The format of these entries is:

```
\text{template}(\text{name, layout, flags})
```

In simplex, you have the following entries:

**Proper Name Entries**

In running glue & XTE...

names and values are_ability to
The lexicon in semantic contains some illustrations entries for proper

Some documentation

A preliminary semantic lexicon

Copy the following files from the class directory to your own glue...
controlling errors (see class-doc lex)

The lexcon compiler tries to produce some helpful warnings if the file

\[ \text{\texttt{-c lexcon}} \]

At the prolog prompt, enter

mands

Or select "Break semantics" from the Structure window com-

Either hit Ctrl-C in the prolog window

To load these new entities, first get to a prolog prompt

Add new entities to your semantic file

---

Modifying and Loading Your Lexcon

---

Note Prolog Syntax

---
Type raising is often regarded as deeply mysterious.

Suppose we have the instantiated premises:

**Scope Ambiguity:** Everyone saw someone

\[ f : (\langle X \rangle \text{name}\langle X \rangle \text{person}) \]

\[ f \leftarrow (f \leftarrow b) : (\langle X \rangle d \langle X \rangle \text{person} \text{'} \langle X \rangle \text{person}) \]

\[ f \leftarrow b : \text{name} \]

\[ \text{some} \]

\[ \text{quantified NPs as functions looking for predicates as argument} \]

\[ (\langle X \rangle \text{name}\langle X \rangle \text{person}) \equiv \text{some} \]

\[ \text{but quantified NPs don't act as simple arguments} \]

\[ f : \text{name} \]

\[ b : \text{char} \]

\[ f \leftarrow b : \text{name} \]

\[ \text{quantified NPs as modifers} \]

\[ \text{view quantified NPs as modifiers} \]

\[ \text{next week} \]

\[ \text{quantifier scope ambiguity} \]

\[ \text{quantified NPs \& proper names} \]

\[ \text{quantified NPs \& proper names} \]

\[ \text{today} \]
The nuclear scope of the quantifier can be any (well-)almost any semantically resource serving as

\( H \vdash (H \vdash \phi) \vdash X \phi \)

over scope resources e.g. To allow quantifiers to front up, give formula universally quantifies

\( (((\lambda) x) y) \text{woman}, (((\lambda) x) y) \text{woman} \)

\( (((\lambda) x) y) \text{woman}, (((\lambda) x) y) \text{woman} \)

\( (((\lambda) x) y) \text{woman}, (((\lambda) x) y) \text{woman} \)

\( (((\lambda) x) y) \text{woman}, (((\lambda) x) y) \text{woman} \)

Quantified NPs “a woman” can scope over main or subordinate clause.

Every man believes a woman yamlled.

One Derivation, with Meaning Terms

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : \langle A', X \rangle \text{see}, XY \]

\[ f : \langle A', X \rangle \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]

\[ f : (\langle A', X \rangle)^{*} \text{see}, XY \]
A Non-Derivation

If the "someone" quantifier can float up, why can't the "everyone" quantifier float down?

Why prevents us from deriving the ill-formed meaning

\[ \text{believe}(x, \text{some}(y, \text{woman}(y)), \text{every}(x, \text{man}(x), \text{yawn}(y))) \]

The quantifier meaning constructor

\[ \text{everyone} : \forall H. (g \leadsto H) \leadsto H \]

says: freely choose a scope \( H \), provided that \( H \) depends on \( g \).

The subordinate clause cannot be a scope for the main clause subject, since its meaning does not depend on the main clause subject.

To float the main clause subject down, we would have to be able to take a derivation step

\[
\begin{align*}
\text{[k]} & & k \leadsto h \\
\forall H. (g \leadsto H) & \leadsto H \\
\hline
h & \quad (g \leadsto h) \leadsto h \\
\hline
h & \quad (g \leadsto h) \leadsto h
\end{align*}
\]

There is no way to do this

---

e and t Resources

To get scoping right, we need to restrict possible scopes to semantic resources that correspond to propositional meanings.

We thus sort all semantic resources into two types:

- \( t \)-resources (truth value denoting)
- \( e \)-resources (entity denoting)

The quantified variable, \( H \) in \( \text{everyone} : \forall H. (g \leadsto H) \leadsto H \) needs to be restricted to range over only \( t \)-resources.

Note that \( g \) is an \( e \)-resource.

From the glue formula \( (g e \leadsto H_t) \leadsto H_t \)
we can read of the type of the meaning expression: \( ((e, t), t) \).

Experiment: see what happens of you allow the variable \( H \) to range over type \( e \) resources.