Liveness Checking as Safety Checking for Infinite State Spaces

Viktor Schuppan\textsuperscript{1}, Armin Biere\textsuperscript{2}

\textsuperscript{1}Computer Systems Institute, ETH Zürich
\textsuperscript{2}Institute for Formal Models and Verification, JKU Linz

\url{http://www.inf.ethz.ch/~schuppan/}

INFINITY’05, August 27, 2005, San Francisco, USA
Liveness vs. Safety: Finite State Systems

[Biere, Artho, Schuppan, 2002; Schuppan, Biere, 2004/2005]

Transform

system $K + \omega$-reg. property $\phi$

into

system $K^S +$ safety property $\phi^S$

such that

$K \models \phi \iff K^S \models \phi^S$

Benefits:

- Selected examples: exponential speed-up
- Shortest counterexamples (competitive with BMC)
- More tools/optimizations
- Q & d liveness algorithms
- Fewer liveness proofs
Contents

1. Introduction
2. Finite State Systems
3. Regular Model Checking
4. Pushdown Systems
5. Timed Automata
6. Conclusions
Finite State Case — Example

(Buggy) traffic light

(Negation of) specification: \( ! G F g \)

Product automaton

Counterexample: \( (r,1) (y,1) (g,1) (r,!g) (y,!g) \) \( \omega \)
Finite State Case — Example transformation

1. Nondeterministically guess loop start, save state

2. Find fair state in loop

3. Find second occurrence of saved state, close loop

<table>
<thead>
<tr>
<th>s</th>
<th>(r,1)</th>
<th>(y,1)</th>
<th>(g,1)</th>
<th>(r,!g)</th>
<th>(y,!g)</th>
<th>(r,!g)</th>
<th>(y,!g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy of s</td>
<td>^s₀</td>
<td>^s₀</td>
<td>^s₀</td>
<td>(r,!g)</td>
<td>(y,!g)</td>
<td>(r,!g)</td>
<td>(y,!g)</td>
</tr>
<tr>
<td>lasso</td>
<td>st</td>
<td>st</td>
<td>st</td>
<td>lb</td>
<td>lb</td>
<td>lc</td>
<td>lc</td>
</tr>
<tr>
<td>fair</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(can stop here!)
Finite State Case — Formal Definition

Let
- \( K = (S, T, I, L, F = \{F_0\}) \) be a fair finite Kripke structure,
- \( \hat{s}_0 \in S \) arbitrary but fixed.

Then \( K^S = (S^S, T^S, I^S, L^S, F^S) \) is defined as:

\[
S^S = S \times S \times \{st, lb, lc\} \times \mathbb{B}
\]

\[
I^S = \{(s_0, \hat{s}_0, st, 0) \mid s_0 \in I\} \cup \{(s_0, s_0, lb, f) \mid s_0 \in I \land (f \rightarrow s_0 \in F_0)\}
\]

\[
T^S = \{((s, \hat{s}, lo, f), (s', \hat{s}', lo', f')) \mid (s, s') \in T \land
((lo = st \land lo' = st) \land \neg f \land \neg f')\}
\]

\[
L^S((s^S)) = L(s), \text{ where } s^S = (s, \hat{s}, lo, f)
\]

\[
F^S = \emptyset
\]

\( K \) has reachable fair loop \( \iff \) \( K^S \) has reachable state \( s^S \) w. \( lo(s^S) = lc \)

© 2005 V. Schuppan – Computer Systems Institute, ETH Zürich.
Finite State Case — Complexity

- Loop closed
- Loop body, fair
- Loop body, not fair
- Stem

|S| branches, no changing between branches

\[
\begin{align*}
|S^S| &= O(|S|^2) \\
|T^S| &= O(|S| \cdot |T|) \\
r^S, d^S &= O(d) \\
|(T^S)^*| &= O(|S| \cdot |T^*|)
\end{align*}
\]
Regular Model Checking:

- Initial configurations: finite automaton on finite words
- Transition relation: finite transducer on finite words length-preserving ⇒ lasso-shaped counterexamples

Example: Token Passing:

Initial configurations

Transition relation
Problem: finite automaton can’t store unbounded words

Solution:
- Use pairs of characters instead of character: first is original, second is saved component
- Prefix with position on lasso

Initial configurations:

```
start on stem:
don’t save config.
(t,−)
(n,−)

start on loop body:
save config.
(t,t)
(n,n)

start on stem:
don’t save config.
(t,−)
(n,−)
```

Initial configurations:
Transition relation:

- Remain in stem, loop body or loop closed:
  - \(((n,\hat{a}),(n,\hat{a}))\)
  - \(((t,\hat{a}),(n,\hat{a}))\)
  - \(((n,\hat{a}),(t,\hat{a}))\)
  - \(((n,\hat{a}),(n,\hat{a}))\)
  - \(((t,t),(t,t))\)
  - \(((t,n),(n,n))\)
  - \(((n,t),(t,t))\)
  - \(((n,n),(n,n))\)

- Save config:
  - Switch from stem to loop body:
    - \((st,st)\) v \((lb,lb)\) v \((lc,lc)\)

- Close loop:
  - Switch from loop body to loop closed:
    - \(((n,\sim),(n,n))\)
    - \(((t,\sim),(n,n))\)
    - \(((n,\sim),(t,t))\)
    - \(((n,\sim),(n,n))\)
Bouajjani et al. show that **bounded local depth** is sufficient for termination of their computation of the transitive closure.

Assume, the original system has bounded local depth $k$. The transformation **preserves boundedness**: 

\[
\begin{array}{cccc}
\text{st} & \text{st} & \text{lb} & \text{lb} \\
{a_{0,0}} & {a_{1,0}} & {a_{k-1,0}} & {a_{k+1,0}} \\
{a_{0,1}} & {a_{1,1}} & {a_{k-1,1}} & {a_{k+1,1}} \\
\ldots & \ldots & \ldots & \ldots \\
{a_{0,n}} & {a_{1,n}} & {a_{k-1,n}} & {a_{k+1,n}} \\
\end{array}
\]

\[k + 1 + k + 1 + k \leq 3k + 2\]
Pushdown Systems — Repeatable Heads 1

[Bojajani, Esparza, Maler, 1997]

head
repeatable head

(control state, top symbol)

1. matching heads
2. sufficient stack height

head
repeatable head

(control state, top symbol)

1. matching heads
2. sufficient stack height

stack
(top symbol)

stack grows

control state

s t u v w x y z u w z u

© 2005 V. Schuppan – Computer Systems Institute, ETH Zürich.
Pushdown Systems — Repeatable Heads 2

[Bouajjani, Esparza, Maler, 1997]

- head
- repeatable head
  - (control state, top symbol)
  - 1. matching heads
  - 2. sufficient stack height
  - => can repeat infinitely often
  - => found in every infinite run

- stack
  - (top symbol)

- control state
  - s t u v w x y z u v w x y z
Pushdown Systems — Transformation

start loop: save head, mark stack height
on loop: check stack height, set error flag
loop closure: check head, error flag

stack

control state

control state (copy)

stack top (copy)

lasso

stack height error
The soonest second occurrence of a **repeatable head** does not guarantee shortest counterexamples.

That requires **repeatable prefixes**.
W.r.t. $\omega$-regular properties, timed automata can be abstracted to ordinary finite state automata [Alur, Dill, 1994].

Region construction can be expressed within formalism (with difference constraints).

⇒ technical, “can be done”. 
Related Work

**Infinite state systems:**


**Bouajjani, Esparza, Maler, 1997** is reduction to reachability. Requires separate computation of “bad states”.


**Finite state systems:**

**Burch, 1990** Reduction for timed trace structures. Requires user to come up with appropriate time constraint.

**Ultes-Nitsche, 2002** Satisfaction within fairness corresponds to some safety property. Not always desired semantics.
Conclusions

– Reduction usually is “pulling the algorithm into the model.”

– System size typically grows moderately

Future work

– Experimental evaluation.

– When does it not work?

– Use it to come up with liveness algorithm.