

On proving liveness properties of programs

Alexey Gotsman

University of Cambridge

joint work with Byron Cook,
Andreas Podelski, and Andrey Rybalchenko

BCTCS'06, 6 April 2006

State-of-the-art

Systems	Model checking	Abstraction	Properties
SLAM, BLAST	Symbolic	Automatic	Safety
JPF, Bandera	Explicit	User-defined	Full LTL
?	Symbolic	Automatic	Full LTL

Formal setting

- ▶ Program P (transition system)
- ▶ Property φ – LTL
- ▶ Fairness requirements

Does program P satisfy property φ under the given fairness requirements?

Fairness requirements

- ▶ \mathcal{C} – set of compassion requirements $\langle p, q \rangle$
 - ▶ a.k.a. strong fairness
- ▶ Computation σ is fair wrt compassion requirement $\langle p, q \rangle$ if
 - ▶ either there exist finitely many p -states in σ
 - ▶ or there exist infinitely many q -states in σ
- ▶ Intuition: if you request something sufficiently many times (p), then eventually you will receive it (q)
- ▶ Computation is fair if it is fair wrt all the compassion requirements

From liveness to fair termination

- ▶ A program is fair terminating if it has no infinite fair computation
- ▶ Property $\varphi \Rightarrow$ Streett automaton $A_{\neg\varphi}$
- ▶ Program $P_{\neg\varphi} = P || A_{\neg\varphi}$
- ▶ Compassion requirements on $P_{\neg\varphi}$:
 - ▶ requirements on P
 - ▶ requirements from the accepting condition of $A_{\neg\varphi}$
- ▶ The program P satisfies the property φ under the fairness requirements iff the program $P_{\neg\varphi}$ is fair terminating

Fair computation segments

- ▶ σ – computation segment
 - ▶ a finite fragment of a computation
- ▶ σ is fair wrt the compassion requirement $\langle p, q \rangle$ if it
 - ▶ either does not visit any p -states
 - ▶ or visits some q -state
- ▶ σ is fair if it is fair wrt every compassion requirement
- ▶ Intuition: repeating a fair computation segment gives a fair computation

Proving fair termination

- ▶ Binary reachability relation for fair termination:

$$\mathcal{R} = \{ \langle s_1, s_n \rangle \mid \exists \text{ fair computation segment } \sigma = s_1, \dots, s_n \}$$

- ▶ Relation T is disjunctively well-founded iff it is a finite union of well-founded relations.

Theorem (Pnueli, Podelski, Rybalchenko, 2005)

The program P is fair terminating iff there exists a disjunctively well-founded relation T such that $\mathcal{R} \subseteq T$

We will construct the relation T by counterexample-guided refinement

Fair computation paths

- ▶ π – path
 - ▶ a finite sequence of program statements
- ▶ Each computation has the corresponding path
- ▶ π is fair if some computation segment σ obtained by executing statements in π is fair
- ▶ Path relation of a path $\pi = \tau_1 \dots \tau_n$: $\rho_\pi = \rho_{\tau_1} \circ \dots \circ \rho_{\tau_n}$
- ▶ We will try to cover ρ_π for each π by a disjointively well-founded relation

Construction of fair termination arguments

input

Program P with fairness assumptions

begin

$T := \emptyset$

repeat

if exists path π such that $\text{fair}(\pi)$ and $\rho_\pi \not\subseteq T$ **then**

if $\text{well-founded}(\rho_\pi)$ **then**

$T := T \cup \{\rho_\pi\}$

else

return "Counterexample path π "

else

return "Fair termination argument T "

end.

Program transformation (1)

Solution: Transform program P to program \hat{P} such that the set of reachable states of \hat{P} corresponds the relation \mathcal{R}

Variables of the program \hat{P} :

- ▶ Variables of the program P : v_1, \dots, v_n, pc
 - ▶ record the current state (the end of the current computation segment)
- ▶ Pre-variables: $'v_1, \dots, 'v_n, 'pc$
 - ▶ record the beginning of the current computation segment
 - ▶ initially equal to their counterparts in P
- ▶ Variables for keeping track of fairness: $in_p_1, \dots, in_p_m, in_q_1, \dots, in_q_m$
 - ▶ $in_p_i = 1$ iff there was a p -state on the current computation segment
 - ▶ $in_q_i = 1$ iff there was a q -state on the current computation segment

Program transformation (2)

```
L: stmt;
```



```
L: fair = ((!p1 && !in_p1) || q1 || in_q1) &&  
      ...  
      ((!pm && !in_pm) || qm || in_qm);  
assert(!fair || T(pc, 'pc, vi, 'vi));  
if (nondet()) {  
    'vi = vi;           /* for each i ∈ 1..n */  
    'pc = L;  
    in_pi = 0;         /* for each i ∈ 1..m */  
    in_qi = 0;         /* for each i ∈ 1..m */  
}  
if (pi) in_pi = 1;   /* for each i ∈ 1..m */  
if (qi) in_qi = 1;   /* for each i ∈ 1..m */  
stmt;
```

Fair termination argument validation via safety

- ▶ Error-state is unreachable in program \hat{P} iff T is a valid fair termination argument
- ▶ Can apply a safety checker (SLAM, BLAST) to verify this
- ▶ If the check fails, the counterexample produced by model checker is the required path π

Experimental results

- ▶ Prototype implementation for C programs
- ▶ SLAM as a safety checker
- ▶ Podelski&Rybalchenko's algorithm for synthesis linear of ranking functions
- ▶ Property:
 $\mathbf{G}(\text{KeEnterCriticalRegion} \Rightarrow \mathbf{F} \text{KeLeaveCriticalRegion})$

Driver	Time (seconds)	Lines of code	True bugs	False bugs
1	15	1K	1	0
2	314	7K	0	0
3	2344	15K	0	3
4	3122	20K	1	0
1R	16	1K	0	0
4R	3217	20K	0	0