Syntax of the λ-Calculus

Cameron Moy

November 2, 2018

Grammar of Explicit Parentheses

Languages are often defined using a context-free grammar, or a variation thereof, and the λ-calculus is no exception.

$$T \to (\lambda X.T) \mid (T\ T) \mid X$$
$$X \to a \mid b \mid c \mid d \mid ...$$

A λ term is any string in this language. From the grammar we can see that any λ term can be one of three forms:

- an abstraction $(\lambda X.T)$,
- an application $(T\ T)$,
- or a variable X.

Example. Is $((\lambda y.y)\ z)$ a well-formed λ term?

Solution. Yes, we can give it's left-most derivation.

$$\begin{aligned}
\underline{T} &\to (\underline{T}\ T) \\
&\to ((\lambda \underline{X}.T)\ T) \\
&\to ((\lambda y.\underline{T})\ T) \\
&\to ((\lambda y.\underline{X})\ T) \\
&\to ((\lambda y.y)\ \underline{T}) \\
&\to ((\lambda y.y)\ \underline{X}) \\
&\to ((\lambda y.y)\ z)
\end{aligned}$$

❑

Example. Is $(\lambda y.y)\ z$ a well-formed λ term?

Solution. No. We don't have parentheses surrounding the application so it is not a well-formed λ term according to the above grammar. ❑

Grammar with Omitted Parentheses

This is silly! Our second example clearly intends the same thing as the first one. We shouldn't have to write all those extra parentheses. Doing so just makes our λ terms messy. Let's rewrite our grammar to omit these parentheses by default.

$$\begin{aligned} T &\to \lambda X.T \mid T\ T \mid X \mid (T) \\ X &\to a \mid b \mid c \mid d \mid ... \end{aligned}$$

So now our second example $(\lambda y.y)\ z$ is a valid λ term. Unfortunately, defining our language this way has introduced ambiguity. Removing parentheses in both the abstraction and application cases cause two different kinds of ambiguity.

Application Ambiguity

Example. Show the λ term $a\ b\ c$ is ambiguous.

Solution. We construct two distinct left-most derivation of $a\ b\ c$. The first corresponds to left-associative application.

$$\begin{aligned} \underline{T} &\to \underline{T}\ T \\ &\to \underline{X}\ T \\ &\to a\ \underline{T} \\ &\to a\ \underline{T}\ T \\ &\to a\ \underline{X}\ T \\ &\to a\ b\ \underline{T} \\ &\to a\ b\ \underline{X} \\ &\to a\ b\ c \end{aligned}$$

Alternatively, we can construct a left-most derivation that corresponds to right-associative application.

$$\begin{aligned}
\underline{T} &\rightarrow \underline{T}\ T \\
&\rightarrow \underline{T}\ T\ T \\
&\rightarrow \underline{X}\ T\ T \\
&\rightarrow a\ \underline{T}\ T \\
&\rightarrow a\ \underline{X}\ T \\
&\rightarrow a\ b\ \underline{T} \\
&\rightarrow a\ b\ \underline{X} \\
&\rightarrow a\ b\ c
\end{aligned}$$

❏

This problem can be solved in the same way we handle this issue for arithmetic. What does $10 - 5 - 4$ mean? We have a convention that subtraction left-associates so we know that this is really $((10 - 5) - 4) = 1$ and not $(10 - (5 - 4)) = 9$.

So for the λ-calculus we set up the same convention so that this isn't a problem.

Convention. Application left-associates in the λ-calculus.

Exercise. Why do we use this convention as opposed to right-associating application? (Hint: Would currying work if our convention was right-association?)

Abstraction Ambiguity

Example. Show the λ term $\lambda y.y\ z$ is ambiguous.

Solution. We construct two distinct left-most derivations of $\lambda y.y\ z$. One interpretation of this term is that of a λ abstraction whose body is the application $y\ z$.

$$\begin{aligned} \underline{T} &\to \lambda \underline{X}.T \\ &\to \lambda y.\underline{T} \\ &\to \lambda y.\underline{T}\ T \\ &\to \lambda y.\underline{X}\ T \\ &\to \lambda y.y\ \underline{T} \\ &\to \lambda y.y\ \underline{X} \\ &\to \lambda y.y\ z \end{aligned}$$

The other interpretation is that this is an application, applying the identity function $\lambda y.y$ to the argument z.

$$\begin{aligned} \underline{T} &\to \underline{T}\ T \\ &\to \lambda \underline{X}.T\ T \\ &\to \lambda y.\underline{T}\ T \\ &\to \lambda y.\underline{X}\ T \\ &\to \lambda y.y\ \underline{T} \\ &\to \lambda y.y\ \underline{X} \\ &\to \lambda y.y\ z \end{aligned}$$

❏

Once again, to resolve this ambiguity we set up a convention. We will always assume that the body of a λ abstraction extends as far as possible. Or another way to put it, application has higher precedence than abstraction. In the above example, that means taking the first interpretation.

Convention. The body of an abstraction extends as far as possible in the λ-calculus.

What "as far as possible" means is slightly imprecise at the moment. Let's take a closer look at that.

Example. How should $(\lambda x.\lambda y.(x\ y))\ z$ be interpreted?

Solution. Here is should be clear that "as far as possible" doesn't mean until the end of the string. That would mean the body of our λx would be $\lambda y.(x\ y))\ z$ and that the body of our λy would be $(x\ y))\ z$. Neither make any sense because of that stray parenthesis.

Here is a trick for determining the extent of the body. Consider everything after the period. The body goes until either

- the first unbalanced right parenthesis, or
- the end of the string.

That means for λx we consider $\lambda y.(x\ y)\underline{)}\ z$. Underlined is the first unbalanced right parenthesis, so the body of λx extends up until that point. The same goes for our λy. We consider $(x\ y)\underline{)}\ z$. So, the body of λy will be $(x\ y)$. ❏

Making Parentheses Explicit

So, let's take strings in our parentheses-anemic ambiguous language and use our conventions to find the equivalent string in the parentheses-profuse unambiguous language.[1]

Example. Make parentheses explicit in $(\lambda x.x\ z)\ \lambda y.w\ \lambda w.w\ y\ z\ x$.

Solution. Scanning left-to-right I see λx so I need to figure out how far its body extends. It extends until the first unbalanced right-parenthesis. That means its body is $x\ z$. Next up I see λy. Afterwards I see no unbalanced parenthesis so this one goes until the end of the string. We can make this body

[1] Equivalent here means they correspond to the same abstract syntax tree.

explicit by surrounding it with parentheses. We end up with two abstractions next two each other so that's an application. I surround that with parentheses too. I'm left with

$$((\lambda x.x\ z)\ (\lambda y.w\ \lambda w.w\ y\ z\ x))$$

Next we will dig into the bodies. First comes the λx body. This is simply an application so we surround that with parentheses.

$$((\lambda x.(x\ z))\ (\lambda y.w\ \lambda w.w\ y\ z\ x))$$

Here comes the λy body. I see a λw so I need to figure out how far that extends. It goes until that first right parenthesis. I have the variable w next to the λw abstraction so that becomes an application.

$$((\lambda x.(x\ z))\ (\lambda y.(w\ (\lambda w.w\ y\ z\ x))))$$

Finally, we consider the inside of λw. This is a bunch of applications so they left associate.

$$((\lambda x.(x\ z))\ (\lambda y.(w\ (\lambda w.(((w\ y)\ z)\ x)))))$$

❏