NOTES 12: μOCAML COMPILER

Cameron Moy

Tuesday July 10th

1 λ syntax and semantics

Here is a grammar for the λ calculus.

$$T \to T\ T \ \mid\ \lambda X.T \ \mid\ X$$

$$X \to a \ \mid\ b \mid\ \ldots\ \mid\ z\ \mid aa\ \mid\ \ldots$$

And here are the semantics.

$$(\alpha)\frac{}{(\lambda x.t) \Rightarrow_r (\lambda y.t[x \mapsto y])}$$

$$(\beta)\frac{}{(\lambda x.t_1)\ t_2 \Rightarrow_r t_1[x \mapsto t_2])}$$

$$(3)\frac{t_1 \Rightarrow_r t_1'}{t_1\ t_2 \Rightarrow_r t_1'\ t_2}\ (4)\frac{t_2 \Rightarrow_r t_2'}{t_1\ t_2 \Rightarrow_r t_1\ t_2'}\ (5)\frac{t \Rightarrow_r t'}{(\lambda x.t) \Rightarrow_r (\lambda x.t')}$$

The semantics define a reduction relation, not a function, $\Rightarrow_r : T \to T$. This style of semantics is called small-step. The fact $\Rightarrow_r$ is a relation means our semantics are now non-determinstic.

Rule (α) is α-conversion. Rule (β) is β-reduction. Rules (3) through (5) don't actually do anything, but allow us to apply the others in any sub-expression. Rules that do the work, like (α) and (β), are called computation rules. Rules that allow computation to happen in sub-expressions, like (3) through (5), are called congruence rules.

Example. Reduce $(\lambda x.x\ (\lambda y.x\ y))\ y$ to normal form.

Proof. We could do this proof as a series of trees, but this is a lot of writing. Typically we just show the series of $\Rightarrow_r$ directly, underlining the redex.

$$\begin{aligned}(\lambda x.\underline{(\lambda y.x\ y)}\ x)\ y &\Rightarrow_r^{\alpha} \underline{(\lambda x.(\lambda z.x\ z)\ x)\ y} \\ &\Rightarrow_r^{\beta} \underline{(\lambda z.y\ z)\ y} \\ &\Rightarrow_r^{\beta} y\ y\end{aligned}$$

❑

Since $\Rightarrow_r$ is a relation (and hence non-deterministic), there is actually another way to do this proof.

Proof.

$$\begin{aligned}(\lambda x.\underline{(\lambda y.x\ y)\ x})\ y &\Rightarrow_r^{\beta} \underline{(\lambda x.x\ x)\ y} \\ &\Rightarrow_r^{\beta} y\ y\end{aligned}$$

❑

Notice that we get the exact same answer. This is actually a guarantee.

Theorem (Church-Rosser)**.** If $t \Rightarrow_r^* u$ and $t \Rightarrow_r^* v$ (where $\Rightarrow_r^*$ is an arbitrary series of $\Rightarrow_r$ steps) and u and v are normal forms, then $u = v$.

This property is known as confluence and tells us that terms in our calculus have a unique normal form, no matter what reduction strategy you take.

2 Reduction strategies

This is all well-and-good, but non-determinism is bad. If we want to write a λ calculus interpreter we need to define a deterministic strategy. We can describe such a strategy informally.

1. The call-by-value strategy requires that an argument must be evaluated before a β reduction can happen.
2. The call-by-name strategy states that β reductions happen without evaluating the argument at all.

However, we can use our other style of semantics, big-step semantics, to formally describe a deterministic reduction strategy. Here is such a CBV semantics.

$$(1)\,\frac{}{(\texttt{fun}\ x\ \texttt{->}\ t) \Downarrow_v (\texttt{fun}\ x\ \texttt{->}\ t)}$$

$$(2)\,\frac{t_1 \Downarrow_v (\texttt{fun}\ x\ \texttt{->}\ t_{12}) \qquad t_2 \Downarrow_v v_2 \qquad t_{12}[x \mapsto v_2] \Downarrow_v v}{(t_1\ t_2) \Downarrow_v v}$$

3 It was λ the whole time

Pause for a moment. Doesn't this look familiar?

You should have realized by now that last week we implemented a CBV λ calculus interpreter. We just called it "NanOCaml". Our syntax had `fun` instead of λ, an `->` instead of ., and extra parentheses (to make parsing easier), but really that's it!