Transfinite Semantics for Program Slicing

Härmel Nestra

Institute of Computer Science

University of Tartu

e-mail: harmel.nestra@ut.ee

Program Slicing

An example

`sum := 0;`
`prod := 1;`
`i := 0;`
`while i < n do (`
`  i := i + 1;`
`  sum := sum + i;`
`  prod := prod * i`
`);`
	$\longrightarrow$	`sum := 0;`
		
		`i := 0;`
		`while i < n do (`
		`  i := i + 1;`
		`  sum := sum + i;`
		
		`);`

Criterion: $\{(\lambda, \texttt{sum})\}$, where λ denotes the end point.

An attempt to define

Let P be a program and $\mathcal{X}$ the set of all variables in it.

A **slicing criterion** is a set $C \subseteq V(\text{cfg}\, P) \times \mathcal{X}$.

A **slice of** P **w.r.t.** C is any program Q where

- all statements occur also in P in the same order,
- for every $(\mathsf{p}, Z) \in C$, program Q computes the same sequence of values as program P for program point p and variable Z.

A problem

If the original program loops then we might have slices which execute the program points in the criterion more times than the original program.

For example, for criterion $\{(\lambda, \{\mathtt{x}\})\}$ where λ is the end vertex, we get

```
a := 1;
x := 0
```
$\longrightarrow$
```
x := 0
```

```
while true do ;
x := 0
```
$\longrightarrow$
```
x := 0
```

So-called **semantic anomaly**.

One more attempt to define

A **slice of** P **w.r.t.** C is any program Q where

- all statements occur also in P in the same order,
- for every $(\mathsf{p}, Z) \in C$, program Q computes a sequence of values at p for Z which contains the sequence of values computed by P at p for Z as a prefix and equals to it if P terminates.

Not good again

If the original program is

> **while** `true` **do** ;
> S,

then the 2nd condition in the definition is trivially true, so any subset of S is considered as a slice.

Transfinite semantics

The consequence is that the standard trace semantics is not suitable for describing program slicing.

Transfinite trace semantics is a semantics according to which computation may continue after an infinite number of steps from some limit state determined somehow by the infinite computation performed.

This semantics of a program, assuming the initial state being fixed, gives a transfinite list of states or, if the computation is indeterministic, a set of transfinite lists. The states in the transfinite list are normally indexed with ordinals.

$$s_0, s_1, s_2, \ldots\ldots s_\omega, s_{\omega+1}, \ldots s_{\omega\cdot 2}, s_{\omega\cdot 2+1}, \ldots\ldots s_{\omega^2}, s_{\omega^2+1}, \ldots$$

The body of every loop is repeated at most ω times during one execution.

How to Define Transfinite Semantics?

Proposal: transfinite corecursion

Transfinite corecursion would enable us to define transfinite semantics easily.

A Standard Example on Corecursion

Functions giving lists of *a priori* unknown length as values are often defined using **corecursion**.

This example defines continued fractions:

$$c(x) = \lfloor x \rfloor : \begin{cases} c\left(\frac{1}{\langle x \rangle}\right), & \text{if } x \notin \mathbb{Z}, \\ \varepsilon & \text{otherwise} \end{cases}$$

Transfinite Corecursion

By **transfinite corecursion** we mean some analogous way of defining functions giving transfinite lists as values.

Transfinite list is a sequence of values which can continue after an infinite number of components from some limit value.

$$1, 0, \underbrace{1, 1, 1, \ldots\ldots}_{\text{infinitely many}} 1, 0, \ldots$$

Direction of generalization

$\omega, \omega \cdot 2, \ldots, \omega^2, \omega^2 + \omega, \ldots$ are called **limit** ordinals (they don't have an immediate predecessor).

We must specify the rest of the transfinite list under construction after any limit number of initial components.

No need for handling every limit ordinal as a special case.

Selfish ordinals

We call an ordinal $\gamma > 0$ **selfish** if, for every $\alpha < \gamma$ and x,

$$\alpha + x = \gamma \iff x = \gamma.$$

In other words, γ is selfish iff the well-order of the part remaining when cutting away any initial segment from the well-order represented by γ is isomorphic to γ itself.

Selfish are those non-zero ordinals that cannot be represented as the sum of two ordinals both less than it.

For example, ω, ω^2, ω^3, etc. are selfish; $\omega \cdot 2 = \omega + \omega$ is not since representable as a sum of two ordinals both less than it.

Puzzle: are all the selfish ordinals limit ordinals?

A lemma

(i) Every ordinal $o > 0$ is uniquely representable in the form $o = \alpha + \gamma$ where γ is selfish and α is the least ordinal for which this equality holds.

(ii) Every ordinal $o > 0$ is uniquely representable in the form $o = \gamma + \beta$ where γ is selfish and $\beta < o$.

Theorem?

Let X, A be sets. Let

$$\varphi : X \to 1 + A,$$
$$\psi : \left(\bigcup_{\substack{\gamma < \propto \\ \gamma \text{ selfish}}} (\mathbb{O}_\gamma \to A) \right) \to X.$$

Then there exists a unique $f : X \to \mathrm{TList}\, A$ such that

1. if $\varphi(x) = \mathrm{Right}\, a$ then $\mathrm{head}(f(x)) = a$,
   if $\varphi(x) \in 1$ then $f(x) = \varepsilon$;
2. if $|f(x)| \geqslant \gamma$ and $\gamma < \propto$, γ selfish, then

$$\mathrm{drop}\, \gamma\, (f(x)) = f(\psi(\mathrm{take}\, \gamma\, (f(x)))).$$

An error

Suppose $\mathrm{take}\,\omega(f(x)) = l$ and $\psi(l) \neq \psi(\mathrm{tail}\,l)$. We must have

$$
\begin{aligned}
f(\psi(l)) &= f(\psi(\mathrm{take}\,\omega\ (f(x)))) = \\
&= \mathrm{drop}\,\omega\ (f(x)) = \\
&= \mathrm{drop}\,\omega\ (\mathrm{drop}\,1\ (f(x))) = \\
&= \mathrm{drop}\,\omega\ (f(\psi(\mathrm{take}\,1\ (f(x))))) = \\
&= f(\psi(\mathrm{take}\,\omega\ (f(\psi(\mathrm{take}\,1\ (f(x))))))) = \\
&= f(\psi(\mathrm{take}\,\omega\ (\mathrm{drop}\,1\ (f(x))))) = \\
&= f(\psi(\mathrm{drop}\,1\ (\mathrm{take}\,\omega(f(x))))) = \\
&= f(\psi(\mathrm{tail}\,l))
\end{aligned}
$$

which can break.

Theorem (for deterministic semantics)

Let X, A be sets. Let

$$\varphi : X \to 1 + A,$$

$$\psi : \left(\bigcup_{\substack{\gamma < \propto \\ \gamma \text{ selfish}}} (\mathbb{O}_\gamma \to A) \right) \to X.$$

Assume

$$\psi(l) = \psi(\mathrm{drop}\, \lambda\, l).$$

for any selfish ordinals λ, γ, $\lambda < \gamma < \propto$ and transfinite list $l \in (\mathbb{O}_\gamma \to A)$. Then there exists a unique $f : X \to \mathrm{TList}\, A$ such that

1. if $\varphi(x) = \mathrm{Right}\, a$ then $\mathrm{head}(f(x)) = a$,
   if $\varphi(x) \in 1$ then $f(x) = \varepsilon$;
2. if $|f(x)| \geqslant \gamma$ and $\gamma < \propto$, γ selfish, then

$$\mathrm{drop}\, \gamma\ (f(x)) = f(\psi(\mathrm{take}\, \gamma\ (f(x)))).$$

Usual lists as a special case

Taking $\gamma = 1$ in condition 2, one gets

$$\mathrm{tail}(f(x)) = f(\psi(\mathrm{head}(f(x))))$$

for non-empty lists l.

Theorem (for non-deterministic semantics)

Let X, A be sets. Let

$$\varphi : X \to 1 + A,$$
$$\psi : \left(\bigcup_{\substack{\gamma < \propto \\ \gamma \text{ selfish}}} (\mathbb{O}_\gamma \to A) \right) \to \wp X.$$

Assume that, for any selfish ordinals λ, γ such that $\lambda < \gamma < \propto$, and transfinite list $l \in (\mathbb{O}_\gamma \to A)$, the equality

$$\psi(l) = \psi(\mathrm{drop}\, \lambda\, l)$$

holds. Then there is a largest function $f : X \to \wp(\mathrm{TList}\, A)$ such that

1. if $\varphi(x) = a \in A$ then, for every $l \in f(x)$, $\mathrm{head}\, l = a$,
   if $f(x) \in 1$ then $f(x) = \{\varepsilon\}$;
2. for any selfish γ and transfinite list $l \in (\mathbb{O}_\gamma \to A)$ such that there exists an $m \in f(x)$ for which $\mathrm{take}\, \gamma\, m = l$, the equality

$$\{m \in f(x) \mid \mathrm{take}\, \gamma\, m = l \bullet \mathrm{drop}\, \gamma\, m\} = \bigcup_{z \in \psi(l)} f(z)$$

   holds.

Further Notes

Our work in progress

Our aim is to define the relation “being slice of” emanating from transfinite semantics, abstracting from syntax.

This would enable, for example, to talk about correctness of slicing algorithms.

In

> R. Giacobazzi, I. Mastroeni. *Non-Standard Semantics for Program Slicing.* In: Higher-Order and Symbolic Computation, 16, 297–339, 2003,

an attempt has been made to describe the relation “being slice of” via transfinite semantics. However, they make several assumptions losing the generality.

The main result of Giacobazzi et al on slicing

They define a transfinite semantics $(\mathcal{S}; \sqsubseteq)$ such that, for any programs P, Q, if Q is a slice of P then $\mathcal{S}[\![Q]\!] \sqsubseteq \mathcal{S}[\![P]\!]$.

The proof uses essentially the assumption that every variable occurring in the slice has the same final value as in the original program. This does not hold generally.

Example 1

$$\boxed{\begin{array}{l} {}^{0}\texttt{x} \mathrel{\texttt{:=}} 1; \\ {}^{1}\texttt{x} \mathrel{\texttt{:=}} 2 \end{array}} \longrightarrow \boxed{\texttt{x} \mathrel{\texttt{:=}} 1}$$

(criterion: $\{(1, \mathtt{x})\}$).

The limit operator

How to define the state from which the computation continues after an infinite limit ordinal number of steps?

Let $X \in \mathcal{X}$. Define

$$(l_\gamma)_X = \begin{cases} v, & \text{kui } \mathrm{map}\,\mathrm{pr}_X(\mathrm{take}\,\gamma\, l) \text{ stabilizes at } v, \\ \top & \text{otherwise.} \end{cases}$$

Example 2

```
while i>0 do (
  i := 1;
  i := 2
);
x := i
```

$\longrightarrow$

```
while i>0 do (

  i := 2
);
x := i
```

The final value of `x` is $\top$ in the original program but 2 in the slice.

Program points

Program point system is an equivalence relation on all possible states the computation can be.

A **program point** w.r.t system $\sim$ is a class of $\sim$.

The set of vertices of the control flow graph is a subset of the factor set w.r.t $\sim$.

Transfinite arcs

Every infinite sequence l of length γ (selfish) of program points determines a non-empty set of **transfinite arcs** (p, q) where

- for infinitely many ordinals $\alpha < \gamma$, $l_\alpha \in \mathsf{p}$ and $l_{\alpha+1} \not< l_\alpha$,
- p is a minimal among such program points,
- $\mathsf{q} = \max\{\mathsf{r} \mid \mathsf{r} < \mathsf{p} \bullet \mathsf{r}\}$.

What are the transfinite arcs of

$(\texttt{while}\ \texttt{true}\ \texttt{do}\ S); T$,

$\texttt{while}\ \texttt{true}\ \texttt{do}\ \texttt{while}\ \texttt{true}\ \texttt{do}\ S$?

The limit operation revisited

The limit state is defined according to the transfinite arc the computation uses for escaping the loop.

Let $X \in \mathcal{X}$ and escaping use the transfinite arc (p, q) at γ. Then

$$(l_\gamma)_X = \begin{cases} v, & \text{kui } \mathrm{map}\,\mathrm{pr}_X(\mathrm{filter}(\in \mathsf{p})\ (\mathrm{take}\,\gamma\, l)) \text{ stabilizes at } v, \\ \top & \text{otherwise.} \end{cases}$$

Example 3

^{0}**if** B **then** (
  ^{1}b := 2;
  ^{2}a := 1
) **else** (
  ^{3}b := 2
);
^{4}x := b
	$\longrightarrow$	b := 2;
		
		x := b

Is the right-hand program a slice w.r.t criterion $\{(2, \texttt{b})\}$?