

A Brief Introduction to ECG Workbench

(Draft)

Author: Luca Gilardi lucag@berkeley.edu

ECG Workbench Website: <http://www.icsi.berkeley.edu/~lucag>

NOTE: This document is relative to an older version of ECG Workbench, which now is undergoing significant changes. A new version of the HOWTO will be available when the Workbench code will have reached a new fixed point. For the time being, this document should help you get the flavor of what ECG Workbench can do for you. Please refer to the ECG Workbench Website above for the latest news.

Welcome to ECG Workbench. This document is intended as a quick introduction to the main features of the program. It assumes some knowledge of ECG (Embodied Construction Grammar, see reference at the end of this document for more information).

ECG Workbench is a support for the grammar writer. Very broadly speaking, with ECG Workbench you can:

- Open an existing grammar and navigate its structure;
- Create and edit a grammar, checking its syntactic and structural correctness;
- Parse (or Analyze, in ECG terms) sentences, producing a semantic specification (SemSpec) that, among other things, can be printed and embedded in your documents.

How to install ECG Workbench

At this time, no automated installation procedure is available (but one will be shortly). For now, your only option is to download the archive file for your platform, expand it in a place where you can easily reach it (for instance on your desktop or your user directory). The executable for your platform requires that a Java Virtual Machine, version 1.5 or better, be already installed on your system. You can download the Java Virtual Machine Standard Edition version 1.6 from Sun by clicking [here](http://java.sun.com/javase/downloads/). Older versions are available from <http://java.sun.com/javase/downloads/>. The available platforms are:

- Microsoft Windows Intel 32 bit (any recent version supporting Java 1.5, including Windows 2000, XP, and Vista);
- Apple Mac OS X Intel and PPC (any recent version should work);
- Linux Intel 32 bit on GTK (any distribution supporting Java 1.5+)

The archive file (a zip file) contains a directory named `ecg-workbench-<version number>.zip` (you might be unable to see the extension on certain platforms/configurations). This directory (after having unzipped the archive) contains the executable file, the one marked with the blue ball icon. To start ECG Workbench, double-click on the blue icon.

How to open a grammar file set

ECG Workbench (which I'll sometimes refer to as EW or "the workbench") assumes that you will be working on a single set of grammar files (or units) at a time. An ECG grammar is usually defined in more than one file: the rationale is that keeping different constructions in different files helps in keeping the grammar organized and thus easily comprehensible. Some files may define lexical items (or different categories of lexical items) for instance, other files may describe phrases or particular kinds of phrases, other files may describe clauses, and so on.

The workbench, however, doesn't assume any particular criteria for breaking up the grammar. As a way to group all the grammar files and see them as a single unit, ECG Workbench uses a metafile (the preferences, or prefs, file) describing the grammar files and various other parameters used by the ECG analyzer. The preferences file is a plain text file containing pointers to various elements making up the grammar: the folder containing the actual grammar files, the file extensions for different types of files, and even some example sentences. Although you probably will very seldom need to deal with a preferences file, a description of its main aspects is contained below.

Assuming that you've just installed ECG Workbench, and that you have already launched it as described above, select Grammar | Open Preferences File... and navigate to one of the example grammars you have downloaded. To open the `starter` grammar (contained in the file `starter.zip`, which should be downloadable off of the same web page on which you found this file (or <http://www.icsi.berkeley.edu/~lucag/>) and the zip file containing EW itself, select `starter.prefs`, and click OK. Depending on your platform, you should see something similar to the following:

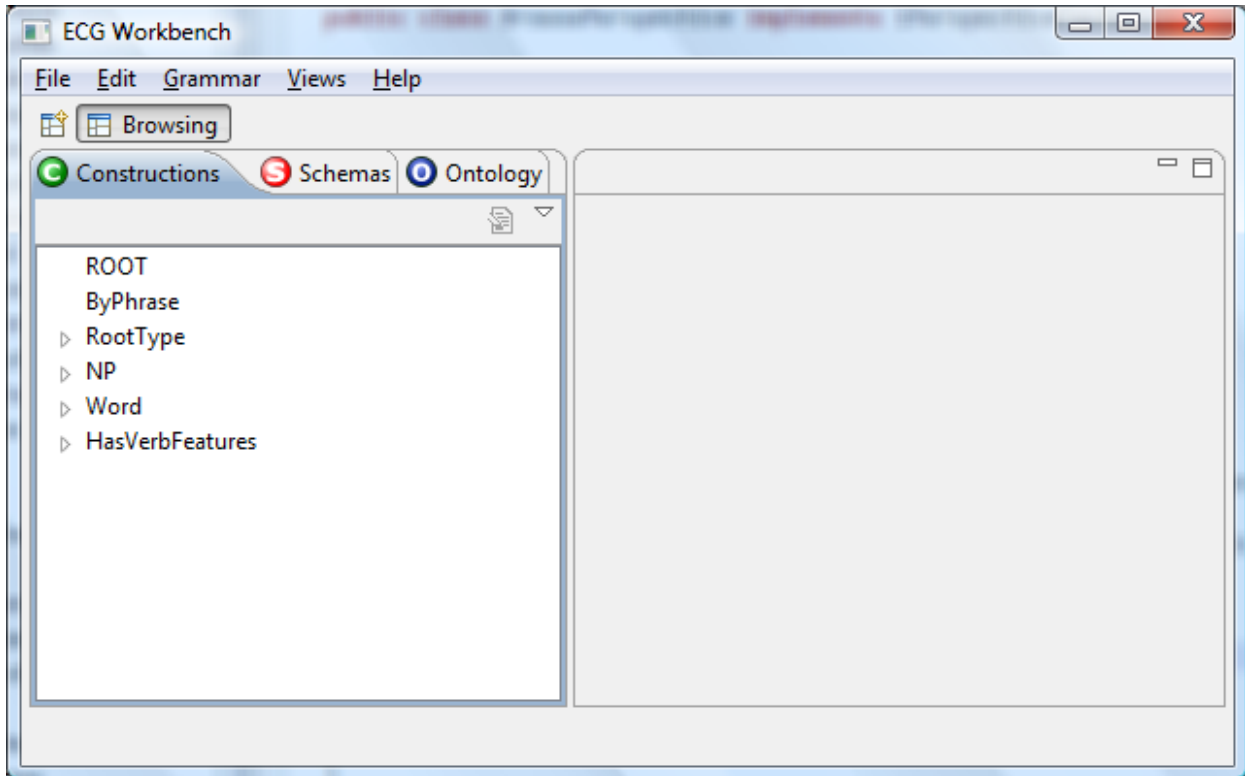


Figure 1

Perspectives


The Workbench is organized into Perspectives, currently only Browsing (fig. 1) and Analysis. (I'm using capitalized words to signal that they have a special meaning in the context of the workbench). You can return to Browsing by clicking the button just below the tool bar. The Browsing Perspective shows just the Views (Constructions, Schemas, Ontology) you can see in the picture above; their layout is fixed and they cannot be closed.

Analyzing a sentence

There are two different ways to start analyzing (i.e., parsing) a sentence. One is to open the Analyzer View in Browsing. The other, probably simpler, is to change to Analysis Perspective and let EW open the Analyzer View for you.

Changing to Analysis perspective will open many more Views, and if you don't have a high-resolution screen, the workbench window may become too cluttered. If so, it's probably advisable to open only the Analyzer View by selecting Views | Open View | Other..., open the ECG folder, select Analyzer, and click OK.

Changing to the Analysis perspective

Click on the  icon at the far left just below the menu bar, choose Other.... A dialog will pop up; choose Analysis. This should be the result, after stretching the window a bit to enlarge it:

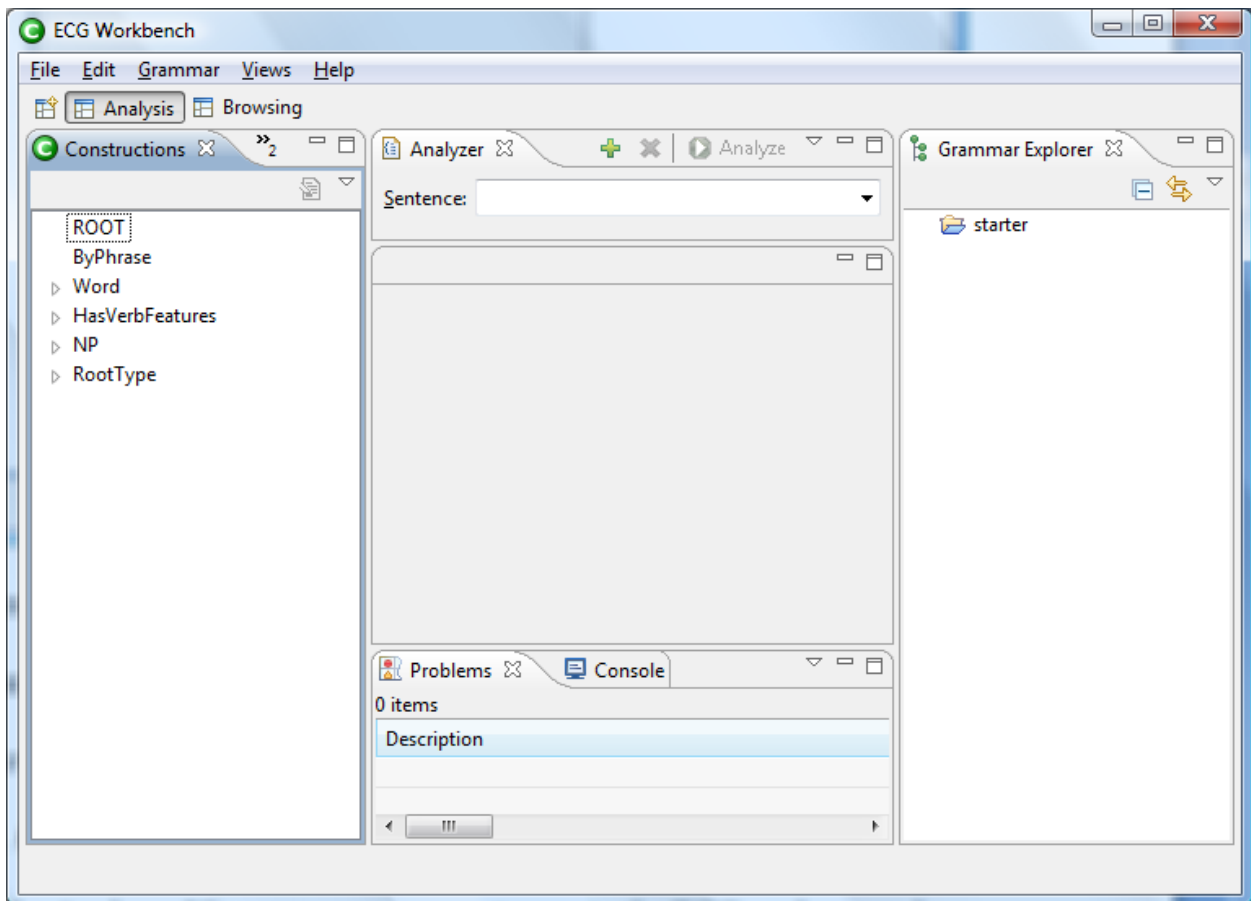


Figure 2

A few more View tabs have been opened. At the bottom, there are two new Views: Problems and Console. The former lists any errors that may happen during editing and the grammar checking

phase (more about this below). The Console View is mainly for debugging purposes: here the workbench will signal internal problems it cannot deal with. At the top you can see the Analyzer View. This is the only new View that you'll see if you manually opened the Analyzer View. Through the Analyzer View you can input sentences into the ECG analyzer and obtain a semantic specification (or more than one in some cases). If the preferences file specifies examples sentences—which is the case if you opened `starter.prefs`—you can choose one from the drop-down list. Click on the down arrow at the far right of Sentence edit box, select for instance “he slapped the box” and click the Analyze button just above the drop-down list. After a few seconds a new Editor (the tabbed windows appearing in the central part of the workbench window are called Editors) containing the analysis should appear in the central part. The new Editor's title will remind you that its content is relative to the sentence you just analyzed:

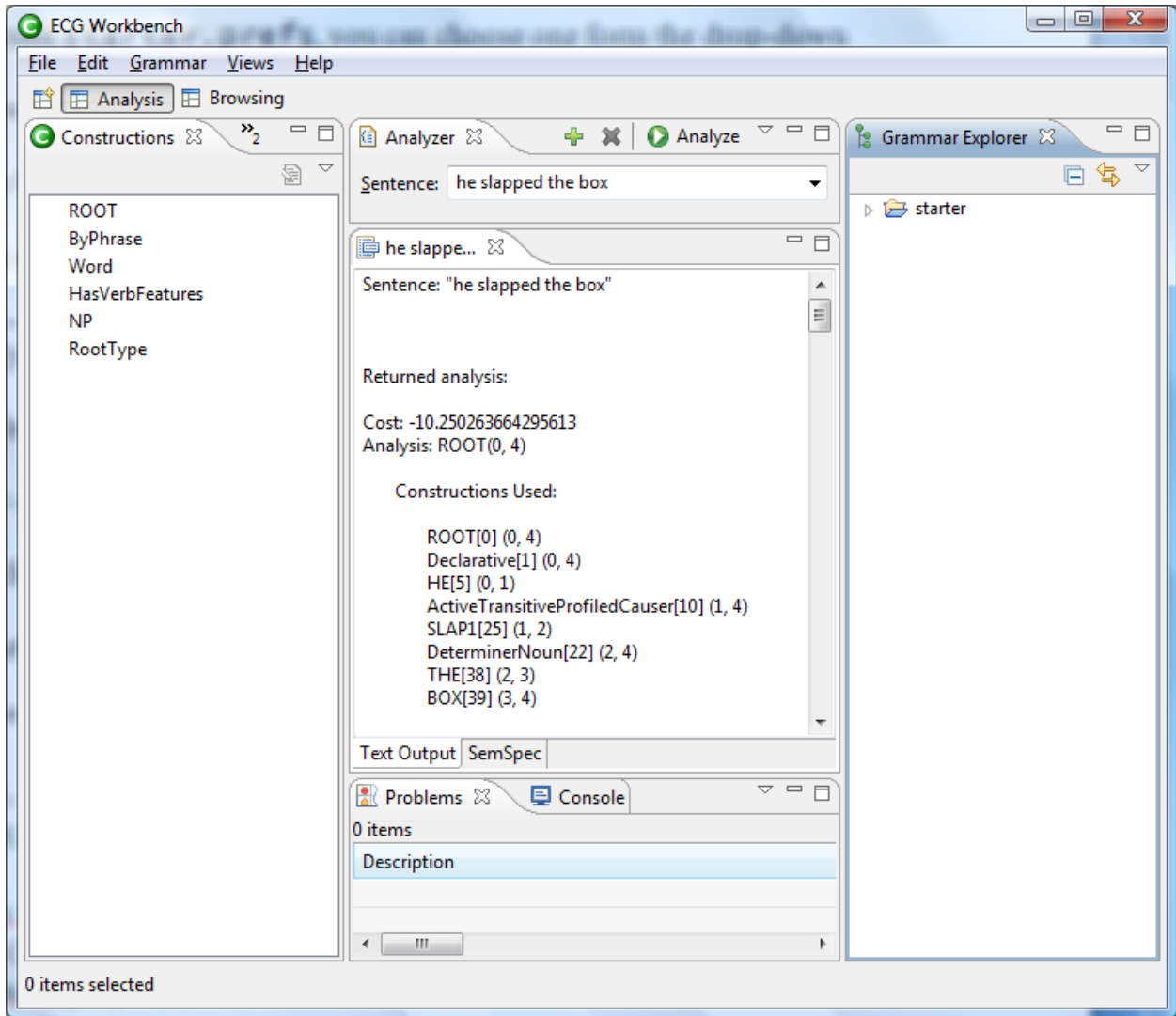


Figure 3

Editors and Views

An EW Editor looks like a View since it's a tabbed window like all Views. The difference is that Editors are the result of operations you perform on elements in the workbench window (like typing in a sentence and hitting the Analyze button, or clicking on a node in the Construction or Schemas Views) rather than selecting from a menu. Editors, unlike Views, permit editing, and are created automatically by the workbench depending on the operations you perform on its elements. There can be any number of the same kind of Editors open at the same

time, whereas only one instance of a certain kind of View (say for instance the Constructions View) can be open at any given time.

More on how Editors are managed in the section below, *Editor management*.

Back to sentence analysis

As you can see (fig. 3), at the bottom of the new Editor window there are two tabbed panes. The one that is displayed by default (“Text Output”) is a textual view that shows all the constructions and schemas used. For example, for the sentence you just analyzed, you can see that the Constructions used are the following:

```
ROOT[0] (0, 4)
Decl arati ve[1] (0, 4)
HE[14] (0, 1)
Acti veTransi ti veProfi l edCauser[11] (1, 4)
SLAP1[30] (1, 2)
Determi nerNoun[22] (2, 4)
THE[36] (2, 3)
BOX[38] (3, 4)
```

The numbers in parentheses are the “spans” of the sentence that the relative construction recognized, starting from zero: ₀he ₁sl apped ₂at ₃the ₄box. Thus the Decl arati ve construction was used to recognize the entire sentence (it covers the words from 0 to 4), whereas Acti veTransi ti veProfi l edCauser spans words 1 to 4, and so on for the other Constructions. From this output, drawing the tree typical of the generative grammar tradition should be straightforward. After the Constructions Used section, there’s a section showing the Schemas involved in the analysis, and a final section containing all the unification bindings. The SemSpec tab opens a more graphical view of these last two sections.

The SemSpec tab (also at the bottom of the Analysis Editor) for “he slapped at the box” contains the semantic specification, that is, the Schemas involved in the analysis of the sentence together with their relationships. The format employed is very similar to that used for printing the classic attribute-valued matrices that’s common in the literature on unification grammars like HPSG, but it includes some additional features.

If you have clicked on the SemSpec tab, you might be seeing see only a part of it (again, depending on your screen’s resolution):

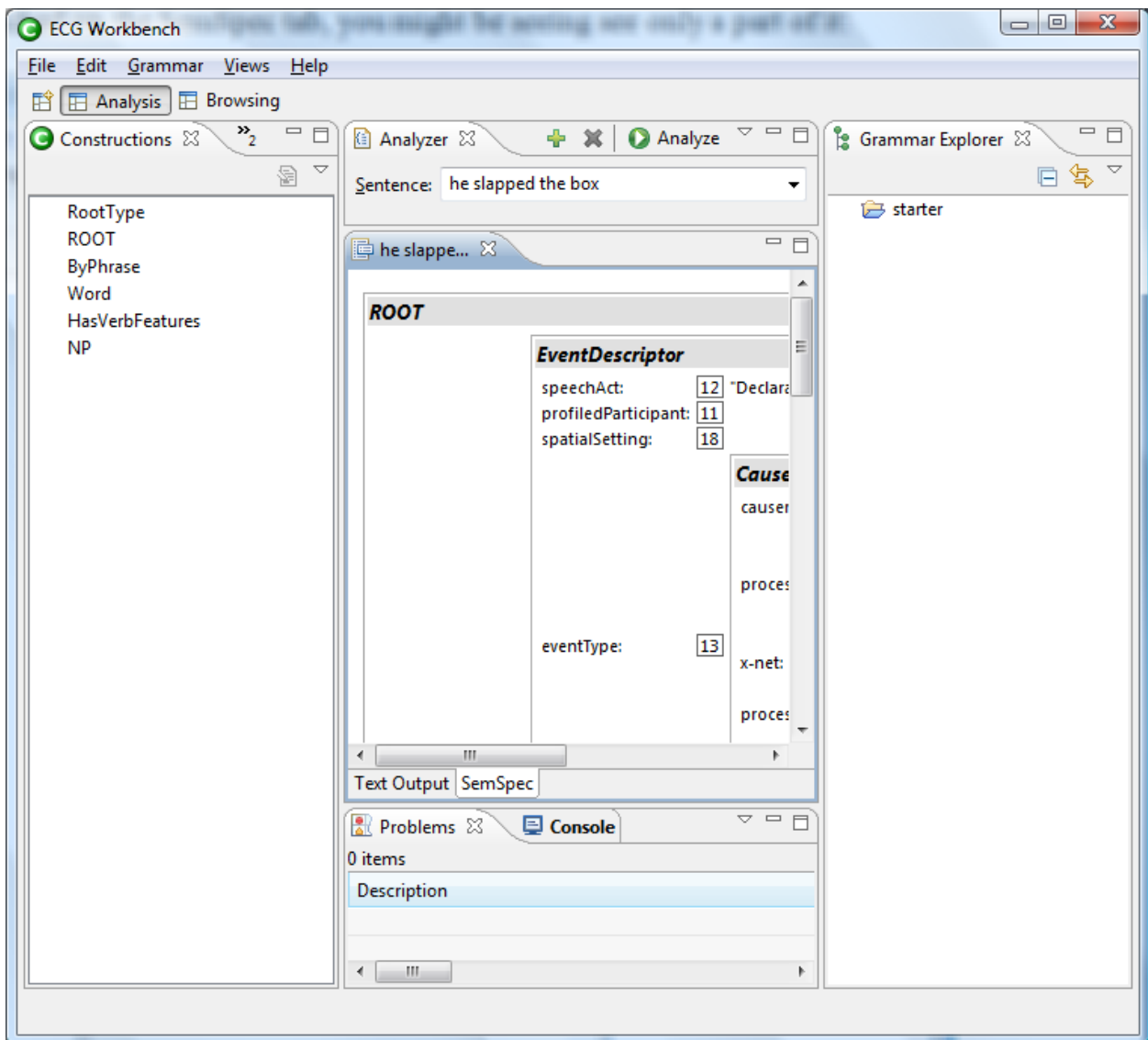


Figure 4

To see the contents more easily, you can maximize the Editor window. There are four different ways to do this: you can click on the maximize icon (☐) in the same frame that contains the analysis results Editor, you can double-click on the caption (the blue tab containing the beginning of the sentence), you can right-click on the caption and choose Minimize, or hit Ctrl+M (Option+M on the Mac). Here is what you should see:

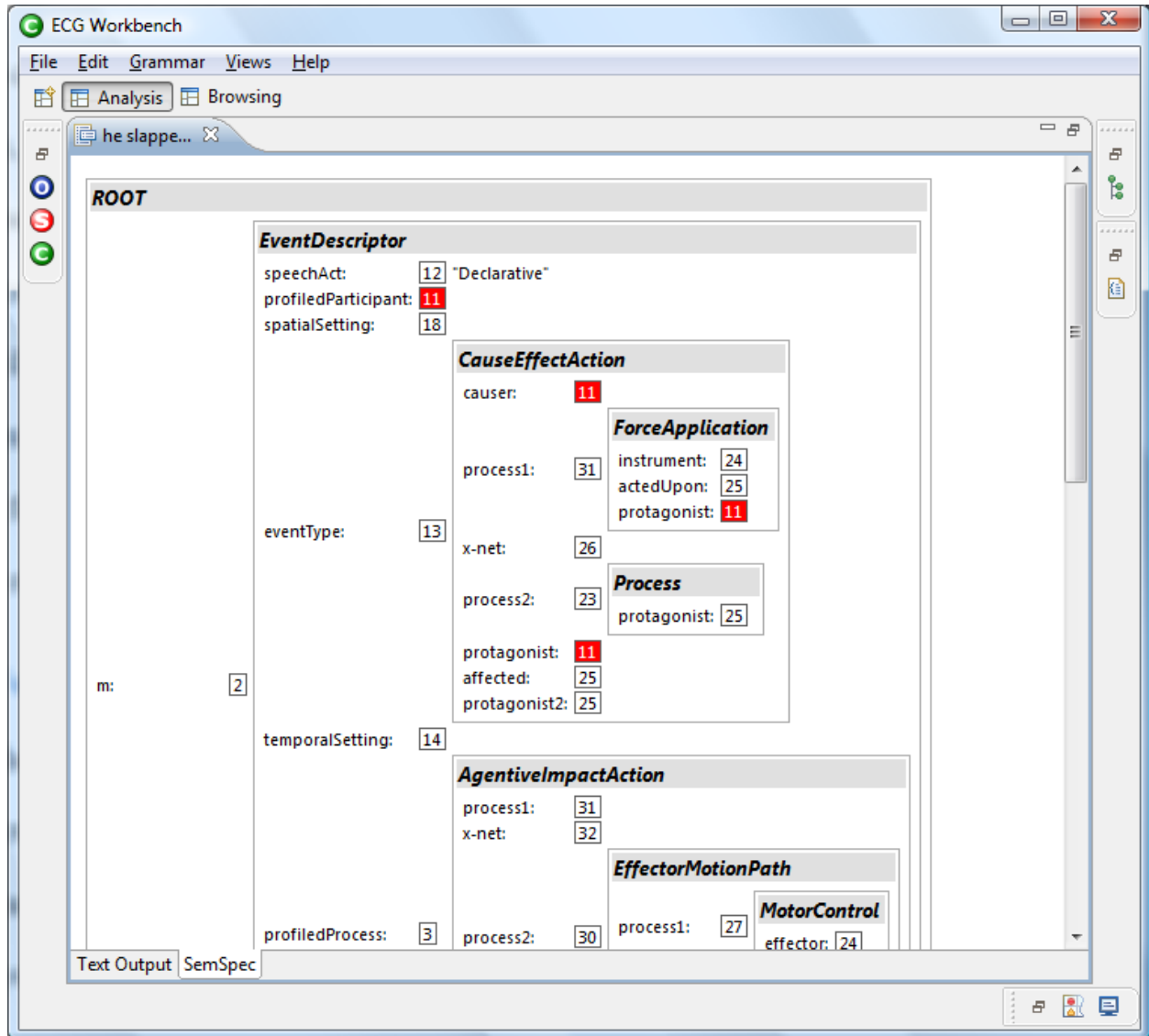



Figure 5

As you can see, some of the coindexed elements are highlighted in red. This doesn't happen automatically, though: you have to click on a boxed number, and the ones that have the same

value will light up as well. If you click on the 25 next to `protagonist` in the Process schema, all the other coindexed roles will be colored in red (you'll have to scroll down to see them all). Another feature of this SemSpec representation is that you can collapse any of the elements by clicking on their headers (the grey bars containing the Schema names). If you click again on the grey bar, the element will expand again. From here you can print, by right-clicking anywhere in the SemSpec window and selecting Print. To restore Analysis result window to its original size, you can hit again Ctrl +M (or Option+M on the Mac), double-click on the title, right-click and select Restore, or click on the restore icon ().

Analyzing a sentence not in the list

Of course you can analyze other sentences, provided that the required lexical items and constructions are defined by the grammar. Later on, in the section *Exploring and modifying the grammar*, I'll explain how to add a lexical item. For now, let's use what is already defined by the starter grammar. You can type a new sentence in the Analyzer Editor, replacing the previous sentence. Let's click on that sentence and type in a new one that will work with the starter grammar: `"the cake slid into the box"` (without quotes), hit enter (this will have the workbench remember this sentence in the list), and then Analyze. After a few moments a new Editor pane will show up with the results of the analysis (the old one will remain available as a tab).

Editor management

If you analyzed the two sentences as described above, you'll see the two Editors containing the results of each analysis are still there. Editors are created by EW and never closed automatically. The tabs that accumulate in the upper part have an "X" icon that you can use to

close them. At the far right, a small “>>” icon signals that there are more Editors than the workbench window can fit as tabs. If you click on the “>>” icon, a drop-down list containing all the Editors’ captions will appear. If you have a lot, typing the first letter(s) of the captions’ names will select only those beginning with the letters you typed.

If you’re used to Web browsers, you might expect a Forward and Back button to navigate back and forth to the “places” you’ve visited. There are no such buttons, but the fact that the Editors remain accessible should help you keep track of what you did. Again, the list of all the open Editors that is always available through the “>>” icon (or by pressing Ctrl+E or Option+E on the Mac) also should help navigation.

Browsing and navigation

To simplify things a bit, let’s click again on the Browsing button (below the menu bar). The Constructions View on the left shows all the constructions that are available—from all the files making up the grammar—in a hierarchical view. Clicking on an arrow on the left of an element (not on the element itself) will expand it and show that element’s subcases. For instance, if you click on the arrow at the left of NP, you’ll see that it contains WH-NP, NounHeadedNP, PossessivePronoun, Pronoun, and SpecifierNP. This is because these are all subcases of NP. Besides the Constructions View, others are available, by just clicking on their tabs. The Schemas View contains a similar hierarchical view for all the available schemas defined in the grammar. And similarly for the Ontology View.

If you go back to the Constructions View and click on the NP item itself, you’ll see that its actual definition will be shown on the right, where a new tabbed window—a new Editor showing NP in its caption—will show up:

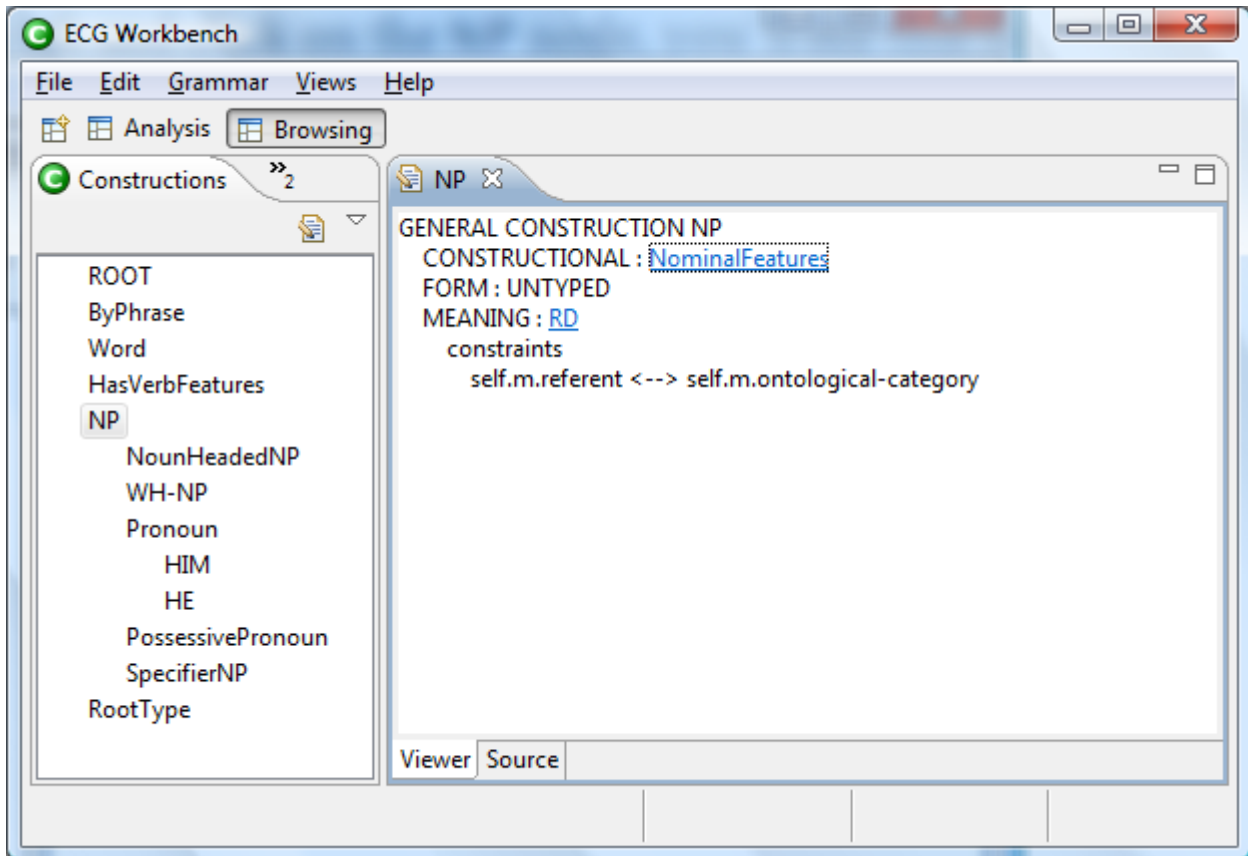


Figure 5

Let's look at the NP Editor window. There are two aspects to be especially aware of:

- First, if you click on Source tab near the bottom, you'll see that NP is not alone, but is followed by various other constructions. This is because the Source tab is a view on the actual file, while the Viewer tab shows the content of a single construction or schema in isolation.
- Second, as you can see from the left View, NP is not a subcase of any other construction. However, in general the Viewer tab shows more information for Constructions and Schemas that are subcases of other Constructions or Schemas. To see what happens in a different case, try clicking on the PRONOUN node on the left. Using the Viewer tab you'll see something like the following:

GENERAL CONSTRUCTION Pronoun

```
SUBCASE OF NP, Word
CONSTRUCTI ONAL: Nomi nal Features
FORM: WordForm
MEANI NG: RD
  constraints
    sel f. m. referent <--> sel f. m. ontol ogical -category // i nheri ted from NP
```

But the actual file content (click on the Source tab to see it) is a lot shorter:

```
general constructi on Pronoun
subcase of NP, Word
```

The reason is that the Viewer tab shows all the inherited elements from the construction that Pronoun is a subcase of. More specifically, it shows that Pronoun inherits a constructional block of type Nomi nal Features, a form pole of type WordForm, a meaning pole of type RD, and the meaning constraints (the one shown is inherited from NP, as the comment says) are also shown explicitly.

Another difference is in the case: in the first example, keywords (like **constructi on**, **subcase**, and so on) are all uppercase, while in the second they are lowercase. This is because some software packages that are part EW take quite literally the ECG specification, which requires all the keywords be case-insensitive. Therefore, you are free as well to use upper- or lowercase (and even mixed-case) letters for the various ECG keywords. Notice though that all type identifiers (Pronoun, NP, Word, etc.) are instead case-sensitive.

Type identifiers are hyperlinked (and shown in blue) in the Viewer tab: clicking on one of them will open an Editor with the corresponding definition in the central area. In the Source tab, a similar effect is obtained by hovering the mouse cursor over a construction or schema identifier holding down the Ctrl (or the Option key on the Mac): the identifier will turn into a hyperlink. Hovering (without holding the Ctrl/Option key) on a Construction or Schema identifier will just show a tooltip containing the definition of the identifier in the synthesized form with all the inherited elements. For instance, if you go to the NP Editor and click on the

Source tab at the bottom, you'll see the NP construction definition. Try hover on the Nominal Feature (the constructional constraint on NP), and a tooltip will show you the details of the Nominal Feature schema.

Exploring and modifying the grammar

Now try returning to the Analysis Perspective. A click on the Analysis button will restore all the Views that were defined there when you switched to the Browsing Perspective. Let's take a look at the Grammar Explorer View on the right. This shows all the files that are part of the grammar, the ones that are pointed to by the preferences file. In the Grammar Explorer View, double-clicking any of these files will open an Editor in the central part of the workbench. But this time the Editor won't have any tabs at the bottom.

The file Editor features syntax highlighting and supports cut and paste, search and replace, both regular and incremental, and undo and redo of operations. The key combinations needed should be the ones you're familiar with on your platform—see the Edit menu, or press Ctrl+Shift+L, or Option+Shift+L on the Mac, to see a comprehensive list of all the key bindings. As already mentioned, holding down the Ctrl/Option key and hovering the mouse pointer over type identifiers opens their definition, and just hovering over them opens a tooltip showing their complete definition (i.e., the one including all the inherited features).

Modifying the grammar, actually

Let's try to add a lexical item, for instance HOUSE, in the Analysis perspective. In the starter grammar, lexical items are defined in lex. grm. Using Grammar Explorer in the right pane, open the starter folder (by clicking on the triangle or plus at its left), scroll down to the lex. grm node and double-click on it. This should be what you see:

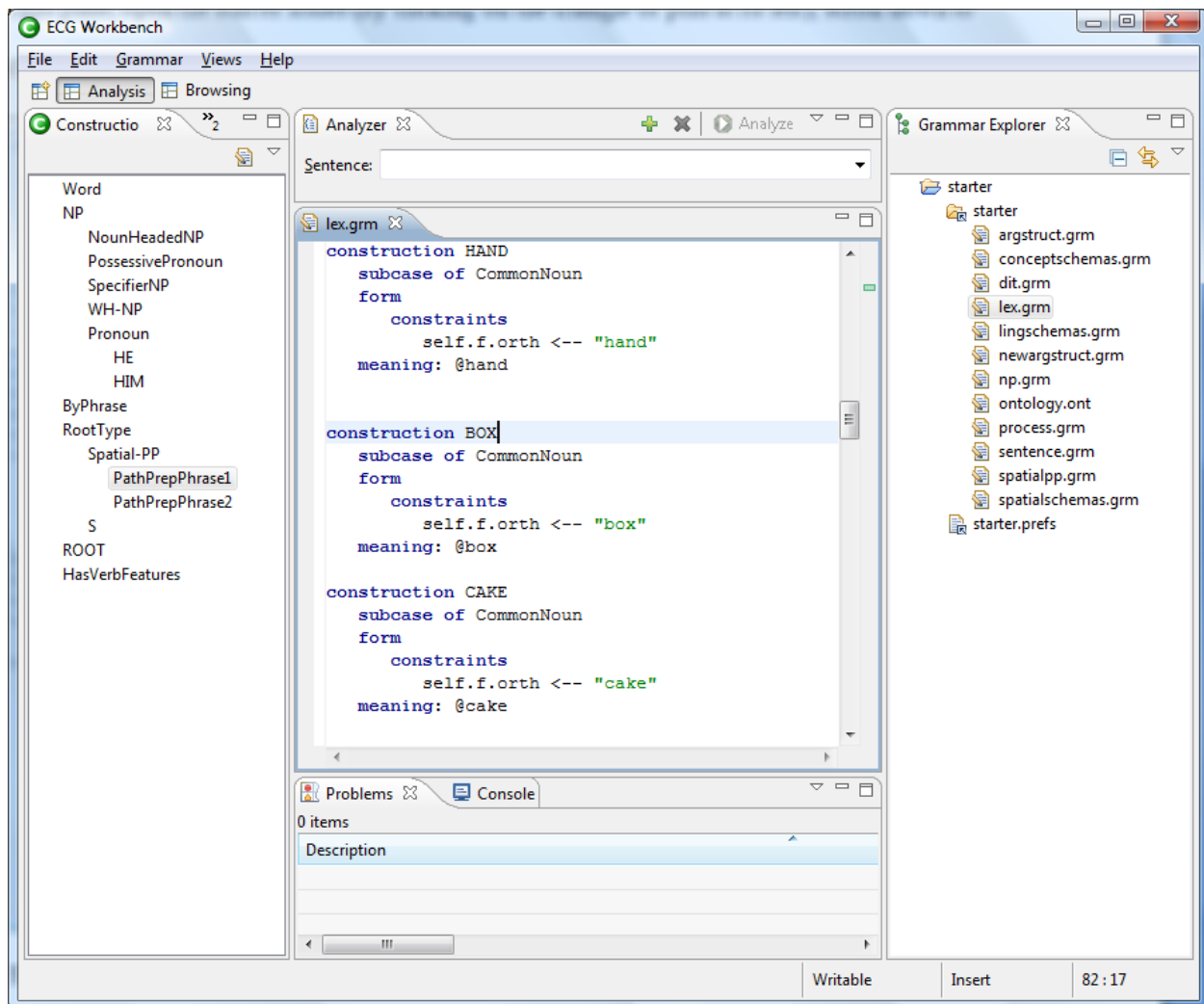



Figure 6

The easiest way to define a new lexical item is to copy and modify an old one. Modifying box seems the right thing to do. You can scroll through the code file in the middle pane to find the code for BOX. Alternatively, you can use the Edit | Find/Replace menu to search the file. Select the construction for BOX, copy it (by selecting with Edit | Copy or by pressing Ctrl+C, or Option+C on the Mac), hit the down arrow to cancel the selection and press Ctrl+V (or Option-V on the Mac). Now you have two BOX constructions.


Now we will show that the grammar checker rejects what we have done: we can't have two constructions with the same name in the same grammar. To test that, you need to save the

file you've just modified if you haven't done so yet (File | Save from the menu or Ctrl+S on the keyboard), and then check the grammar by selecting Grammar | Check (or hitting Ctrl+Shift+C). This will lead to EW signaling a bunch of errors in the Problems tab at the bottom. On certain platforms you might have to select Group By | None from the Problems view menu to see them all (click on the small triangle pointing downward in the frame containing the Problems tab). Unfortunately, the most important error is the last listed, but you can hover with your mouse over the  icon next to the position in which the faulty construction (underlined by the red squiggle) is: "There are at least two definitions of the construction: BOX." Take a look at the Grammar Explorer: you'll see various red icons signaling various (possibly spurious) errors that the grammar checker generated because it was unable to finish checking the `lex.grm` file. If you go ahead and change the second BOX into HOUSE, and also the assignment to the `self.forth` role from "box" to "house", and save and check the grammar again: everything should check fine now.

You'll probably have noticed that there's still an inconsistency in the HOUSE construction: the referent is still an object of type BOX (in `@BOX`, the "@" sign signals that BOX is actually an element in the ontology). Try to change `@BOX` into `@HOUSE` in the new construction, and then save and check the grammar. At this point the following error should be showing next to your new HOUSE construction: "Construction HOUSE does not have a consistent inherited type for its MEANING pole out of the inherited types: [entity, house]." This rather cryptic message means that there's something wrong in the meaning pole of your new construction and you have to add the appropriate element to the ontology. To do this, go to the Grammar Explorer, find the node for `ontology.ont`, and double click on it. Go to the Editor, select, copy and

paste a new `BOX` type, replace `BOX` with `HOUSE` in the copied code, and save and check the grammar; everything should be fine again.

As a last check, let's try to use the new word in a small sentence. Go to the Analyzer tab in the upper part of the window. In the edit field labeled "Sentence" type "he slapped at the house" (again, no quotes), hit enter, and then Analyze. After a few seconds a new Analysis Editor should appear; click on the SemSpec tab at its bottom, and maximize the newly created Editor window (as described above, by double-clicking on its caption, by hitting

Ctrl/Option+M, or clicking on the  icon). Here is the result:

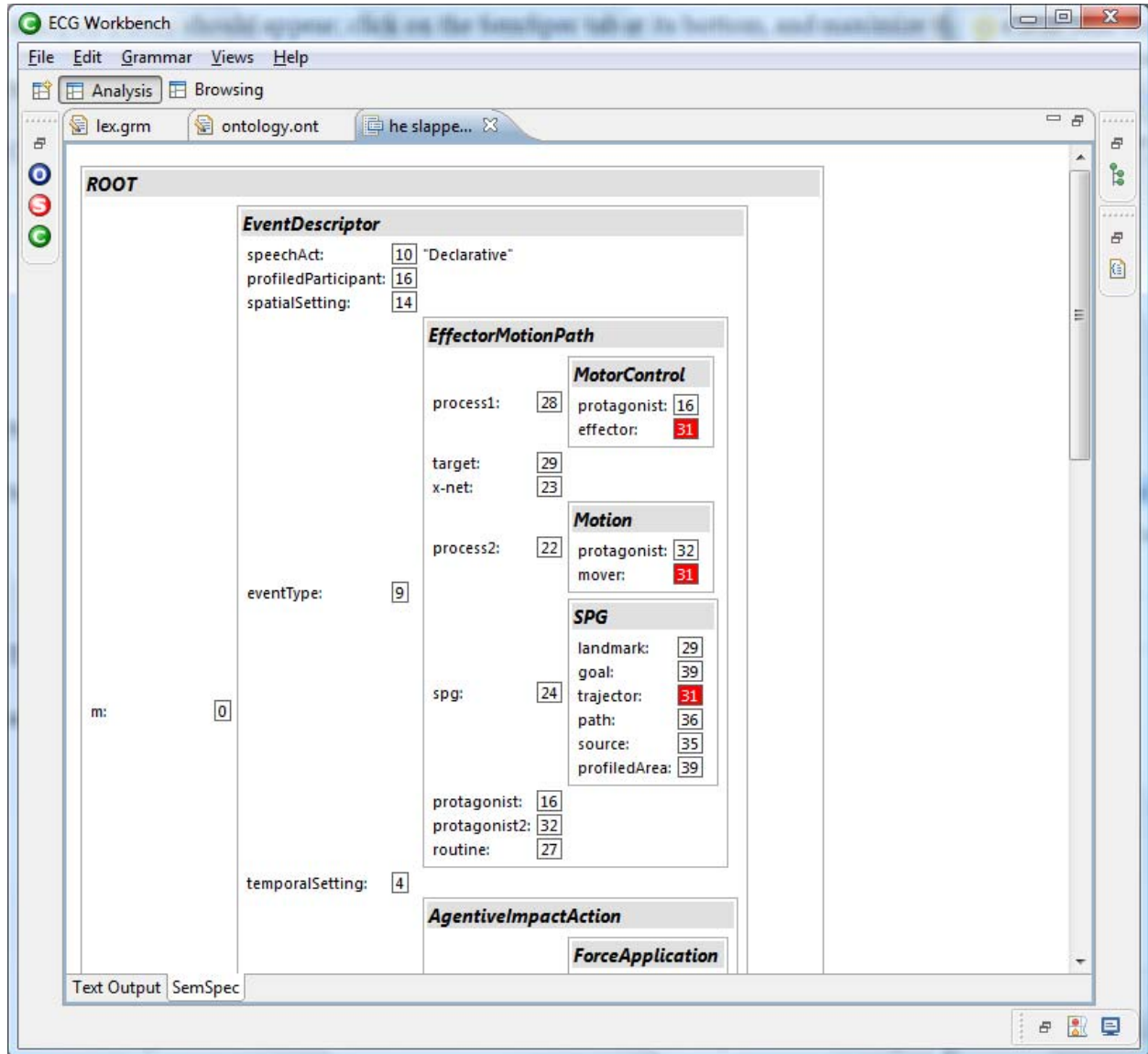


Figure 7

This last example terminates this small tutorial. If you have problems or comments, please email me, the author (see at the beginning of this document).

Advanced features

This section gathers a few tricks for power users. One that's quite useful is to press F2 to turn a tooltip into a floating window (or simply moving the mouse cursor into it), and copy the text from there.

Key combinations

There are lots of key combinations that are useful. Some are contained in the menus, but most are not. A few are (replace Ctrl with the Option key on the Mac)

- To duplicate text (like you did with BOX above) you don't need to copy and paste: you can just select the text to duplicate and type Ctrl+Alt+Down Arrow.
- To delete a line, just use Ctrl+D.
- To insert a line above the cursor, press Shift+Ctrl+Enter.
- To insert a line above the cursor, press Shift+Enter.
- To see most of the defined key combinations, hit Ctrl+Shift+L: an overlay will appear in the right corner of the workbench. Most, but not all, the key combinations you'll see are active.

Additional Views

A very useful add-on to the file editor is the Outline View, which will show all the constructs defined by the file in the active Editor. To open the Outline View from the Analysis perspective, select Views | Outline. From the Browsing perspective, select Views | Open View | Other... and choose Outline from the General folder.

The Outline View synchronizes itself automatically with the file Editor's contents, without the need to save the file, and provides a list of all the items currently in the foreground

Editor (the one with the blue tab). You can drag the Outline tab (which now will have covered the Grammar Explorer) down with your mouse until you see the grey rectangle—which represents the future position of the tab if you release the mouse button—moving to the bottom half of the right column. Release the button there. Now you should have two panes on the right with Grammar in the upper part, and the Outline View in the lower part, like this:

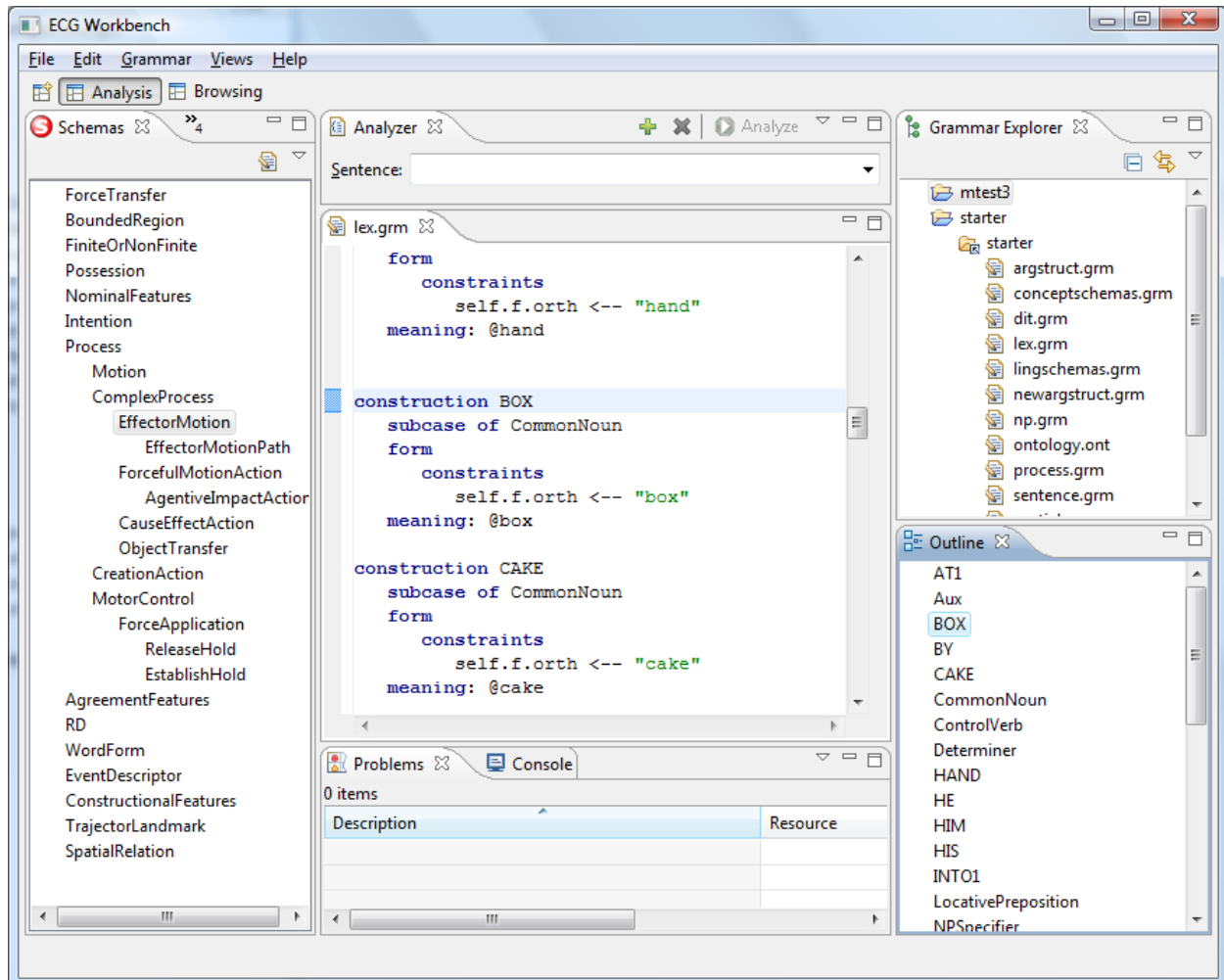


Figure 8

This way, you can see the Grammar Explorer and the Outline at the same time. Of course you can rearrange all the tabs in the upper part of the workbench window as you please.

Cloning a grammar (to make a new one)

At this time, there is no way to create automatically a preferences file. Therefore, you have to look at those that come with the example grammars, and copy and modify one of them. To create a new grammar, right-click inside the Grammar Explorer view, select New | Project..., select Project in the wizard dialog that will appear, hit Next, insert a name in the Project name edit field, and finally hit Finish. To create a .prefs file, right-click on the newly created Project, select New | File, type a file name (don't forget the .prefs extension) in the appropriate field, hit Finish. Double-click it to edit it. The relevant entries are:

- GRAMMAR_EXTENSIONS: a space-separated list of extensions.
 - Example:

```
GRAMMAR_EXTENSIONS = grm sch
```
- ONTOLOGY_EXTENSIONS: same as above for ontology-defining files;
- GRAMMAR_PATHS: a newline-separated list of directories in which the files with the extensions defined above will be looked for. The last line must be a semicolon.
 - Example:

```
GRAMMAR_PATHS ::=  
    ./starter  
    ;
```
- ONTOLOGY_PATHS: a newline-separated list of files (not directories) if ONTOLOGY_EXTENSIONS is not defined that will be looked up as ontology files, or a list of directories (as in GRAMMAR_EXTENSIONS) otherwise. Same format as GRAMMAR_PATHS.
 - Example:

```
ONTOLOGY_PATHS ::=  
    ./starter/ontology.ont  
;
```

References

- ECGweb Wiki: <http://ecgweb.pbwiki.com/>