A Brief Introduction to ECG Workbench

(V0.6, draft)

Author: Luca Gilardi lucag@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 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 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. You can download the Java Virtual Machine Standard Edition version 1.6 from Sun

by clicking here. Older versions are available from http://java.sun.com/javase/downloads/. The

available platforms are:

- Microsoft Windows Intel 32 and 64 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 and 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**

(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 schemas and

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 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 **starter2** grammar (contained in the file **starter2.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 **starter2.prefs**, and click OK. Depending on your platform, you should see something similar to the following:
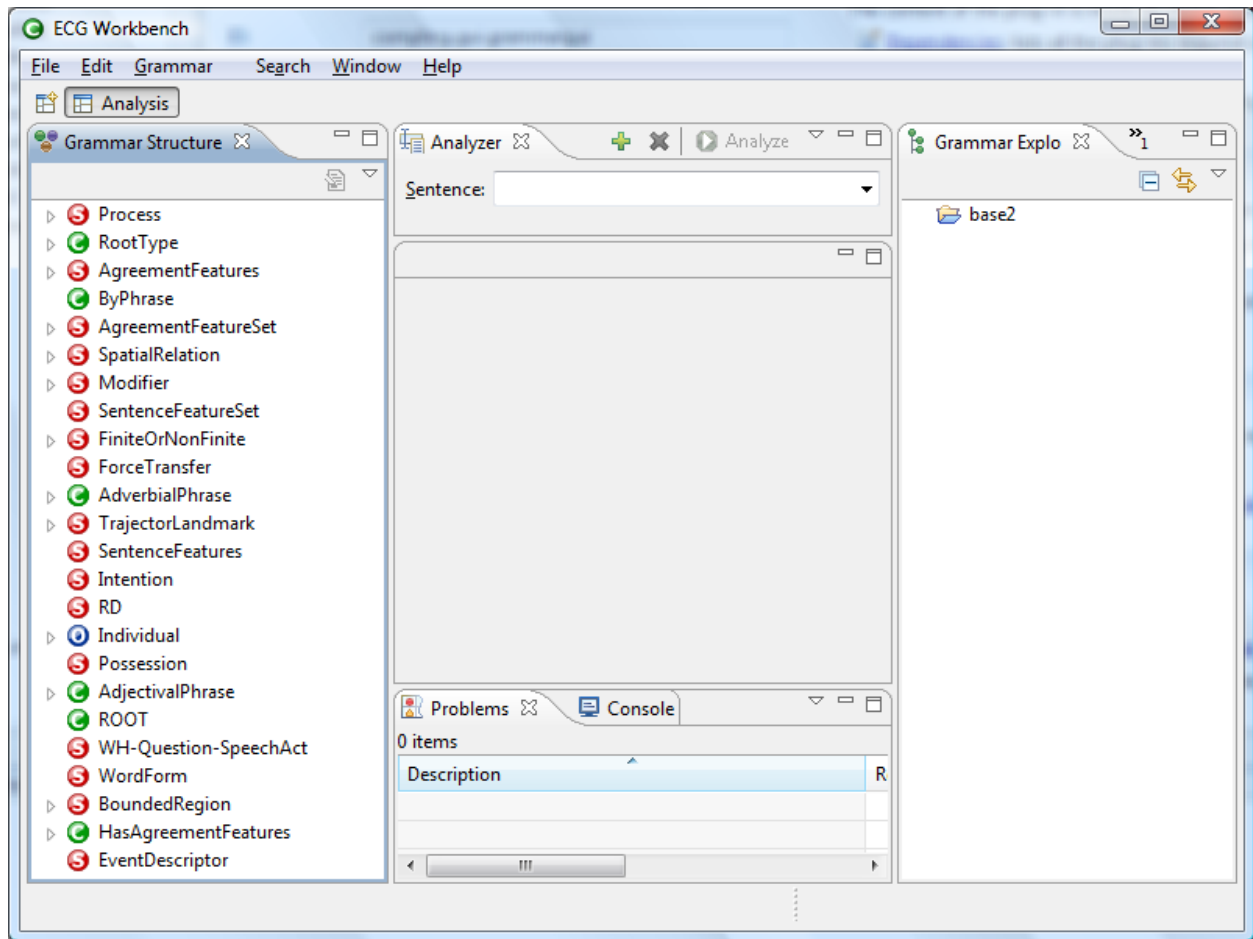
Figure 1

Perspectives

The Workbench is organized into Perspectives, 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

things, or content—an ECG grammar, more accurately. 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

just want to avoid making inadvertent changes to it, the Browsing Perspective is definitely your best option. It's also a good choice if you don't have a high-resolution screen since it's far less cluttered that the other one.

Changing to the Browsing Perspective

Click on the ![icon] icon at the far left just below the menu bar, choose **Other...**. A dialog will pop up; choose **Browsing**. The result will be the following:
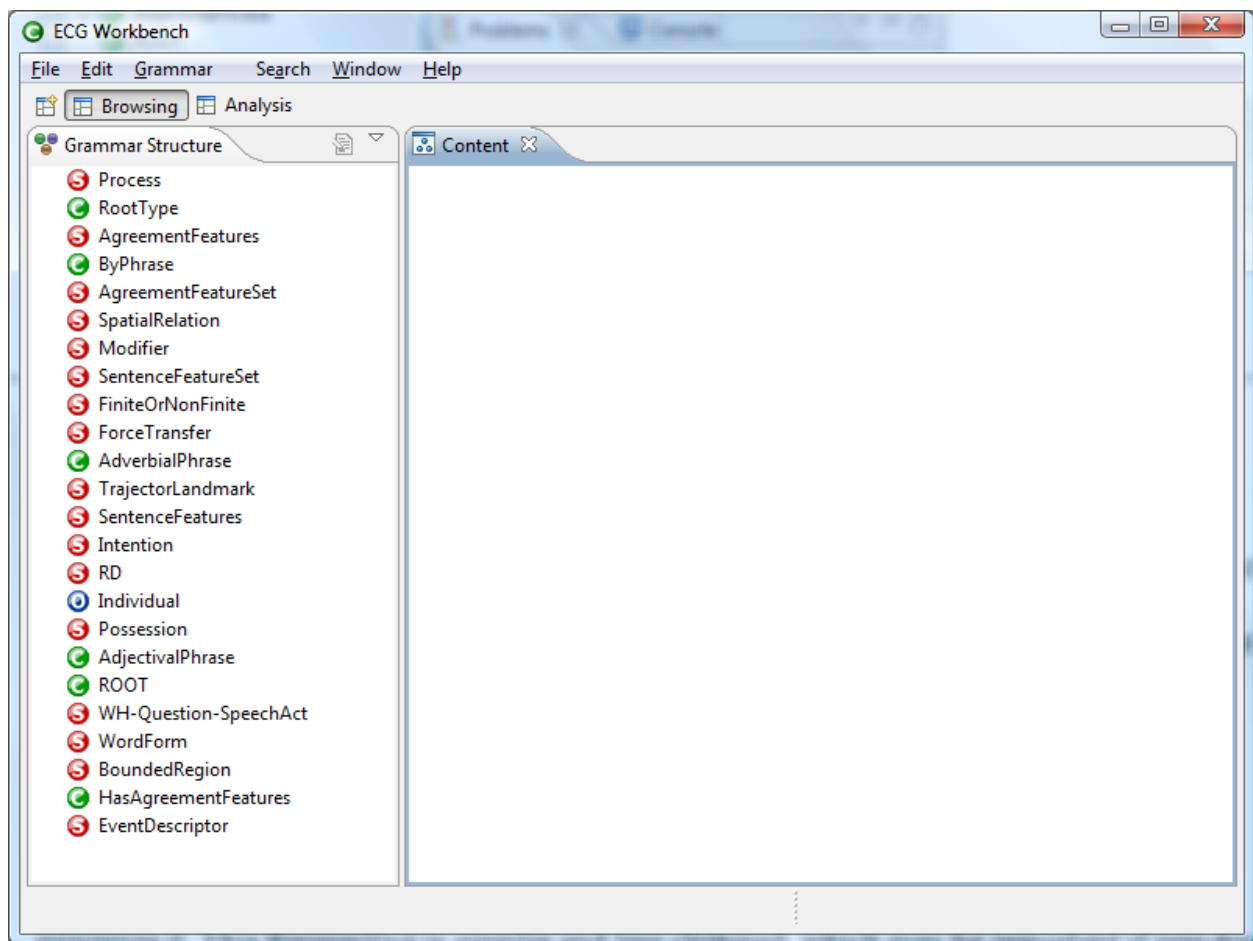

Figure 2

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

First, let's describe the Workbench in the Analysis Perspective, the default. At the bottom (refer to fig. 1), there are two 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 (see Changing to the Browsing Perspective above). 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 are using **starter2.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 **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:
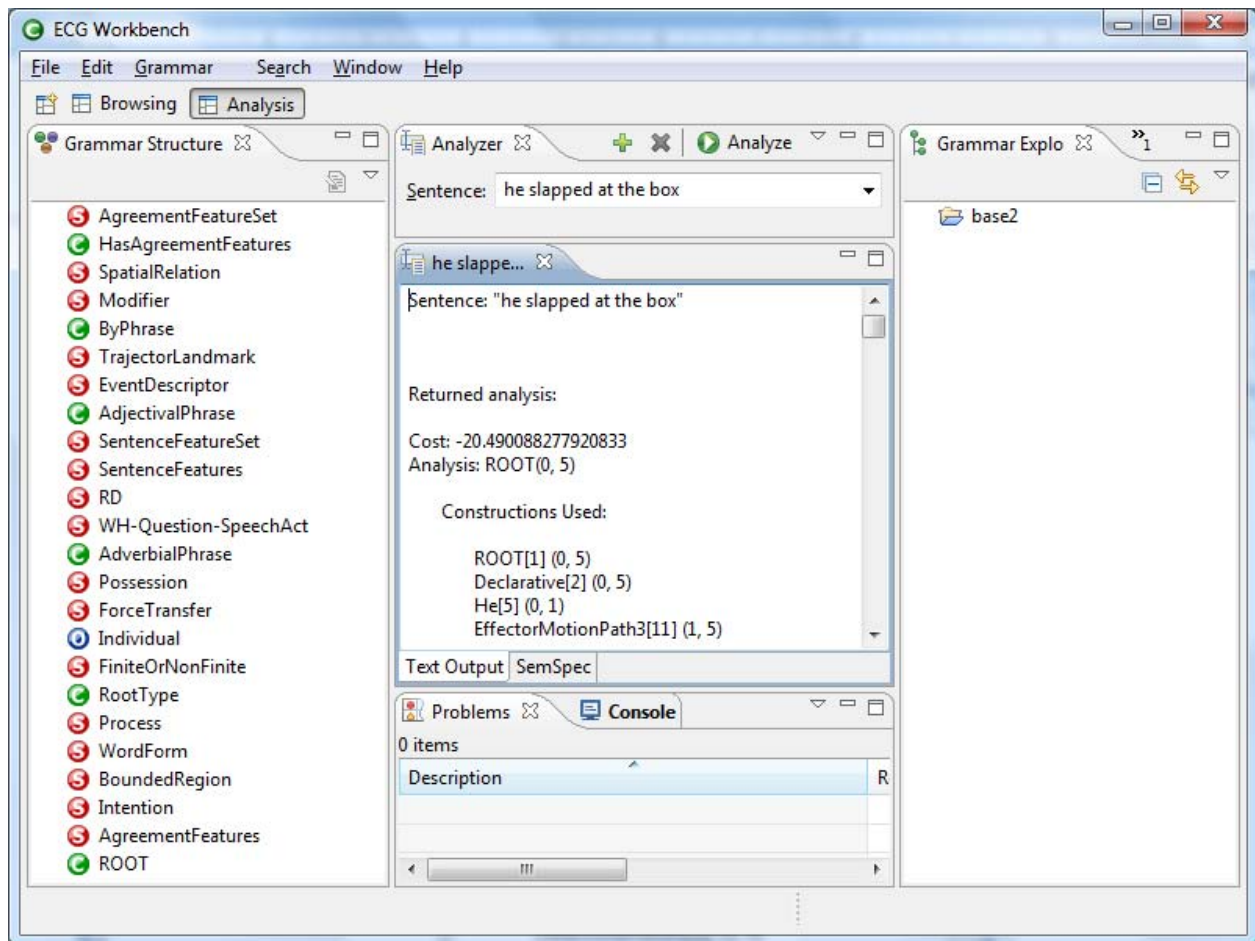
Figure 3

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 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, as in others, Constructions are marked by the icon, Schemas are marked by , and ?M? identifies ontology elements.

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 [1] (0, 5)

Declarative [2] (0, 5)

He [5] (0, 1)

EffectorMotionPath3 [11] (1, 5)

SLAP1 [23] (1, 2)

PathPrepPhrase2 [27] (2, 5)

AT1 [32] (2, 3)

DeterminerPlusKernel [37] (3, 5)

The [42] (3, 4)

SimpleKernel [44] (4, 5)

Box [47] (4, 5)

The numbers in parentheses are the "spans" of the sentence that the relative construction recognized, starting from zero: $_0$ **he** $_1$ **slapped** $_2$ **at** $_3$ **the** $_4$ **box** $_5$. Thus the **Declarative** construction was used to recognize the entire sentence (it covers the words from 0 to 5), whereas

EffectorMotionPath3 spans words 1 to 5, and so on for the other Constructions. From this

output, drawing the tree typical of the generative grammar tradition should be 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 tab, 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. 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
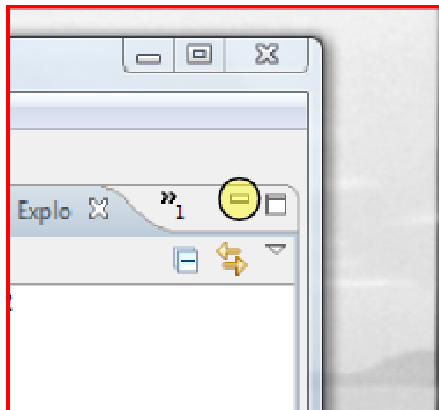
the classic attribute-valued matrices that's common in the

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). You'll probably have 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 and the Console and Problems view at the bottom. This is
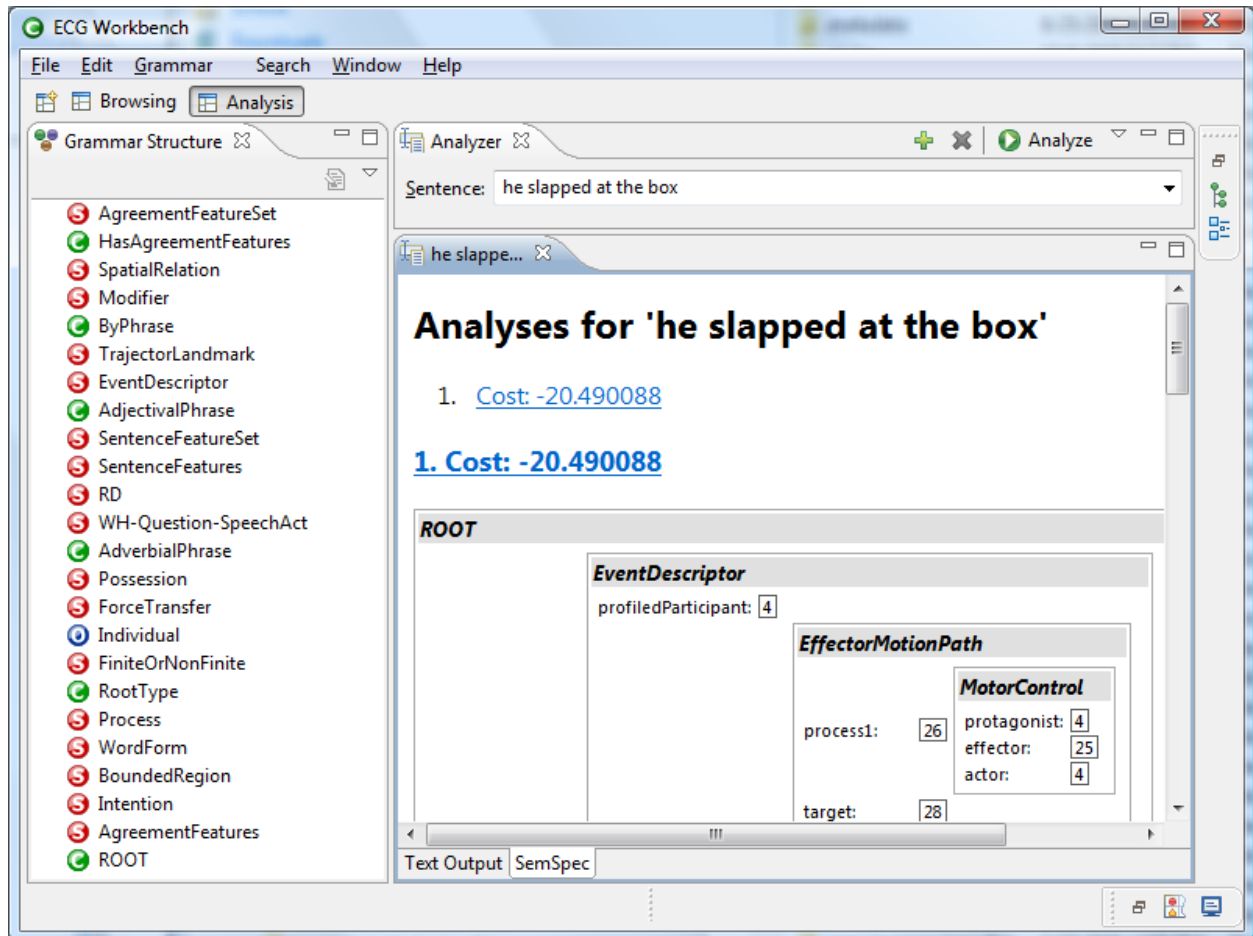
what you'll see:

Figure 4

To make even more room, 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:
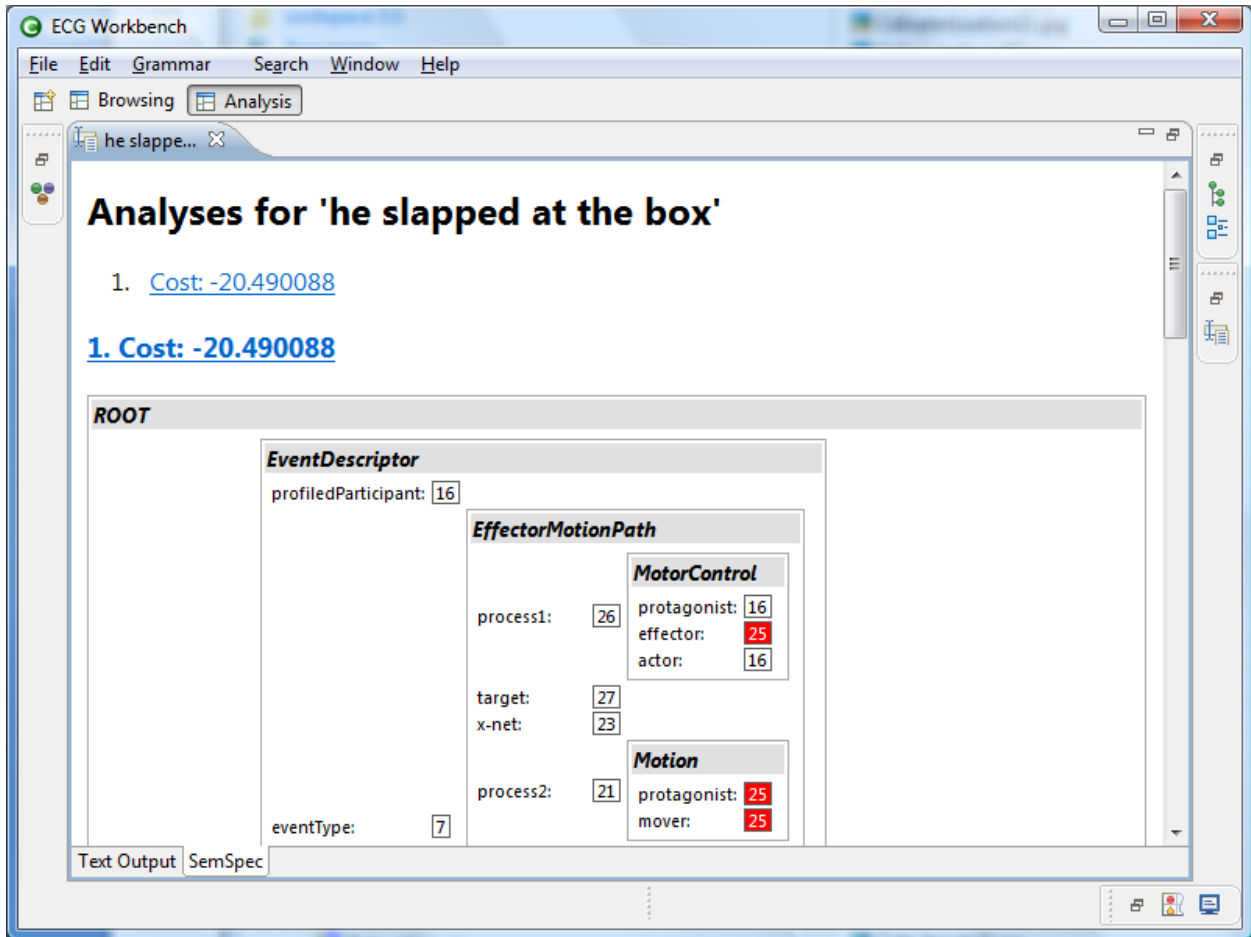
Figure 5 There are 2 Fig 5

As you can see, some of the coindexed elements, which also appear in square brackets in the

Text Output tab, are highlighted in red. This doesn't happen automatically, though: you have to

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 ⟨25⟩ next to **protagonist** in the **Motion** 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 a collapsed

grey bar, the contained element will expand again. From here you can print, by just selecting File

| Print from the main menu. To restore the Analysis result window to its original size, you have

again the options to hit again Ctrl+M (or Option+M on the Mac), 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 **starter2** grammar: **the cake slid into the box**. Press 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, while the old one will remain available as another 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 forward and back buttons to navigate back and forth to the "places" you've visited. There are no such buttons, at least 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 (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 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
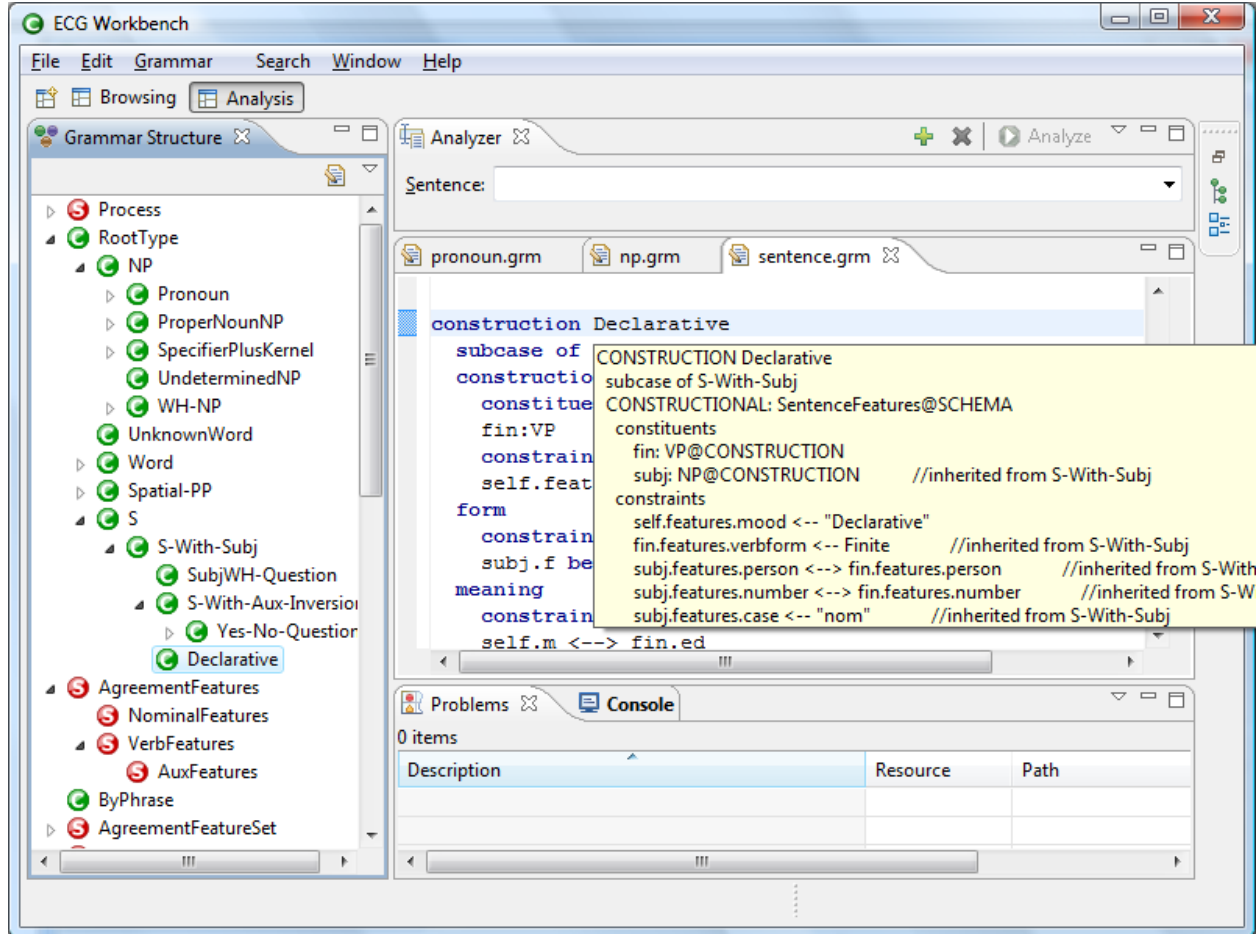
caption—will show up:

Figure 5 There are 2 Fig 5

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 (hint) box 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 with respect the file content. This is a feature, to show 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: SentenceFeatures@SCHEMA
  constituents
    fin: VP@CONSTRUCTION
    subj: NP@CONSTRUCTION                           // inherited from S-With-Subj
  constraints
    self.features.mood <-- "Declarative"
    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
    subj.m.referent <--> self.m.profiledParticipant // inherited from S-With-Subj
```

But the actual file content is a bit shorter, since it need not specify the structure defined by all its supertypes:

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

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 quite 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.

A note: this feature is a similar to that of Content View (the right pane in the Browsing Perspective). The only difference is that there you don't need to hold down the Ctrl/Option key to turn type identifiers into hyperlinks: they are already rendered as such (and colored in blue).

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 Search menu or by hitting Ctrl/Option+H. This will open the following dialog:
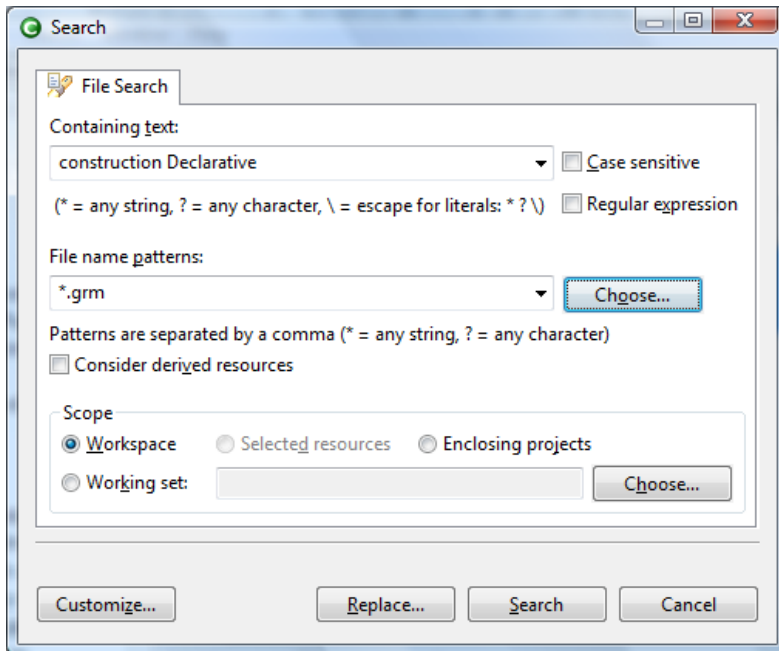
Ignoring the **Scope** group at the bottom, many of the other options are quite self explanatory.

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 **starter2**). A new tabbed Search View will be

opened at the bottom, as follows:
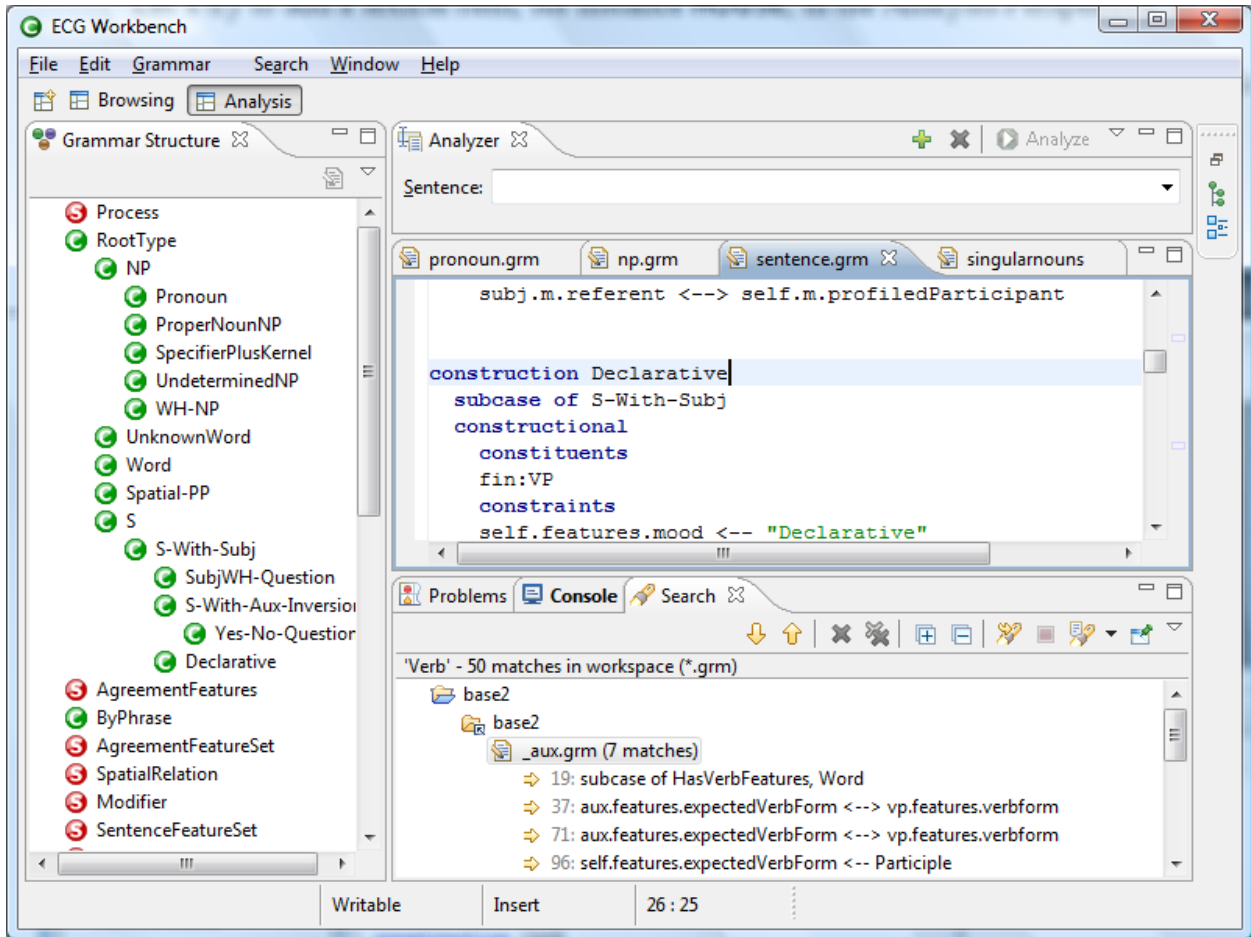
You can reach all the occurrences found by clicking on the lines indicated by the right-pointing arrows. The two up and downward arrows will walk you through all the matches. As with any other View, you can drag the Search view from the bottom to the sides, or minimize it (see above, Analyzing a sentence).

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 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. 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 **Help** 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 **Door**, in the Analysis Perspective. In the **starter2** grammar, many lexical items are defined in **singularnouns.grm**. Using Grammar Explorer in the right pane, open the **starter2** folder (by clicking on the triangle or plus at its left), scroll down to the **singularnouns.grm** node and double-click on it. This should be what you see:
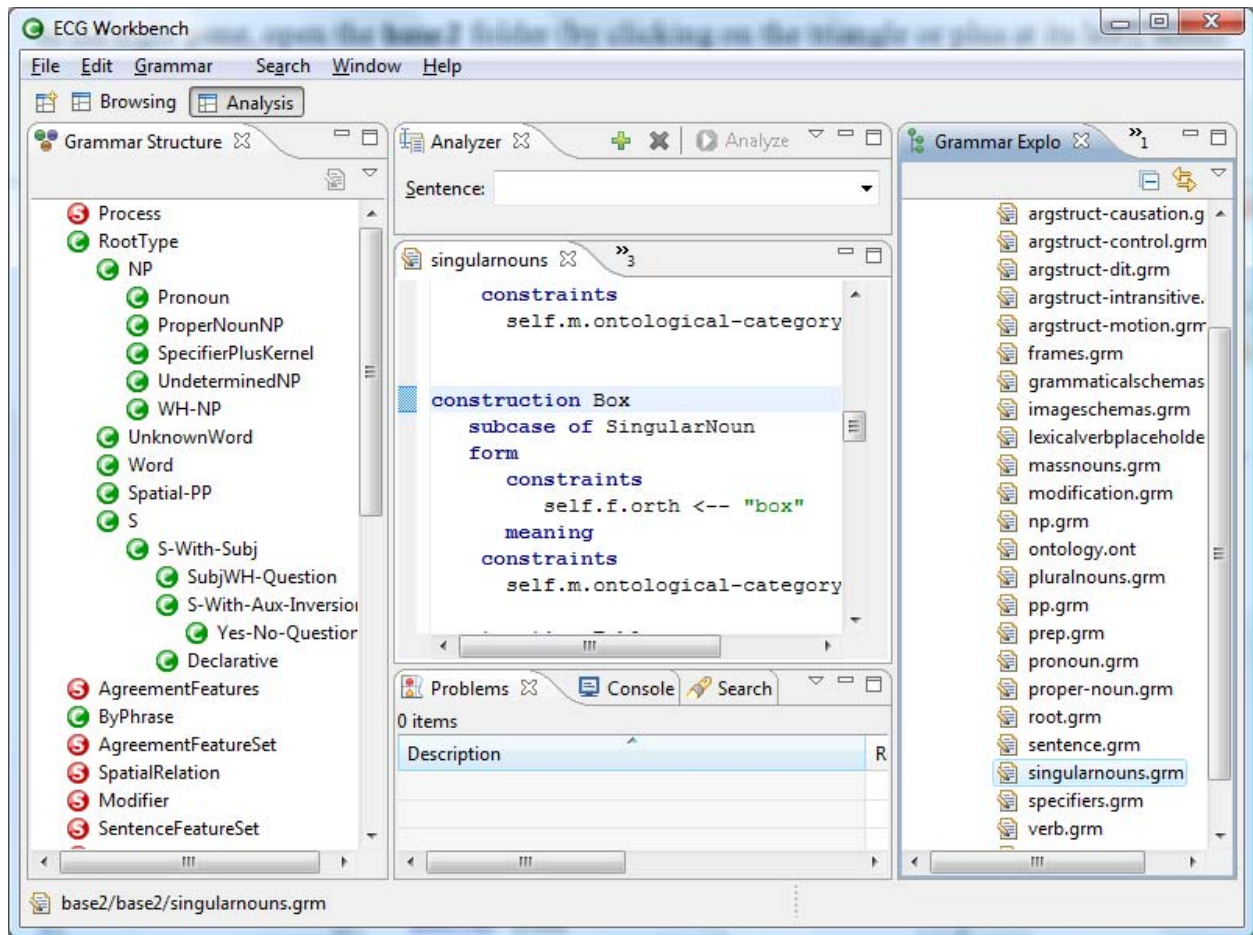
Figure 6

The easiest way to define a new lexical item is to copy and modify an old one. Modifying

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, 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), hit the

down arrow to cancel the selection 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). 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 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: you'll see various red icons signaling various (possibly spurious) errors that the grammar checker generated because it was unable to finish checking the **singularnouns.grm** file. You can go ahead and change the second **Table** into **Door**, and also the assignment to the **self.f.orth** role from "**table**" to "**door**." Save and check the grammar again: everything should 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 **table** (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 **table** type definition, replace **table** 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. In 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:
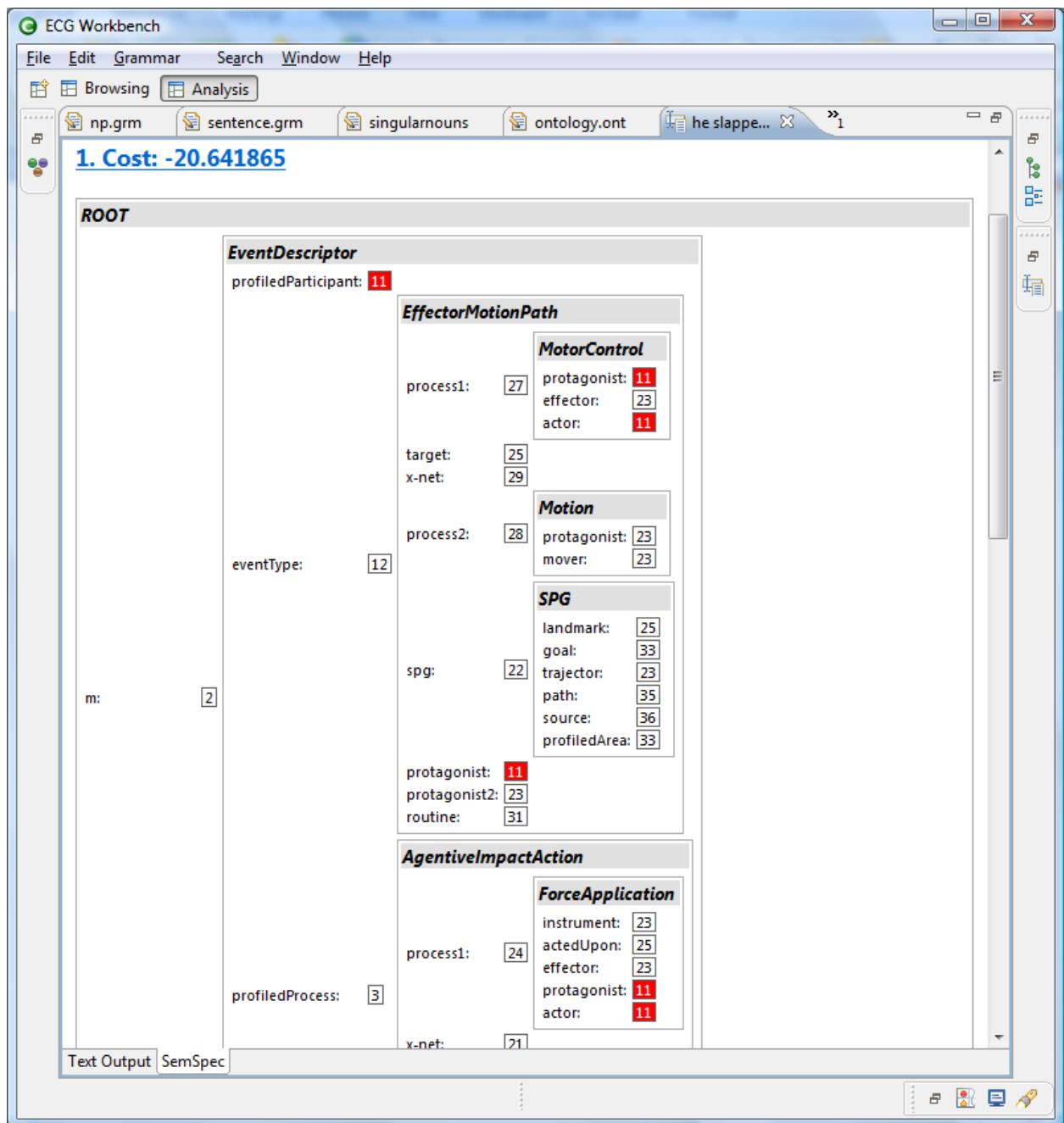
Figure 7

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).

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.
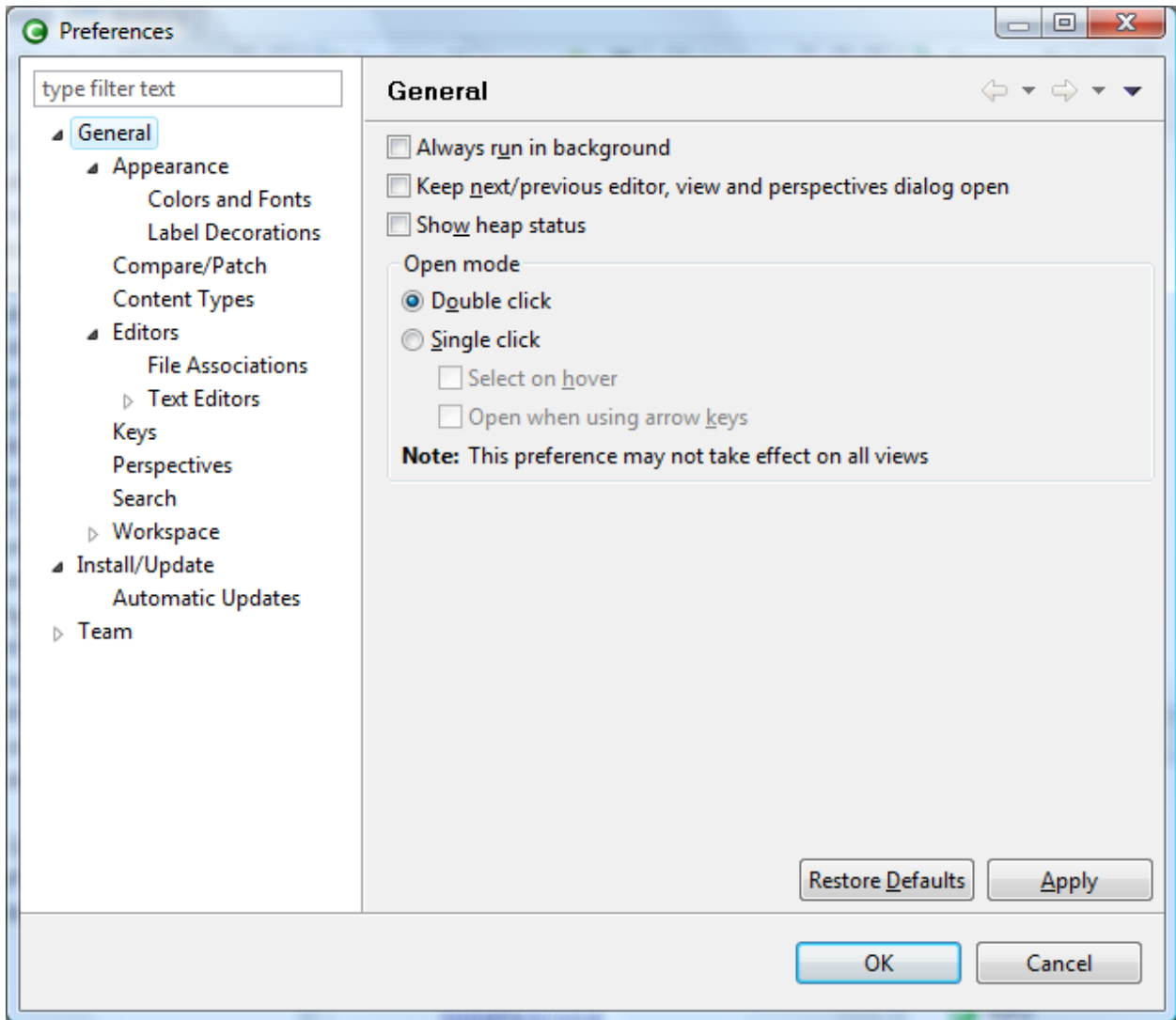
Figure 8

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.
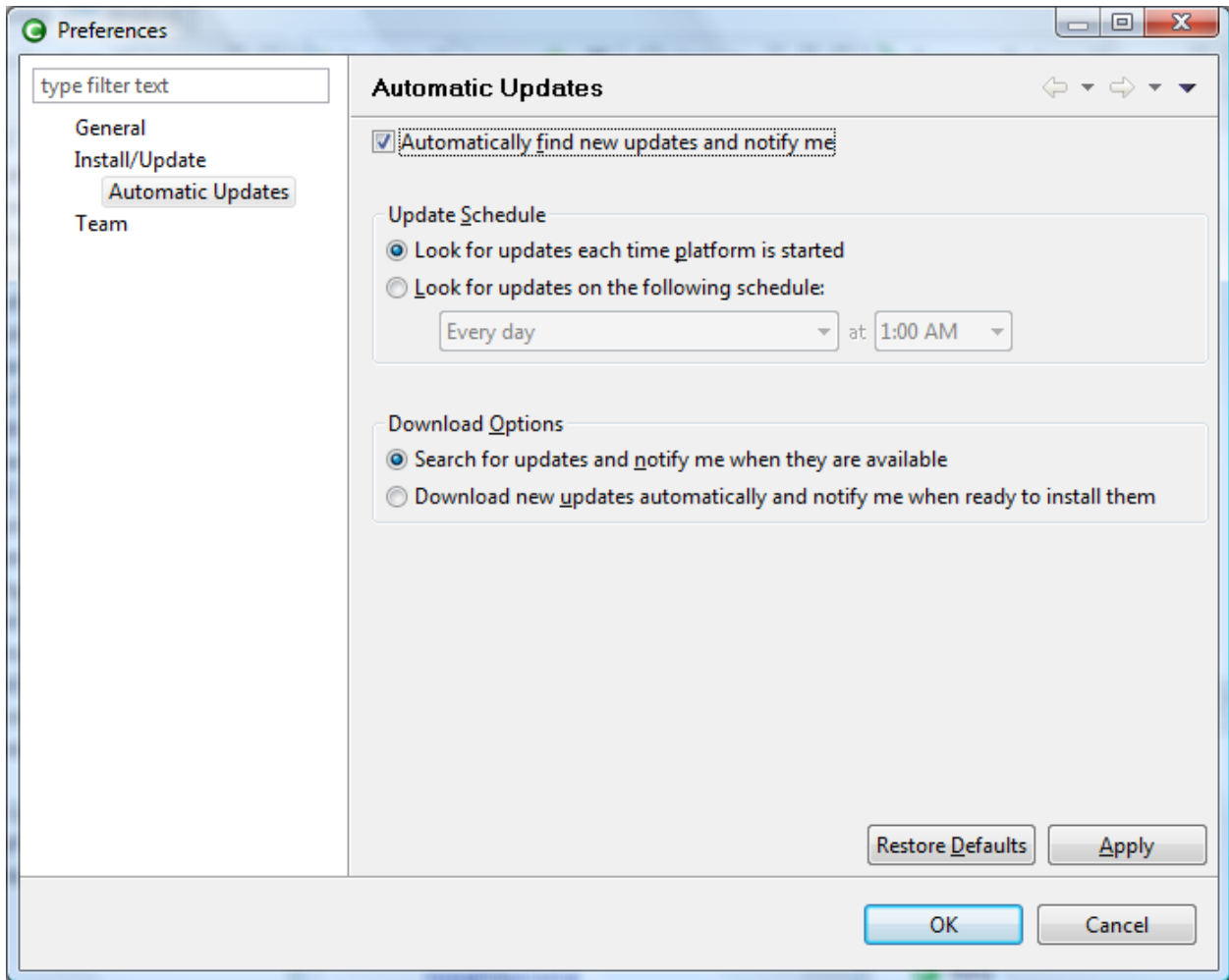
Figure 9

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. Here you can see the global list and you can change the key bindings to your liking:
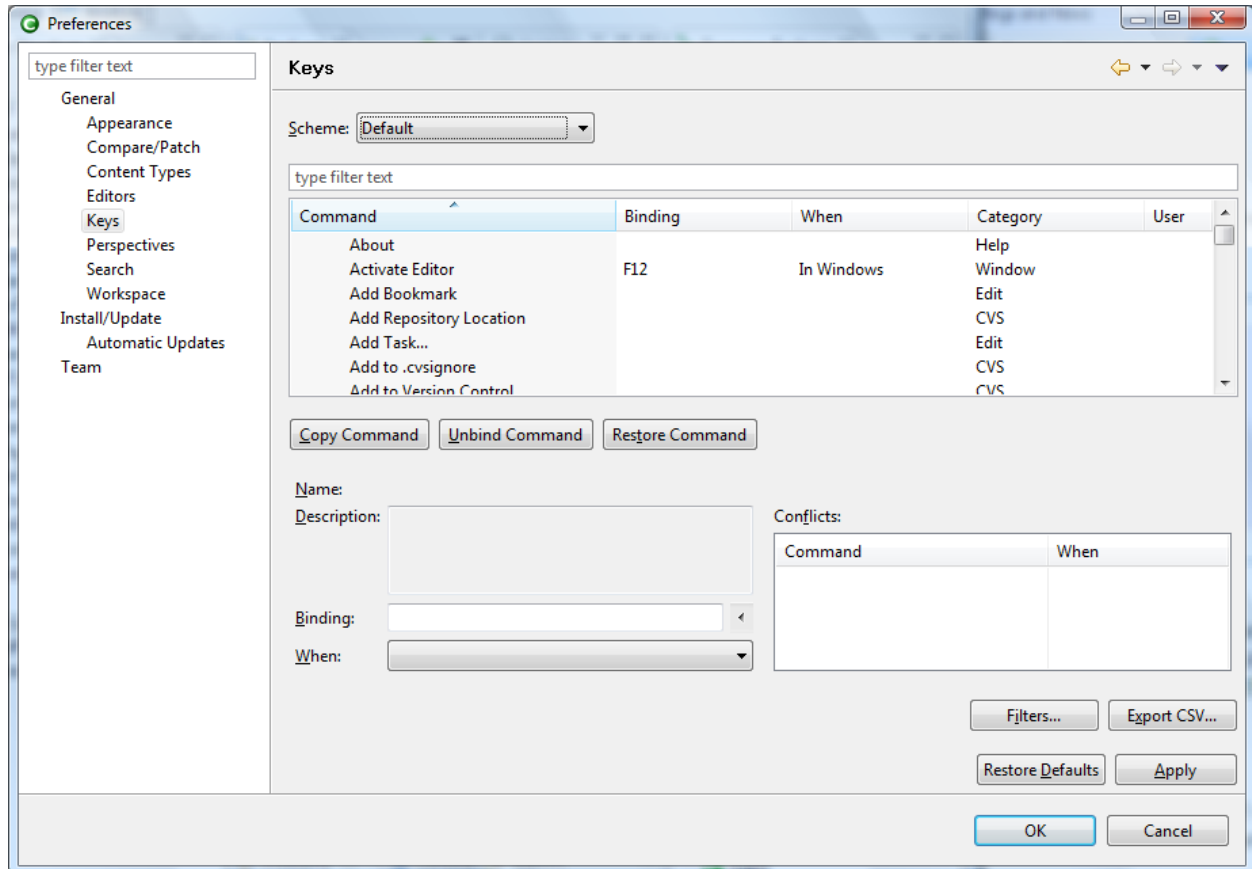
If you're used to Emacs, you can choose the Emacs scheme from the pull-down menu **Scheme**.

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, more or less like this:
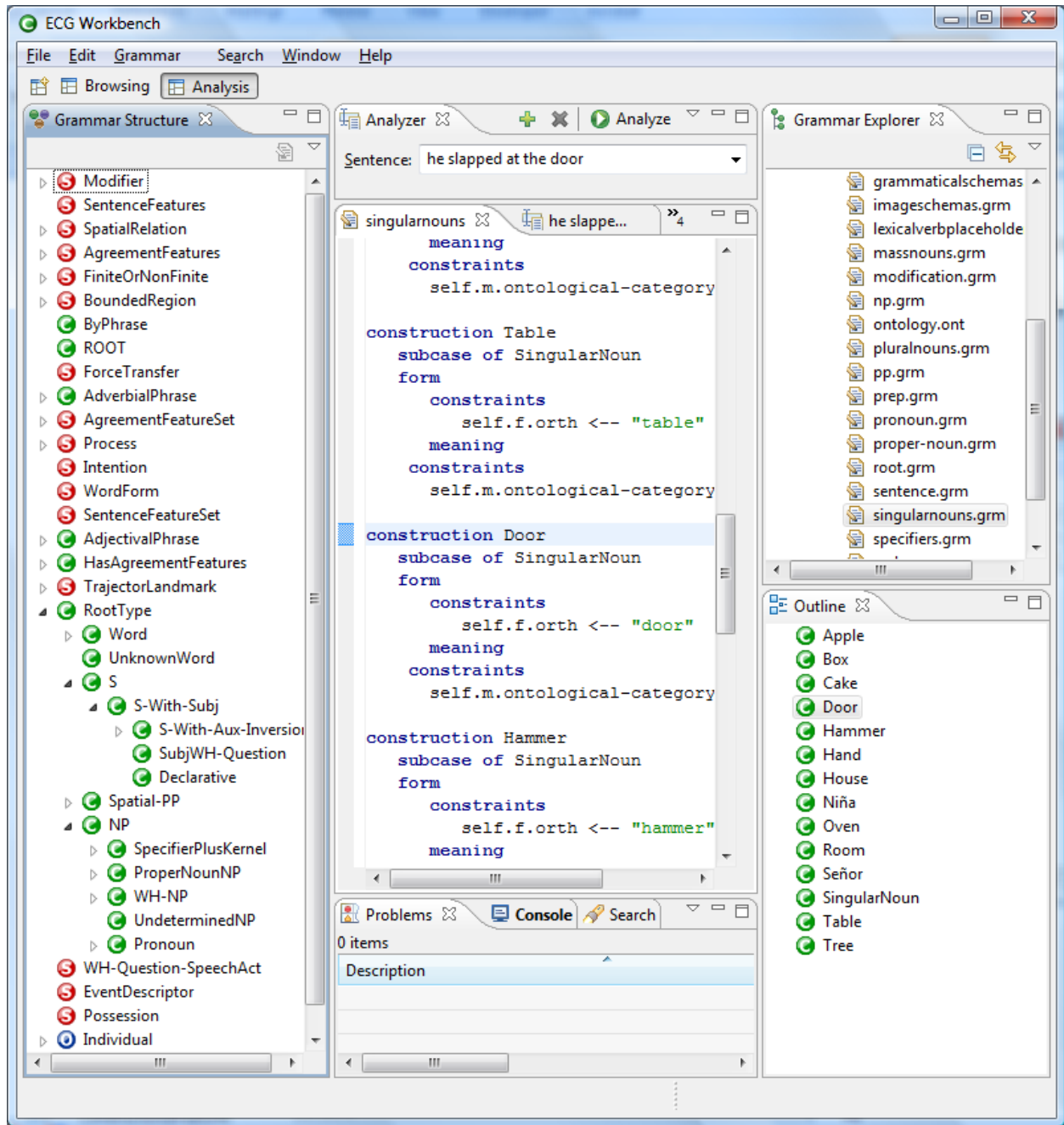

Figure 10

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)

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.

    o 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.

    o 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**.

- o Example:

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

References

- ECGweb Wiki: http://ecgweb.pbworks.com/