Relational Semantics

Mark Saaltink
I.P. Sharp Associates
265 Carling Avenue, Suite 600
Ottawa, Ontario, CANADA

1.0 INTRODUCTION

Relational Semantics is a technique for describing the imperative aspects of programming languages, based within a simple set theoretic framework.

The relational approach has the benefits of (1) separating the modelling concerns of statements and expressions, (2) simplifying the mathematical foundation, when compared with other techniques, and thereby making the work more accessible and amenable to metamathematical analyses, and (3) allowing for underdetermined semantics or for the semantics of nondeterministic statements.

These benefits come at some cost; we can not handle the possibility of "jumps" from expression evaluation (as might happen in a Pascal program), nor do we allow recursive domains for expressions. This limits the applicability of the technique to "well behaved" languages. In practice, this has not been a harsh restriction. The language we are formalizing is to be verifiable, and our experience is that the requirements from the proof system are more stringent than those from this semantic technique.

2.0 THE METHOD

We view the meaning of a program as a binary relation upon states of the mechanism it controls. This relation will consist of all pairs of states, <s,t> say, where invocation in a state s can result in the program terminating in the state t. This use of relations allows for the formal characterization of nondeterministic mechanisms. It does not, however, allow for a characterization of non-termination or failure of a mechanism.

The meaning of of program or program part is described by semantic functions in the same manner as in Denotational Semantics. If necessary, Stores, Environments, Locations, and Continuations can be used in the defining equations. The best introduction to the technique would be introduction to Denotational Semantics such as [Gor79] or [Ten76]. It should be borne in mind, however, that this technique is not the same as Denotational Semantics. It can be thought of as a simple framework where many of the techniques of Denotational Semantics can be applied.

We will illustrate the techniques here with a few examples.

A variable declaration associates a type with the variable identifier. The meaning of such a declaration is that the state is modified to hold the new type information and value of the variable:

```
M(var id:tx) =
    { <s,t> | for some v in ValuesOfType(tv):
                  t = s[id <- Var(tv,v)]   }

  where
    tv = Mtx(tx,s)
```

In this case, we state that id is to be initialized, possibly nondeterministically, to a value, v, which is within the range of values of the type. One can, of course, debate whether this is an appropriate idea for var declarations, but the formalism makes the decision clear. Further, note that there is interaction at this point between the statement model (in this case Relational Semantics) with the model being used to define expressions. Mtx, which is not addressed in this paper, would define the meaning of the type expression tx, in a state s.

As a second example, the meaning of a while statement is defined as follows:

```
M(while B do S) = smallest relation R satisfying
 R = { <s, t> | if Mex(B, s) = true
                  then <s, t> in ( M(S) o R )
                  else s = t
     }
```

The existence of such a relation R is guaranteed by various fixed-point theorems. (Here "o" denotes relational composition, and "in" denotes set membership; Mex(E, s) is the meaning of expression E in state s.)

3.0 RELATIONSHIP TO OTHER WORK

The relational approach to semantics has been used by Parnas in describing the semantics of a generalized iteration construct [Par83]. We have extended the relational approach to deal with language features, such as procedures and sequencers, not covered in Parnas' work. The semantics described in this paper is somewhat simplified (but weakened) by the omission of the "competence sets" Parnas uses to deal with termination.

Pratt has investigated the application of Relational Semantics to Floyd-Hoare logics [Pra76], focussing primarily on proof issues. His work does not, however, deal directly with the

issues of defining the semantics of a complete programming language.

Obviously, we have used many techniques and concepts from Denotational Semantics, including continuations, environment and store, and the limit construction of least fixed-points. The need for Domains rather than Sets has been avoided, allowing not only a simpler mathematical foundation, but also (we hope) leading to more tractable proofs of consistency of proof systems with semantics.

There is a fairly direct relation between the model of programs as relations that we use, and the model of programs as predicates proposed by Hehner [Heh84]. It is just the same relation as holds between a predicate and a set. However, Hehner differs in his treatment of nonterminating programs. If program P fails to terminate when started in state s, Hehner would have <s,t> in Ms(P) for all states t, while we would have it for none.

There is a similar relationship between the relational technique and the use of weakest preconditions [Dij76] to define semantics. It is a simple matter to determine a state relation from a predicate transformer:

Ms(S) = { <s,t> | (x=s) -> Wlp(S, x=t) }

Similarly, a predicate transformer can be determined from a relation

Wlp(S, P) = for all s,t: [<s,t> in Ms(S) and x=s] -> P(t)

We find it conceptually and technically simpler to relate states directly rather than to relate predicates on states.

4.0 CONCLUSION

In this paper we have discussed a semantic technique based on the mathematical concept of relations. A major attraction to this approach is the underlying mathematical simplicity. We find this approach easy to use, and useful in clarifying our understanding of programming language constructs.

The approach discussed in this paper has been applied to Ottawa Euclid, so as to determine a mathematically tractable subset of Ottawa Euclid declarations and statements [Cra83].

5.0 REFERENCES

[Cra83] D. Craigen, M. Saaltink. "A Formal Semantics of VOE: Part 2", I.P. Sharp Associates Technical Report, FR-5172-83-1, November 1983.

[Cra84] D. Craigen, "Ottawa Euclid and EVES: A Status Report", Proceedings of the 1984 IEEE Symposium on Security and Privacy, Oakland, California.

[Dij76] E. W. Dijkstra, "A Discipline of Programming", Prentice-Hall, 1976

[Gor79] M. J. Gordon, "The Denotational Description of Programming Languages", Springer-Verlag, 1979

[Heh84] E. C. R. Hehner, "Predicative Programming Part I", CACM 27(2), Feb. 1984

[Par83] D. L. Parnas, "A Generalized Control Construct and its Formal Definition", CACM 26(8), Aug. 1983

[Pra76] V. R. Pratt, "Semantical Considerations on Floyd-Hoare Logic", 17th IEEE Conf. on Foundations of Comp. Sci., Oct. 1976

[Ten76] R.D. Tennent, "The Denotational Semantics of Programming Languages", CACM 19(8), Aug. 1976.