A Simple Semantic Model for Flat GHC

3D-2

Kazunori Ueda and Koichi Furukawa

Institute for New Generation Computer Technology

1. Introduction

In designing a set of transformation rules for Flat GHC programs [1], we were faced with the problem of justifying them on an appropriate semantic model. This paper informally describes the semantic model we designed for that purpose. The full description of the semantics will be found in [1].

2. Design Criteria

- *Modeling Behaviors.* A multiset of GHC goals can be regarded as a process that communicates with the outside world by observing and generating substitutions. The semantics should model this behavioral aspect.
- *Abstractness.* The semantics should concentrate on communication. It should abstract away internal affairs of a process such as the number of (sub)goals and the number of commitments done. Also, it should abstract away *how* unification is specified in the source text.
- *Modeling non-terminating programs.* We must be able to define the semantics of programs that do not terminate but are still useful.
- *Modeling anomalous behaviors.* Anomalous behaviors such as failure of a unification goal in a clause body, irreducibility of a non-unification goal and infinite computation without observable substitution must be modeled, because we have to prove that such behaviors are not introduced by program transformation.
- *Simplicity and generality.* The semantics should be as simple and general as possible to be widely used. We decided to use standard tools like *finite* terms, substitutions defined over them, and least fixpoints. We decided *not* to use mode systems. We decided *not* to handle discontinuous concepts like fairness.
- *Usefulness.* It should not be just a description, but a tool to be used (at least) for proving the correctness of the transformation rules.

3. The Model

The semantics of a multiset B_0 of goals under a program $\mathcal{P}$, denoted $[\![B_0]\!]_{\mathcal{P}}$, is modeled as the set of all possible finite sequences of transactions with it. A *(normal) transaction*, denoted $\langle \alpha, \beta \rangle$, is an act of providing a multiset of goals with a possibly empty *input substitution* α and getting an observable (see below) *output substitution* β. An output substitution is also called a *partial answer substitution.*

The first transaction $\langle \alpha_1, \beta_1 \rangle$ must be made through the variables in $\mathrm{var}(B_0)$, called the *interface.* The above observability condition for β_1 can be written as $B_0\alpha_1\beta_1 \not\equiv B_0\alpha_1$. As the result of the first transaction, B_0 will be reduced to a multiset B_1 of goals, which represents the rest of the computation. Then the second transaction $\langle \alpha_2, \beta_2 \rangle$ must be made through the interface $\mathrm{var}(B_0\alpha_1\beta_1)$.

The 'size' of a transaction depends on how the outside world observes an output substitution. Suppose B_0 returns a complex (or even infinite) data structure t in response to an input α_1. What should β_1 be, or what should the outside world see in one transaction? The answer is that the outside world can observe any *finite* template of t (i.e., a term of which t is an instance). In our model, the result

A Simple Semantic Model for Flat GHC
Kazunori Ueda, Koichi Furukawa
Institute for New Generation Computer Technology

of one unification goal may be observed using two or more transactions, and the result of two or more unification goals may be observed in one transaction. A transaction is of a finite nature; it is realized by a finite number of reductions and can return only a finite data structure.

The outside world may not communicate with B_0 at all. This is modeled by always including ϵ (empty sequence) in $[\![B_0]\!]_{\mathcal{P}}$. The empty sequence is used as a base case in defining the model of B_0 inductively.

An input α_1 to B_0 may not necessarily cause a normal transaction as defined above. First, it may cause failure of a unification goal in a clause body. This is modeled by letting $[\![B_0]\!]_{\mathcal{P}} \ni \langle \alpha_1, \top \rangle$, where $\top$ means failure. Second, B_0 may succeed (i.e., be reduced out) with no observable output. Third, B_0 may deadlock (i.e., be reduced to a multiset of goals that does not allow further reduction) with no observable output. Fourth, B_0 may fall into infinite computation that does not generate observable output. The last three cases mean *inactivity* of B_0 and cannot be distinguished from outside; so they are all modeled by letting $[\![B_0]\!]_{\mathcal{P}} \ni \langle \alpha_1, \bot \rangle$, where $\bot$ stands for 'no output'. However, if necessary, these cases could be distinguished in the model by using $\bot_{success}$, $\bot_{deadlock}$ and $\bot_{divergence}$ instead of $\bot$. Failure and inactivity are called *special transactions* and are used as base cases in defining the model of B_0.

4. Examples

Consider a single clause program

$\mathcal{P}$: `p(X) :- true | X=f(Y), p(Y).`

and autonomous (i.e., empty input) transactions with $\mathcal{P}$. Then $[\![\texttt{p(X)}]\!]_{\mathcal{P}}$ has

ϵ, $\langle \emptyset, \{\texttt{X} \leftarrow \texttt{f(X1)}\} \rangle$,

$\langle \emptyset, \{\texttt{X} \leftarrow \texttt{f(X1)}\} \rangle \langle \emptyset, \{\texttt{X1} \leftarrow \texttt{f(X2)}\} \rangle$,

$\langle \emptyset, \{\texttt{X} \leftarrow \texttt{f(X1)}\} \rangle \langle \emptyset, \{\texttt{X1} \leftarrow \texttt{f(X2)}\} \rangle$
$\langle \emptyset, \{\texttt{X2} \leftarrow \texttt{f(X3)}\} \rangle$,

...

and also

$\langle \emptyset, \{\texttt{X} \leftarrow \texttt{f(f(X2))}\} \rangle$,

$\langle \emptyset, \{\texttt{X} \leftarrow \texttt{f(f(f(X3)))}\} \rangle$,

... .

Our model successfully circumvents Brock-Ackerman anomaly [2]. Let $\mathcal{BA}$ be:

```
d([A|_],O) :- true | O=[A,A].

merge([A|X1],Y,Z) :- true |
  Z=[A|Z1],merge(X1,Y,Z1).
merge(X,[A|Y1],Z) :- true |
  Z=[A|Z1],merge(X,Y1,Z1).
merge([],Y,Z) :- true | Z=Y.
merge(X,[],Z) :- true | Z=X.

p1([A|Z1],O) :- true |
  O=[A|O1],p11(Z1,O1).
p11([B|_],O1) :- true | O1=[B].

p2([A,B|_],O) :- true | O=[A,B].

g1(I,J,O) :- true |
  d(I,X),d(J,Y),merge(X,Y,Z),p1(Z,O).
g2(I,J,O) :- true |
  d(I,X),d(J,Y),merge(X,Y,Z),p2(Z,O).
```

Then, the computation

$\langle \{\texttt{I} \leftarrow \texttt{[5|\_]}\}, \{\texttt{O} \leftarrow \texttt{[5|O']}\} \rangle$

belongs both to $[\![\texttt{g1(I,J,O)}]\!]_{\mathcal{BA}}$ and to $[\![\texttt{g2(I,J,O)}]\!]_{\mathcal{BA}}$ (O' being a fresh variable), but

$\langle \{\texttt{I} \leftarrow \texttt{[5|\_]}\}, \{\texttt{O} \leftarrow \texttt{[5|O']}\} \rangle$
$\langle \{\texttt{J} \leftarrow \texttt{[6|\_]}\}, \{\texttt{O'} \leftarrow \texttt{[6]}\} \rangle$

belongs only to $[\![\texttt{g1(I,J,O)}]\!]_{\mathcal{BA}}$ and not to $[\![\texttt{g2(I,J,O)}]\!]_{\mathcal{BA}}$.

References

[1] Ueda, K. and Furukawa, K. (1988) Transformation Rules for GHC Programs. To appear as ICOT Tech. Report.

[2] Brock, J. D. and Ackerman, W. B. (1981) Scenarios: A Model of Non-determinate Computation. In *Formalization of Programming Concepts*, LNCS 107, Springer-Verlag, pp. 252–259.