A Brief Introduction to the ECG Workbench and a First English Grammar

(V1.0, draft)

Authors: Luca Gilardi lucag@icsi.berkeley.edu

Jerome Feldman: feldman@icsi.berkeley.edu

ECG Workbench Website: http://www.icsi.berkeley.edu/icsi/projects/ai/ntl

Introduction

Welcome to ECG Workbench. This document is intended as a quick introduction to the main features of the program and to its use in building ECG grammars. 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+)

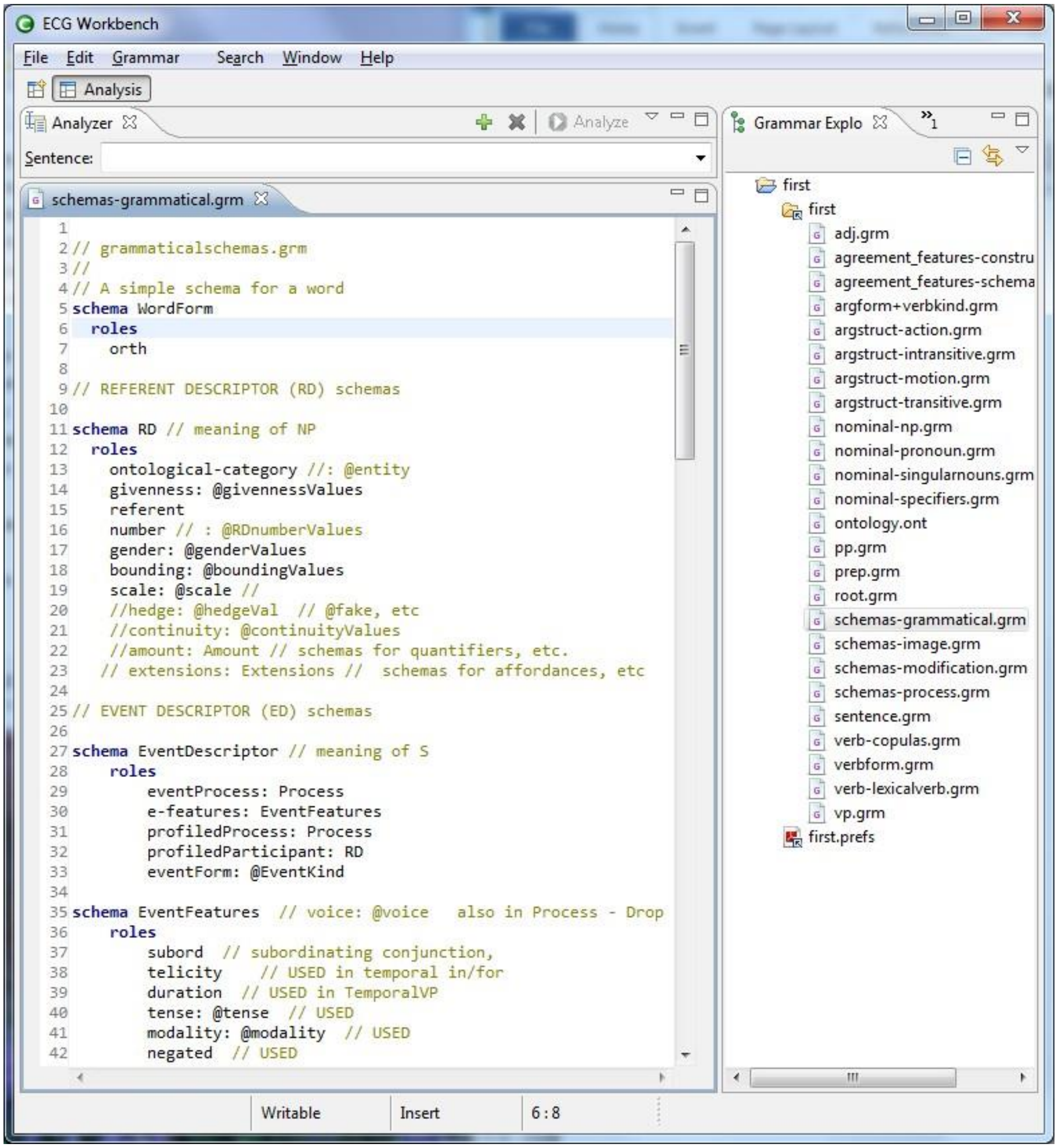
The archive file (a zip file) contains a directory named **ecg-workbench-<version number>.zip** on the site http://www.icsi.berkeley.edu/icsi/projects/ai/ntl . This directory (after having unzipped the archive) contains the executable file, the one marked with the blue ball icon. To start ECG Workbench, double-click on the blue icon.

How to open a grammar file set

ECG Workbench (which we'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 **first** grammar (contained in the file **first.zip**, which should be downloadable off of the same web page on which you found this file (http://www.icsi.berkeley.edu/icsi/projects/ai/ntl/) and the zip file containing EW itself, select **first .prefs**, and click OK. You should see something similar to the following:



```

// grammaticalschemas.grm
//
// A simple schema for a word
schema WordForm
  roles
    orth

// REFERENT DESCRIPTOR (RD) schemas

schema RD // meaning of NP
  roles
    ontological-category //: @entity
    givenness: @givennessValues
    referent
    number // : @RDnumberValues
    gender: @genderValues
    bounding: @boundingValues
    scale: @scale //
    //hedge: @hedgeVal  // @fake, etc
    //continuity: @continuityValues
    //amount: Amount // schemas for quantifiers, etc.
   // extensions: Extensions //  schemas for affordances, etc

// EVENT DESCRIPTOR (ED) schemas

schema EventDescriptor // meaning of S
    roles
        eventProcess: Process
        e-features: EventFeatures
        profiledProcess: Process
        profiledParticipant: RD
        eventForm: @EventKind

schema EventFeatures  // voice: @voice   also in Process - Drop
    roles
        subord  // subordinating conjunction,
        telicity    // USED in temporal in/for
        duration  // USED in TemporalVP
        tense: @tense  // USED
        modality: @modality  // USED
        negated  // USED
```

```
first
  first
    adj.grm
    agreement_features-constru
    agreement_features-schema
    argform+verbkind.grm
    argstruct-action.grm
    argstruct-intransitive.grm
    argstruct-motion.grm
    argstruct-transitive.grm
    nominal-np.grm
    nominal-pronoun.grm
    nominal-singularnouns.grm
    nominal-specifiers.grm
    ontology.ont
    pp.grm
    prep.grm
    root.grm
    schemas-grammatical.grm
    schemas-image.grm
    schemas-modification.grm
    schemas-process.grm
    sentence.grm
    verb-copulas.grm
    verbform.grm
    verb-lexicalverb.grm
    vp.grm
  first.prefs
```

Figure 1

Figure 1 shows what the screen looks like before any input has been analyzed. The right hand column shows the file structure that was used in specifying the “first” grammar. The large window shows some of the definitions in the schemas-grammtical.grm folder. The WordForm schema has a single role, “orth” that is used to specify the orthographic form of words as they are defined. Particularly important is the definition of the RD schema, starting in line 11. This is used generally in ECG for capturing the meaning of noun phrases. The roles that are light colored (commented out) are not used in this first grammar, but are important in the more general “base” grammar.

Perspectives

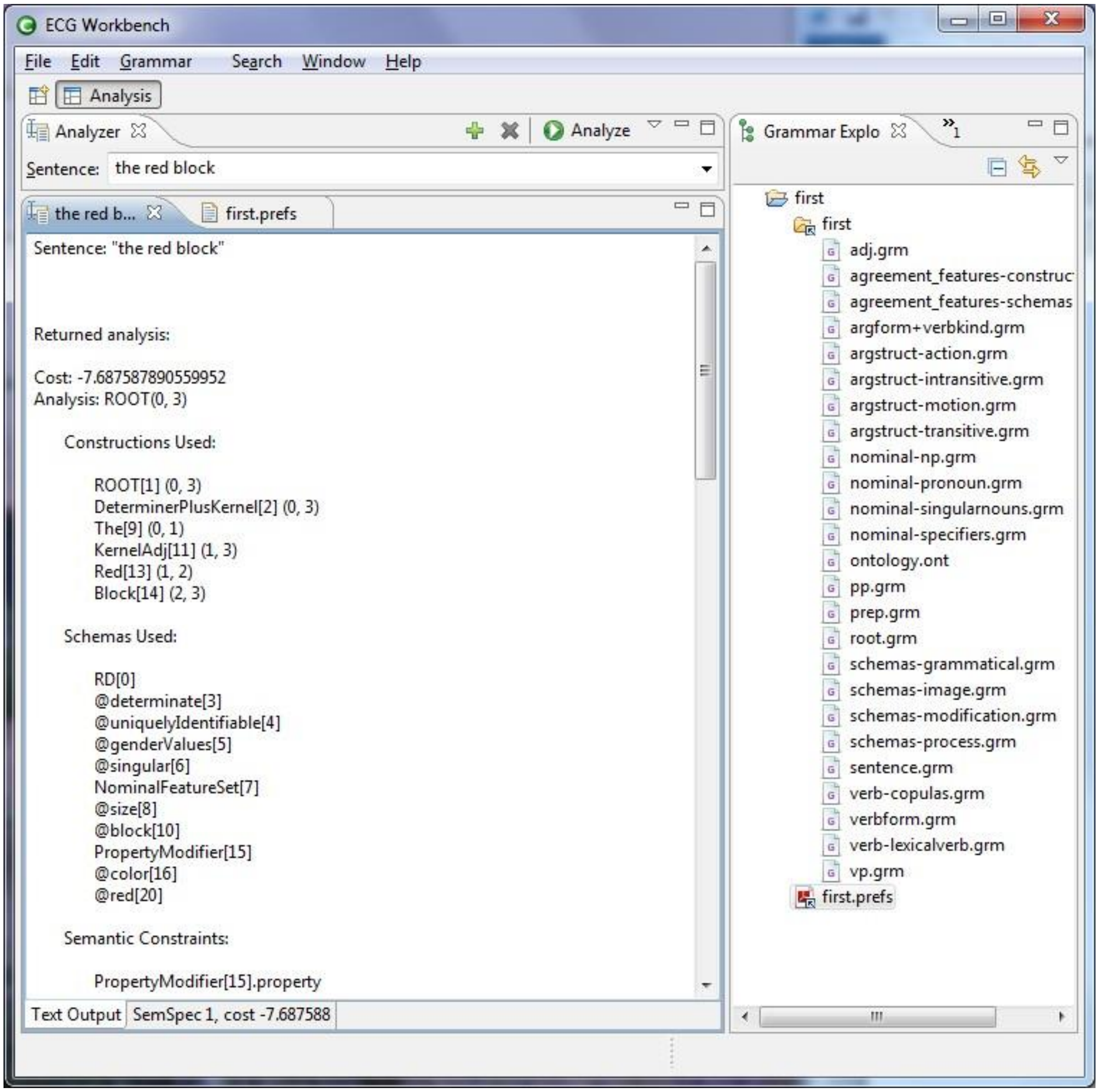
The Workbench is organized into Perspectives, we will use mainly Analysis (fig. 1) and occasionally Browsing. We’ll be using capitalized words to signal that they have a special meaning in the context of the workbench. Perspectives are what the word suggests: different way of looking at content—an ECG 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.

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 **Window | 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 **first.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 red block** 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 based on the sentence you just analyzed:



```
Sentence: "the red block"



Returned analysis:

Cost: -7.687587890559952
Analysis: ROOT(0, 3)

    Constructions Used:

        ROOT[1] (0, 3)
        DeterminerPlusKernel[2] (0, 3)
        The[9] (0, 1)
        KernelAdj[11] (1, 3)
        Red[13] (1, 2)
        Block[14] (2, 3)

    Schemas Used:

        RD[0]
        @determinate[3]
        @uniquelyIdentifiable[4]
        @genderValues[5]
        @singular[6]
        NominalFeatureSet[7]
        @size[8]
        @block[10]
        PropertyModifier[15]
        @color[16]
        @red[20]

    Semantic Constraints:

        PropertyModifier[15].property
```

Figure 2

As you can see (fig. 2), at the bottom of the 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:

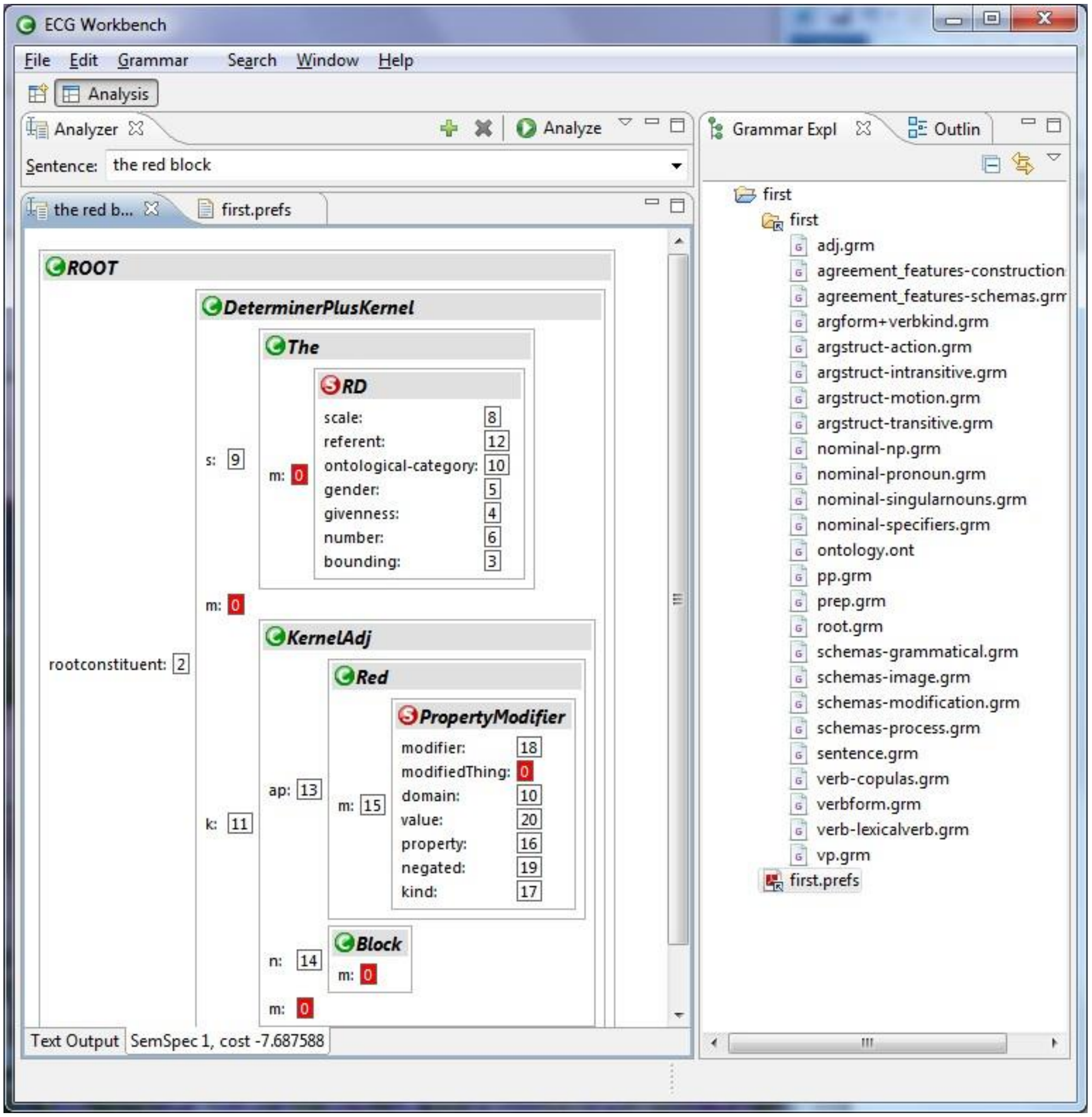
```
ROOT[2] (0, 3)
DeterminerPlusKernel[0] (0, 3)
The[10] (0, 1)
KernelAdj[3] (1, 3)
Red[15] (1, 2)
Block[14] (2, 3)
```

Recall that ECG constructions are the form-meaning pairs that drive analysis. The numbers in parentheses are the "spans" of the sentence that the construction recognized, starting from zero: $_0$ **the** $_1$ **red** $_2$ **block** $_3$. Thus the **Root** construction was used to recognize the entire phrase (it covers the words from 0 to 3), whereas **KernelAdj** spans words 1 to 3, and so on for the other constructions. In this grammar, noun phrases are well-formed inputs (ROOT). You can type a new sentence in the Analyzer Editor, replacing the previous sentence. Try **the red blocks** and **a red blocks.**

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 and depicted in Figure 3. 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 at the bottom opens a more graphical view of the analysis. The **SemSpec** tab for **the red block** contains the semantic specification (SemSpec) (Figure 3), that is, the Schemas involved in the analysis of the sentence together with their binding relationships. The format employed is very similar to that used for printing the attribute-value matrices that's common in the literature on unification grammars like HPSG (http://hpsg.stanford.edu/) , but it includes some additional features.

If you have clicked on the SemSpec tab, you might be seeing see only a part of the SemSpec (depending on your screen's resolution). You may need to make room for the Analysis Editor by clicking the minimize button at the upper right corner of the Grammar Explorer View tab. This is what you'll see:



```
ROOT
  rootconstituent: 2
  DeterminerPlusKernel
    s: 9
      The
        m: 0
          RD
            scale:                8
            referent:             12
            ontological-category: 10
            gender:               5
            givenness:            4
            number:               6
            bounding:             3
    m: 0
    k: 11
      KernelAdj
        ap: 13
          Red
            m: 15
              PropertyModifier
                modifier:         18
                modifiedThing:    0
                domain:           10
                value:            20
                property:         16
                negated:          19
                kind:             17
        n: 14
          Block
            m: 0
        m: 0
```

```
first
  first
    adj.grm
    agreement_features-construction
    agreement_features-schemas.grm
    argform+verbkind.grm
    argstruct-action.grm
    argstruct-intransitive.grm
    argstruct-motion.grm
    argstruct-transitive.grm
    nominal-np.grm
    nominal-pronoun.grm
    nominal-singularnouns.grm
    nominal-specifiers.grm
    ontology.ont
    pp.grm
    prep.grm
    root.grm
    schemas-grammatical.grm
    schemas-image.grm
    schemas-modification.grm
    schemas-process.grm
    sentence.grm
    verb-copulas.grm
    verbform.grm
    verb-lexicalverb.grm
    vp.grm
  first.prefs
```

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. 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🗗.

As you can see, some of the co-indexed 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 others that have the same value will light up as well. For instance, if you click on the **0** next to **modifedThing** in the **PropertyModifer** schema, all the other co-indexed roles will be colored red. In this case, this includes the whole phrase, the RD and the word Block. Having modifiers point to the thing modified (instead of the converse) supports more general cases.

If you hover over a boxed number, the value associated with that index will be temporarily displayed. For example if you hover over the **10** , which is the **scale** role in the **RD** schema, a box with word “size” will appear. This shows that size is the default scale property for the concept Block.

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 a SemSpec, by selecting File | Print from the main menu.

The SemSpec is the main link between language analysis and subsequent action, as described in Reference 2. The goal of writing an ECG grammar is to capture (in the SemSpec) all the semantics expressed by an input utterance. This almost always underspecifies the full intent of the utterance in context, which must be processed by subsequent stages (Reference 2).

Analyzing a sentence not in the dropdown 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*, we'll explain how to add a lexical item. You can type a new sentence in the Analyzer Editor, replacing the previous sentence. Let's type in a new one that will work with the **first** grammar: **he moved into the room**. 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. The Constructions Used in the analysis are:

```
ROOT[1] (0, 5)
Declarative[0] (0, 5)
He[13] (0, 1)
ActiveMotionPath[7] (1, 5)
MovePast[34] (1, 2)
Path-PP[27] (2, 5) INTO-
Path[41] (2, 3)
DeterminerPlusKernel[39] (3, 5)
The[54] (3, 4)
KernelNoAdj[56] (4, 5)
Room[59] (4, 5)
```

This is a complete sentence involving several parts of speech and phrases. If you try **he moved into the block,** the system will complain: No complete analysis found. This is because, in our first grammar, "room" but not "block" is specified to be a container and the definition of INTO-Path requires that the object of this preposition be a kind of container. As we saw in Figure 1, you can view (and even change) any ECG definition when it is displayed in the main Editor window

You might want to click the SemSpec tab, but the resulting display will be too large for the screen, as happens with all complex examples. It is often useful to close boxes (clicking on the grey bar) that are not of current interest. For example, the last 4 constructions above cover a simplified version of our first example.

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.

Figures 1 -3, have only the Grammar Structure View in the right column. You can open other views by clicking on “Window” the top bar , follow the pointer “Open View” and selecting one of the choices. In the Grammar Structure View (set up on the left), Constructions are marked by the Ⓒ icon, Schemas are marked byⓈ, and Ⓞ identifies ontology elements. Another useful view is Outline which lists each element of the currently active Editor. There is also a conventional Editor available in the top bar.

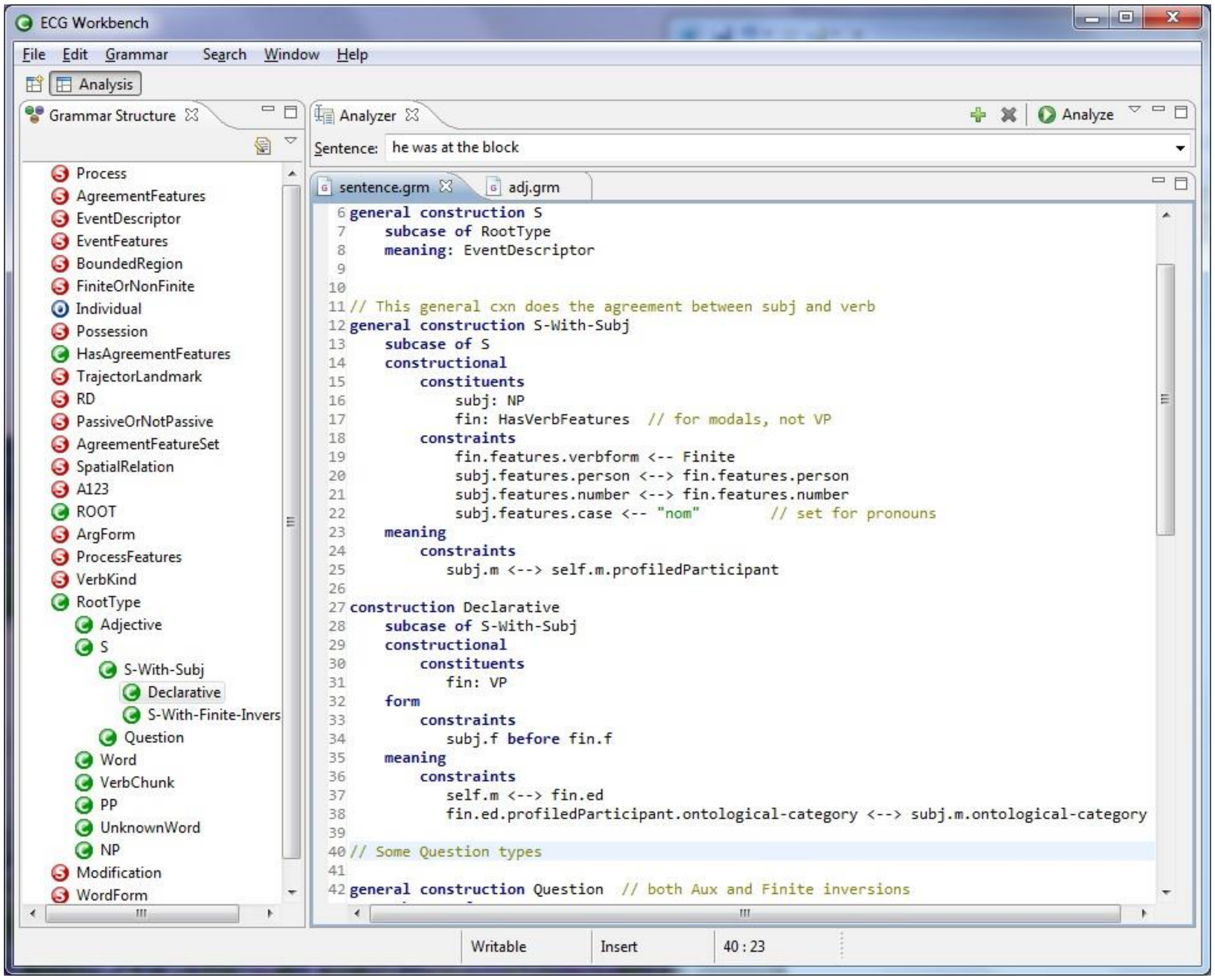
If you analyzed some sentences as described above, you'll see tabs for the two Editors containing the results of each analysis are still there. Editors are created by EW and never closed automatically. The tabs that accumulate in the upper part have an "X" icon that you can use to close them. At the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

Now let's open another window, the Grammar Structure View (Window/Open View/ Grammar Structure) , which appears on the left and shows all the grammar elements that are available—from all the files making up the grammar—in a hierarchical view. Clicking on a triangle, or plus depending on our platform, to 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 triangle at the left of the construction **RootType (**marked by ●), you'll see that, among others, it contains a node called **S**, which in turn contains **S-with-Subj**, which in turn contains **Declarative** and **S-With-Finite-Inversion**. This is because these are both subcases of **S-with-Subj**.

If you goto the Grammar Structure View and and navigate to the **Declarative** item and click on its and you'll see that its actual definition will be shown in the center, where a new tabbed window—a new Editor showing **sentence.grm** in its caption—will appear:

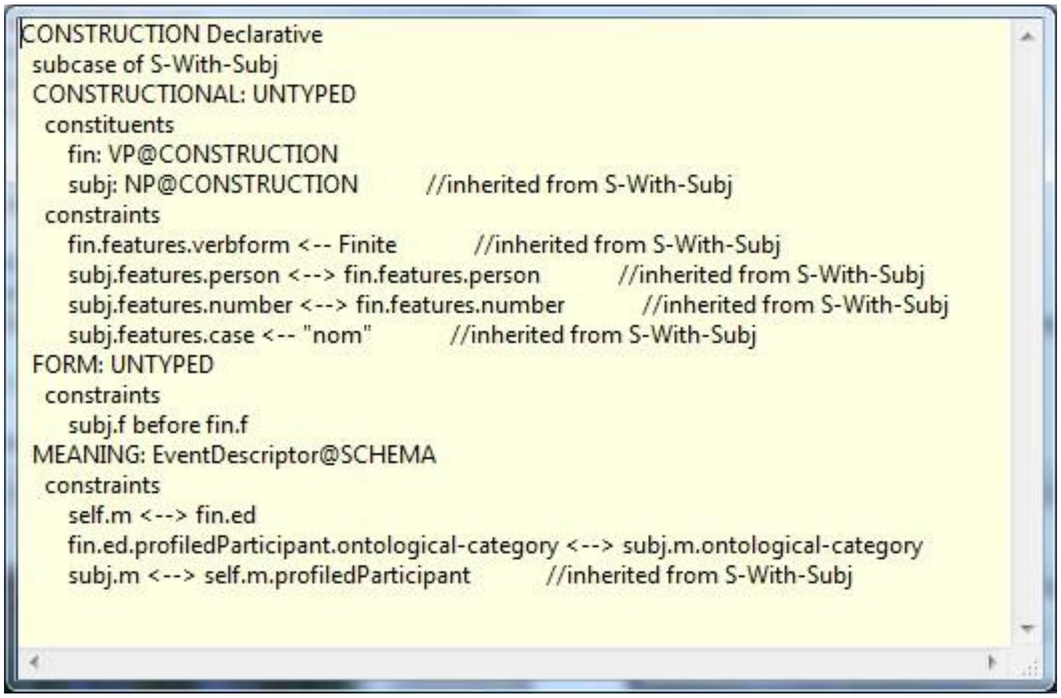


```
general construction S
    subcase of RootType
    meaning: EventDescriptor


// This general cxn does the agreement between subj and verb
general construction S-With-Subj
    subcase of S
    constructional
        constituents
            subj: NP
            fin: HasVerbFeatures  // for modals, not VP
        constraints
            fin.features.verbform <-- Finite
            subj.features.person <--> fin.features.person
            subj.features.number <--> fin.features.number
            subj.features.case <-- "nom"        // set for pronouns
    meaning
        constraints
          subj.m <--> self.m.profiledParticipant

construction Declarative
    subcase of S-With-Subj
    constructional
        constituents
          fin: VP
    form
        constraints
          subj.f before fin.f
    meaning
        constraints
          self.m <--> fin.ed
          fin.ed.profiledParticipant.ontological-category <--> subj.m.ontological-category

// Some Question types

general construction Question  // both Aux and Finite inversions
```

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 various elements. Figure. 5 below shows 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:



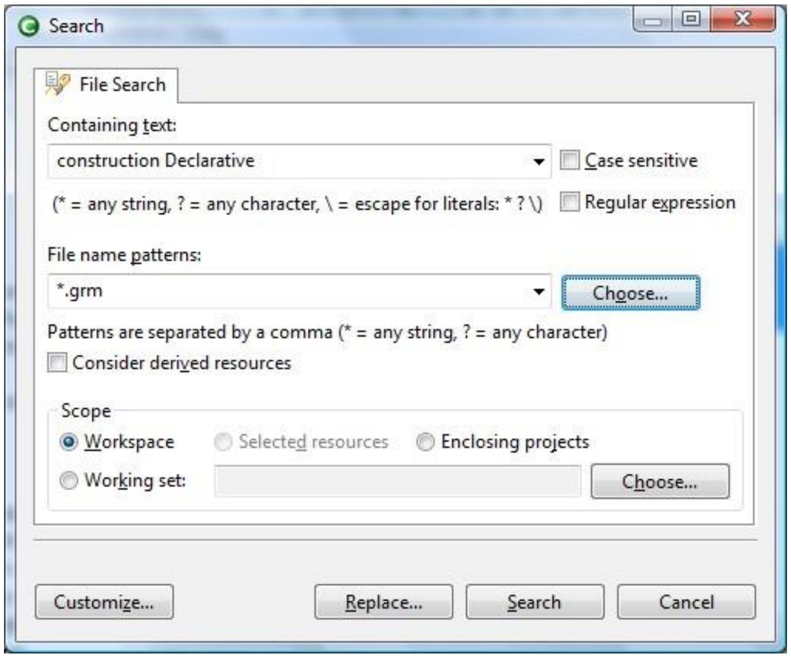
```
CONSTRUCTION Declarative
 subcase of S-With-Subj
 CONSTRUCTIONAL: UNTYPED
  constituents
    fin: VP@CONSTRUCTION
    subj: NP@CONSTRUCTION        //inherited from S-With-Subj
  constraints
    fin.features.verbform <-- Finite         //inherited from S-With-Subj
    subj.features.person <--> fin.features.person         //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
    fin.ed.profiledParticipant.ontological-category <--> subj.m.ontological-category
    subj.m <--> self.m.profiledParticipant         //inherited from S-With-Subj
```

Figure 5. The hint box for the Declarative construction

But the text file content is a shorter, since it omits the structure defined by its supertypes. The hint box shows all the inherited elements from the constructions that **Declarative** is a subcase of. The comments (generated automatically) specify which elements are inherited and from which supertype. For example, it shows that **Declarative** inherits a **subj** role from **S-With-Subj**, as well as four constructional constraints, and a meaning constraint.

Third, search and replace (local to the file shown in the title) can be done by choosing **Edit | Find/Replace** (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:



construction Declarative

(* = any string, ? = any character, \ = escape for literals: * ? \)

*.grm

Patterns are separated by a comma (* = any string, ? = any character)

Figure 6

Ignoring the **Scope** group at the bottom, many of the other options are self-explanatory. 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 **first**).

Exploring and modifying the grammar

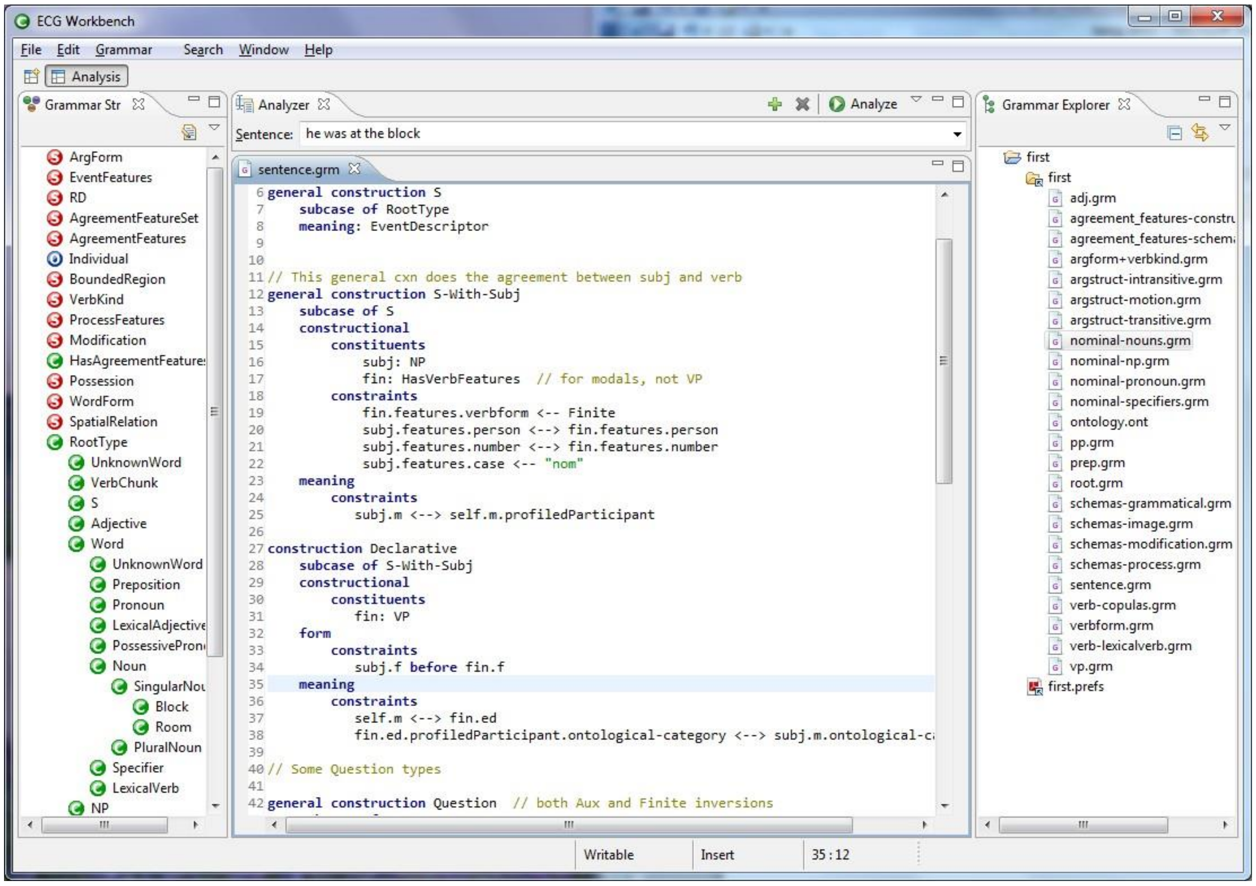
Let's take a look at the Grammar Explorer View on the right of Figure 1. This shows all the files that are part of the grammar, the ones that are named by the preferences file. In the

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, hovering the mouse pointer over type identifiers 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 **first** grammar, two lexical items are defined in **nominal-nouns.grm**. Using Grammar Explorer in the right pane, open the **first** folder (by clicking on the triangle or plus at its left), scroll down to the **nominal-nouns.grm** node and double-click on it. This should be what you see:



Sentence: he was at the block

```
general construction S
    subcase of RootType
    meaning: EventDescriptor


// This general cxn does the agreement between subj and verb
general construction S-With-Subj
    subcase of S
    constructional
        constituents
            subj: NP
            fin: HasVerbFeatures  // for modals, not VP
        constraints
            fin.features.verbform <-- Finite
            subj.features.person <--> fin.features.person
            subj.features.number <--> fin.features.number
            subj.features.case <-- "nom"
    meaning
        constraints
           subj.m <--> self.m.profiledParticipant

construction Declarative
    subcase of S-With-Subj
    constructional
        constituents
           fin: VP
    form
        constraints
           subj.f before fin.f
    meaning
        constraints
           self.m <--> fin.ed
           fin.ed.profiledParticipant.ontological-category <--> subj.m.ontological-c

// Some Question types

general construction Question  // both Aux and Finite inversions
```

Figure 7

When expanded, the left column shows the grammar structure window with the Noun node expanded to show the 2 SingularNoun items in our First grammar. The center pane contains an editable version of the ECG definitions of the words in nominal-nouns.grm and a grammatical construction SingularNoun.

The easiest way to define a new lexical item is to copy and modify an old one. Modifying the construction for block seems the right thing to do. In larger file, you could scroll through the code file in the middle pane to find the code for **Block**. 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 **Block**, copy and paste it. Now you have two **Block** 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: Block*. 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 **nominal-noun.grm** file. You can go ahead and change the second **Block** into **Door**, and also the assignment to the **self.f.orth** role from "**block**" 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 **block** (in **@block**, the "**@**" sign signals that **block** is actually an element in the ontology). Try to change **@block** into **@door** in the new construction, and then save and check the grammar. At this point the following error should be marked with a next to your new **Door** construction: In Door, constraint ⊗ self.m.ontological-category <-- @door has undefined role: @door in slotchain @door

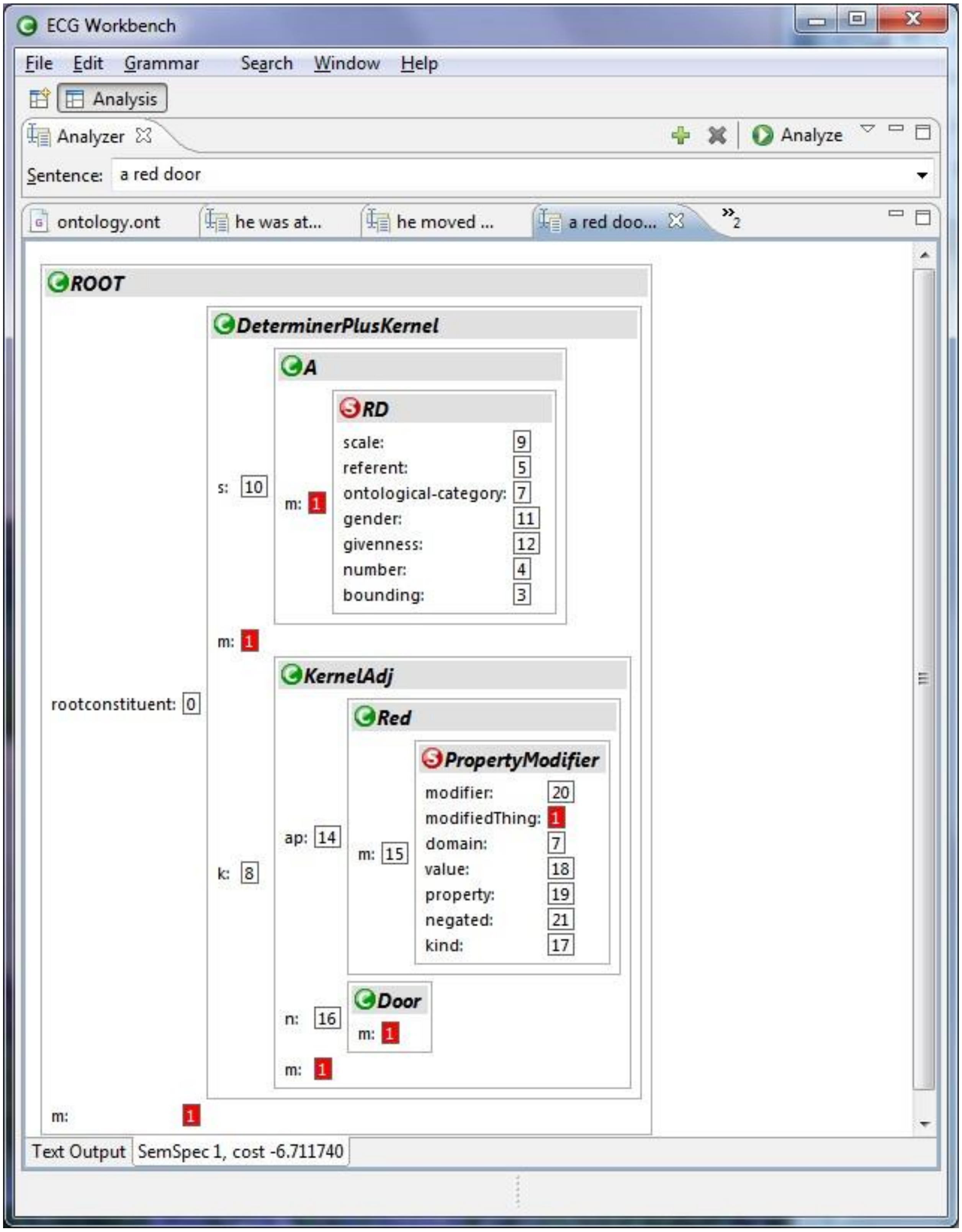
This rather cryptic message means that there’s something wrong in the meaning pole of your new construction and you have to add the appropriate element to the ontology.

To do this, go to the Grammar Explorer, find the node for **ontology.ont**, and double click on it. The ontology associated with first is much larger than needed; it should be close to adequate for the much more ambitious **base** grammar. The best way to work with it is to use the Outline view. The ontology is important in more complex grammars; for example the entry for room is:

`(type room sub container)` and this supports uses like “into the room”. We need an ontology entry for

Door that will capture some of its semantics. For now, we can just copy the entry for table and add `(type door sub artifact) .!`

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 **a red door**, hit enter, and then Analyze. A new Analysis Editor should appear; click on the SemSpec tab and you get the following analog of Figure 2.



Sentence: a red door

ROOT
DeterminerPlusKernel
A
RD
scale: 9
referent: 5
ontological-category: 7
gender: 11
givenness: 12
number: 4
bounding: 3
s: 10
m: 1
m: 1
KernelAdj
Red
PropertyModifier
modifier: 20
modifiedThing: 1
domain: 7
value: 18
property: 19
negated: 21
kind: 17
ap: 14
m: 15
rootconstituent: 0
k: 8
Door
n: 16
m: 1
m: 1
m: 1

SemSpec 1, cost -6.711740

Figure 8

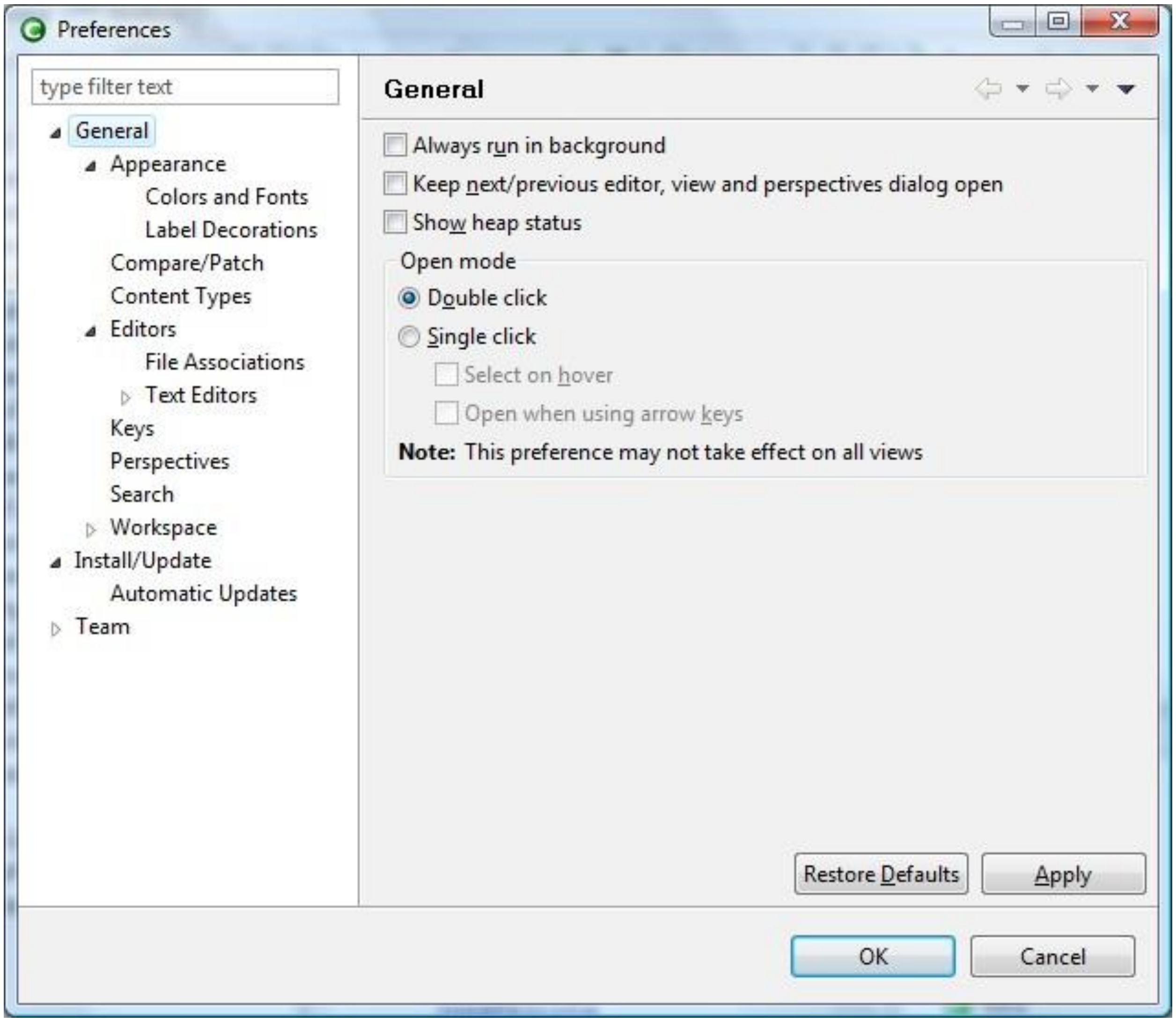
As before, the various roles associated with the head noun, here "door" are all highlighted with red boxes, once you select the *m* role of the *Root* construction-box (seen at the top). This last example terminates this small tutorial. It is much more difficult to effectively add schemas or constructions to an ECG grammar. As in any complex system, interactions with existing structures all need to be worked out. If you have problems or comments, please email the authors (see at the beginning of this document).

Some advanced features

This section gathers together a few tricks for 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.



Preferences

type filter text

- General
  - Appearance
    - Colors and Fonts
    - Label Decorations
  - Compare/Patch
  - Content Types
  - Editors
    - File Associations
    - Text Editors
  - Keys
  - Perspectives
  - Search
  - Workspace
- Install/Update
  - Automatic Updates
- Team

General

- [ ] Always run in background
- [ ] Keep next/previous editor, view and perspectives dialog open
- [ ] Show heap status

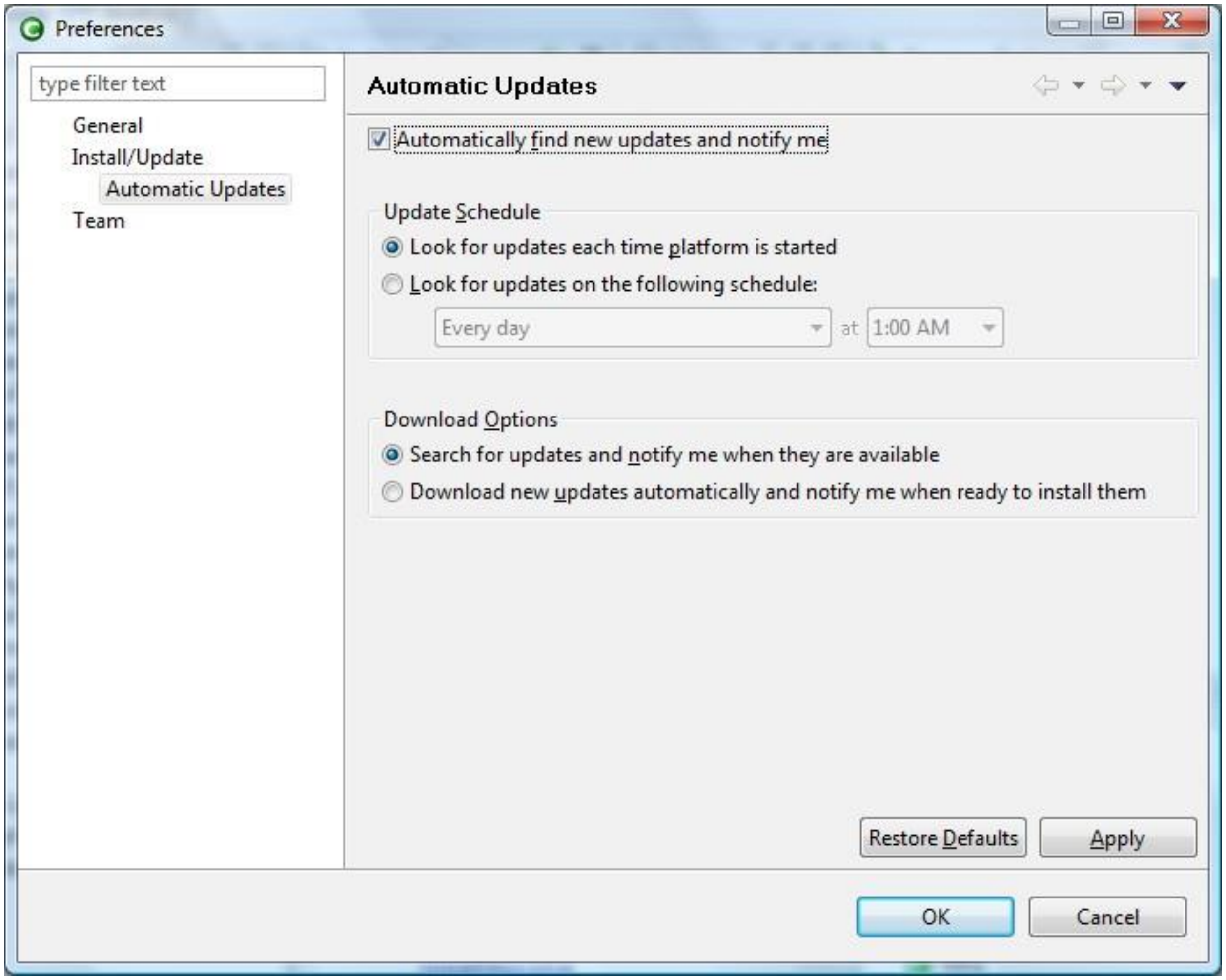
Open mode

- [✓ Double click]
- Single click
  - [ ] Select on hover
  - [ ] Open when using arrow keys

Note: This preference may not take effect on all views

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.


Preferences

type filter text

- General
- Install/Update
  - Automatic Updates
- Team

Automatic Updates

[✓] Automatically find new updates and notify me

Update Schedule

(●) Look for updates each time platform is started

( ) Look for updates on the following schedule:

Every day at 1:00 AM

Download Options

(●) Search for updates and notify me when they are available

( ) Download new updates automatically and notify me when ready to install them

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 (replacing 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**.

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).

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 ::==
            ./first
    ;
    ```
- **ONTOLOGY_PATHS**: a newline-separated list of files (not directories) if **ONTOLOGY_EXTENSIONS** is not defined that will be looked up as ontology files,

r a list of directories (as in **GRAMMAR_EXTENSIONS**) otherwise. Same format as **GRAMMAR_PATHS**.

- Example:

```
ONTOLOGY_PATHS ::==
        ./first/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