Constructive Logic (15-317), Fall 2016
Recitation 10: Operational Semantics and Unification

Evan Cavallo (ecavallo@cs.cmu.edu), Oliver Daids (ojd@andrew), Giselle Reis (greis@andrew)

1 Operational Semantics

A strategy to give an operational semantics to a language is to construct a *meta-interpreter* for that language in itself. The idea is that simpler constructs should be used to represent more complex ones. This can be refined further and further until the fragment of the language used in the meta-interpreter is so simple that it can be straightforwardly translated to an abstract machine.

Our first proposal for a prolog meta-interpreter was a very simple one[1]:

```
solve(ttrue).
solve((A, B)) :- solve(A), solve(B).
solve(P) :- clause(P, B), solve(B).
```

Task 1. Extend the meta-interpreter above to a bounded interpreter which fails if no proof of a given depth can be found. In terms of proof trees, the depth is the length of the longest path from the final conclusion to an axiom.

[1] We are using `ttrue` so that it is not confused with the meta-language `true` and to allow this code to run on SWI-Prolog.

2 Unification

Before we talk about unification, we need to understand *substitutions*. A substitution is a function σ from variables to terms. Since this is a function, it makes no sense to talk about substitutions that take the same variable x to two different terms. Remember that variables are also terms, so substitutions that replace variables by variables are completely acceptable.

The notation[2] we will use for substitutions is: $\sigma : (t_1/x_1, ..., t_n/x_n)$, meaning that σ replaces the variable x_i by the term t_i. Another (more widely accepted) convention is to write the application of a substitution σ to a term t as $t\sigma$ instead of the usual function application notation $\sigma(t)$.

Task 2. Apply each of the following substitutions, when well-defined, to the term $x \wedge x(x)$ and write the result:

- $(r/x, z/x)$
- $(r/x, z/r)$
- $(r/x)(z/r)$
- $(z/r)(r/x)$

Task 3. Substitution of terms can also be applied to formulas. Define how substitutions will be applied to formulas of the shape $\neg A$, $A \vee B$, $\forall x.A$.

Now to unifiers. The substitution σ is a unifier for the terms s and t if $s\sigma = t\sigma$. Two terms s and t are called unifiable if a unifier σ exists.

Task 4. Analyse whether each of the pairs of terms below are unifiable and, if they are, write down a unifier for them:

- $x,\ y$
- $x,\ f(x)$
- $g(x),\ f(x)$
- $f(x, g(y)),\ f(h(a), g(z))$
- $f(x, y),\ f(g(y), g(z))$
- $f(x, y),\ f(g(y), g(x))$
- $f(x, y),\ f(g(z), g(x))$

We have discovered last class that prolog has actually a unification bug. There are attempts to work around it, but it still causes some unwanted behaviors.

Task 5. Type the following queries to the prolog interpreter and explain what happens:

- `X = f(X).`
- `f(X) = Y, f(Y) = X, X = Y.`

[2] Amazingly enough, researchers have been unable to agree on a unified notation for substitutions, so they write them as they please. Watch out for that when reading other sources.