

Berkeley, 29 September 2003

Counterexample-Guided Model Checking

Maria Sorea

sorea@csl.sri.com

<http://www.csl.sri.com/users/sorea/>

Joint work with N. Shankar

Computer Science Laboratory

SRI International

Menlo Park, CA

Model Checking

Given Kripke structure: $\mathcal{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$.

CTL formula: φ .

Set of initial states: $I \subseteq \mathbf{S}$.

Model checking problem: $\mathcal{M}, I \stackrel{?}{\models} \varphi$.

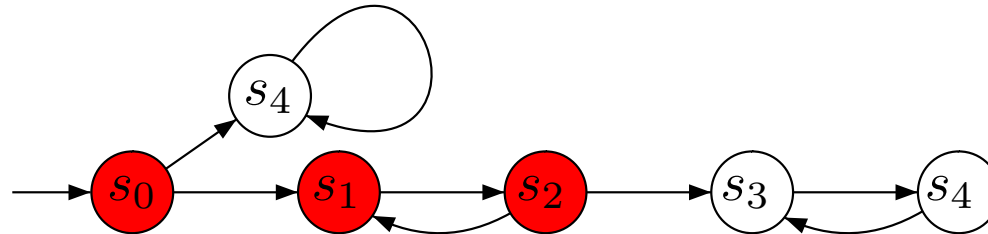
NO \longrightarrow Counterexample
YES \longrightarrow Witness

Duality: *counterexample* for **ACTL** is a *witness* for **ECTL**.

Contents

- State-of-the-art counterexamples for CTL: traces vs. trees.
- General form: Set-like counterexamples.
- CTL and naïve model-checking algorithm.
- Our approach – WMC algorithm.
- CTL fixpoint characterization vs. WMC.
- Independent characterization and construction of witnesses and counterexamples.
- Theoretical and experimental results.
- Applications: predicate abstraction for timed systems and controller synthesis.
- Conclusion.
- Related work.

Linear Witnesses / Counterexamples



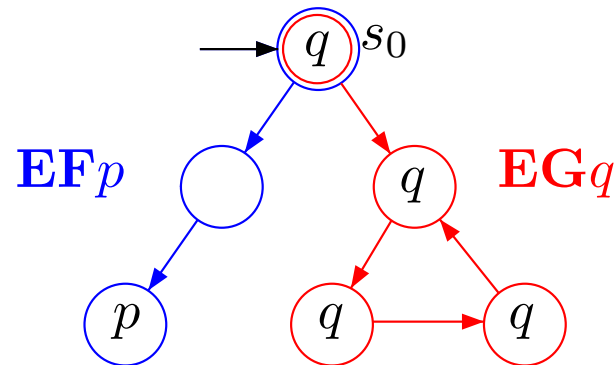
Assume: p holds in the *red* states.

Witness for $\mathcal{M}, s_0 \models \mathbf{EG}p$: $\langle s_0, s_1, s_2, s_1 \rangle$.

Counterexample for $\mathcal{M}, s_0 \models \mathbf{AF}\neg p$.

Only for $\text{ACTL} \cap \text{LTL}$ formulas.

Tree-Like Witnesses / Counterexamples



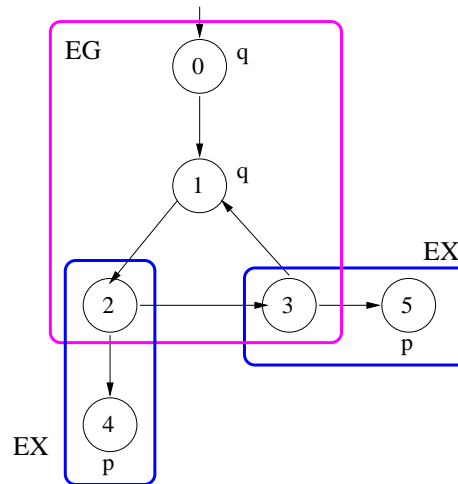
Witness for $\mathcal{M}, s_0 \models \mathbf{EF}p \wedge \mathbf{EG}q$

Counterexample for $\mathcal{M}, s_0 \models \mathbf{AG}\neg p \vee \mathbf{AF}\neg q$

- There is a finite path to a p state.
- There is an infinite path along which q is always true.

Only for ACTL.

$\mathbf{EG}(q \vee \mathbf{EX}p)$ – Clarke et al '02



Uses the last stage S_n of the model-checking computation according to the greatest fixpoint definition of $\mathbf{EG}(q \vee \mathbf{EX}p) = \nu Z.(q \vee \mathbf{EX}Z)$:

$$S_n = \llbracket \mathbf{EG}(q \vee \mathbf{EX}p) \rrbracket = \{0, 1, 2, 3\}.$$

Search for a cycle starting in 0: 1, 2, 3, 1.

Witness is sequence of path and loop descriptors

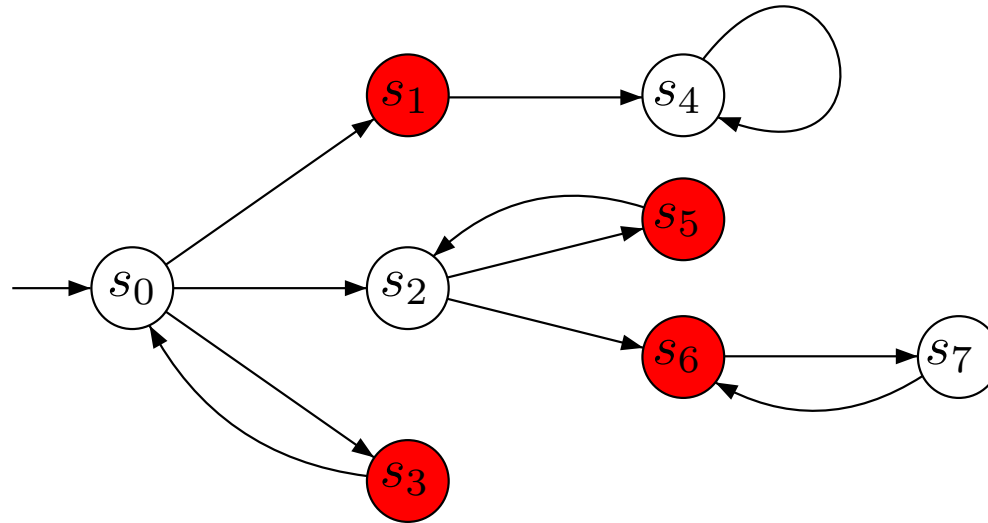
- $\langle 0, 1 \rangle$ path descriptor (prefix) for 'EG-loop'.
- $\langle 1, 2, 3, 1 \rangle$ loop descriptor for 'EG-loop'.
- $\langle 0 \rangle$, $\langle 1 \rangle$, $\langle 2, 4 \rangle$, $\langle 3, 5 \rangle$ witnesses for the subformula $q \vee \mathbf{EX}p$ in states 0, 1, 2, 3 of the main EG path.

Let's Summarize!

- Counterexample for ACTL formulas are traces (linear) or trees.
- Only formulas in $ACTL \cap LTL$ have linear counterexamples.
- Counterexamples are generated only after the model checker achieves a refutation.
- Entire state space (reachable states) has to be explored.
- When model checking succeeds there is no witness for justifying the relation between the model and the property.
- Counterexamples representation is explicit, not symbolic.

We want counterexamples for the full CTL logic.

Set-Like Witnesses / Counterexamples



Assume p holds in the *red* states.

Witness for $\mathcal{M}, s_0 \models \mathbf{AF}p$.

Counterexample for $\mathcal{M}, s_0 \models \mathbf{EG}\neg p$.

Trace over sets of states

$\langle \{s_1, s_3, s_5, s_6\}, \{s_1, s_2, s_3, s_5, s_6\}, \{s_0, s_1, s_2, s_3, s_5, s_6\} \rangle$

Symbolic representation!

Our Approach

Symbolic model checking.

Generation of witnesses if model checking succeeds, and of counterexamples, otherwise.

Witnesses and counterexamples are traces over sets of states, as well as single states.

Forward + Backward:

Every temporal formula is evaluated by means of a forward unfolding of the state space, starting in I , followed by a local fixpoint computation.

Partition of I in I^+ and I^- , such that $\mathcal{M}, I^+ \models \varphi$ and $\mathcal{M}, I^- \not\models \varphi$.

Symbolic witnesses and counterexamples are constructed from the results produced by the model checker.

CTL

AP set of atomic propositions, and $p \in \mathbf{AP}$

CTL in negation normal form:

$$\begin{aligned} \varphi \quad := \quad & p \mid \neg p \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \\ & \mathbf{EX}\varphi \mid \mathbf{EF}\varphi \mid \mathbf{EG}\varphi \mid \mathbf{E}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{E}[\varphi_1 \mathbf{R} \varphi_2] \mid \\ & \mathbf{AX}\varphi \mid \mathbf{AF}\varphi \mid \mathbf{AG}\varphi \mid \mathbf{A}[\varphi_1 \mathbf{U} \varphi_2] \mid \mathbf{A}[\varphi_1 \mathbf{R} \varphi_2] \end{aligned}$$

Semantics: $\mathcal{M} = \langle \mathbf{AP}, \mathbf{S}, \mathbf{N} \rangle$

Predicate transformers:

$$\textit{Postcondition} \quad \textit{post}(\mathbf{N})(S) := \{s' \in \mathbf{S} \mid \exists s \in S. s' \in \mathbf{N}(s)\}$$

$$\textit{Preimage} \quad \textit{pre}(\mathbf{N})(S) := \{s \in \mathbf{S} \mid \exists s' \in S. s' \in \mathbf{N}(s)\}$$

$$\textit{Precondition} \quad \widetilde{\textit{pre}}(\mathbf{N})(S) := \{s \in \mathbf{S} \mid \mathbf{N}(s) \subseteq S\}$$

CTL – Symbolic Model Checking

Fixpoint characterization: $\mathbf{AF}\varphi = \mu Z. \varphi \vee \mathbf{AX}Z$.

$[\mathbf{AF}\varphi]$ can be computed as a BDD given by $[\mathbf{AF}\varphi]_k$ for the least k such that

$$[\mathbf{AF}\varphi]_k = [\mathbf{AF}\varphi]_{k+1}$$

where

$$\begin{aligned} [\mathbf{AF}\varphi]_0 &= [\varphi] \\ [\mathbf{AF}\varphi]_{i+1} &= [\mathbf{AF}\varphi]_i \vee \widetilde{\text{pre}}(\mathbf{N})([\mathbf{AF}\varphi]_i) \end{aligned}$$

Does $\mathcal{M}, I \models \mathbf{AF}\varphi$, (I initial states)?

Is $[I] \wedge \neg[\mathbf{AF}\varphi]$ satisfiable?

- Yes \longrightarrow counterexample.
- No $\longrightarrow \mathcal{M}, I \models \mathbf{AF}\varphi$.

Our Approach – WMC

Bad states: $\neg[\mathbf{AF}\varphi] = \nu Z. [\neg\varphi] \wedge pre(\mathbf{N})(Z)$

Reachable states: $R = \mu Z. [I] \vee post(\mathbf{N})(Z)$

We have:

$$[I] \wedge \nu Z. [\neg\varphi] \wedge pre(\mathbf{N})(Z) \Leftrightarrow [I] \wedge \nu Z. [\neg\varphi] \wedge R \wedge pre(\mathbf{N})(Z)$$

Let: $R^- = \mu Z. ([\neg\varphi] \wedge [I]) \vee ([\neg\varphi] \wedge post(\mathbf{N})(Z)) \subseteq [\neg\varphi] \wedge R$

We have:

$$[I] \wedge \nu Z. [\neg\varphi] \wedge R \wedge pre(\mathbf{N})(Z) \Leftrightarrow [I] \wedge \underbrace{\nu Z. R^- \wedge pre(\mathbf{N})(Z)}_W$$

Thus: $[I] \wedge \neg[\mathbf{AF}\varphi] \equiv [I] \wedge W.$

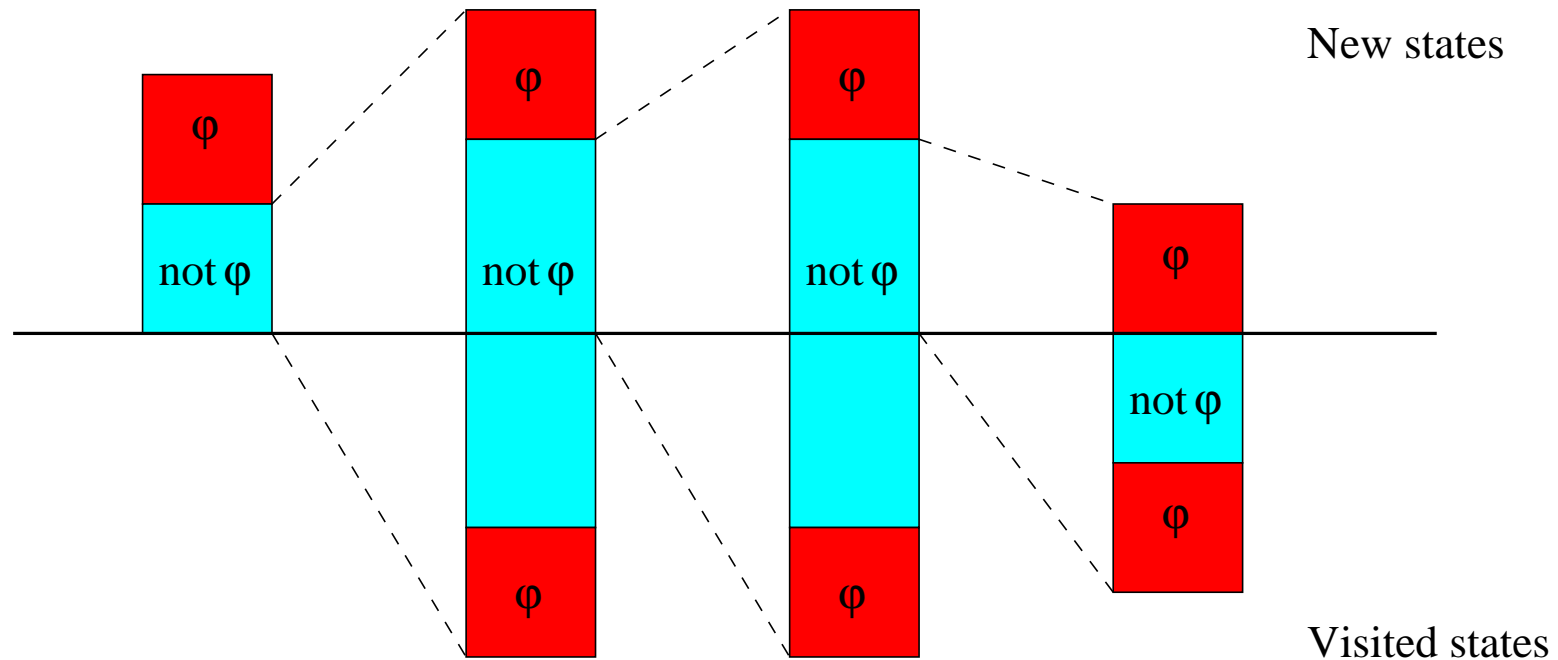
The wmc-algorithm consists of 2 phases:

1. *Forward iteration*: computes R^- using $post(\mathbf{N})$.
2. *Backward iteration*: computes W relative to R^- using $pre(\mathbf{N})$.

If $[I] \wedge W = \emptyset$ then *witness*, else *counterexample*.

$\mathbf{AF}\varphi$ – Forward Exploration

Computing $R^- = \mu Z. ([\neg\varphi] \wedge [I]) \vee ([\neg\varphi] \wedge \text{post}(\mathbf{N})(Z))$



WMC Algorithm

Input: $\mathbf{AF}\varphi, I, \mathbf{N}, V, V^-$

Output: Pair of lists $O = \langle [U_0, \dots, U_k], [W_0, \dots, W_m] \rangle$:

- $U_i = \langle S_i, B_i, O'_i, O''_i \rangle$.
 - $S_0 = I$.
 - $B_i \subseteq S_i$ states violating $\mathbf{AF}\varphi$.
 - O'_i corresponds to the subformula φ of $\mathbf{AF}\varphi$.
 - O''_i only for binary CTL operators.
- W_i set of states representing the stages in the fixpoint computation.

Algorithm for $\text{AF}\varphi$

$$\mathbf{AFMC}(\varphi, I, \mathbf{N}, V, V^-) = \tag{1}$$

$$\text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \tag{2}$$

$$\langle \vec{U}', \vec{W}' \rangle = O'; \tag{3}$$

$$I' = U'_0.B - V; \tag{4}$$

$$V = V \cup I; \tag{5}$$

$$V^- = V^- \cup I' \tag{6}$$

$$\text{in (if } I' = \emptyset \tag{7}$$

$$\text{then} \tag{8}$$

$$\text{let } \vec{W}^m = \vec{\nu}Z.V^- \wedge \text{pre}(\mathbf{N})(Z) \tag{9}$$

$$\text{in } \langle \langle [I, I \cap W_m, O', -], \vec{W} \rangle \tag{10}$$

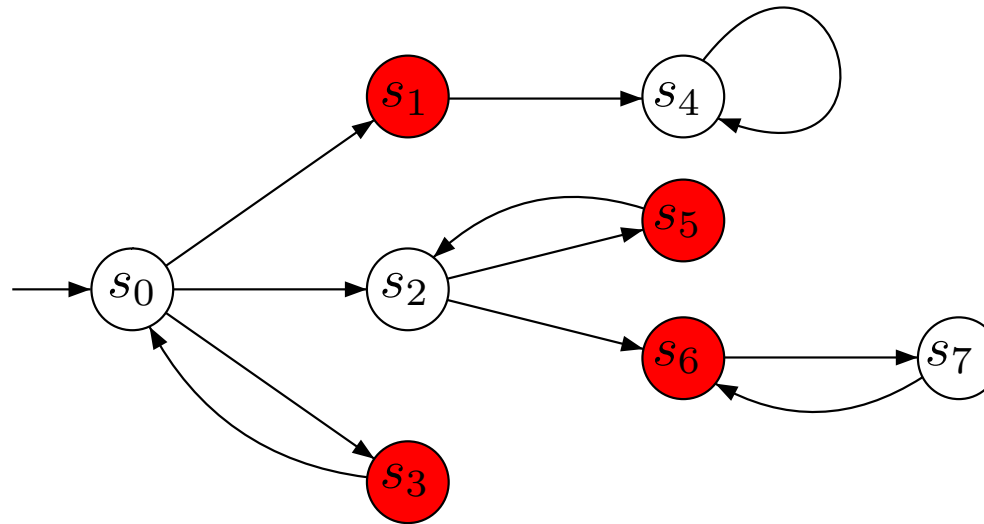
$$\text{else} \tag{11}$$

$$\text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{AFMC}(\varphi, \text{post}(\mathbf{N})(I'), \mathbf{N}, V, V^-) \tag{12}$$

$$\text{in } \langle \langle [I, (I \cap W_m), O', -]; \vec{U} \rangle, \vec{W} \rangle \tag{13}$$

$$\text{endif) } \tag{14}$$

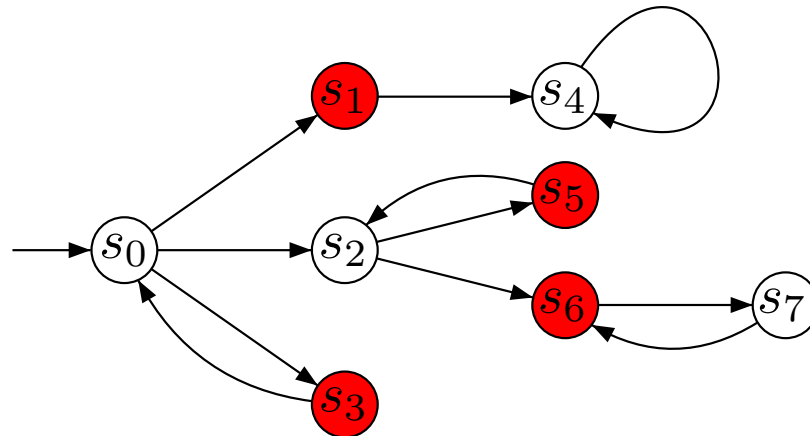
Example



Forward iteration:

Step	I	I'	V	V^-
0	$\{s_0\}$	$\{s_0\}$	$\{s_0\}$	$\{s_0\}$
1	$\{s_1, s_2, s_3\}$	$\{s_2\}$	$\{s_0, s_1, s_2, s_3\}$	$\{s_0, s_2\}$
2	$\{s_5, s_6\}$	\emptyset	$\{s_0, s_1, s_2, s_3, s_5, s_6\}$	$\{s_0, s_2\}$

Example (Cont.)



Local backward fixpoint computation: starts with $V^- = \{s_0, s_2\}$.

Bad states: $\vec{W} = \vec{v}Z.V^- \wedge pre(\mathbf{N})(Z) = \langle \{s_0, s_2\}, \{s_0\}, \emptyset \rangle$

Output: $O = \langle [\langle I_i, (I_i \cap W_m), O'_i, - \rangle_{i=0,1,2}], \vec{W} \rangle$
 $\langle [\langle \{s_0\}, \emptyset, O'_0, - \rangle, \langle \{s_1, s_2, s_3\}, \emptyset, O'_1, - \rangle, \langle \{s_5, s_6\}, \emptyset, O'_2, - \rangle], [\{s_0, s_2\}, \{s_0\}, \emptyset] \rangle$

Witness: sequence $\langle X_0, X_1, X_2 \rangle$ where $X_i = V - W_i$.

$\langle \{s_1, s_3, s_5, s_6\}, \{s_1, s_2, s_3, s_5, s_6\}, \{s_0, s_1, s_2, s_3, s_5, s_6\} \rangle$.

WMC vs. Fixpoint Characterization

Local fixpoint computation in WMC for $\mathbf{AF}\varphi$:

$$\vec{W} = pre(\mathbf{N})^\wedge(V^-) = \vec{\nu}Z. V^- \wedge pre(\mathbf{N})(Z)$$

Fixpoint definition of $\mathbf{AF}\varphi$:

$$\mathbf{AF}\varphi = \mu Z. \varphi \vee \mathbf{AX}Z$$

Negation: $\neg(\mathbf{AF}\varphi) = \nu Z. \neg\varphi \wedge \mathbf{EX}Z$

Characterization of Witnesses / Counterexamples

Witness $w \vdash \mathcal{M}, G \models \varphi$, respectively counterexample $c \vdash \mathcal{M}, C \not\models \varphi$.

Witness is a pair $\langle \vec{X}, \vec{w} \rangle$ with $|\vec{X}| = m + 1$, $|\vec{w}| = k + 1$.

$w \vdash \mathcal{M}, G \models \mathbf{AF}\varphi$ is characterized by:

1. $\exists G_0, \dots, G_k. w_i \vdash \mathcal{M}, G_i \models \varphi$, for $i \leq k$
2. $X_0 \subseteq \bigcup_{i=0}^k G_i$
3. $X_{i+1} \subseteq X_i \cup \widetilde{\text{pre}}(\mathbf{N})(X_i)$, for $i < m$
4. $G \subseteq X_m$

Characterization is independent of the model-checking algorithm.

A CTL proof can be extracted from these conditions.

CTL Proof Structure for $\mathbf{AF}\varphi$

- | | | |
|---|--------------------|--|
| 1. $\exists G_0, \dots, G_k. w_i \vdash \mathcal{M}, G_i \models \varphi$ | \rightsquigarrow | $\vdash G_i \Rightarrow \varphi, \forall 0 \leq i \leq k$ |
| 2. $X_0 = \bigcup_{i=0}^k G_i$ | \rightsquigarrow | $\vdash X_0 \Rightarrow G_0 \vee \dots \vee G_k$ |
| 3. $X_{i+1} \subseteq X_i \cup \widetilde{\text{pre}}(\mathbf{N})(X_i)$ | \rightsquigarrow | $\mathcal{M} \models X_{i+1} \Rightarrow X_i \vee \mathbf{AX}(X_i), \forall i < m$ |
| 4. $G \subseteq X_m$ | \rightsquigarrow | $\vdash G \Rightarrow X_m$ |
| | | $\vdash G \Rightarrow \mathbf{AF}\varphi$ |

Theoretical Results

Theorem 1 [Witness Validity]

If $w \vdash \mathcal{M}, G \models \varphi$ then $\mathcal{M}, G \models \varphi$, and if $c \vdash \mathcal{M}, C \not\models \varphi$, then $\mathcal{M}, C \not\models \varphi$.

Theorem 2 [Correctness]

Let O be $\mathbf{WMC}(\varphi, I, \mathbf{N})$, then there exist

- Disjoint sets G and C such that $I \subseteq G \cup C$,
- Witness w ,
- Counterexample c ,

such that $w \vdash \mathcal{M}, G \models \varphi$ and $c \vdash \mathcal{M}, C \not\models \varphi$.

Experimental Results – Synchronous Arbiter

Mutual exclusion property of a buggy version.

No. cells	WMC			MC		
	BDD size	msecs	No. iter	BDD size	msecs	No. iter
5	40	0	3	381	10	3
10	92	130	3	1041	260	4
15	142	80	3	1317	440	3
20	199	229	3	2297	1210	5
25	251	290	3	4838	3510	4
30	301	470	3	5533	2500	3
35	347	530	3	5657	1710	3
40	407	660	3	9976	11360	4
45	455	880	3	6801	4804	3
47	101	150	2	3338	1330	3

Experimental Results – Synchronous Arbiter

Mutual exclusion property of the correct version.

No. cells	WMC			MC		
	BDD size	msecs	No. iter	BDD size	msecs	No. iter
5	25	0	1	186	20	3
10	38	10	1	673	140	3
15	62	20	1	892	320	3
20	86	210	1	1265	250	4
25	102	40	1	2934	890	4
30	121	90	1	3086	1170	4
35	137	140	1	3406	1260	4
40	162	120	1	8164	4040	4
45	181	380	1	3737	2890	4
47	96	80	1	1647	540	3

Benefits and Problems

Witnesses and counterexamples generation for the entire CTL.

Fixpoint definitions of the CTL operators are not evaluated on the entire state space.

In general not necessary to compute the reachable states set.

The unfolding of the state space and the fixpoint computation are done according to the given temporal property.

A bunch of counterexamples can be constructed.

For nested formulas the resulting counterexamples are very large.

Combine WMC with fixpoint-based CTL model checking (similar to [Iwashita, Nakata, Hirose '96]): WMC for the outermost CTL operator, conventional model checking for nested operators.

Predicate Abstraction for Real-Time Systems

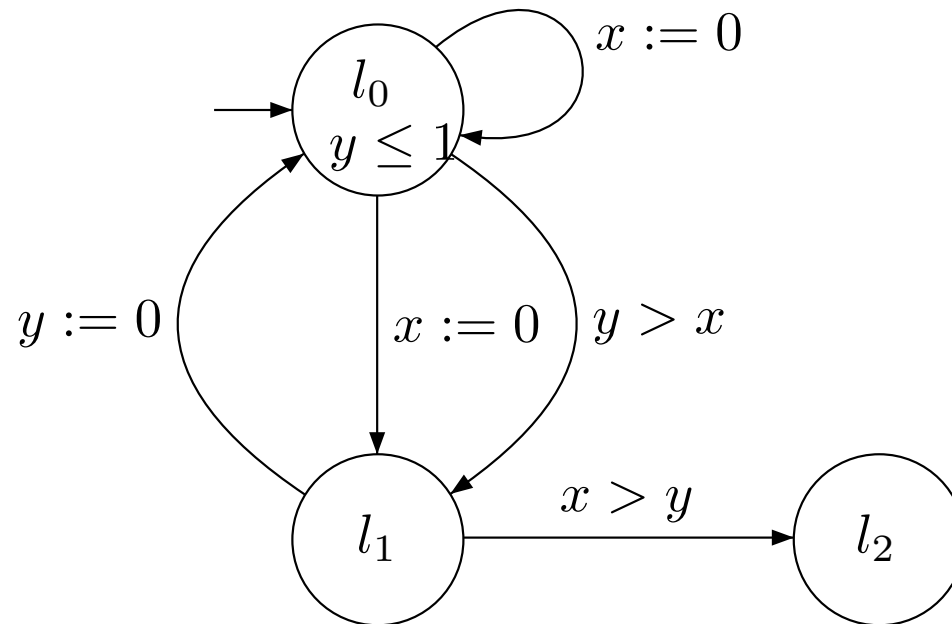
Given timed system \mathcal{S} , and TCTL property φ .

Algorithm for verifying safety and liveness properties of timed systems.

- Computing finite abstractions of timed automata and TCTL formulas using predicate abstraction.
- Finite-state model checking.
- Successive refinement of abstractions using counterexamples.

Real-Time System

Timed automaton: $\mathcal{S} = \langle L, l_0, P, C, E, I \rangle$



Semantics as transition system $\mathcal{M} = \langle L \times \mathcal{V}_C, \mathbf{AP}, \Rightarrow, (l_0, \nu_0) \rangle$.

Model-checking problem: $\mathcal{M}, (l_0, \nu_0) \models \varphi$.

Finite quotient for timed systems: *region construction*.

Abstraction Predicates

- Formula with the set of free variables in the clocks set C .
- Set of abstractions predicates $\Psi = \{\psi_0, \dots, \psi_{n-1}\}$.
- Examples: $\psi_0 \equiv x > 2$, $\psi_1 \equiv y \leq 3$, $\psi_2 \equiv x = y$.

Basis: set of abstraction predicates, expressive enough to distinguish between two clock regions.

Soundness and completeness: If Ψ basis then:

$$\mathcal{M} \models \varphi \text{ iff } \mathcal{M}_{\Psi}^A \models_A \varphi^A$$

Not practicable to compute *exact* abstraction!

Lazy Abstraction

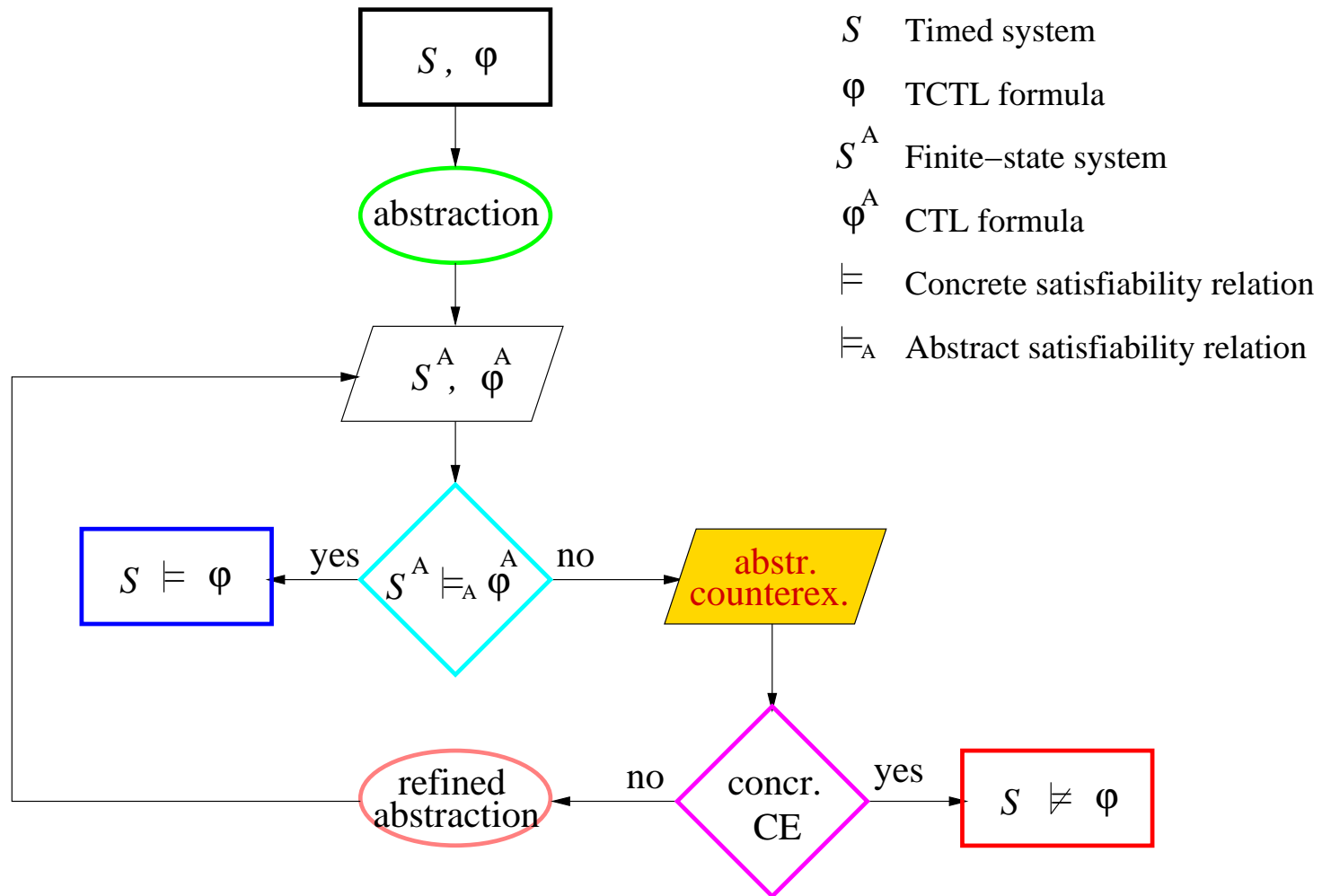
Compute only approximations of the exact abstraction, and refine them until the property is verified or refuted.

Previous work: [TPTS'02, ENTCS 65(6) 2002]

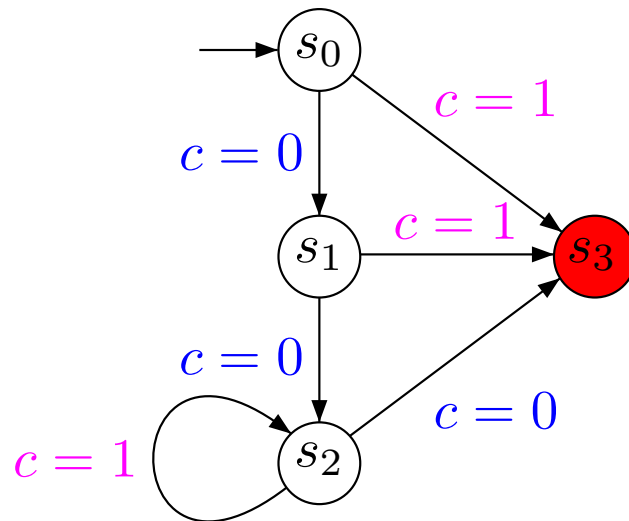
- (Untimed) μ -calculus.
- Only linear counterexamples.

Work in progress: TCTL and set-like counterexamples.

Lazy Abstraction for Real-Time Systems



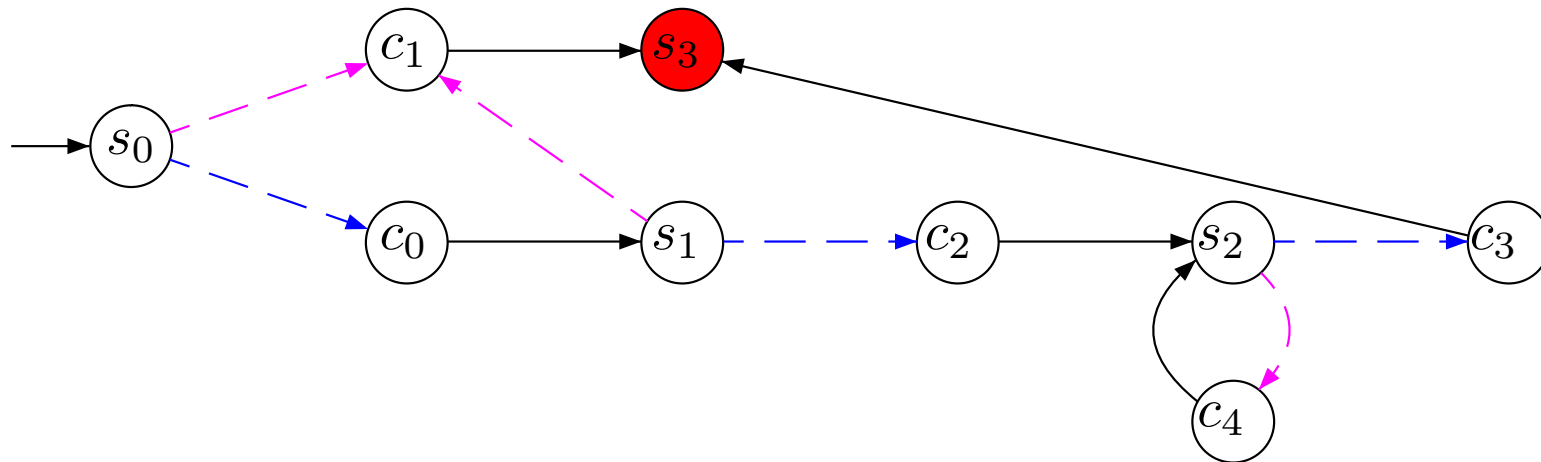
WMC and Controller Synthesis – Work in progress !!!



Given: system with controllable transitions.

Find: controller input s.t. $\mathbf{AG}p$ holds in s_0 .

WMC and Controller Synthesis (Cont.)

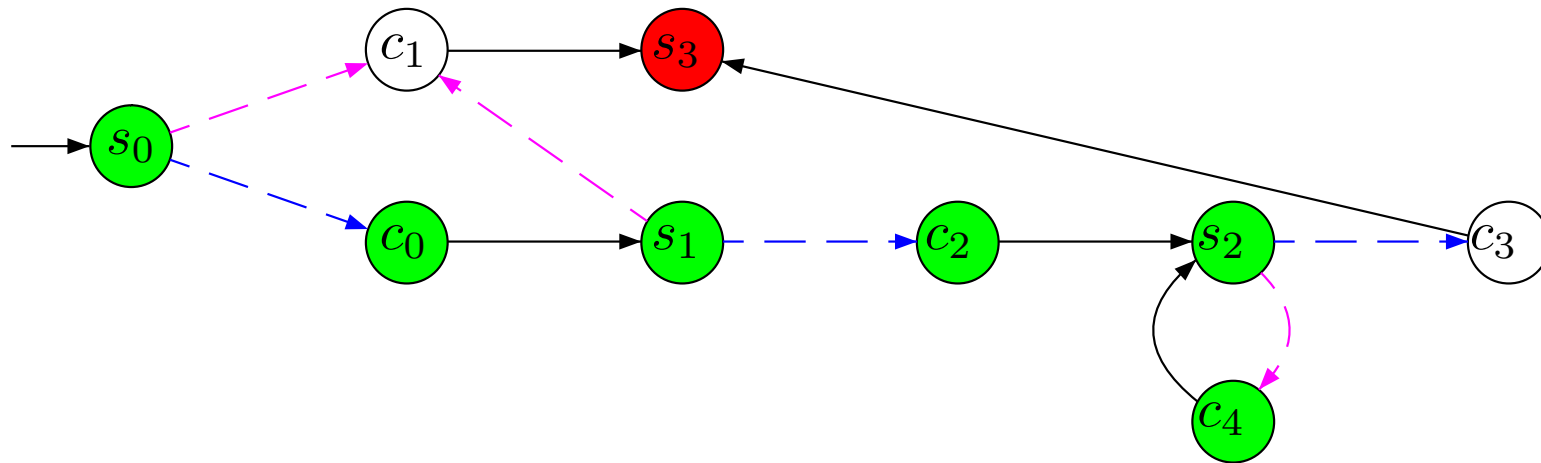


First: Build a symbolic transition graph going forward using both the controller and the system moves.

Second: Apply WMC with a *controlled precondition operation*.

- \widetilde{pre} on system states
- pre on controller states

WMC and Controller Synthesis (Cont.)



Witness: $\langle G_0, G_1 \rangle$ with

$$G_0 = \{s_0, s_1, s_2, c_0, c_1, c_2, c_3, c_4\}$$

$$G_1 = \{s_0, s_1, s_2, c_0, c_2, c_4\} \supseteq G \quad (\text{"good" states})$$

Omega regular expression: $\langle 0 \ 0 \ 1^\omega \rangle \dots$ controller input

Predicate Abstraction, Controller Synthesis, Timed Systems (Work in progress!)

Given: Timed automaton (TA), TCTL formula

Find: Controller input (signal, time) s.t. formula holds.

1. Abstract TA and TCTL formula.
2. Synthesize abstract controller input as omega regular expression.
3. Concretize this input, obtaining timed omega regular expression.
4. Build equivalent event-clock automaton [Henzinger, Raskin, Schobbens, '98].

Related Work – On Counterexample Generation

[Clarke, Grumberg, McMillan, and Zhao '95] Counterexample for fragments of ACTL and Fair ACTL. Finding fair counterexamples: NP complete. Strongly connected components: finding cycles.

[Clarke, Jha, Lu, Veith '02] Tree-like counterexamples for ACTL.

[Kick '96] Tree-like counterexamples for μ -calculus, the resulting trees are large and quite complicated.

[Shoham, Grumberg '03] Counterexamples for CTL are traces annotated with formulas.

Our approach uses a compact representation of witnesses / counterexamples as sequences over set of states. Good states have symbolic witnesses, bad states a symbolic counterexamples.

Related Work – Independently Checkable Witnesses

[Namjoshi '01] Certifying model checker: generates independently checkable witnesses for properties verified by a model checker.

[Peled, Pnueli, and Zuck '01] Deductive proofs for successfully model checked LTL formulas. Identifying the strongly connected components in the model checking tableau. Generating a proof for the absence of feasible paths.

[Gurfinkel and Chechik '03] Annotates model checker witnesses with proof steps. Generates proof obligations that can be independently verified with a theorem prover.

Our approach integrates the generation of witnesses and counterexamples into the model-checking algorithm.

Related Work – Forward Model Checking

[Iwashita, Nakata, and Hirose '96] CTL model-checking algorithm based on forward state traversal for a fragment of CTL.

[Henzinger, Kupferman, and Qadeer '98] Systematic study of the class of properties that can be checked by symbolic forward state traversal: ω -regular (linear-time) specifications.

[Biere, Clarke, and Zhu '99] Local tableau construction for LTL model checking based on sets of states and forward image computation. Explicit detection of strongly connected components in the tableau.

Our two-phase symbolic model checking algorithm for CTL consists of a forward traversal that identifies relevant reachable states followed by a backward traversal that partitions the initial set of states into good and bad states.

Conclusions

Simple local model-checking algorithm for CTL that constructs witnesses and counterexamples.

Constrained, forward unfolding of the state space starting from the initial states, followed by a fixpoint computation.

Symbolic witnesses and counterexamples are constructed from the results produced by the model checker.

The model-checking algorithm has been proved to produce valid witnesses and counterexamples.

Work in progress

Extension to CTL with fairness constraints, CTL*.

Generation of the weakest environment.

Additional Slides

WMC – AG φ

$$\mathbf{AGMC}(\varphi, I, \mathbf{N}, V, V^+) = \tag{1}$$

$$\text{let } O' = \mathbf{WMC}(\varphi, I, \mathbf{N}); \tag{2}$$

$$\langle \vec{U}', \vec{W}' \rangle = O'; \tag{3}$$

$$I' = I - (V \cup U'_0.B); \tag{4}$$

$$V = V \cup I; \tag{5}$$

$$V^+ = V^+ \cup I' \tag{6}$$

$$\text{in (if } I' = \emptyset \tag{7}$$

$$\text{then} \tag{8}$$

$$\text{let } \vec{W}^m = \widetilde{pre}^\wedge(\mathbf{N})(V^+) \quad \% \vec{v}Z.V^+ \wedge \widetilde{pre}(\mathbf{N})(Z) \tag{9}$$

$$\text{in } \langle \langle [I, I - W_m, O', -], \vec{W} \rangle \tag{10}$$

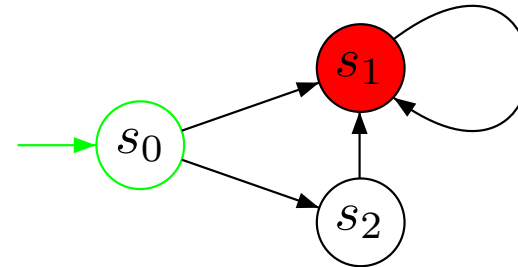
$$\text{else} \tag{11}$$

$$\text{let } \langle \vec{U}, \vec{W}^m \rangle = \mathbf{AGMC}(\varphi, post(\mathbf{N})(I'), \mathbf{N}, V, V^+) \tag{12}$$

$$\text{in } \langle \langle [I, I - W_m, O', -]; \vec{U} \rangle, \vec{W} \rangle \tag{13}$$

$$\text{endif) } \tag{14}$$

AF(AGp)



Step 0: $I = \{s_0\}$ $V = \emptyset$ $V^- = \emptyset$

$O[0]' = \mathbf{WMC}(\mathbf{AG}p, \{s_0\})$

Step 0: $I = \{s_0\}$ $V = \emptyset$ $V^+ = \emptyset$

$$O' = \mathbf{WMC}(p, \{s_0\}) = \langle \underbrace{[\langle \{s_0\}, \{s_0\}, -, - \rangle]}_{\vec{U}'}, \underbrace{\langle \emptyset \rangle}_{\vec{W}'} \rangle$$

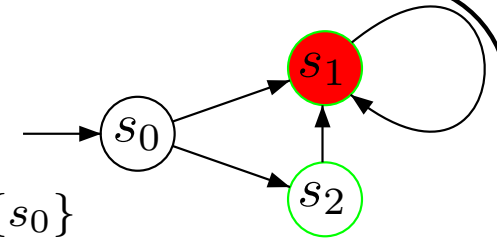
$$I' = \emptyset \quad V = \{s_0\} \quad V^+ = \emptyset$$

$$\vec{W} = V^+ = \emptyset$$

$$O'[0] \Leftarrow O = \langle \underbrace{[\langle \underbrace{\{s_0\}}_I, \underbrace{\{s_0\}}_{I-W}, -, - \rangle]}_{\vec{U}'}, \underbrace{\langle \emptyset \rangle}_{\vec{W}'} \rangle$$

$$I' = \underbrace{\{s_0\}}_{U'_0 \cdot B - V} \quad V = \{s_0\} \quad V^- = \{s_0\}$$

AF(**AG** p) (Cont.)



Step 1: $I = \text{post}(\mathbf{N})(I'[0]) = \{s_1, s_2\}$ $V = \{s_0\}$ $V^- = \{s_0\}$

$O[1]' = \mathbf{WMC}(\mathbf{AG}p, \{s_1, s_2\})$

Step 0: $I = \{s_1, s_2\}$ $V = \emptyset$ $V^+ = \emptyset$

$$O' = \mathbf{WMC}(p, \{s_1, s_2\}) = \langle \overbrace{[\langle \{s_1, s_2\}, \{s_2\}, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \emptyset \rangle}^{\vec{W}'} \rangle$$

$I' = \{s_1\}$ $V = \{s_1, s_2\}$ $V^+ = \{s_1\}$

Step 1: $I = \text{post}(\mathbf{N})(I') = \{s_1\}$ $V = \{s_1, s_2\}$ $V^+ = \{s_1\}$

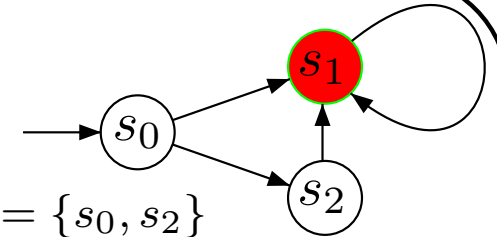
$$O' = \mathbf{WMC}(p, \{s_1\}) = \langle \overbrace{[\langle \{s_1\}, \emptyset, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \emptyset \rangle}^{\vec{W}'} \rangle$$

$I' = \emptyset$ $V = \{s_1, s_2\}$ $V^+ = \{s_1\}$ $\vec{W} = \langle \{s_1\} \rangle$

$$O'[1] \Leftarrow O = \langle \overbrace{[\langle \underbrace{\{s_1, s_2\}}_{I[0]}, \underbrace{\{s_2\}}_{I[0]-W[0]}, -, - \rangle; \langle \underbrace{\{s_1\}}_{I[1]}, \underbrace{\emptyset}_{I[1]-W[1]}, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \{s_1\} \rangle}^{\vec{W}'}} \rangle$$

$I' = \underbrace{\{s_2\}}_{U'_0.B-V}$ $V = \{s_0, s_1, s_2\}$ $V^- = \{s_0, s_2\}$

AF(**AG** p) (Cont.)



Step 2: $I = \text{post}(\mathbf{N})(I'[1]) = \{s_1\}$ $V = \{s_0, s_1, s_2\}$

$V^- = \{s_0, s_2\}$

$O[2]' = \mathbf{WMC}(\mathbf{AG}p, \{s_1\})$

Step 0: $I = \{s_1\}$ $V = \emptyset$ $V^+ = \emptyset$

$$O' = \mathbf{WMC}(p, \{s_1\}) = \langle \overbrace{[\langle \{s_1\}, \emptyset, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \emptyset \rangle}^{\vec{W}'} \rangle$$

$I' = \{s_1\}$ $V = \{s_1\}$ $V^+ = \{s_1\}$

Step 1: $I = \text{post}(\mathbf{N})(I') = \{s_1\}$ $V = \{s_1, s_2\}$ $V^+ = \{s_1\}$

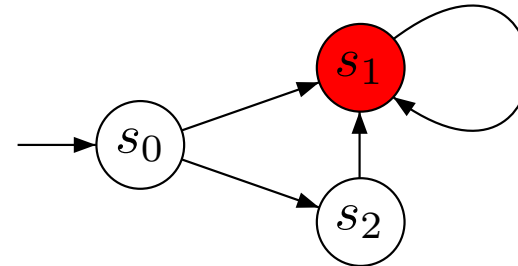
$$O' = \mathbf{WMC}(p, \{s_1\}) = \langle \overbrace{[\langle \{s_1\}, \emptyset, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \emptyset \rangle}^{\vec{W}'} \rangle$$

$I' = \emptyset$ $V = \{s_1\}$ $V^+ = \{s_1\}$ $\vec{W}' = \langle \{s_1\} \rangle$

$$O'[2] \Leftarrow O = \langle \overbrace{[\langle \underbrace{\{s_1\}}_{I[0]}, \underbrace{\emptyset}_{I[0]-W[0]}, -, - \rangle]}^{\vec{U}'}, \overbrace{\langle \underbrace{\{s_1\}}_{\vec{W}'} \rangle}^{\vec{W}'}}_{\substack{[\langle \underbrace{\{s_1\}}_{I[1]}, \underbrace{\emptyset}_{I[1]-W[1]}, -, - \rangle]}} \rangle$$

$I' = \underbrace{\emptyset}_{U'_0.B-V}$ $V = \{s_0, s_1, s_2\}$ $V^- = \{s_0, s_2\}$

AF(**AG** p) (Cont.)



Backward computation of \vec{W} relative to $V^- = \{s_0, s_2\}$.

$$\vec{W} = \vec{v}Z.V^- \wedge \text{pre}(\mathbf{N})(Z) = \langle \{s_0, s_1\}, \{s_0\}, \emptyset \rangle$$

Output:

$$O = \left\langle \left[\begin{array}{l} \langle \{s_0\}, \emptyset, O'[0], - \rangle; \\ \langle \{s_1, s_2\}, \emptyset, O'[1], - \rangle; \\ \langle \{s_1\}, \emptyset, O'[2], - \rangle; \\ \langle \{s_1\}, \emptyset, O'[3], - \rangle \end{array} \right], \quad [\{s_0, s_2\}, \{s_0\}, \emptyset] \right\rangle$$

Witness is a pair $\langle \vec{X}, \vec{w} \rangle$ with $|\vec{X}| = 3$ and $|\vec{w}| = 1$

$$\langle [\{s_1\}, \{s_1, s_2\}, \{s_0, s_1, s_2\}], [\{s_1\}] \rangle$$

$EG\varphi$ – Clarke et al '95

Given $EG\varphi$.

Set of fairness constraints H .

Incremental witness construction: sequence of prefixes of increasing length until a cycle is found.

In every step is hat to be guaranteed that current prefix can be extended to a fair path on which each state satisfies φ .

EG φ – Clarke et al '95 (Cont.)

Phase 1 Collecting informations for witness construction

1. $\mathbf{EG}\varphi = \nu Z. \varphi \wedge \bigwedge_{k=1}^n \mathbf{EX}(\mathbf{E}[\varphi \mathbf{U} Z \wedge h_k])$
 - Evaluating the **EU** operator for every constraint h
 - Q_0^h, Q_1^h, \dots
 - $Q_0^h = Z \wedge h, Q_i^h = (Z \wedge h) \vee (\varphi \wedge \mathbf{EX}Q_{i-1}^h)$
2. $Z \in \llbracket \mathbf{EG}\varphi \rrbracket$: Q_i^h for each h are saved.

Phase 2 Constructing witness using saved sequences Q_i^h

1. Forward computation (for every h):
 $(s_0 \in Q_0^h) \rightarrow (s_1 \in Q_1^h) \rightarrow \dots \rightarrow (s' \in \llbracket (\mathbf{EG}\varphi) \wedge h \rrbracket)$
2. Find cycle from s' to some previously visited t .
 - Examining SCC of Kripke.
 - Descending in the SCC until cycle is found.

Experimental Results – Bakery

Mutual exclusion property of the correct version.

	No. of customers	3	4
	Max. ticket number	7	10
WMC	Max. BDD size	434	2537
	Time (msecs)	60	1,180
	No. of iterations	1	1
MC	Max. BDD size	7523	489055
	Time (msecs)	1,650	114,210
	No. of iterations	24	18

Experimental Results – Bakery

Mutual exclusion property of a buggy version.

	No. of customers	3	4
	Max. ticket number	7	10
WMC	Max. BDD size	1365	12597
	Time (msecs)	600	17,260
	No. of iterations	13	15
MC	Max. BDD size	7435	466273
	Time (msecs)	1,710	98,010
	No. of iterations	24	18