## Relational logic and its applications to verification.

How to verify several programs at once.

Mihir Mehta

Department of Computer Science University of Texas at Austin mihir@cs.utexas.edu

01 April 2015

## Relational logic

- Relational logic started out as a means to prove program equivalence.
- Hoare quadruples intended to express equivalence of two programs in a certain context.
- Benton's initial work limited to structurally identical programs.
- ► Later, Zaks and Pnueli, cross products again limited to structurally equivalent programs.
- Separately, self-composition sound and complete but gives really hard verification conditions.
- ► This paper: serves to unite both into one that's actually machine-checkable.

▶ The program syntax is:

$$c ::= x := e \mid a[e] := e \mid \mathsf{skip} \mid \mathsf{assert}(b) \mid c; c \mid \mathsf{if} \ b \ \mathsf{then} \ c_1 \ \mathsf{else} \ c_2 \mid \mathsf{while} \ b \ \mathsf{do} \ c$$

▶ The program semantics are:

Fig. 2. Program semantics (excerpt)

$$\frac{c_1 \times c_2 \rightarrow c}{c_1 \times c_2 \rightarrow c} \qquad \frac{c_1 \times c_2 \rightarrow c}{(c_1;c_1') \times (c_2;c_2') \rightarrow c;c'} \\ c_1 \times c_2 \rightarrow c \\ \hline (\text{while } b_1 \text{ do } c_1) \times (\text{while } b_2 \text{ do } c_2) \rightarrow \text{assert}(b_1 \Leftrightarrow b_2); \text{ while } b_1 \text{ do } (c; \text{assert}(b_1 \Leftrightarrow b_2)) \\ c_1 \times c_2 \rightarrow c \qquad c_1' \times c_2' \rightarrow c' \\ \hline (\text{if } b_1 \text{ then } c_1 \text{ else } c_1') \times (\text{if } b_2 \text{ then } c_2 \text{ else } c_2') \rightarrow \text{assert}(b_1 \Leftrightarrow b_2); \text{ if } b_1 \text{ then } c \text{ else } c_2' \\ \hline (\text{if } b \text{ then } c_1 \text{ else } c_2) \times c \rightarrow \text{if } b \text{ then } c_1' \text{ else } c_2' \\ \hline (\text{if } b \text{ then } c_1 \text{ else } c_2) \times c \rightarrow \text{if } b \text{ then } c_1' \text{ else } c_2' \\ \hline \end{cases}$$

Fig. 3. Product construction rules

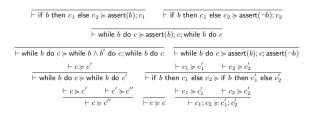


Fig. 4. Syntactic reduction rules

## Our work I

- ▶ We'd like to work with proving properties between the input and output of several calls to the same function.  $\forall x1,...,xn.(Pre(x1,...,xn) \Rightarrow Post(f(x1),...f(xn)))$
- ▶ By making product programs, we should be able to prove properties such as symmetry, reflexivity, transitivity, determinism by expressing them as Hoare pre- and post-conditions and then passing them to a theorem prover (Z3).
- Evidently, we can't assume structural equivalence in general. For instance, conditionals may go down different branches, and loops may have different numbers of iterations.
- However, we should be able to benefit from the somewhat common structure.

## Our work II

- At present, we're trying to implement a verifier of this kind, and also figure out judgement rules that might be useful for the kinds of things we're trying to prove.
- We're focussing on Java programs with comparators those are the main motivating example.
- We're able to deal with loop-free programs, but will re-write to deal better with looping and recursing programs, incorporating the product construction.