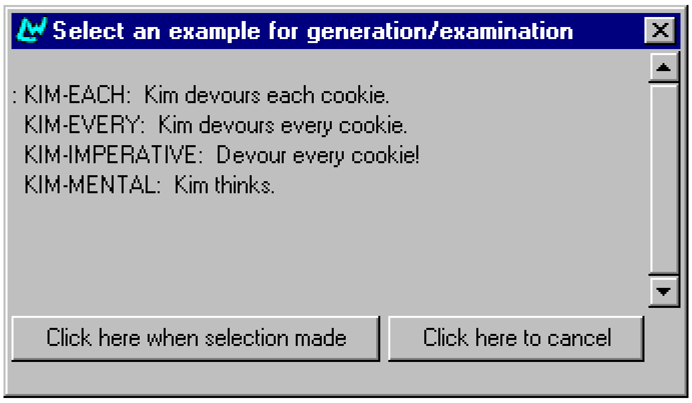
Tutorial 2: working with semantics...

Up until now, you have had to do all the work of choosing grammatical features and directing the generator to take the particular grammatical choices that you wanted to occur. When we are working with these functional grammars it is often more convenient and more useful to have the system making the choices itself. This it can only do, of course, if it has some idea as to what might be the right or appropriate choice to make at any point. We acheive this by giving it a *semantic specification*, i.e., an indication of what the clause or sentence that it is trying to generate should mean.

In this tutorial we see how this is managed with functional grammars of the kind you have been building.

1. First, start up the basic KPML novice tool again.
2. Make sure that the 'language' that you are using is the one called 'KIM-ENGLISH'. You can see this by looking at the Development window and reading what language is being displayed in the middle command window. You should see something like "KIM-ENGLISH:KPML>"—if you do not, change the langauge with the Set Language command in the Development window.
3. To generate using this grammar, we no longer tell the system which features to choose. We select instead an example semantic specification. The list of semantic specifications that the system already knows about can be brought up in the menu Generate Sentence. This should bring up a menu looking like the following:


Select an example for generation/examination
: KIM-EACH: Kim devours each cookie.
KIM-EVERY: Kim devours every cookie.
KIM-IMPERATIVE: Devour every cookie!
KIM-MENTAL: Kim thinks.
Click here when selection made
Click here to cancel

Select one of these options, say KIM-EACH, from the menu and click the lower left button.

The result should be shown (after a few messages from the system that you can ignore for the present) in the usual Development window lower window pane.

4. Look at the structure generated, by clicking on the Graph Structure command as usual. This structure was generated in exactly the same way as all the other structures that you have seen so far. The only difference is that this time, the system did not need to ask you which features to take.
5. The *input* to this generation process, i.e., the specification that makes the entire thing work, was the following:

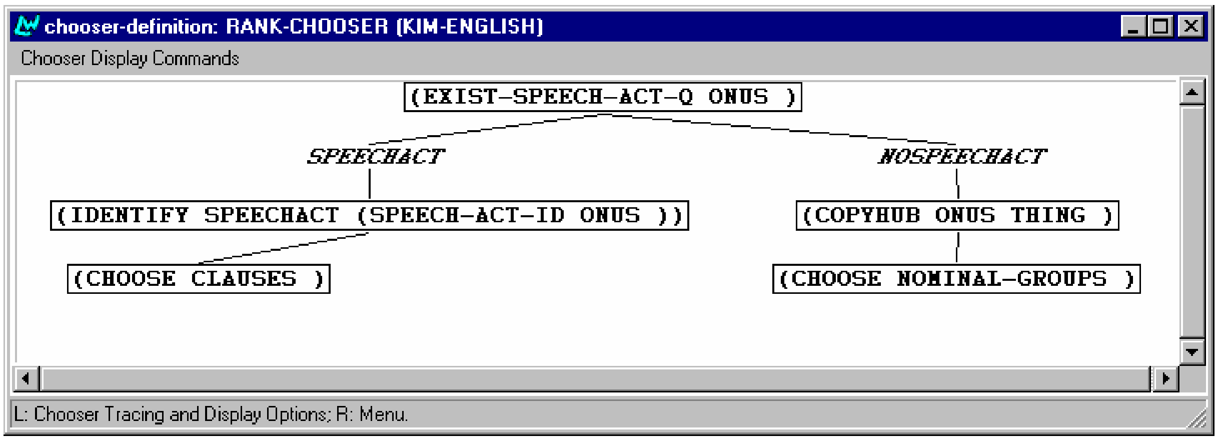
```
(D / DISPOSITIVE-MATERIAL-ACTION
          :LEX DEVOUR
          :ACTOR
          (P / PERSON
               :FAVOR-Q CLASSIFY
               :LEX KIM )

          :ACTEE
          (C / OBJECT
           :set-totality-individuality-q individual
           :LEX COOKIE )
          )
```

This says that there was some kind of **action** (dispositive-material) and this action was done by an *actor* (called p) and was done to something (the *actee*, called c). Further, the actor was a kind of **person**, which a particular name, and the actee was a kind of **object**. The complicated looking "`:set-totality-individuality-q individual`" is simply the way that we specify that we want some kind of *quantifier*, such as each, every, all and so on to be used.

The best way to find out about these semantic specifications is to play with them—taking some basic examples and experimenting with them, changing them to see what comes out. This can only work with a large grammar of course: otherwise it would be easy to make some semantics that the grammar was not clever enough to deal with. The KIM-ENGLISH grammar is again a very small grammar (you can take a look at it in the normal way, by graphing it from the Inspect menu command), however, so we will only use it here to see how the grammar is making its choices on the basis of the given semantics.

6. Graph the RANKING region of the grammar and find the RANK system. If you now **right** click on the name of the system, you will get a further set of options. Select the Show Chooser ... option. This will bring up the following window:


chooser-definition: RANK-CHOOSER (KIM-ENGLISH)
Chooser Display Commands
SPEECHACT
NOSPEECHACT
L: Chooser Tracing and Display Options; R: Menu.

```
(EXIST-SPEECH-ACT-Q ONUS )
(IDENTIFY SPEECHACT (SPEECH-ACT-ID ONUS ))
(CHOOSE CLAUSES )
(COPYHUB ONUS THING )
(CHOOSE NOMINAL-GROUPS )
```

This is called a ***chooser*** and is responsible for choosing automatically the features of its associated system. All systems can have their own choosers associated with them. The way to look at the chooser is as a small 'decision expert' that knows how and why particular grammatical features should be chosen. They do this by looking at the semantics given as input.

The current chooser has to choose between the grammatical features clauses, nominal groups and clauses. To do this it first asks whether we want to make some speech act or not (exist-speech-act-q), if we do (and we do if we want to make an assertion such as 'Kim devours a cake'), then it picks the 'speech act' branch of the decision tree. At the end of that path is the instruction "choose clauses" which is what the grammar then does.