Let R be a regular language, and for strings s and t define

$$s \longrightarrow t \Leftrightarrow (\exists x, y, z, r : xyz = s \ \wedge \ xrz = t \ \wedge \ r \in R)$$

Show that $\{t : 0 \longrightarrow^{*} t\}$ is regular.

Getting to first-order

Ernie Cohen

March 30, 2003

why do we want first-order invariants?

- going from HOL to FOL can be a big win for automatic verification
- first-order data = spatial locality, so no problem with pointer swing
- first-order invariants more robust to asynchrony

goal: HOL at the top (strategy) level, FOL at the bottom

today: some good ways to get to first-order invariants

- don't introduce unnecessary inductive structures
- replace inductive structures with first-order structures
- replace spatial induction with temporal induction

ex: permutation inversion

problem: invert a permutation (stored in an array) in-place, using at most one extra bit per element (plus a fixed number of index variables)

obvious approach: invert one cycle at a time

Gasteren/Gries: translate cycles to nonrepeating cyclic sequences spec: $\{(\forall i : b.i = B.i)\}\ P\ \{(\forall i : b.(B.i) = i)\}$

inv1: $(\forall i : m.i \Rightarrow b.(B.i) = i)$

obvious update: $\neg m.i; b.(B.i), m.i := i, true$

problem: how to compute B?

inv2: $\neg m\ B = \neg m\ (cb \lhd b)$

$$\begin{array}{ll} & cb.i := b.i \\ \vee & (m.i; cb.i := \bot) \\ \vee & (\neg m.i; cb.(B.i); b.(B.i), m.i := i, true \end{array}$$

choose operation order to keep the size of cb at most 2

no cycles!

ex: reference counters

idea: a reference counter represents an algebra of sets with operations

- empty set
- test for emptiness
- singleton set
- disjoint union
- subset difference

implement these as $\mathbf{0}$,$= \mathbf{0}$,$\mathbf{1}$,$+$, and $-$

Depth-First Search

usual invariants include “every visited node is marked or an ancestor of the current node in the DFS tree”

instead of defining ancestors inductively, keep the ancestry relation in a ghost variable

FIFO queues

usual approach: model queues as sequences

alternatives:

- use commutativity (as in omega algebra)

$$\boldsymbol{c!u\ c?v = c.e\ \{u = v\} + c?v\ c\neg u}$$

- keep messages as explicit objects; use graph homorphism instead of prefix

ex: distributed dining philosophers

idea: philosopher eats only when it has all forks that it shares

how do we guarantee progress?

solution: make sure that the “waits-for” graph is well-founded.

problem: introduces a recursive data structure; concurrent updates are tricky

alternative solution: maintain the invariant “every eventually hungry philosopher will eventually eat” (global in time, but local in space)

to guarantee the invariant holds initially, make sure that each philosopher eats at least once