Transition Invariants for Program Termination

Andreas Podelski

January 9, 2012

Ramsey's theorem

every infinite complete graph that is colored with finitely many colors contains a monochrome infinite *complete subgraph*

termination

a program P is *terminating* if

- its transition relation R_P is well-founded
- the relation R_P does not have an infinite chain
- there exists no infinite sequence

  $$s_1, s_2, s_3, \ldots$$

  where each pair (s_i, s_{i+1}) is contained in the relation R_P

proving termination

- classical method for proving program termination:
  construction of a ranking function
  (one single ranking function for the entire program)
- construction not supported by predicate abstraction

predicate abstraction

- proof of safety of program
- construction of a (finite) abstract reachability graph
- edges = transitions between (finitely many) abstract states
- abstract reachability graph (with, say, n abstract states) will contain a loop (namely, to accomodate executions with length greater than n)
- example: abstraction of `while(x>0){x--}` with set of predicates $\{(x > 0), (x \leq 0)$
- finiteness of executions can *not* be demonstrated by finiteness of paths in abstract reachability graph

new concepts

- transition invariant: combines several ranking functions into a single termination argument
- transition predicate abstraction: automates the computation of transition invariants using automated theorem proving techniques

backward computation for termination

- terminatingStates = set of terminating states
  = states s that do not have an infinite execution
- exitStates = set of states without successor
- state s terminating if s does not have any successor or every successor of s is a terminating state
- terminatingStates = least solution of fixpoint equation:
  $$X = \text{weakestPrecondition}(X) \cup \text{exitStates}$$
- program terminates if initialStates $\subseteq$ terminatingStates
- check of termination requires abstraction of fixpoint (of function based on weakest precondition) from below
- underapproximation - ???

example program: ANY-Y

```
l1: y := read_int();
l2: while (y > 0) {
       y := y-1;
    }
```

$$\rho_1 : pc = \ell_1 \wedge pc' = \ell_2$$
$$\rho_1 : pc = \ell_2 \wedge pc' = \ell_2 \wedge y > 0 \wedge y' = y - 1$$

- unbounded non-determinism at line l1 (for $pc = \ell_1$)
- termination of ANY-Y cannot be proved with ranking functions ranging over the set of natural numbers
- initial rank must be at least the ordinal ω

example program BUBBLE (nested loop)

```
l1: while (x => 0) {
          y := 1;
l2:       while (y < x) {
             y := y+1;
          }
          x := x-1;
      }
```

$$\rho_1 : pc = \ell_1 \wedge pc' = \ell_2 \wedge x \geq 0 \wedge x' = x \wedge y' = 1$$
$$\rho_2 : pc = \ell_2 \wedge pc' = \ell_2 \wedge y < x \wedge x' = x \wedge y' = y + 1$$
$$\rho_3 : pc = \ell_2 \wedge pc' = \ell_1 \wedge y \geq x \wedge x' = x - 1 \wedge y' = y$$

- *lexicographic* ranking function $\langle x, x - y \rangle$
- ordered pair of two ranking functions, x and $x - y$

program CHOICE

```
l: while (x > 0 && y > 0) {
     if (read_int()) {
       (x, y) := (x-1, x);
     } else {
       (x, y) := (y-2, x+1);
     }
   }
```

$$\rho_1 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x - 1 \wedge y' = x$$
$$\rho_2 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = y - 2 \wedge y' = x + 1$$

- simultaneous-update statements in loop body
- non-determinstic choice
- ranking function?

example program without simple ranking function

```
l: while (x > 0 && y > 0) {
     if (read_int()) {
       x := x-1;
       y := read_int();
     } else {
       y := y-1;
     }
   }
```

$$\rho_1 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x - 1$$
$$\rho_2 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x \wedge y' = y - 1$$

- non-deterministic choice
- decrement x, forget value of y or
  don't change x, decrement y

transition invariant

given a program P with transition relation R_P,

a binary relation T is a a *transition invariant*
if it contains the transitive closure of the transition relation:

$$R_P^+ \subseteq T$$

- compare with *invariant*
- inductiveness

disjunctively well-founded relation

a relation T is *disjunctively well-founded*
if it is a finite union of well-founded relations:

$$T = T_1 \cup \cdots \cup T_n$$

- in general, union of well-founded relations is itself not well-founded

proof rule for termination

a program P is terminating
if and only if
there exists a disjunctively well-founded transition invariant T for P

T must satisfy two conditions,

transition invariant:

$$R_P^+ \subseteq T$$

disjunctively well-founded:

$$T = T_1 \cup \cdots \cup T_n$$

where $T_1, \ldots, T_n$ well-founded

completeness of proof rule

- "only if" $(\Rightarrow)$
- program P is terminating *implies* there exists a disjunctively well-founded transition invariant for P
- trivial:
- if P is terminating, then both R_P and R_P^+ are well-founded
- choose $n = 1$ and $T_1 = R_P^+$

soundness of proof rule

- "If" ($\Leftarrow$):
- a program P is terminating *if* there exists a disjunctively well-founded transition invariant for P
- contraposition:
  if
  $R_P^+ \subseteq T$,
  $T = T_1 \cup \cdots \cup T_n$, and
  P is *not* terminating,
  then
  at least one of $T_1, \ldots, T_n$ is not well-founded

assume $R_P^+ \subseteq T$, $T = T_1 \cup \cdots \cup T_n$, P non-terminating

- there exists an infinite computation of P:

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$$

- each pair (s_i, s_j) lies in one of T_1, ..., T_n
- one of T_1, ..., T_n (say, T_k) contains infinitely many pairs (s_i, s_j)
- contradiction if we obtain an infinite chain in T_k (since T_k is a well-founded relation)
- but ... in general, those pairs (s_i, s_j) do not form a chain

Ramsey's theorem

every infinite complete graph that is colored with finitely many colors contains a monochrome infinite *complete subgraph*

assume $R_P^+ \subseteq T$, $T = T_1 \cup \cdots \cup T_n$, P non-terminating

- there exists an infinite computation of P:

$$s_0 \to s_1 \to s_2 \to \dots$$

- take infinite complete graph formed by s_i's
- edge = pair (s_i, s_j) in R_P^+, i.e., in one of $T_1, \dots, T_n$
- edges can be colored by n different colors
- exists monochrome infinite complete subgraph
- all edges in subgraph are colored by, say, T_k
- infinite complete subgraph has an infinite path
- obtain infinite chain in T_k
- contradicition since T_k is a well-founded relation

assume $R_P^+ \subseteq T$, $T = T_1 \cup \cdots \cup T_n$, P non-terminating

- there exists an infinite computation of P:

$$s_0 \to s_1 \to s_2 \to \dots$$

- let a choice function f satisfy

$$f(k,\ell) \in \{\ T_i \mid (s_k, s_\ell) \in T_i\ \}$$

  for $k, \ell \in \mathbb{N}$ with $k < \ell$
- condition $R_P^+ \subseteq T_1 \cup \cdots \cup T_n$ implies that f exists (but does not define it uniquely)
- define equivalence relation $\simeq$ on f's domain by

$$(k,\ell) \simeq (k',\ell') \text{ if and only if } f(k,\ell) = f(k',\ell')$$

- relation $\simeq$ is of finite index since the set of T_i's is finite
- by Ramsey's Theorem there exists an infinite sequence of natural numbers $k_1 < k_2 < \dots$ and fixed $m, n \in \mathbb{N}$ such that

$$(k_i, k_{i+1}) \simeq (m, n) \qquad \text{for all } i \in \mathbb{N}.$$

example program: ANY-Y

```
l1: y := read_int();
l2: while (y > 0) {
       y := y-1;
    }
```

$$\rho_1 : pc = \ell_1 \wedge pc' = \ell_2$$
$$\rho_1 : pc = \ell_2 \wedge pc' = \ell_2 \wedge y > 0 \wedge y' = y - 1$$

$$T_1 : pc = \ell_1 \wedge pc' = \ell_2$$
$$T_2 : y > 0 \wedge y' < y$$

example program BUBBLE (nested loop)

```
l1: while (x => 0) {
          y := 1;
l2:       while (y < x) {
             y := y+1;
          }
          x := x-1;
    }
```

$$\rho_1 : pc = \ell_1 \wedge pc' = \ell_2 \wedge x \geq 0 \wedge x' = x \wedge y' = 1$$

$$\rho_2 : pc = \ell_2 \wedge pc' = \ell_2 \wedge y < x \wedge x' = x \wedge y' = y + 1$$

$$\rho_3 : pc = \ell_2 \wedge pc' = \ell_1 \wedge y \geq x \wedge x' = x - 1 \wedge y' = y$$

$$T_1 : pc = \ell_1 \wedge pc' = \ell_2$$

$$T_2 : pc = \ell_2 \wedge pc' = \ell_1$$

$$T_3 : x \geq 0 \wedge x' < x$$

$$T_4 : x - y > 0 \wedge x' - y' < x - y$$

program CHOICE

```
l: while (x > 0 && y > 0) {
     if (read_int()) {
       (x, y) := (x-1, x);
     } else {
       (x, y) := (y-2, x+1);
     }
   }
```

$$\rho_1 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x - 1 \wedge y' = x$$

$$\rho_2 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = y - 2 \wedge y' = x + 1$$

$$T_1 : x > 0 \wedge x' < x$$

$$T_2 : y > 0 \wedge y' < y$$

$$T_3 : x + y > 0 \wedge x' + y' < x + y$$

example program without simple ranking function

```
l: while (x > 0 && y > 0) {
     if (read_int()) {
       x := x-1;
       y := read_int();
     } else {
       y := y-1;
     }
   }
```

$\rho_1 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x - 1$
$\rho_2 : pc = pc' = \ell \wedge x > 0 \wedge y > 0 \wedge x' = x \wedge y' = y - 1$

$T_1 : x \geq 0 \wedge x' < x$
$T_2 : y > 0 \wedge y' < y$

prove termination of program P

- compute a disjunctively well-founded superset of the transitive closure of the transition relation of the program P, i.e.,
- construct a finite number of well-founded relations $T_1, \ldots, T_n$ whose union covers R_P^+
- show that the inclusion $R_P^+ \subseteq T_1 \cup \cdots \cup T_n$ holds
- show that each of the relations $T_1, \ldots, T_n$ is indeed well-founded

prove termination in 3 steps

1. find a finite number of relations $T_1, \ldots, T_n$
2. show that the inclusion $R_P^+ \subseteq T_1 \cup \cdots \cup T_n$ holds
3. show that each relation $T_1, \ldots, T_n$ is well-founded

it is possible to execute the 3 steps in a different order

conclusion

- disjunctively well-founded transition invariants: basis of a new proof rule for program termination
- (next) transition predicate abstraction: basis of automation of proof rule
- new class of automatic methods for proving program termination
  - combine multiple ranking functions for reasoning about termination of complex program fragments
  - rely on abstraction techniques to make this reasoning efficient