Formal Methods
Fixpoints

Nachum Dershowitz

May 2000

We use $\perp$ to denote an *undefined* value. For any set S not containing $\perp$, let $S^\perp = S \cup \{\perp\}$.

Consider a partial function $f : D \to R$ to be a total function $f : D \to R^\perp$. Define equality of partial functions over the same domain as follows:

$$f = g \quad \Leftrightarrow \quad \forall x \in D. f(x) = g(x)$$

We always extend the domain and range of partial functions $f : D \to R$ so that it is a total function $f : D^\perp \to R^\perp$. We assume the following:

$$\begin{aligned}
\textbf{if } T \textbf{ then } A \textbf{ else } \ldots &\mapsto A \\
\textbf{if } F \textbf{ then } \ldots \textbf{ else } B &\mapsto B \\
\textbf{if } \perp \textbf{ then } \ldots \textbf{ else } \ldots &\mapsto \perp
\end{aligned}$$

We also assume $\perp + 1 \mapsto \perp$, etc., as well as $\perp = \perp \mapsto \perp$.

Consider the function definition:

$$f \quad \stackrel{!}{=} \quad \lambda x, y.\ \textbf{if } x = y \textbf{ then } y + 1 \textbf{ else } f(x, f(x-1, y+1))$$

A *fixpoint* of such a definition is a partial function that satisfies the equation. Examples of fixpoints of the above include

$$\begin{aligned}
f_1 &= \lambda x, y.\ \textbf{if } x = y \textbf{ then } y + 1 \textbf{ else } x + 1 \\
f_2 &= \lambda x, y.\ \textbf{if } x < y \textbf{ then } y - 1 \textbf{ else } x + 1 \\
f_3 &= \lambda x, y.\ \textbf{if } x \geq y \wedge 2 | (x - y) \textbf{ then } x + 1 \textbf{ else } \perp
\end{aligned}$$

The definedness of fixpoints (and functions in general) can be compared, as follows:

$$f \sqsubseteq g \quad \Leftrightarrow \quad \forall x \in D^\perp. f(x) \sqsubseteq g(x)$$

where the partial ordering $\sqsubseteq$ on domain elements is defined as $\perp \sqsubseteq x$ and $\perp \sqsubseteq x$ for all $x \in D^{\perp}$. The *least fixpoint* of a function definition is the smallest fixpoint in this ordering.

The always undefined function is

$$\Omega(\bar{x}) \quad \stackrel{!}{=} \quad \perp$$

For any function f, we have $\Omega \sqsubseteq f$.

A function f is *monotonic* if

$$x \sqsubseteq y \quad \Rightarrow \quad f(\ldots x \ldots) \sqsubseteq f(\ldots y \ldots)$$

Constants are monotonic zero-ary functions. **if-then-else** is monotonic. A function f is *strict* if $f(\ldots \perp \ldots)$ always yields $\perp$. Strict functions are monotonic.

The limit (least upper bound) $\lim_{i \to \infty} f_i$ of a chain $f_0 \sqsubseteq f_1 \sqsubseteq \cdots$ of functions is the smallest function g such that $f_i \sqsubseteq g$, for all i. It need not exist.

A monotonic function(al) B is *continuous* if

$$\lim_{i \to \infty} B[f_i] = B[\lim_{i \to \infty} f_i]$$

for any chain $\{f_i\}$. The composition of monotonic functions is continuous.

Theorem 1 (First Recursion Theorem (Kleene)) *Every continuous function $B[f]$ has a unique least fixpoint,* $\lim_{i \to \infty} B^i[\Omega]$.

A *computation rule* for recursive programs is a function that chooses a subset of redexes in a term for replacement. Examples include *call-by-value* (leftmost-innermost); *call-by-name* (leftmost-outermost); *parallel-outermost* (all outermost redexes); and *outermost-fair* (no outermost redex ignored forever).