of PARKA's most efficient inferencing to work on the classification algorithms for description logics. We show that PARKA's techniques yield a new type of classification algorithm relying on the structure of concepts descriptions and prove it sound and complete.

2 Classification

Description logics offer a convenient way to represent the taxonomic and conceptual knowledge of a domain in the form of a *terminology*. To reason about terminologies, two main inferential services are necessary. *Subsumption*, which determines whether all instances of a concept are necessarily instances of another concept by taking into account the concept definitions, and *classification*, which computes the concept *taxonomy*, i. e. *all the subsumption relationships* that hold between all the atomic concepts in the terminology.
The efficiency of classification is crucial to the overall performance of descriptive rea-

soners, because it plays an important role in many reasoning activities of these systems. The informal definition of classification as given above suggests that one can classify a concept by computing its subsumption relationships to all other concepts. This idea of *classification through subsumption* is a de facto standard in current descriptive reasoning systems such as CLASSIC [4], KRIS [2], and LOOM [8]. In a preparatory step, all concept descriptions of a given terminology are expanded by replacing defined concepts through their definitions. Subsequently, the expanded descriptions are transformed into a normal form that is required by a specific subsumption algorithm. The classifier is working on the resulting normal-form terminology by invoking the subsumption algorithm when needed.

There is common agreement that the cost for a single subsumption test is less critical to the overall performance of a classifier than the *number* of subsumption tests that are necessary to correctly classify a concept [6]. Optimization techniques therefore try to minimize these tests [3]. A standard classifier traverses the taxonomy both from the top down and the bottom up to find both the immediate predecessors and successors of a concept in the taxonomy. Bottom and top search use positive and negative information about successful or failed tests to infer further subsumption relationships without invoking the costly test. These optimizations rely first of all on the transitivity of the subsumption relation and can reduce the number of explicit tests significantly when the taxonomy is deep, i. e. there are long "vertical chains" of concepts that the standard classifier can prune. If a taxonomy is large, but rather flat with many concepts classified directly between *top* and *bottom*, significant optimizations are much harder to achieve.

To make it possible for PARKA to scale to the extremely large knowledge bases needed or the applications it is being put to, we developed an extremely efficient, parallelizable "structure-matching" algorithm for computing numerous knowledge based inferences in both serial and parallel versions. In formalizing this algorithm, we have realized that the heart of it is a classifier which does exploit optimizations that are different to those used by standard classifiers.

The structural algorithm we are going to present uses optimization techniques that exploit the structure of concepts as it is available in their original and folded representation. Expansion and normalization steps "destroy" the internal structure of concept descriptions and make all available information about existing subsumption relationships vanish that can directly be read off a concept. For example, let us consider the three concepts $a \doteq \forall r.d$, $b \doteq \forall r.f$, and $c \doteq a \sqcap b$. The folded representation of c tells a classifier immediately that a and b are its direct predecessors, while this information is hidden in its expanded and normalized representation $c \doteq \forall r.(d \sqcap f)$. In contrast to the standard classifier, which starts from the top and the bottom of a taxonomy to classify c, our structural algorithm starts its search at the subconcepts a and b mentioned in the description of c, i. e. somewhere inside the taxonomy and then works towards the bottom and the top.

The PARKA classifier consists of two parts: In a preprocessing step, concepts are tested for unsatisfiability or tautology. Here, standard expansion and normalization algorithms are applied to the concept description as in any common approach to classification. In the second step, the structural classifier is activated on the folded terminology.

Its normal-form representation can be discarded, but the knowledge of inconsistent and tautological concepts and subconcepts is made available to the structural algorithm. In the remainder of this paper, we first describe the particular description logic which underlies the PARKA knowledge representation language (Section 3), and then describe PARKA's generative classifier (Section 4). Section 5 states the soundness and completeness of the classifier for two different languages, and Section 6 presents an example of its use. Section 7 presents conclusions and directions for future work, while all proofs of theorems can be found in the Appendix.

3 The Description Logic $\mathcal{TF}$

PARKA uses slightly different languages to represent the knowledge bases in the various applications. The languages differ mainly in the presence or absence of certain concept forming operators. To make the presentation of the classification algorithm more straightforward, we consider a language that covers these representation formalisms as sublanguages. This enables us to define one classification algorithm instead of presenting several versions of the classifier for each sublanguage. The superset of all these languages corresponds to the concept description language $\mathcal{TF}$ defined in [10]. (PARKA also offers an assertional representation formalism, multiple inheritance, transitive closure of roles, role composition, exceptions, and an expressive query language but these are not considered in this paper.) In the following, we very briefly review the syntax and semantics of $\mathcal{TF}$ and define only the notation that we need to precisely describe the classifier.

3.1 Syntax

The basic "building blocks" of the language $\mathcal{TF}$ are *concepts* and *roles*, which denote subsets of the objects of the application domain $\mathcal{D}$ and binary relationships over $\mathcal{D}$ connecting objects, respectively. The concept $\top$ (TOP) denotes the whole domain, while $\bot$ (BOTTOM) denotes the empty set. Concepts are defined intensionally in terms of descriptions that specify the properties which an object must satisfy to belong to that concept.

Definition 1 *Let c and d be syntactic variables for concept descriptions that are built out of atomic concepts denoted by a, let r be a syntactic variable for role names, and n be a non-negative finite integer. The following concept descriptions can be formed:*

a	*(atomic concept)*
$\top$	*(universal concept)*
$\bot$	*(empty concept)*
$c \sqcap d$	*(concept conjunction)*
$\forall r.c$	*(value restriction)*
$\exists_{\geq} nr$	*(*atleast *number restriction)*
$\exists_{\leq} nr$	*(*atmost *number restriction)*

New atomic concepts can be introduced with the help of terminological axioms and terminological specializations.

Definition 2 *Let a be an atomic concept and d be a concept description, then $a \doteq d$ is a terminological axiom.*

Terminological axioms introduce atomic concepts as *defined* concepts by linking the name of the atomic concept to its definition. Terminological specializations introduce atomic concepts or roles as *primitive* concepts or roles, which remain completely undefined in the terminology.

Definition 3 *Let a be an atomic concept, d be a concept description, and r, s be roles, then $a \mathrel{\dot{\leq}} d$ and $r \mathrel{\dot{\leq}} s$ are terminological specializations.*

Definition 4 *A terminology $\mathcal{T}$ is a set of terminological axioms and terminological specializations.*

As usual, we require that a terminology $\mathcal{T}$ is *unique*, which means that each atomic concept and each role occur only once on the left-hand side of either an axiom or a specialization (they are either primitive or defined) and that $\mathcal{T}$ is *cycle-free*, i. e. an atomic concept is not used (directly or indirectly) in its own definition. Roles are restricted to be primitive and can only be introduced through terminological specializations. But in contrast to CLASSIC [4] for example, we allow subroles in the language. The role hierarchy forms a tree structure, since each subrole is restricted to be a subrole of exactly one role.

Definition 5 *An atomic concept c is defined in $\mathcal{T}$ if it occurs on the left-hand side of an axiom in $\mathcal{T}$. It is primitive if it occurs on the left-hand side of a specialization.*

To describe our classification algorithm we have to make a clear distinction between the name of an atomic concept and its definition. When we speak of an atomic concept we always refer to its name as occurring on the left-hand side of an axiom or a specialization. When we speak of the definition of a concept c we mean the right-hand side of the terminological axiom or specialization introducing c.

Definition 6 *Let $c \doteq \sqcap_i c_i$ be a terminological axiom or $c \mathrel{\dot{\leq}} \sqcap_i c_i$ be a terminological specialization in $\mathcal{T}$. $DEF(c)$ is the set of concept descriptions $c_1, c_2, \ldots, c_i$ contained in the right-hand side of the axiom or specialization introducing c.*

As an example, $\forall r.d \in DEF(c)$ denotes that the definition of c contains a value restriction of the form $\forall r.d$ where r is a syntactic variable for a role name and d is a syntactic variable for a concept description.

Definition 7 *A term in $\mathcal{T}$ is either an atomic concept or a primitive role.*

3.2 Semantics

The formal model theoretic semantics of description logics uses the set of objects, the domain $\mathcal{D}$, for the interpretation of concept descriptions. Concepts denote a (sub)set of objects, while each role denotes a set of object pairs. These sets are called *extensions* of concepts and roles.

Definition 8 *An interpretation $\mathcal{I}$ consists of a domain $\mathcal{D}$ and an extension function $\mathcal{E}$ mapping each atomic concept c to a subset $\mathcal{E}[c]$ from $\mathcal{D}$ and each role name r to a binary relation $\mathcal{E}[r]$ over $\mathcal{D}$.*

Definition 9 *Let C be the set of concept symbols and R be the set of role symbols. $\mathcal{D}$ is an arbitrary set and $\mathcal{E}$ an arbitrary function:*

$$\mathcal{E} = \begin{cases} C & \rightarrow & 2^{\mathcal{D}} \\ R & \rightarrow & 2^{\mathcal{D} \times \mathcal{D}} \end{cases}$$

$\mathcal{E}$ is an extension function iff the following equations hold:

$$\begin{array}{rcl} \mathcal{E}[\top] & = & \mathcal{D} \\ \mathcal{E}[\bot] & = & \emptyset \\ \mathcal{E}[c \sqcap d] & = & \mathcal{E}[c] \cap \mathcal{E}[d] \\ \mathcal{E}[\forall\, r.c] & = & \{\alpha \in \mathcal{D} \mid \forall \beta : \langle \alpha, \beta \rangle \in \mathcal{E}[r] \Rightarrow \beta \in \mathcal{E}[c]\} \\ \mathcal{E}[\exists_{\geq} n\, r] & = & \{\alpha \in \mathcal{D} \mid \|\{\beta \in \mathcal{D} | \langle \alpha, \beta \rangle \in \mathcal{E}[r]\}\| \geq n\} \\ \mathcal{E}[\exists_{\leq} n\, r] & = & \{\alpha \in \mathcal{D} \mid \|\{\beta \in \mathcal{D} | \langle \alpha, \beta \rangle \in \mathcal{E}[r]\}\| \leq n\} \end{array}$$

The interpretation of a terminological axiom $c \doteq d$ is the equation $\mathcal{E}[c] = \mathcal{E}[d]$, while the interpretation of a terminological specialization $c \dot{\leq} d$ is the subset relationship $\mathcal{E}[c] \subseteq \mathcal{E}[d]$.

Definition 10 *An extension function $\mathcal{E}$ is a model of a terminology $\mathcal{T}$ iff $\mathcal{E}[c] = \mathcal{E}[d]$ for all terminological axioms $c \doteq d$ and $\mathcal{E}[c] \subseteq \mathcal{E}[d]$ for all terminological specializations $c \dot{\leq} d$ are satisfied in $\mathcal{T}$.*

Definition 11 *A concept c is satisfiable iff it has a nonempty extension $\mathcal{E}[c] \neq \emptyset$ in some model of $\mathcal{T}$.*

In order to prove soundness and completeness of the classifier we will need the notion of subsumption.

Definition 12 *Let $\mathcal{T}$ be a terminology and c, d be terms. d subsumes c in $\mathcal{T}$ ($c \sqsubseteq_{\mathcal{T}} d$) iff $\mathcal{E}[c] \subseteq \mathcal{E}[d]$ holds in all models of $\mathcal{T}$.*

Definition 13 *Two terms t_1 and t_2 are equivalent in $\mathcal{T}$ (written $t_1 \equiv t_2$) if and only if $\mathcal{E}[t_1] = \mathcal{E}[t_2]$ in all interpretations $\mathcal{I} = \langle \mathcal{D}, ext \rangle$ of $\mathcal{T}$.*

The subsumption problem of this language has been proven to be co-NP-hard in [9]. The interaction of the subrole hierarchy, number restrictions, and value restrictions makes the satisfiability problem PSPACE-complete [5, 7]. The availability of subroles and number restrictions allows for the formulation of disjoint concepts, which seems to be of interest for practical applications. Problems in computing subsumption have already been illustrated in [10] (see also Section 4.1), but no solution has been devised until today that would scale to real-world applications and common practice is to live with an "almost" complete algorithm. Currently, all knowledge bases that have been represented in PARKA did not make use of the full expressivity of the language $\mathcal{TF}$, but used one of the following two sublanguages:

1. $\mathcal{TF}^{\ominus}$ provides no subroles, but the whole range of $\mathcal{TF}$-operators $\sqcap$, $\forall$, $\exists_{\geq}$, $\exists_{\leq}$.
2. $\mathcal{TF}^{\star}$ allows to formulate subrole hierarchies (restricted to trees), but provides no $\exists_{\leq}$ operator

The classifier is complete for these sublanguages, but incomplete for $\mathcal{TF}$ because it misses certain subsumption relationships that require to compute the *actual* number restriction of a concept, cf. Section 4.1. The new approach to reasoning about number restrictions over role hierarchies based on diophantine equations that has been developed in [11] would overcome this deficiency, but has not yet been integrated into the classifier.
In discussing classification, we will also need the notion of the *immediate successor* and the *immediate predecessor* of a term in a taxonomy.

Definition 14 *Let c, d be terms for which $c \sqsubseteq_{\mathcal{T}} d$ but $d \not\equiv c$ holds in all models of $\mathcal{T}$. The term c is an immediate successor of d (written $c \prec_{\mathcal{T}} d$) if and only if for all terms b holds: If $b \sqsubseteq_{\mathcal{T}} d$ then either $c \equiv b$ or $c \not\sqsubseteq_{\mathcal{T}} b$. The term d is an immediate predecessor of c (written $d \succ_{\mathcal{T}} c$) if and only if for all terms b holds: If $c \sqsubseteq_{\mathcal{T}} b$ then either $d \equiv b$ or $b \not\sqsubseteq_{\mathcal{T}} d$.*

Again, this treatment has been necessarily brief, and the reader is directed to [10] for further details and discussion.

4 The generative Classifier

Given a terminology consisting of a fixed set of terminological axioms $c \doteq \sqcap_i c_i$ and terminological specializations $c \dot{\leq} \sqcap_i c_i$, the general process of classification proceeds as usual. The classifier starts with the empty taxonomy containing only the *bottom* and *top* concepts and begins to insert primitive concepts $c \dot{\leq} \top$ and roles $r \dot{\leq} \top$. Subsequently, defined concepts that are built out of *already inserted* subconcepts can be classified. We assume that "redefinitions" of concepts are not allowed, i. e. we cannot remove a concept definition and replace it by a new one. In this case, a reclassification of the affected parts of the taxonomy is necessary that we do not address in this paper. Each single classification step proceeds in two main phases in PARKA. The preprocessing phase takes a concept c and computes its normal-form representation $\mathcal{X}(c, \mathcal{T})$.

Then it invokes the tests TAUT and USAT to determine whether c is tautological and unsatisfiable, respectively. After the tests have been performed, $\mathcal{X}(c,\mathcal{T})$ is discarded and the main classification phase returns to the original folded representation of c and analyses the c_i used in its definition. Starting from their position in the taxonomy, successors and predecessors are determined. This leads to the set of *all* (including non-immediate) successors $\ell_{\uparrow c}$ and *all* (also including non-immediate) predecessors $\ell_{\downarrow c}$ of c. A reduction operation removes all non-immediate successors and predecessors from the two sets and with that the sets of of immediate successors SUCC(c) and immediate predecessors PREC(c) of the concept c in the taxonomy are known.

4.1 Testing for Unsatisfiability and Tautologies

In principle, in order to deal with the two special cases $c \equiv \top$ and $c \equiv \bot$ any available satisfiability and tautology checkers, for example methods based on constraint solving techniques [5] or tableaux calculi [12], can be implemented here. In the following, we describe the specific tests used in PARKA.

Normal-Form Representations

The preprocessing begins with the standard transformation of the terminology into a normal form, see for example [10]. Primitive concepts that are introduced through a concept specialization are transformed into defined concepts using auxiliary concepts (usually called primitive components) that remain completely undefined in the terminology, i. e. if $c \stackrel{\cdot}{\leq} d$ then $c \doteq d \sqcap \bar{c}$. Roles do not need to be normalized because we do not have role forming operators in the language and the role hierarchy is thus fix for all concepts in the taxonomy. The resulting terminology contains only defined concepts, primitive components, and primitive roles. Similarly to the definition of an extension function $\mathcal{E}$ and an interpretation $\mathcal{I} = \langle \mathcal{D}, \mathcal{E} \rangle$ for the terminology, an extension function $\mathcal{E}'$ and interpretation $\mathcal{I}' = \langle \mathcal{D}', \mathcal{E}' \rangle$ for the corresponding normal-form terminology can be defined with $\mathcal{E}[t] = \mathcal{E}'[t]$ for any term t in the terminology. For details and a proof we refer again to [10]. Based on the normal-form terminology, all concepts are expanded, i. e. defined concepts are replaced by their definition, which makes all restrictions explicit that a concept inherits.

Definition 15 *The expansion of a concept c in $\mathcal{T}$ is defined as*

$$\mathcal{X}(c,\mathcal{T}) := \begin{cases} \mathcal{X}(d,\mathcal{T}) & \textit{if } c \doteq d \in \mathcal{T} \\ \forall r.\mathcal{X}(d,\mathcal{T}) & \textit{if } c = \forall r.d \\ \mathcal{X}(c_1,\mathcal{T}) \sqcap \mathcal{X}(c_2,\mathcal{T}) & \textit{if } c = c_1 \sqcap c_2 \\ c & \textit{otherwise} \end{cases}$$

The expansion process terminates, since we assume that the terminology is cycle-free. The process is also completely deterministic due to the uniqueness assumption. Expansion is extension-preserving, because replacing atomic concepts by their definition does not change their meaning, i. e. $\mathcal{E}[c] = \mathcal{E}[\mathcal{X}(c,\mathcal{T})]$.

Based on the expansion of a concept c, PARKA computes the concept $\hat{r}_c$ that makes all restrictions of a role r explicit that are inherited over the role hierarchy by c and that combines them with value restrictions of r that are spread over the definition of c.

Definition 16 *Let c be an atomic concept and $\mathcal{X}(c, \mathcal{T})$ its expansion in $\mathcal{T}$. If $\forall\, r.d \in DEF(\mathcal{X}(c, \mathcal{T}))$ then the actual value restriction $\hat{r}_c$ of r in c is obtained as $\hat{r}_c = \sqcap_i f_i \sqcap d$ for all concept descriptions of the form $\forall\, s_i.f_i \in DEF(\mathcal{X}(c, \mathcal{T}))$ where $s_i \in \ell_{\downarrow r}$. The set $\ell_{\downarrow r}$ comprises all predecessor roles of r and is computed as*

$$\ell_{\downarrow r} \;:=\; \{s \,|\, s \succ_{\mathcal{T}} r \textit{ or } s \in \ell_{\downarrow q} \textit{ with } q \succ_{\mathcal{T}} r\} \cup \{r\}$$

The set $\ell_{\downarrow r}$ is obtained by first looking up r in the taxonomy and then following its links to its immediate predecessors s. Then starting from all immediate predecessors s, their predecessors q are collected recursively until $\top$ is reached and the process terminates. When describing the classification algorithm, we will also need the set of all successor roles of r that are collected from all paths starting in r and leading to $\bot$:

$$\ell_{\uparrow r} \;:=\; \{s \,|\, s \prec_{\mathcal{T}} r \;\textit{ or }\; s \in \ell_{\uparrow q} \textit{ with } q \prec_{\mathcal{T}} r\} \cup \{r\}$$

As an example let us consider the terminolog $c \doteq a \sqcap \forall\, r.d$, $a \doteq \forall\, s.f$, $r \dot{\leq} s$, $f \doteq g \sqcap h$. We obtain $\hat{r}_c = d \sqcap g \sqcap h$, which represents the actual value restriction of r in c with respect to $\mathcal{T}$. Replacing the concept description d with $\hat{r}_c$ in all occurrences of value restrictions on r of form $\forall\, r.d$ is an extension-preserving transformation.

Tautologies in $\mathcal{TF}$

In general, tautologies can be formulated using disjunction and negation as in $c \sqcup \neg c$. Disjunction alone allows for the formulation of a tautology as $\top \sqcup \bot$, i. e. joining a tautology with an unsatisfiable concept. The lack of disjunction and negation in $\mathcal{TF}$ makes it impossible to formulate tautologies in this way and only two syntactical forms of concept descriptions can lead to tautologies: $c = \forall\, r.\top$ and $c = \exists_{\geq} 0\, r$.

Definition 17 *Let $\mathcal{X}(c, \mathcal{T})$ be the expanded normal-form representation of the concept c. We define the recursive function $TAUT(c)$ as follows: $TAUT(c)$ is true if and only if one of the following conditions holds*

1. $\mathcal{X}(c, \mathcal{T}) = \top$
2. $\mathcal{X}(c, \mathcal{T}) = \exists_{\geq} 0\, r$
3. $\mathcal{X}(c, \mathcal{T}) = \forall\, r.d$ *with* $TAUT(\hat{r}_c) =$ TRUE
4. $\mathcal{X}(c, \mathcal{T}) = a \sqcap b$ *with* $TAUT(a) =$ TRUE *and* $TAUT(b) =$ TRUE

Proposition 1 *The test TAUT(c) for c given in either $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$ delivers* TRUE *if and only if $\mathcal{E}[c] = \mathcal{D}$ in all models of $\mathcal{T}$.*

All proofs of propositions and theorems can be found in the Appendix.

Unsatisfiability in $\mathcal{TF}$

Although the language $\mathcal{TF}$ contains no negation, unsatisfiability of concepts can be caused by contradicting value and number restrictions.

- $\forall r.\bot \sqcap \exists_{\geq} 1\, r$ is unsatisfiable since the value of a role is restricted to the empty set in the value restriction, while the number restriction requires the existence of atleast one role filler.
- $\exists_{\leq} 3r \sqcap \exists_{\geq} 2\, s \sqcap \exists_{\geq} 2\, t$ is unsatisfiable if s and t are subroles of r *and* the sets of role fillers for s and t are disjoint.

Disjointness of role fillers can be formulated in $\mathcal{TF}$ by using contradicting number restrictions as value restricting concepts. The following examples illustrate problems that reveal how difficult it can be to discover the unsatisfiability of a concept.

Example 1 *Given is the following set of concept definitions*

$$\begin{aligned} \mathsf{B} &\doteq \mathsf{B1} \sqcap \mathsf{B2} \\ \mathsf{B1} &\doteq \forall \mathsf{r}.(\exists_{\leq} 1\, \mathsf{s}) \\ \mathsf{B2} &\doteq \forall \mathsf{r}.(\exists_{\geq} 2\, \mathsf{s}) \end{aligned}$$

with s *and* r *arbitrary roles. The concepts* B1 *and* B2 *are satisfiable when considered in isolation, but the conjunction* B *leads to an interaction between the roles, i. e. we have*

$$\mathsf{B} \equiv \forall \mathsf{r}.(\exists_{\leq} 1\, \mathsf{s} \sqcap \exists_{\geq} 2\, \mathsf{s}) \equiv \forall \mathsf{r}.\bot$$

where the value restricting concept is unsatisfiable. Therefore B *would be a successor of any concept* $\mathsf{A} \doteq \forall \mathsf{r}.x$ *since* $\bot$ *is a successor of any concept.*

In general, the inheritance of restrictions over the role and concept hierarchies can lead to contradictions that are much trickier to detect. The subsequent example illustrates a simple case caused by disjoint value restrictions on subroles.

Example 2 *The concept* P *introduces a person who has exactly two children, where atleast two of them are girls and one of them is a boy.*

$$\begin{aligned} \mathsf{P} \doteq\ & \exists_{\geq} 2\, kids \sqcap \exists_{\leq} 2\, kids \sqcap \exists_{\geq} 2\, girl \sqcap \exists_{\geq} 1\, boy \\ & \sqcap \forall girl.\exists_{\geq} 2\, xchrom \sqcap \forall boy.\exists_{\leq} 1\, xchrom \end{aligned}$$

The roles girl *and* boy *are subroles of the* kid*-role, i. e.* $girl \mathrel{\dot{\leq}} kids$ *and* $boy \mathrel{\dot{\leq}} kids$ *reflecting the fact that each girl or boy is also a child. The fillers for the girls and boys roles are defined as being disjoint by stating a contradicting property for them using the has-x-chromosomes role. Since the sets of girls and boys are disjoint, it can be concluded that* P *must have atleast three children, which contradicts that* P *has atmost two children, i. e.* P *has an empty extension. When the requirement* $\exists_{\leq} 2\, kids$ *is absent, i. e. we only know that* P *has atleast two children, it follows immediately that* $\exists_{\geq} 3\, kids$ *must hold. Thus* P *would be subsumed by any concept description containing* $\exists_{\geq} 3\, kids$.

In the following, we describe the unsatisfiability test used by PARKA. This test is incomplete for the language $\mathcal{TF}$, but complete for the sublanguages.

Definition 18 *Let $\mathcal{X}(c, \mathcal{T})$ be the expanded normal-form representation of the concept c. We define the recursive function $USAT(c)$ as follows: $USAT(c) =$ TRUE if and only if one of the following conditions holds*

1. $\mathcal{X}(c, \mathcal{T}) = \bot$
2. $\mathcal{X}(c, \mathcal{T}) = a \sqcap b$ *with* $USAT(a) =$ TRUE *<u>or</u>* $USAT(b) =$ TRUE
3. $\mathcal{X}(c, \mathcal{T}) = \forall\, r.d \sqcap \exists_{\geq} n\, s \sqcap c'$ *with* $USAT(\hat{r}_c) =$ TRUE *and* $n > 0$ *and* $s \in \ell_{\uparrow r}$
4. $\mathcal{X}(c, \mathcal{T}) = \exists_{\geq} n\, r \sqcap \exists_{\leq} k\, s \sqcap c'$ *with* $s \in \ell_{\downarrow r}$ *and* $n > k$.

Proposition 2 *Let c be a concept description given either in $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$ then $USAT(c) =$ TRUE if and only if $\mathcal{E}[c] = \emptyset$ in all models of $\mathcal{T}$.*

For the special cases TAUT(c) = TRUE or USAT(c) = TRUE no further classification steps are necessary, since c inherits successors and predecessors of $\top$ and $\bot$, respectively. If c is neither tautological nor unsatisfiable, USAT and TAUT forward a list of concepts $\hat{r}_c$ representing the actual value restrictions $\hat{r}_c$ of each role r contained in c to the classifier. If $\hat{r}_c$ is tautological it is simplified to $\hat{r}_c = \top$, if it is unsatisfiable it is simplified to $\hat{r}_c = \bot$. This knowledge is used during various steps of the classification process and decides which successors or predecessors are generated. Other expansions are no longer needed since the classifier works with the folded concept description.

4.2 Generating Immediate Successors

The basic idea for the generation of successors for a concept c, is to split c into its defining conjunctive elements c_i and find concepts y that contain conjunctive elements y_i such that for all c_i there is a y_i which is a successor of it. In this case, y is a successor of c even if it contains or inherits concept descriptions x. As an example, let us consider $c \doteq c_1 \sqcap c_2 \sqcap c_3$ and $y \doteq y_1 \sqcap y_2 \sqcap x$. If $y_1 \sqsubseteq c_1$, $y_2 \sqsubseteq c_2$, and $y_2 \sqsubseteq c_3$ holds, then $y_1 \sqcap y_2 \sqsubseteq c_1 \sqcap c_2 \sqcap c_3$ is a valid conclusion and of course, $y_1 \sqcap y_2 \sqcap x \sqsubseteq c_1 \sqcap c_2 \sqcap c_3$ follows for arbitrary concepts x, i. e. $y \sqcap c$. This means that *local* tests on y_i and c_i are sufficient for y to be a successor of c.
In the following, x and y denote atomic concept names contained in the taxonomy, while c and c_i always refer to the current concept to be classified and its subconcepts. Given a new axiom or a new specialization $c \doteq \sqcap_i c_i$ or $c \stackrel{\cdot}{\leq} \sqcap_i c_i$, the classifier deals with two special cases first:

- If $c \stackrel{\cdot}{\leq} \sqcap_i c_i$ has to be classified, the algorithm immediately returns SUCC(c) $= \{\bot\}$. The justification for this result becomes apparent when we consider the expanded normal form of c, which is $\mathcal{X}(c, \mathcal{T}) = \mathcal{X}(\sqcap_i c_i, \mathcal{T}) \sqcap \bar{c}$. Note that c is a new atomic concept name not contained in the taxonomy and that $\bar{c}$ remains completely undefined. Therefore there can be no concept y that contains $\bar{c}$ and with that, no y_i can be a successor of $\bar{c}$. Consequently, only *bottom* as a successor of any concept remains in the solution set. Let us consider the following example, where $\mathcal{T}$ comprises the two concepts $\mathsf{man} \stackrel{\cdot}{\leq} \mathsf{person}$ and $\mathsf{father} = \mathsf{man} \sqcap \exists_{\geq} 1\, \mathsf{child}$. Obviously, father is a successor of man, but the algorithm does not return it the reason being that man has to be classified *before* father can be inserted into the taxonomy to make all subconcepts of father known to the system. This means, when man is classified, no concept can exist in the taxonomy that uses it directly or indirectly in its definition. Of course, when classifying father later on, man is found as a predecessor of it.

- If $c \doteq c_1$ has to be classified and c_1 is atomic, then c and c_1 are equivalent and c inherits all successors of the node c_1 in the taxonomy.

```
input: $c \doteq \sqcap_i c_i$ or $c \stackrel{\cdot}{\leq} \sqcap_i c_i$
preprocessing:
    $TAUT(c) \rightarrow \{\text{TRUE}, \text{FALSE}\}$
    $USAT(c) \rightarrow \{\text{TRUE}, \text{FALSE}\}$
    $\mathcal{X}(c, \mathcal{T}) \rightarrow \hat{r}_c$ for any value restriction $\forall r.d$ in $\mathcal{X}(c, \mathcal{T})$
SUCC:   if $c \doteq \sqcap_i c_i$
            then if $i = 1\,(c \doteq c_1, c_1$ atomic$)$ then $SUCC(c) := SUCC(c_1)$
                 if $TAUT(c) = \text{TRUE}$       then $SUCC(c) := \{\top\}$
                 if $USAT(c) = \text{TRUE}$       then $SUCC(c) := \{\bot\}$
                                                  else compute $\ell_{\uparrow c}$
                                                       reduce $\ell_{\uparrow c}$ to SUCC(c)
            else $(c \stackrel{\cdot}{\leq} \sqcap_i c_i)$
                 if $USAT(\sqcap_i c_i) = \text{TRUE}$   then ERROR ($c \stackrel{\cdot}{\leq} \bot$ not admitted)
                                                  else $SUCC(c) := \{\bot\}$
```

Figure 1: The simple Cases for SUCC

Figure 1 summarizes the simple cases for the SUCC algorithm. Note that the concepts $\top$ and $\bot$ represent also all concepts y that are equivalent to $\top$ (the test $TAUT(y)$ delivered TRUE) and $\bot$ (the test $USAT(y)$ delivered TRUE), respectively. The classifier returns only $\top$ and $\bot$ as the representatives of these classes of equivalent concepts.
The main and much more elaborate part of SUCC is the computation of the set $\ell_{\uparrow c}$ that is necessary for all c that are defined as a single non-atomic c_i or as a "true" concept conjunction. For each $c_i \in DEF(c)$ four possible cases are distinguished for which separate generation steps are activated: c_i can be (1) an atomic concept, (2) a value restriction, (3) an atleast number restriction, or (4) an atmost number restriction.

Finding $\ell_{\uparrow c}$

Case 1: c_i is atomic

Since c_i is either a primitive or defined concept and thus already contained in the taxonomy its immediate successors can easily be collected by moving down the taxonomy. Note that we add c_i to the set $\ell_{\uparrow c_i}$ contains c_i.

$$\ell_{\uparrow c_i} \;:=\; \{y \,|\, y \prec_{\mathcal{T}} c_i\} \,\cup\, \{c_i\} \tag{1}$$

`;; ` c_i `and the set of all immediate successors of` c_i

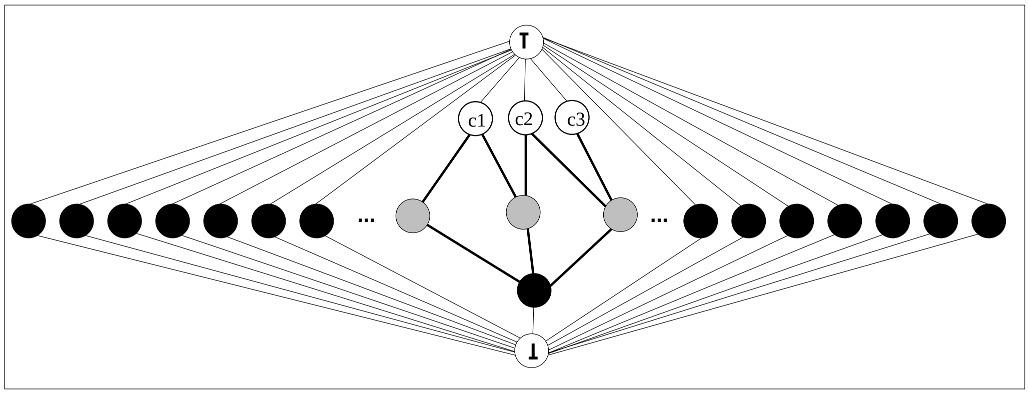

Figure 2: Search Space during Successor Generation

Figure 2 illustrates the search performed by the classifier during Step 1. Let us assume that a concept $c = c_1 \sqcap c_2 \sqcap c_3$ has to be classified where each c_i is an atomic concept, i. e. a node in the taxonomy. The structural classifier starts at these nodes and adds their immediate successors (grey nodes) by following all possible paths towards the bottom concept. The only concept in the taxonomy (besides $\perp$) that is a valid successor of all three c_i is the black node in the middle of the taxonomy that can be reached from each of the grey nodes. A standard classifier when searching successors in its bottom phase as described in [3] tests all black nodes first and discards all but the middle one. Then the grey nodes are tested and since they are not a successor of c, it concludes that c must be placed above the middle black node. For a taxonomy that is rather flat than deep, the structural approach seems to be of advantage.

Case 2: $c_i = \forall r.d$

The classifier accesses the actual value restriction of r in c as represented in $\hat{r}_c$ and proceeds depending on whether $\hat{r}_c$ is a tautology or not. Remember that it has been simplified to $\top$ in this case during preprocessing.

If $\hat{r}_c = \top$

$$\ell_{\uparrow c_i} \;:=\; \{\top\} \tag{2}$$

`;; the special case` $\top \sqsubseteq \forall r.\top$

If $\hat{r}_c \neq \top$

$$\ell_{\uparrow c_i} \quad := \quad \begin{array}{l} \{y \mid \forall s.f \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } \hat{s}_y = \bot\,\} \cup \\ \texttt{;; the special case } r \sqsubseteq s \Rightarrow \forall s.\bot \sqsubseteq \forall r.d \\ \{y \mid \forall s.f \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } \hat{s}_y \in \ell_{\uparrow \hat{r}_c}\,\} \cup \\ \texttt{;; the rule } r \sqsubseteq s \wedge f \sqsubseteq d \Rightarrow \forall s.f \sqsubseteq \forall r.d \texttt{ is satisfied} \\ \{y \mid \exists_{\leq} 0\, s \in DEF(y) \text{ and } s \in \ell_{\downarrow r}\,\} \\ \texttt{;; the rule } r \sqsubseteq s \Rightarrow \exists_{\leq} 0\, s \sqsubseteq \forall r.d \texttt{ is satisfied} \end{array} \qquad (3)$$

The rules require to test all concepts from the taxonomy that mention value restrictions on predecessor roles of r explicitely in their definition. To find them with reasonable effort, the name of each concept with such a value restriction is stored in a hash-table relating concept names to role names. Note that we do not consider concepts that inherit value restrictions over the taxonomy in this generation step, but they are added later.
The second generation step states the condition $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$ according to the underlying subsumption rule. This condition requires to return to the expanded normal-form representation of the actual value restrictions for s in y and r in c (as computed during preprocessing according to Definition 16) in order to achieve soundness and completeness.* Since neither $\hat{s}_y$ nor $\hat{r}_c$ need to be contained in the taxonomy, this test leads to additional classification tasks. The special case, $\hat{r}_c = \bot$ is simple, since it requires that $\hat{s}_y = \bot$ holds for the test to be successful. In all other cases, the classifier needs to access $\hat{r}_c = \sqcap_i d_i$ and compute the sets of all successors for all d_i. Note that $\hat{r}_c$ is a concept given in the fully expanded normal-form representation and therefore each d_i is either primitive or non-atomic. For each d_i the set $\ell_{\uparrow d_i}$ is computed depending on the syntactic structure of d_i:

- d_i is primitive. The set $\ell_{\uparrow d_i}$ can be directly read off the taxonomy by following all paths from d_i to $\bot$.
- $d_i = \exists_{\geq} n\, r$ or $d_i = \exists_{\leq} n\, r$. An auxiliary concept $c' \doteq d_i$ is classified and $\ell_{\uparrow d_i} := \ell_{\uparrow c'}$.
- $d_i = \forall r'.e$. A recursive subcall of the algorithm with the actual value restriction of r' as input is activated to determine the successors for d_i.

The resulting set of concepts $\mathcal{Y}_i$ contains the successors with respect to each single d_i.

$$\mathcal{Y}_i \quad := \quad \{y \mid \forall s.f \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } \hat{s}_y \in \ell_{\uparrow d_i}\,\} \qquad (4)$$

To test whether the condition $\hat{s}_y \in \ell_{\uparrow d_i}$ is satisfied, either $\hat{s}_y$ can be classified to test whether d_i is a predecessor or the classifier can verify whether $\hat{s}_y = \sqcap_j f_j$ contains a f_j that is contained in $\ell_{\uparrow d_i}$. For example, if d_i is primitive, it must be contained in

*In fact, using the completely expanded normal form means to ignore a very important potential for optimization of the structural algorithm and is not always necessary to be sound and complete. For example in $c \doteq \forall r.d$ with d being a defined concept, the algorithm can immediately work with d because it is a node in the taxonomy instead of returning to the normal form of a possibly very longwinded concept definition. Further work is necessary to make use of this optimization potential.

$\hat{s}_y$, i. e. there must be a $f_j = d_i$. This test is often simpler than classification of $\hat{s}_y$, but the condition is only sufficient, not necessary, i. e. if $f \in \ell_{\uparrow d_i}$ succeeds $\hat{s}_y \in \ell_{\uparrow d_i}$ can be concluded, but if the test fails, $\hat{s}_y \notin \ell_{\uparrow d_i}$ cannot be concluded and the original test requiring the classification of $\hat{s}_y$ has to be performed. The next step completes the sets $\mathcal{Y}_i$ by adding all successors for each concept contained in the set, i. e. concepts inheriting successor value restrictions are now added:

$$\ell_{\uparrow \mathcal{Y}_i} := \mathcal{Y}_i \cup \{x \mid \forall y \in \mathcal{Y}_i : x \in \ell_{\uparrow y}\} \qquad (5)$$

;; for all y all their successors x are added

Finally, the intersection of the sets $\ell_{\uparrow \mathcal{Y}_i}$ is computed that preserves all concepts that are successors of each d_i and with that of $\hat{r}_c$. This leads to the final set $\ell_{\uparrow c_i}$:

$$\ell_{\uparrow c_i} := \bigcap_i \ell_{\uparrow \mathcal{Y}_i} \qquad (6)$$

The following example illustrates the generation of successors for the second generation step:

Example 3 *Let us assume that the classifier has to find successors of the concept* $c_i = \forall\, r.[a \sqcap b]$ *with* a, b *primitive and that the taxonomy contains the concepts* $y \doteq c_1 \sqcap c_2$, $c_1 \doteq \forall\, r.a$, *and* $c_2 \doteq \forall\, r.b$. *The classifier determines* $\hat{r}_c = a \sqcap b$ *and proceeds according to Rule (3) because of* $\hat{r}_c \neq \top$. *The first and third generation steps return the empty set, because the taxonomy contains no concepts with unsatisfiable value or number restrictions.*
The second generation step sets $d_1 = a$ *and* $d_2 = b$ *and begins to compute all their successors as* $\{a, \bot\}$ *and* $\{b, \bot\}$, *which involves a call to the classifier for each* d_i. *Now the sets* $\mathcal{Y}_1 = \{c_1\}$ *and* $\mathcal{Y}_2 = \{c_2\}$ *are generated. The concepts* c_1 *and* c_2 *satisfy the tests, because the expanded normal-form representations of their value restricting concepts* a *and* b *are successors of the respective* d_i. *The sets are completed by adding all successors of* c_1 *and* c_2 *to them:*

$$\ell_{\uparrow \mathcal{Y}_1} := \{c_1, y, \bot\}$$

$$\ell_{\uparrow \mathcal{Y}_2} := \{c_2, y, \bot\}$$

The intersection of both sets returns the successors of c_i *in the taxonomy as* $\ell_{\uparrow c_i} := \{y, \bot\}$.

Case 3: $c_i = \exists_{\geq} n\, r$

Generation proceeds depending on the value of n.

If $n = 0$

$$\ell_{\uparrow c_i} := \{\top\} \qquad (7)$$

;; the special case $\top \sqsubseteq \exists_{\geq} 0\, r$

If $n \geq 1$

$$\ell_{\uparrow c_i} \;:=\; \{ y \,|\, \exists_{\geq} k\, s \in DEF(y) \text{ and } s \in \ell_{\uparrow r} \text{ and } k \geq n \} \tag{8}$$

`;; the rule` $s \sqsubseteq r \wedge k \geq n \Rightarrow \exists_{\geq} k\, s \sqsubseteq \exists_{\geq} n\, r$ `is satisfied`

Case 4: $c_i = \exists_{\leq} n\, r$

$$\ell_{\uparrow c_i} \;:=\; \begin{array}{l} \{ y \,|\, \forall s.f \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } \hat{s}_y = \bot \} \,\cup \\ \texttt{;; the rule } r \sqsubseteq s \Rightarrow \forall s.\bot \sqsubseteq \exists_{\leq} n\, r \texttt{ is satisfied} \\ \{ y \,|\, \exists_{\leq} k\, s \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } k \leq n \} \\ \texttt{;; the rule } r \sqsubseteq s \wedge k \leq n \Rightarrow \exists_{\leq} k\, s \sqsubseteq \exists_{\leq} n\, r \texttt{ is satisfied} \end{array} \tag{9}$$

After all sets $\ell_{\uparrow c_i}$ have been generated for all c_i, the classifier still does not know all successors with respect to the single c_i. In particular, the concept $\bot$ that is a successor of any concept has been ignored in all generation steps. Furthermore, a set $\ell_{\uparrow c_i}$ can be empty if none of the concepts in the taxonomy satisfied the tests required in the various structural rules. In this case, $\bot$ is added. All non-empty sets need to be extended by adding for all y all their successors x. This situation was already illustrated in Figure 2. In the current stage of the classification process, each $\ell_{\uparrow c_i}$ contains only the grey nodes, but we are interested in finding the black node that is a successor of them. Therefore, the classifier continues to follow the $\prec_{\mathcal{T}}$ links from each y until bottom is reached.

For all $\ell_{\uparrow c_i}$

$$\begin{array}{lll} \textbf{if } \ell_{\uparrow c_i} = \emptyset & \textbf{then} & \ell_{\uparrow c_i} := \{\bot\} \\ & & \texttt{;; the rule } \bot \sqsubseteq c \texttt{ is satisfied} \\ & \textbf{else} & \ell_{\uparrow c_i} := \ell_{\uparrow c_i} \,\cup\, \{ x \,|\, \forall y \in \ell_{\uparrow c_i} : x \in \ell_{\uparrow y} \} \\ & & \texttt{;; for all } y \texttt{ add all their successors } x \texttt{ including } \bot \end{array} \tag{10}$$

Finally, the classifier computes the intersection of the sets $\ell_{\uparrow c_i}$ to determine those concepts that are successors of all c_i and with that of c:

$$\ell_{\uparrow c} \;:=\; \{ \bigcap_i \ell_{\uparrow c_i} \} \tag{11}$$

`;; the rule` $y \sqsubseteq c_1 \wedge y \sqsubseteq c_2 \Rightarrow y \sqsubseteq c_1 \sqcap c_2$ `is satisfied`

With $\ell_{\uparrow c}$ the classifier has determined the set of all successors of c in the taxonomy. If $\ell_{\uparrow c}$ is a singleton, we are done. Otherwise, the set can comprise immediate and non-immediate successors. In order to eliminate all non-immediate successors from $\ell_{\uparrow c}$ each element is analysed whether it is a successor of another element in the set. For all elements y in $\ell_{\uparrow c}$ we know that $y \sqsubseteq c$ and if we find an element x with $y \sqsubseteq x$ that is not equivalent to y, it follows that y cannot be an immediate successor according to Definition 14.

Definition 19 *The set of immediate successors of a concept c is the set SUCC(c) with*

$$SUCC(c) := \begin{cases} \ell_{\uparrow c} & \textit{if } \ell_{\uparrow c} \textit{ singleton} \\ \{ y \,|\, y \in \ell_{\uparrow c} \textit{ and } \forall x \in \ell_{\uparrow c} \textit{ with } y \not\equiv x : y \notin \ell_{\uparrow x} \} & \textit{otherwise} \end{cases}$$

4.3 Generating Immediate Predecessors

One might argue that the generation of predecessors for c proceeds symmetric to the generation of successors, but this is not the case. There are two main differences: First, the local tests performed on the various y_i during successor generation are no longer sufficient. When a concept y satisfied a structural rule for one of its y_i, then it was indeed a valid successor of a c_i since $y_i \sqsubseteq c_i$ implies $y \sqsubseteq c_i$. For predecessors, this is not automatically true. Again, we split c into its conjunctive elements c_i and if we find a concept y which is a predecessor of c_i then y will also be a predecessor of c. The problem is that we cannot consider conjunctive elements y_i in isolation, since $c \sqsubseteq y_1$ usually does not imply $c \sqsubseteq y_1 \sqcap y_2$ unless some global criterion is satisfied by *all* y_i.

Example 4 *Let us consider the concepts* $c \doteq a \sqcap \exists_{\geq} 4\, r$ *and the concepts* $y_1 \doteq \exists_{\geq} 4\, r$ *and* $y_2 \doteq b \sqcap \exists_{\geq} 4\, r$*. When analysing* c *the subconcept* $c_i = \exists_{\geq} 4\, r$ *is isolated and we look for concepts that are predecessors of it. Both* y_1 *and* y_2 *contain* $\exists_{\geq} 4\, r$ *in their definition, but only* y_1 *is a valid predecessor, while* y_2 *is not because of* $c \not\sqsubseteq b$*.*

As the example suggests, any candidate predecessor has to meet additional requirements concerning its own predecessors in order to be a valid predecessor of c.
The second difference is that the classifier has to "switch" search directions. During successor generation, search starts in the c_i and proceeds downwards towards the bottom concept, i. e. only successors are considered. Now search cannot be restricted to predecessors alone, but the classifier has to take into consideration *successors* of predecessors, since they can be predecessors of c as well. We call these concepts *indirect predecessors.*

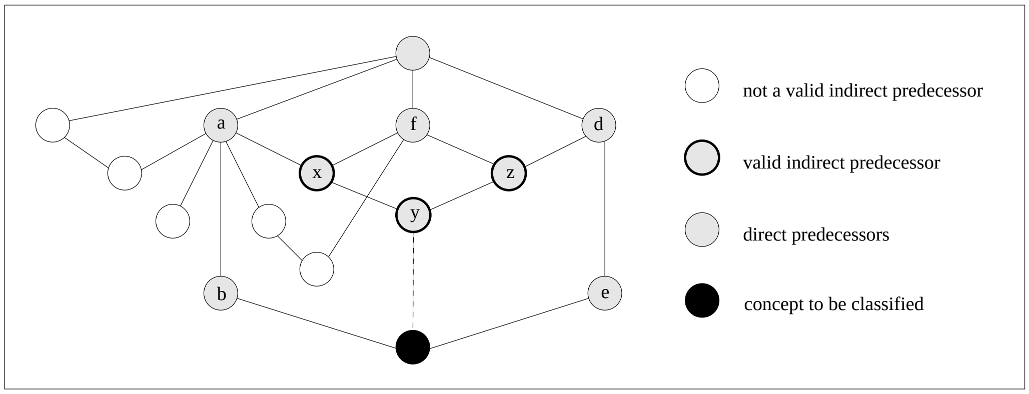

Figure 3: Predecessors and Non-Predecessors of a Concept

Example 5 *Let us assume that* $c \doteq b \sqcap e \sqcap f$ *and that* $b \dot{\leq} a$, $x \doteq a \sqcap f$ *and* $e \dot{\leq} d$, $z \doteq d \sqcap f$ *are contained in* $\mathcal{T}$*. A concept* $y \doteq x \sqcap z$ *is a predecessor of* c*, but it is neither a predecessor of* b *nor a predecessor of* e*, i. e. the classifier would never find it when only considering predecessors of* c_i*. Instead,* y *is a successor of the predecessors* a *and* d*. Figure 3 helps illustrate problems that are related to finding indirect predecessors.*

Given a concept $c \doteq \sqcap_i c_i$ or $c \dot{\leq} \sqcap_i c_i$, two special cases are again treated separately. If $c \doteq c_1$ (c_1 atomic) then c inherits all immediate predecessors from c_1, because both concepts are equivalent. If $c \dot{\leq} c_1$ (c_1 atomic) then c_1 is the only immediate predecessor of c. Figure 4 summarizes the simple cases for the predecessor-generation algorithm.

```
input: $c \doteq \sqcap_i c_i$ or $c \dot{\leq} \sqcap_i c_i$
preprocessing:
    $TAUT(c) \rightarrow \{\text{TRUE}, \text{FALSE}\}$
    $USAT(c) \rightarrow \{\text{TRUE}, \text{FALSE}\}$
    $\mathcal{X}(c,\mathcal{T}) \rightarrow \hat{r}_c$ for any value restriction $\forall r.d$ in $\mathcal{X}(c,\mathcal{T})$

PREC:  if $c \doteq \sqcap_i c_i$
           then if $i = 1\,(c \doteq c_1, c_1$ atomic$)$ then $PREC(c) := PREC(c_1)$
                if $TAUT(c) = \text{TRUE}$              then $PREC(c) := \{\top\}$
                if $USAT(c) = \text{TRUE}$              then $PREC(c) := \{\bot\}$
                                                        else compute $\ell_{\downarrow c}$
                                                             reduce $\ell_{\downarrow c}$ to $PREC(c)$
           else $(c \dot{\leq} \sqcap_i c_i)$
                if $i = 1\,(c \doteq c_1, c_1$ atomic$)$ then $PREC(c) := \{c_1\}$
                if $USAT(\sqcap_i c_i) = \text{TRUE}$   then ERROR ($c \dot{\leq} \bot$ not admitted)
                if $TAUT(\sqcap_i c_i) = \text{TRUE}$   then $PREC(c) := \{\top\}$
                                                        else compute $\ell_{\downarrow c}$
```

Figure 4: The simple Cases for PREC

For all other cases, we distinguish again the four syntactic possibilities for a c_i. Concept specializations $c \dot{\leq} \sqcap_i c_i$ are treated identically to concept definitions $c \doteq \sqcap_i c_i$.

Finding $\ell_{\downarrow c}$

Case 1: c_i is atomic

As a first step, the classifier generates the set of "direct" predecessors for each of the atomic c_i. All atomic c_i are nodes in the taxonomy, i. e. starting at these nodes the classifier moves upward following their explicit links to their immediate and non-immediate predecessors.

$$\ell_{\downarrow c_i} \quad := \quad \begin{array}{l} \{y \,|\, y \succ_{\mathcal{T}} c_i\} \;\cup\; \{c_i\} \cup \\ \texttt{;; } c_i \texttt{ and the set of all immediate predecessors of } c_i \\ \{x \,|\, \forall y \text{ with } y \succ_{\mathcal{T}} c_i : x \in \ell_{\downarrow y} \text{ and } TAUT(x) = \text{FALSE}\} \\ \texttt{;; for all } y \texttt{ all their predecessors } x \texttt{ (except } \top\texttt{) are added} \end{array} \qquad (12)$$

The concept *top* and all concepts that are equivalent to it are ignored in this generation step. *Top* is a predecessor of any concept and is added at the end of the generation. The set of direct predecessors $\wp^0_{\downarrow c}$ of c is collected as the union of the sets generated for the individual c_i.

$$\wp^0_{\downarrow c} \ := \ \bigcup_i \ell_{\downarrow c_i} \tag{13}$$

Any element of $\wp^0_{\downarrow c}$ is guaranteed to be a valid predecessor of it, since the links in the taxonomy guarantee that $c_i \sqsubseteq y$, which implies $c \sqsubseteq y$. Direct predecessors provide therefore a set of "safe seeds" for the subsequent generation of further predecessors. Note that each c_i is also contained in $\wp^0_{\downarrow c}$, since a concept is its own predecessor.
In a second phase, the classifier generates the set of "indirect" predecessors of c by analysing successors of direct predecessors. Starting at concepts contained in $\wp^0_{\downarrow c}$ it moves downward through the taxonomy towards the *bottom* concept. The generation proceeds recursively by first analysing *all* immediate successors of *all* direct predecessors. Any immediate successor satisfying the requirements of Definition 21 given below is added to $\wp^0_{\downarrow c}$ leading to a set $\wp^1_{\downarrow c}$. For this extended set, again all immediate successors of its elements are investigated leading to a set $\wp^2_{\downarrow c}$ and so on, until *bottom* is reached or no further valid indirect predecessors are found, i. e. $\wp^n_{\downarrow c} = \wp^{n-1}_{\downarrow c}$ and a fixed point of the predecessor generation is reached.[†]

Definition 20 *The set $\wp^n_{\downarrow c}$ of indirect predecessors of c in the taxonomy is the set of concepts x*

$$\wp^n_{\downarrow c} \ := \ \{x \,|\, \forall y \in \wp^{n-1}_{\downarrow c} : x \in SUCC(y)\} \tag{14}$$

`;; for all` y `all their immediate successors` x `are analysed`

where each x is covered *by c as defined in Definition 21.*

Definition 21 *A concept x is covered by a concept c if and only if*

1. $x \doteq \sqcap_i x_i \in \mathcal{T}$
   The concept x is restricted to be defined, i. e. it is introduced into $\mathcal{T}$ with the help of a terminological axiom. No concept specializations need to be considered.

2. $\forall x_i \in DEF(x) : TAUT(x_i) = \text{TRUE}$ *or* $x_i \in \ell_{\downarrow c}$
   Any x_i contained in the definition of x must be a valid predecessor of c.

The condition $x_i \in \ell_{\downarrow c}$ is tested differently depending on the syntactic structure of x_i.

- If x_i is atomic, the classifier tests whether x_i is covered.
  In many cases, a simple look up $x_i \in \bigcup_{i=0}^{n-1} \wp^i_{\downarrow c}$ will verify the covering condition for x_i. If x_i is not yet generated as a predecessor, then a recursive call of the generation process is necessary to verify the covering condition for it. Since the terminology is cycle-free the process is guaranteed to terminate.

[†]At this point, it becomes also obvious why we did not add $\top$ to the set of direct predecessors. This would blow up the search space to the entire taxonomy if all successors of $\top$ had to be analysed.

- If x_i is non-atomic, the classifier tests $\forall\, x_i \in DEF(x) : \exists\, c_i \in DEF(c)$ with $x_i \in \ell_{\downarrow c_i}$.
  Any x_i of the syntactic structure $\forall\, r.d$, $\exists_{\geq} n\, r$, and $\exists_{\leq} n\, r$ contained in x must also be a valid predecessor of c. This condition is verified using the structural rules 16 to 21 defined below and testing the existence of a c_i for which the x_i satisfy at least one of the rules.

No concept specialization can be a predecessor unless it is a direct predecessor of c that can be reached when the classifier follows the $\succ_{\mathcal{T}}$ relations. If x is introduced through a concept specialization then its expanded normal form is $\mathcal{X}(x, \mathcal{T}) = \mathcal{X}(\sqcap_i x_i, \mathcal{T}) \sqcap \bar{x}$. Since $\bar{x}$ remains undefined, x can only be a predecessor of c if x is contained in the definition of c (i. e. x must be an atomic c_i) or if x is inherited by one of the c_i, i. e. x must be a predecessor of at least one c_i. In both cases, x must be a direct predecessor. Figure 5 illustrates the general line of inference for the generation of indirect predecessors. The classifier knows of a y subsuming c and of y subsuming x. The conclusion that x must subsume c is valid if and only if all concepts a subsuming x are also subsuming c.

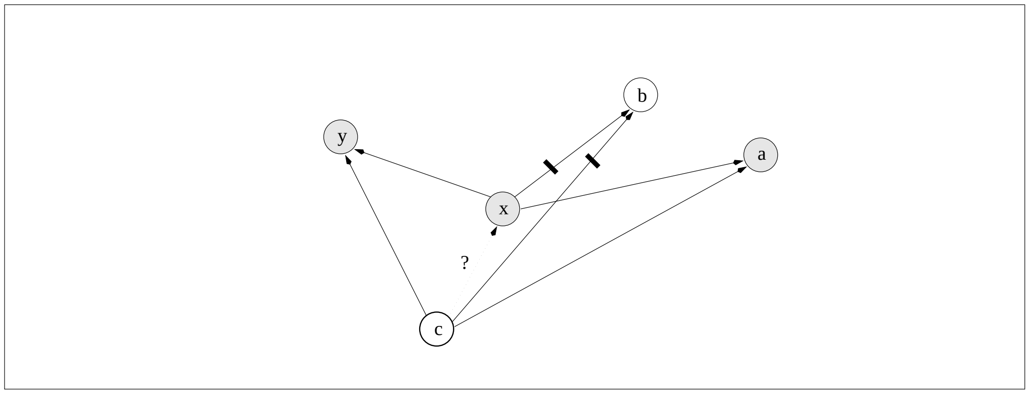

Figure 5: Necessary and Sufficient Criteria for Indirect Predecessors

The set of indirect predecessors of c, which also comprises the set of direct predecessors as the base case of the recursion, is obtained as the union of the sets $\wp^i_{\downarrow c}$

$$\wp_{\downarrow c} \quad := \quad \bigcup_{i=0}^{n} \wp^i_{\downarrow c} \tag{15}$$

In the third phase of the predecessor generation, the classifier analyses the non-atomic $c_i \in DEF(c)$ using structural rules to determine whether these c_i allow for the generation of "structural" predecessors.

Case 2: $c_i = \forall r.d$

If $\hat{r}_c = \bot$

$$\ell_{\downarrow\forall} \;:=\; \begin{array}{l} \{y \,|\, \forall s.f \in DEF(y) \text{ and } s \in \ell_{\uparrow r}\,\} \cup \\ \texttt{;; the special case } s \sqsubseteq r \Rightarrow \forall r.\bot \sqsubseteq \forall s.f \\ \{y \,|\, \exists_{\le} k\, s \in DEF(y) \text{ and } s \in \ell_{\uparrow r}\,\} \\ \texttt{;; the rule } s \sqsubseteq r \Rightarrow \forall r.\bot \sqsubseteq \exists_{\le} k\, s \texttt{ is satisfied} \end{array} \tag{16}$$

If $\hat{r}_c \neq \bot$

$$\ell_{\downarrow\forall} \;:=\; \{y \,|\, \forall s.f \in DEF(y) \text{ and } s \in \ell_{\uparrow r} \text{ and } \hat{s}_y \in \ell_{\downarrow \hat{r}_c}\,\} \tag{17}$$

$\texttt{;; the rule } s \sqsubseteq r \wedge d \sqsubseteq f \Rightarrow \forall r.d \sqsubseteq \forall s.f \texttt{ is satisfied}$

Note that the classifier has to use again the actual value restrictions $\hat{r}_c$ and $\hat{s}_y$ of r and s for both value restricting concepts in the last generation step. To reduce computational costs, the classifier addresses the special case $\hat{r}_c = \top$ first. In this case, $\hat{s}_y = \top$ has to hold. In any other cases, the classifier generates the candidate predecessors according to the structural rule and then tries to verify that each f_j in $\hat{s}_y = \sqcap_j f_j$ is a predecessor of $\hat{r}_c$. Testing $f_j \in \ell_{\downarrow d_i}$ for one d_i is sufficient for $f_j \in \ell_{\downarrow \hat{r}_c}$, but not necessary, i. e. a failed test $f_j \in \ell_{\downarrow d_i}$ requires to verify $f_j \in \ell_{\downarrow \hat{r}_c}$, i. e. classification of $\hat{r}_c$ becomes necessary.
The following example illustrates why a restriction to d would violate completeness.

Example 6 *Let us consider the example* $c \doteq \forall r.a \sqcap \forall r.b$ *and the concept in the taxonomy* $y \doteq \forall r.(a \sqcap b)$. *The successor generation can consider* a *and* b *as the single* d_i *in isolation, since all successors of* $a \sqcap b$ *have to be successors of* a *as well as of* b. *During predecessor generation, such a local testing would be insufficient. All predecessors of* a *and all predecessors of* b *are also predecessors of* $a \sqcap b$, *but a conjunction usually implies more than each of its single conjuncts. In the example,* $a \sqcap b$ *is only found as a valid predecessor when* $\hat{r}_c = a \sqcap b$ *is taken as input to the classifier.*

Case 3: $c_i = \exists_{\ge} n\, r$

If $n = 0$

$$\ell_{\downarrow\exists_{\ge}} \;:=\; \emptyset \tag{18}$$

$\texttt{;; the special case } \exists_{\ge} 0\, r \sqsubseteq \top$

If $n \ge 1$

$$\ell_{\downarrow\exists_{\ge}} \;:=\; \{y \,|\, \exists_{\ge} k\, s \in DEF(y) \text{ and } s \in \ell_{\downarrow r} \text{ and } k \le n\,\} \tag{19}$$

$\texttt{;; the rule } r \sqsubseteq s \wedge k \le n \Rightarrow \exists_{\ge} n\, r \sqsubseteq \exists_{\ge} k\, s \texttt{ is satisfied}$

In the case $n = 0$ the empty set is returned as a preliminary result. A concept $c_i = \exists_{\ge} 0\, r$ is equivalent to $\top$, which is its only predecessor, but also a predecessor of any concept and therefore added at the end of the predecessor generation in Step 23.

Case 4: $c_i = \exists_{\leq} n\, r$

If $n = 0$

$$\ell_{\downarrow\exists_{\leq}} := \begin{array}{l} \{y \mid \forall s.f \in DEF(y) \text{ and } s \in \ell_{\uparrow r}\} \cup \\ \texttt{;; the rule } s \sqsubseteq r \Rightarrow \exists_{\leq} 0\, r \sqsubseteq \forall s.f \texttt{ is satisfied} \\ \{y \mid \exists_{\leq} k\, s \in DEF(y) \text{ and } s \in \ell_{\uparrow r}\} \\ \texttt{;; the rule } s \sqsubseteq r \wedge k \geq n \Rightarrow \exists_{\leq} n\, r \sqsubseteq \exists_{\leq} k\, s \texttt{ is satisfied} \end{array} \tag{20}$$

If $n \geq 1$

$$\ell_{\downarrow\exists_{\leq}} := \{y \mid \exists_{\leq} k\, s \in DEF(y) \text{ and } s \in \ell_{\uparrow r} \text{ and } k \geq n\} \tag{21}$$

$$\texttt{;; the rule } s \sqsubseteq r \wedge k \geq n \Rightarrow \exists_{\leq} n\, r \sqsubseteq \exists_{\leq} k\, s \texttt{ is satisfied}$$

The sets $\ell_{\downarrow\forall}$, $\ell_{\downarrow\exists_{\geq}}$, and $\ell_{\downarrow\exists_{\leq}}$ add candidates y that satisfy that one of their y_i is subsumed by c. Obviously, the classifier has to verify that for each y *all* y_i are subsumed by c.

Definition 22 *The set $\aleph_{\downarrow c}$ of structural predecessors of c in the taxonomy is the set of concepts y with $y \in \ell_{\downarrow\exists_{\geq}} \cup \ell_{\downarrow\exists_{\leq}} \cup \ell_{\downarrow\forall}$ and y being covered by c, i. e. satisfying Definition 21.*

Definition 21 makes sure that y is a defined concept and that all y_i are valid predecessors of c. With that, $\aleph_{\downarrow c}$ contains only valid predecessors. The set is now completed by adding all their predecessors (except $\top$) to it.

$$\aleph_{\downarrow c} := \aleph_{\downarrow c} \cup \{x \mid \forall y \in \aleph_{\downarrow c} : x \in \ell_{\downarrow y} \text{ and } TAUT(x) = \text{FALSE}\} \tag{22}$$

$$\texttt{;; for all } y \texttt{ all their predecessors } x \texttt{ (except } \top\texttt{) are added}$$

Finally, there is still the possibility that successors of $\wp_{\downarrow c}$ and $\aleph_{\downarrow c}$ are indirect predecessors of c. Therefore, $\wp^n_{\downarrow c}$ is called again with $\wp_{\downarrow c} \cup \aleph_{\downarrow c}$ as input. If new indirect predecessors are found, a further recursive call is necessary taking the updated sets as input. The recursion terminates when no new elements are added to $\wp_{\downarrow c}$ and $\aleph_{\downarrow c}$, i. e. until a fixed point of the generation process is reached leading to the final sets of predecessors $\wp^+_{\downarrow c}$ and $\aleph^+_{\downarrow c}$. With that, the set $\ell_{\downarrow c}$ is obtained as

$$\ell_{\downarrow c} := \wp^+_{\downarrow c} \cup \aleph^+_{\downarrow c} \cup \{\top\} \tag{23}$$

Based on $\ell_{\downarrow c}$ the set PREC(c) of immediate predecessors of c can be computed performing a reduction operation dual to the one defined for $\ell_{\uparrow c}$.

Definition 23 *The set of immediate predecessors of a concept c is the set PREC(c) with*

$$\mathit{PREC(c)} := \begin{cases} \ell_{\downarrow c} & \text{if } \ell_{\downarrow c} \text{ singleton} \\ \{y \mid y \in \ell_{\downarrow c} \text{ and } \forall x \in \ell_{\downarrow c} \text{ with } y \not\equiv x : y \notin \ell_{\downarrow x}\} & \text{otherwise} \end{cases}$$

If PREC(c) $\cap$ SUCC(c) $\neq \emptyset$, then c is equivalent to the concepts contained in this intersection.

5 Soundness and Completeness

The following theorems state the soundness and completeness of the structural classifier for the languages $\mathcal{TF}^\star$ and $\mathcal{TF}^\ominus$.

Theorem 1 ((Soundness and Completeness of SUCC)) *For any two atomic concepts $x, c \in \mathcal{TF}^\star$ or $x, c \in \mathcal{TF}^\ominus$ and an arbitrary terminology $\mathcal{T}$ satisfying the syntactic restrictions of either one of the two sublanguages, $x \in$ SUCC(c) if and only if in all models of $\mathcal{T}$ $x \sqsubseteq c$ and for all atomic concepts z in $\mathcal{T}$ holds: If $z \sqsubseteq_\mathcal{T} c$ then either $z \equiv x$ or $x \not\sqsubseteq_\mathcal{T} z$.*

Theorem 2 ((Soundness and Completeness of PREC)) *For any two atomic concepts $x, c \in \mathcal{TF}^\star$ or $x, c \in \mathcal{TF}^\ominus$ and an arbitrary terminology $\mathcal{T}$ satisfying the syntactic restrictions of either one of the two sublanguages, $x \in$ PREC(c) if and only if in all models of $\mathcal{T}$ $c \sqsubseteq x$ and for all atomic concepts z in $\mathcal{T}$ holds: If $c \sqsubseteq_\mathcal{T} z$ then either $z \equiv x$ or $z \not\sqsubseteq_\mathcal{T} x$.*

All proofs can be found in the Appendix.

6 An Example

The following small example illustrates the behavior of the structural classifier. We assume that $\mathcal{T}$ contains the specializations and axioms shown in Figure 6. (The left-hand side of the figure shows the taxonomy that is induced by $\mathcal{T}$.)

$$\begin{array}{rcl}
\mathsf{Color} & \dot{\leq} & \top \\
\mathsf{Red} & \dot{\leq} & \mathsf{Color} \\
\mathsf{Blue} & \dot{\leq} & \mathsf{Color} \\
\mathsf{Yellow} & \dot{\leq} & \mathsf{Color} \\
\mathsf{Purple} & \doteq & \mathsf{Red} \sqcap \mathsf{Blue} \\
\mathsf{Orange} & \doteq & \mathsf{Red} \sqcap \mathsf{Yellow} \\
\mathsf{Green} & \doteq & \mathsf{Blue} \sqcap \mathsf{Yellow} \\
\mathsf{Black} & \doteq & \mathsf{Purple} \sqcap \mathsf{Orange} \sqcap \mathsf{Green} \\
\mathsf{Item} & \dot{\leq} & \top \\
\mathsf{Item1} & \doteq & \mathsf{Item} \sqcap \forall\ \mathsf{coloring.Red} \\
\mathsf{Item2} & \doteq & \mathsf{Item} \sqcap \forall\ \mathsf{coloring.Blue} \\
\mathsf{Item3} & \doteq & \mathsf{Item1} \sqcap \mathsf{Item2}
\end{array}$$

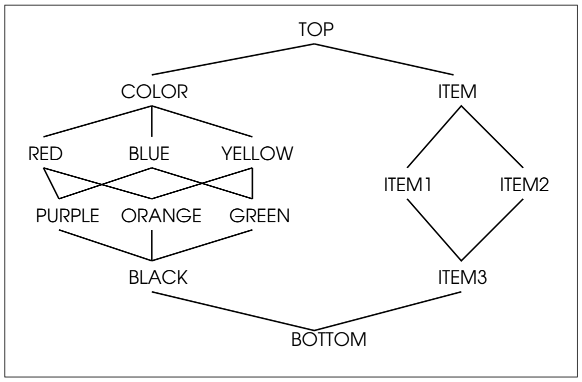

Figure 6: The Example Terminology

Let us assume that we want to insert the concept N-Item, which is defined as

$$\mathsf{N} - \mathsf{Item} \doteq \mathsf{Item} \sqcap \forall \mathsf{coloring.Purple}$$

We start by generating SUCC(N-Item). The classifier sets $c_1 = \mathsf{Item}$ and $c_2 = \forall\,\mathsf{coloring.Purple}$ and begins with the generation of $\ell_{\uparrow c_1}$ and $\ell_{\uparrow c_2}$. Case 1 matches to c_1, because it is primitive, while Case 2 matches to c_2, because it contains a value restriction.
For c_1, its immediate successors are determined from the taxonomy from which the classifier can read off $\mathsf{Item1} \prec_{\mathcal{T}} \mathsf{Item}$ and $\mathsf{Item2} \prec_{\mathcal{T}} \mathsf{Item}$. This leads to

$$\ell_{\uparrow c_1} := \{\mathsf{Item1}, \mathsf{Item2}\}$$

For $c_2 = \forall\,\mathsf{coloring.Purple}$, the generation depends on whether the value restriction is a tautology or not. In the example, the classifier obtains $\hat{r}_c = \mathsf{Red} \sqcap \mathsf{Blue}$, which is satisfiable, but not a tautology. Therefore, the three structural rules (3) are applied. The terminology contains neither number restrictions nor unsatisfiable value restrictions and thus, only the second rule can return successors. Since d is non-primitive, the classifier needs to invoke tests with respect to each d_i following Rule (4)
This binds

$$d_1 = \mathsf{Red} \qquad\qquad d_2 = \mathsf{Blue}$$

Now, the classifier has to generate the sets $\ell_{\uparrow Red}$ and $\ell_{\uparrow Blue}$. It traverses the taxonomy and adds all successors of these two concepts.

$$\ell_{\uparrow Red} := \{\mathsf{red}, \mathsf{purple}, \mathsf{black}\} \qquad\qquad \ell_{\uparrow Blue} := \{\mathsf{blue}, \mathsf{purple}, \mathsf{black}\}$$

As the next step in order to generate $\ell_{\uparrow c_2}$, the classifier has to generate the set $\ell_{\downarrow coloring}$, which is the singleton $\{\mathsf{coloring}\}$ in $\mathcal{T}$.
Now, the classifier looks for concepts that contain $\forall s.f$ in their definition where $s \in \{\mathsf{coloring}\}$ and where $\hat{s}_y \in \{\mathsf{red}, \mathsf{purple}, \mathsf{black}\}$ or $\hat{s}_y \in \{\mathsf{blue}, \mathsf{purple}, \mathsf{black}\}$. It finds Item1 and Item2 that contain matching concept descriptions. All their successors are added to the candidate set. Thus, it generates

$$\ell_{\uparrow Item1} := \{\mathsf{Item1}, \mathsf{Item3}, \bot\} \qquad\qquad \ell_{\uparrow Item2} := \{\mathsf{Item2}, \mathsf{Item3}, \bot\}$$

Still working on the generation of successors for c_2, the intersection of the last two sets is computed and the set $\{\mathsf{Item3}, \bot\}$ is obtained as the final result for $\ell_{\uparrow \forall coloring.Purple}$. With that the sets $\ell_{\uparrow c_1}$ and $\ell_{\uparrow c_2}$ can be completed by adding all their successors leading to

$$\ell_{\uparrow c_1} := \{\mathsf{Item1}, \mathsf{Item2}, \mathsf{Item3}, \bot\} \qquad\qquad \ell_{\uparrow c_2} := \{\mathsf{Item3}, \bot\}$$

This completes the successor generation for the single c_i. Finally, according to Rule (11), the intersection of the two sets is computed. We obtain $\ell_{\uparrow N-Item}$ as

$$\ell_{\uparrow N-Item} := \{\mathsf{Item3}, \bot\}$$

The reduction operation eliminates $\bot$, since it is a successor of Item3. With that we obtain SUCC(N-Item) as $\{\mathsf{Item3}\}$.
Now, PREC(N-Item) can be computed and the classifier starts generating the predecessors of the atomic concepts contained in the definition of N-Item. In the example,

this amounts to computing the predecessors if Item ignoring *top*, which is only Item itself, i. e.

$$\wp^0_{\downarrow N-Item} := \{\mathsf{Item}\}$$

Now, successors of all elements in $\wp^0_{\downarrow \mathsf{N-Item}}$ have to be investigated. The classifier starts by generating $\wp^1_{\downarrow \mathsf{N-Item}}$ which contains the immediate successors of Item. Candidates for this set are the concepts $\mathsf{Item1}$ and $\mathsf{Item2}$. Both concepts are defined as conjunctions and the classifier has to test that each conjunct is a valid predecessor. Item is contained in $\wp^0_{\downarrow N-Item}$ and for the conjuncts $\forall\,\mathsf{coloring.Red}$ and $\forall\,\mathsf{coloring.Blue}$ the structural Rule (17) is activated. Since Red and Blue are primitive concepts and the roles are identical, the tests are very easy to perform. The concept $\hat{r}_c = \mathsf{Red} \sqcap \mathsf{Blue}$ is classified, which returns Red and Blue as predecessors.

$$\wp^1_{\downarrow N-Item} := \{\mathsf{Item1}, \mathsf{Item2}\}$$

Now the successors of $\mathsf{Item1}$ and $\mathsf{Item2}$, which is only $\mathsf{Item3}$, have to be analyzed. $\mathsf{Item3}$ is also defined as a conjunction of valid predecessors and it is therefore a predecessor of $\mathsf{N\text{-}Item}$ as well.

$$\wp^2_{\downarrow N-Item} := \{\mathsf{Item3}\}$$

The successor of $\mathsf{Item3}$ is *bottom* and with that the generation of indirect successors is completed and we obtain

$$\wp_{\downarrow N-Item} := \{\mathsf{Item}, \mathsf{Item1}, \mathsf{Item2}, \mathsf{Item3}\}$$

Now, the classifier addresses the remaining non-atomic concept $\forall\,\mathsf{coloring.Purple}$ contained in the definition of $\mathsf{N\text{-}Item}$ following Rule (17). The expansion of Purple leads to $\hat{r}_c = \mathsf{Red} \sqcap \mathsf{Blue}$. With that Rule (17) generates $\mathsf{Item1}$ and $\mathsf{Item2}$ as possible predecessors, which satisfy the condition that the remaining defined concept Item contained in their definition is a predecessor of $\mathsf{N\text{-}Item}$.

$$\aleph_{\downarrow N-Item} := \{\mathsf{Item1}, \mathsf{Item2}\}$$

In this example, $\aleph_{\downarrow N-Item}$ did not lead to any further predecessors and thus the fixed point of the generation process is achieved as

$$\ell_{\downarrow N-Item} := \wp_{\downarrow N-Item} \cup \{\top\} = \{\top, \mathsf{Item}, \mathsf{Item1}, \mathsf{Item2}, \mathsf{Item3}\}$$

Finally the reduction operation is applied to this set leading to PREC(c)= $\{\mathsf{Item3}\}$. Since PREC(c) $\cap$ SUCC(c) = $\mathsf{Item3}$ the classifier concludes that $\mathsf{N\text{-}Item}$ is equivalent to $\mathsf{Item3}$.

The reader should note that the classifier would execute exactly the same generation steps if the terminology contained millions of concepts not referring to any color or item concepts in their definitions. For the example, the amount of necessary tests does only increase when more concepts defined with the help of value restrictions on the role $\mathsf{coloring}$ or its sub- or superroles are be added.

7 Conclusion and Future Work

We have presented a structural classification algorithm that works on the folded representation of concepts. Current work is devoted to both further theoretical investigation and implementational improvement of the PARKA classifier. First of all, the current formalization of the classifier does not take full advantage of all optimization possibilities. In particular, the preprocessing phase, during which a fully expanded normal-form representation of value restrictions is computed needs further improval. Besides this, we work on a detailed comparison of the structural approach with the optimization techniques discussed in [3].
A classifier for an extension of the language $\mathcal{TF}$ allowing for arbitrary role hierarchies and set operations on roles that combines the structural approach with an arithmetic problem solver in order to achieve completeness for number restrictions over role hierarchies [11] is currently under development.
We are also working on improving the performance of the classifier to allow scaling to significantly larger knowledge bases. In order to handle very large taxonomies that do not fit into resident computer memory, we are currently working on a representation of these taxonomies using relational database technology and the implementation of the generative classifier in terms of database operations. Preliminary results show that the database-based generative classifier should allow us to deal with virtually arbitrarily sized KBs by exploiting the database's memory management schemes. In addition, we are also working on a high performance implementation of PARKA's classifier, which will exploit a number of parallel optimizations and should improve on the current parallel implementation, allowing the scaling to significantly larger taxonomies running on a larger number of processors.

Acknowledgments

This research was supported in part by grants from NSF(IRI-9306580), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039), the ARPA I3 Initiative (N00014-94-10907) and ARPA contract DAST-95-C0037. Jana Koehler is supported through an ICSI Fellowship Award. Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSF EEC 94-02384).
We want to thank Hans Jürgen Ohlbach and Werner Nutt for their helpful comments on a draft version of the algorithm's formalization. Hans Jürgen Ohlbach helped us in getting the formalization right and in relating it to results from the theory of description logics by pointing out many critical examples. His comments heavily influenced the paper.

8 Appendix: Proofs of Theorems

Soundness and Completeness of TAUT and USAT

Proposition 3 (1) *The test TAUT(c) for c given in either $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$ delivers* TRUE *if and only if $\mathcal{E}[c] = \mathcal{D}$ in all models of $\mathcal{T}$.*

Proof: The proof proceeds by structual induction over c.
Soundness: $TAUT(c) =$ TRUE only if $\mathcal{E}[c] = \mathcal{D}$ in all models of $\mathcal{T}$.

1. $\mathcal{E}[\top] = \mathcal{D}$ follows immediately from Definition 9.
2. $\mathcal{E}[\exists_{\geq} 0\, r] = \mathcal{D}$ holds because the requirement of at least zero fillers is satisfied for any domain element in $\mathcal{D}$ independent of whether the relation is defined for it or not.
3. $\mathcal{E}[\forall\, r.d] = \mathcal{D}$ for $TAUT(\hat{r}_c) =$ TRUE because of $\mathcal{E}[\hat{r}_c] = \mathcal{D}$ based on the induction hypothesis and thus there can be no domain element α with $\alpha \notin \mathcal{E}[\hat{r}_c] = \mathcal{D}$.
4. $\mathcal{E}[a \sqcap b] = \mathcal{D}$ only if $\mathcal{E}[a] \cap \mathcal{E}[b] = \mathcal{D}$, which is only possible when $\mathcal{E}[a] = \mathcal{D}$ and $\mathcal{E}[b] = \mathcal{D}$ according to the semantics of set intersection.

Completeness: $TAUT(c) =$ FALSE only if $\mathcal{E}[c] \subset \mathcal{D}$ in some model of $\mathcal{T}$. The function $TAUT(c) =$ returns FALSE only in the following cases:

1. $c = \bot$ with $\mathcal{E}[\bot] = \emptyset \subset \mathcal{D}$ follows immediately from Definition 9.
2. $c = a$ with $\mathcal{E}[a] \subset \mathcal{D}$ because the interpretation of a primitive concept is a subset of $\mathcal{D}$ according to Definition 9.
3. $c = a \sqcap b$ then $TAUT(c) =$ FALSE if either $TAUT(a) =$ FALSE or $TAUT(b) =$ FALSE. Thus there exists a model $\mathcal{E}$ with either $\mathcal{E}[a] \subset \mathcal{D}$ or $\mathcal{E}[b] \subset \mathcal{D}$ and therefore $\mathcal{E}[c] \subset \mathcal{D}$.
4. $c = \exists_{\leq} n\, r$ then $TAUT(c) =$ FALSE since we can construct a model in which r has more than n role fillers for at least one object in the domain, i. e. $\mathcal{E}[\exists_{\leq} n\, r] \subset \mathcal{D}$ unless $n = \infty$, which is not admitted since n is required to be finite.
5. $c = \exists_{\geq} n\, r$ then $TAUT(c) =$ FALSE for $n \geq 1$ and we can construct a model in which r has no role fillers or is undefined for at least one object in the domain, i. e. $\mathcal{E}[c] \subset \mathcal{D}$.
6. $c = \forall\, r.d$ then $TAUT(c) =$ FALSE if $TAUT(\hat{r}_c) =$ FALSE. Thus there must be a $\beta \in \mathcal{D}$ with $\beta \notin \mathcal{E}[\hat{r}_c]$ according to the induction hypothesis. Therefore, we can construct a model $\mathcal{E}'$ with $\langle \alpha, \beta \rangle \in \mathcal{E}'[r]$, but $\beta \notin \mathcal{E}'[\hat{r}_c]$ and thus $\mathcal{E}'[c] \subset \mathcal{D}$.

■

Proposition 4 (2) *Let c be a concept description given ein either in $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$ then $USAT(c) =$* TRUE *if and only if $\mathcal{E}[c] = \emptyset$ in all models of $\mathcal{T}$.*

Proof: Soundness: $USAT(c) =$ TRUE only if $\mathcal{E}[c] = \emptyset$ in all models of $\mathcal{T}$.

1. $\mathcal{E}[\bot] = \emptyset$ according to Definition 9.
2. $\mathcal{E}[a \sqcap b] = \emptyset$ only if $\mathcal{E}[a] = \emptyset$ or $\mathcal{E}[a] = \emptyset$ according to the semantics of set intersection.

3. $\mathcal{E}[\forall\, r.\bot \sqcap \exists_{\geq} n\, s] = \emptyset$ because it requires the existence of a domain element x for which no y exists satisfying r, but there is atleast one y satisfying s because of $n \geq 1$. Since $\mathcal{E}[s] \subseteq \mathcal{E}[r]$, and $\langle x, y\rangle$ in $\mathcal{E}[s]$ it follows that $\langle x, y\rangle$ in $\mathcal{E}[r]$, i. e. there must be a y for x satisfying r and thus a contradiction occurs.

4. $\mathcal{E}[\exists_{\geq} n\, r \sqcap \exists_{\leq} k\, s] = \emptyset$ because we have atleast n pairs $\langle x, y\rangle$ in $\mathcal{E}[r]$ and since $\mathcal{E}[r] \subseteq \mathcal{E}[s]$ there must be atleast the same n pairs in $\mathcal{E}[s]$, while $\mathcal{E}[s]$ is allowed to be satisfied by only a strictly smaller number of pairs because of $k < n$.

We prove completeness of USAT for the language $\mathcal{TF}^{\ominus}$ by constructing for the case USAT(c)=FALSE an interpretation $\mathcal{E}$ with $\mathcal{E}[c] \neq \emptyset$. Without loss of generality we can assume that $\mathcal{X}(c, \mathcal{T})$ does not contain redundant information of the kind $\exists_{\geq} n\, r \sqcap \exists_{\geq} m\, r$ and $\exists_{\leq} n\, r \sqcap \exists_{\leq} m\, r$. In the first case $\exists_{\geq} max(n,m)\, r$ and in the second case $\exists_{\leq} min(n,m)\, r$ are sufficient.
The technique for constructing such an interpretation is borrowed from the completeness proof technique for tableaux calculi. We define a set of labeled tableaux rules which computes an open branch from which we can read off an interpretation satisfying c. We get only one single such branch, since the language contains no disjunction and negation. Starting with an initial set $B = \{a : c\}$ where a is a fresh constant, we extend B by adding new facts according to the following rules:

$$\begin{array}{lcl} a : c_1 \sqcap c_2 & \rightarrow & a : c_1 \text{ and } a : c_2 \\ a : \exists_{\geq} n\, r & \rightarrow & r(a, a_1) \text{ and } \ldots \text{ and } r(a, a_n) \text{ where the } a_i \text{ are fresh} \\ a : \forall r.d \text{ and } r(a,b) & \rightarrow & b : d \end{array}$$

The rules mean that whenever B contains instances of the facts from the left hand side of the rule then the corresponding instances of the facts from the right hand side are to be added to B, and this as long as possible, but without repeated application to the same facts. Since c is finite, this rule system always terminates.
A fully expanded branch B can be turned into an interpretation $\mathcal{E}$ with domain $\mathcal{D}$ and corresponding interpretation of the roles and concepts.

$$\begin{array}{rcl} \mathcal{D} & = & \{a \mid a : d \in B\} \cup \{a \mid r(a,b) \in B \text{ or } r(b,a) \in B\} \textit{ for some d,r,b} \\ \mathcal{E}[d] & = & \{a \mid a : d \in B\} \textit{ for an atomic concept d} \\ \mathcal{E}[r] & = & \{(a,b) \mid r(a,b) \in B\} \textit{ for a role r} \end{array}$$

Now we have to show that if USAT(c) = FALSE and B is a fully expanded branch according to the above rules and $\mathcal{E}$ is the interpretation generated by B then $\mathcal{E}$ satisfies all facts in B in the following sense:

If $a : d \in B$ then $a \in \mathcal{E}[d]$ and if $r(a,b) \in B$ then $(a,b) \in \mathcal{E}[r]$.

In particular, this means for the starting term c and the starting fact $a : c \in B$ that $a \in \mathcal{E}[c]$ which implies immediately that $\mathcal{E}[c] \neq \emptyset$. By the construction of the interpretation of the roles, this is obviously true for the role terms $r(a,b) \in B$. For the concept terms $a : d$ we prove this claim by induction on the structure of d. The base case where d is atomic holds by the construction of $\mathcal{E}$. For the induction step we have to consider the following cases:

- $\mathbf{d = d_1 \sqcap d_2}$: If USAT(d) = FALSE then by definition of USAT it holds that USAT(d_1) = USAT(d_2) = FALSE. Furthermore by the $\sqcap$ expansion rule, $a : d_1 \in B$ and $a : d_2 \in B$. Thus we can apply the induction hypothesis which yields $a \in \mathcal{E}[d_1]$ and $a \in \mathcal{E}[d_2]$ and therefore $a \in \mathcal{E}[d_1 \sqcap d_2]$.
- $\mathbf{d} = \exists_{\geq} \mathbf{n\, r}$: The expansion rule for $\exists_{\geq}$ generates n r-successors for a. Therefore $\mathcal{E}$ satisfies $a : \exists_{\geq} n\, r$.
- $\mathbf{d} = \exists_{\leq} \mathbf{k\, r}$: Suppose $r(a, a_1), \ldots, r(a, a_m) \in B$ with $m > k$. According to the construction of B this can only happen if $a : \exists_{\geq} m\, r \in B$, which is only possible if $a : (\exists_{\geq} m\, r \sqcap \exists_{\leq} k\, r) \in B$. This violates the USAT(d) = FALSE assumption. Therefore there cannot be more than k $r-$successors of a in B.
- $\mathbf{d} = \forall\, \mathbf{r.e}$: If there are any r-successors a_i of a in B we have $a_i : e \in B$ according to the extension rule for $\forall$. Furthermore $a : \exists_{\geq} n\, r$ must be in B as well as $a : \forall\, r.e \sqcap \exists_{\geq} n\, r$. Since USAT(d) = FALSE and by definition of USAT we know that USAT(e) = FALSE. Thus, the induction hypothesis can be applied and we obtain for all r-successors a_i of a: $a_i \in \mathcal{E}[e]$. Thus $a \in \mathcal{E}[\forall\, r.e]$. If there are no r-successors of a in B then $\mathcal{E}[r(a)] = \emptyset$ and therefore $a \in \mathcal{E}[\forall\, r.e]$.

■

Soundness and Completeness of SUCC

In order to prove soundness and completeness of SUCC, we first show soundness and completeness of $\ell_{\uparrow c}$ for the languages $\mathcal{TF}^{\ominus}$ and $\mathcal{TF}^{\star}$. The algorithm obtains a concept description $c \doteq \sqcap_i c_i$ or $c \dot{\leq} \sqcap_i c_i$ as input and returns a set of successor concepts y from the taxonomy. The proofs proceed by structural induction over c and y.

Soundness of $\ell_{\uparrow c}$

Proposition 5 *Let c, y be concept descriptions in a terminology $\mathcal{T}$ given in either $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$ then $y \in \ell_{\uparrow c}$ only if $\mathcal{E}[c] \supseteq \mathcal{E}[y]$ holds in all models of $\mathcal{T}$.*

Proof:
We prove soundness directly by showing that $y \in \ell_{\uparrow c}$ implies $\mathcal{E}[c] \supseteq \mathcal{E}[y]$. The first part of the proof considers the various syntactical base cases for c, then soundness is proven for c being a concept conjunction or concept specialization. For each of the syntactical possibilities of c we proceed recursively over the syntactic structure of the successor concepts y that are returned by the classifier.

1. $c \doteq \top$

This case is completely addressed in the preprocessing phase which returns $SUCC(c) = \{\top\}$. According to Definition 9 and soundness of TAUT we know that $\mathcal{E}[c] = \mathcal{D}$ and $\mathcal{E}[y] = \mathcal{D}$ and therefore $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \supseteq \mathcal{E}[y]$.

2. $c \doteq \bot$

The preprocessing phase returns $SUCC(c) = \{\bot\}$. According to Definition 9 and soundness of USAT we know that $\mathcal{E}[c] = \emptyset$ and $\mathcal{E}[y] = \emptyset$ and again $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \supseteq \mathcal{E}[y]$.

3. $c \doteq a$ with a atomic

The concept a is a node in the taxonomy and the classifier returns the successors of a as a result following the $\prec_{\mathcal{T}}$ relation in Step 1. According to Definitions 14 and 12

$$y \prec_{\mathcal{T}} a \Rightarrow y \sqsubseteq a \Rightarrow \mathcal{E}[y] \subseteq \mathcal{E}[a] \Rightarrow \mathcal{E}[a] \supseteq \mathcal{E}[y]$$

Since $\mathcal{E}[c] = \mathcal{E}[a]$, it follows that $\mathcal{E}[c] \supseteq \mathcal{E}[y]$. A similar construction is used in Steps 10 and 5 when the successors x of a successor y are recursively added to $\ell_{\uparrow c}$. Soundness of these steps follows from the transitivity of the subsumption and the successor relations

$$x \prec_{\mathcal{T}} y \wedge y \prec_{\mathcal{T}} a \Rightarrow x \prec_{\mathcal{T}} a \Rightarrow \mathcal{E}[a] \supseteq \mathcal{E}[x]$$

4. $c \doteq \forall\, r.d$

Note that c is not a conjunction, but that d can be an arbitrary concept description. The concept $\hat{r}_c$ represents the expanded normal-form representation of d, because c does not contain or inherit other value restriction on r.
Step 2 addresses the special case $\hat{r}_c = \top$ and returns $y = \top$. We know that $TAUT(\hat{r}_c)$ must have returned TRUE because of $\hat{r}_c = \top$ and since $TAUT$ is sound and complete, it follows $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d] = \mathcal{D}$ and thus $\mathcal{E}[c] = \mathcal{D}$. Furthermore $\mathcal{E}[y] = \mathcal{D}$ according to Definition 9 and thus $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \supseteq \mathcal{E}[y]$.
Step 3 considers the case $\hat{r}_c \neq \top$ (including $\hat{r}_c = \bot$). We proceed by structural induction over the returned successor concepts y:

- $y = \forall\, s.f$ with $s \in \ell_{\downarrow r}$ and $\hat{s}_y = \bot$
  The condition $s \in \ell_{\downarrow r}$ holds only if $\mathcal{E}[r] \subseteq \mathcal{E}[s]$ according to Definition 16. The preprocessing phase sets $\hat{s}_y = \bot$ only if $USAT(\hat{s}_y) =$ TRUE and because of Proposition 2 it holds $\mathcal{E}[f] = \mathcal{E}[\hat{s}_y] = \emptyset$ since $\hat{s}_y$ represents the expanded normal-form representation of f and y is not a conjunction. Thus $\mathcal{E}[d] \supseteq \mathcal{E}[f]$ is satisfied and soundness follows according to the subsequent observation for any extension function $\mathcal{E}$:

$$\mathcal{E}[d] \supseteq \mathcal{E}[f] \wedge \mathcal{E}[r] \subseteq \mathcal{E}[s] \Rightarrow \mathcal{E}[\forall\, s.d] \supseteq \mathcal{E}[\forall\, s.f] \Rightarrow \mathcal{E}[\forall\, r.d] \supseteq \mathcal{E}[\forall\, s.f]$$

- $y = \forall\, s.f$ with $s \in \ell_{\downarrow r}$ and $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$
  Note that again $\mathcal{E}[\hat{s}_y] = \mathcal{E}[f]$ and $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d]$ because of the syntactic restrictions on c and y and that $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$ implies $\mathcal{E}[d] \supseteq \mathcal{E}[f]$ because of the induction hypothesis. Thus $\mathcal{E}[r] \subseteq \mathcal{E}[s]$ and $\mathcal{E}[d] \supseteq \mathcal{E}[f]$ are satisfied and soundness follows based on the above observation for $\mathcal{E}$.

- $y = \exists_{\leq} 0\, s$ with $s \in \ell_{\downarrow r}$
  The condition $s \in \ell_{\downarrow r}$ satisfies $\mathcal{E}[r] \subseteq \mathcal{E}[s]$ according to Definition 16. For any extension function holds $\mathcal{E}[r] \subseteq \mathcal{E}[s] \Rightarrow \mathcal{E}[\forall\, r.d] \supseteq \mathcal{E}[\forall\, s.d]$.

  Any arbitrary concept d satisfies $\mathcal{E}[d] \supseteq \mathcal{E}[\bot]$ and based on Definition 9 we know that $\mathcal{E}[d] \supseteq \mathcal{E}[\bot] \Rightarrow \mathcal{E}[\forall\, s.d] \supseteq \mathcal{E}[\forall\, s.\bot]$.

  With that $\mathcal{E}[\forall\, r.d] \supseteq \mathcal{E}[\forall\, s.\bot]$ must hold and thus $\mathcal{E}[\forall\, r.d] \supseteq \mathcal{E}[\exists_{\leq} 0\, s]$ follows because of $\mathcal{E}[\forall\, s.\bot] = \mathcal{E}[\exists_{\leq} 0\, s]$.

- $y \doteq \sqcap_i y_i$ and $\exists\, y_i : \mathcal{E}[c] \supseteq \mathcal{E}[y_i]$
  A concept conjunction y is returned as a successor only if one of its conjuncts satisfies the previous cases. Since $\mathcal{E}[\sqcap_i y_i] = \cap_i\, \mathcal{E}[y_i]$ and $\mathcal{E}[y_i] \supseteq \cap_i\, \mathcal{E}[y_i]$ it follows $\mathcal{E}[c] \supseteq \mathcal{E}[y]$. If y is a successor because of an $y_i = \forall\, s.f$ satisfying the generation rules, then the conditions $\mathcal{E}[r] \subseteq \mathcal{E}[s]$ and $\mathcal{E}[\hat{r}_c] \supseteq \mathcal{E}[\hat{s}_y]$ are sufficient for $\mathcal{E}[c] \supseteq \mathcal{E}[y]$ although $\mathcal{E}[\hat{s}_y] \subseteq \mathcal{E}[f]$ since y contains no disjunction. But $\hat{s}_y$ represents the actual value restriction on s, i. e. replacing all occurrences of $\forall\, s.f$ with $\forall\, s.\hat{s}_y$ in y is extension preserving. Soundness of the test $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$ follows from the following observations:

  - In the special case $\hat{r}_c = \bot$, soundness and completeness of $USAT$ guarantee that $\mathcal{E}[\hat{r}_c] = \emptyset$. The algorithm considers only value restrictions with $\hat{s}_y = \bot$ which satisfy $\mathcal{E}[\hat{r}_c] \supseteq \mathcal{E}[sy]$ because of $\mathcal{E}[\hat{s}_y] = \emptyset$.
  - If $\hat{r}_c \neq \bot$ the test is based on the syntactical structure of $\hat{r}_c = \sqcap_i d_i$:

    $$\begin{aligned} \mathcal{E}[\hat{r}_c] \supseteq \mathcal{E}[\hat{s}_y] &\Leftrightarrow \mathcal{E}[\sqcap_i d_i] \supseteq \mathcal{E}[\hat{s}_y] \\ &\Leftrightarrow \cap_i\, \mathcal{E}[d_i] \supseteq \mathcal{E}[\hat{s}_y] \\ &\Leftrightarrow \mathcal{E}[d_1] \supseteq \mathcal{E}[\hat{s}_y] \wedge \mathcal{E}[d_2] \supseteq \mathcal{E}[\hat{s}_y] \wedge \ldots \wedge \mathcal{E}[d_n] \supseteq \mathcal{E}[\hat{s}_y] \end{aligned}$$

    This justifies soundness of the successor generation for the single d_i. Step 4 generates candidates y satisfying that $\hat{s}_y$ is a successor of d_i leading to the sets $\mathcal{Y}_i$. Then in Step 5 all their successors are added, for which soundness has already been proven. The intersection of the single sets $\ell_{\uparrow \mathcal{Y}_i}$ makes sure that only concepts are preserved that satisfy $\cap_i\, \mathcal{E}[d_i] \supseteq \mathcal{E}[\hat{s}_y]$ and with that $\mathcal{E}[\hat{r}_c] \supseteq \mathcal{E}[\hat{s}_y]$.

  If y is generated because of $y_i = \exists_{\leq} 0\, s$ then problems could occur if the actual number restriction on s would be greater than 0. But if y contains or inherits a $y_j = \exists_{\leq} m\, s$ and $m \geq 1$ then no increase occurs because of $min(0, m)$. If y inherits or contains a number restriction $y_j = \exists_{\geq} m\, s$ with $m \geq 1$ then y itself becomes unsatisfiable and is a successor of any concept. Other possibilites are excluded in a language without role hierarchies.

5. $c \doteq \exists_{\geq} n\, r$

- $y = \top$
  is returned in Step 7 for the case $n = 0$. Soundness follows from soundness of $TAUT$: $\mathcal{E}[c] = \mathcal{E}[y] = \mathcal{D}$ and thus $\mathcal{E}[c] \supseteq \mathcal{E}[y]$.

- $y = \exists_{\geq} k\, s$ with $k \geq n$ and $s \in \ell_{\uparrow r}$
  is returned in Step 8 for the case $n \geq 1$. Soundness of this step follows from the observation

  $$\mathcal{E}[r] \supseteq \mathcal{E}[s] \wedge k \geq n \Rightarrow \mathcal{E}[\exists_{\geq} n\, s] \supseteq \mathcal{E}[\exists_{\geq} k\, s] \Rightarrow \mathcal{E}[\exists_{\geq} n\, r] \supseteq \mathcal{E}[\exists_{\geq} k\, s]$$

- $y \doteq \sqcap_i y_i$ and $\exists\, y_i : \mathcal{E}[c] \supseteq \mathcal{E}[y_i]$
  A concept conjunction y is returned as a successor only if one of its conjuncts satisfies $y = \exists_{\geq} k\, s$ with $k \geq n$ and $s \in \ell_{\uparrow r}$. The test on k becomes invalid when the actual number restriction of s in y is smaller than k. If y contains or inherits a $y_j = \exists_{\geq} m\, s$ and $m \leq k \leq n$, soundness would still be guaranteed, since the number restriction on s in y is $max(m, k)$. If $y_j = \exists_{\leq} m\, s$ and $m \leq k$ then y itself becomes unsatisfiable and

is a successor of any concept. In the language $\mathcal{TF}^{\star}$ the latter situation cannot occur, because no $\exists_{\leq}$ operator is provided.

6. $c \doteq \exists_{\leq} n\, r$

No case analysis over n is necessary and Step 9 returns:

- $y = \forall\, s.f$ with $s \in \ell_{\downarrow r}$ and $\hat{s}_y = \perp$
  Since $\mathcal{E}[\forall\, s.\perp] = \mathcal{E}[\exists_{\leq} 0\, s]$ and $\mathcal{E}[\exists_{\leq} 0\, s] \subseteq \mathcal{E}[\exists_{\leq} n\, s]$ we can conclude that $\mathcal{E}[\forall\, s.\perp] \subseteq \mathcal{E}[\exists_{\leq} n\, r]$ if $\mathcal{E}[r] \subseteq \mathcal{E}[s]$.

- $y = \exists_{\leq} k\, s$ with $k \leq n$ and $s \in \ell_{\downarrow r}$
  Soundness follows from the observation

$$k \leq n\, \wedge \mathcal{E}[s] \supseteq \mathcal{E}[r] \Rightarrow \mathcal{E}[\exists_{\leq} n\, r] \supseteq \mathcal{E}[\exists_{\leq} k\, r] \Rightarrow \mathcal{E}[\exists_{\leq} n\, r] \supseteq \mathcal{E}[\exists_{\leq} k\, s]$$

- $y \doteq \sqcap_i y_i$ and $\exists\, y_i : \mathcal{E}[c] \supseteq \mathcal{E}[y_i]$
  If y was returned as a successor because of one $y_i = \forall\, s.f$ satisfying the generation step, then the test condition $\hat{s}_y = \perp$ will still be satisfied because $\hat{s}_y$ takes into consideration all value restrictions that are inherited by y or spread over the various y_i.

  If $y_i = \exists_{\leq} k\, s$ then the test $k \leq n$ becomes invalid if the actual number restriction of s in y is greater than k. In the language $\mathcal{TF}^{\ominus}$ this is impossible without y becoming unsatisfiable if it contained or inherited a $y_j = \exists_{\geq} m\, s$ with $m \geq n \geq k$. But an unsatisfiable concept is a succesor of any concept.

This completes the proof of soundness for the base cases of c. In the following we investigate the impacts of c being a concept conjunction and y being a concept spezialization.

7. $c \doteq \sqcap_i c_i$

- $\sqcap_i c_i \equiv \top$
  Soundness and completeness of TAUT guarantee that $\mathcal{E}[c] = \mathcal{D}$ is discovered. The classifier returns $y = \top$ for which holds $\mathcal{E}[y] = \mathcal{D}$. Soundness follows because of $\mathcal{E}[c] = \mathcal{E}[y] = \mathcal{D} \Rightarrow \mathcal{E}[c] \supseteq \mathcal{E}[y]$.

- $\sqcap_i c_i \equiv \perp$
  Soundness and completeness of USAT guarantee that $\mathcal{E}[c] = \emptyset$ is discovered. The classifier returns $y = \perp$ for which holds $\mathcal{E}[y] = \emptyset$. Soundness follows because of $\mathcal{E}[c] = \mathcal{E}[y] = \emptyset \Rightarrow \mathcal{E}[c] \supseteq \mathcal{E}[y]$.

If $\sqcap_i c_i \not\equiv \top \not\equiv \perp$, the classifier generates the sets $\ell_{\uparrow c_i}$ for each c_i and computes their intersection in Step 11. Soundness of this step follows from the observation

$$\mathcal{E}[c_1] \supseteq \mathcal{E}[y]\, \wedge\, \ldots\, \wedge\, \mathcal{E}[c_n] \supseteq \mathcal{E}[y] \Rightarrow \cap_i \mathcal{E}[c_i] \supseteq \mathcal{E}[y] \Rightarrow \mathcal{E}[\sqcap_i c_i] \supseteq \mathcal{E}[y] \Rightarrow \mathcal{E}[c] \supseteq \mathcal{E}[y]$$

Now we have to investigate whether soundness of the individual generation steps can be affected because of interactions among value or number restrictions within c.

7a. $c = \forall\, r.d \sqcap c'$

If c' contains or inherits further value restrictions on r then the extension of the value restriction on r in c can be smaller than the extension of d, i. e. $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[d]$. This means, testing $\hat{s}_y \in \ell_{\uparrow d}$ would violate soundness and therefore the classifier uses the actual value restriction of r represented in $\hat{r}_c$. Soundness of the test $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$ has already been proven in Case 4.

7b. $c = \exists_{\geq} n\, r \sqcap c'$

Let us assume that c' contains or inherits a $c_j = \exists_{\geq} m\, r$.

- If $n = 0$ and $m \geq 1$ then $y = \top$ is no longer a valid successor. Soundness is preserved in Step 11 because y is not contained in the successor set of c_j because of soundness of Step 8.
- If $n \geq 1$ and $m > n$, the classifier returns $y = \exists_{\geq} k\, s$ satisfying $k \geq n$. If $k \leq m$ then y is not a valid successor of c because it is not a valid successor of c_j. Soundness is preserved in Step 11.

In both sublanguages under consideration, the actual number restriction of r can only be greater than n if such a concept c_j is contained in or inherited by c'. The language $\mathcal{TF}^{\ominus}$ excludes role hierarchies and thus it is impossible that r inherits additional role fillers over the role hierarchy. In $\mathcal{TF}^{\star}$, a role hierarchy is admitted, but no $\exists_{\leq}$ operator is provided, which makes it impossible to formulate disjointness on role fillers.

7c. $c = \exists_{\leq} n\, r \sqcap c'$

Soundness of Step 9 would be affected if the actual number restriction on s were smaller than n. In the language $\mathcal{TF}^{\ominus}$ without subroles this can only happen, when c' contains or inherits $c_j = \exists_{\leq} m\, r$ and $m \leq k \leq n$. But y satisfying the generation tests with respect to n would not satisfy it with respect to m, i. e. soundness is again preserved by Step 11 when the intersection of successor sets for $c_i = \exists_{\leq} n\, r$ and c_j is computed. Other possibilities for c to have a smaller number restriction than n on r are excluded in $\mathcal{TF}^{\ominus}$. We do not need to investigate the language $\mathcal{TF}^{\star}$ because it does not provide the $\exists_{\leq}$ operator.

8. $\boldsymbol{c} \mathrel{\dot{\leq}} \sqcap_i \boldsymbol{c}_i$

If c is a specialization, $y = \bot$ is returned. Soundness follows immediatley since $\mathcal{E}[\bot] = \emptyset$ and $\mathcal{E}[c] \supseteq \emptyset$ for arbitrary c according to Definition 9.

9. $\boldsymbol{y} \mathrel{\dot{\leq}} \sqcap_i \boldsymbol{y}_i$

Concept specializations y are returned by the classifier if one y_i satisfies $\mathcal{E}[c] \supseteq \mathcal{E}[y_i]$. Since $\mathcal{E}[\sqcap_i y_i] \supseteq \mathcal{E}[y]$ according to the semantics of terminological specializations and the rule $\mathcal{E}[c] \supseteq \mathcal{E}[y_i] \Rightarrow \mathcal{E}[c] \supseteq \mathcal{E}[\sqcap_i y_i]$ it follows $\mathcal{E}[c] \supseteq \mathcal{E}[y]$ for $y \mathrel{\dot{\leq}} \sqcap_i y_i$.

With that the proof of soundness of $\ell_{\uparrow c}$ and for the simple cases directly solved by $SUCC$ has been completed. ■

Completeness of $\ell_{\uparrow c}$

Proposition 6 *Let c, y be concept descriptions in a terminology $\mathcal{T}$ given in either $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$. $y \in \ell_{\uparrow c}$ if $\mathcal{E}[c] \supseteq \mathcal{E}[y]$ holds in all models of $\mathcal{T}$.*

We prove completeness indirectly by twofold induction over c and y. Instead of showing that $\mathcal{E}[c] \supseteq \mathcal{E}[y] \Rightarrow y \in \ell_{\uparrow c}$ in all models, we show that $y \notin \ell_{\uparrow c} \Rightarrow \mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ in some model by constructing a model in which an object is contained in the extension of y, but not in the extension of c. In some cases, the proof will also be based on general set-theoretic properties of extension functions.
Proof:

1. $c \doteq \top$

The classifier returns $SUCC(\top) := \{\top\}$ after preprocessing, with $\top$ representing the set of all tautological concepts $\{y \mid \mathcal{E}[y] = \mathcal{D}\}$ because of soundness and completeness of $TAUT$ and Definition 14. $SUCC(\top)$ is complete if all immediate successors of $\top$ are contained in this set. Since $TAUT(y) = \text{TRUE} \Leftrightarrow \mathcal{E}[y] = \mathcal{D}$ all concepts x that satisfy $TAUT(x) \neq \text{TRUE}$ satisfy $\mathcal{E}[x] \subset \mathcal{D}$. Applying Definition 14, such an x cannot be an immediate successor of $\top$ because of $x \not\equiv y$ (since $\mathcal{E}[y] \neq \mathcal{E}[x]$) but $x \sqsubseteq y$ because of $\mathcal{E}[x] \subseteq \mathcal{D}$ and $\mathcal{E}[y] = \mathcal{D}$.

2. $c \doteq \bot$

The classifier returns $SUCC(\bot) := \{\bot\}$ after preprocessing, with $\bot$ representing the set of all unsatisfiable concepts $\{y \mid \mathcal{E}[y] = \emptyset\}$. Soundness and Completeness of $USAT$ guarantee that $USAT(y) = \text{TRUE} \Leftrightarrow \mathcal{E}[y] = \emptyset$. Thus any concept x not returned by the classifier has a non-empty extension in all models and therefore $\mathcal{E}[c] \not\supseteq \mathcal{E}[x]$ because of $\mathcal{E}[c] = \emptyset$, but $\mathcal{E}[x] \neq \emptyset$.

3. $c \doteq a$ with a atomic

In Step 1 the classifier follows all possible paths in the taxonomy based on $\prec_{\mathcal{T}}$ relations. All concepts x that are not reached over these paths are not contained in $\ell_{\uparrow c}$. According to Definition 14 $x \not\prec_{\mathcal{T}} a$ if and only if either (1) $x \not\sqsubseteq a$ or (2) $x \sqsubseteq a$ and there is a concept z with $z \not\equiv x$ and $x \sqsubseteq z$ and $z \prec_{\mathcal{T}} a$. For (1) holds that $\mathcal{E}[x] \not\subseteq \mathcal{E}[a]$ according to the semantics of subsumption and it follows $\mathcal{E}[c] \not\supseteq \mathcal{E}[x]$ because of $\mathcal{E}[c] = \mathcal{E}[a]$. In the second case, it follows that $\mathcal{E}[c] \supseteq \mathcal{E}[x]$, i. e. the concepts x have to be added to $\ell_{\uparrow c}$ to achieve completeness. This is done in Step 10, when the classifier recursively follows the $\prec_{\mathcal{T}}$ relations down to $\bot$ and x is added to $\ell_{\uparrow c}$ when reaching the path starting in z.

4. $c \doteq \forall\, r.d$

The special case $\hat{r}_c = \top$ is covered by $TAUT$ during the preprocessing phase because of $TAUT(\forall\, r.\top) = \text{TRUE}$. Completeness follows from the proof for Case 1.
For the case $d \neq \top$ (i. e. $\mathcal{E}[d] \subset \mathcal{D}$ and therefore also $\mathcal{E}[c] \subset \mathcal{D}$), we proceed recursively over the structure of all y that are not returned in $\ell_{\uparrow c}$ by Steps 3 and 10 and show that $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ holds:

- $y \doteq \top$
  $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ because of $\mathcal{E}[\top] = \mathcal{D}$, but $\mathcal{E}[c] \subset \mathcal{D}$.

- $y \doteq a$ with a primitive
  We can construct a model with a domain element $\alpha \in \mathcal{E}[y]$, but $\alpha \notin \mathcal{E}[c]$. Such a domain element must exist because of $\mathcal{E}[c] \subset \mathcal{D}$.

- $y \doteq a$ with a defined
  Completeness is achieved by Step 10: a is not added to $\ell_{\uparrow c}$ only if it is a successor of a concept $x \notin \ell_{\uparrow c}$ and thus $\mathcal{E}[c] \not\supseteq \mathcal{E}[x]$. Since $\mathcal{E}[a] \subseteq \mathcal{E}[x]$ there is a model with $\mathcal{E}[c] \not\supseteq \mathcal{E}[a]$.

- $y \doteq \forall\, s.f$ and $s \notin \ell_{\downarrow r}$ or $\hat{s}_y \notin \ell_{\uparrow \hat{r}_c}$.
  If $s \notin \ell_{\downarrow r}$ then $\mathcal{E}[s] \not\supseteq \mathcal{E}[r](inductionhypothesis).Thus, weconstructamodel$E' with $\langle \alpha, \beta \rangle \in \mathcal{E}'[r]$, but $\langle \alpha, \beta \rangle \notin \mathcal{E}'[s]$ and $\mathcal{E}'[c] \not\supseteq \mathcal{E}'[y]$ follows.

  If $\hat{s}_y \notin \ell_{\uparrow \hat{r}_c}$, then $\mathcal{E}[\hat{r}_c] \not\supseteq \mathcal{E}[\hat{s}_y]$ (induction hypothesis) and a model similar to above can be constructed, i. e. $\mathcal{E}'[c] \not\supseteq \mathcal{E}'[y]$ follows.

- $y \doteq \exists_{\geq} n\, s$
  We construct a model with $\langle \alpha, \beta \rangle \in \mathcal{E}[r]$ and $\langle \alpha, \beta \rangle \in \mathcal{E}[s]$, but with $\beta \notin \mathcal{E}[d]$. Such an object β must exist because of $\mathcal{E}[d] \subset \mathcal{D}$ and $n \geq 0$.

- $y \doteq \exists_{\leq} 0\, s$ with $s \notin \ell_{\downarrow r}$
  There is a model with an object α for which no object β exists with $\langle \alpha, \beta \rangle \in \mathcal{E}[s]$, i. e. s has no role fillers in this model and $\alpha \in \mathcal{E}[y]$. But $\alpha \notin \mathcal{E}[c]$ because of $\langle \alpha, \delta \rangle \notin \mathcal{E}[r]$, which is possible since $\mathcal{E}[c] \subset \mathcal{D}$ and $\mathcal{E}[r] \not\subseteq \mathcal{E}[s]$.

- $y \doteq \exists_{\leq} n\, s$ with $n \geq 1$
  We construct a model with $\langle \alpha, \beta \rangle \in \mathcal{E}[r]$ and $\langle \alpha, \beta \rangle \in \mathcal{E}[s]$, but with $\beta \notin \mathcal{E}[d]$. Such an object β must exist because of $\mathcal{E}[d] \subset \mathcal{D}$ and $n \geq 1$.

Now we prove completeness for $y \doteq \sqcap_i y_i$ with $\forall\, y_i : y_i \notin \ell_{\uparrow c}$, i. e. none of the y_i is a successor, but y can be a successor in the following cases:

- $USAT(y) = $ TRUE
  Even if all y_i are satisfiable and $y_i \notin \ell_{\uparrow c}$ holds, their conjunction can become unsatisfiable and thus $\mathcal{E}[y] = \emptyset$, i. e. y would be a successor. Completeness of $USAT$ guarantees that all such y are discovered and since $\perp$ is added to $\ell_{\uparrow c}$ through Step 10 also y is represented in $\ell_{\uparrow c}$.

- $y \doteq \forall\, s.f \sqcap y'$
  Since y' can contain or inherit further value restrictions on s, the test $\hat{s}_y \in \ell_{\uparrow \hat{r}_c}$ is performed with $\hat{s}_y$ representing the actual value restriction on s in y according to Definition 16. Testing $f \in \ell_{\uparrow \hat{r}_c}$ would violate completeness since if $\mathcal{E}[d] \not\supseteq \mathcal{E}[f]$, it is still possible that $\mathcal{E}[d] \supseteq \mathcal{E}[\hat{s}_y]$ because of $\mathcal{E}[\hat{s}_y] \subseteq \mathcal{E}[f]$.

- $y \doteq \exists_{\leq} n\, s \sqcap y'$
  There is no possibility in $\mathcal{TF}^{\ominus}$ that the actual number restriction on s is 0 if y contains or inherits only number restrictions that are greater or equal to 1 because the language contains no subroles. If y' contains or inherits $y_i = \exists_{\geq} k\, s$ with $k \geq n$, it becomes unsatisfiable. Completeness of $USAT$ guarantees that unsatisfiability of y is detected when y was added to the taxonomy. Therefore, y will be added to $\ell_{\uparrow c}$ in Step 10 when $\perp$ is added to the successor set.

Completeness for concept specializations $y \stackrel{\cdot}{\leq} \sqcap_i y_i$ is proven similarly. A specialization y is only not considered as a successor if $\forall y_i : \mathcal{E}[c] \not\supseteq \mathcal{E}[y_i]$. It follows that $\mathcal{E}[c] \not\supseteq \mathcal{E}[\sqcap_i y_i]$ unless $\sqcap_i y_i$ is unsatisfiable or the concept description $\sqcap_i y_i$ is a successor because of $y \stackrel{\cdot}{\leq} \forall s.f \sqcap y'$ and $\mathcal{E}[\hat{r}_c] \supseteq \mathcal{E}[\hat{s}_y]$ since $c \doteq \forall r.d$. In the first case, y is not a valid concept specialization, which is detected during preprocessing and y is not added to the taxonomy. In the second case, completeness is achieved because $\forall s.f$ satisfies the generation rule and thus there is one y_i that is a successor of c and y is therefore added as a successor.

5. $c \doteq \exists_{\geq} n\, r$

For the case $n = 0$, Step 7 returns $\top$ as a successor and Step 10 adds all successors of $\top$, i. e. all concepts in $\mathcal{T}$.
For the case $n \geq 1$ we proceed recursively over the structure of all y that are not returned in $\ell_{\uparrow c}$ by Steps 8 and 10 and show that $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ holds:

- $y \doteq \top$
  Since $\mathcal{E}[c] \subset \mathcal{D}$, but $\mathcal{E}[\top] = \mathcal{D}$ it follows $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$.

- $y \doteq a$ and a primitive
  We can construct a model with a domain element $\alpha \in \mathcal{E}[y]$, but $\alpha \notin \mathcal{E}[c]$. Since $\mathcal{E}[c] \subset \mathcal{D}$ such an element must exist.

- $y \doteq a$ and a defined
  Then a must not be a successor of a concept $x \in \ell_{\uparrow c}$ according to Step 10, i. e. a is only a successor of non-successors z for which we know $\mathcal{E}[c] \not\supseteq \mathcal{E}[z]$. Since $\mathcal{E}[z] \supseteq \mathcal{E}[a]$ there is model in which $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$.

- $y \doteq \forall s.d$
  We can construct a model in which either some pairs $\langle \alpha, \beta \rangle$ are in $\mathcal{E}[s]$, but not in $\mathcal{E}[r]$ or where s has less than n role fillers.

- $y \doteq \exists_{\geq} k\, s$ and $s \notin \ell_{\uparrow r}$ or $k < n$
  If $s \notin \ell_{\uparrow r}$ then there are pairs $\langle \alpha, \beta \rangle$ in $\mathcal{E}[s]$, but not in $\mathcal{E}[r]$ and thus we can construct a model with a domain object $\alpha \in \mathcal{E}[y]$, but $\alpha \notin \mathcal{E}[c]$.

  If $k < n$ then we can construct a model $\mathcal{E}[c]$ with a domain object $\alpha \in \mathcal{E}[y]$ that has less than n, but at least k role fillers for s. Obviously, α would not be in the extension of c. Such an element α must exist because y and c are both satisfiable and $\mathcal{E}[c] \subset \mathcal{D}$.

- $y \doteq \exists_{\leq} k\, s$
  Since s can have zero to k role fillers, but $n \geq 1$, any model in which s has zero role fillers implies $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$.

Finally, we investigate concept conjunctions and concept specializations again. If $y \doteq \sqcap_i y_i$ and $\forall y_i : y_i \notin \ell_{\uparrow c}$, completeness of $USAT$ guarantees that unsatisfiability of y is detected and y is added as a successor in Step 10. For $y_i = \exists_{\geq} k\, s$, it would be critical if the actual number restriction of s in y is smaller than k. One possibility is that y contains or inherits a $y_j = \exists_{\leq} m\, s$ and $m < n$. But in this case, y is unsatisfiable and completeness of $USAT$ for $\mathcal{TF}^{\ominus}$ has been proven. If y contains or inherits a $y_j = \exists_{\geq} m\, s$ with $m < n$, then the actual value restriction of s in y is $max(k, m)$, i. e. no decrease occurs. Other possibilities are excluded in the two languages under consideration. If y is a specialization and $\sqcap_i y_i$

is unsatisfiable, then y is not contained in the taxonomy. If $y \dot{\leq} \exists_{\geq} k\, s \sqcap y'$ the above argumentation applies in the same way.

6. $\boldsymbol{c \doteq \exists_{\leq} n\, r}$

Independent of n, concepts y of the structure $y \doteq \top$, $y \doteq a$ and a primitive, $y \doteq a$ and a defined are not contained in $\ell_{\uparrow c}$. The completeness proof proceeds identical to the previous case.
For the special case $n = 0$ we know that c can only have models where no pairs $\langle \alpha, \beta \rangle$ are in $\mathcal{E}[r]$, while the concepts $y \doteq \forall s.d$ with $\hat{s}_y \neq \bot$, $y \doteq \exists_{\geq} k\, s$, and $y \doteq \exists_{\leq} k\, s$ with $k \geq 1$ can all have models were pairs $\langle \alpha, \beta \rangle$ are in $\mathcal{E}[s]$, i. e. $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ follows.
For the case $n \geq 1$, we have to analyse concepts y of the following syntactical structure that are not contained in $\ell_{\uparrow c}$:

- $y \doteq \forall\, s.d$ with $s \notin \ell_{\downarrow r}$ or $\hat{s}_y \neq \bot$
  If $s \notin \ell_{\downarrow r}$ then $\mathcal{E}[r] \not\subseteq \mathcal{E}[s]$. Thus there is a model in which r and s are satisfied by different object pairs, and thus there is a domain object α in the extension of y, but not in the extension of c.

  If $\hat{s}_y \neq \bot$ we can construct a model for y, in which an object has more than n role fillers for s and all these fillers also satisfy r. This object is in the extension of y, but not in the extension of c because it has too many role fillers for r.

- $y \doteq \exists_{\geq} k\, s$
  We construct a model in which a domain object has more than n role fillers for s, and all fillers would also be in r. This object is not in the extension of c because it has too many role fillers for r.

- $y \doteq \exists_{\leq} k\, s$ with $s \notin \ell_{\downarrow r}$ or $k > n$
  If $s \notin \ell_{\downarrow r}$ then $\mathcal{E}[r] \not\subseteq \mathcal{E}[s]$. Thus there is a model in which r and s are satisfied by different object pairs, and thus there is a domain object α in the extension of y, but not in the extension of c.

  If $k > n$ we construct a model in which a domain object α in the extension of y must have at least $n + 1$ role fillers for s and all these fillers are also in the extension of r. Then α has too many role fillers for r and therefore cannot be in the extension of c.

If $y \doteq \sqcap_i y_i$ and $\forall\, y_i : y_i \notin \ell_{\uparrow c}$, then y is a successor if $USAT(y) =$ TRUE or if the actual number restriction of a role s would be greater than k. Completeness for the first case is again achieved by completeness of $USAT$. The actual value restriction of an atmost number restriction cannot change in $\mathcal{TF}^{\ominus}$, because even if y contains or inherits bigger atmost number restriction on the same role, the actual number restriction is $min(k, m)$. For specializations $y \dot{\leq} \sqcap_i y_i$ the analysis proceeds in the same way.

7. $\boldsymbol{c \doteq \sqcap_i c_i}$

A concept y is not contained in $\ell_{\uparrow c}$ if y is not a successor of at least one c_i. In this case, we know that $\mathcal{E}[c_i] \not\supseteq \mathcal{E}[y]$, which implies $\mathcal{E}[\sqcap_i c_i] \not\supseteq \mathcal{E}[y]$ because of $\mathcal{E}[c_i] \supseteq \mathcal{E}[\sqcap_i c_i]$.

- $c \doteq \forall\, r.d \sqcap c'$
  The extension of the actual value restriction of r can be smaller than $\mathcal{E}[d]$ because

$\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[d]$ in both languages under consideration. Completeness would not be affected because $\hat{s}_y \in \ell_{\uparrow d}$ is sufficient for $\hat{s}_y \notin \ell_{\uparrow \hat{r}_c}$.

- $c \doteq \exists_{\geq} n\, r \sqcap c'$
  Additional concepts could become successors if the number restriction on r is smaller than n. In the two languages this can only happen if c contains or inherits number restrictions $\exists_{\geq} m\, r$ or $\exists_{\leq} m\, r$ with $m < n$. In the first case, the actual number restriction is $max(k, m)$. In the second case, c becomes unsatisfiable and completeness of $USAT$ applies.

- $c \doteq \exists_{\leq} n\, r \sqcap c'$
  Additional concepts could become successors if the number restriction on r is bigger than n. Similar to the previous case, if c contains or inherits a number restriction $\exists_{\leq} m\, r$ with $m > n$, then the actual number restriction is $min(m, n)$. If c contains or inherits a number restriction $\exists_{\geq} m\, r$ with $m > n$ then c becomes unsatisfiable, which is discovered by $USAT$.

8. $c \dot{\leq} \sqcap_i c_i$

If c is a specialization and $\sqcap_i c_i$ is satisfiable, then the classifier returns only $\perp$ as a successor and with that all concepts that are equivalent to it. Transformation into normal-form terminologies leads to $c \doteq \sqcap_i c_i \sqcap \bar{c}$ with $\bar{c}$ remaining undefined in $\mathcal{T}$. Since no classified concept y can have an undefined concept term in its definition, $\mathcal{E}[\bar{c}] \not\supseteq \mathcal{E}[y]$ and with that $\mathcal{E}[c] \not\supseteq \mathcal{E}[y]$ for arbitrary y unless $\mathcal{E}[y] = \emptyset$. ■

Finally, we have to prove that the reduction operation in Definition 19 preserves completeness. Because of soundness and completeness of $\ell_{\uparrow c}$ we know $y \in \ell_{\uparrow c} \Leftrightarrow \mathcal{E}[c] \supseteq \mathcal{E}[y]$. The reduction operation removes now all concepts from the set that are successors of another concept in the set, i. e. x is removed if there is a y with $\mathcal{E}[x] \neq \mathcal{E}[y]$ and $\mathcal{E}[y] \supset \mathcal{E}[x]$. Using Definition 14, it follows that x is not an immediate successor of c, i. e. only non-immediate successors are removed from the set and completeness follows.

Soundness and Completeness of PREC

We prove soundness of $\ell_{\downarrow c}$ directly using again structual induction over c and y.

Soundness of $\ell_{\downarrow c}$

Proposition 7 *Let c, y be concept descriptions in $\mathcal{TF}$. $y \in \ell_{\downarrow c}$ only if $\mathcal{E}[c] \subseteq \mathcal{E}[y]$ in all models of $\mathcal{T}$.*

Proof:

1. $c \doteq \top$

This case is covered by the preprocessing phase, which returns $PREC(c) = \{\top\}$. According to Definition 9 and soundness of TAUT we know that $\mathcal{E}[c] = \mathcal{D}$ and $\mathcal{E}[y] = \mathcal{D}$ and therefore $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \subseteq \mathcal{E}[y]$.

2. $c \doteq \bot$

The preprocessing phase returns $PREC(c) = \{\bot\}$. According to Definition 9 and soundness of USAT we know that $\mathcal{E}[c] = \emptyset$ and $\mathcal{E}[y] = \emptyset$ and again $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \subseteq \mathcal{E}[y]$.

3. $c \doteq a$ with a atomic

Step 12 first returns $y \succ_{\mathcal{T}} a$ with $\mathcal{E}[a] \subseteq \mathcal{E}[y]$ because of Definition 14. Since $\mathcal{E}[c] = \mathcal{E}[a]$, it follows $\mathcal{E}[c] \subseteq \mathcal{E}[y]$. Subsequently, all predecessors x of a are added, i. e. $\mathcal{E}[a] \subseteq \mathcal{E}[x]$ and thus $\mathcal{E}[c] \subseteq \mathcal{E}[x]$ due to transitivity of set inclusion.

4. $c \doteq \forall r.d$

Note that c is not a conjunction and that therefore $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d]$.
Step 16 addresses the case $\hat{r}_c = \bot$. We know that $USAT(d)$ must have returned true and thus that $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d] = \emptyset$. We proceed by structural induction over y.

- $y = \forall s.f$ and $s \in \ell_{\uparrow r}$
  The condition $s \in \ell_{\uparrow r}$ implies $\mathcal{E}[s] \subseteq \mathcal{E}[r]$, furthermore $\mathcal{E}[f] \supseteq \mathcal{E}[d]$ because of $\mathcal{E}[d] = \emptyset$. Soundness follows from the observation for any extension function $\mathcal{E}$:

$$\mathcal{E}[d] \subseteq \mathcal{E}[f] \wedge \mathcal{E}[r] \supseteq \mathcal{E}[s] \Rightarrow \forall r.d \subseteq \forall r.f \Rightarrow \forall r.d \subseteq \forall s.f$$

- $y = \exists_{\leq} k\, s$ and $s \in \ell_{\uparrow r}$
  We now that $\mathcal{E}[s] \subseteq \mathcal{E}[r] \Rightarrow \mathcal{E}[\forall r.\bot] = \mathcal{E}[\exists_{\leq} 0\, s]$ and that $\mathcal{E}[\exists_{\leq} 0\, s] \subseteq \mathcal{E}[\exists_{\leq} k\, s]$ for arbitrary k. With that we can conclude $\mathcal{E}[s] \subseteq \mathcal{E}[r] \Rightarrow \mathcal{E}[\forall r.\bot] \subseteq \mathcal{E}[\exists_{\leq} k\, s]$ and soundness follows.

Step 17 addresses the case $\hat{r}_c \neq \bot$ and the classifier generates predecessors $y = \forall s.f$ with $s \in \ell_{\uparrow r}$ and $\hat{s}_y \in \ell_{\downarrow \hat{r}_c}$. It holds $\mathcal{E}[\hat{s}_y] = \mathcal{E}[f]$ and $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d]$ due to the syntactic restrictions on c and y. With the induction hypothesis we have $\mathcal{E}[s] \subseteq \mathcal{E}[r]$ and $\mathcal{E}[d] \subseteq \mathcal{E}[f]$ and soundness follows from the observation

$$\mathcal{E}[d] \subseteq \mathcal{E}[f] \wedge \mathcal{E}[s] \subseteq \mathcal{E}[r] \Rightarrow \mathcal{E}[\forall s.d] \subseteq \mathcal{E}[\forall s.f] \Rightarrow \mathcal{E}[\forall r.d] \subseteq \mathcal{E}[\forall s.f]$$

We investigate the case of y being a concept conjunction separately in Case 9.

5. $c = \exists_{\geq} n\, r$

The case $n = 0$ is addressed in Step 18, which returns the empty set, followed by Step 23, which adds $\top$ as a predecessor. Since $\mathcal{E}[\exists_{\geq} 0\, r] = \mathcal{D}$ and $\mathcal{E}[\top] = \mathcal{D}$ it follows $\mathcal{E}[c] = \mathcal{E}[y]$ which implies $\mathcal{E}[c] \subseteq \mathcal{E}[y]$.
If $n \geq 1$ generation proceeds according to Step 19, which returns $y = \exists_{\geq} k\, s$ with $s \in \ell_{\downarrow r}$ and $k \leq n$. Due to the syntactic restrictions on c and y the numbers n and k represent the actual number restrictions on r and s. Soundness follows from the observation

$$\mathcal{E}[r] \subseteq \mathcal{E}[s] \wedge k \leq n \Rightarrow \mathcal{E}[\exists_{\geq} n\, r] \subseteq \mathcal{E}[\exists_{\geq} k\, s]$$

6. $\boldsymbol{c = \exists_{\leq} n\, r}$

Independent of n, generation Steps 20 and 21 return

- $y = \exists_{\leq} k\, s$ with $s \in \ell_{\uparrow r}$ and $k \geq n$
  Due to the syntactic restrictions on c and y, the numbers n and k represent the actual number restrictions on r and s. Soundness follows from the observation

$$\mathcal{E}[s] \subseteq \mathcal{E}[r] \wedge k \geq n \Rightarrow \mathcal{E}[\exists_{\leq} n\, r] \subseteq \mathcal{E}[\exists_{\leq} n\, s] \subseteq \mathcal{E}[\exists_{\leq} k\, s]$$

If $n = 0$ Step 20 additionally generates

- $y = \forall\, s.f$ with $s \in \ell_{\uparrow r}$.
  We know that $\mathcal{E}[\exists_{\leq} 0\, r] = \mathcal{E}[\forall r.\bot]$ according to Definition 9. Furthermore, $\mathcal{E}[\forall r.\bot] \subseteq \mathcal{E}[\forall\, r.f]$ for abitrary value restrictions f. Since $\mathcal{E}[s] \subseteq \mathcal{E}[r]$ (induction hypothesis) we can conclude $\mathcal{E}[\forall r.\bot] \subseteq \mathcal{E}[\forall\, s.f]$ and thus it follows

$$\mathcal{E}[s] \subseteq \mathcal{E}[r] \Rightarrow \mathcal{E}[\exists_{\leq} 0\, r] \subseteq \mathcal{E}[\forall\, s.f]$$

7. $\boldsymbol{c \doteq \sqcap_i c_i}$

- $\sqcap_i c_i \equiv \top$
  The preprocessing phase returns $y = \top$. Soundness of $TAUT$ guarantees that $\mathcal{E}[c] = \mathcal{D}$ and thus $\mathcal{E}[c] = \mathcal{E}[y] = \mathcal{D}$, which implies $\mathcal{E}[c] \subseteq \mathcal{E}[y]$.

- $\sqcap_i c_i \equiv \bot$
  The preprocessing phase returns $y = \bot$. Soundness of $USAT$ guarantees that $\mathcal{E}[c] = \emptyset$ and thus $\mathcal{E}[c] = \mathcal{E}[y] = \emptyset$, which implies $\mathcal{E}[c] \subseteq \mathcal{E}[y]$.

If $\sqcap_i c_i \not\equiv \top \not\equiv \bot$, the classifier generates the predecessors with respect to a single c_i and returns their union. Soundness follows from the observation

$$\mathcal{E}[c_i] \subseteq \mathcal{E}[y] \Rightarrow \cap_i \mathcal{E}[c_i] \subseteq \mathcal{E}[y] \Rightarrow \mathcal{E}[\sqcap_i c_i] \subseteq \mathcal{E}[y] \Rightarrow \mathcal{E}[c] \subseteq \mathcal{E}[y]$$

Again we investigate whether the validity of local tests on single c_i is affected if c contains or inherits additional value or number restrictions.

7a. $c \doteq \forall\, r.d \sqcap c'$

If c contains or inherits additional value restrictions over c' then $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[d]$. Steps 16 tests $\hat{r}_c = \bot$ and Step 17 tests $\hat{s}_y \in \ell_{\downarrow \hat{r}_c}$. Performing the same tests on d instead of on $\hat{r}_c$ would not violate soundness: If $\mathcal{E}[d] = \emptyset$ then $\mathcal{E}[\hat{r}_c] = \emptyset$ and $\mathcal{E}[d] \subseteq \mathcal{E}[\hat{s}_y]$ implies $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[\hat{s}_y]$

7b. $c \doteq \exists_{\geq} n\, r \sqcap c'$

We do not need to analyse the case $n = 0$, because the classifier returns an empty predecessor set in Step 18. If $n \geq 1$, the classifier returns $y = \exists_{\geq} k\, s$ satisfying $k \leq n$. If the actual number restriction on r were smaller than n, then this test would become invalid:

- If c' contains or inherits a concept $c_j = \exists_{\geq} m\, r$ and $m < n$ then the actual number restriction on r is $max(m, n)$, i. e. no decrease occurs.

- If c' contains or inherits a concept $c_j = \exists_{\leq} m\, r$ and $m < n$ then c itselfs becomes unsatisfiable and any concept is a predecessor of it.

Other possibilities are excluded in the two languages under consideration.

7c. $c \doteq \exists_{\leq} n\, r \sqcap c'$

Independent of the value of n, the Steps 20 and 21 would generate invalid predecessors if the actual number restriction on r would be greater than n:

- If c' contains or inherits a concept $c_j = \exists_{\leq} m\, r$ and $m > n$ then the actual number restriction on r is $min(m, n)$, i. e. no increase occurs.
- If c' contains or inherits a concept $c_j = \exists_{\geq} m\, r$ and $m > n$ then c itselfs becomes unsatisfiable and any concept is a predecessor of it.

Other possibilities are excluded in the two languages under consideration.

8. $\boldsymbol{c} \mathrel{\dot{\leq}} \sqcap_i \boldsymbol{c}_i$

The special cases $\sqcap_i c_i \equiv \bot$ and $\sqcap_i c_i \equiv \top$ are addressed during preprocessing. In the first case, c is not a valid concept specialization. In the second case, $y = \top$ is returned for which soundness is obvious because of $\mathcal{E}[c] \subseteq \mathcal{D}$ for any concept c.
If $\sqcap_i c_i \not\equiv \bot \not\equiv \top$ then $\ell_{\uparrow c}$ returns predecessors y with $\mathcal{E}[\sqcap_i c_i] \subseteq \mathcal{E}[y]$. Since $\mathcal{E}[c] \subseteq \mathcal{E}[\sqcap_i c_i]$ we can conclude $\mathcal{E}[c] \subseteq \mathcal{E}[y]$ and soundness follows.
This justifies to treat concept specializations identically to concept definitions, i. e. a valid predecessor of the defined concept is also a predecessor of the corresponding specialization.

9. $\boldsymbol{y} \doteq \sqcap_i \boldsymbol{y}_i$

For all previous cases we have always assumed that y is not a conjunction, i. e. that $\mathcal{E}[c_i] \subseteq \mathcal{E}[y]$ holds which implied $\mathcal{E}[c] \subseteq \mathcal{E}[y]$. If y is a conjunction the classifier tests that y is covered by c as defined in Definition 21 and the covering condition on y is sufficient to maintain soundness since it requires that each y_i is a valid predecessor of c, i. e. $\forall\, y_i : \mathcal{E}[c] \subseteq \mathcal{E}[y_i]$ and thus, $\mathcal{E}[c] \subseteq \cap_i \mathcal{E}[y_i]$.
Finally, we have to exclude the possibility that a test $\mathcal{E}[c] \subseteq \mathcal{E}[y_i]$ is invalidated by interactions among the various conjuncts contained in y.

9a. $y \doteq \forall\, s.f \sqcap y'$

If y' inherits or contains additional value restrictions on s, then the actual value restriction $\hat{s}_y$ on s can be smaller than f, i. e. $\mathcal{E}[\hat{s}_y] \subseteq \mathcal{E}[f]$. Since Steps 16 and 17 test $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[\hat{s}_y]$ soundness is maintained, while testing $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[f]$ would lead to incorrect predecessors.

9b. $y \doteq \exists_{\geq} k\, s \sqcap y'$

If y' inherits or contains a number restriction $y_j = \exists_{\geq} m\, s$ with $m > k$ then indeed the actual number restriction on s would be $max(m, k)$ and y is no longer a valid predecessor until $\mathcal{E}[c] \subseteq \mathcal{E}[y_j]$. But since the covering condition has to be verified on y_j, soundness is maintained because y is only kept as a predecessor when y_j is a predecessor.
Other possibilities of an increase of the actual number restriction are excluded in the two languages under consideration.

9c. $y \doteq \exists_{\leq} k\, s \sqcap y'$

The test $k \geq n$ in Step 21 would become invalid if the actual number restriction on s in y is smaller than k. This can happen if y' inherits or contains a number restriction $y_j = \exists_{\leq} m\, s$ with $m < k$. Indeed the actual number restriction on s would be $min(m,k)$ and y is no longer a valid predecessor until $\mathcal{E}[c] \subseteq \mathcal{E}[y_j]$, which is verified when testing the covering condition for y_j and thus soundness is maintained.
If y' inherits or contains a number restriction $y_j \exists_{\geq} m\, s$ with $m > k$ then y becomes unsatisfiable. The covering condition requires that both conjuncts of y have to be predecessors of c, which is impossible until c is unsatisfiable itself. Since $\ell_{\uparrow c}$ is only activated for satisfiable concept descriptions c, only one of the conjuncts can be a valid predecessor, i. e. the covering condition for at least one conjunct is violated and thus y would not be returned as a predecessor.
Other possibilities of a decrease of the actual number restriction are excluded in the language $\mathcal{TF}^{\ominus}$.

10. $\boldsymbol{y} \mathrel{\dot{\leq}} \sqcap_i \boldsymbol{c}_i$

A concept specialization y is only returned as a predecessor if it can be reached following the $\succ_{\mathcal{T}}$ links starting in at least one c_i. In this case, $\mathcal{E}[c_i] \subseteq \mathcal{E}[y]$ holds according to Definition 14 and thus $\cap_i \mathcal{E}[c_i] \subseteq \mathcal{E}[y]$, i. e. soundness folllows.

■

Completeness of $\ell_{\downarrow c}$

Proposition 8 *Let c, y be concept descriptions in a terminology $\mathcal{T}$ given in either $\mathcal{TF}^{\ominus}$ or $\mathcal{TF}^{\star}$. $y \in \ell_{\downarrow c}$ if $\mathcal{E}[c] \subseteq \mathcal{E}[y]$ holds in all models of $\mathcal{T}$.*

We prove completeness again indirectly by structural induction over c and y and prove that whenever $y \notin \ell_{\downarrow c}$ then $\mathcal{E}[c] \nsubseteq \mathcal{E}[y]$ in some model of $\mathcal{T}$.

Proof:

1. $\boldsymbol{c} \doteq \top$

The preprocessing phase returns $PREC(\top) = \{\top\}$. Due to soundness and completeness of TAUT, we know that $TAUT(c) = \text{TRUE} \Leftrightarrow \mathcal{E}[c] = \mathcal{D}$. The predecessor $\{\top\}$ represents the set of all tautological conceps $\{y \,|\, TAUT(y) = \text{TRUE}\}$, i. e. for all y not returned in this set holds $\mathcal{E}[y] \subset \mathcal{D}$ because of soundness and completeness of $TAUT$. Thus completeness follows with $\mathcal{E}[c] \nsubseteq \mathcal{E}[y]$ because of $\mathcal{E}[c] = \mathcal{D}$.

2. $\boldsymbol{c} \doteq \bot$

The preprocessing phase returns $PREC(\bot) = \{\bot\}$. $PREC$ is complete if all immediate predecessors of $\bot$ are contained in this set. The predecessor $\bot$ represents all unsatisfiable concepts contained in $\mathcal{T}$, i. e. the set $\{y \,|\, USAT(y) = \text{TRUE}\}$. Because of soundness and completeness of $USAT$ we know that $USAT(y) = \text{TRUE} \Leftrightarrow \mathcal{E}[y] = \emptyset$ and furthermore $\mathcal{E}[c] = \emptyset$. All other concepts x with $\mathcal{E}[x] \supset \emptyset$ cannot be immediate predecessors because of Definition 14.

3. $c \doteq a$ with a atomic

If c is atomic and thus contained as a node in the taxonomy, the classifier starts collecting predecessors in a and following $\succ_{\mathcal{T}}$ links until $\top$ is reached using Step 12 and then adding $\top$ at the end of the generation in Step 23. The set of direct predecessors is complete according to Definition 14 because the classifier performs an exhaustive search over $\succ_{\mathcal{T}}$. To see that the generation of indirect predecessors is complete when starting search at direct predecessors note that all successors x of non-predecessors z of c cannot be predecessors of c: Let us assume that z is a non-predecessor of c, i. e. $\mathcal{E}[c] \not\subseteq \mathcal{E}[z]$, but x is a successor of z and thus $\mathcal{E}[x] \subseteq \mathcal{E}[z]$. Clearly, $\mathcal{E}[c] \not\subseteq \mathcal{E}[x]$ follows and thus x is not a predecessor.
Thus, the search performed by the classifier during the generation of indirect predecessors is complete. It terminates if either $\perp$ is reached, because $\perp$ cannot be a predecessor of c since c is satisfiable, i. e. $\mathcal{E}[c] \supset \emptyset$ and thus $\mathcal{E}[c] \not\subseteq \emptyset$, or if a non-predecessor is reached because all successors of non-predecessors cannot be predecessors due to the above argumentation.
An indirect predecessor is not a valid predecessor if it violates the covering condition in Definition 21, i. e. if it is a concept specialization or a successor of a non-predecessor.
To see that a concept specialization $y \dot{\leq} \sqcap_i y_i$ cannot be a valid indirect predecessor we consider its transformation into a defined concept $y \doteq \sqcap_i y_i \sqcap \bar{y}$, which is extension-preserving. $\bar{y}$ remains completely undefined in the terminology and can only be a predecessor of c if it is reached following the $\succ_{\mathcal{T}}$ links. But in this case, y must be a direct predecessor of c. Otherwise, y is a successor of the non-predecessor $\bar{y}$.
With that, completeness for the generation of direct and indirect predecessors follows with the classifier starting in c and performing a complete search over the $\succ_{\mathcal{T}}$ links and then starting in all valid direct predecessors and performing a complete search of all their successors until a non-predecessor is reached. It remains to prove that the test for the covering condition for non-atomic y_i and the generation of predecessors for non-atomic c_i is complete.

4. $c \doteq \forall\, r.d$

Note that due to the syntactic restrictions on c we have $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d]$.
For the special case $\hat{r}_c = \perp$ (i. e. $\mathcal{E}[d] = \emptyset$), concepts y of the following syntactical structure are not returned by the classifier in Steps 16 and 23: $y = \perp$, $y = a$ with a primitive, $y = \exists_{\geq} k\, s$ and $y = \forall\, s.f$ with $s \notin \ell_{\uparrow r}$. Since $\emptyset \subset \mathcal{E}[c] \subset \mathcal{D}$ completeness follows because of $\mathcal{E}[c] \not\subseteq \mathcal{E}[y]$ in at least one model for $\mathcal{T}$ for all these y: For $y = \perp$ it holds that $\mathcal{E}[y] = \emptyset$ and in all other cases we can construct a model in which an object α is contained in the extension of c, but not in the extension of y. Since $\mathcal{E}[c] \subset \mathcal{D}$ such an object must exist.

- If $y = a$ and a is not primitive then y is either a defined concept or introduced through a specialization. A concept specialization cannot be a valid predecessor unless it is a direct predecessor as we have proven previously.
- If y is defined (i. e. $y \doteq \sqcap_i y_i$) then y is not a predecessor if one of its y_i does not satisfy the covering condition from Definition 21. But in this case, y is a successor of a non-predecessor and therefore $\mathcal{E}[c] \not\subseteq \mathcal{E}[y]$.

The impact of a conjunctive y on number and value restrictions contained in y does not need to be investigated for $c \doteq \forall\, r.d$ since solely subsumption between roles is tested and thus the generation result of the classifier is independent of the actual number or value restriction on a role. For all remaining syntactical possibilities we do not again address the case that y is a specialization or violates the covering condition, since it should be obvious that it cannot be a predecessor in this case.

If $\hat{r}_c \neq \bot$ ($\mathcal{E}[d] \neq \emptyset$) generation proceeds according to Steps 17 and 23.

- If $\mathcal{E}[d] = \mathcal{D}$ then $\mathcal{E}[c] = \mathcal{D}$ and completeness follows as in the case of $c \doteq \top$ because Step 23 returns $\top$ representing all y with $\mathcal{E}[y] = \mathcal{D}$.

- If $\mathcal{E}[d] \subset \mathcal{D}$ then $\emptyset \subset \mathcal{E}[c] \subset \mathcal{D}$ and concepts y of the following syntactical structure are not returned by the classifier in Steps 16 and 23: $y = \bot$, $y = a$ with a primitive, $y = \exists_{\geq} k\, s$, $y = \exists_{\leq} k\, s$ and $y = \forall\, s.f$ with $s \notin \ell_{\uparrow r}$ or $\hat{s}_y \notin \ell_{\downarrow \hat{r}_c}$. The proof of completeness is identical to the case $\hat{r}_c = \bot$ with the two additional cases of $y = \exists_{\leq} k\, s$ and $y = \forall\, s.f$ with $\hat{s}_y \notin \ell_{\downarrow \hat{r}_c}$.

If y is a concept conjunction, we only need to investigate whether it can be a predecessor even if its value restricting concept $\hat{s}_y$ does violate the test $\hat{s}_y \in \ell_{\downarrow \hat{r}_c}$. This cannot be the case. Note that $\mathcal{E}[\hat{r}_c] = \mathcal{E}[d]$, but that $\mathcal{E}[\hat{s}_y] \subseteq \mathcal{E}[f]$. To achieve completeness, only testing $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[f]$ would be sufficient, because a concept y satisfying $\mathcal{E}[\hat{r}_c] \nsubseteq \mathcal{E}[f]$ for its value restriction can never satisfy $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[\hat{s}_y]$. But as we have seen in the proof of soundness for $\ell_{\downarrow c}$ using the actual value restriction on s in y is necessary to maintain soundness.

5. $c \doteq \exists_{\geq} n\, r$

If $n = 0$ Step 23 returns $\top$ and since $\mathcal{E}[c] = \mathcal{D}$ completeness follows as proven in Case 1.
If $n \geq 1$ then $\mathcal{E}[c] \subset \mathcal{D}$ and the following y are not returned by $\ell_{\downarrow c}$: $y = \bot$, $y = a$ with a primitive, $y = \exists_{\leq} k\, s$, $y = \forall\, s.f$, and $y = \exists_{\geq} k\, s$ with $s \notin \ell_{\downarrow r}$ or $k > n$. Because of $\emptyset \subset \mathcal{E}[c] \subset \mathcal{D}$ we can construct a model with an object α contained in the extension of c, but not in the extension of y. For example, if $y = \exists_{\geq} k\, r$ with $k > n$ then α with exactly n role fillers would be in the extension of c, but not in the extension of y which requires for all objects to have at least $n + 1$ role fillers for r.
If y is a concept conjunction then y not returned by the generation steps could be a predecessor if its actual number restriction is smaller than k. In the two languages under consideration, this cannot happen. If y inherits or contains a $y_j = \exists_{\geq} m\, s$ with $m < k$ then the actual number restriction on s is $max(m, k)$, i. e. no decrease occurs. If y inherits or contains a $y_j = \exists_{\leq} m\, s$ with $m < k$ then y becomes unsatisfiable and is not a predecessor of c because c is satisfiable.

6. $c \doteq \exists_{\leq} n\, r$

Independent of n, the following y are not returned by the classifier: $y = \bot$, $y = a$ with a primitive, $y = \exists_{\leq} k\, s$ with $s \notin \ell_{\uparrow r}$ or $k < n$, and $y = \exists_{\geq} k\, s$. Because of $\emptyset \subset \mathcal{E}[c] \subset \mathcal{D}$ we can construct a model with an object α contained in the extension of c, but not in the extension of y for all the syntactical cases (except $y = \bot$) and completeness follows. If $y = \bot$ then $\mathcal{E}[y] = \emptyset$ and $\mathcal{E}[c] \nsubseteq \mathcal{E}[y]$ follows because of $\mathcal{E}[c] \supset \emptyset$.
If $n = 0$ the classifier does not return $y = \forall\, s.f$ with $s \notin \ell_{\uparrow r}$. With the induction hypothesis, we know that $\mathcal{E}[s] \nsubseteq \mathcal{E}[r]$ and a model can be constructed in which some role fillers for s for an object α exist.
If $n \geq 1$ no value restrictions $y = \forall\, s.f$ are returned and we can construct a model with an object $\alpha \in \mathcal{E}[c]$, but $\alpha \notin \mathcal{E}[y]$, e. g. where at least one of α's role fillers is not in $\mathcal{E}[f]$.

7. $c \doteq \sqcap_i c_i$

A concept y is not returned as a predecessor of a conjunctive c if

1. y is a spezialization not reachable following $\succ_{\mathcal{T}}$ links
2. there is no c_i such that y is a predecessor of it
3. y does not satisfy the covering condition

Completeness for the first case has already been proven. For the second case, we know $\forall c_i : \mathcal{E}[c_i] \not\subseteq \mathcal{E}[y]$ and thus it follows $\cap_i \mathcal{E}[c_i] \not\subseteq \mathcal{E}[y]$ and therefore $\mathcal{E}[c] \not\subseteq \mathcal{E}[y]$.
For the third case, completeness follows from $\exists y_i : \mathcal{E}[c] \not\subseteq \mathcal{E}[y_i] \Rightarrow \mathcal{E}[c] \not\subseteq \cap_i \mathcal{E}[y_i] \Rightarrow \mathcal{E}[c] \not\subseteq \mathcal{E}[y]$. Since the classifier is testing $\mathcal{E}[c_i] \not\subseteq \mathcal{E}[y_i]$ for all c_i we have to make sure that the tests performed are sufficient to generate all predecessors if interactions among value or number restrictions in a concept conjunction occur:

- $c \doteq \forall r.d \sqcap c'$
  The extension of the actual value restriction of r can be smaller than $\mathcal{E}[d]$ because $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[d]$ in both languages under consideration. Completeness would be affected when testing $\hat{s}_y \in \ell_{\downarrow d}$, because if $\mathcal{E}[d] \not\subseteq \mathcal{E}[\hat{s}_y]$ it is still possible that $\mathcal{E}[\hat{r}_c] \subseteq \mathcal{E}[\hat{s}_y]$. Using the actual value restriction is therefore necessary and achieves completeness.

- $c \doteq \exists_{\geq} n\, r \sqcap c'$
  Additional concepts could become predecessors if the actual number restriction on r would be greater than n. If c' contains or inherits $\exists_{\geq} m\, r$ the actual number restriction on r is indeed $max(m, n)$, i. e. an increase occurs. But these predecessors are generated when $\ell_{\downarrow c'}$ is computed and they are returned in the final set of predecessors $\ell_{\downarrow c}$ when satisfying the covering condition.

  Other possibilities of an increase cannot occur in the languages under consideration.

- $c \doteq \exists_{\leq} n\, r \sqcap c'$
  If the actual number restriction on r is smaller than n, completeness would be affected. In the two languages under consideration this can only happen if c' contains or inherits $\exists_{\leq} m\, r$ with $m < n$, since a decrease occurs with $min(m, n)$. But again these predecessors are generated when $\ell_{\downarrow c'}$ is computed and they are returned in the final set of predecessors $\ell_{\downarrow c}$ when satisfying the covering condition.

Completeness for the special case that c is tautological or unsatisfiable follows from completeness of $TAUT$ and $USAT$.

8. $c \stackrel{\cdot}{\leq} \sqcap_i c_i$

If c is a specialization then the classifier returns all predecessors of the corresponding defined concept $c \doteq \sqcap_i c_i$. This set would be incomplete if there are predecessors of a specialization that are not predecessors of the defined concept. To see that this is impossible let us consider the normal-form representation of c with $c = \sqcap_i c_i \sqcap \bar{c}$. The concept $\bar{c}$ cannot have any predecessors except of $\top$ because it remains completely undefined in $\mathcal{T}$ and a concept y can only be a predecessor of $\bar{c}$ unless $\bar{c}$ is contained in the definition of y or inherited by y. But then c must already have been defined in $\mathcal{T}$, i. e. the terminology would violate the uniqueness assumption and also contain terminological cycles, which we do not admit.

■

With that completeness of $\ell_{\downarrow c}$ has been proven. Finally we have to show that the reduction operation from Definition 23 on $\ell_{\downarrow c}$ only removes non-immediate predecessors. All concepts in $\ell_{\downarrow c}$ satisfy $\mathcal{E}[c] \subseteq \mathcal{E}[y]$. To obtain PREC, concepts x with $\mathcal{E}[x] \neq \mathcal{E}[y]$ and $\mathcal{E}[y] \subseteq \mathcal{E}[x]$ are removed. Thus $\mathcal{E}[y] \subset \mathcal{E}[x]$ follows and therefore x cannot be an immediate predecessor according to Definition 14.

References

[1] J.A. Allen, R. Fikes, and E. Sandewall, editors. *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann, San Mateo, 1991.

[2] F. Baader and B. Hollunder. KRIS: Knowledge representation and inference system. *SIGART Bulletin*, 2(2):8–15, 1991.

[3] F. Baader, B. Hollunder, B. Nebel, H.-J. Profitlich, and E. Franconi. An empirical analysis of optimization techniques for terminological representation systems, or making KRIS get a move on. In B. Nebel, W. Swartout, and C. Rich, editors, *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 270–281. Morgan Kaufmann, San Mateo, 1992.

[4] A. Borgida and P. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[5] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In Allen et al. [1], pages 151–162. extended version available as DFKI report RR-95-07.

[6] J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. *Artificial Intelligence*, 68(2):367–397, 1994.

[7] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In Allen et al. [1].

[8] R. MacGregor. A description classifier for the predicate calculus. In *Proceedings of the 12th National Conference of the American Association for Artificial Intelligence.* AAAI Press, MIT Press, 1994.

[9] B. Nebel. Computational complexity of terminological reasoning in BACK. *Artificial Intelligence*, 34:371–383, 1988.

[10] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems.* Lecture Notes in Artificial Intelligence 422. Springer, 1990.

[11] H. J. Ohlbach and J. Koehler. Reasoning about sets via atomic decomposition. Research report, ICSI, 1996. In Preparation.

[12] H. J. Ohlbach, R. Schmidt, and U. Hustadt. Translating graded modalities into predicate logic. Research Report MPI-I-95-2-008, Max-Planck-Institute of Computer Science, 1995. To appear in H. Wansing (ed), Proof Theory for Modal Logic, Oxford Univ. Press.

[13] K. Stoffel, W. Anderson, and J. Hendler. PARKA: Support for extremely large knowledge bases. In *Proceedings of the KRUSE Symposium on Knowledge Retrieval, Use and Storage for Efficiency*, 1995.

[14] K. Stoffel and J. Hendler. PARKA on MIMD-supercomputers. *Parallel Processing for Artificial Intelligence*, pages 132–142, 1995.

[15] K. Stoffel, S. Sharma, J. Hendler, and J. Saltz. Integrating task-parallel computations into data-parallel applications. In *Proceedings of the SIPAR-95 Workshop*, 1995.