Transfinite Semantics in the form of Greatest Fixpoint

Härmel Nestra

Institute of Computer Science

University of Tartu

e-mail: `harmel.nestra@ut.ee`

Transfinite semantics

Transfinite semantics

Transfinite semantics: program execution can continue after completing an infinite subcomputation.

- Studied during the last decade.
- Can entail:
  * transfinite traces of execution steps in the case of iteration;
  * fractal traces of execution steps in the case of recursion.
- Useful in formalizing program slicing to avoid semantic anomaly.

Program slicing: example

```
^0 sum := 0;
^1 prod := 1;
^2 i := 0;
 while ^3 i < n do
 (
   ^4 i := i + 1;
   ^5 sum := sum + i;
   ^6 prod := prod * i
 );
^7
```

$\longrightarrow$

```
^0 sum := 0;

^2 i := 0;
 while ^3 i < n do
 (
   ^4 i := i + 1;
   ^5 sum := sum + i;

 );
^7
```

Criterion: $\{(7, \texttt{sum})\}$.

Semantic anomaly: example

If the original program loops then we might have slices which assign to interesting variables more times than the original program:

^{0}**while** true **do** **skip** ; ^{1}x **:=** 0 ; ^{2}	$\longrightarrow$	^{1}x **:=** 0 ^{2}

Criterion: $\{(2, \mathtt{x})\}$.

Greatest Fixpoint

Goal: greatest fixpoint form

We represent transfinite semantics in the form of greatest fixpoint of a monotone operator on complete lattices.

Subgoals

- Express transfinite semantics in a standard framework.
  - Express both transfinite and standard semantics in a uniform algebraic way.
- Provide an exhaustive definition of infinitely deep recursion semantics.
- As a plan for future: build a Cousot's hierarchy.

Epiphenomenons

- Usual traces must be replaced by either fractional traces or trees.
- Explicit determinism is lost.

Fractional semantics

Fractional traces

In the case of fractional traces, computation steps are indexed by rational numbers from a fixed interval.

- The interval of rationals within which an execution of a statement of the program falls does not depend on the initial state.
- Traces grow into depth rather than into length.

Example: swap

The fractional trace of the execution of program

$$\mathsf{z} \;\mathtt{:=}\; \mathsf{x} \,;\, (\mathsf{x} \;\mathtt{:=}\; \mathsf{y} \,;\, \mathsf{y} \;\mathtt{:=}\; \mathsf{z})$$

at initial state

$$\left\{\begin{array}{l} \mathsf{x} \mapsto 1 \\ \mathsf{y} \mapsto 2 \\ \mathsf{z} \mapsto 0 \end{array}\right\}$$

is

$$\begin{array}{cccc} 0 & \frac{1}{2} & \frac{3}{4} & 1 \\ \left\{\begin{array}{l} \mathsf{x} \mapsto 1 \\ \mathsf{y} \mapsto 2 \\ \mathsf{z} \mapsto 0 \end{array}\right\} & \left\{\begin{array}{l} \mathsf{x} \mapsto 1 \\ \mathsf{y} \mapsto 2 \\ \mathsf{z} \mapsto 1 \end{array}\right\} & \left\{\begin{array}{l} \mathsf{x} \mapsto 2 \\ \mathsf{y} \mapsto 2 \\ \mathsf{z} \mapsto 1 \end{array}\right\} & \left\{\begin{array}{l} \mathsf{x} \mapsto 2 \\ \mathsf{y} \mapsto 1 \\ \mathsf{z} \mapsto 1 \end{array}\right\} \end{array}$$

Example: infinite loops

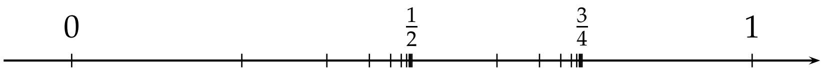
If

$$S_1 = S_2 = \texttt{\textbf{while} true \textbf{do} skip}$$
$$S_3 = \texttt{x \textbf{:=} 1}$$

then the domain of the execution trace of statement

$$S_1 \,;\, (S_2 \,;\, S_3)$$

is depicted in the following figure:



Tree semantics

Trees

In tree semantics, an execution is depicted in the form of tree.

- The tree structure reflects the proof of that execution within a deduction system.

Example: swap

Here is the tree of the execution of the swap program

$$\texttt{z := x ; (x := y ; y := z)}$$

at the same initial state as before:

$$\dfrac{\dfrac{}{\left\{\begin{array}{c}\mathsf{x}\mapsto 1\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 0\end{array}\right\} \to \left\{\begin{array}{c}\mathsf{x}\mapsto 1\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 1\end{array}\right\}} \qquad \dfrac{\dfrac{}{\left\{\begin{array}{c}\mathsf{x}\mapsto 1\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 1\end{array}\right\} \to \left\{\begin{array}{c}\mathsf{x}\mapsto 2\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 1\end{array}\right\}} \qquad \dfrac{}{\left\{\begin{array}{c}\mathsf{x}\mapsto 2\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 1\end{array}\right\} \to \left\{\begin{array}{c}\mathsf{x}\mapsto 2\\ \mathsf{y}\mapsto 1\\ \mathsf{z}\mapsto 1\end{array}\right\}}}{\left\{\begin{array}{c}\mathsf{x}\mapsto 1\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 1\end{array}\right\} \to \left\{\begin{array}{c}\mathsf{x}\mapsto 2\\ \mathsf{y}\mapsto 1\\ \mathsf{z}\mapsto 1\end{array}\right\}}}{\left\{\begin{array}{c}\mathsf{x}\mapsto 1\\ \mathsf{y}\mapsto 2\\ \mathsf{z}\mapsto 0\end{array}\right\} \to \left\{\begin{array}{c}\mathsf{x}\mapsto 2\\ \mathsf{y}\mapsto 1\\ \mathsf{z}\mapsto 1\end{array}\right\}}$$

Example generalized

For any program of form S_1 ; (S_2 ; S_3), the tree grows as follows:

$$
\dfrac{\dfrac{\vdots}{s_0 \longrightarrow s_{\frac{1}{2}}} \qquad \dfrac{\dfrac{\vdots}{s_{\frac{1}{2}} \longrightarrow s_{\frac{3}{4}}} \qquad \dfrac{\vdots}{s_{\frac{3}{4}} \longrightarrow s_1}}{s_{\frac{1}{2}} \longrightarrow s_1}}{s_0 \longrightarrow s_1}
$$

The framework and results

Language

- Statements:

$$
\begin{aligned}
\mathit{Stmt} \rightarrow\ & \mathit{Var}\ \texttt{:=}\ \mathit{Expr} \\
\mid\ & \mathit{Stmt}\ ;\ \mathit{Stmt} \\
\mid\ & \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Stmt}\ \texttt{else}\ \mathit{Stmt} \\
\mid\ & \texttt{while}\ \mathit{Expr}\ \texttt{do}\ \mathit{Stmt} \\
\mid\ & \texttt{call}\ \mathit{Proc}(\mathit{Var},\ \ldots,\ \mathit{Var})
\end{aligned}
$$

- Modules:

$$
\begin{aligned}
\mathit{Module} \rightarrow\ & \texttt{proc}\ \mathit{Proc}(\mathit{Var},\ \ldots,\ \mathit{Var})\ \texttt{is}\ \mathit{Stmt} \\
\mid\ & \mathit{Module}\ ;\ \mathit{Module}
\end{aligned}
$$

Kinds of semantics

We have considered the following kinds:

	Finite	Standard	Transfinite
Integral trace	$\overrightarrow{+}$	$\overrightarrow{\omega}$	$\overrightarrow{\propto}$
Fractional trace	$\widetilde{+}$	$\widetilde{\omega}$	$\widetilde{\propto}$
Tree	$\widehat{+}$	$\widehat{\omega}$	$\widehat{\propto}$

Domains

$$\begin{aligned} \mathrm{Val} & \quad \text{the set of values} \\ \mathrm{State} &= \mathit{Var} \to \mathrm{Val} \end{aligned}$$

$$\begin{aligned} \mathrm{Dom}_\kappa & \quad \text{the set of individual semantic objects (traces, trees etc.)} \\ \mathrm{Env}_\kappa &= \mathit{Proc} \to (\mathrm{State} \to \mathrm{Val}^*) \to \wp(\mathrm{Dom}_\kappa) \end{aligned}$$

The semantic domains $\wp(\mathrm{Dom}_\kappa)$ are equipped with inclusion order, lifted componentwise to functions.

Signatures

- Statement level.

$$\mathcal{F}_\kappa \in \mathrm{Env}_\kappa \to (\mathit{Stmt} \to \wp(\mathrm{Dom}_\kappa)) \to (\mathit{Stmt} \to \wp(\mathrm{Dom}_\kappa))$$
$$\mathcal{S}_\kappa \in \mathrm{Env}_\kappa \to (\mathit{Stmt} \to \wp(\mathrm{Dom}_\kappa))$$
$$\mathcal{S}_\kappa(S)(e) = \mathrm{gfp}(\mathcal{F}_\kappa(e))(S)$$

- Module level.

$$\mathcal{G}_\kappa \in (\mathit{Module} \to \mathrm{Env}_\kappa) \to (\mathit{Module} \to \mathrm{Env}_\kappa)$$
$$\mathcal{T}_\kappa \in \mathit{Module} \to \mathrm{Env}_\kappa$$
$$\mathcal{T}_k(M) = \mathrm{gfp}(\mathcal{G}_k)(M)$$

Correctness

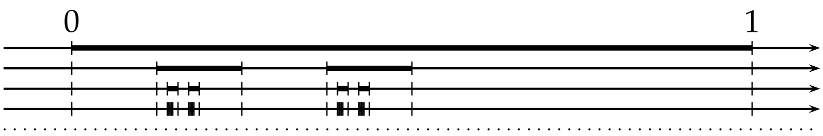
- The functions $\mathcal{F}_\kappa(e)$ and $\mathcal{G}_\kappa$ are monotone.
  - By Tarski's theorem, the greatest fixpoint always exists and the definition is correct.
- The functions $\mathcal{F}_\kappa(e)$ and $\mathcal{G}_\kappa$ are Scott-cocontinuous for $\kappa = \widetilde{\propto}$, $\kappa = \widehat{\propto}$.
  - ∗ By Kleene's theorem, the greatest fixpoint of the transfinite semantics can be obtained by an iteration which is not transfinite!

Example

Let procedure `q` be defined by

```
proc q() is (call q();call q())
```

The iteration of its semantics goes as follows:



Remarks

The choice of the kind of semantics

Why do we need the fractional traces or trees? Why couldn't we use transfinite sequences?

- It is not possible to express fractal structures that arise in the case of infinitely deep recursion using transfinite sequences.
- Even in the case of infinite iteration only, the greatest fixpoint of our function would contain too many traces.
  - Besides the desired traces, all traces having a desired trace as a prefix would be included.
  - But in fractional semantics, the interval $[0;1]$ is wholly distributed between all statements occurring in the program and no space is left for garbage.

Connection between different kind of semantics

Fractional traces reflect the deduction tree structure within a linear order. They have both trace and tree properties.

This way, fractional semantics is an intermediate level between trace and tree semantics.

Non-determinism

The price we pay in this approach is that explicit determinism is lost.

- It is not clear whether the execution trace of a program at an initial state is unique.
- It is not clear whether there exists an execution trace after all!
    - What would the absence of execution traces mean?

Under some natural restrictions, it can be proven that non-determinism can be introduced by infinitely deep recursion only.