CTL Counterexamples and CTL* Model Checking

Lecture #21 of Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: `katoen@cs.rwth-aachen.de`

January 14, 2009

Overview Lecture #21

$\Rightarrow$ CTL Counterexamples

- CTL* model checking

Counterexamples

- Model checking is an effective and efficient “bug hunting” technique
- Counterexamples are of utmost importance:
  - diagnostic feedback, the key to abstraction-refinement, schedule synthesis . . .
- LTL: counterexamples are finite paths
  - $\bigcirc\Phi$: a path on which the next state refutes Φ
  - $\Box\Phi$: a path leading to a $\neg\Phi$-state
  - $\Diamond\Phi$: a $\neg\Phi$-path leading to a $\neg\Phi$ cycle
- Counterexample generation for LTL:
  - use stack contents of nested DFS on encountering an accept cycle
  - use a variant of BFS top find shortest counterexamples

Counterexamples in CTL

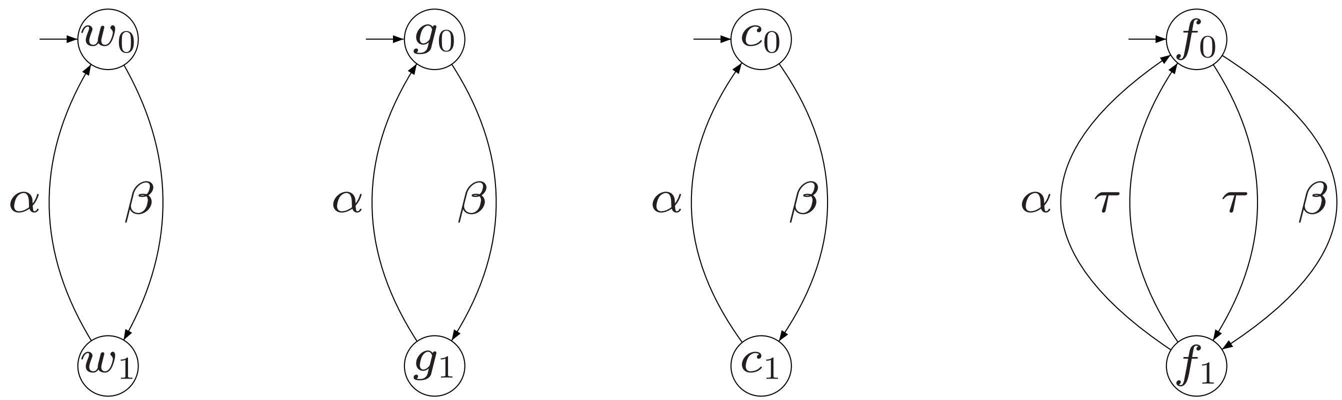
- $TS \not\models \forall\varphi$ where $\forall\varphi$ is also on LTL
  - counterexample = a sufficiently long prefix of a path refuting φ (as in LTL)
  - this is a subset of the so-called universal fragment of CTL
- $TS \not\models \exists\varphi$ where φ is arbitrary CTL formula
  - all paths satisfy φ! $\Rightarrow$ no clear notion of counterexample
  - witness = a sufficiently long prefix of a path satisfying φ
- So:
  - for $\forall\varphi$, a prefix of π with $\pi \not\models \varphi$ acts as counterexample
  - for $\exists\varphi$, a prefix of π with $\pi \models \varphi$ acts as witness

The wolf-goat-cabbage problem

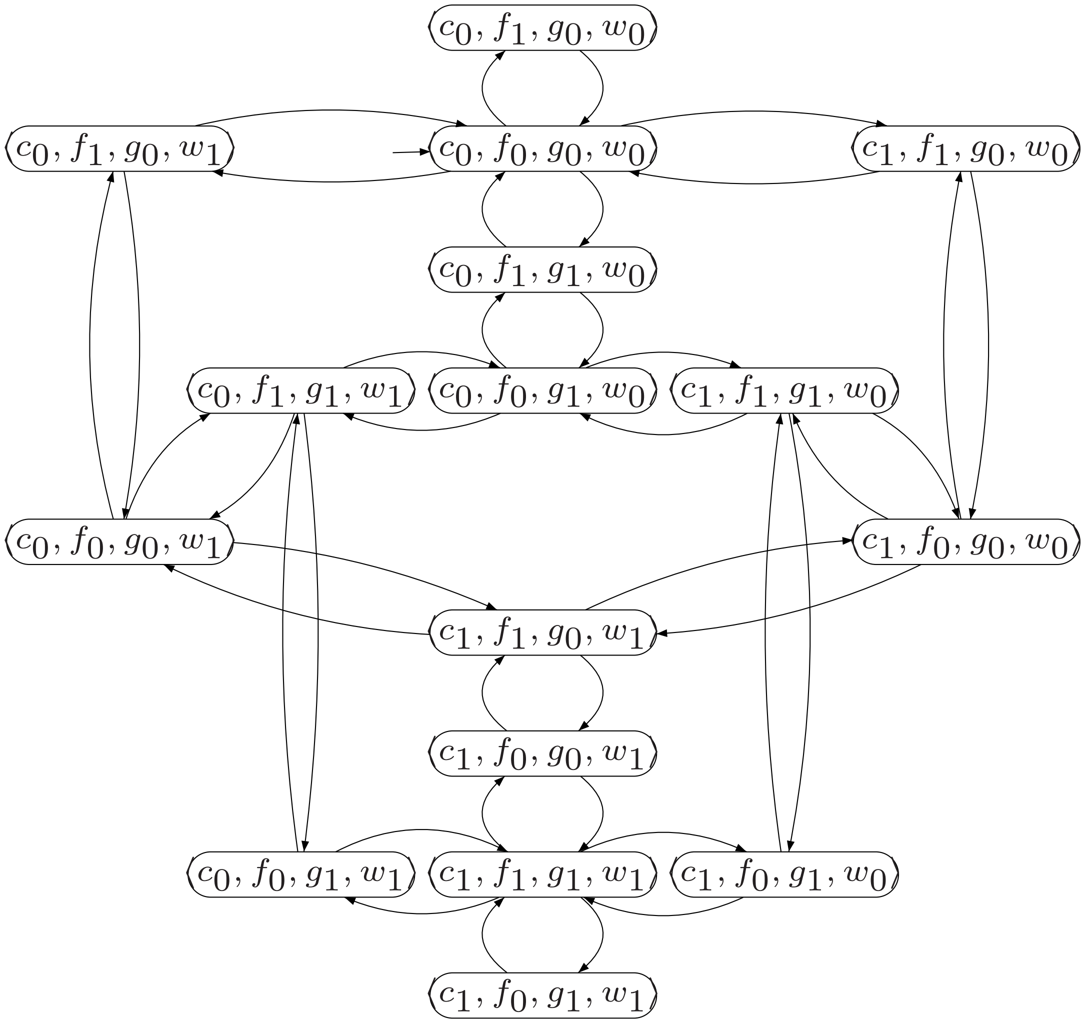
- A goat (g), a cabbage (c) and a wolf (w) and two riverbanks (0 and 1)
  - A boat with ferryman (f) that can carry at most two occupants
  - Only the ferryman can steer the boat
  - Goat and cabbage, goat and wolf should neither travel nor left together
- Is there a schedule such that brings c, g, and w to the other side?
- . . . Model this as a CTL model-checking problem
  - transition system *TS* $= \ (wolf \mid\mid\mid goat \mid\mid\mid cabbage) \parallel ferryman$
  - check whether *TS* $\models \exists\varphi$ with

$$\varphi \ = \ \left(\bigwedge_{i=0,1} (w_i \wedge g_i \rightarrow f_i) \ \wedge \ (c_i \wedge g_i \rightarrow f_i) \right) \ \mathsf{U} \ (c_1 \wedge f_1 \wedge g_1 \wedge w_1)$$

The wolf-goat-cabbage problem



$$\mathit{TS} \;=\; (\mathit{wolf} \;|||\; \mathit{goat} \;|||\; \mathit{cabbage}) \;\|\; \mathit{ferryman}$$


$\langle c_0, f_1, g_0, w_0 \rangle$
$\langle c_0, f_1, g_0, w_1 \rangle$
$\langle c_0, f_0, g_0, w_0 \rangle$
$\langle c_1, f_1, g_0, w_0 \rangle$
$\langle c_0, f_1, g_1, w_0 \rangle$
$\langle c_0, f_1, g_1, w_1 \rangle$
$\langle c_0, f_0, g_1, w_0 \rangle$
$\langle c_1, f_1, g_1, w_0 \rangle$
$\langle c_0, f_0, g_0, w_1 \rangle$
$\langle c_1, f_0, g_0, w_0 \rangle$
$\langle c_1, f_1, g_0, w_1 \rangle$
$\langle c_1, f_0, g_0, w_1 \rangle$
$\langle c_0, f_0, g_1, w_1 \rangle$
$\langle c_1, f_1, g_1, w_1 \rangle$
$\langle c_1, f_0, g_1, w_0 \rangle$
$\langle c_1, f_0, g_1, w_1 \rangle$

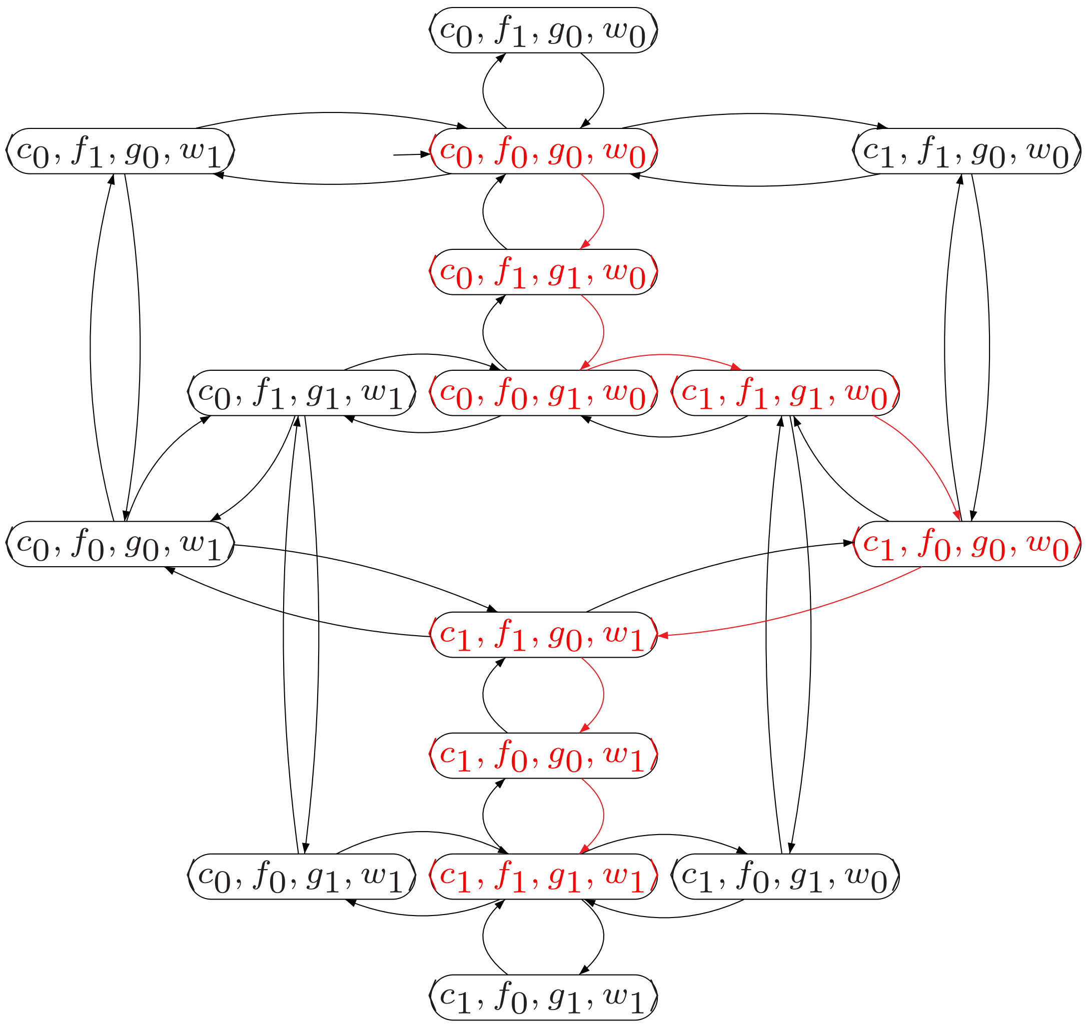
Wolf-goat-cabbage problem

A witness of $\exists\varphi$ with:

$$\varphi = \Big(\bigwedge_{i=0,1} (w_i \wedge g_i \rightarrow f_i) \;\wedge\; (c_i \wedge g_i \rightarrow f_i) \Big) \;\mathsf{U}\; (c_1 \wedge f_1 \wedge g_1 \wedge w_1)$$

is a path fragment from initial state $\langle c_0, f_0, g_0, w_0 \rangle$ to target state $\langle c_1, f_1, g_1, w_1 \rangle$ such that g, c and g, w are not left on a single riverbank. Such as:

$\langle c_0, f_0, g_0, w_0 \rangle$	goat to riverbank 1
$\langle c_0, f_1, g_1, w_0 \rangle$	ferryman comes back to riverbank 0
$\langle c_0, f_0, g_1, w_0 \rangle$	cabbage to riverbank 1
$\langle c_1, f_1, g_1, w_0 \rangle$	goat back to riverbank 0
$\langle c_1, f_0, g_0, w_0 \rangle$	wolf to riverbank 1
$\langle c_1, f_1, g_0, w_1 \rangle$	ferryman comes back to riverbank 0
$\langle c_1, f_0, g_0, w_1 \rangle$	goat to riverbank 1
$\langle c_1, f_1, g_1, w_1 \rangle$	


$\langle c_0, f_1, g_0, w_0 \rangle$
$\langle c_0, f_1, g_0, w_1 \rangle$
$\langle c_0, f_0, g_0, w_0 \rangle$
$\langle c_1, f_1, g_0, w_0 \rangle$
$\langle c_0, f_1, g_1, w_0 \rangle$
$\langle c_0, f_1, g_1, w_1 \rangle$
$\langle c_0, f_0, g_1, w_0 \rangle$
$\langle c_1, f_1, g_1, w_0 \rangle$
$\langle c_0, f_0, g_0, w_1 \rangle$
$\langle c_1, f_0, g_0, w_0 \rangle$
$\langle c_1, f_1, g_0, w_1 \rangle$
$\langle c_1, f_0, g_0, w_1 \rangle$
$\langle c_0, f_0, g_1, w_1 \rangle$
$\langle c_1, f_1, g_1, w_1 \rangle$
$\langle c_1, f_0, g_1, w_0 \rangle$
$\langle c_1, f_0, g_1, w_1 \rangle$

Counterexamples for $\bigcirc\Phi$

- A counterexample of $\bigcirc\Phi$ is a path fragment $s\,s'$ with
  - $s \in I$ and $s' \in \mathit{Post}(s)$ with $s' \not\models \Phi$
- A witness of $\bigcirc\Phi$ is a is a path fragment $s\,s'$ with
  - $s \in I$ and $s' \in \mathit{Post}(s)$ with $s' \models \Phi$
- Algorithm: inspection of direct successors of initial states

Counterexamples for $\Phi \, \mathsf{U} \, \Psi$

- A witness is an initial path fragment $s_0 \, s_1 \ldots s_n$ with
  - $s_n \models \Psi$ and $s_i \models \Phi$ for $0 \leqslant i < n$
- Algorithm: backward search starting in the set of Ψ-states
- A counterexample is an initial path fragment that indicates a path π:
  - for which either $\pi \models \Box(\Phi \wedge \neg\Psi)$ or $\pi \models (\Phi \wedge \neg\Psi) \, \mathsf{U} \, (\neg\Phi \wedge \neg\Psi)$
- Counterexample is initial path fragment of either form:
  - $\underbrace{s_0 \ldots s_{n-1} \, \underbrace{s_n \, s'_1 \ldots s'_r}_{\text{cycle}}}_{\text{satisfy } \Phi \wedge \neg\Psi}$ with $s_n = s'_r$ or $\underbrace{s_0 \ldots s_{n-1}}_{\text{satisfy } \Phi \wedge \neg\Psi} \; s_n$ with $s_n \models \neg\Phi \wedge \neg\Psi$

Counterexample generation

Determine the SCCs by of the digraph $G = (S, E)$ where

$$E \;=\; \{\, (s, s') \in S \times S \mid s' \in \mathit{Post}(s) \;\wedge\; s \models \Phi \wedge \neg \Psi \,\}$$

Each path in G that starts in an initial state $s_0 \in S$ and leads to a non-trivial SCC C in G provides a counterexample of the form:

$$s_0\, s_1 \ldots s_n \underbrace{s'_1 \ldots s'_r}_{\in C} \quad \text{with} \quad s_n = s'_r$$

Each path in G that leads from an initial state s_0 to a trivial terminal SCC

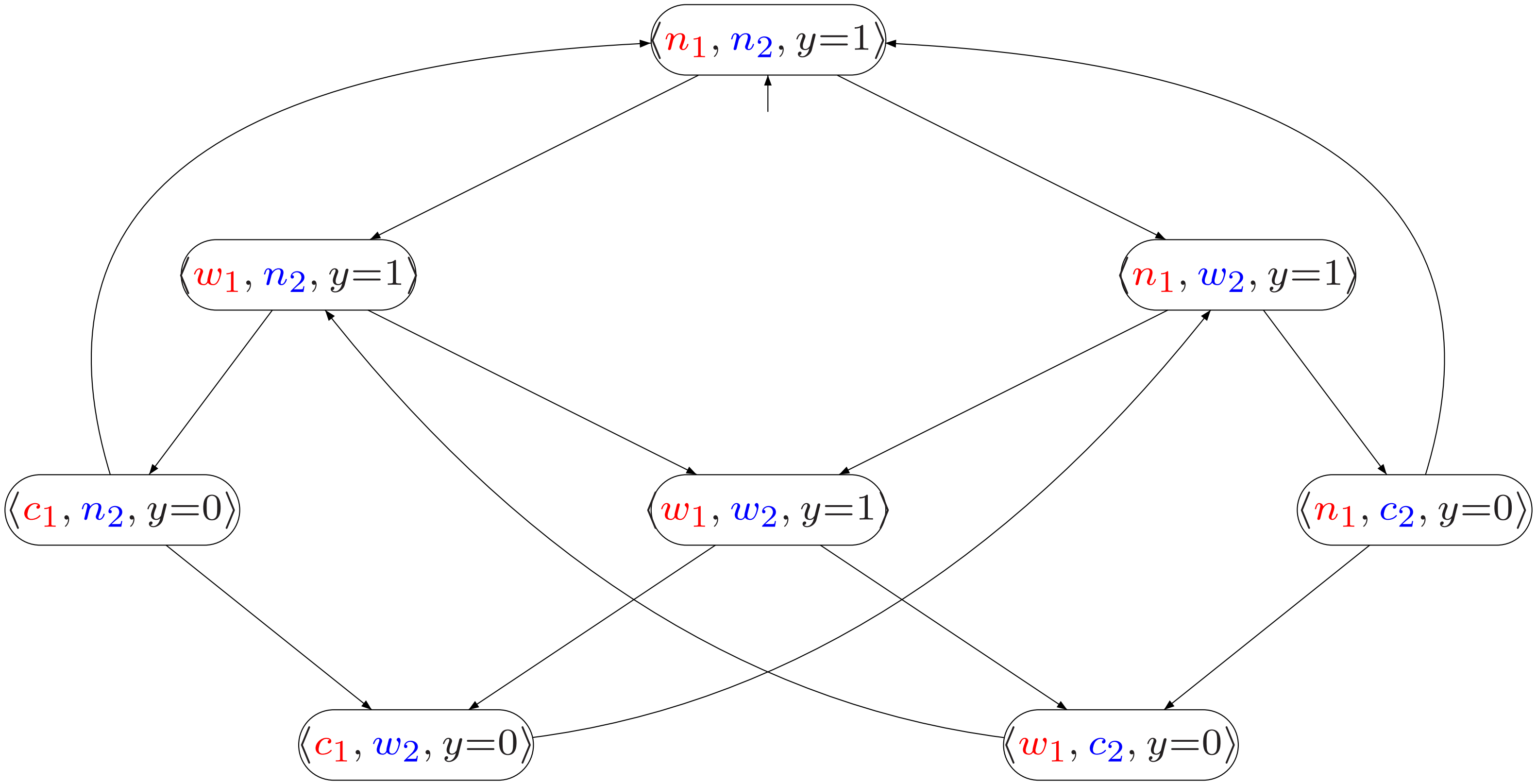
$$C = \{\, s' \,\} \quad \text{with} \quad s' \not\models \Psi$$

provides a counterexample of the form $s_0\, s_1 \ldots s_n$ with $s_n \models \neg \Phi \wedge \neg \Psi$

Counterexamples for $\Box\Phi$

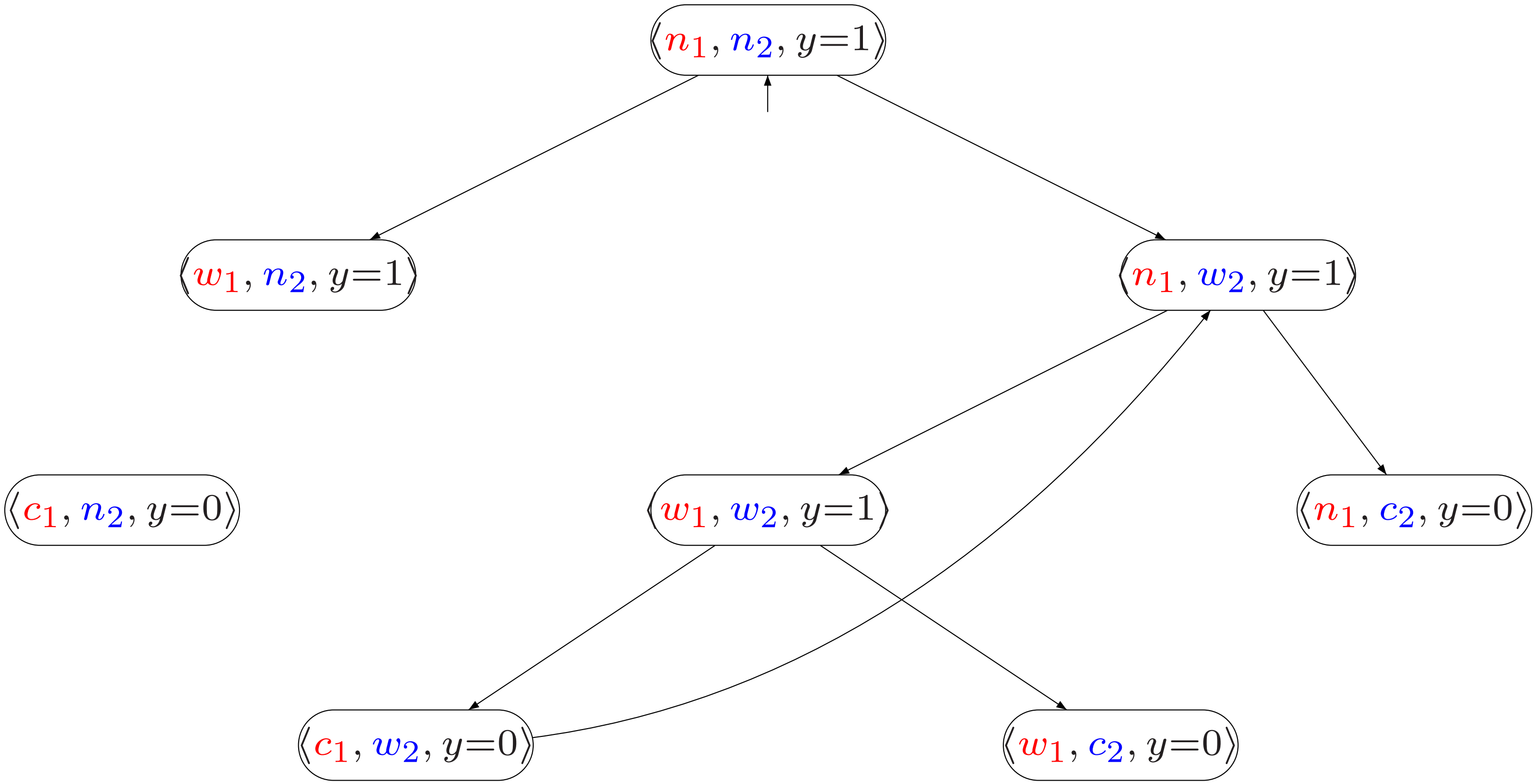
- Counterexample is initial path fragment $s_0\, s_1 \ldots s_n$ such that:
  - $s_0, \ldots, s_{n-1} \models \Phi$ and $s_n \not\models \Phi$
- Algorithm: backward search starting in $\neg\Phi$-states
- A witness of $\varphi = \Box\Phi$ consists of an initial path fragment of the form:
  - $\underbrace{s_0\, s_1 \ldots s_n\, s_1' \ldots s_r'}_{\text{satisfy } \Phi}$ with $s_n = s_r'$
- Algorithm: cycle search in the digraph $G = (S, E)$ where the set of edges E:
  - $E \;=\; \{\, (s, s') \mid s' \in \mathit{Post}(s) \;\wedge\; s \models \Phi \,\}$

Example



$$\forall\Big(\underbrace{((n_1 \wedge n_2) \ \vee \ w_2)}_{\Phi} \ \mathsf{U} \ \underbrace{c_2}_{\Psi}\Big)$$

SCC graph


$\langle n_1, n_2, y{=}1 \rangle$
$\langle w_1, n_2, y{=}1 \rangle$
$\langle n_1, w_2, y{=}1 \rangle$
$\langle c_1, n_2, y{=}0 \rangle$
$\langle w_1, w_2, y{=}1 \rangle$
$\langle n_1, c_2, y{=}0 \rangle$
$\langle c_1, w_2, y{=}0 \rangle$
$\langle w_1, c_2, y{=}0 \rangle$

Time complexity

Let *TS* be a transition system *TS* with N states and K transitions and φ a CTL- path formula

If *TS* $\not\models \forall\varphi$ then a counterexample for φ in *TS* can be determined in time $\mathcal{O}(N+K)$.

The same holds for a witness for φ, provided that *TS* $\models \exists\varphi$.

Overview Lecture #21

- CTL Counterexamples

$\Rightarrow$ CTL* model checking

Syntax of CTL*

CTL* *state-formulas* are formed according to:

$$\Phi ::= \text{true} \;\Big|\; a \;\Big|\; \Phi_1 \wedge \Phi_2 \;\Big|\; \neg\Phi \;\Big|\; \exists\varphi$$

where $a \in AP$ and φ is a path-formula

CTL* *path-formulas* are formed according to the grammar:

$$\varphi ::= \Phi \;\Big|\; \varphi_1 \wedge \varphi_2 \;\Big|\; \neg\varphi \;\Big|\; \bigcirc\varphi \;\Big|\; \varphi_1 \,\mathsf{U}\, \varphi_2$$

where Φ is a state-formula, and φ, φ_1 and φ_2 are path-formulas

in CTL*: $\forall\varphi \;=\; \neg\exists\neg\varphi$. This does not hold in CTL!

CTL* semantics

$s \models a$	iff	$a \in L(s)$
$s \models \neg\,\Phi$	iff	not $s \models \Phi$
$s \models \Phi \wedge \Psi$	iff	$(s \models \Phi)$ and $(s \models \Psi)$
$s \models \exists\varphi$	iff	$\pi \models \varphi$ for some $\pi \in \mathit{Paths}(s)$

$\pi \models \Phi$	iff	$\pi[0] \models \Phi$
$\pi \models \varphi_1 \wedge \varphi_2$	iff	$\pi \models \varphi_1$ and $\pi \models \varphi_2$
$\pi \models \neg\varphi$	iff	$\pi \not\models \varphi$
$\pi \models \bigcirc\varphi$	iff	$\pi[1..] \models \varphi$
$\pi \models \varphi_1 \mathsf{U} \varphi_2$	iff	$\exists\, j \geqslant 0.\ (\pi[j..] \models \varphi_2 \ \wedge\ (\forall\, 0 \leqslant k < j.\ \pi[k..] \models \varphi_1))$

Transition system semantics

- For CTL*-state-formula Φ, the *satisfaction set* *Sat*(Φ) is defined by:

$$Sat(\Phi) \;=\; \{\, s \in S \mid s \models \Phi \,\}$$

- *TS* satisfies CTL*-formula Φ iff Φ holds in all its initial states:

$$TS \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I.\, s_0 \models \Phi$$

this is exactly as for CTL

Embedding of LTL in CTL*

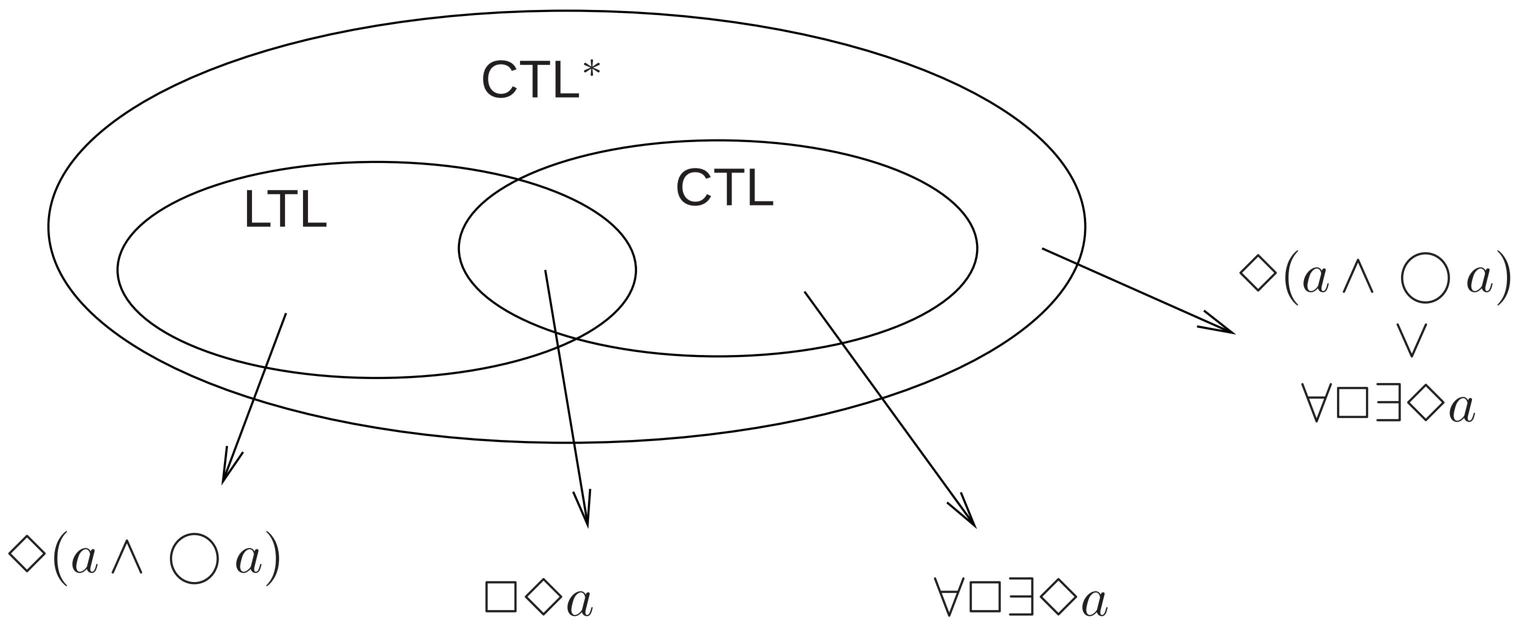
For LTL formula φ and *TS* without terminal states (both over *AP*) and for each $s \in S$:

$$\underbrace{s \models \varphi}_{\text{LTL semantics}} \quad \text{if and only if} \quad \underbrace{s \models \forall\varphi}_{\text{CTL}^* \text{ semantics}}$$

In particular:

$$TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL*} \forall\varphi$$

Expressivity of CTL*



CTL* model checking

[Emerson & Lei, 1985]

- Adopt the same bottom-up procedure as for CTL
- Replace *maximal proper state sub-formula* Ψ by new proposition a_Ψ
  - adjust labeling such that $a_\Psi \in L(s)$ if and only if $s \in Sat(\Psi)$
- In the end, this yields an LTL formula!
- Most interesting case: formulas of the form $\exists\varphi$

$$s \models \exists\varphi \quad \text{iff} \quad \underbrace{s \not\models \forall\neg\varphi}_{\text{CTL}^* \text{ semantics}} \quad \text{iff} \quad \underbrace{s \not\models \neg\varphi}_{\text{LTL semantics}}$$

  - $Sat_{CTL*}(\exists\varphi) = S \setminus Sat_{LTL}(\neg\varphi) = S \setminus \{ s \in S \mid s \models_{LTL} \neg\varphi \}$

Abstract example

CTL* model-checking algorithm

for all $i \leqslant |\,\Phi\,|$ **do**
  for all $\Psi \in$ *Sub*(Φ) with $|\,\Psi\,| = i$ **do**
    switch(Ψ):

true	:	*Sat*$(\Psi) := S$;
a	:	*Sat*$(\Psi) := \{\, s \in S \mid a \in L(s) \,\}$;
$a_1 \wedge a_2$	:	*Sat*$(\Psi) :=$ *Sat*$(a_1) \cap$ *Sat*(a_2);
$\neg a$	:	*Sat*$(\Psi) := S \setminus$ *Sat*(a);
$\exists \varphi$	:	determine *Sat*$_{LTL}(\neg \varphi)$;
	:	*Sat*$(\Psi) := S \setminus$ *Sat*$_{LTL}(\neg \varphi)$

    end switch
    AP $:=$ *AP* $\cup \{\, a_\Psi \,\}$; (* introduce fresh atomic proposition *)
    replace Ψ with a_Ψ;
    forall $s \in$ *Sat*(Ψ) **do** $L(s) := L(s) \cup \{\, a_\Psi \,\}$; **od**
  od
od
return $I \subseteq$ *Sat*(Φ)

Example

Time complexity

For transition system *TS* with N states and M transitions, CTL* formula Φ, the CTL* model-checking problem *TS* $\models \Phi$ can be determined in time $\mathcal{O}((N+M)\cdot 2^{|\Phi|})$.

The CTL* model-checking problem is PSPACE-complete

Complexity overview

	CTL	LTL	CTL*
model checking without fairness	PTIME $\mathit{size}(\mathit{TS}) \cdot \lvert\Phi\rvert$	PSPACE-complete $\mathit{size}(\mathit{TS}) \cdot \exp(\lvert\Phi\rvert)$	PSPACE-complete $\mathit{size}(\mathit{TS}) \cdot \exp(\lvert\Phi\rvert)$
satisfiability check best known technique	EXPTIME $\exp(\lvert\Phi\rvert)$	PSPACE-complete $\exp(\lvert\Phi\rvert)$	2EXPTIME $\exp(\exp(\lvert\Phi\rvert))$