Liveness Checking as Safety Checking for Infinite State Spaces

Viktor Schuppan[1], Armin Biere[2]

[1]Computer Systems Institute, ETH Zürich

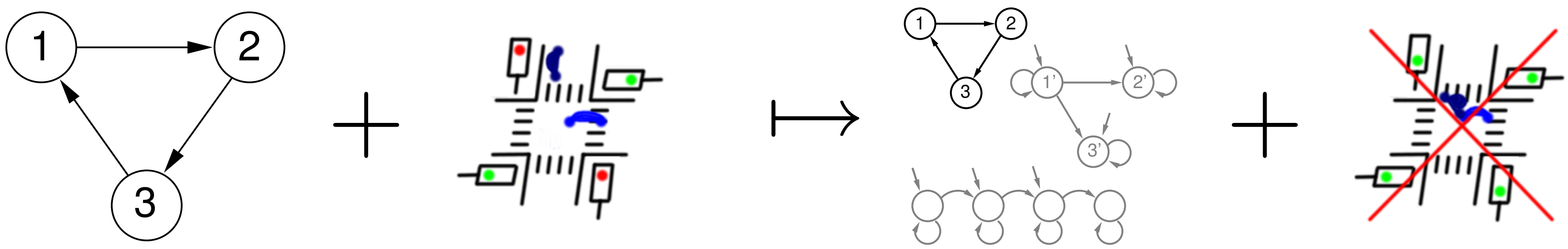
[2]Institute for Formal Models and Verification, JKU Linz

`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 + ω-reg. property ϕ

into

system $K^{\mathbf{S}}$ + safety property $\phi^{\mathbf{S}}$

such that

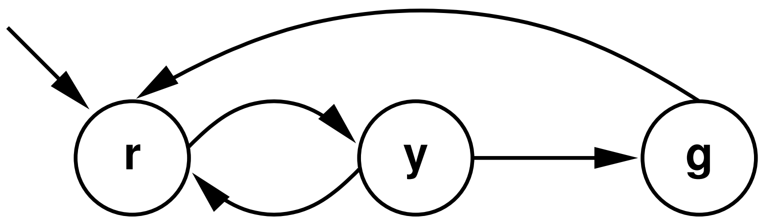
$$K \models \phi \Leftrightarrow K^{\mathbf{S}} \models \phi^{\mathbf{S}}$$

Benefits:

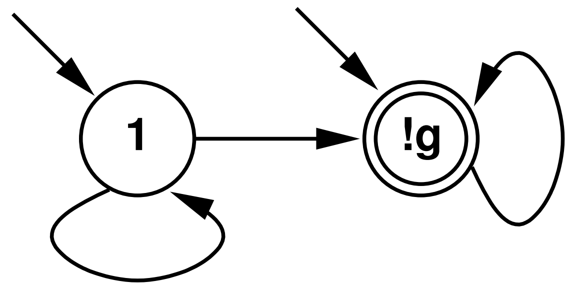
- Selected examples: exponential speed-up
- Shortest counterexamples (competitive with BMC)
- More tools/optimizations
- Q & d liveness algorithms
- Fewer liveness proofs

Contents

Finite State Case — Example

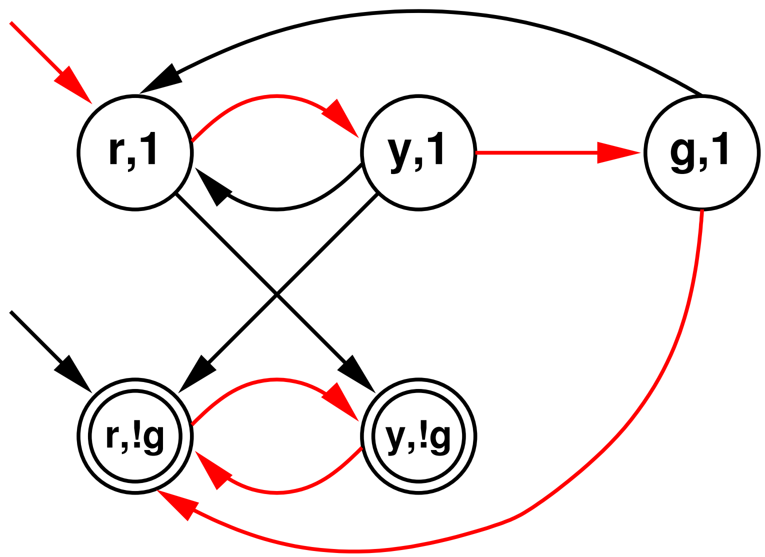


(Buggy) traffic light



(Negation of) specification:

! G F g



Product automaton

Counterexample: $(r,1)\ (y,1)\ (g,1)\ \Big((r,!g)\ (y,!g)\Big)^{\omega}$

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

can stop here!

s	(r,1)	(y,1)	(g,1)	(r,!g)	(y,!g)	(r,!g)	(y,!g)
copy of s	$\hat{s}_0$	$\hat{s}_0$	$\hat{s}_0$	(r,!g)	(r,!g)	(r,!g)	(r,!g)
lasso	st	st	st	lb	lb	lc	lc
fair	0	0	0	1	1	1	1

Finite State Case — Formal Definition

Let

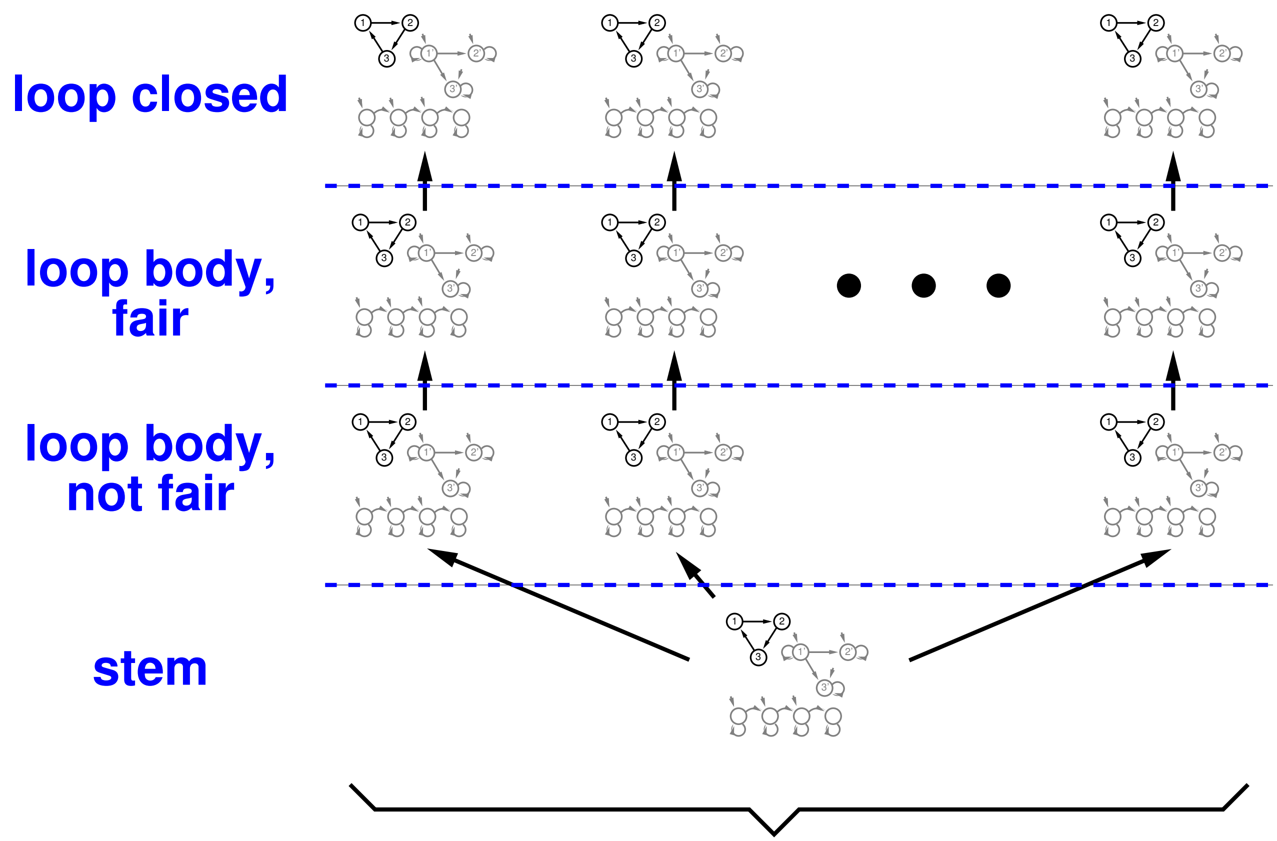
- $K = (S, T, I, L, F = \{F_0\})$ be a fair finite Kripke structure,
- $\hat{s}_0 \in S$ arbitrary but fixed.

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

$$
\begin{aligned}
S^{\mathbf{S}} &= S \times S \times \{st, lb, lc\} \times \mathbb{B} \\
I^{\mathbf{S}} &= \{(s_0, \hat{s}_0, st, 0) \mid s_0 \in I\} \cup \\
&\quad\ \{(s_0, s_0, lb, f) \mid s_0 \in I \wedge (f \rightarrow s_0 \in F_0)\} \\
T^{\mathbf{S}} &= \{((s, \hat{s}, lo, f), (s', \hat{s}', lo', f')) \mid (s, s') \in T \wedge \\
&\quad\ ((lo = st \wedge lo' = st \ \wedge\ \neg f \wedge \neg f' \quad \wedge \hat{s} = \hat{s}' = \hat{s}_0) \vee \\
&\quad\ \ (lo = st \wedge lo' = lb \ \wedge\ \neg f \wedge (f' \rightarrow s' \in F_0) \quad \wedge \hat{s} = \hat{s}_0 \wedge s' = \hat{s}') \vee \\
&\quad\ \ (lo = lb \wedge lo' = lb \ \wedge\ (f \rightarrow f') \wedge (f' \rightarrow f \vee s' \in F_0) \wedge \hat{s} = \hat{s}') \vee \\
&\quad\ \ (lo = lb \wedge lo' = lc \ \wedge\ f \wedge f' \quad \wedge \hat{s} = s' = \hat{s}') \vee \\
&\quad\ \ (lo = lc \wedge lo' = lc \ \wedge\ f \wedge f' \quad \wedge \hat{s} = \hat{s}'))\} \\
L^{\mathbf{S}}(s^{\mathbf{S}}) &= L(s), \text{ where } s^{\mathbf{S}} = (s, \hat{s}, lo, f) \\
F^{\mathbf{S}} &= \emptyset
\end{aligned}
$$

K has reachable fair loop $\Leftrightarrow$ $K^{\mathbf{S}}$ has reachable state $s^{\mathbf{S}}$ w. $lo(s^{\mathbf{S}}) = lc$

Finite State Case — Complexity



|S| branches,

no changing between branches

$$|S^{\mathbf{S}}| = \mathbf{O}(|S|^2) \qquad |T^{\mathbf{S}}| = \mathbf{O}(|S| \cdot |T|)$$

$$r^{\mathbf{S}}, d^{\mathbf{S}} = \mathbf{O}(d) \qquad |(T^{\mathbf{S}})^*| = \mathbf{O}(|S| \cdot |T^*|)$$

Regular Model Checking

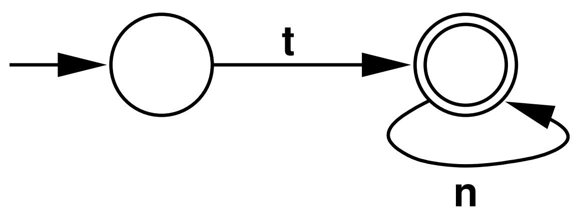
after [Bouajjani, Jonsson, Nilsson, Touili, 2000]

Regular model checking:

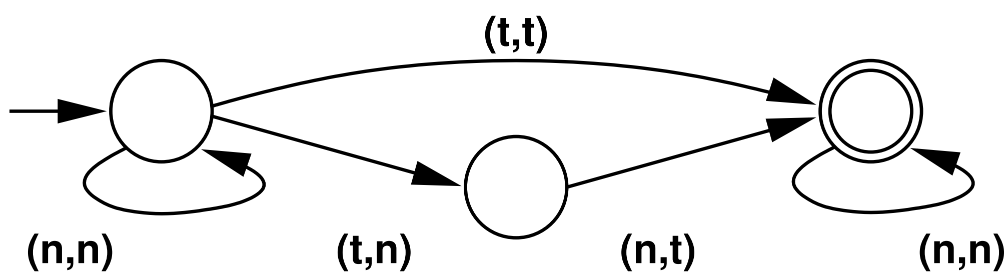
- Initial configurations: finite automaton on finite words
- Transition relation: finite transducer on finite words
  length-preserving ⇒ lasso-shaped counterexamples

Example: Token Passing:

Initial configurations



Transition relation

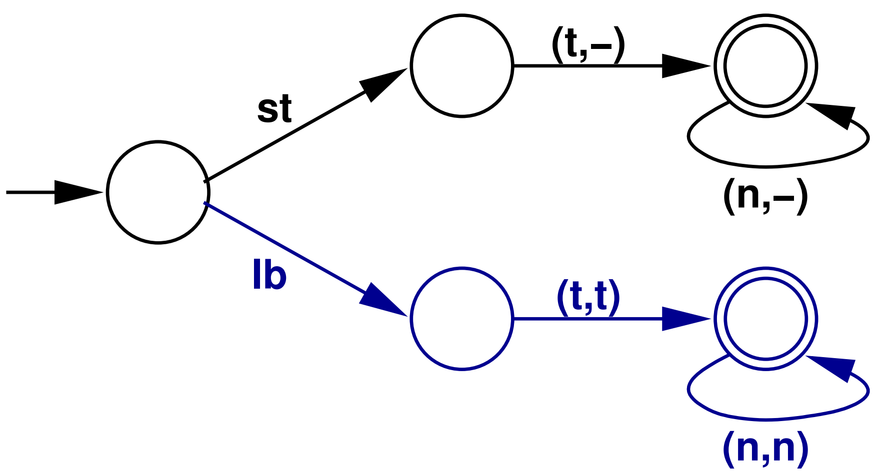


Problem: finite automaton can't store unbounded words

Solution:

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

Initial configurations:

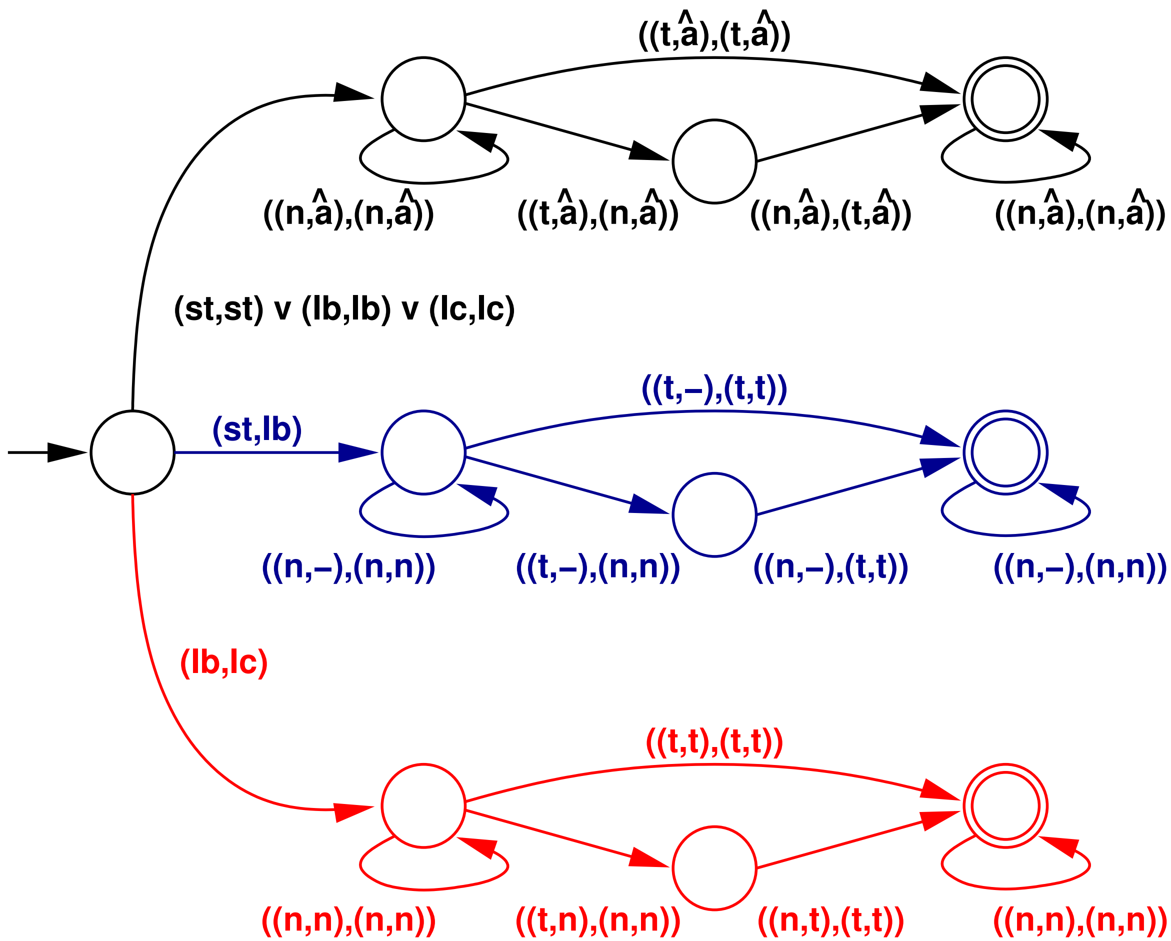


start on stem: don't save config.

start on loop body: save config.

Regular Model Checking — Transformation 2

Transition relation:



remain in stem, loop body or loop closed

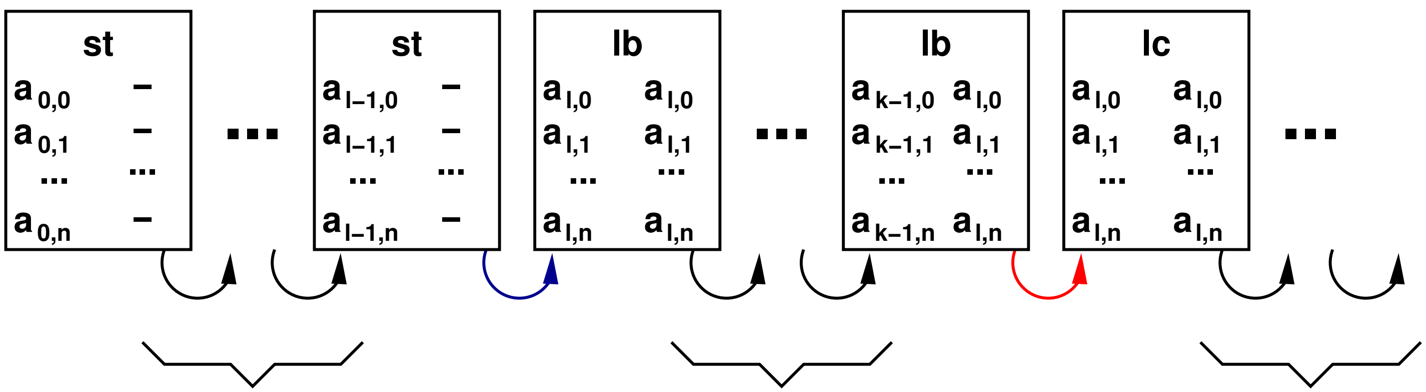
save config: switch from stem to loop body

close loop: switch from loop body to loop closed

Regular Model Checking — Bounded Local Depth

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:



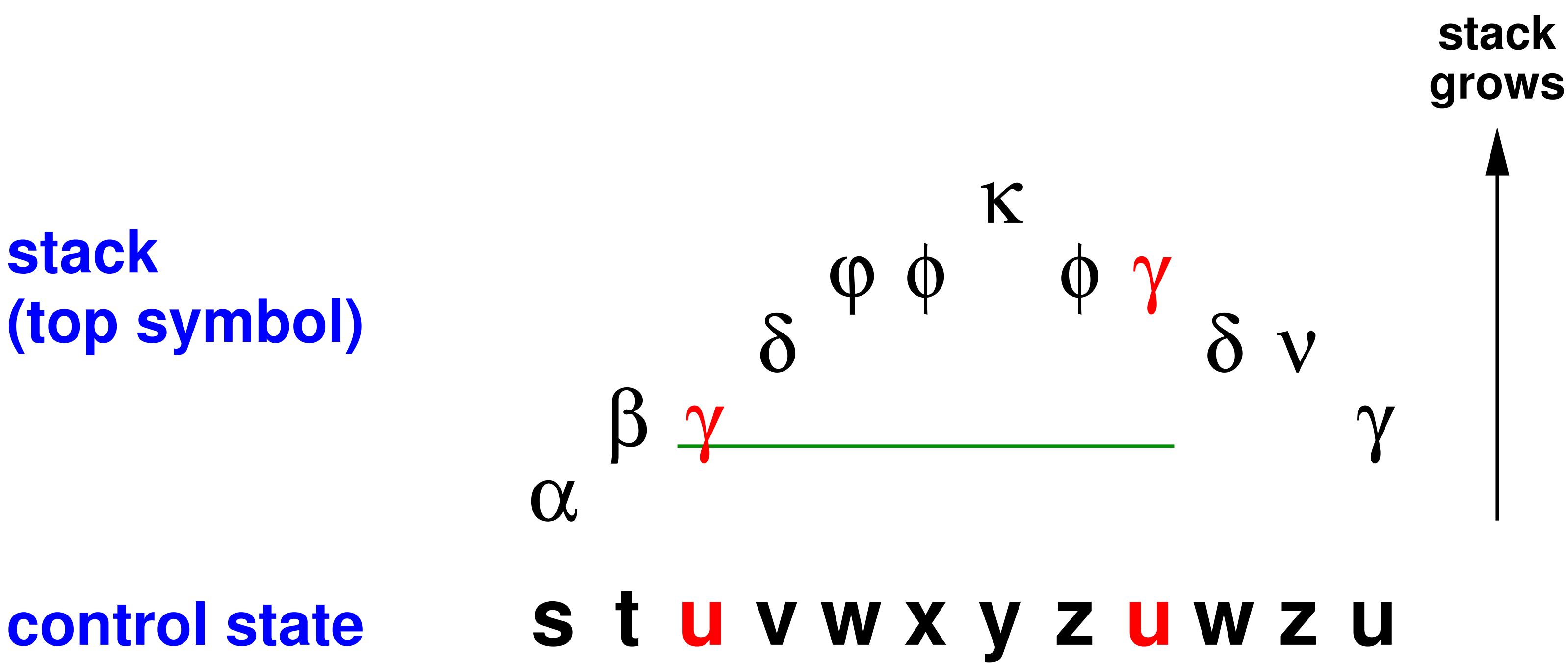
$$k + 1 + k + 1 + k \leqq 3k + 2$$

Pushdown Systems — Repeatable Heads 1

[Bouajjani, Esparza, Maler, 1997]

head (control state, top symbol)

repeatable head 1. matching heads
2. sufficient stack height



Pushdown Systems — Repeatable Heads 2

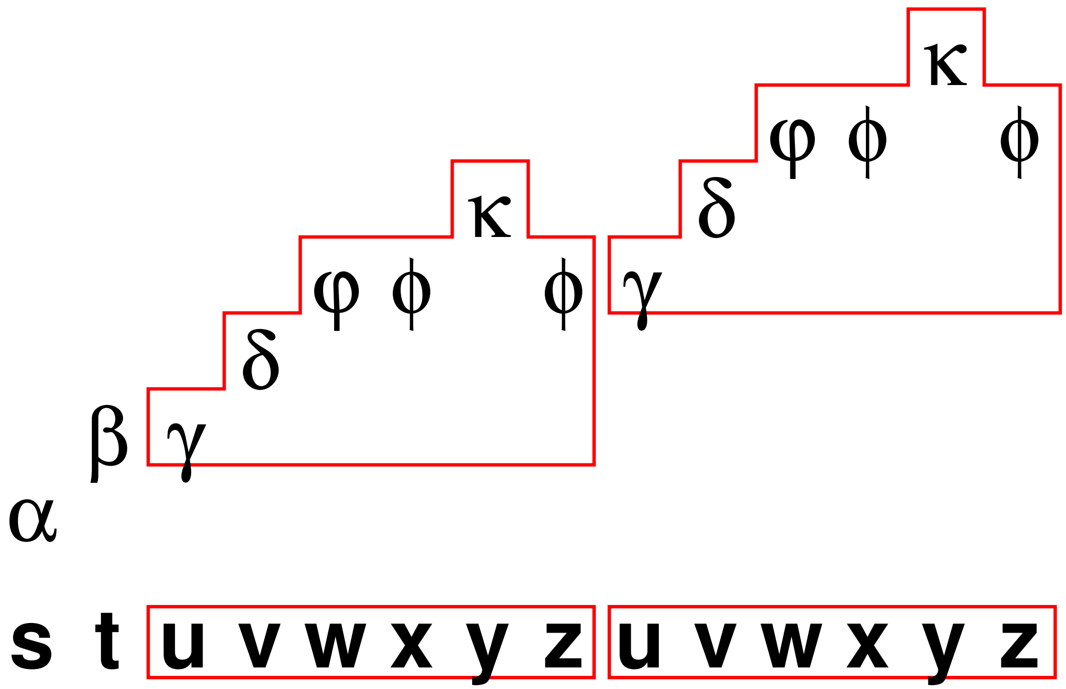
[Bouajjani, Esparza, Maler, 1997]

head — **(control state, top symbol)**

repeatable head
1. matching heads
2. sufficient stack height
=> can repeat infinitely often
=> found in every infinite run

stack
(top symbol)

control state



Pushdown Systems — Transformation

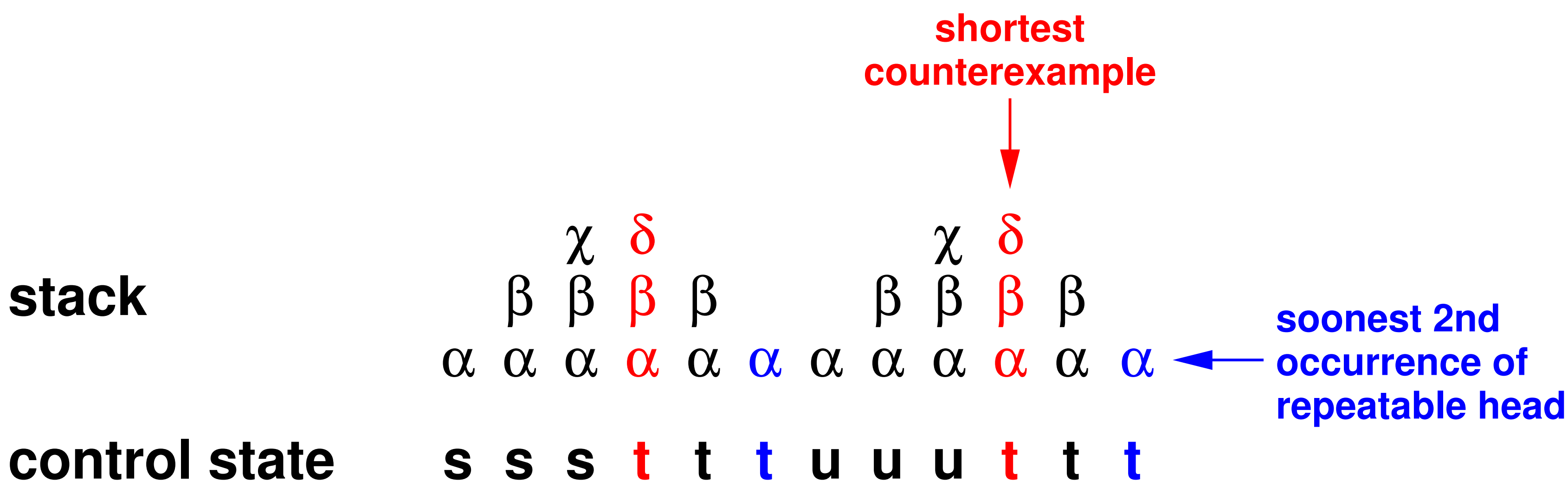
start loop: save head, mark stack height

on loop: check stack height, set error flag

loop closure: check head, error flag

stack						$\kappa,0$						
					$\varphi,0$	$\phi,0$	$\phi,0$	$\phi,0$	$\gamma,0$			
				$\delta,0$	$\delta,0$	$\delta,0$	$\delta,0$	$\delta,0$	$\delta,0$	$\delta,–$	$\nu,–$	
		$\beta,–$	$\gamma,–$	$\gamma,1$	$\gamma,1$	$\gamma,1$	$\gamma,1$	$\gamma,1$	$\gamma,1$	$\gamma,–$	$\gamma,–$	$\gamma,–$
	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$	$\alpha,–$
control state	s	t	u	v	w	x	y	z	u	w	z	u
control state (copy)	–	–	–	u	u	u	u	u	u	u	u	u
stack top (copy)	–	–	–	γ	γ	γ	γ	γ	γ	γ	γ	γ
lasso	st	st	st	lb	lb	lb	lb	lb	lb	lc	lc	lc
stack height error	–	–	–	0	0	0	0	0	0	0	0	0

Pushdown Systems — No Shortest Counterexamples



The soonest second occurrence of a repeatable head does not guarantee shortest counterexamples.

That requires repeatable prefixes.

W.r.t. ω-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:

Shilov, Yi, Eo, O, Choe, 2001/2005 Reduction of SOEPDL ($>$ 2M of C. Stirling) to reachability. Requires closure under Cartesian product and subset constructions. Doubly exponential.

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

Aceto, Bouyer, Burgueño, Larsen, 1998/2003 Power of reachability testing for timed automata.

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.