

A Brief Introduction to ECG Workbench

(V0.7, draft)

Author: Luca Gilardi lucag@icsi.berkeley.edu

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

Introduction

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 references at the end of this document for more information).

ECG Workbench is a support for the grammar writer. With ECG Workbench you can:

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

How to install ECG Workbench

At this time, no automated installation procedure is available (but one will be eventually). 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, your desktop or your user folder. The executable for your platform requires that a Java Virtual Machine, version 1.5 or better, be already installed on your system.

Microsoft Windows Intel 64 bit , any recent version supporting Java 1.5,

Apple Mac OS X 64 bit Intel and PPC (any recent version should work);

Linux Intel 64 bit on GTK (any distribution supporting Java 1.5+)

The archive file (a zip file) contains a directory named **ecg-workbench-<version number>.zip**.

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: keeping different schemas and constructions in different files helps in keeping the grammar organized and thus easily comprehensible. Some files may define lexical items for instance, other files may describe schemas or particular kinds of schemas, other files may describe constructions, and so on. The workbench doesn't assume any particular criteria for breaking up the grammar. The 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 given in the Advanced Features section.

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 (<http://www.icsi.berkeley.edu/~lucag/>) and the zip file containing EW itself, select **starter**.
.prefs, and click OK. You should see something similar to the following:

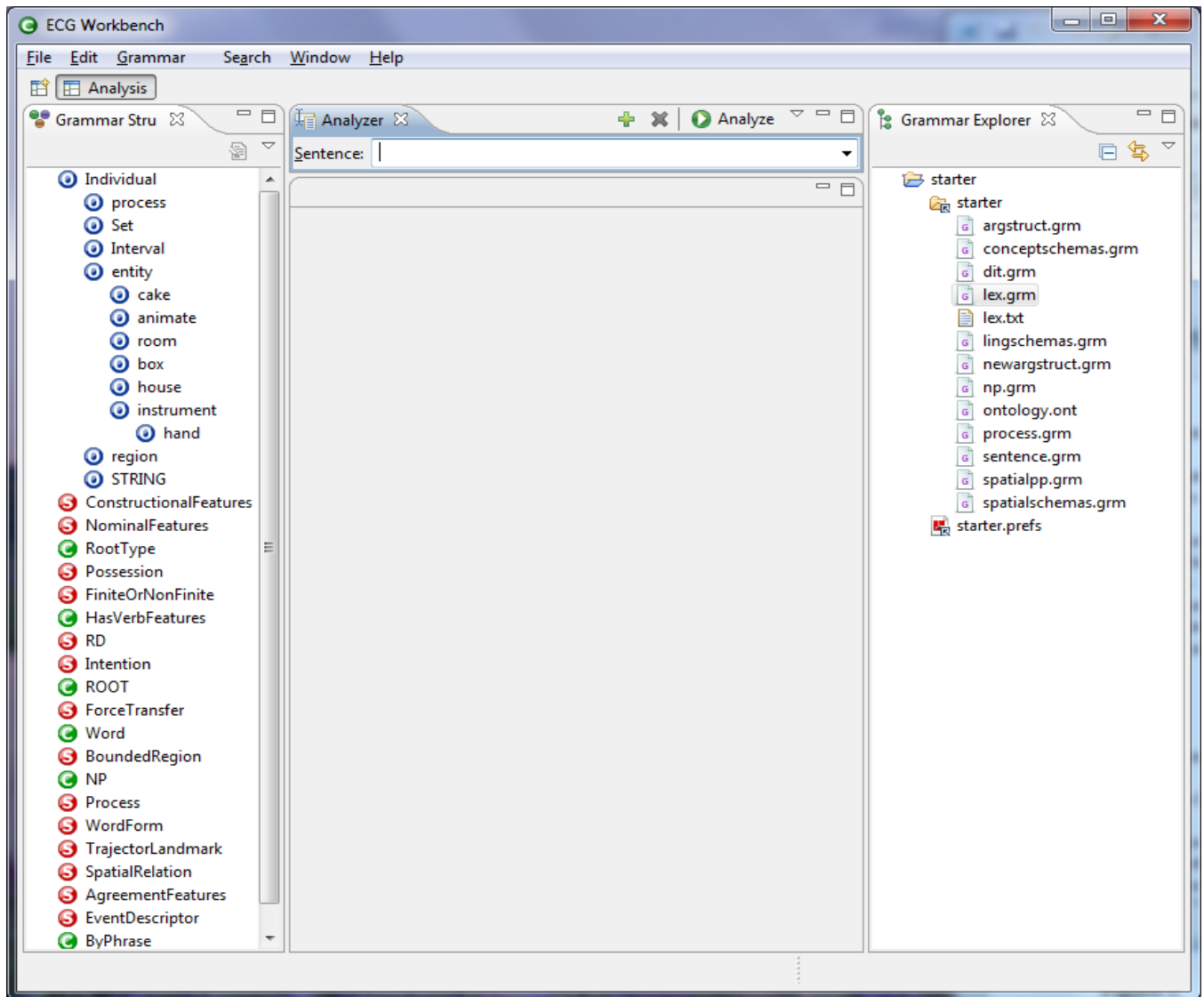




Figure 1

Figure 1 shows what the screen looks like before any input has been analyzed. The left hand column depicts the schemas, constructions, and ontology items in the “starter” grammar. Part of

the ontology lattice has been expanded. The right hand column shows the file structure that was used in specifying this grammar.

Perspectives

The Workbench is organized into Perspectives, we will use only Analysis (fig. 1) and Browsing. I'll be using capitalized words to signal that they have a special meaning in the context of the workbench. Perspectives are what the word suggests: different way of looking at content—an ECG grammar. The Analysis Perspective is the default perspective and enables all the functionalities of the Workbench. The Browsing Perspective, accessible by clicking the button with the  icon, shows just the Grammar Structure and Content Views with a fixed layout, that is, you cannot close it or minimize them. The Browsing Perspective is simpler than the Analysis. If you don't plan to make changes to the grammar or using it, the Browsing Perspective is definitely your best option. Otherwise stay in the Analysis Perspective.

To change to the Browsing Perspective, click on the  icon at the far left just below the menu bar, choose **Other...**. A dialog will pop up; choose **Browsing**. From this Perspective you can still analyze sentences by opening the Analyzer View by selecting **Views | Open View | Other...**, opening the ECG folder, selecting Analyzer, and finally clicking OK.

Analyzing a sentence.

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. Using the Analyzer View you can input sentences into the ECG analyzer and obtain a semantic specification (or more than one in some cases). You can enter sentences in the Sentence edit box; near the top left and tagged by the word Sentence. If the preferences file specifies examples sentences—which is the case if you are using

starter.prefs—you can choose one from the drop-down list. Click on the down arrow at the far right of the Sentence edit box, select for instance **the box slid** and click the Analyze button just above the drop-down list. The resulting analysis will appear in a new Editor (the tabbed windows appearing in the central part of the workbench window are called Editors). The new Editor's title will remind you that its content is relative to the sentence you just analyzed:

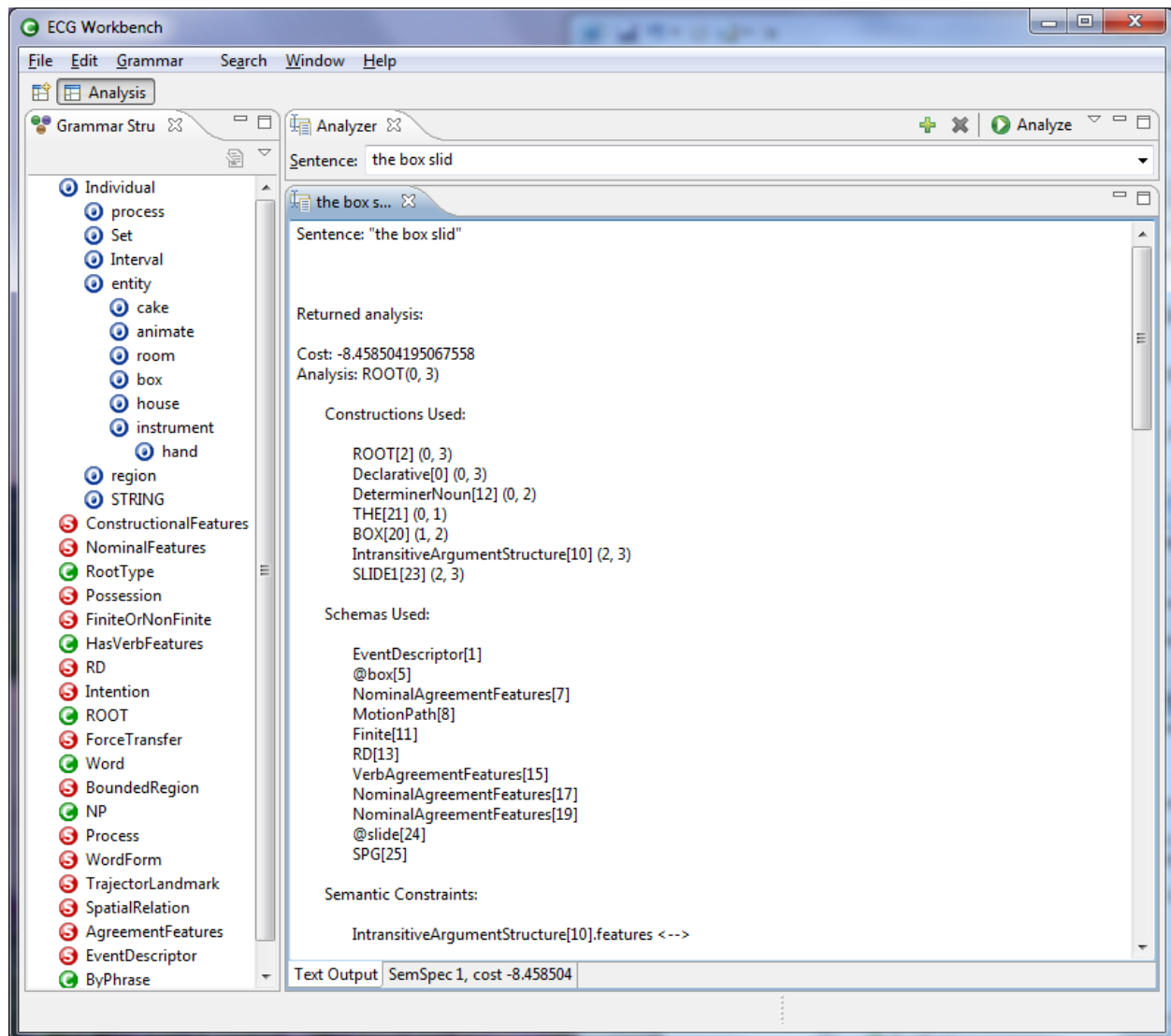


Figure 2

More on sentence analysis

As you can see (fig. 2), 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[2] (0, 3)
Declarative[0] (0, 3)
DeterminerNoun[12] (0, 2)
THE[21] (0, 1)
BOX[20] (1, 2)
IntransitiveArgumentStructure[10] (2, 3)
SLIDE1[23] (2, 3)
```

The numbers in parentheses are the “spans” of the sentence that the relative construction recognized, starting from zero: **0 the 1 box 2 slid 3**. Thus the **Declarative** construction was used to recognize the entire sentence (it covers the words from 0 to 3), whereas **IntransitiveArgument Structure** spans words 2 to 3, and so on for the other Constructions. From this output, drawing the parse tree is straightforward.

All the numbers in square brackets are the instances (or “features”) that show up in the other view, the **SemSpec**, described in the next paragraph. Still in this view, the Constructions Used Section, is followed by another section showing the Schemas involved in the analysis, and a final section containing all the unification bindings, not all of which fit in Figure 2.

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 **the box slid** contains the semantic specification (Figure 3), 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 only a part of it (again, depending on your screen's resolution). You may need to make room for the Analysis Editor by clicking the minimize buttons (see figure to the left) at the upper left corner of the Grammar Explorer View. This is what you'll see:

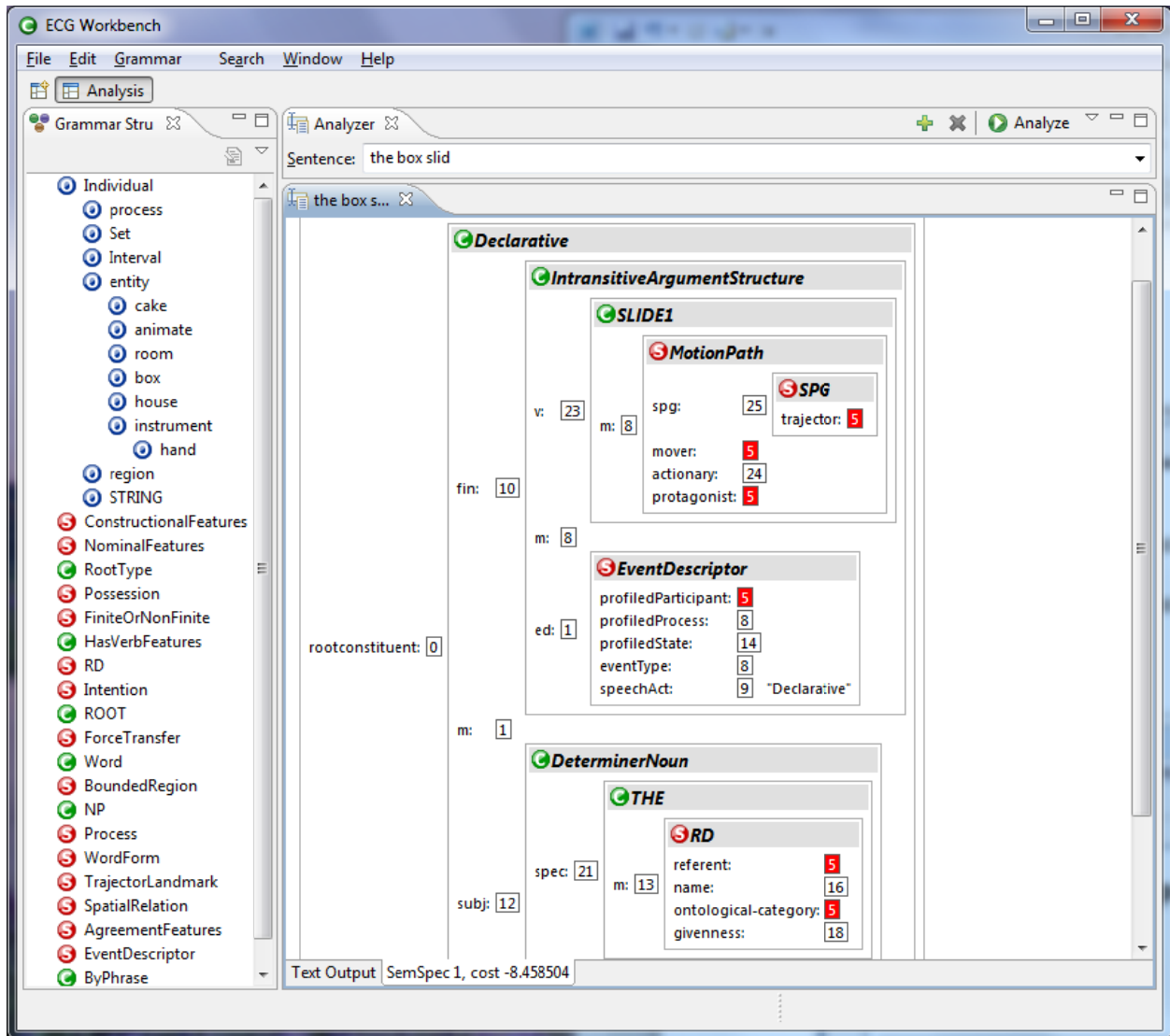



Figure 3

To make even more room, you can maximize the Editor window. You can click on the maximize icon (☐) in the same frame that contains the analysis results Editor. Here is what you should see:

As you can see, some of the coindexed elements, which also appear in square brackets in the **Text Output** tab, are highlighted in red. When you click on a boxed number, and all the ones that have the same value will light up as well. For instance, if you click on the **5** next to **protagonist** in the **Motion** schema, all the other coindexed roles will be colored in red (you might have to scroll down to see them all). If you hover over a boxed number, the value associated with that index will be temporarily displayed. For example if you hover over the **24**, which is the actionary role in the MotionPath schema, the a box with word “slide” will appear. The SemSpec is the main link between language analysis and subsequent action, as described in Reference 2.




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 a collapsed grey bar, the contained element will expand again. You can also print, by just selecting File | Print from the main menu. To restore the Analysis result window to its original size, you double-click on the title, right-click and select Restore, or click 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**. Press enter (this will make the workbench

remember this sentence in the list), and then click Analyze. A new Editor pane will show up with the results of the analysis, while the old example will remain available as another tab at the top.

Editor management – can be skipped initially


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 Grammar Structure View) rather than selecting from a menu. Unlike Views, Editors typically (but not always) permit the modification of the grammar files; they 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 Grammar Structure View, can be open at any given time. In the Grammar Structure View, shown on the left in Figures 1-3, Constructions are marked by the  icon, Schemas are marked by , and  identifies ontology elements.


If you analyzed the two sentences as described above, you'll see tabs for 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 of 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 forward and back buttons to navigate back and forth to the "places" you've visited. There are no such buttons, in the current version of the Workbench, 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 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 Grammar Structure View on the left shows all the grammar elements that are available—from all the files making up the grammar—in a hierarchical view. Clicking on an arrow, or plus, depending on our platform, 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 the construction **RootType** (marked by ) , you'll see that, among others, it contains a node called **NP**, which in turn contains **WH-NP**, **SpecifierPlusKernel**, **Pronoun**, **UndeterminedNP**, and **ProperNounNP**. This is because these are all subcases of **NP**.

If you return to the Grammar Structure View and look for the Construction **S**, click on it, and navigate to the **Declarative** item, click on its  and you'll see that its actual definition will be shown to the right, where a new tabbed window—a new Editor showing **sentence.grm** in its

caption—will show up:

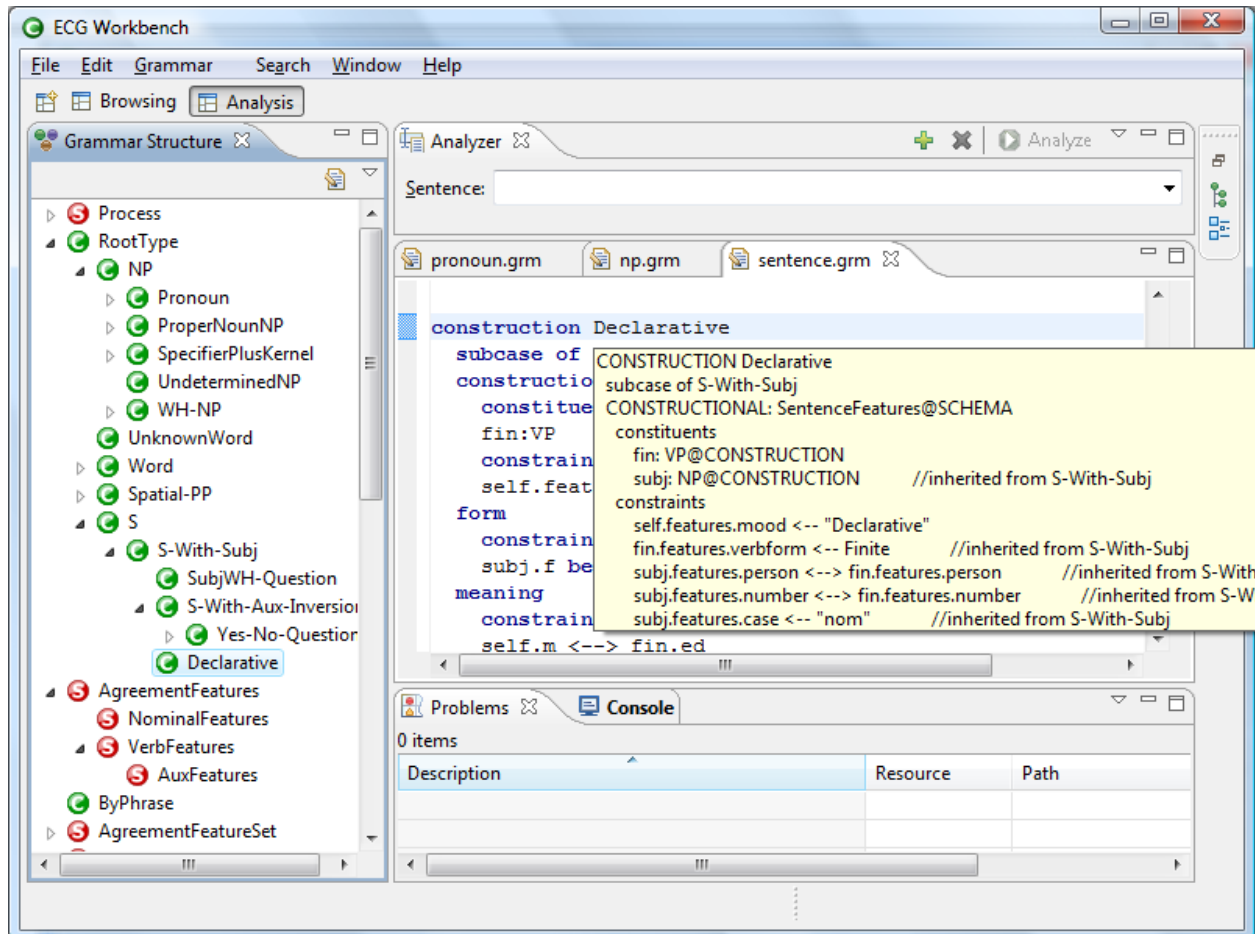


Figure 4

Let's look at the **sentence.grm** Editor tabbed window. There are a few aspects to be especially aware of.

First of all, you can hover on many elements. In fig. 5 above is shown the hover result for Declarative. The information shown in the hover info box (you can make it a real window that you can stretch and copy from by hitting F2) may be different from the file content. This is because it shows the complete structure of an element, which includes local and inherited structure. For instance, the hint box for Declarative contains the following:

```

CONSTRUCTION Declarative
subcase of S-With-Subj
CONSTRUCTIONAL: UNTYPED
constituents
  fin: VerbPlusArguments@CONSTRUCTION
  subj: NP@CONSTRUCTION //inherited from S-With-Subj
constraints
  fin.features.verbform <-- Finite //inherited from S-With-Subj
  subj.features.person <--> fin.features.person //inherited from S-With-Subj
  subj.features.number <--> fin.features.number //inherited from S-With-Subj
  subj.features.case <-- "nom" //inherited from S-With-Subj
FORM: UNTYPED
constraints
  subj.f before fin.f
MEANING: EventDescriptor@SCHEMA
constraints
  self.m <--> fin.ed
  self.m.speechAct <-- "Declarative"
  subj.m.referent <--> self.m.profiledParticipant //inherited from S-With-Subj

```

But the text file content is a shorter, since it omits the structure defined by its supertypes:

```

construction Declarative
subcase of S-With-Subj
constructional
constituents
  fin:VerbPlusArguments
form
constraints
  subj.f before fin.f
meaning
constraints
  self.m <--> fin.ed
  self.m.speechAct <-- "Declarative"

```

The hint box thus shows all the inherited elements from the constructions that **Declarative** is a subcase of. The comments (generated automatically) tell you which elements are inherited and from which supertype. More specifically, it shows that **Declarative** inherits a **subj** role from **S-With-Subj**, and also four different constructional constraints, and a meaning constraint.

Another difference between the ECG code shown by the file Editor and the hint box is in the font case: in the first example, keywords (like **construction**, **subcase**, and so on) are all uppercase, while in the second they are lowercase. This is because some software packages that are part of EW take quite literally the ECG specification, which requires all the keywords be case-insensitive. Therefore, you are free to use upper- or lowercase (and even mixed-case) letters for the various ECG keywords. Notice though that all type identifiers (**Declarative**, **S-With-Subj**, **Pronoun**, **NP**, **Word**, etc.) are instead case-sensitive.

Second, another useful feature of the Editor is that the type identifiers are hyperlinked. Hovering the mouse cursor over a Construction or Schema identifier, this time holding down the Ctrl (or the Option key on the Mac), will turn the identifier itself into a hyperlink. You can jump to the type definition by just clicking on it.

Third, search and replace (local to the file shown in the title) can be done by choosing **Edit | Search** (or by hitting Ctrl/Option+F).

Fourth, a global search and replace is possible from the top Search menu or by hitting Ctrl/Option+H. This will open the following dialog:

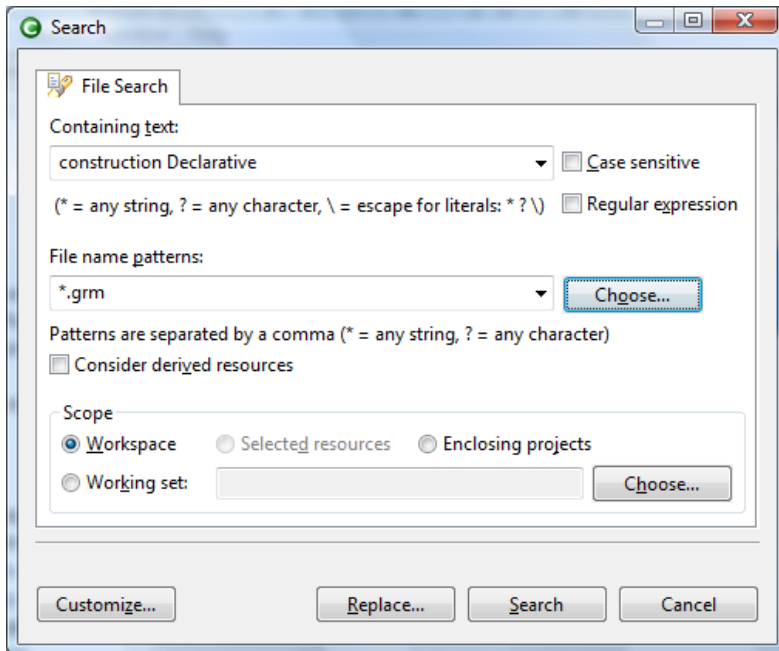


Figure 5

Ignoring the **Scope** group at the bottom, many of the other options are self-explanatory. You can choose the file types to search by pressing **Choose...** and ticking off all the files that are recognized by the Workbench as valid grammar files. Hitting **Replace** will open a dialog that will prompt you to enter the substitution string, and an optional preview of the result, before the actual substitution takes place. By hitting **Search**, the Workbench will find all occurrences of in all open grammars (only one if you just opened [starter](#)). A new tabbed Search View will be opened at the bottom, as follows:

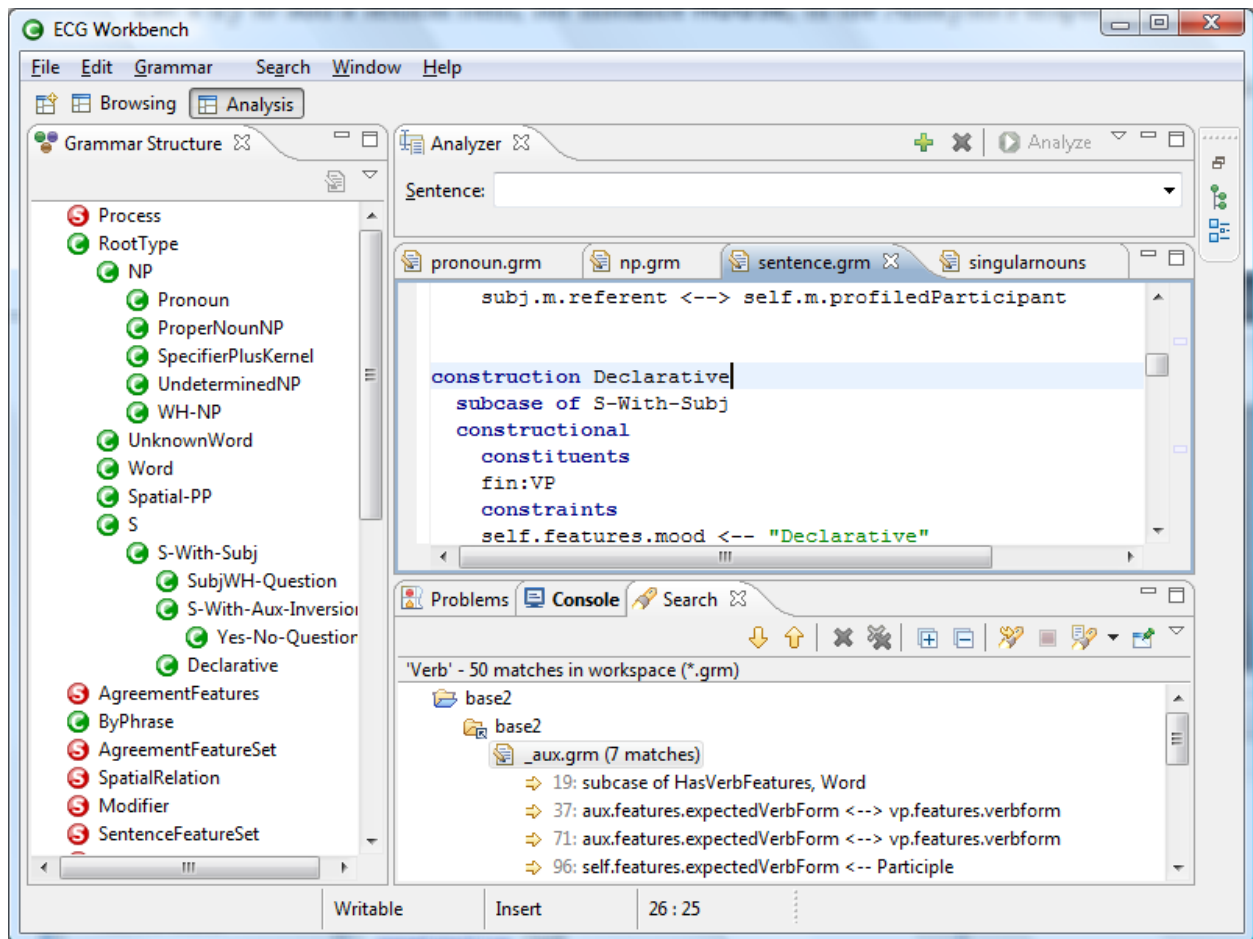


Figure 6

You can reach all the occurrences found by clicking on the lines indicated by the right-pointing arrows. The two up and downward arrows will walk you through all the matches. As with any other View, you can drag the Search view from the bottom to the sides, or minimize it.

Exploring and modifying the grammar

Now return to the Analysis Perspective. A click on the Analysis button will restore all the Views that were defined there if you switched to the Browsing Perspective. Let's take a look at the Grammar Explorer View on the right of Figure 1. This shows all the files that are part of the grammar, the ones that are named 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.

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 **Help** menu, 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 **Door**, in the Analysis Perspective. In the **starter** grammar, many 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:

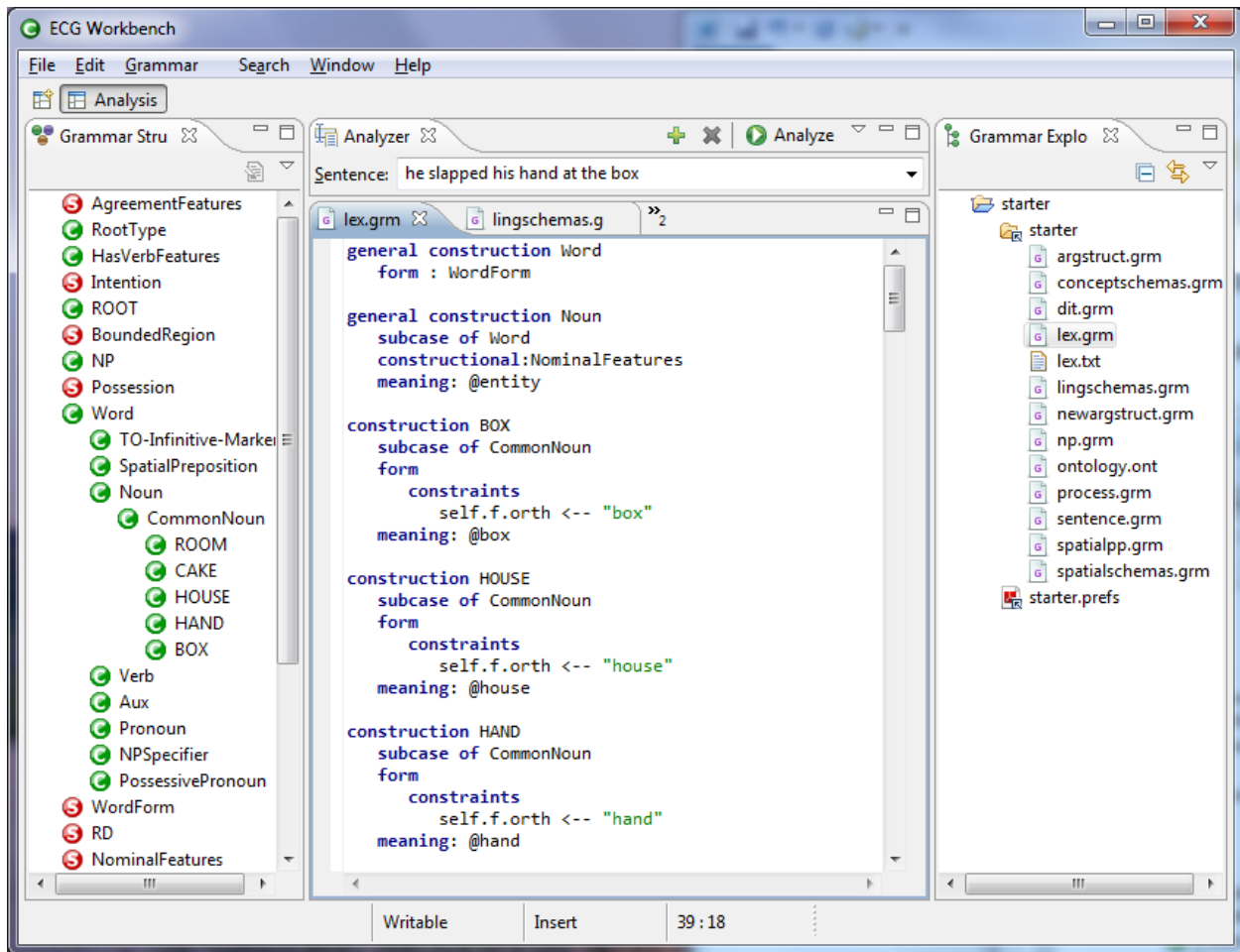



Figure 7


The left column shows the grammar structure window with the Noun node expanded to show the 4 CommonNoun items in our Starter grammar. The center pane contains an editable version of the ECG definitions of all the words in lex.grm and some grammatical constructions like Word and Noun.

The easiest way to define a new lexical item is to copy and modify an old one. Modifying the construction for 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, or, even simpler, use the Outline View (see below). Select the construction for

Box, copy it (by selecting with **Edit | Copy** or by pressing Ctrl+C, (or Option+C on the Mac) and press Ctrl+V (or Option+V on the Mac). Now you have two **Box** constructions.

Just as a small exercise, we can 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). The errors will be marked on the text in the center pane. You can hover with your mouse over the underlined element or the  icon next to the position in which the faulty construction is: *There are at least two definitions of the construction: BOX*. Take a look at the Grammar Explorer in the right pane: 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. You can go ahead and change the second **Box** into **Door**, and also the assignment to the **self.f.orth** role from "**box**" to "**door**." Save and check the grammar again: everything should grammatically check fine now.

You will probably have noticed that there's still an inconsistency in the **Door** 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 **@door** in the new construction, and then save and check the grammar. At this point the following error should be showing next to your new **Door** construction: "Construction Door does not have a consistent inherited type for its MEANING pole out of the inherited types: [entity, door]." 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 definition, replace **box** with **door** in the copied code. 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 door**, hit enter, and then Analyze. 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:

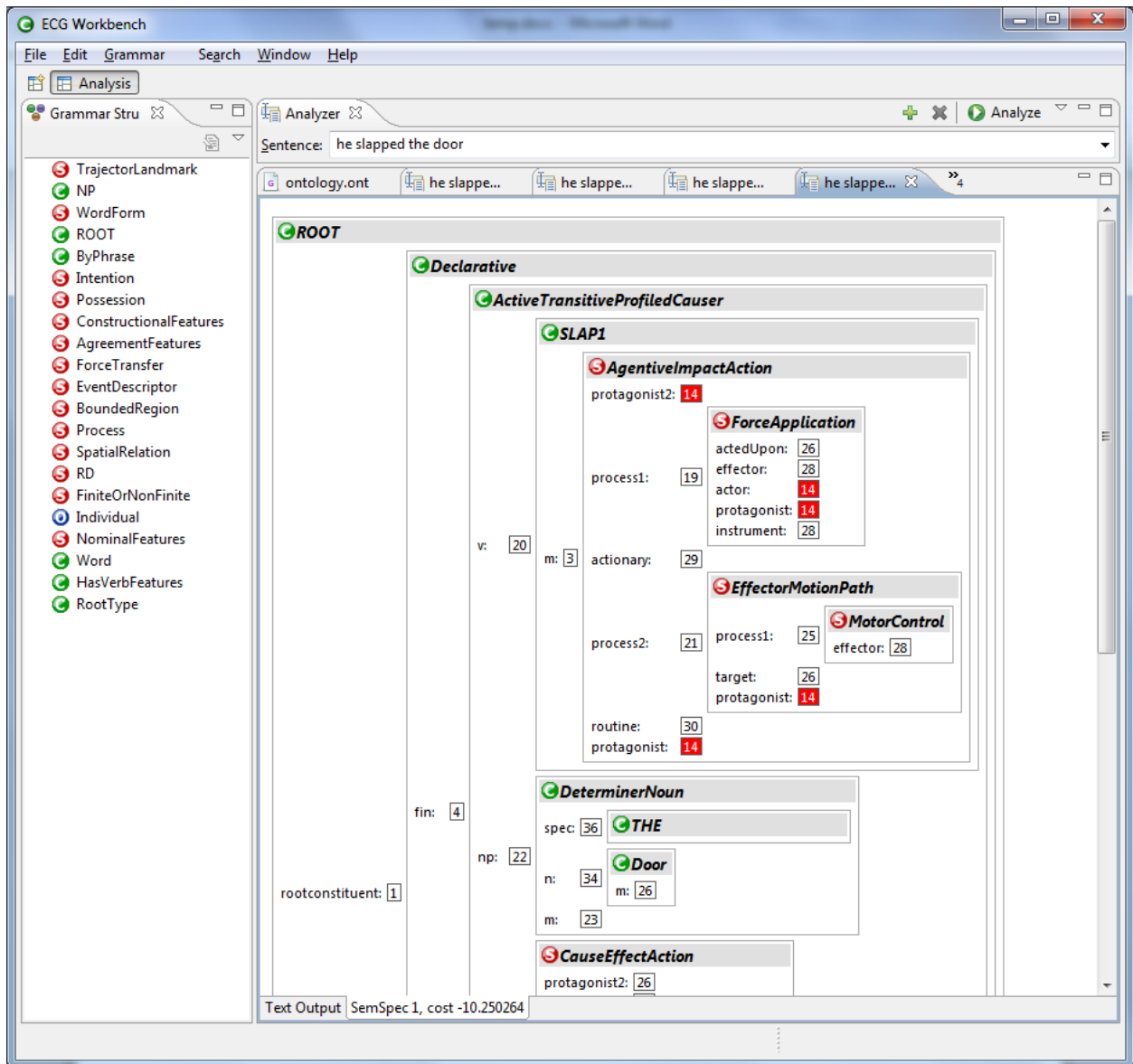


Figure 8

As before, the various roles associated with the agent “he” are all highlighted and here the object of the action “door” is also visible. 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).

Some advanced features

This section gathers together 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.

Preferences

Preferences are not to be confounded with the preferences file, which, as we've seen above, describes a grammar. The dialog window accessible through the menu **Window | Preferences** allows you to define a wealth of parameters that affect the Workbench. To the left all the preference topics are organized hierarchically. Clicking on a topic will determine the content of the right hand side. Below the General topic is shown. One interesting setting is the possibility to open files (those shown in the Grammar Explorer View) with a single click instead of a double click. This is done choosing Single click in the Open mode group below.

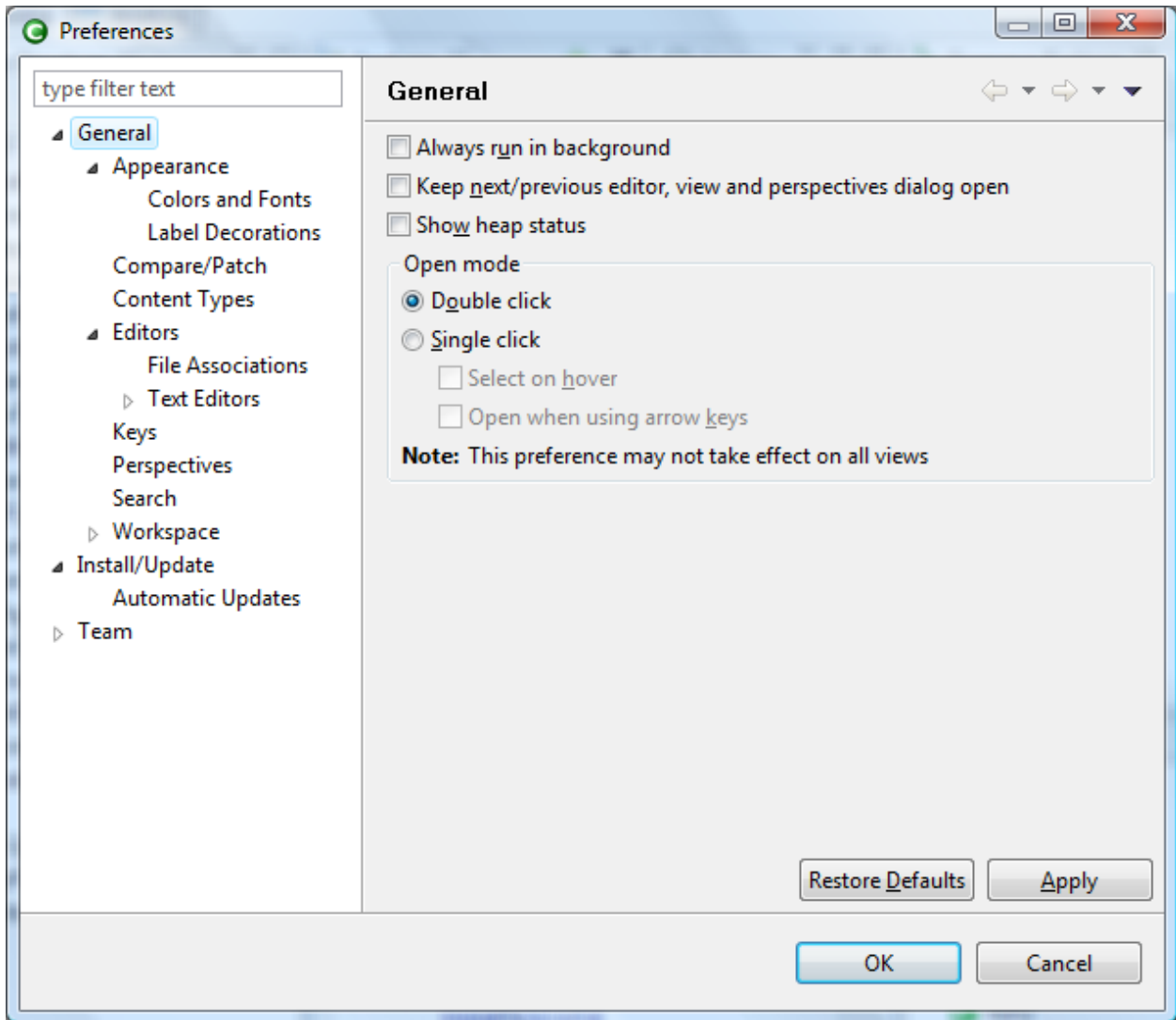


Figure 9

If you want to change the Editors' font, you can click **Colors and Fonts**. Another list will appear on the right: open the **Basic** node, select **Text Font**, and click on the **Edit** button. Here you can choose the font you like. Another very useful feature is Automatic Update. If you click on the checkbox, the Workbench will look automatically for new updates according to the schedule you choose.

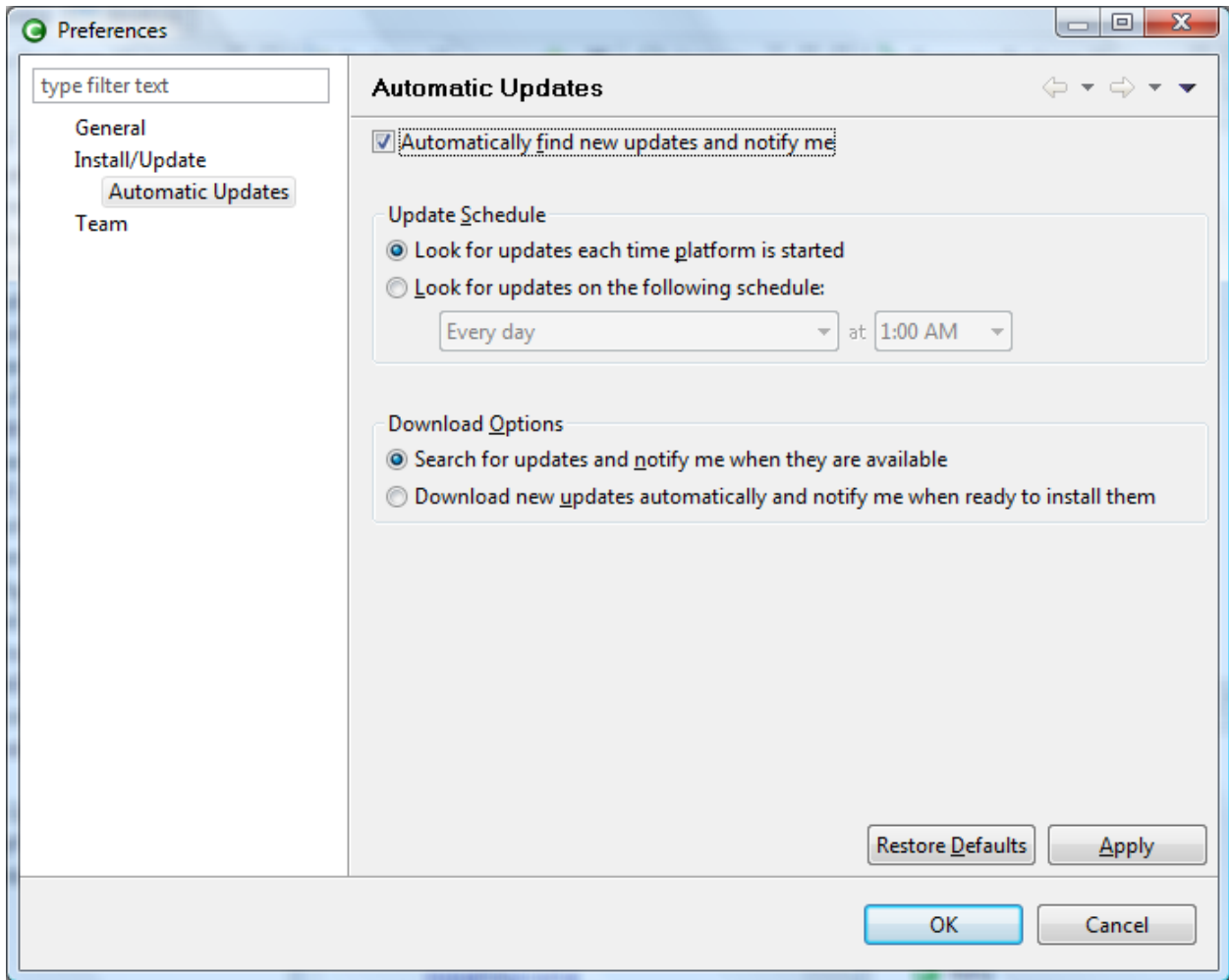


Figure 10

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 **Table** 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.

All these key combinations are accessible and modifiable from the Preferences dialog, Keys topic.

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.

This way, you can see the Grammar Explorer and the Outline at the same time. Again, you can rearrange all the tabs in the Workbench window as you please.

Cloning a grammar (to make a new one) – rarely used.

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

1. Outdated ECGweb Wiki: <http://ecgweb.pbworks.com/>
2. Latest published paper: <ftp://ftp.icsi.berkeley.edu/pub/feldman/cslp.final.docx>
3. Oxford Handbook Chapter
<ftp://ftp.icsi.berkeley.edu/pub/feldman/OxfordHandbook.pdf>
4. Ellen's ICLC talk <ftp://ftp.icsi.berkeley.edu/pub/feldman/ICLC13.ellen.ppt>