Semantics of FOPL and the lambda calculus

Simon Charlow (simon.charlow@rutgers.edu)

October 14, 2015

1 FOPL syntax refresher

Vocabulary:

- **Terms**: **individual constants** like *bob* and *sam*, plus an infinite stock of **variables**: x, y, z, ...
- A collection of n-ary **predicates**: *runs*, *likes*, *gave*, ...
- The propositional logic **connectives**: $\neg$, $\wedge$, $\vee$, $\Rightarrow$. Plus punctuation: ., (, and).
- An **existential quantifier** $\exists$, and a **universal quantifier** $\forall$.

Complex formulas. The WFF of propositional logic is the smallest set such that:

- Predicates applied to the appropriate number of terms are in WFF. E.g., $\textit{left}\, x$, $\textit{saw}(\textit{bob}, y)$, ... These are the atomic formulas.
- If φ is in WFF, then $\neg\varphi$ is in WFF.
- If φ and ψ are in WFF, then $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, and $(\varphi \Rightarrow \psi)$ are all in WFF.
- If φ is in WFF, then $(\exists v.\, \varphi)$ and $(\forall v.\, \varphi)$ are in WFF, for any variable v.

As before, we adopt the convention of omitting outermost parentheses. I also like to omit parentheses when doing so doesn't create ambiguity, but whether you do so is up to you.

2 FOPL semantics

As in propositional logic, we need a way to assign values to variables. This time, however, the variables denote things of type e:

- $[\![v]\!]^g = g\, v$

Proper names and predicates will be valued by $[\![\cdot]\!]^g$ in the way we've done in class (though notice that the meaning of relations is given as the characteristic function of a set of ordered pairs):

- $[\![\textit{bob}]\!]^g = \text{B}$
- $[\![\textit{left}]\!]^g = \lambda x.\, \text{LEFT}\, x$
- $[\![\textit{likes}]\!]^g = \lambda (x, y).\, \text{LIKES}\, (x, y)$
- ...

Predications are evaluated by finding the values of the predicate and its arguments, and then applying the former to the latter:

- $[\![\mathtt{P}\,v]\!]^g = [\![\mathtt{P}]\!]^g\,[\![v]\!]^g$
- $[\![\mathtt{R}\,(v, u)]\!]^g = [\![\mathtt{R}]\!]^g\,([\![v]\!]^g, [\![u]\!]^g)$
- …

The meanings of the connectives are unchanged from propositional logic:

- $[\![\neg\varphi]\!]^g = 1 - [\![\varphi]\!]^g$
- $[\![\varphi \wedge \psi]\!]^g = \mathtt{Min}\,\{[\![\varphi]\!]^g, [\![\psi]\!]^g\}$
- $[\![\varphi \vee \psi]\!]^g = \mathtt{Max}\,\{[\![\varphi]\!]^g, [\![\psi]\!]^g\}$
- $[\![\varphi \Rightarrow \psi]\!]^g = \mathtt{Max}\,\{[\![\neg\varphi]\!]^g, [\![\psi]\!]^g\}$

Finally, the meanings of the quantifiers rely on *assignment modification*:

- $[\![\exists v.\,\varphi]\!]^g = \mathtt{Max}\,\{[\![\varphi]\!]^{g[v \to a]} : a \in e\}$
- $[\![\forall v.\,\varphi]\!]^g = \mathtt{Min}\,\{[\![\varphi]\!]^{g[v \to a]} : a \in e\}$

Relies on **minimal assignment modification**:

- $g[v \to a]$ is the assignment h such that $h\,v = a$, and for any $u \neq v$, $h\,u = g\,u$.

Mnemonically, you can think of $g[v \to a]$ as "the assignment mapping v to a, but otherwise just like g".

Essentially, existential quantification is like a huge disjunction (if a **or** b **or** c **or** … makes φ true, then $\exists v.\,\varphi$ is true), and universal quantification is like a huge conjunction (if a **and** b **and** c **and** … make φ true, then $\forall v.\,\varphi$ is true).

3 Examples

Unrestricted quantification. There is a man:

$$\begin{aligned}
[\![\exists x.\,\mathit{man}\,x]\!]^g &= \mathtt{Max}\,\{[\![\mathit{man}\,x]\!]^{g[x \to a]} : a \in e\} \\
&= \mathtt{Max}\,\{[\![\mathit{man}]\!]^{g[x \to a]}\,[\![x]\!]^{g[x \to a]} : a \in e\} \\
&= \mathtt{Max}\,\{\textsf{MAN}\,[\![x]\!]^{g[x \to a]} : a \in e\} \\
&= \mathtt{Max}\,\{\textsf{MAN}\,a : a \in e\}
\end{aligned}$$

The set $\{\textsf{MAN}\,y : y \in e\}$ will contain a 1 iff we can find at least one man in the domain of indivudals. Otherwise, it will only contain 0. Therefore, the truth condition is that at least one individual is a man. Here, for completeness, are the three possible outputs:

$$\begin{aligned}
\{0\} &\quad \text{(no men exist; } \mathtt{Max} \text{ returns } 0) \\
\{1\} &\quad \text{(only men exist; } \mathtt{Max} \text{ returns } 1) \\
\{0, 1\} &\quad \text{(both men and non-men exist; } \mathtt{Max} \text{ returns } 1)
\end{aligned}$$

Adding a negation would, of course, resulting in flipping whatever value is returned.

And similarly for the universal. Everything is a man:

$$\begin{aligned}
[\![\forall x.\, \mathit{man}\, x]\!]^g &= \mathtt{Min}\, \{[\![\mathit{man}\, x]\!]^{g[x \to a]} : a \in \mathtt{e}\} \\
&= \mathtt{Min}\, \{[\![\mathit{man}]\!]^{g[x \to a]}\, [\![x]\!]^{g[x \to a]} : a \in \mathtt{e}\} \\
&= \mathtt{Min}\, \{\textsf{MAN}\, [\![x]\!]^{g[x \to a]} : a \in \mathtt{e}\} \\
&= \mathtt{Min}\, \{\textsf{MAN}\, a : a \in \mathtt{e}\}
\end{aligned}$$

This, this set will contain a 0 iff we can find at least one non-man in the domain of indivudals. Otherwise, it will only contain 1. Therefore, the truth condition is that every individual is a man. Here are the three possible outputs:

$$\begin{aligned}
\{0\} &\quad \text{(no men exist; } \mathtt{Min} \text{ returns } 0) \\
\{1\} &\quad \text{(only men exist; } \mathtt{Min} \text{ returns } 1) \\
\{0, 1\} &\quad \text{(both men and non-men exist; } \mathtt{Min} \text{ returns } 0)
\end{aligned}$$

Restricted quantification. Bob met a linguist:

$$\begin{aligned}
[\![\exists z.\, \mathit{linguist}\, z \wedge \mathit{met}(\mathit{bob}, z)]\!]^g &= \mathtt{Max}\, \{[\![\mathit{linguist}\, z \wedge \mathit{met}(\mathit{bob}, z)]\!]^{g[z \to a]} : a \in \mathtt{e}\} \\
&= \mathtt{Max}\, \{\mathtt{Min}\, \{[\![\mathit{linguist}\, z]\!]^{g[z \to a]}, [\![\mathit{met}(\mathit{bob}, z)]\!]^{g[z \to a]}\} : a \in \mathtt{e}\} \\
&= \mathtt{Max}\, \{\mathtt{Min}\, \{\textsf{LINGUIST}\, a, \textsf{MET}(\textsf{BOB}, a)\} : a \in \mathtt{e}\}
\end{aligned}$$

Suppose there's a linguist who Bob met. Call her ℓ. Then $\mathtt{Min}\, \{\textsf{LINGUIST}\, \ell, \textsf{MET}(\textsf{BOB}, \ell)\} = \mathtt{Min}\, \{1\} = 1$, and $\mathtt{Max}$ will necessarily return 1. If we can find no such individual, $\mathtt{Min}\, \{\textsf{LINGUIST}\, x, \textsf{MET}(\textsf{BOB}, x)\}$ will always return 0, and the so $\mathtt{Max}$ will necessarily return 0.

Multiple quantification. Something is near something:

$$\begin{aligned}
[\![\exists x.\, \exists y.\, \mathit{near}(x, y)]\!]^g &= \mathtt{Max}\, \{[\![\exists y.\, \mathit{near}(x, y)]\!]^{g[x \to a]} : a \in \mathtt{e}\} \\
&= \mathtt{Max}\, \left\{\mathtt{Max}\, \{[\![\mathit{near}(x, y)]\!]^{g[x \to a][y \to b]} : b \in \mathtt{e}\} : a \in \mathtt{e}\right\} \\
&= \mathtt{Max}\, \left\{\mathtt{Max}\, \{\textsf{NEAR}(a, b) : b \in \mathtt{e}\} : a \in \mathtt{e}\right\}
\end{aligned}$$

Suppose we can find something near something else. Call those things, respectively, ℓ_1 and ℓ_2. Then there will be at least one true member of $\mathtt{Max}\, \{\textsf{NEAR}(a, b)\}$, namely $\textsf{NEAR}(\ell_1, \ell_2)$, and there will thus be a true member of $\mathtt{Max}\, \left\{\mathtt{Max}\, \{\textsf{NEAR}(a, b) : b \in \mathtt{e}\} : a \in \mathtt{e}\right\}$, namely $\mathtt{Max}\, \{\textsf{NEAR}(\ell_1, \ell_2)\}$. In which case the value ultimately returned is 1. If we can find no such ℓ_1 and ℓ_2, the inner and outer $\mathtt{Max}$'s will both return 0.

4 Semantics of the lambda calculus

By cases:

- $[\![v]\!]^g = g\, v$ if v is a variable
- $[\![M\ N]\!]^g = [\![M]\!]^g\, [\![N]\!]^g$
- $[\![\lambda v.\, \varphi]\!]^g$ = the function f such that for any a, $f\, a = [\![\varphi]\!]^{g[v \to a]}$

Proper names and predicates will be valued by $[\![\cdot]\!]^g$ in the way we've done in class (same as FOPL):

- $[\![\mathit{bob}]\!]^g = \textsf{B}$

- $[\![\textit{left}]\!]^g = \lambda x.\, \textsf{LEFT}\, x$, i.e. (by η-equivalence), $\textsf{LEFT}$
- $[\![\textit{likes}]\!]^g = \lambda(x, y).\, \textsf{LIKES}\,(y, x)$
- …

A simple case. x left:

$$\begin{aligned}[\![\textit{left}\, x]\!]^g &= [\![\textit{left}]\!]^g\, [\![x]\!]^g \\ &= \textsf{LEFT}\,(g\, x)\end{aligned}$$

An abstraction. The function mapping x to 1 iff x left:

$$\begin{aligned}[\![\lambda x.\, \textit{left}\, x]\!]^g &= \text{the } f \text{ such that } f\, a = [\![\textit{left}\, x]\!]^{g[x\to a]} \\ &= \text{the } f \text{ such that } f\, a = [\![\textit{left}]\!]^{g[x\to a]}\, [\![x]\!]^{g[x\to a]} \\ &= \text{the } f \text{ such that } f\, a = \textsf{LEFT}\, [\![x]\!]^{g[x\to a]} \\ &= \text{the } f \text{ such that } f\, a = \textsf{LEFT}\, a\end{aligned}$$

Combining lambdas with FOPL. The function mapping x to 1 iff there's something x is near:

$$\begin{aligned}[\![\lambda x.\, \exists y.\, \textsf{NEAR}(x, y)]\!]^g &= \text{the } f \text{ s.t. } f\, a = [\![\exists y.\, \textit{near}(x, y)]\!]^{g[x\to a]} \\ &= \text{the } f \text{ s.t. } f\, a = \texttt{Max}\, \{[\![\textit{near}(x, y)]\!]^{g[x\to a][y\to b]} : b \in e\} \\ &= \text{the } f \text{ s.t. } f\, a = \texttt{Max}\, \{\textsf{NEAR}(a, b) : b \in e\}\end{aligned}$$

Given an a, this function returns 1 iff there's some b that a is near.

To emphasize, we will be using the λ-calculus and FOPL as part of our semantic **metalanguage**. In other words, we won't bother with interpreting them: we'll simply use them to specify the model-theoretic denotations of expressions. The reason to bother with understanding their semantics is that they turn out to be quite useful for interpreting *natural* language!

Alpha-equivalence.

$$\begin{aligned}[\![\lambda x.\, \textit{left}\, x]\!]^g &= \text{the } f \text{ s.t. } f\, a = [\![\textit{left}\, x]\!]^{g[x\to a]} \\ &= \text{the } f \text{ s.t. } f\, a = \textsf{LEFT}\, a \\ [\![\lambda y.\, \textit{left}\, y]\!]^g &= \text{the } f \text{ s.t. } f\, a = [\![\textit{left}\, y]\!]^{g[y\to a]} \\ &= \text{the } f \text{ s.t. } f\, a = \textsf{LEFT}\, a\end{aligned}$$

Multiple modification.

$$\begin{aligned}[\![\lambda x.\, \lambda x.\, \textit{likes}(x, x)]\!]^g &= \text{the } f \text{ s.t. } f\, a = [\![\lambda x.\, \textit{likes}(x, x)]\!]^{g[x\to a]} \\ &= \text{the } f \text{ s.t. } f\, a = \text{the } g \text{ s.t. } g\, b = [\![\textit{likes}(x, x)]\!]^{g[x\to a][x\to b]} \\ &= \text{the } f \text{ s.t. } f\, a = \text{the } g \text{ s.t. } g\, b = \textsf{LIKES}(b, b)\end{aligned}$$

In other words, $f\, a\, b = \textsf{LIKES}(b, b)$; the lower λx takes precedence, and the first argument is thrown out. The reason this happens is that g is modified twice. The first time turns it into something mapping x to a, and the second time turns it into something mapping x to b. The first modification is **lost**.

This corresponds **precisely** to the syntactic characterization of β-reduction we gave a couple weeks back: '$(\lambda x.\, \varphi)\, a$' β-reduces to $\varphi[x \to a]$, i.e. φ with all the **free** occurrences of x replaced with a.