Macros for Typesetting Semantics (Linguistics) – Version 0.3

Adam Liter

Last updated: July 9, 2013

Abstract

This document is documentation (pun intended) for LaTeX macros that have been written in order to make the typesetting of semantics both easier and more readable. Section 1 includes explanations of how the macros work, using examples for illustration purposes. Section 2 provides definitions of the macros themselves. Section 3 includes some kudos to Alan Munn for helping with these macros. And, finally, section 4 documents changes that have been made to these macros over time.

0 Introductory Remarks

These are macros that I intend to develop over the course of my linguistics (semantics) career in response to any issues that I might encounter. Of course, such development would be greatly aided by any suggestions, comments, and improvements that **you** suggest.

Thus, the source code for these macros is documented below, and the source code can also be directly downloaded here. So, don't hesitate to make suggestions or improvements! I can be reached via email with any such suggestions or improvements.

1 Examples & Explanations

1.1 The Interpretation Function

There are two relevant macros for typesetting text inside of the semantic interpretation function ('⟦ ⟧'). They are `\interp` and `\wraptext`. `\interp` takes an optional argument, which can be used to relativize the interpretation function to an assignment.[1] For example, `\interp{fast}` produces the expression in (1a), while `\interp[g]{fast}` produces the expression in (1b).

[1] Assignment functions are used primarily to account for indexicals like pronouns (*e.g.*, Heim & Kratzer, 1998:90–95). When doing intensional semantics, we might also want to relativize the interpretation function to a world of evaluation. The optional argument allows us to do so.

Note that the `\interp` command is written so that the optional argument is in a math-mode environment. Thus, for example, if we want to relativize the interpretation function to a world of interpretation, say w_7, we write `\interp[w_{7}]{fast}`—which produces the expression in (1c)—and **not** `\interp[w$_{7}$]{fast}`, which will cause the compilation to crash.

(1) a. $[\![\text{fast}]\!]$
b. $[\![\text{fast}]\!]^{g}$
c. $[\![\text{fast}]\!]^{w_7}$

The second macro, `\wraptext`, is used for handling denotations that we wish to specify that are lengthy. It exploits the fact that the two commands from the stmaryrd package—`\llbracket` and `\rrbracket`—are written so as to be delimiters. `\wraptext` goes inside the `\interp` command. It is written so that it's default optional argument value is '1in'.

Thus, `\interp{\wraptext{example of . . .}}` will produce the expression in (2a).

Similarly, if we write, `\interp{\wraptext[2in]{example of . . .}}`, that will produce the expression in (2b).

The definition of the `\wraptext` command uses a varwidth environment, which is made available by the varwidth package. This environment sets the width of its contents to their natural width; thus, we need never worry about setting the width of the `\wraptext` command such that the right denotation bracket is offset because of how the text wraps, as is the case in (2c).

(2) a. $[\![\text{example of really long text inside an interpretation function which will continue as long as we like}]\!]$
b. $[\![\text{example of really long text inside an interpretation function which will continue as long as we like}]\!]$
c. $[\![\text{example of really long text inside an interpretation function which will continue as long as we like}]\!]$

1.2 Denotations & Arguments

By making minor adjustments to the `\interp` command, we can define two new commands, `\den` and `\argum`, which do the exact same thing for denotations and arguments that `\interp` does for text inside of the interpretation func-

tion.[2] So, `\den{x is fast in w}` produces the expression in (3a). Similarly, `\argum{Fred}` produces the expression in (3b). Both of these commands also work with the `\wraptext` command just like `\interp` does (*cf.* (3c) & (3d)).

(3) a. [x is fast in w]

b. (Fred)

c. $\left[\begin{array}{l}\text{example of a}\\ \text{really long de-}\\ \text{notation which}\\ \text{will continue as}\\ \text{long as we like}\end{array}\right]$

d. $\left(\begin{array}{l}\text{example of a}\\ \text{really long argu-}\\ \text{ment which will}\\ \text{continue as long}\\ \text{as we like}\end{array}\right)$

1.3 λ-expressions

For lambda expressions, we can use one of two commands, either `\lam` or `\lamexp`. `\lamexp` allows one to write a complete and explicit λ-expression as in (4a). This command necessarily takes two arguments. Thus, to produce (4a), we write `\lamexp{p}{<s,t>}`.

(4) a. $\lambda p \in D_{<s,t>}$.

b. λp.

c. $\lambda p_{<s,t>}$.

Often, though, we are satisfied with either of the abbreviations seen in (4b) or (4c). We can produce either of these with the `\lam` command. This macro takes an optional argument which can be used to specify the domain of the variable in the λ-expression. So, `\lam{p}` produces the expression in (4b), and `\lam[<s,t>]{p}` produces the expression in (4c).

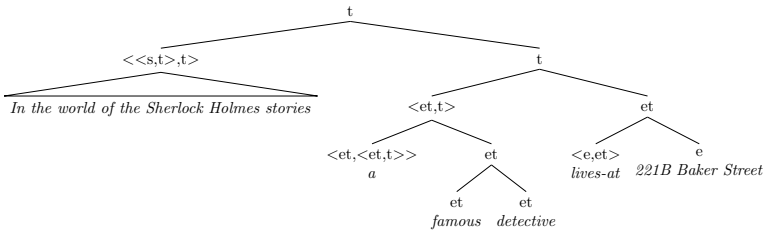
Similar to the `\interp` command, note that both the `\lam` command and the `\lamexp` are written so that all of their arguments are in a math-mode environment.

1.4 Putting It All Together

(6) is an example—glossing over many details—of how these commands can be used for readability's sake for computing the semantic derivation of the structure in (5), whereas (7) is an example of what (6) might look like without these macros.

[2] Note that these new commands no longer take an optional argument, as one—as far as I am aware—never need specify another argument of a denotation or an argument like one often must specify an assignment function or world of evaluation for the interpretation function.

(5)



(6) a. $[\![\text{a famous detective lives at 221B Baker Street}]\!]^{w_7,g}$ = 1 iff ∃x : x is famous in w_7 and x is a detective in w_7 and x lives at 221B Baker Street in w_7

b. $[\![\text{In the world of the Sherlock Holmes stories, a famous detective lives at 221B Baker Street}]\!]^{w_7,g} = [\![\text{In the world of the Sherlock Holmes stories}]\!]^{w_7,g}\left(\lambda w.\ [\![\text{a famous detective lives at 221B Baker Street}]\!]^{w,g}\right)$

$= [\![\forall w' \text{ compatible with the Sherlock Holmes stories in w, } p(w') = 1]\!]\left(\lambda w.\ [\![\text{a famous detective lives at 221B Baker Street}]\!]^{w,g}\right)$

= 1 iff, ∀w′ compatible with the Sherlock Holmes stories in w, ∃x : x is famous in w′ and x is a detective in w′ and x lives at 221B Baker Street in w′

(7) a. $[\![\text{a famous detective lives at 221B Baker Street}]\!]^{w_7,g}$ = 1 iff ∃x : x is famous in w_7 and x is a detective in w_7 and x lives at 221B Baker Street in w_7

b. $[\![\text{In the world of the Sherlock Holmes stories, a famous detective lives at 221B Baker Street}]\!]^{w_7,g} = [\![\text{In the world of the Sherlock Holmes stories}]\!]^{w_7,g}(\lambda w.[\![\text{a famous detective lives at 221B Baker Street}]\!]^{w,g})$

$= [\forall w' \text{ compatible with the Sherlock Holmes stories in w, } p(w') = 1](\lambda w.[\![\text{a famous detective lives at 221B Baker Street}]\!]^{w,g})$

= 1 iff, ∀w′ compatible with the Sherlock Holmes stories in w, ∃x : x is famous in w′ and x is a detective in w′ and x lives at 221B Baker Street in w′

2 Writing the Macros

2.1 Writing the Macros Yourself

If you would like, you can follow the instructions below to write the macros yourself.

2.1.1 Required Packages

In order to write these macros, you will need to load the following packages in your .tex document.

- stmaryrd
- amsmath
- ragged2e
- varwidth

2.1.2 The Macros

In order to use the macros, define the following commands in the preamble of your .tex document.

- `\interp` is defined as follows:

```
\newcommand{\interp}[2][]{
\(
\left\llbracket\,\text{#2}\,\right\rrbracket^{#1}
\)
}
```

- `\den` is defined as follows:

```
\newcommand{\den}[1]{
\(
\left[\,\text{#1}\,\right]
\)
}
```

- `\argum` is defined as follows:

```
\newcommand{\argum}[1]{
\(
\left(\,\text{#1}\,\right)
\)
}
```

- `\wraptext` is defined as follows:

```
\newcommand{\wraptext}[2][1in]{
\begin{varwidth}{#1}{\RaggedRight#2}\end{varwidth}
}
```

- `\lam` is defined as follows:

```
\newcommand{\lam}[2][]{$\lambda {#2}_{#1}$.}
```

- `\lamexp` is defined as follows:

```
\newcommand{\lamexp}[2]{$\lambda {#1} \in D_{#2}$.}
```

2.2 Downloading the Macros

If you would rather not write the macros yourselves, you can simply download the source code directly for these macros here, as noted above, to be placed in your preamble.

Or, if you use TeXShop, this .tex file can then be placed in the Templates directory (`~\Library\TeXShop\Templates`), which will allow you to access the macros from the Templates menu in TeXShop.

3 Acknowledgements

The two macros, `\interp` & `\wraptext`, were written by Alan Munn in response to a question that I posted on TeX.SX. I adapted the `\den` and `\argum` commands from his macro for `\interp`, and I wrote the two macros for λ-expressions myself.

4 Change Log

Version 0.3 (2013.07.09) Thick spaces in the `\interp`, `\den`, and `\argum` were changed to thin spaces. `\lamexp` was added for complete and explicit λ-expressions.

Version 0.2 (2013.07.01) The macro `\den` was renamed `\interp` because, more accurately, the command was for typesetting text inside of the semantic interpretation function. The macros `\den` and `\argum` were added in order to be used in conjunction with the `\wraptext` command for typesetting complete semantic derivations, including denotations and arguments, respectively.

Version 0.1 (2013.06.29) The macros `\den` and `\lam` were written for the interpretation function and λ-expressions, respectively.

References

Heim, I. & Kratzer, A. (1998). *Semantics in Generative Grammar*. Malden, MA: Blackwell Publishers Inc.