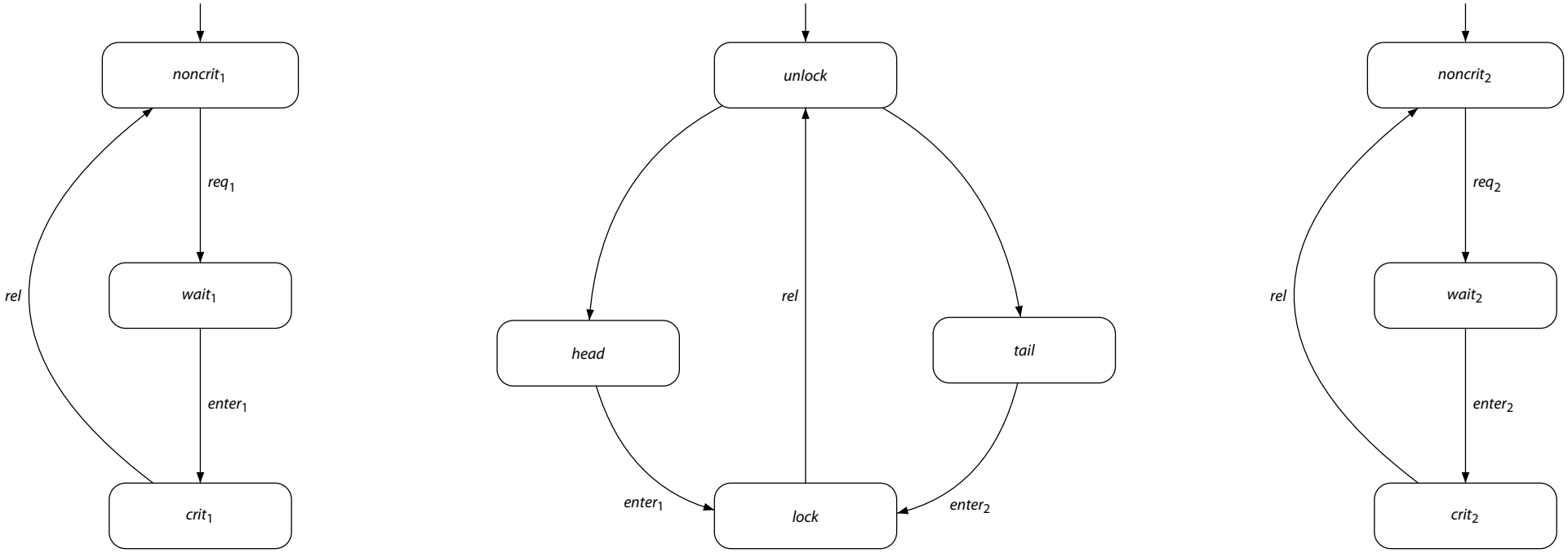
Verification

Lecture 11

Bernd Finkbeiner
Peter Faymonville
Michael Gerke

UNIVERSITÄT
DES
SAARLANDES

Randomized arbiter



$$TS_1 \parallel Arbiter \parallel TS_2 \not\models \mathsf{AG}\,\mathsf{AF}\, crit_1$$

But: $TS_1 \parallel Arbiter \parallel TS_2 \models_{fair} \mathsf{AG}\,\mathsf{AF}\, crit_1 \wedge \mathsf{AG}\,\mathsf{AF}\, crit_2$ with

$$fair = \mathsf{G}\,\mathsf{F}\, head \wedge \mathsf{G}\,\mathsf{F}\, tail$$

REVIEW: CTL fairness constraints

- A strong CTL fairness constraint is a formula of the form:

$$sfair = \bigwedge_{0<i\leq k} (\mathsf{G}\,\mathsf{F}\,\Phi_i \rightarrow \mathsf{G}\,\mathsf{F}\,\Psi_i)$$

  where Φ_i and Ψ_i (for $0 < i \leq k$) are CTL-formulas over AP.

- A weak CTL fairness constraint is a formula of the form:

$$wfair = \bigwedge_{0<i\leq k} (\mathsf{F}\,\mathsf{G}\,\Phi_i \rightarrow \mathsf{G}\,\mathsf{F}\,\Psi_i)$$

- a CTL fairness assumption *fair* is a conjunction of CTL fairness constraints.

$\Rightarrow$ a CTL fairness constraint is an LTL formula over CTL state formulas!

REVIEW: Semantics of fair CTL

For CTL fairness assumption *fair*, relation $\models_{fair}$ is defined by:

$s \models_{fair} a$ iff $a \in Label(s)$

$s \models_{fair} \neg \Phi$ iff $\neg (s \models_{fair} \Phi)$

$s \models_{fair} \Phi \vee \Psi$ iff $(s \models_{fair} \Phi) \vee (s \models_{fair} \Psi)$

$s \models_{fair} \mathsf{E}\, \varphi$ iff $\pi \models_{fair} \varphi$ for some fair path π that starts in s

$s \models_{fair} \mathsf{A}\, \varphi$ iff $\pi \models_{fair} \varphi$ for all fair paths π that start in s

$\pi \models_{fair} \mathsf{X}\, \Phi$ iff $\pi[1] \models_{fair} \Phi$

$\pi \models_{fair} \Phi\, \mathsf{U}\, \Psi$ iff $(\exists j \geq 0.\ \pi[j] \models_{fair} \Psi \wedge (\forall\, 0 \leq k < j.\ \pi[k] \models_{fair} \Phi))$

π is a fair path iff $\pi \models$ *fair* for CTL fairness assumption *fair*

REVIEW: Fair CTL model checking

- Determine $Sat_{fair}(\mathrm{EG}\,\mathrm{true}) = \{ q \in S \mid FairPaths(q) \neq \varnothing \}$
- Introduce an atomic proposition a_{fair} such that:
  - $a_{fair} \in L(q)$ if $q \in Sat_{fair}(\mathrm{EG}\,\mathrm{true})$
- Compute the sets $Sat_{fair}(\Psi)$ for all subformulas Ψ of Φ (in ENF)

by:

$$
\begin{aligned}
Sat_{fair}(a) &= \{ q \in S \mid a \in L(q) \} \\
Sat_{fair}(\neg a) &= S \setminus Sat_{fair}(a) \\
Sat_{fair}(a \wedge a') &= Sat_{fair}(a) \cap Sat_{fair}(a') \\
Sat_{fair}(\mathrm{EX}\,a) &= Sat\,(\mathrm{EX}\,(a \wedge a_{fair})) \\
Sat_{fair}(\mathrm{E}\,(a\,\mathrm{U}\,a')) &= Sat\,(\mathrm{E}\,(a\,\mathrm{U}\,(a' \wedge a_{fair}))) \\
Sat_{fair}(\mathrm{EG}\,a) &= \text{(to be discussed)}
\end{aligned}
$$

- Thus: model checking CTL under fairness constraints is
  - CTL model checking + algorithm for computing $Sat_{fair}(\mathrm{EG}\,a)$!

Characterization of $Sat_{fair}(\mathsf{EG}\, a)$

$$q \vDash_{sfair} \mathsf{EG}\, a \quad \text{where} \quad sfair = \bigwedge_{0<i\leq k} (\mathsf{GF}\, b_i \rightarrow \mathsf{GF}\, c_i)$$

iff there exists a finite path fragment $q_0 \ldots q_n$ and a cycle $q'_0 \ldots q'_r$ with:

1. $q_0 = q$ and $q_n = q'_0 = q'_r$
2. $q_i \vDash a$, for all $0 \leq i \leq n$, and $q'_j \vDash a$, for all $0 \leq j \leq r$, and
3. $Sat(b_i) \cap \{ q'_1, \ldots, q'_r \} = \varnothing$ or
   $Sat(c_i) \cap \{ q'_1, \ldots, q'_r \} \neq \varnothing$ for $0 < i \leq k$

Computing $Sat_{fair}(\mathrm{EG}\, a)$

- Consider state q only if $q \models a$, otherwise eliminate q
  - change TS into $TS[a] = (S', Act, \rightarrow', I', AP, L')$ with $S' = Sat(a)$,
  - $\rightarrow' = \rightarrow \cap (S' \times Act \times S')$, $I' = I \cap S'$, and $L'(s) = L(s)$ for $s \in S'$

  $\Rightarrow$ each infinite path fragment in $TS[a]$ satisfies $\mathrm{G}\, a$
- $q \models_{fair} \mathrm{EG}\, a$ iff there is a non-trivial SCC D in $TS[a]$ reachable from q such that
  - $D \cap Sat(b_i) = \varnothing$ or
  - $D \cap Sat(c_i) \neq \varnothing$

  for $0 < i \leq k$
- $Sat_{sfair}(\mathrm{EG}\, a) = \{ q \in S \mid Reach_{TS[a]}(s) \cap T \neq \varnothing \}$
  - T is the union of all such SCCs D.

how to compute T?

Unconditional fairness

$$ufair \equiv \bigwedge_{0<i\leq k} \mathsf{G}\,\mathsf{F}\,b_i$$

Let T be the set union of all non-trivial SCCs C of $TS[a]$ satisfying

$$C \cap Sat(b_i) \neq \varnothing \quad \text{for all } 0 < i \leq k$$

It now follows:

$$s \vDash_{ufair} \mathsf{EG}\,a \quad \text{if and only if} \quad Reach_{G[a]}(s) \cap T \neq \varnothing$$

$\Rightarrow$ T can be determined by a simple graph analysis (DFS)

Strong fairness: single constraint ($k = 1$)

- $sfair \;=\; \mathsf{G}\,\mathsf{F}\,b_1 \rightarrow \mathsf{G}\,\mathsf{F}\,c_1$
- $q \models_{sfair} \mathsf{E}\,\mathsf{G}\,a$ iff C is a non-trivial SCC in $TS[a]$ reachable from q with:
  (1) $C \cap Sat(c_1) \neq \emptyset$, or
  (2) there exists a non-trivial SCC D in $C[\neg b_1]$
- For the union T of all such SCCs C:

$$q \models_{sfair} \mathsf{E}\,\mathsf{G}\,a \quad \text{if and only if} \quad Reach_{S[a]}(q) \cap T \neq \emptyset$$

Strong fairness: general case ($k > 1$)

Check each non-trivial SCC C recursively as follows:

Check(C, $\bigwedge_{0<i\leq k} (\mathsf{GF}\, b_i \rightarrow \mathsf{GF}\, c_i)$):

if $\forall i \in \{1, \ldots, k\} : C \cap Sat(c_i) \neq \varnothing$ **return** *true*

else

choose some $j \in \{1, \ldots, k\} : C \cap Sat(c_j) = \varnothing$.
remove all states in $Sat(b_j)$ from C
for all non-trivial SCCs D do

if Check(D, $\bigwedge_{0<i\leq k, i\neq j} (\mathsf{GF}\, b_i \rightarrow \mathsf{GF}\, c_i)$) **return** *true*

return *false*

T is the union of all SCCs C that pass the check.

Time complexity

For state graph *TS* with *N* states and *M* edges,

CTL formula Φ, and CTL fairness constraint *fair* with *k* conjuncts,

the CTL model-checking problem $TS \models_{fair} \Phi$

can be determined in time $\mathcal{O}(|\Phi| \cdot (N + M) \cdot k)$

Syntax of CTL*

CTL* state-formulas are formed according to:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathsf{E}\,\varphi$$

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

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

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathsf{X}\,\varphi \mid \varphi_1\,\mathsf{U}\,\varphi_2$$

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

in CTL*: $\mathsf{A}\,\varphi = \neg\mathsf{E}\,\neg\varphi$.

CTL* semantics

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg \Phi & \text{iff} & \text{not } s \models \Phi \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \text{ and } (s \models \Psi) \\
s \models \mathsf{E}\, \varphi & \text{iff} & \pi \models \varphi \text{ for some } \pi \in \mathit{Paths}(s)
\end{array}
$$

$$
\begin{array}{lll}
\pi \models \Phi & \text{iff} & \pi[0] \models \Phi \\
\pi \models \varphi_1 \wedge \varphi_2 & \text{iff} & \pi \models \varphi_1 \text{ and } \pi \models \varphi_2 \\
\pi \models \neg \varphi & \text{iff} & \text{not } \pi \models \varphi \\
\pi \models \mathsf{X}\, \Phi & \text{iff} & \pi[1..] \models \Phi \\
\pi \models \Phi\, \mathsf{U}\, \Psi & \text{iff} & \exists j \geq 0.\ (\pi[j..] \models \Psi \wedge (\forall\, 0 \leq k < j.\ \pi[k..] \models \Phi))
\end{array}
$$

State graph semantics

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

$$Sat(\Phi) \;=\; \{\, q \in S \mid q \vDash \Phi \,\}$$

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

$$TS \vDash \Phi \quad \text{if and only if} \quad \forall q \in I.\, q_0 \vDash \Phi$$

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 $q \in S$:

$$\underbrace{q \models \varphi}_{\text{LTL semantics}} \quad \text{if and only if} \quad \underbrace{q \models \mathrm{A}\,\varphi}_{\text{CTL}^* \text{ semantics}}$$

In particular:

$$TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL*} \mathrm{A}\,\varphi$$

CTL* is more expressive than LTL and CTL

For the CTL*-formula over $AP = \{ a, b \}$:

$$\Phi \;=\; (\mathsf{AF}\,\mathsf{G}\; a) \;\vee\; (\mathsf{AG}\,\mathsf{EF}\; b)$$

there does not exist any equivalent LTL- or CTL formula

This logic is as expressive as CTL

CTL$^+$ state-formulas are formed according to:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$$

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

CTL$^+$ path-formulas are formed according to the grammar:

$$\varphi ::= \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \mathsf{X}\,\Phi \mid \Phi_1\,\mathsf{U}\,\Phi_2$$

where Φ, Φ_1, Φ_2 are state-formulas, and φ, φ_1 and φ_2 are path-formulas

CTL$^+$ is as expressive as CTL

For example:

$$\underbrace{\mathsf{E}\,(\mathsf{F}\,a \wedge \mathsf{F}\,b)}_{\text{CTL}^+\text{ formula}} \equiv \underbrace{\mathsf{E}\,\mathsf{F}\,(a \wedge \mathsf{E}\,\mathsf{F}\,b) \vee \mathsf{E}\,\mathsf{F}\,(b \wedge \mathsf{E}\,\mathsf{F}\,a)}_{\text{CTL formula}}$$

Some rules for transforming CTL$^+$ formulas into equivalent CTL ones:

$$\begin{aligned}
\mathsf{E}\,(\neg(\Phi_1\,\mathsf{U}\,\Phi_2)) &\equiv \mathsf{E}\,((\Phi_1 \wedge \neg\Phi_2)\,\mathsf{U}\,(\neg\Phi_1 \wedge \neg\Phi_2)) \vee \mathsf{E}\,\mathsf{G}\,\neg\Phi_2 \\
\mathsf{E}\,(\mathsf{X}\,\Phi_1 \wedge \mathsf{X}\,\Phi_2) &\equiv \mathsf{E}\,\mathsf{X}\,(\Phi_1 \wedge \Phi_2) \\
\mathsf{E}\,(\mathsf{X}\,\Phi \wedge (\Phi_1\,\mathsf{U}\,\Phi_2)) &\equiv (\Phi_2 \wedge \mathsf{E}\,\mathsf{X}\,\Phi) \vee (\Phi_1 \wedge \mathsf{E}\,\mathsf{X}\,(\Phi \wedge \mathsf{E}\,(\Phi_1\,\mathsf{U}\,\Phi_2))) \\
\mathsf{E}\,((\Phi_1\,\mathsf{U}\,\Phi_2) \wedge (\Psi_1\,\mathsf{U}\,\Psi_2)) &\equiv \mathsf{E}\,((\Phi_1 \wedge \Psi_1)\,\mathsf{U}\,(\Phi_2 \wedge \mathsf{E}\,(\Psi_1\,\mathsf{U}\,\Psi_2))) \vee \\
&\quad\ \mathsf{E}\,((\Phi_1 \wedge \Psi_1)\,\mathsf{U}\,(\Psi_2 \wedge \mathsf{E}\,(\Phi_1\,\mathsf{U}\,\Phi_2))) \\
&\ \ \vdots
\end{aligned}$$

adding boolean combinations of path formulas to CTL does not change its expressiveness

but CTL$^+$ formulas can be much shorter than shortest equivalent CTL formulas

CTL* model checking

- Adopt the same bottom-up procedure as for (fair) CTL
- Replace each maximal proper state subformula Ψ by new proposition a_Ψ
  - $a_\Psi \in L(s)$ if and only if $s \in Sat(\Psi)$
- Most interesting case: formulas of the form $\mathsf{E}\,\varphi$
  - by replacing all maximal state sub-formulas in φ, an LTL-formula results!
- $q \models \mathsf{E}\,\varphi$ iff $\underbrace{q \not\models \mathsf{A}\,\neg\varphi}_{\text{CTL}^* \text{ semantics}}$ iff $\underbrace{q \not\models \neg\varphi}_{\text{LTL semantics}}$
  - $Sat_{CTL*}(\mathsf{E}\,\varphi) = S \setminus Sat_{LTL}(\neg\varphi)$

CTL* model-checking algorithm

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

true	:	$Sat(\Psi) := S$;
a	:	$Sat(\Psi) := \{ q \in S \mid a \in L(q) \}$;
$a_1 \wedge a_2$	:	$Sat(\Psi) := Sat(a_1) \cap Sat(a_2)$;
$\neg a$	:	$Sat(\Psi) := S \setminus Sat(a)$;
E φ	:	determine $Sat_{LTL}(\neg\varphi)$ by means of an LTL model checker;
	:	$Sat(\Psi) := S \setminus Sat_{LTL}(\neg\varphi)$

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

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

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
  - X Φ: a path on which the next state refutes Φ
  - G Φ: a path leading to a $\neg\Phi$-state
  - F Φ: 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 to find shortest counterexamples

Counterexamples in CTL

- $TS \not\models \mathrm{A}\,\varphi$ where φ only contains universal path quantifiers
  - counterexample = a sufficiently long prefix of a path refuting φ (as in LTL)
  - this fragment of the logic is known as universal fragment of CTL
- $TS \not\models \mathrm{E}\,\varphi$ where φ is arbitrary CTL formula
  - all paths satisfy $\neg\varphi$! $\Rightarrow$ no clear notion of counterexample
  - witness = a sufficiently long prefix of a path satisfying φ
- So:
  - for $\mathrm{A}\,\varphi$, a prefix of π with $\pi \not\models \varphi$ acts as counterexample
  - for $\mathrm{E}\,\varphi$, a prefix of π with $\pi \models \varphi$ acts as witness

Counterexamples for X Φ

- A counterexample of X Φ is a path fragment $q\,q'$ with
  - $q \in I$ and $q' \in Post(q)$ with $q' \not\models \Phi$
- A witness of X Φ is a is a path fragment $q\,q'$ with
  - $q \in I$ and $q' \in Post(q)$ with $q' \models \Phi$
- Algorithm: inspection of direct successors of initial states

Counterexamples for G Φ

- Counterexample is initial path fragment $q_0\, q_1 \ldots q_n$ such that:
  - $q_0, \ldots, q_{n-1} \vDash \Phi$ and $q_n \nvDash \Phi$
- Algorithm: backward search starting in $\neg\Phi$-states
- A witness of $\varphi = \mathrm{G}\,\Phi$ consists of an initial path fragment of the form:
  - $\underbrace{q_0\, q_1 \ldots q_n\, q'_1 \ldots q'_r}_{\text{satisfy } \Phi}$ with $q_n = q'_r$
- Algorithm: cycle search in the digraph $G = (S, E')$ where the set of edges E':
  - $E' = \{ (q, q') \mid q' \in \mathit{Post}(q) \ \wedge \ q \vDash \Phi \}$

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

- A witness is an initial path fragment $q_0\, q_1 \ldots q_n$ with
  - $q_n \vDash \Psi$ and $q_i \vDash \Phi$ for $0 \leq i < n$
- 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
    $\pi \vDash \mathsf{G}\,(\Phi \wedge \neg\Psi)$ or $\pi \vDash (\Phi \wedge \neg\Psi)\,\mathsf{U}\,(\neg\Phi \wedge \neg\Psi)$
- Counterexample is initial path fragment of either form:
  - $\underbrace{q_0 \ldots q_{n-1}\ \underbrace{q_n\, q'_1 \ldots q'_r}_{\text{cycle}}}_{\text{satisfy } \Phi \wedge \neg\Psi}$ with $q_n = q'_r$ or
    $\underbrace{q_0 \ldots q_{n-1}}_{\text{satisfy } \Phi \wedge \neg\Psi}\ q_n$ with $q_n \vDash \neg\Phi \wedge \neg\Psi$

Counterexample generation

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

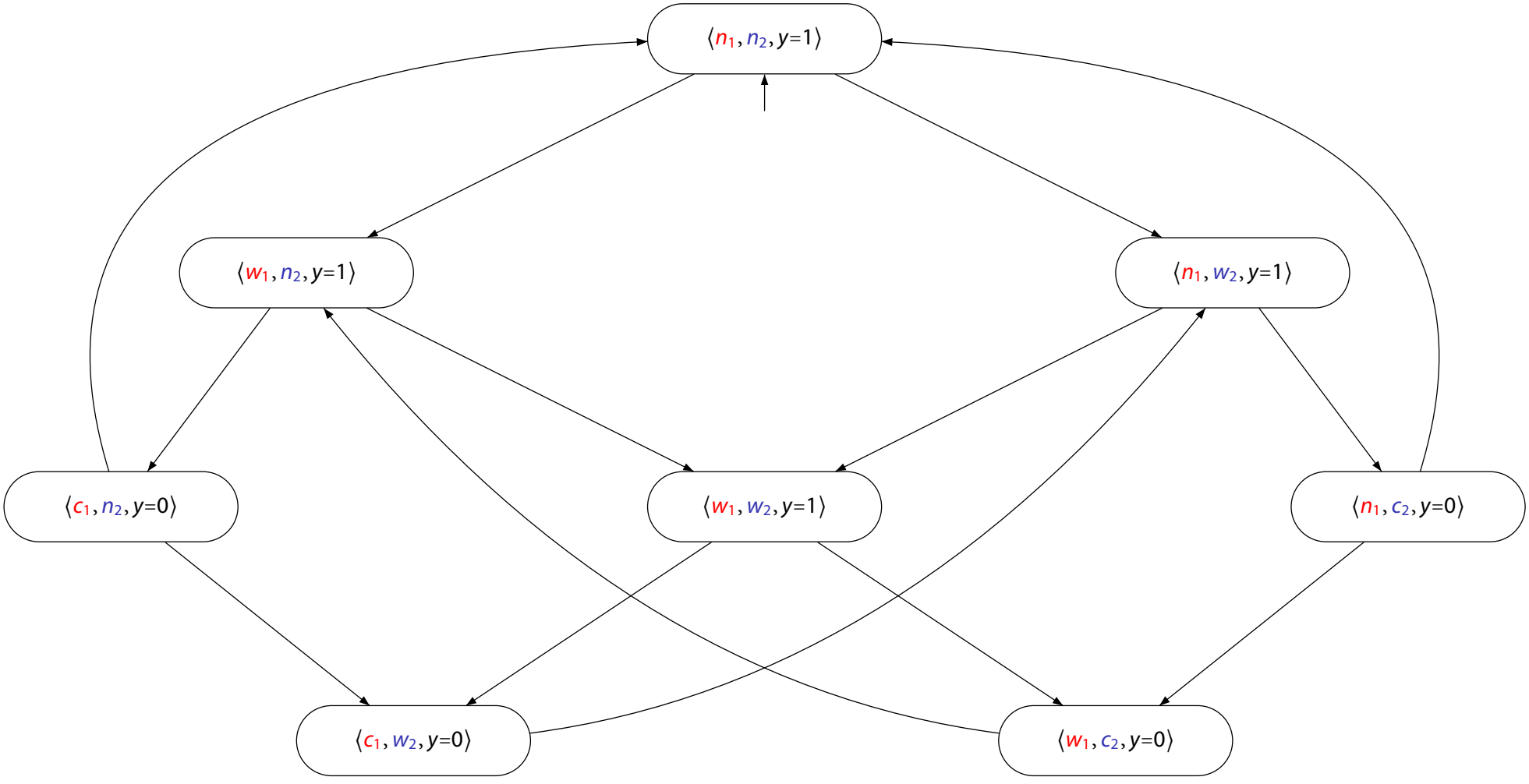
$$E' = \{ (q, q') \in S \times S \mid q' \in Post(q) \ \wedge \ q \models \Phi \ \wedge \ \neg\Psi \}$$

- Each path in G that starts in an initial state $q_0 \in I$ and leads to a non-trivial SCC C in G provides a counterexample of the form:

$$q_0\, q_1 \ldots q_n \underbrace{q'_1 \ldots q'_r}_{\in C} \quad \text{with} \quad q_n = q'_r$$

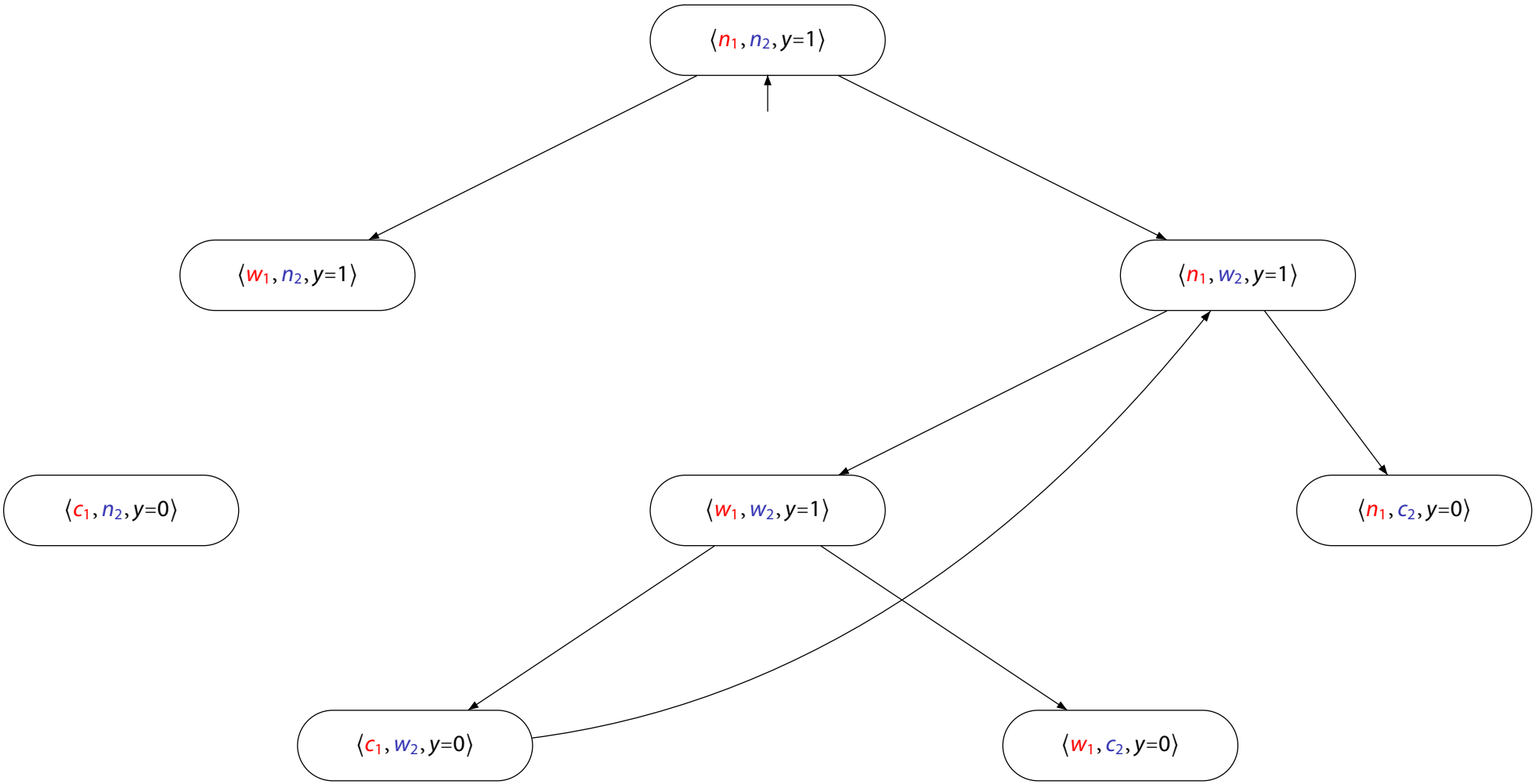
- Each path in G that leads from an initial state q_0 to a trivial terminal SCC $C = \{ q' \}$ with $q' \not\models \Psi$ provides a counterexample of the form $q_0\, q_1 \ldots q_n$ with $q_n \models \neg\Phi \wedge \neg\Psi$

Example



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

SCC graph



Time complexity

Let TS be a transition system TS with N states and K transitions and φ a CTL- path formula
If $TS \not\models \mathrm{A}\,\varphi$ 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 $TS \models \mathrm{E}\,\varphi$.

Summary of CTL model checking (1)

- CTL is a logic for formalizing properties over computation trees
- The expressiveness of LTL and CTL is incomparable
- Fairness constraints cannot be expressed in CTL
  - but are incorporated by adapting the CTL semantics such that quantification is over fair paths
- CTL model checking is by a recursive descent over parse tree of Φ
  - $Sat(\mathsf{E}\,(\Phi\,\mathsf{U}\,\Psi))$ is determined using a least fixed point computation
  - $Sat(\mathsf{E}\,\mathsf{G}\,\Phi)$ is determined by a greatest fixed point computation

Summary of CTL model checking (2)

- Time complexity of CTL model-checking $TS \models \Phi$ is:
  - is linear in $|TS|$ and $|\Phi|$ and linear in k for k fairness constraints
- Checking $TS \models_{fair} \Phi$ is $TS \models \Phi$ plus computing $Sat_{fair}(\mathsf{E}\,\mathsf{G}\,a)$
- Counterexamples and witnesses for CTL path-formulas can be determined using graph algorithms
- CTL* is more expressive than both CTL and LTL
- The CTL* model-checking problem can be solved by an appropriate combination of the CTL and the LTL model-checking algorithm
- The CTL*-model checking problem is PSPACE-complete

Symbolic Model Checking

Boolean functions

- Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ for $n \geq 0$ where $\mathbb{B} = \{ 0, 1 \}$
  - examples: $f(x_1, x_2) = x_1 \ \wedge \ (x_2 \ \vee \ \neg x_1)$, and $f(x_1, x_2) = x_1 \leftrightarrow x_2$
- Finite sets are boolean functions
  - let $|S| = N$ and $2^{n-1} < N \leq 2^n$
  - encode any element $s \in S$ as boolean vector of length n: $[\![\ ]\!] : S \to \mathbb{B}^n$
  - $T \subseteq S$ is represented by f_T such that:

$$f_T([\![s]\!]) = 1 \quad \text{iff} \quad s \in T$$

  - this is the characteristic function of T
- Relations are boolean functions
  - $\mathcal{R} \subseteq S \times S$ is represented by $f_{\mathcal{R}}$ such that:

$$f_R([\![s]\!], [\![t]\!]) = 1 \quad \text{iff} \quad (s, t) \in \mathcal{R}$$

Representing boolean functions

- Truth tables
  - very space inefficient (2^n lines)
  - satisfiability and equivalence check: easy; boolean operations also easy
  - . . . but have to consider exponentially many lines (so are hard)
- Propositional formulas
  - more compact representation
  - satisfiability problem is NP-complete (Cook's theorem)
  - boolean operations are just syntactic operations
- . . . in Disjunctive Normal Form (DNF)
  - satisfiability is easy: find a disjunct that does have complementary literals
  - negation expensive (dnf of $\neg\Phi$ may be exponentially longer than Φ)
  - conjunction complicated ($\Phi \wedge (\Psi_1 \vee \Psi_2) \equiv (\Phi \wedge \Psi_1) \vee (\Phi \wedge \Psi_2)$
- . . . in Conjunctive Normal Form (CNF)

Representing boolean functions

representation	compact?	sat	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	easy	easy	easy
DNF	sometimes	easy	hard	easy	hard
CNF	sometimes	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard

Representing boolean functions

representation	compact?	sat	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	easy	easy	easy
DNF	sometimes	easy	hard	easy	hard
CNF	sometimes	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard
reduced ordered binary decision diagram	often	easy	medium	medium	easy