No Need for Liveness Model Checking Algorithms?

April, 2004

Armin Biere

(joint work with Viktor Schuppan and Cyrille Artho)

Computer Systems Institute
ETH Zürich

Beyond Safety

International Workshop

April 25–28, 2004, Ringberg Castle, Germany

Simple Temporal Properties

- simple **safety** properties:
  - LTL: $\mathbf{G}p$
  - CTL: $\mathbf{AG}p$

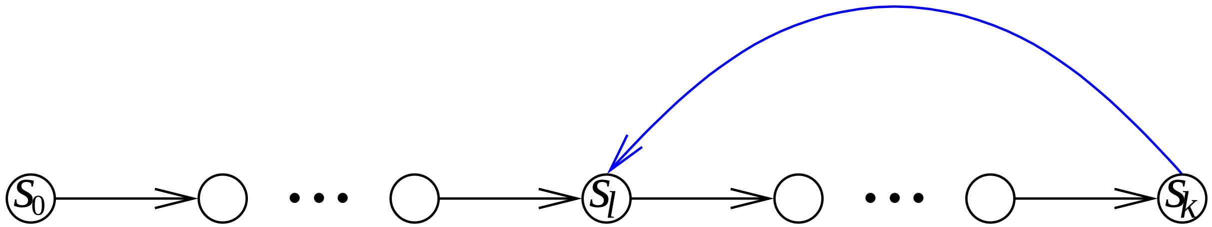
- simple **liveness** properties:
  - LTL: $\mathbf{F}p$
  - CTL: $\mathbf{AF}p$
  - plus fairness constraints (generalized Büchi Automata)

- full LTL can be translated to simple liveness + fairness

Completeness Thresholds in BMC

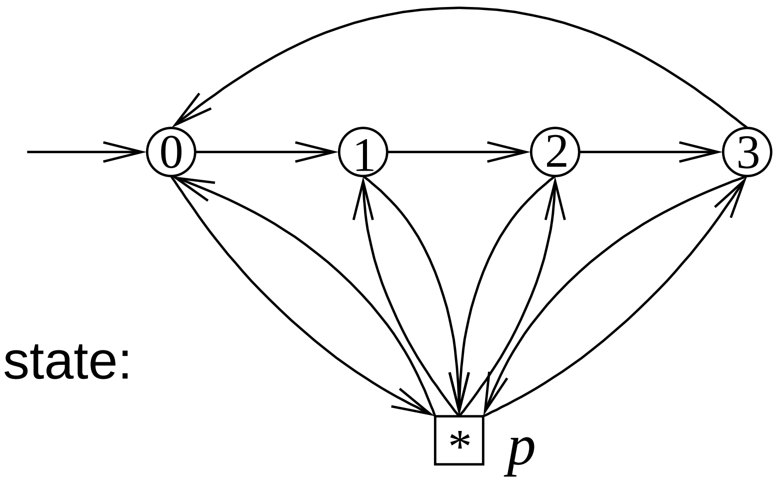
- counter examples to a **safety** property are finite traces
  - **radius** is the length of shortest initialized path to an arbitrary state
  - radius is a **completeness threshold** for (simple) safety properties
  - no longer potential counter example traces have to be checked

- every counter example trace to a liveness property is *lasso* shaped:



  - **diameter** is the length of the longest shortest path between two states
  - wrong: completeness threshold for liveness properties is $\boxed{\text{radius + diameter}}$

Completeness Threshold for Liveness Revised



- modulo n (here $n = 4$) counter with an explicit set state:

- radius and diameter both constant, but shortest counter example is of length n

- solution: use $\neg p$ predicated $\boxed{\text{radius + diameter}}$:
  - restrict Kripke structure to states in which $\neg p$ holds
  - calculate radius and diameter in restricted Kripke structure

Translating Liveness to Safety for Finite State Systems

- liveness is actually bounded liveness: $\mathbf{F}p \quad \equiv \quad \mathbf{F}_{\leq |S|}\, p$

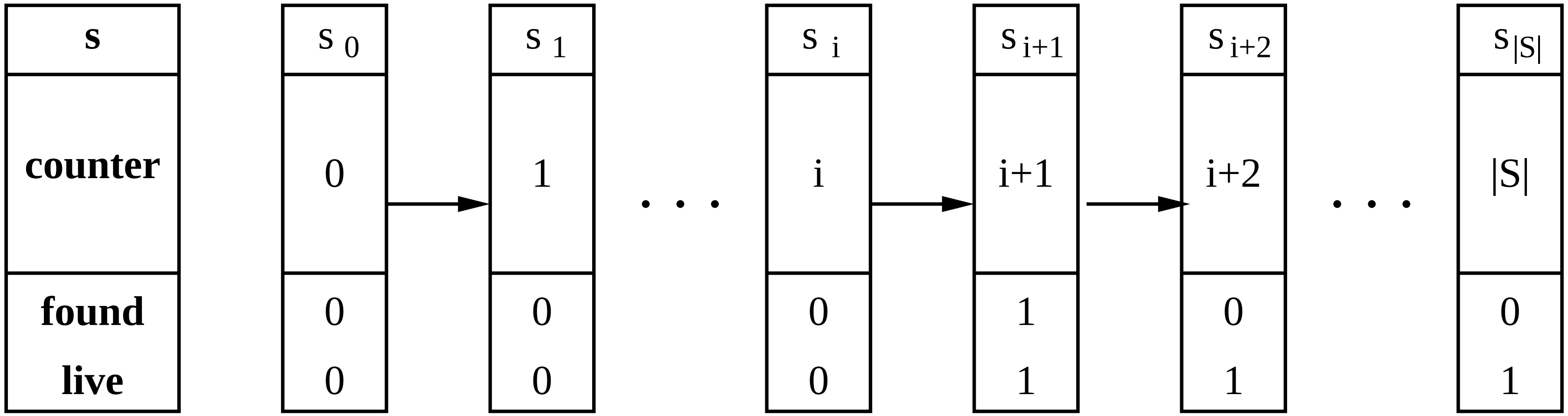
$$\mathbf{F}_{\leq |S|}\, p \quad \equiv \quad p \lor \mathbf{X}p \lor \ldots \lor \underbrace{\mathbf{X} \cdots \mathbf{X}}_{|S|} p$$

- brute force expansion needs exponential space for symbolic model checking

  (via the standard Büchi-Automata translation)

- counting translation requires twice the number of state bits

Counting Translation of Liveness to Safety



s = original state component

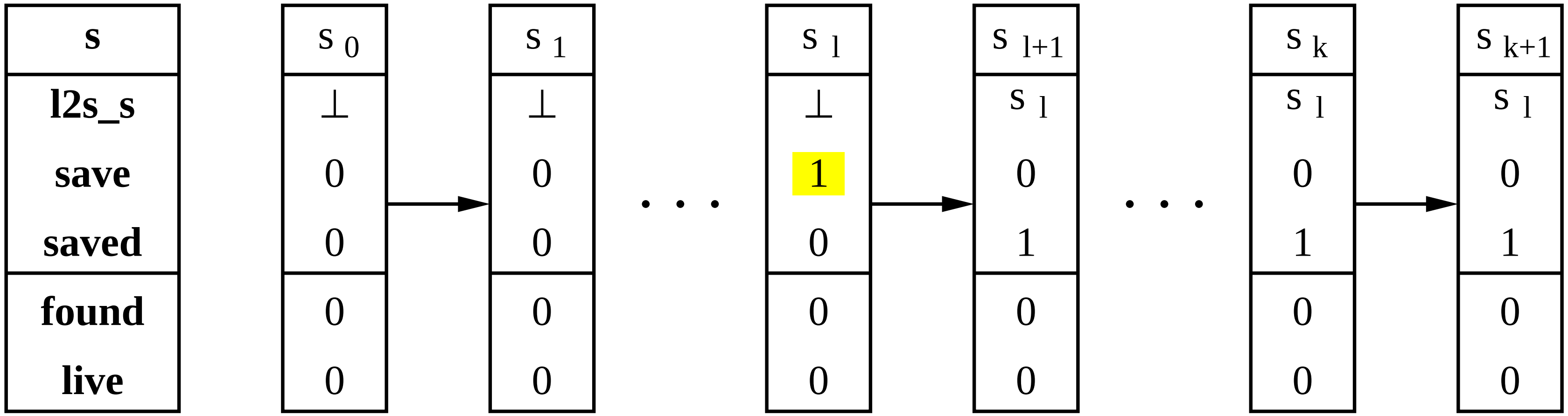
counter = $\lceil \log_2 |S| \rceil$-bit counter ($|S|$ = number of original states)

found = boolean flag: body of liveness property is satisfied

live = boolean state bit: **found** is or was true

$$\mathbf{G}\ (\mathbf{counter} = |S| \quad \rightarrow \quad \mathbf{live})$$

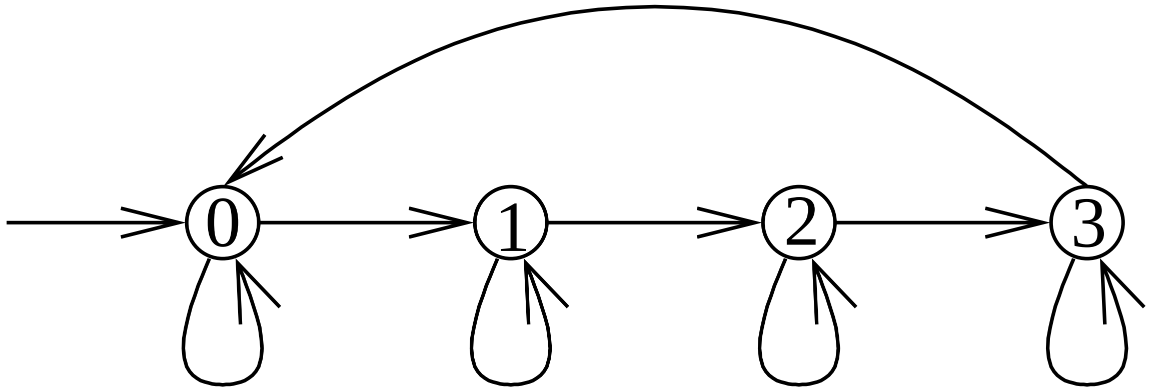
State Recording Translation of Liveness to Safety



s	=	original state component
l2s_s	=	copy of original state component to save a state
save	=	oracle (new primary input) to control when a state is saved
saved	=	boolean flag set to true when state has been saved
found	=	boolean flag: body of liveness property is satisfied
live	=	boolean state bit: **found** is or was true

$$\mathbf{G}\,(\mathbf{s} = \mathbf{l2s\_s} \quad \rightarrow \quad \mathbf{live})$$

Example: 2-Bit Counter with Self-Loops



$$\not\models \mathbf{F}(\mathtt{s} = 3)$$

```
DULE main
R
s: {0, 1, 2, 3};
SIGN
init(s) := 0;
next(s) := case
  s = 0: {1, s};
  s = 1: {2, s};
  s = 2: {3, s};
  s = 3: {0, s};
esac;

EC AF s = 3
```

```
MODULE main
VAR
  s: {0, 1, 2, 3};
ASSIGN
  init(s) := 0;
  next(s) := case
    s = 0: {1, s};
    s = 1: {2, s};
    s = 2: {3, s};
    s = 3: {0, s};
  esac;

-- loop detection part
VAR
  counter: 0..4;
ASSIGN
  init(counter) := 0;
  next(counter) := case
    counter < 4: counter + 1;
    1: counter;
  esac;

DEFINE
  looped := counter = 4;

-- property observing part
VAR live: boolean;
DEFINE found := s = 3;
ASSIGN
  init(live) := 0;
  next(live) := live | found;
SPEC AG (looped -> live)
```

```
MODULE main
VAR
  s: {0, 1, 2, 3};
ASSIGN
  init(s) := 0;
  next(s) := case
    s = 0: {1, s};
    s = 1: {2, s};
    s = 2: {3, s};
    s = 3: {0, s};
  esac;

-- loop detection part
VAR
  save: boolean;
  saved: boolean;
  l2s_s: {0, 1, 2, 3};
ASSIGN
  init(saved) := 0;
  next(saved) := on_loop;
  init(l2s_s) := s;
  next(l2s_s) := case
    save & !saved: s;
    1: l2s_s;
  esac;
DEFINE
  looped := saved & (s = l2s_
  on_loop := save | saved;

-- property observing part
VAR live: boolean;
DEFINE found := s = 3;
ASSIGN
  init(live) := 0;
  next(live) := live | found;
SPEC AG (looped -> live)
```

Comparison

- both translations are *complete*

- both translations double the number of state bits (in symbolic model checking)

- both translations may double the number of reachable states
  (really bad for explicit model checking)

- radius in counting translation may increase exponentially:
  (in symbolic model checking)

$$r^{\text{counting}} \quad \geq \quad |S|$$

- radius in state recording translation (optimizations possible):

$$r^{\text{recording}} \quad \leq \quad max\{r+2d+2, r_{\neg p}+d_{\neg p}+1\} \quad = \quad O(max\{d, d_{\neg p}\})$$

- counter examples found are indeed counter examples (correctness)

- conditions for completeness (modulo reachability):
  - if there is a counter example, then there is also a lasso shaped one
  - each trace visits only finite many states

- examples where it works (state variables $\in \mathbb{N}$):

$$
\begin{aligned}
I(s) &\equiv s \in \mathbb{N} \\
T(s,s') &\equiv s \geq s' \wedge \mathit{details}(s,s')
\end{aligned}
$$

$$
\begin{aligned}
I(s,b) &\equiv s = 0 \wedge b \in \mathbb{N} \\
T((s,b),(s',b')) &\equiv I(s',b') \vee \\
& s \leq s' \wedge b = b' \wedge \mathit{details}(s,s')
\end{aligned}
$$

Conclusion

- completeness threshold is different for liveness and safety
  - predicated diameter instead of ordinary diameter as bound
- finite states: efficient translation of liveness to safety
  - through state-recording translation
  - works in practice for symbolic model checking (e.g. with interpolation)
- infinite states: state recording workds for some examples
  - combination of state recording with fairness?
  - can we always (efficiently) translate liveness to safety?