Equality in Lazy Computation Systems

Vincent Rahli

CS6110 Lecture 6

Monday Feb 2, 2015

This document collects some notes on Doug Howe's paper "Equality in Lazy Computation Systems":

`http://nuprl.org/documents/Howe/EqualityinLazy_LICS89.pdf`

The goal of that paper is to define a computational equivalence relation $\sim$ for lazy computation systems that is a (observational) congruence.

A computation system in said to be lazy if its values are terms with canonical outermost operators. An operator *op* (or a term with outermost operator *op*) is said to be canonical if there is no rule to reduce it further and non-canonical if there is a rule to reduce it further. For example the λ operator of the λ-calculus is canonical while the application operator is not. Let's now add pairs to the λ-calculus. If our calculus is lazy, $\langle a, b\rangle$ is a value whatever a and b are.

Howe's computational equivalence $\sim$ is used in Nuprl to prove equalities between programs (bisimulations) without having to worry about well-typedness. For example given a term $u = C[t]$ (i.e., the term t occurs in u at a position given by the context C), if $t \sim t'$ then in Nuprl it is possible to rewrite t into t' in u to obtain $C[t']$ without having to prove that C is a well-formed context over some type (t's type(s) for example).

Other systems such as Coq, Agda, or Idris are only closed under computation, meaning that one can only substitute a term for another term if they compute to the same value (computations preserve types).

In Nuprl, we can prove for example that $map(f, map(g, l)) \sim map(f \circ g, l)$ for any f, g, and l. This is useful, among other things, to do program optimization without having to worry about types.

Howe's computation equivalence is defined on closed terms as follows: $a \sim b$ if $a \preceq b \wedge b \preceq a$ (Definition 3). The relation $\preceq$ (called an approximation or simulation relation) is coinductively defined as the largest relation R on closed terms such that $R \subset [R]$ (Definition 2), where $[\cdot]$ is the following closure operator (also defined on closed terms): $a\ [R]\ b$ if whenever a computes to a value of the form $\theta(\overline{t})$ (where θ is the outermost operator of the term and $\overline{t}$ its subterms) then b also computes to a value $\theta(\overline{t'})$ such that $\overline{t}\ R\ \overline{t'}$ (Definition 1).

Because Nuprl supports diverging terms (as opposed to Coq, Agda, and Idris) such as $\text{fix}(\lambda x.x)$ or $(\lambda x.xx)(\lambda x.xx)$ often written as $\bot$, one can prove for example that $\bot \preceq t$, and that for any term t (this trivially follows from $\preceq$'s definition).

It now remains to prove that $\sim$ is a congruence, i.e., that $\tau(\overline{t}) \sim \tau(\overline{t'})$ whenever $\overline{t} \sim \overline{t'}$. Howe first proves that for $\preceq$. Unfortunately, it is not easy to prove directly, e.g., doing a simple codinductive proof. Therefore, Howe defines another relation $\preceq^*$ that is a congruence by definition and also contains $\preceq$ (Definition 4). Howe defines $a \preceq^* b$ by induction on a: if a is a variable then $a \preceq^* b$ if $a \preceq b$, and if a is of the form $\tau(\overline{t})$ then $a \preceq^* b$ if there exists $\overline{t'}$ such that $\overline{t} \preceq^* \overline{t'}$ and $\tau(\overline{t'}) \preceq b$. Note that we are now using $\preceq$ on open terms too. Each relation defined above can be extended to open terms as follows: if a and b are open terms, then we write $a \; R \; b$, where R is a relation on closed terms, if for all substitution sub such that $a[sub]$ and $b[sub]$ are closed we have $a[sub] \; R \; b[sub]$.

By definition we get that $\preceq^*$ is a congruence and that it contains $\preceq$, i.e., $\overline{t} \sim \overline{t'} \Rightarrow \tau(\overline{t}) \preceq^* \tau(\overline{t'})$ and $a \preceq b \Rightarrow a \preceq^* b$. Note that this relation is reflexive and transitive (not symmetric though). However, it is not easy to prove that it is transitive (I personally wasted several days trying to prove it using a "direct" method). By definition, we get that $a \preceq^* b \Rightarrow b \preceq c \Rightarrow a \preceq^* c$, which is enough for what Howe wants to do in the rest of the paper. (One can get that $\preceq^*$ is transitive once we have proved that it is equivalent to $\preceq$ but we don't really care anymore about $\preceq^*$ once we have proved that $\preceq$ is a congruence: $\preceq^*$'s sole purpose is to prove that.)

to prove that $a \preceq b$ is a congruence, it now remains to prove that $a \preceq^* b \Rightarrow a \preceq b$. If we can prove that then because $a \preceq b \Rightarrow a \preceq^* b$, we get that $\preceq^*$ and $\preceq$ are equivalent and so $\preceq$ is a congruence. To prove that $a \preceq^* b \Rightarrow a \preceq b$ and because $\preceq$ is defined as the greatest fixpoint of $[\cdot]$, it suffices to prove that $\preceq^*$ respects computation, i.e., given that $a \preceq^* b$, if a computes to a value of the form $\theta(\overline{t})$ then b also computes to a value $\theta(\overline{t'})$ such that $\overline{t} \preceq^* \overline{t'}$. Howe proves that this is true when a is already a value in Lemma 2. He then defines a condition called *extensionality* (Definition 5) that non-canonical operators of lazy computation systems have to satisfy in order for $a \preceq^* b \Rightarrow a \preceq b$ and therefore in order for $\sim$ to be a congruence. In a nutshell, this extensionality requirement is roughly the condition used in $[\cdot]$'s definition.

It is trivial to prove that the operators of the untyped λ-calculus are all extensional and therefore that $\sim$ is a congruence. It is fairly easy to prove the same thing for Nuprl.

Finally, note that the same machinery can be used for non-deterministic systems too.