Formal Methods
Module II: Formal Verification
Ch. 06: **Symbolic Model Checking**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: https://disi.unitn.it/rseba/DIDATTICA/fm2023/
Teaching assistant: Giuseppe Spallitta – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2022-2023

last update: Monday 1st May, 2023, 17:17

Outline

Outline

Outline

The Main Problem of M.C.: State Space Explosion

- The bottleneck:
  - Exhaustive analysis may require to store all the states of the Kripke structure, and to explore them one-by-one
  - The state space may be exponential in the number of components and variables
    - E.g., 300 Boolean vars $\Longrightarrow$ up to $2^{300} \approx 10^{100}$ states!
  - State Space Explosion:
    - too much memory required
    - too much CPU time required to explore each state
- A solution: Symbolic Model Checking

Symbolic Model Checking

Symbolic representation:

- manipulation of sets of states (rather than single states);
- sets of states represented by formulae in propositional logic;
  - set cardinality not directly correlated to size
- expansion of sets of transitions (rather than single transitions);

Symbolic Model Checking [cont.]

- Two main symbolic techniques:
  - Ordered Binary Decision Diagrams (OBDDs)
  - Propositional Satisfiability Checkers (SAT solvers)
- Different model checking algorithms:
  - Fix-point Model Checking (historically, for CTL)
  - Fix-point Model Checking for LTL (conversion to fair CTL MC)
  - Bounded Model Checking (historically, for LTL)
  - Invariant Checking
  - ...

Symbolic Representation of Kripke Models

- Symbolic representation:
  - sets of states as their characteristic function (Boolean formula)
  - provide logical representation and transformations of characteristic functions
- Example:
  - three state variables x_1, x_2, x_3:
    { 000, 001, 010, 011 } represented as "first bit false": $\neg x_1$
  - with five state variables x_1, x_2, x_3, x_4, x_5:
    { 00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111,. . . , 01111 } still represented as "first bit false": $\neg x_1$

Kripke Models in Propositional Logic

- Let $M = (S, I, R, L, AF)$ be a Kripke model
- States $s \in S$ are described by means of an array V of Boolean state variables.
- A state is a truth assignment to each atomic proposition in V.
  - 0100 is represented by the formula $(\neg x_1 \wedge x_2 \wedge \neg x_3 \wedge \neg x_4)$
  - we call $\xi(s)$ the formula representing the state $s \in S$
    (Intuition: $\xi(s)$ holds iff the system is in the state s)
- A set of states $Q \subseteq S$ can be represented by any formula which is logically equivalent to the formula $\xi(Q)$:

$$\bigvee_{s \in Q} \xi(s)$$

  (Intuition: $\xi(Q)$ holds iff the system is in one of the states $s \in Q$)
- Bijection between models of $\xi(Q)$ and states in Q

Remark

- Every propositional formula is a (typically very compact) representation of the set of assignments satisfying it
- Any formula equivalent to $\xi(Q)$ is a representation of Q
  $\Longrightarrow$ Typically Q can be encoded by much smaller formulas than $\bigvee_{s\in Q}\xi(s)$!
- Example: $Q = \{$ 00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, ..., 01111 $\}$ represented as "first bit false": $\neg x_1$

$$\bigvee_{s\in Q}\xi(s) = \left.\begin{array}{l} (\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge \neg x_5) \vee \\ (\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge \neg x_4 \wedge x_5) \vee \\ (\neg x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5) \vee \\ \ldots \\ (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5) \end{array}\right\} 2^4 \textit{disjuncts}$$

Symbolic Representation of Set Operators

One-to-one correspondence between sets and Boolean operators

- Set of all the states: $\xi(S) := \top$
- Empty set : $\xi(\emptyset) := \bot$
- Union represented by disjunction:
  $\xi(P \cup Q) := \xi(P) \vee \xi(Q)$
- Intersection represented by conjunction:
  $\xi(P \cap Q) := \xi(P) \wedge \xi(Q)$
- Complement represented by negation:
  $\xi(S/P) := \neg\xi(P)$

Symbolic Representation of Transition Relations

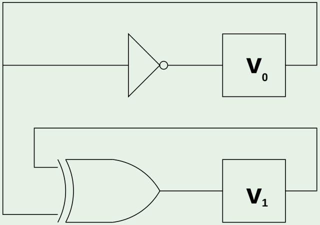
- The transition relation R is a set of pairs of states: $R \subseteq S \times S$
- A transition is a pair of states (s, s')
- A new vector of variables V' (the next state vector) represents the value of variables after the transition has occurred
- $\xi(s, s')$ defined as $\xi(s) \wedge \xi(s')$ (Intuition: $\xi(s, s')$ holds iff the system is in the state s and moves to state s' in next step)
- The transition relation R can be represented by any formula equivalent to:

$$\bigvee_{(s,s') \in R} \xi(s, s') = \bigvee_{(s,s') \in R} (\xi(s) \wedge \xi(s'))$$

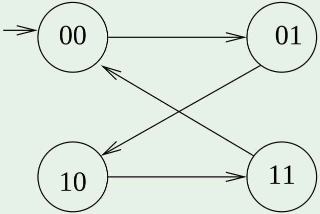
Each formula equivalent to $\xi(R)$ is a representation of R
$\Longrightarrow$ Typically R can be encoded by a much smaller formula than $\bigvee_{(s,s') \in R} \xi(s) \wedge \xi(s')$!

Example: a simple counter

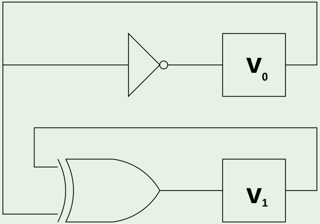
```
MODULE main
 VAR
   v0     : boolean;
   v1     : boolean;
   out    : 0..3;

 ASSIGN
   init(v0)  := 0;
   next(v0)  := !v0;

   init(v1)  := 0;
   next(v1)  := (v0 xor v1);

   out := toint(v0) + 2*toint(v1);
```



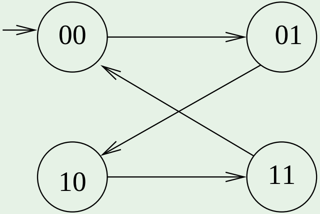
v_1	v_0	v_1'	v_0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0



Example: a simple counter [cont.]



v_1	v_0	v_1'	v_0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

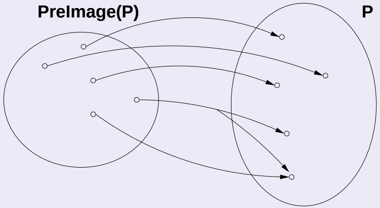


$$\xi(R) = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)$$

$$\begin{aligned}\bigvee_{(s,s')\in R} \xi(s) \wedge \xi(s') = \; & (\neg v_1 \wedge \neg v_0 \wedge \neg v_1' \wedge v_0') \vee \\ & (\neg v_1 \wedge v_0 \wedge v_1' \wedge \neg v_0') \vee \\ & (v_1 \wedge \neg v_0 \wedge v_1' \wedge v_0') \vee \\ & (v_1 \wedge v_0 \wedge \neg v_1' \wedge \neg v_0')\end{aligned}$$

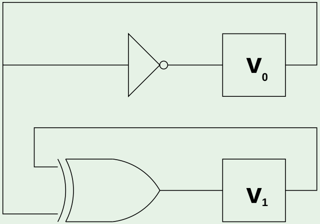
Pre-Image

- (Backward) pre-image of a set of states:

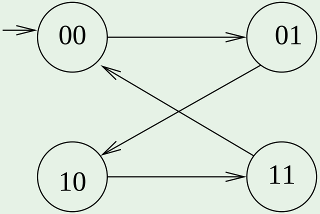


  Evaluate one-shot all transitions ending in the states of the set
- Set theoretic view: $PreImage(P, R) := \{s \mid \text{for some } s' \in P, (s, s') \in R\}$
- Logical view: $\xi(PreImage(P, R)) \; := \; \exists V'.(\xi(P)[V'] \wedge \xi(R)[V, V'])$
- μ over V is s.t $\mu \models \exists V'.(\xi(P)[V'] \wedge \xi(R)[V, V'])$ iff,
  for some μ' over V', we have: $\mu \cup \mu' \models (\xi(P)[V'] \wedge \xi(R)[V, V'])$,
  i.e., $\mu' \models \xi(P)[V']$ and $\mu \cup \mu' \models \xi(R)[V, V'])$
  - Intuition: $\mu \Longleftrightarrow s$, $\mu' \Longleftrightarrow s'$, $\mu \cup \mu' \Longleftrightarrow \langle s, s' \rangle$

Example: simple counter



v_1	v_0	v_1'	v_0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

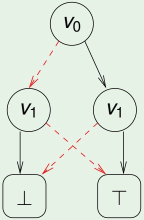


$\xi(R) = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)$

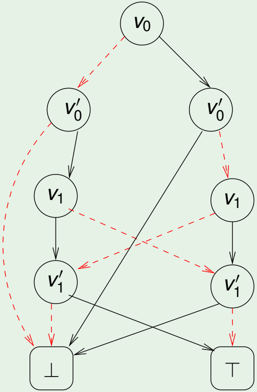
$\xi(P) := (v_0 \leftrightarrow v_1)$ (i.e., $P = \{00, 11\}$)

$$
\begin{aligned}
&\xi(PreImage(P, R)) &= \\
&\exists V'.(\xi(P)[V'] \wedge \xi(R)[V, V']) &= \\
&\exists v_0' v_1'.((v_0' \leftrightarrow v_1') \wedge (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)) &= \\
&\underbrace{(\neg v_0 \wedge v_0 \bigoplus v_1)}_{v_0'=\top, v_1'=\top} \vee \underbrace{\bot}_{v_0'=\top, v_1'=\bot} \vee \underbrace{\bot}_{v_0'=\bot, v_1'=\top} \vee \underbrace{(v_0 \wedge \neg(v_0 \bigoplus v_1))}_{v_0'=\bot, v_1'=\bot} &= \\
&v_1 \quad (i.e., \{10, 11\})
\end{aligned}
$$

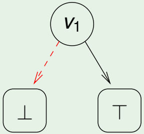
Pre-Image [cont.]



$\xi(P) = v_0 \leftrightarrow v_1$



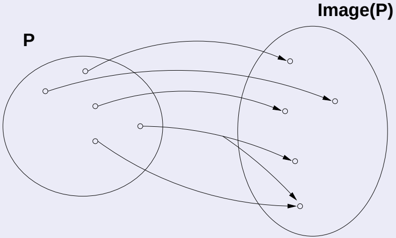
$\xi(R) = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)$



$\xi(PreImage(P, R)) =$
$\exists V'.((v_0' \leftrightarrow v_1') \wedge (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)) =$
v_1

Forward Image

- Forward image of a set:



Evaluate one-shot all transitions from the states of the set

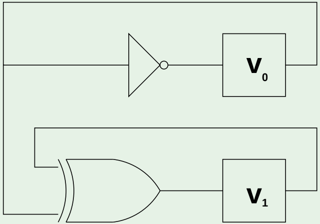
- Set theoretic view:

$$\mathit{Image}(P, R) := \{s' \mid \text{for some } s \in P, (s, s') \in R\}$$

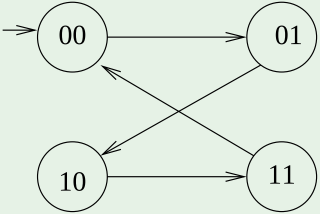
- Logical Characterization:

$$\xi(\mathit{Image}(P, R)) \ := \ \exists V.(\xi(P)[V] \wedge \xi(R)[V, V'])$$

Example: simple counter



v_1	v_0	v_1'	v_0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

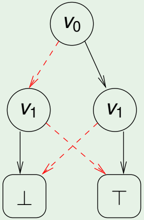


$\xi(R) = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)$

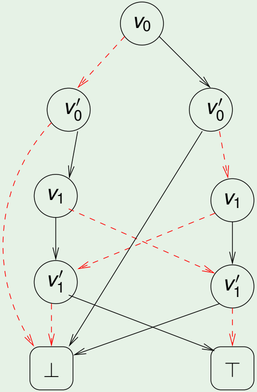
$\xi(P) := (v_0 \leftrightarrow v_1)$ (i.e., $P = \{00, 11\}$)

$$
\begin{aligned}
\xi(Image(P,R)) &= \exists V.(\xi(P)[V] \wedge \xi(R)[V, V']) \\
&= \exists V.((v_0 \leftrightarrow v_1) \wedge (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)) \\
&= ... \\
&= \neg v_1' \quad (i.e., \{00, 01\})
\end{aligned}
$$

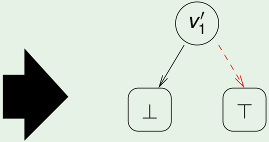
Forward Image [cont.]



$\xi(P) = v_0 \leftrightarrow v_1$



$\xi(R) = (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \bigoplus v_1)$



$\xi(Image(P, R)) =$
$\exists V.((v_0 \leftrightarrow v_1) \wedge (v'_0 \leftrightarrow \neg v_0) \wedge (v'_1 \leftrightarrow v_0 \bigoplus v_1)) =$
$\neg v'_1$

Application of the Transition Relation

- Image and PreImage of a set of states S computed by means of quantified Boolean formulae
- The whole set of transitions can be fired (either forward or backward) in one logical operation
- The symbolic computation of PreImage and Image provide the primitives for symbolic search of the state space of FSM's

Notation Remark

Henceforth, for readability sake, we omit the "$\xi()$" notation in symbolic representations of systems.

- Kripke models represented as $\langle I(V), R(V, V')\rangle$
- Fair Kripke models represented as $\langle I(V), R(V, V'), F(V)\rangle$ s.t. $F(V) \stackrel{\text{def}}{=} \{F_1(V), .., F_k(V)\}$ (see next section)

Outline

CTL MC Procedure

```
STATE-SET Check(CTL_formula β) {
    case β of
    ⊤:          return S;
    ⊥:          return ∅;
    ¬β1:        return S\Check(β1);
    β1 ∧ β2:    return (Check(β1) ∩ Check(β2));
    EXβ1:       return PreImage(Check(β1));
    EGβ1:       return Check_EG(Check(β1));
    E(β1Uβ2):   return Check_EU(Check(β1),Check(β2));
}
```

General Symbolic CTL MC Procedure

```
OBDD      Check(CTL_formula β) {
    if (In_OBDD_Hash(β)) return OBDD_Get_From_Hash(β);
    case β of
    ⊤:            return obdd_true;
    ⊥:            return obdd_false;
    ¬β1:          return ¬ Check(β1);
    β1 ∧ β2:      return (Check(β1) ∧ Check(β2));
    EXβ1:         return PreImage(Check(β1));
    EGβ1:         return Check_EG(Check(β1));
    E(β1Uβ2):     return Check_EU(Check(β1),Check(β2));
}
```

Ingredients

Some primitive functions from CLT Model Checking:

- Symbolic `Check_EX`(ϕ):
  returns an OBDD representing the set of states from which a path verifying $\mathbf{X}\phi$ holds
  (i.e., the symbolic preimage of the set of states where ϕ holds)
- Symbolic `Check_EG`(ϕ):
  returns an OBDD representing the set of states from which a path verifying $\mathbf{G}\phi$ holds
- Symbolic `Check_EU`(ϕ_1, ϕ_2):
  returns an OBDD representing the set of states from which a path verifying $\phi_1 \mathbf{U} \phi_2$ holds

Check_EX

Explicit-state

State Set Check_EX(**State Set** X)
    return $\{s \mid \text{for some } s' \in X, (s, s') \in R\}$;

Symbolic

OBDD Check_EX(**OBDD** X)
    return $\exists V'.(\ X[V'] \wedge R[V, V'])$;

Same as Pre-Image computation.

Check_EG

Explicit-State

State Set Check_EG(**State Set** X)
    $Y' := X$;
    repeat
        $Y := Y'$;
        $Y' := Y \cap Check\_EX(Y)$;
    until $(Y' = Y)$;
return Y;

Symbolic

OBDD Check_EG(**OBDD** X)
    $Y' := X$;
    repeat
        $Y := Y'$;
        $Y' := Y \wedge Check\_EX(Y)$;
    until $(Y' \leftrightarrow Y)$;
return Y;

Hint (tableaux rule): $s \models \mathbf{EG}\phi$ only if $s \models \phi \wedge \mathbf{EXEG}\phi$

Check_EU

Explicit-State

State Set Check_EU(**State Set** X_1,X_2)
  $Y' := X_2$;
  repeat
    $Y := Y'$;
    $Y' := Y \cup (X_1 \cap Check\_EX(Y))$;
  until $(Y' = Y)$;
return Y;

Symbolic

OBDD Check_EU(**OBDD** X_1,X_2)
  $Y' := X_2$;
  repeat
    $Y := Y'$;
    $Y' := Y \vee (X_1 \wedge Check\_EX(Y))$;
  until $(Y' \leftrightarrow Y)$;
return Y;

Hint (tableaux rule): $s \models \mathbf{E}(\phi_1 \mathbf{U} \phi_2)$ if $s \models \phi_2 \vee (\phi_1 \wedge \mathbf{EXE}(\phi_1 \mathbf{U} \phi_2))$

Outline

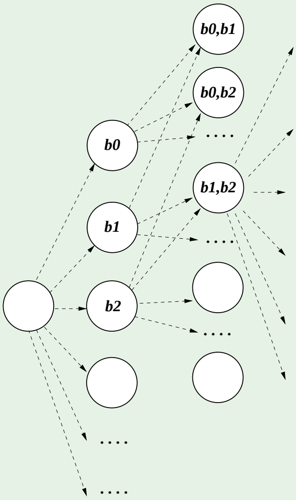
A simple example

```
MODULE main
VAR
  b0 : boolean;
  b1 : boolean;
  ...
ASSIGN
  init(b0) := 0;
  next(b0) := case
    b0  : 1;
    !b0 : {0,1};
  esac;
  init(b1) := 0;
  next(b1) := case
    b1  : 1;
    !b1 : {0,1};
  esac;
  ...
```

A simple example [cont.]

- N Boolean variables $b0, b1, \ldots$
- Initially, all variables set to 0
- Each variable can pass from 0 to 1, but not vice-versa
- 2^N states, all reachable
- (Simplified) model of a student career behaviour.

A simple example: FSM

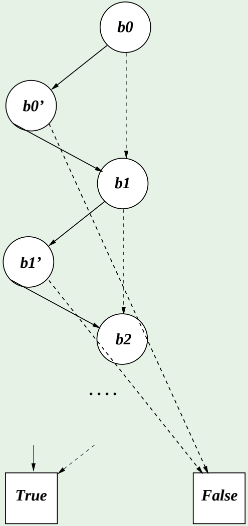


(transitive transitons omitted)

2^N STATES

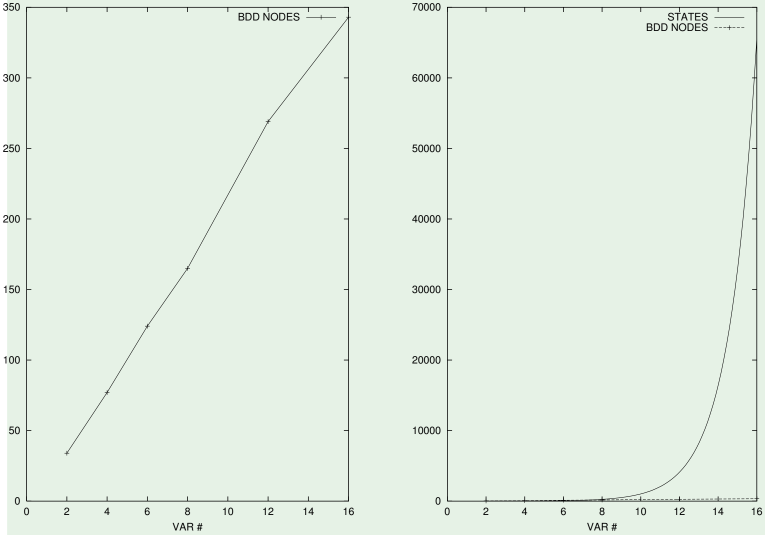
$O(2^N)$ TRANSITIONS

A simple example: $OBDD(\xi(R))$



$2N + 2$ NODES

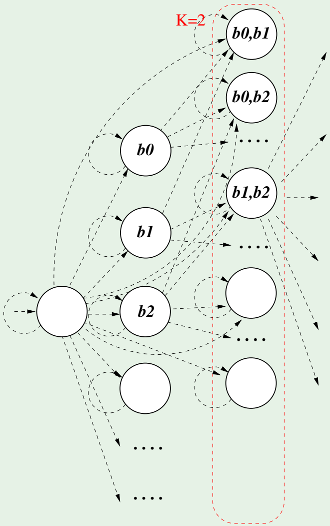
A simple example: states vs. OBDD nodes [NuSMV.2]



A simple example: reaching K bits true

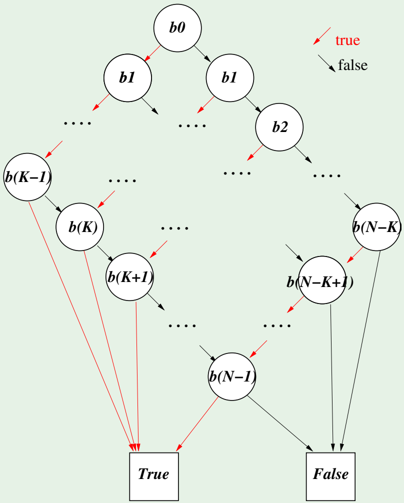
- Property $\mathbf{EF}(b0 + b1 + ... + b(N-1) \geq K)$ $(K \leq N)$
  (it may be reached a state in which K bits are true)
- E.g.: "it is reachable a state where K exams are passed"

A simple example: FSM



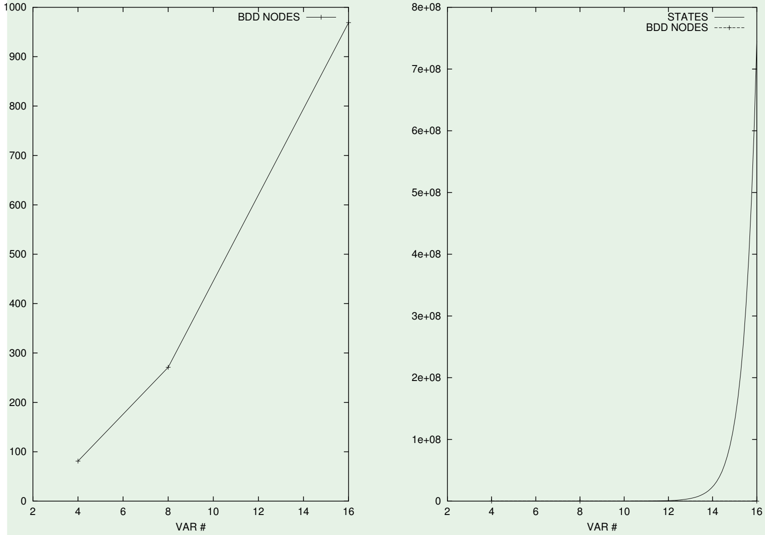
$$\binom{N}{K} + \binom{N}{K+1} + \ldots + \binom{N}{N}$$

A simple example: $OBDD(\xi(\varphi))$



$(N - K + 1) \cdot K + 2$ NODES

A simple example: states vs. OBDD nodes [NuSMV.2]



Outline

Outline

The Need for Fairness Conditions: Intuition

Consider a public restroom. A standard access policy is “first come first served” (e.g., a queue-based protocol).

- Does this policy guarantee that everybody entering the queue will eventually access the restroom?
  - **No**: in principle, somebody might remain in the restroom forever, hindering the access to everybody else
  - In practice, it is considered reasonable to assume that everybody exits the restroom after a finite amount of time

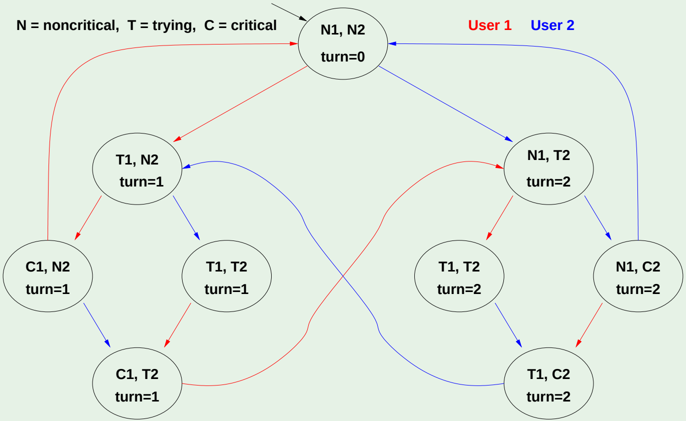
$\Longrightarrow$ It is reasonable enough to assume the protocol suitable under the condition that each user is infinitely often outside the restroom

- Such a condition is called fairness condition

The Need for Fairness Conditions: An Example

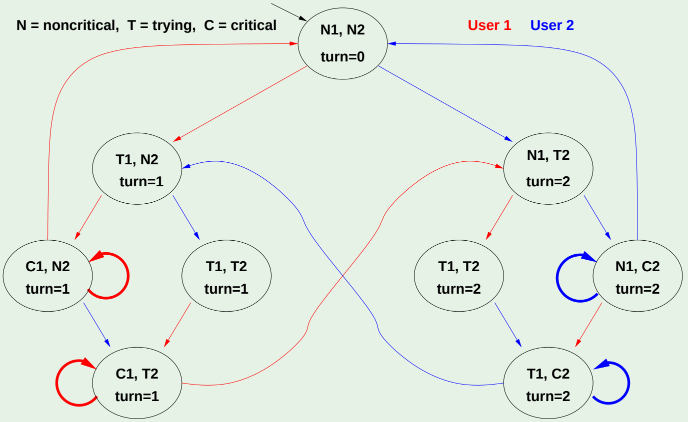
- Consider a variant of the mutual exclusion in which one process can stay permanently in the critical zone
- Do $M \models \mathbf{G}(T_1 \rightarrow \mathbf{F} C_1)$, $M \models \mathbf{G}(T_2 \rightarrow \mathbf{F} C_2)$ still hold?

The Need for Fairness Conditions: An Example [cont.]



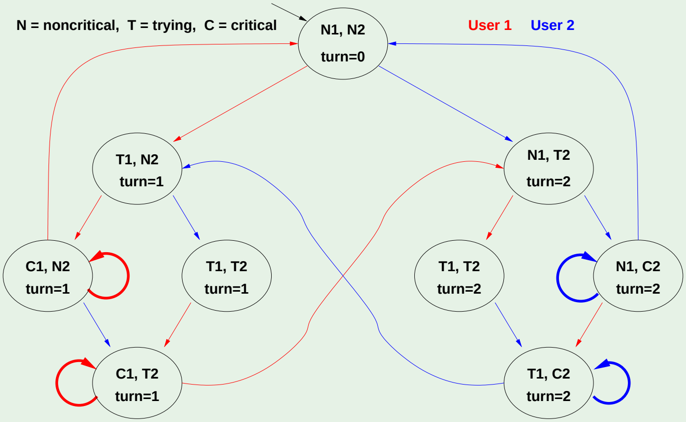
$M \models \mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$ $\qquad$ $M \models \mathbf{G}(T_2 \rightarrow \mathbf{F}C_2)$

The need for fairness conditions: an example [cont.]



$M \models \mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$? $M \models \mathbf{G}(T_2 \rightarrow \mathbf{F}C_2)$?

The need for fairness conditions: an example [cont.]



$\mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$? $\mathbf{G}(T_2 \rightarrow \mathbf{F}C_2)$?

NO: E.g., it can cycle forever in $\{C_1, T_2, turn = 1\}$

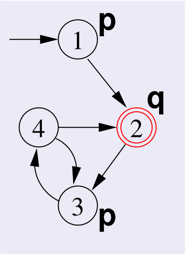
$\Longrightarrow$ Unfair protocol: one process might never be served

Fairness Conditions

- It is desirable that certain (typically Boolean) conditions φ's hold infinitely often: **GF**φ
- **GF**φ is called fairness condition
- Intuitively, fairness conditions are used to eliminate behaviours in which a certain condition φ never holds:
  GFφ: "it is never reached a state from which φ is forever false"
- Example: it is not desirable that, once a process is in the critical section, it never exits: **GF**$\neg C_1$
- A fair condition φ_i can be represented also by the set f_i of states where φ_i holds ($f_i := \{s : \pi, s \models \varphi_i, \textit{ for each } \pi \in M\}$)

Fair Kripke models

- A Fair Kripke model $M_F := \langle S, R, I, AP, L, F \rangle$ consists of:
  - a set of states S;
  - a set of initial states $I \subseteq S$;
  - a set of transitions $R \subseteq S \times S$;
  - a set of atomic propositions AP;
  - a labeling function $L : S \longmapsto 2^{AP}$;
  - a set of fairness conditions $F = \{f_1, \ldots, f_n\}$, with $f_i \subseteq S$.



- E.g., $\{\{2\}\} := \{\{s : L(s) = \{q\}\}\} = \{\mathbf{GF}q\}$ is the set of fairness conditions of the Kripke model above
- Fair path π: at least one state for each f_i occurs infinitely often in π (φ_i holds infinitely often in π: $\pi \models \mathbf{GF}\varphi_i$)
  - E.g., every path visiting infinitely often state 2 is a fair path.
- Fair state: a state through which at least one fair path passes
  - E.g., all states 1,2,3,4 are fair states
- Note: fair state $\neq$ state belonging to a fairness condition

Outline

CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no

CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no

CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no

CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

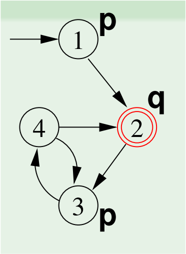
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}true$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}true$? yes
- $M_f \models \mathbf{G}(p \to \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \to \mathbf{F}q)$? no



CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

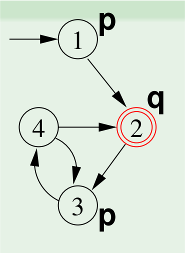
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no



CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

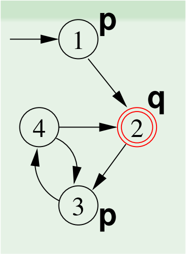
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}true$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}true$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no



CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

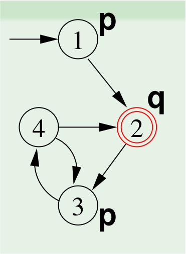
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}true$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}true$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no



CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

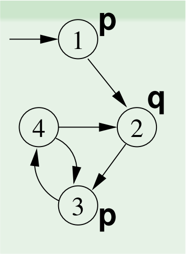
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no



CTL M.C. with Fair Kripke Models

Fair Kripke Models restrict the M.C. process to fair paths:

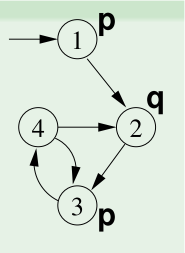
- $M_f \models \varphi$ iff $\pi \models \varphi$ for every fair path π
- Path quantifiers (from CTL) apply only to fair paths:
  - $M_F, s \models \mathbf{A}\varphi$ iff $\pi, s \models \varphi$ for every fair path π s.t. $s \in \pi$
  - $M_F, s \models \mathbf{E}\varphi$ iff $\pi, s \models \varphi$ for some fair path π s.t. $s \in \pi$

$\Longrightarrow$ a fair state s is a state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

- We need a procedure to compute the set of fair states: `Check_FairEG(true)`

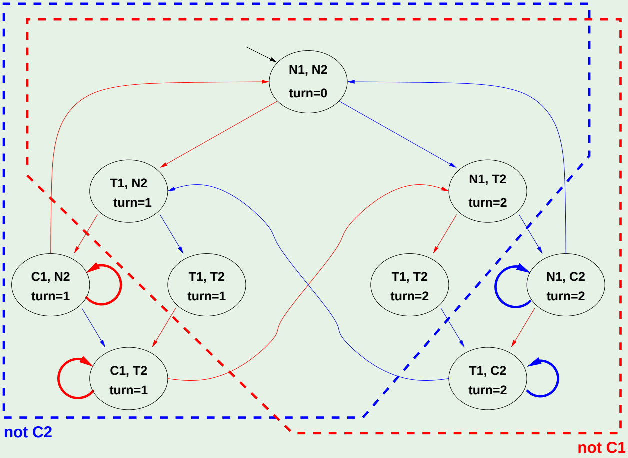
Example

- $M_f \models \mathbf{EG}\mathit{true}$? yes
- $M_f \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? yes
- $M \models \mathbf{G}(p \rightarrow \mathbf{F}q)$? no



Fair CTL Model Checking: Example

F := {{ not C1},{not C2}}



$M_F \models \mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$? $M_F \models \mathbf{G}(T_2 \rightarrow \mathbf{F}C_2)$? YES: every fair path satisfies the conditions

CTL M.C. vs. LTL M.C. with Fair Kripke Models

Remark: fair CTL M.C.

When model checking a CTL formula ψ, fairness conditions cannot be encoded into the formula:

$$M_{\{f_1,\ldots,f_n\}} \models \psi \not\Longleftrightarrow \; M \models (\bigwedge_{i=1}^{n} \mathbf{AGAF} f_i) \rightarrow \psi.$$

$$M_{\{f_1,\ldots,f_n\}} \models \psi \not\Longleftrightarrow \; M \models (\bigwedge_{i=1}^{n} \mathbf{EGEF} f_i) \rightarrow \psi.$$

$\Longrightarrow$ We need specific procedures for Fair CTL Model Checking.

Remark: fair LTL M.C.

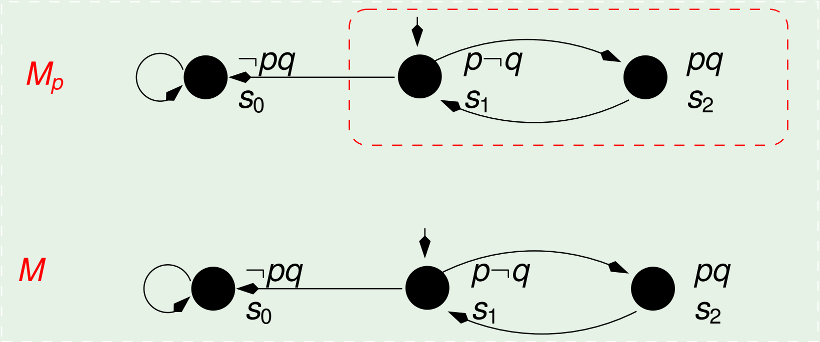
When model checking an LTL formula ψ, fairness conditions can be encoded into the formula:

$$M_{\{f_1,\ldots,f_n\}} \models \psi \Longleftrightarrow M \models (\bigwedge_{i=1}^{n} \mathbf{GF} f_i) \rightarrow \psi.$$

$\Longrightarrow$ There is no need for Fair LTL Model Checking procedures.

Ex. CTL: $M_{\{f_1,\ldots,f_n\}} \models \psi \;\not\Longleftrightarrow\; M \models (\bigwedge_{i=1}^{n} \mathbf{AGAF} f_i) \rightarrow \psi.$

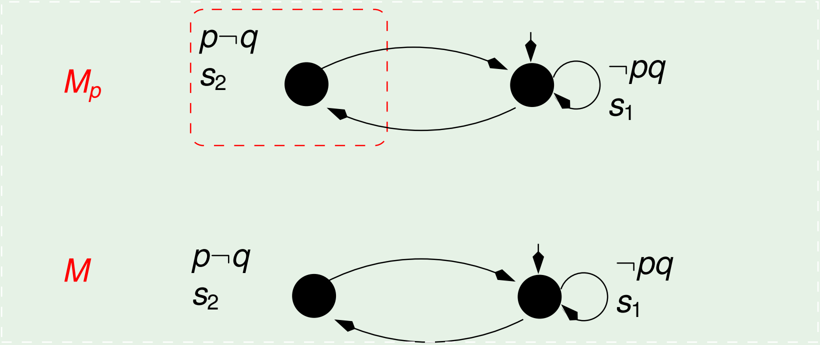
[Example provided by the student Davide Kirchner, 2014]



- $M_p \not\models \mathbf{AG}q$
- $M \models (\mathbf{AGAF}p) \rightarrow \mathbf{AG}q$

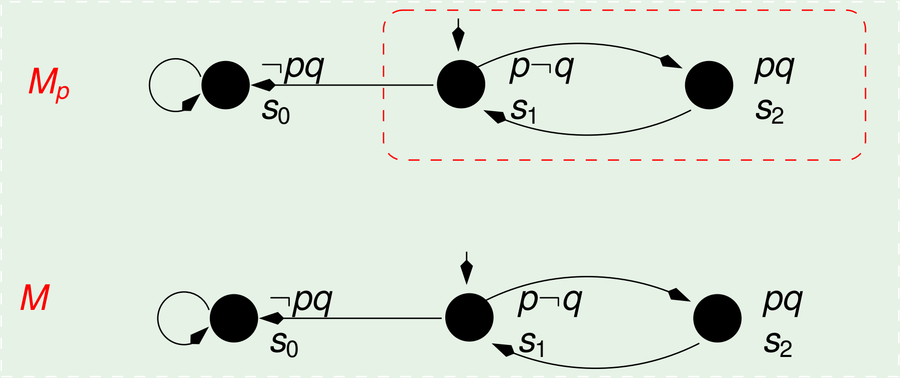
Ex. CTL: $M_{\{f_1,\ldots,f_n\}} \models \psi \not\Longleftrightarrow M \models (\bigwedge_{i=1}^{n} \mathbf{EGEF} f_i) \to \psi$.

[Example provided by the student Daniele Giuliani, 2019]



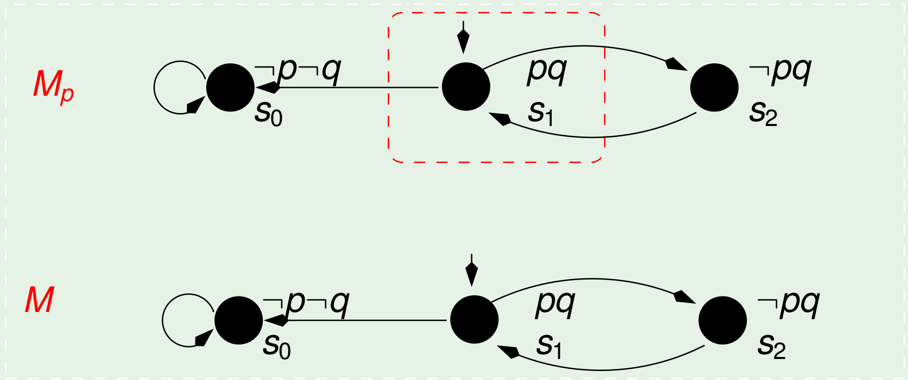
- $M_p \not\models \mathbf{EFEG}q$
- $M \models (\mathbf{EGEF}p) \to \mathbf{EFEG}q$

Ex. LTL (1): $M_{\{f_1,\ldots,f_n\}} \models \psi \iff M \models (\bigwedge_{i=1}^{n} \mathbf{GF} f_i) \rightarrow \psi$.



- $M_p \not\models \mathbf{G} q$
- $M \not\models (\mathbf{GF} p) \rightarrow \mathbf{G} q$

Ex. LTL (2): $M_{\{f_1,\ldots,f_n\}} \models \psi \iff M \models (\bigwedge_{i=1}^{n} \mathbf{GF} f_i) \rightarrow \psi$.



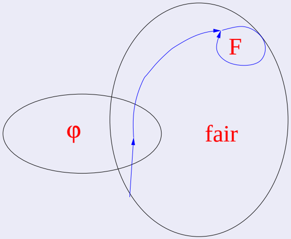
- $M_p \models \mathbf{G}q$
- $M \models (\mathbf{GF}p) \rightarrow \mathbf{G}q$

Fair CTL Model Checking

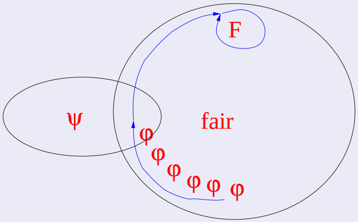
- In order to solve the fair CTL model checking problem, we must be able to compute:
  - $[\varphi_f]$ s.t. φ Boolean (i.e. $[\varphi]$ under fairness conditions f)
  - $[\mathbf{E}_f\mathbf{X}(\varphi)]$ (i.e. $[\mathbf{EX}\varphi]$ under fairness conditions f)
  - $[\mathbf{E}_f(\varphi\mathbf{U}\psi)]$ (i.e. $[\mathbf{E}(\varphi\mathbf{U}\psi)]$ under fairness conditions f)
  - $[\mathbf{E}_f\mathbf{G}\varphi]$ (i.e. $[\mathbf{EG}\varphi]$ under fairness conditions f).
- Suppose we have a procedure `Check_FairEG` to compute $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Let $fair \stackrel{\text{def}}{=} \mathbf{E}_f\mathbf{G}true$. ($M, s \models \mathbf{E}_f\mathbf{G}true$ if s is a fair state.)
- if φ is Boolean, then $M_f, s \models \varphi$ iff $M, s \models (\varphi \wedge fair)$
- We can rewrite all the other fair operators:
  - $\mathbf{E}_f\mathbf{X}(\varphi) \equiv \mathbf{EX}(\varphi \wedge fair)$
  - $\mathbf{E}_f(\varphi\mathbf{U}\psi) \equiv \mathbf{E}(\varphi\mathbf{U}(\psi \wedge fair))$

Fair CTL Model Checking

- $\mathbf{E}_f\mathbf{X}(\varphi) \equiv \mathbf{EX}(\varphi \wedge \mathit{fair})$:



- $\mathbf{E}_f(\varphi\mathbf{U}\psi) \equiv \mathbf{E}(\varphi\mathbf{U}(\psi \wedge \mathit{fair}))$:



Language-Emptiness Checking for Fair Kripke Models

`Fair_CheckEG`

Given: a fair Kripke model $M_F := \langle S, R, I, AP, L, F\rangle$ and a CTL formula φ s.t. $[\varphi] \subseteq S$, `Fair_CheckEG`(φ) returns the subset of the states s in $[\varphi]$ from which at least one fair path π entirely included in $[\varphi]$ passes through

Symbolic `Fair_CheckEG`

Given: the symbolic representation of a fair Kripke model $M_F := \langle I, R, F\rangle$
and a Boolean formula (OBDD) Ψ,
`Fair_CheckEG`(Ψ) returns a Boolean formula (OBDD) representing the subset of the states s in Ψ from which at least one fair path π entirely included in Ψ passes through

`Fair_CheckEG`(*true*) computes (the symbolic representation of) the set of fair states of M_f
$\Longrightarrow$ $I \subseteq$ `Fair_CheckEG`(*true*) iff $\mathcal{L}(M_f) \neq \emptyset$

Ingredients (from CTL Model Checking)

Some primitive functions from CLT Model Checking:

- Symbolic `Check_EX`(ϕ): returns an OBDD representing the set of states from which a path verifying $\mathbf{X}\phi$ holds

  (i.e., the symbolic preimage of the set of states where ϕ holds)
- Symbolic `Check_EG`(ϕ): returns an OBDD representing the set of states from which a path verifying $\mathbf{G}\phi$ holds
- Symbolic `Check_EU`(ϕ_1, ϕ_2): returns an OBDD representing the set of states from which a path verifying $\phi_1 \mathbf{U} \phi_2$ holds

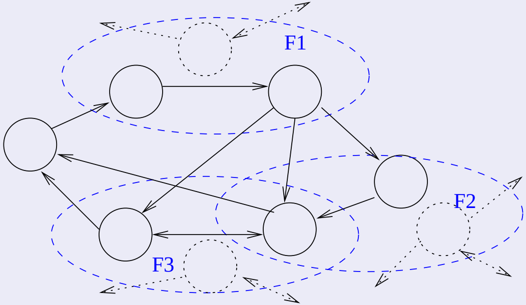
Outline

SCC-based Check_FairEG

A Strongly Connected Component (SCC) of a directed graph is a maximal subgraph s.t. all its nodes are reachable from each other.

Given a fair Kripke model M, a fair non-trivial SCC is an SCC with at least one edge that contains at least one state for every fair condition

$\Longrightarrow$ all states in a fair (non-trivial) SCC are fair states



SCC-based Check_FairEG (cont.)

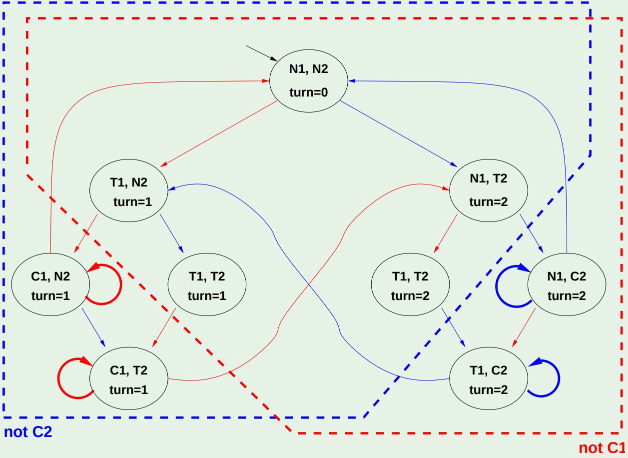
`Check_FairEG(`$[\phi]$`):`

(i) restrict the graph of M to $[\phi]$;
(ii) find all fair non-trivial SCCs C_i
(iii) build $C := \cup_i C_i$;
(iv) compute the states that can reach C (`Check_EU(`$[\phi]$`,` C`)`).

$[\phi]$: set of states where ϕ holds (aka denotation of ϕ)

Example: Check_FairEG

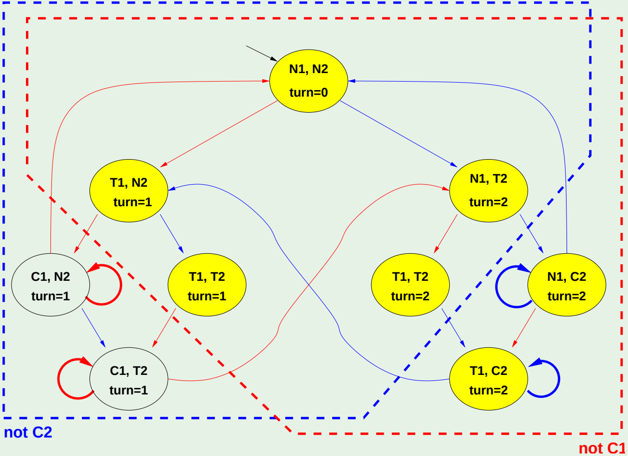
F := {{ not C1},{not C2}}



EG

Example: Check_FairEG

F := {{ not C1},{not C2}}

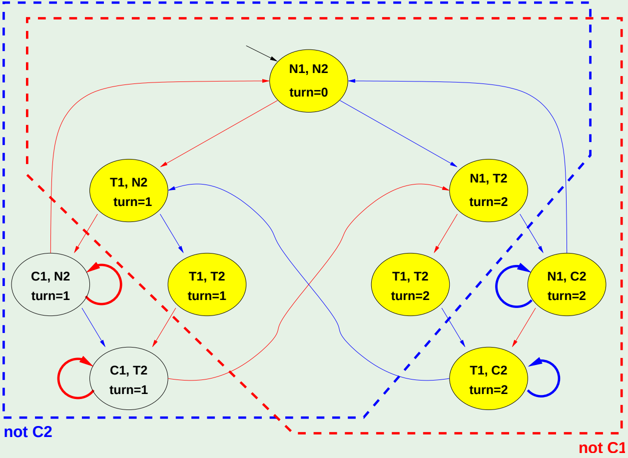


Check_FairEG($\neg C_1$): 1. compute $[\neg C_1]$

EG

Example: Check_FairEG

F := {{ not C1},{not C2}}

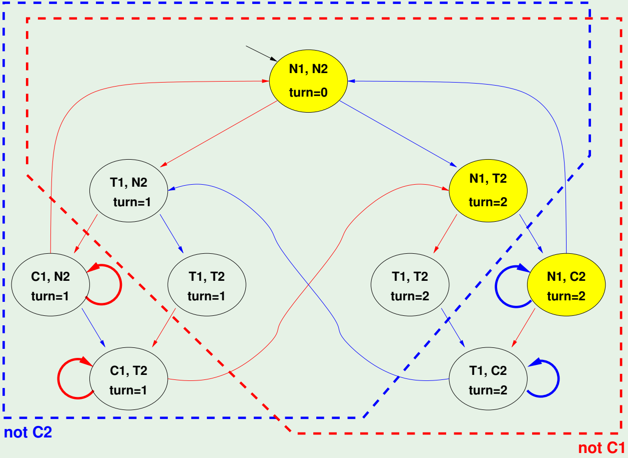


Check_FairEG($\neg C_1$): 2. restrict the graph to $[\neg C_1]$

EG

Example: Check_FairEG

F := {{ not C1},{not C2}}

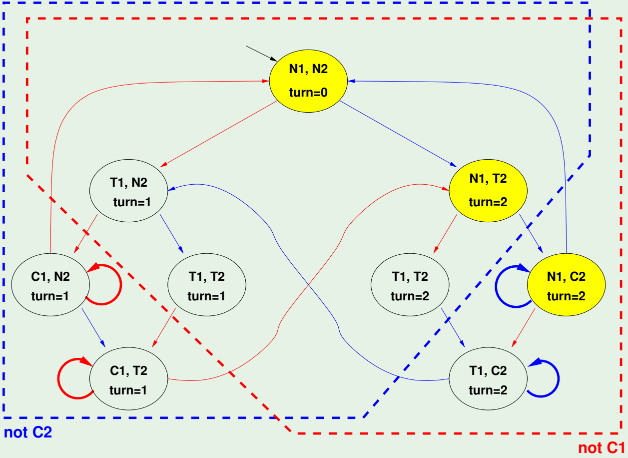


Check_FairEG($\neg C_1$): 3. find all fair non-trivial SCC's

EG

Example: Check_FairEG

F := {{ not C1},{not C2}}

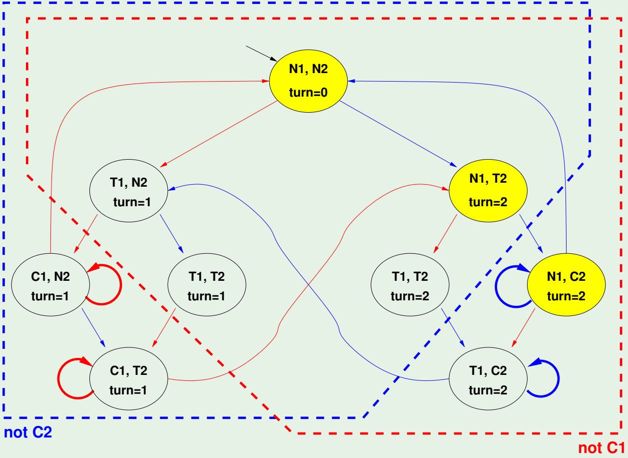


Check_FairEG($\neg C_1$): 4. build the union C of all SCC's

EG

Example: Check_FairEG

F := {{ not C1},{not C2}}



EG

Check_FairEG($\neg C_1$): 5. compute the states which can reach it

SCC-based Check_FairEG - Drawbacks

- SCCs computation requires a linear ($O(\#nodes + \#edges)$) DFS (Tarjan).
- The DFS manipulates the states explicitly, storing information for every state.
- A DFS is not suitable for symbolic model checking where we manipulate sets of states.

$\Longrightarrow$ We want an algorithm based on (symbolic) preimage computation.

Outline

Emerson-Lei Algorithm

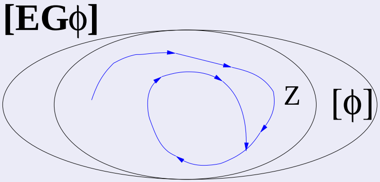
Fixpoint characterization of **EG** and fair **EG**

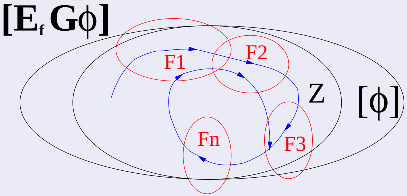
"[ϕ]" denotes the set of states where ϕ holds

- Theorem (Emerson & Clarke): $[\mathbf{EG}\phi] = \nu Z.([\phi] \cap [\mathbf{EX}Z])$
  The greatest set Z s.t. every state z in Z satisfies ϕ and reaches another state in Z in one step.

We can characterize fair **EG** (aka "$\mathbf{E}_f\mathbf{G}$") similarly:

- Theorem (Emerson & Lei): $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \cap F_i))])$
  The greatest set Z s.t. every state z in Z satisfies ϕ and, for every set $F_i \in FT$, z reaches a state in $F_i \cap Z$ by means of a non-trivial path that lies in Z.





Emerson-Lei Algorithm

Recall: $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \cap F_i))])$

```
state_set Check_FairEG( state_set [φ]) {
    Z' := [φ];
    repeat
        Z:= Z';
        for each F_i in FT
            Y:= Check_EU(Z,F_i ∩ Z);
            Z' := Z' ∩ PreImage(Y));
        end for;
    until (Z' = Z);
    return Z;
}
```

Implementation of the above formula

Emerson-Lei Algorithm

Recall: $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \cap F_i))])$

```
state_set Check_FairEG( state_set [φ]) {
    Z' := [φ];
    repeat
        Z:= Z';
        for each F_i in FT
            Y:= Check_EU(Z',F_i ∩ Z');
            Z' := Z' ∩ PreImage(Y));
        end for;
    until (Z' = Z);
    return Z;
}
```

Slight improvement: do not consider states in $Z \setminus Z'$

Emerson-Lei Algorithm (symbolic version)

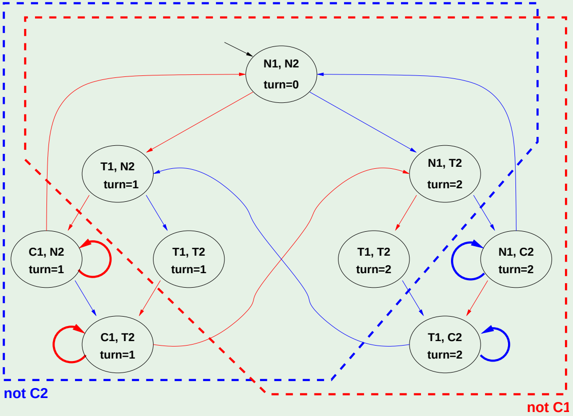
Recall: $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \wedge F_i))])$

```
Obdd Check_FairEG( Obdd φ) {
    Z′ := φ;
    repeat
        Z := Z′;
        for each F_i in FT
            Y := Check_EU(Z′, F_i ∧ Z′);
            Z′ := Z′ ∧ PreImage(Y));
        end for;
    until (Z′ ↔ Z);
    return Z;
}
```

Symbolic version.

Example: Check_FairEG

F := { { not C1},{not C2}}

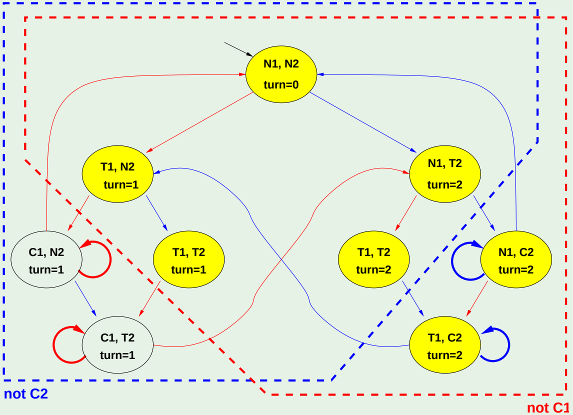


$\mathbf{E}_f\mathbf{G}\neg C_1$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}

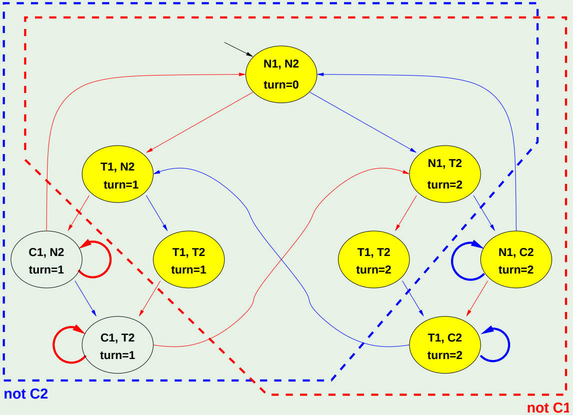


$\mathbf{E}_f\mathbf{G}\neg C_1$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



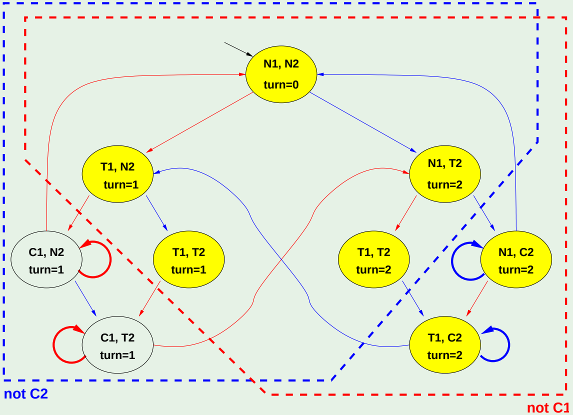
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



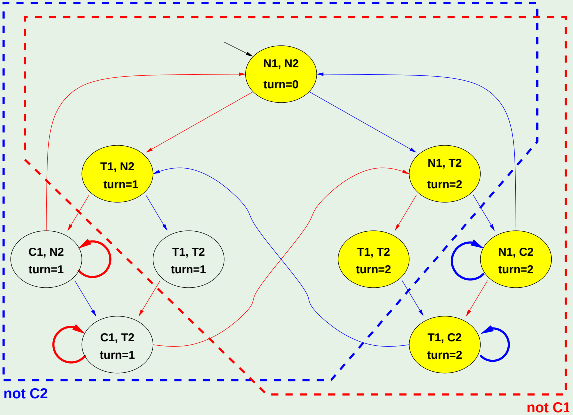
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



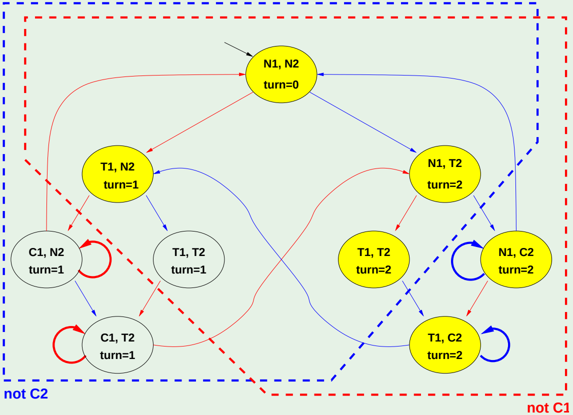
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



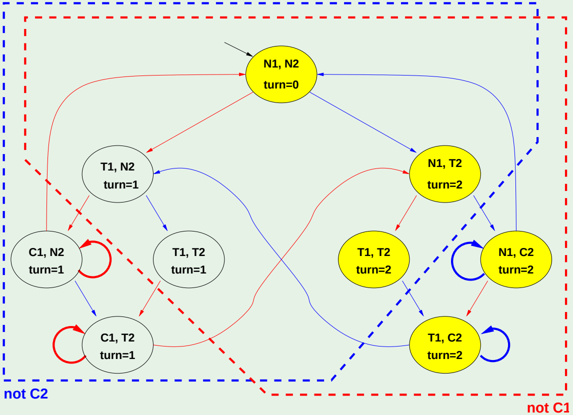
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



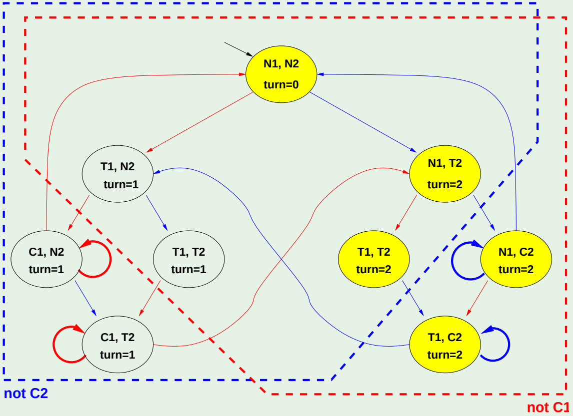
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



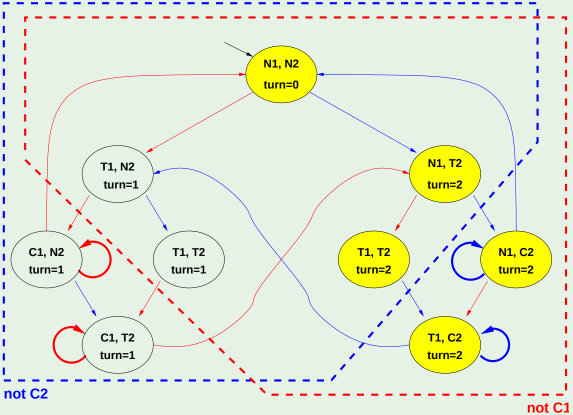
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



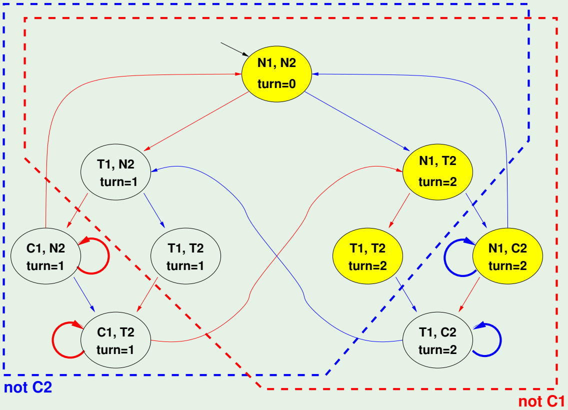
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



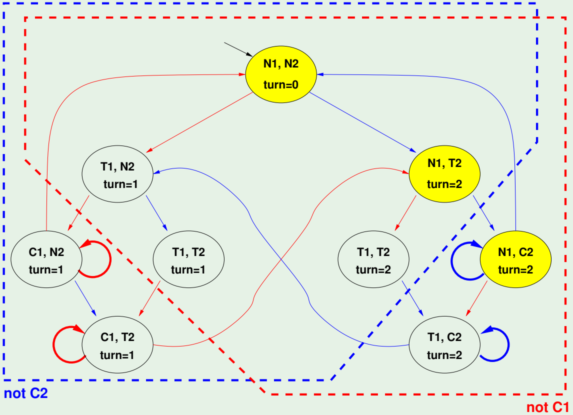
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



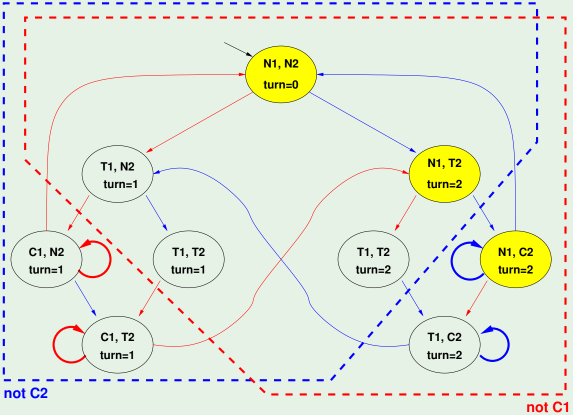
$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Example: Check_FairEG

F := { { not C1},{not C2}}



$\mathbf{E}_f\mathbf{G}\neg C_1$

$\mathbf{E}_f\mathbf{G}g = \nu Z.g \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_1)) \wedge \mathbf{EXE}(Z\mathbf{U}(Z \wedge F_2))$

Fixpoint reached

Outline

Outline

Symbolic LTL Satisfiability and Entailment

LTL Validity/Satisfiability

- Let ψ be an LTL formula

$$
\begin{array}{ll}
 & \models \psi \quad \text{(LTL)} \\
\Longleftrightarrow & \neg\psi \text{ unsat} \\
\Longleftrightarrow & \mathcal{L}(T_{\neg\psi}) = \emptyset
\end{array}
$$

- $T_{\neg\psi}$ is a fair Kripke model (aka tableaux) which represents all and only the paths that satisfy $\neg\psi$ (do not satisfy ψ)

LTL Entailment

- Let φ, ψ be an LTL formula

$$
\begin{array}{ll}
 & \varphi \models \psi \quad \text{(LTL)} \\
 & \models \varphi \rightarrow \psi \quad \text{(LTL)} \\
\Longleftrightarrow & \varphi \wedge \neg\psi \text{ unsat} \\
\Longleftrightarrow & \mathcal{L}(T_{\varphi\wedge\neg\psi}) = \emptyset
\end{array}
$$

- $T_{\varphi\wedge\neg\psi}$ is a fair Kripke model (aka tableaux) which represents all and only the paths that satisfy $\varphi \wedge \neg\psi$ (satisfy φ and do not satisfy ψ)

Symbolic LTL Model Checking

LTL Model Checking

- Let M be a Kripke model and ψ be an LTL formula

$$
\begin{array}{ll}
 & M \models \psi \quad \text{(LTL)} \\
\Longleftrightarrow & \mathcal{L}(M) \subseteq \mathcal{L}(\psi) \\
\Longleftrightarrow & \mathcal{L}(M) \cap \overline{\mathcal{L}(\psi)} = \emptyset \\
\Longleftrightarrow & \mathcal{L}(M) \cap \mathcal{L}(\neg\psi) = \emptyset \\
\Longleftrightarrow & \mathcal{L}(M) \cap \mathcal{L}(T_{\neg\psi}) = \emptyset \\
\Longleftrightarrow & \mathcal{L}(M \times T_{\neg\psi}) = \emptyset
\end{array}
$$

- $T_{\neg\psi}$ is a fair Kripke model (aka tableaux) which represents all and only the paths that satisfy $\neg\psi$ (do not satisfy ψ)

$\Longrightarrow$ $M \times T_{\neg\psi}$ represents all and only the paths appearing in M and not in ψ.

Symbolic LTL Model Checking

Three steps

Let $\varphi \stackrel{\text{def}}{=} \neg\psi$:

(i) Compute T_φ

(ii) Compute the product $M \times T_\varphi$

(iii) Check the emptiness of $\mathcal{L}(M \times T_\varphi)$

Outline

The Set of States

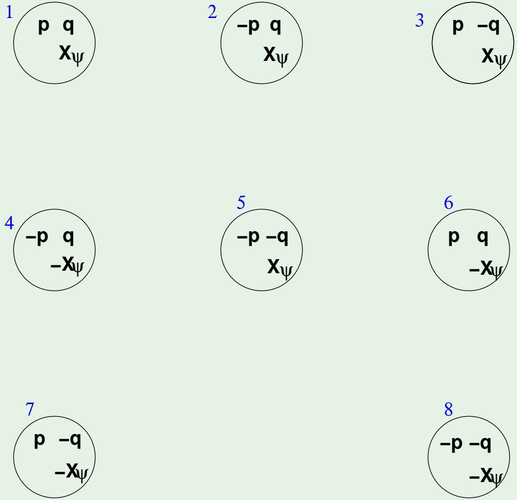
- Elementary subformulas of ψ: $el(\psi)$
  - $el(p) := \{p\}$
  - $el(\neg\varphi_1) := el(\varphi_1)$
  - $el(\varphi_1 \wedge \varphi_2) := el(\varphi_1) \cup el(\varphi_2)$
  - $el(\mathbf{X}\varphi_1) = \{\mathbf{X}\varphi_1\} \cup el(\varphi_1)$
  - $el(\varphi_1 \mathbf{U} \varphi_2) := \{\mathbf{X}(\varphi_1 \mathbf{U} \varphi_2)\} \cup el(\varphi_1) \cup el(\varphi_2)$
- Intuition: $el(\psi)$ is the set of propositions and **X**-formulas occurring ψ', ψ' being the result of applying recursively the tableau expansion rules to ψ
- The set of states S_{T_ψ} of T_ψ is given by $2^{el(\psi)}$
- The labeling function L_{T_ψ} of T_ψ comes straightforwardly (the label is the Boolean component of each state)

Example: $\psi := p\mathbf{U}q$

- $el(p\mathbf{U}q) = el((q \vee (p \wedge \mathbf{X}(p\mathbf{U}q))) = \{p, q, \mathbf{X}(p\mathbf{U}q)\}$

$$\Longrightarrow S_{T_\psi} = \{ \begin{array}{lll} 1: & \{p, q, \mathbf{X}(p\mathbf{U}q)\}, & [p\mathbf{U}q] \\ 2: & \{\neg p, q, \mathbf{X}(p\mathbf{U}q)\}, & [p\mathbf{U}q] \\ 3: & \{p, \neg q, \mathbf{X}(p\mathbf{U}q)\}, & [p\mathbf{U}q] \\ 4: & \{\neg p, q, \neg\mathbf{X}(p\mathbf{U}q)\}, & [p\mathbf{U}q] \\ 5: & \{\neg p, \neg q, \mathbf{X}(p\mathbf{U}q)\}, & [\neg p\mathbf{U}q] \\ 6: & \{p, q, \neg\mathbf{X}(p\mathbf{U}q)\}, & [p\mathbf{U}q] \\ 7: & \{p, \neg q, \neg\mathbf{X}(p\mathbf{U}q)\}, & [\neg p\mathbf{U}q] \\ 8: & \{\neg p, \neg q, \neg\mathbf{X}(p\mathbf{U}q)\} & [\neg p\mathbf{U}q] \end{array} \}$$

Example: $\psi := p\mathbf{U}q$ [cont.]



$sat()$

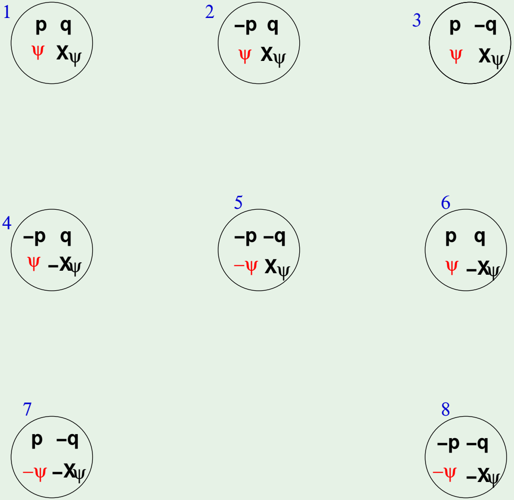
- Set of states in S_{T_ψ} satisfying φ_i: $sat(\varphi_i)$
  - $sat(\varphi_1) := \{s \mid \varphi_1 \in s\}, \varphi_1 \in el(\psi)$
  - $sat(\neg\varphi_1) := S_{T_\psi} / sat(\varphi_1)$
  - $sat(\varphi_1 \wedge \varphi_2) := sat(\varphi_1) \cap sat(\varphi_2)$
  - $sat(\varphi_1 \mathbf{U} \varphi_2) := sat(\varphi_2) \cup (sat(\varphi_1) \cap sat(\mathbf{X}(\varphi_1 \mathbf{U} \varphi_2)))$
- intuition: $sat()$ establishes in which states subformulas are true

Remark

- Semantics of "$\varphi_1 \mathbf{U} \varphi_2$" here induced by tableaux rule: $\varphi_1 \mathbf{U} \varphi_2 \stackrel{\text{def}}{=} \varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1 \mathbf{U} \varphi_2))$

$\Longrightarrow$ weaker than standard semantics (aka "weak until", "$\varphi_1 \mathbf{W} \varphi_2$"):
a path where φ_1 is always true and φ_2 is always false satisfies it

Example: $\psi := p\mathbf{U}q$ [cont.]



Initial States and Transition Relation

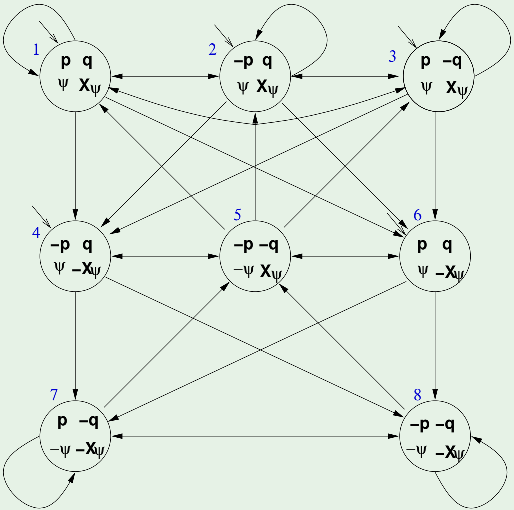
- Set of states in S_{T_ψ} satisfying φ_i: $sat(\varphi_i)$
  - $sat(\varphi_1) := \{s \mid \varphi_1 \in s\}$, $\varphi_1 \in el(\psi)$
  - $sat(\neg\varphi_1) := S_{T_\psi} / sat(\varphi_1)$
  - $sat(\varphi_1 \wedge \varphi_2) := sat(\varphi_1) \cap sat(\varphi_2)$
  - $sat(\varphi_1 \mathbf{U} \varphi_2) := sat(\varphi_2) \cup (sat(\varphi_1) \cap sat(\mathbf{X}(\varphi_1 \mathbf{U} \varphi_2)))$
- Intuition: *sat*() establishes in which states subformulas are true
- The set of initial states I_{T_ψ} is defined as

$$I_{T_\psi} = sat(\psi)$$

- The transition relation R_{T_ψ} is defined as

$$R_{T_\psi}(s, s') = \bigcap_{\mathbf{X}\varphi_i \in el(\psi)} \{(s, s') \mid s \in sat(\mathbf{X}\varphi_i) \Leftrightarrow s' \in sat(\varphi_i)\}$$

Example: $\psi := p\mathbf{U}q$ [cont.]



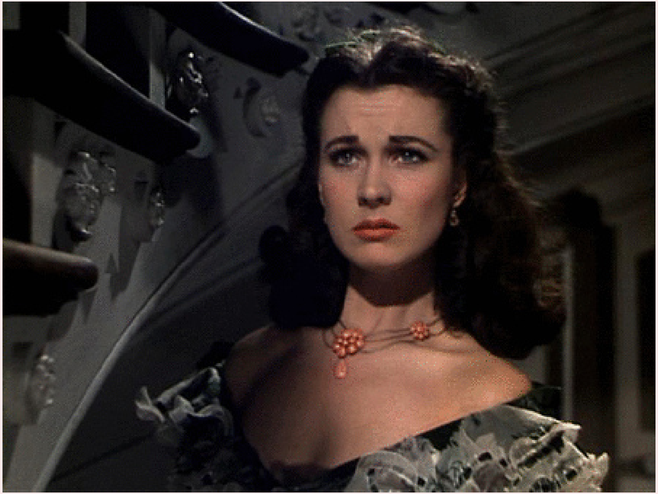
Problems with **U**-subformulas

- R_{T_ψ} does not guarantee that the **U**-subformulas are fulfilled
- Example: state 3 $\{p, \neg q, \mathbf{X}(p\mathbf{U}q)\}$:
  although state 3 belongs to

$$sat(p\mathbf{U}q) := sat(q) \cup (sat(p) \cap sat(\mathbf{X}(p\mathbf{U}q))),$$

  the path which loops forever in state 3 does not satisfy $p\mathbf{U}q$, as q never holds in that path.

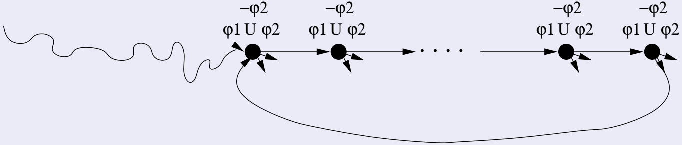
Tableaux Rules: a Quote



"After all... tomorrow is another day."
[Scarlett O'Hara, "Gone with the Wind"]

Fairness conditions for every **U**-subformula

- It must never happen that we get into a state s' from which we can enter a path π' in which $\varphi_1 \mathbf{U} \varphi_2$ holds forever and φ_2 never holds.

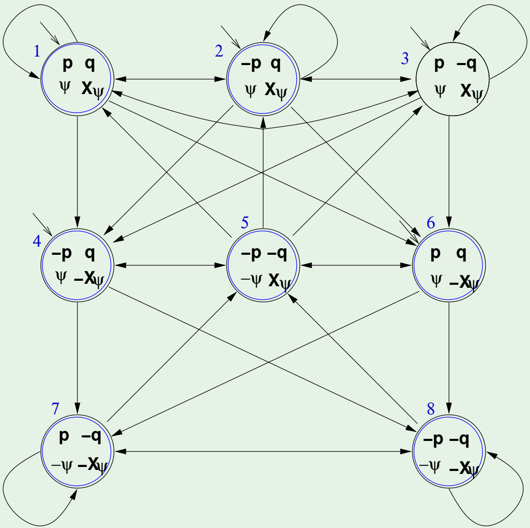


$\Longrightarrow$ For every [positive] **U**-subformula $\varphi_1 \mathbf{U} \varphi_2$ of ψ, we must add a fairness LTL condition $\mathbf{GF}(\neg(\varphi_1 \mathbf{U} \varphi_2) \vee \varphi_2)$
If no [positive] U-subformulas, then add one fairness condition $\mathbf{GF}\top$.

$\Longrightarrow$ We restrict the admissible paths of T_ψ to those which verify the fairness condition:
$T_\psi := \langle S_{T_\psi}, I_{T_\psi}, R_{T_\psi}, L_{T_\psi}, F_{T_\psi} \rangle$

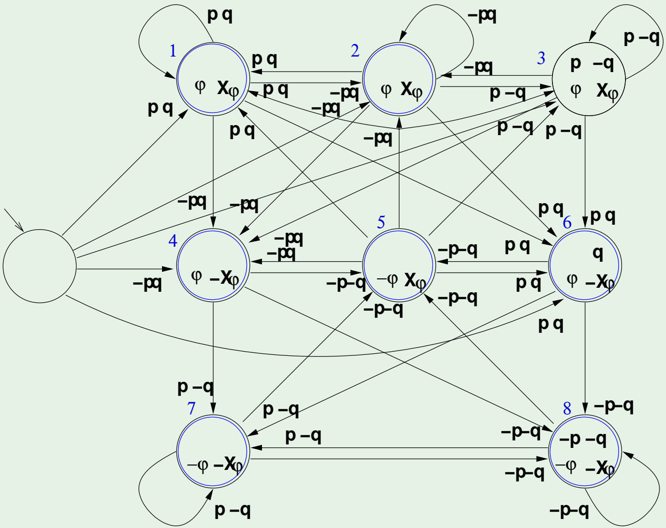
$$F_{T_\psi} := \{sat(\neg(\varphi_1 \mathbf{U} \varphi_2) \vee \varphi_2)) \ s.t. \ (\varphi_1 \mathbf{U} \varphi_2) \ occurs \ [positively] in \ \psi\}$$

Example: $\psi := p\mathbf{U}q$ [cont.]



Note: easily transformed into a generalized Büchi automaton

Example: $\psi := p\mathbf{U}q$ [cont.]



Note: easily transformed into a generalized Büchi automaton

Symbolic Representation of T_ψ

- State variables: one Boolean variable for each formula in $el(\psi)$
  - EX: p, q and x and primed versions p', q' and x'
    [x is a Boolean label for $\mathbf{X}(p\mathbf{U}q)$]
- $sat(\varphi_i)$:
  - $sat(p) := p$, s.t. p Boolean state variable
  - $sat(\neg\varphi_1) := \neg sat(\varphi_1)$
  - $sat(\varphi_1 \wedge \varphi_2) := sat(\varphi_1) \wedge sat(\varphi_2)$
  - $sat(\mathbf{X}\varphi_i) := x_{[\mathbf{X}\varphi_i]}$, s.t. $x_{[\mathbf{X}\varphi_i]}$ Boolean state variable
  - $sat(\varphi_1\mathbf{U}\varphi_2) := sat(\varphi_2) \vee (sat(\varphi_1) \wedge sat(\mathbf{X}(\varphi_1\mathbf{U}\varphi_2)))$

  $\Longrightarrow$ $sat(\varphi_1\mathbf{U}\varphi_2) := sat(\varphi_2) \vee (sat(\varphi_1) \wedge x_{[\mathbf{X}\varphi_1\mathbf{U}\varphi_2]})$
- ...

Symbolic Representation of T_ψ [cont.]

- ...
- Initial states: $I_{T_\psi} = sat(\psi)$
  - EX: $I(p,q,x) = q \vee (p \wedge x)$
- Transition Relation: $R_{T_\psi}(s,s') = \bigcap_{\mathbf{X}\varphi_i \in el(\psi)} \{(s,s') \mid s \in sat(\mathbf{X}\varphi_i) \Leftrightarrow s' \in sat(\varphi_i)\}$
  - $R_{T_\psi} = \bigwedge_{\mathbf{X}\varphi_i \in el(\psi)} (sat(\mathbf{X}\varphi_i) \leftrightarrow sat'(\varphi_i))$
    where $sat'(\varphi_i)$ is $sat(\varphi_i)$ on primed variables
  - EX: $R_{T_\psi}(p,q,x,p',q',x') = x \leftrightarrow (q' \vee (p' \wedge x'))$
- Fairness Conditions: $F_{T_\psi} := \{sat(\neg(\varphi_1 \mathbf{U} \varphi_2) \vee \varphi_2))\ s.t.\ (\varphi_1 \mathbf{U} \varphi_2)\ occurs\ [positively] in\ \psi\}$
  - EX: $F_{T_\psi}(p,q,x) = \neg(q \vee (p \wedge x)) \vee q = ... = \neg p \vee \neg x \vee q$

Symbolic Representation of T_ψ: Examples

- $I_{T_\psi}(p, q, x) = q \vee (p \wedge x)$

  $1: \quad \{p, q, x\} \models I_{T_\psi}$

  $3: \quad \{p, \neg q, x\} \models I_{T_\psi}$

  $\not 5: \quad \{\neg p, \neg q, x\} \not\models I_{T_\psi}$

- $R_{T_\psi}(p, q, x, p', q', x') = \quad x \leftrightarrow (q' \vee (p' \wedge x'))$

  $1 \Rightarrow 1: \quad \{p, q, x, p', q', x'\} \models R_{T_\psi}$

  $6 \Rightarrow 7: \quad \{p, q, \neg x, p', \neg q', \neg x'\} \models R_{T_\psi}$

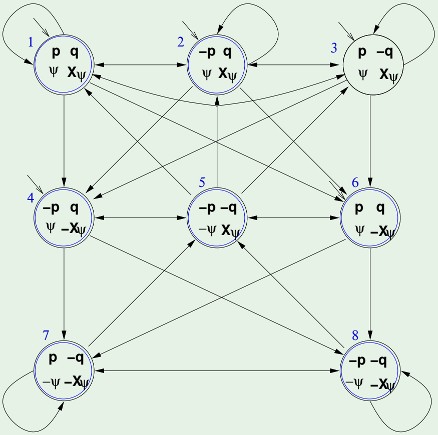
  $6 \not\Rightarrow 1: \quad \{p, q, \neg x, p', q', x'\} \not\models R_{T_\psi}$

- $F_{T_\psi}(p, q, x) = \neg p \vee \neg x \vee q$

  $1: \quad \{p, q, x\} \models F_{T_\psi}$

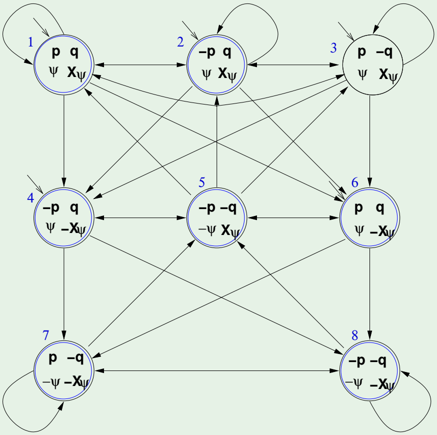
  $5: \quad \{\neg p, \neg q, x\} \models F_{T_\psi}$

  $\not 3: \quad \{p, \neg q, x\} \not\models F_{T_\psi}$



Symbolic Representation of T_ψ: Examples

- $I_{T_\psi}(p, q, x) = q \vee (p \wedge x)$
  - $1: \quad \{p, q, x\} \models I_{T_\psi}$
  - $3: \quad \{p, \neg q, x\} \models I_{T_\psi}$
  - $\not 5: \quad \{\neg p, \neg q, x\} \not\models I_{T_\psi}$
- $R_{T_\psi}(p, q, x, p', q', x') = \quad x \leftrightarrow (q' \vee (p' \wedge x'))$
  - $1 \Rightarrow 1: \quad \{p, q, x, p', q', x'\} \models R_{T_\psi}$
  - $6 \Rightarrow 7: \quad \{p, q, \neg x, p', \neg q', \neg x'\} \models R_{T_\psi}$
  - $6 \not\Rightarrow 1: \quad \{p, q, \neg x, p', q', x'\} \not\models R_{T_\psi}$
- $F_{T_\psi}(p, q, x) = \neg p \vee \neg x \vee q$
  - $1: \quad \{p, q, x\} \models F_{T_\psi}$
  - $5: \quad \{\neg p, \neg q, x\} \models F_{T_\psi}$
  - $\not 3: \quad \{p, \neg q, x\} \not\models F_{T_\psi}$



Symbolic Representation of T_ψ: Examples

- $I_{T_\psi}(p,q,x) = q \vee (p \wedge x)$

  $1: \quad \{p,q,x\} \models I_{T_\psi}$

  $3: \quad \{p,\neg q,x\} \models I_{T_\psi}$

  $\not 5: \quad \{\neg p,\neg q,x\} \not\models I_{T_\psi}$

- $R_{T_\psi}(p,q,x,p',q',x') = \quad x \leftrightarrow (q' \vee (p' \wedge x'))$

  $1 \Rightarrow 1: \quad \{p,q,x,p',q',x'\} \models R_{T_\psi}$

  $6 \Rightarrow 7: \quad \{p,q,\neg x,p',\neg q',\neg x'\} \models R_{T_\psi}$

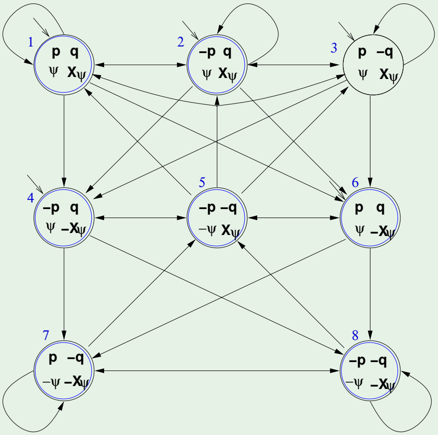
  $6 \not\Rightarrow 1: \quad \{p,q,\neg x,p',q',x'\} \not\models R_{T_\psi}$

- $F_{T_\psi}(p,q,x) = \neg p \vee \neg x \vee q$

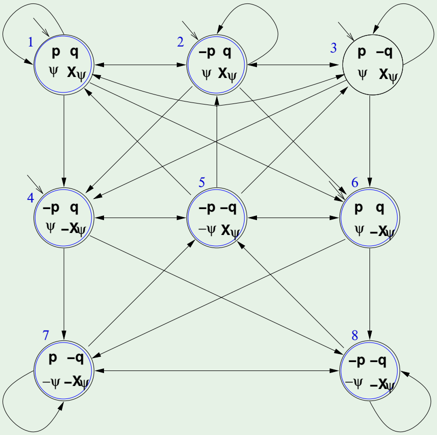
  $1: \quad \{p,q,x\} \models F_{T_\psi}$

  $5: \quad \{\neg p,\neg q,x\} \models F_{T_\psi}$

  $\not 3: \quad \{p,\neg q,x\} \not\models F_{T_\psi}$



- $I_{T_\psi}(p, q, x) = q \lor (p \land x)$
  $1: \quad \{p, q, x\} \models I_{T_\psi}$
  $3: \quad \{p, \neg q, x\} \models I_{T_\psi}$
  $\not 5: \quad \{\neg p, \neg q, x\} \not\models I_{T_\psi}$
- $R_{T_\psi}(p, q, x, p', q', x') = \qquad x \leftrightarrow (q' \lor (p' \land x'))$
  $1 \Rightarrow 1: \quad \{p, q, x, p', q', x'\} \models R_{T_\psi}$
  $6 \Rightarrow 7: \quad \{p, q, \neg x, p', \neg q', \neg x'\} \models R_{T_\psi}$
  $6 \not\Rightarrow 1: \quad \{p, q, \neg x, p', q', x'\} \not\models R_{T_\psi}$
- $F_{T_\psi}(p, q, x) = \neg p \lor \neg x \lor q$
  $1: \quad \{p, q, x\} \models F_{T_\psi}$
  $5: \quad \{\neg p, \neg q, x\} \models F_{T_\psi}$
  $\not 3: \quad \{p, \neg q, x\} \not\models F_{T_\psi}$



Outline

Computing the product $P := T_\psi \times M$

- Given $M := \langle S_M, I_M, R_M, L_M \rangle$ and $T_\psi := \langle S_{T_\psi}, I_{T_\psi}, R_{T_\psi}, L_{T_\psi}, F_{T_\psi} \rangle$, we compute the product $P := T_\psi \times M = \langle S, I, R, L, F \rangle$ as follows:
  - $S := \{(s, s') \mid s \in S_{T_\psi},\ s' \in S_M \textit{ and } L_M(s')|_\psi = L_{T_\psi}(s)\}$
  - $I := \{(s, s') \mid s \in I_{T_\psi},\ s' \in I_M \textit{ and } L_M(s')|_\psi = L_{T_\psi}(s)\}$
  - Given $(s, s'), (t, t') \in S$, $((s, s'), (t, t')) \in R$ iff $(s, t) \in R_{T_\psi}$ and $(s', t') \in R_M$
  - $L((s, s')) = L_{T_\psi}(s) \cup L_M(s')$
- Extension of $sat()$ and F_{T_ψ} to P:
  $(s, s') \in sat(\psi) \Longleftrightarrow s \in sat(\psi)$
  $F := \{sat(\neg(\varphi_1 \mathbf{U} \varphi_2) \vee \varphi_2) \textit{ s.t. } (\varphi_1 \mathbf{U} \varphi_2) \textit{ occurs [positively] in } \psi\}$

Computing the product $P := T_\psi \times M$ symbolically

Let V, W be the array of Boolean state variables of T_ψ and M respectively:

- Initial states: $I(V \cup W) = I_{T_\psi}(V) \wedge I_M(W)$
- Transition Relation: $R(V \cup W, V' \cup W') = R_{T_\psi}(V, V') \wedge R_M(W, W')$
- Fairness conditions: $\{F_1(V \cup W), ..., F_k(V \cup W)\} = \{F_{T_\psi 1}(V), ..., F_{T_\psi k}(V)\}$

Outline

Main theorem [Clarke, Grumberg & Hamaguchi; 94]

Theorem

THEOREM: $M.s' \models \mathbf{E}\psi$ iff there is a state s in T_ψ s.t. $(s,s') \in sat(\psi)$ and $T_\psi \times M, (s,s') \models \mathbf{EG}true$ under the fairness conditions:

$$\{sat(\neg(\varphi_1 \mathbf{U} \varphi_2) \vee \varphi_2))\ s.t.\ (\varphi_1 \mathbf{U} \varphi_2)\ occurs\ in\ \psi\}.$$

$\Longrightarrow$ $M \models \mathbf{E}\psi$ iff $T_\psi \times M \models \mathbf{E}_f\mathbf{G}true$

$\Longrightarrow$ $M \models \neg\psi$ iff $T_\psi \times M \not\models \mathbf{E}_f\mathbf{G}true$

- LTL M.C. reduced to Fair CTL M.C.!!!
- Symbolic OBDD-based techniques apply.

Note

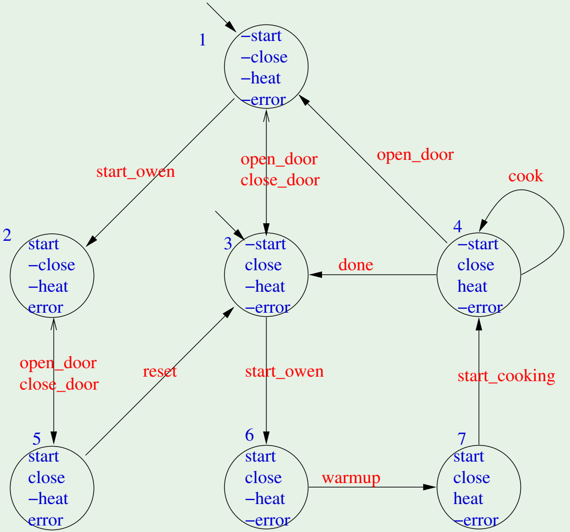
The transition relation R of $T_\psi \times M$ may not be total.
$\Longrightarrow$ `Check_FairEG` does not consider states without successors, restricting R to the remaining states.

Outline

A microwave oven

- 4 state variables: start, close, heat, error
- Actions (implicit): start_oven,open_door, close_door, reset, warmup, start_cooking, cook, done
- Error situation: if oven is started while the door is open
- Represented as a Kripke structure (and hence as a OBDD's)

A microwave oven [cont.]



A microwave oven: symbolic representation

- Initial states: $I_M(s, c, h, e) = \neg s \wedge \neg h \wedge \neg e$
- Transition relation: $R_M(s, c, h, e, s', c', h', e') =$ [a simplification of]

$$
\begin{array}{lll}
(& \neg s \wedge \neg c \wedge \neg h \wedge \neg e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (close\_door,\ no\ error) \\
(& s \wedge \neg c \wedge \neg h \wedge e \wedge s' \wedge c' \wedge \neg h' \wedge e') \vee & (close\_door,\ error) \\
(& \neg s \wedge c \wedge \neg e \wedge \neg s' \wedge \neg c' \wedge \neg h' \wedge \neg e') \vee & (open\_door,\ no\ error) \\
(& s \wedge c \wedge \neg h \wedge e \wedge s' \wedge \neg c' \wedge \neg h' \wedge e') \vee & (open\_door,\ error) \\
(& \neg s \wedge c \wedge \neg h \wedge \neg e \wedge s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (start\_oven,\ no\ error) \\
(& \neg s \wedge \neg c \wedge \neg h \wedge \neg e \wedge s' \wedge \neg c' \wedge \neg h' \wedge e') \vee & (start\_oven,\ error) \\
(& s \wedge c \wedge \neg h \wedge e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (reset) \\
(& s \wedge c \wedge \neg h \wedge \neg e \wedge s' \wedge c' \wedge h' \wedge \neg e') \vee & (warmup) \\
(& s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge h' \wedge \neg e') \vee & (start\_cooking) \\
(& \neg s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge h' \wedge \neg e') \vee & (cook) \\
(& \neg s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') & (done)
\end{array}
$$

Note: the third row represents two transitions: $3 \to 1$ and $4 \to 1$.

LTL Specification

- "necessarily, the oven's door eventually closes and, till there, the oven does not heat":

$$M \models \neg heat \ \mathbf{U} \ close,$$

i.e.,

$$M \models \neg \mathbf{E} \neg (\neg heat \ \mathbf{U} \ close)$$

Tableau construction for $\psi = \neg(\neg heat\ \mathbf{U}\ close)$

- $\varphi := \neg\psi = (\neg heat\ \mathbf{U}\ close)$
- Tableaux expansion: $\psi = \neg(\neg heat\ \mathbf{U}\ close) = \neg(close \vee (\neg heat \wedge \mathbf{X}(\neg heat\ \mathbf{U}\ close)))$
- $el(\psi) = el(\varphi) = \{heat, close, \mathbf{X}\varphi\}$ $(\{h, c, \mathbf{X}\varphi\})$
- States:

$$1 := \{\neg h, c, \mathbf{X}\varphi\},\ 2 := \{h, c, \mathbf{X}\varphi\},\ 3 := \{\neg h, \neg c, \mathbf{X}\varphi\},$$
$$4 := \{h, c, \neg\mathbf{X}\varphi\},\ 5 := \{h, \neg c, \mathbf{X}\varphi\},\ 6 := \{\neg h, c, \neg\mathbf{X}\varphi\},$$
$$7 := \{\neg h, \neg c, \neg\mathbf{X}\varphi\},\ 8 := \{h, \neg c, \neg\mathbf{X}\varphi\}$$

Tableau construction for $\psi = \neg(\neg \textit{heat}\ \mathbf{U}\ \textit{close})$ [cont.]

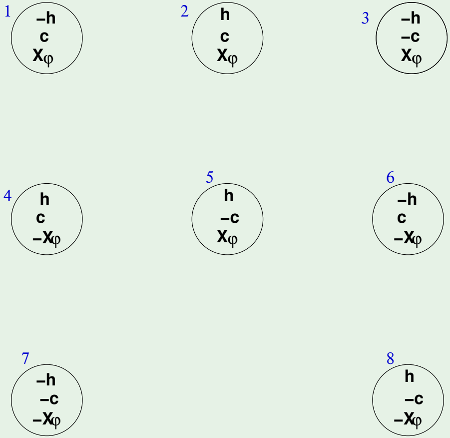


Tableau construction for $\psi = \neg(\neg \textit{heat}\ \mathbf{U}\ \textit{close})$

- ...
- States:

$$1 := \{\neg h, c, \mathbf{X}\varphi\},\ 2 := \{h, c, \mathbf{X}\varphi\},\ 3 := \{\neg h, \neg c, \mathbf{X}\varphi\},$$
$$4 := \{h, c, \neg\mathbf{X}\varphi\},\ 5 := \{h, \neg c, \mathbf{X}\varphi\},\ 6 := \{\neg h, c, \neg\mathbf{X}\varphi\},$$
$$7 := \{\neg h, \neg c, \neg\mathbf{X}\varphi\},\ 8 := \{h, \neg c, \neg\mathbf{X}\varphi\}$$

- *sat*():

$$\begin{aligned}
&sat(h) = \{2,4,5,8\} \implies sat(\neg h) = \{1,3,6,7\},\\
&sat(c) = \{1,2,4,6\} \implies sat(\neg c) = \{3,5,7,8\},\\
&sat(\mathbf{X}\varphi) = \{1,2,3,5\} \implies sat(\neg\mathbf{X}\varphi) = \{4,6,7,8\},\\
&sat(\varphi) = sat(c) \cup (sat(\neg h) \cap sat(\mathbf{X}(\neg h\ \mathbf{U}\ c))) = \{1,2,3,4,6\}\\
&\implies sat(\psi) = sat(\neg\varphi) = \{5,7,8\}
\end{aligned}$$

Tableau construction for $\psi = \neg(\neg$*heat* **U** *close*$)$ [cont.]

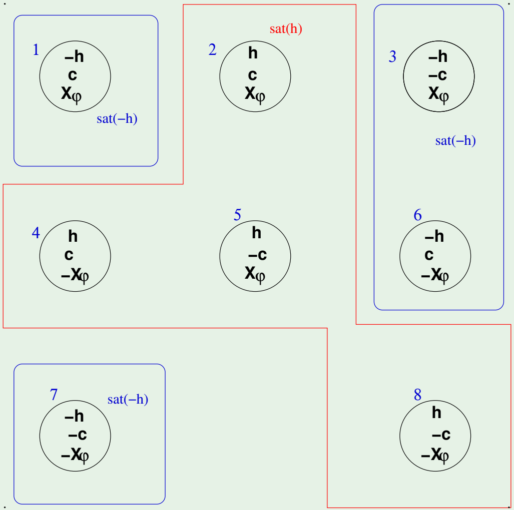


Tableau construction for $\psi = \neg(\neg \textit{heat}\ \mathbf{U}\ \textit{close})$ [cont.]

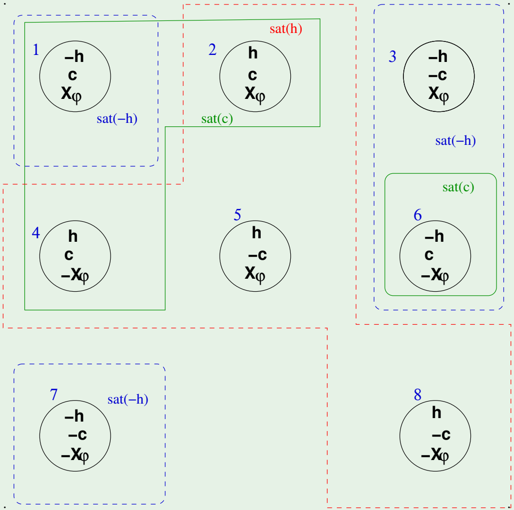


Tableau construction for $\psi = \neg(\neg \textit{heat}\ \mathbf{U}\ \textit{close})$ [cont.]

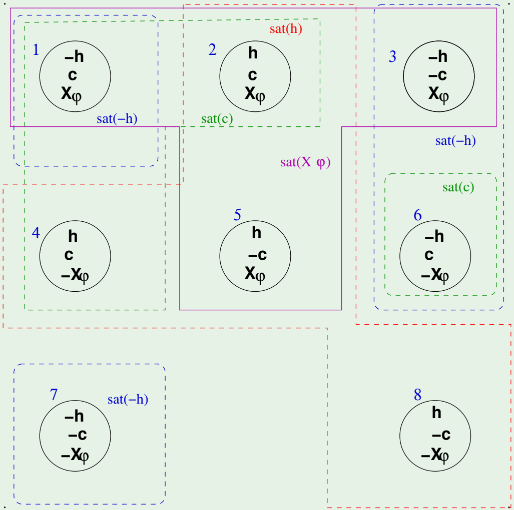


Tableau construction for $\psi = \neg(\neg heat \ \mathbf{U} \ close)$ [cont.]

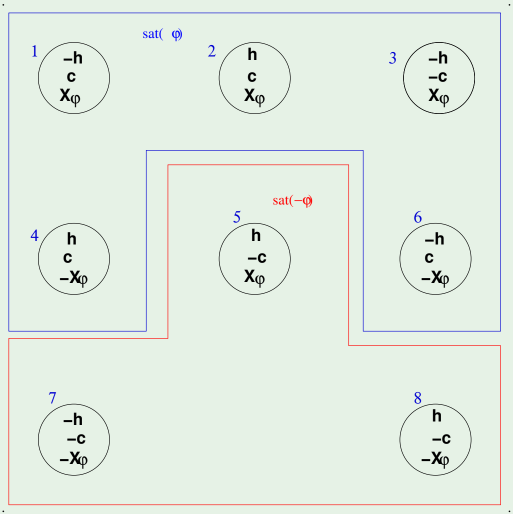


Tableau construction for $\psi = \neg(\neg \mathit{heat}\ \mathbf{U}\ \mathit{close})$ [cont.]

- ...
- *sat*():

$$\begin{array}{l} sat(h) = \{2,4,5,8\} \implies sat(\neg h) = \{1,3,6,7\}, \\ sat(c) = \{1,2,4,6\} \implies sat(\neg c) = \{3,5,7,8\}, \\ sat(\mathbf{X}\varphi) = \{1,2,3,5\} \implies sat(\neg\mathbf{X}\varphi) = \{4,6,7,8\}, \\ sat(\varphi) = sat(c) \cup (sat(\neg h) \cap sat(\mathbf{X}(\neg h\ \mathbf{U}\ c))) = \{1,2,3,4,6\} \end{array}$$

- Initial states *I*: $sat(\psi) = sat(\neg\varphi) = \{5,7,8\}$
- Transition Relation *R*:
  - add an edge from every state in $sat(\mathbf{X}\varphi)$ to every state in $sat(\varphi)$
  - add an edge from every state in $sat(\neg\mathbf{X}\varphi)$ to every state in $sat(\neg\varphi)$

Tableau construction for $\psi = \neg(\neg$*heat* **U** *close*$)$ [cont.]

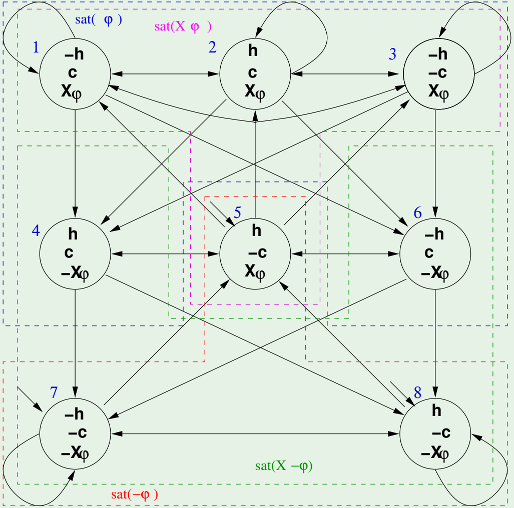
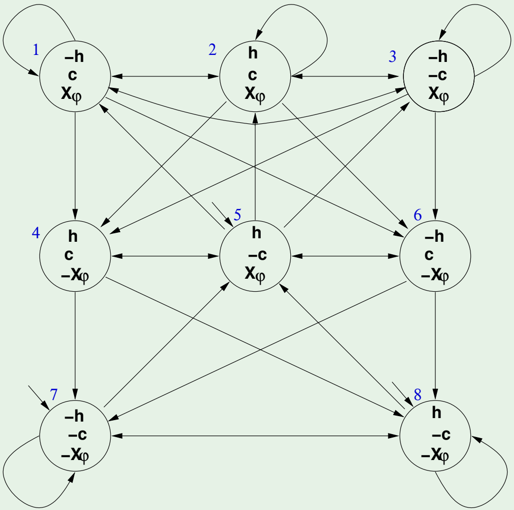


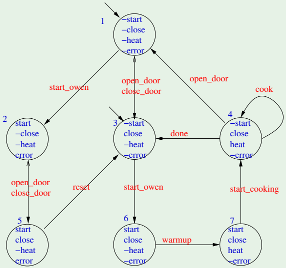
Tableau construction for $\psi = \neg(\neg \textit{heat} \ \mathbf{U} \ \textit{close})$ [cont.]

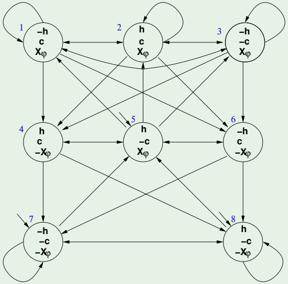


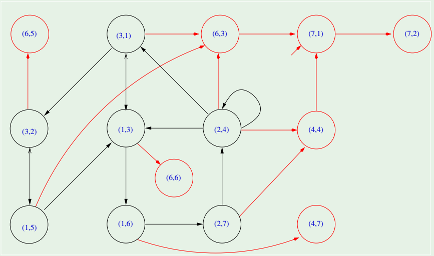
Symbolic representation of T_ψ, s.t. $\psi := \neg(\neg h \mathbf{U} c)$

- State variables: h, c and x and primed versions h', c' and x'
  [x is a Boolean label for $\mathbf{X}(\neg h \mathbf{U} c)$]
- Initial states: $I_{T_\psi} = sat(\psi)$
  $\Longrightarrow I(h, c, x) = \neg(c \vee (\neg h \wedge x))$
- Transition Relation: $R_{T_\psi} = \bigwedge_{\mathbf{X}\varphi_i \in el(\psi)} (sat(\mathbf{X}\varphi_i) \leftrightarrow sat'(\varphi_i))$
  $\Longrightarrow R_{T_\psi}(h, c, x, h', c', x') = x \leftrightarrow (c' \vee (\neg h' \wedge x'))$
- Fairness Property: (due to negative polarity of $(\neg h\, \mathbf{U} c)$ in ψ):
  $F_{T_\psi}(h, c, x) = \top$

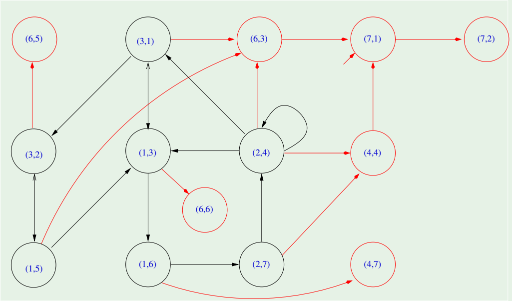
Product $P = T_\psi \times M$







Product $P = T_\psi \times M$ [cont.]



- $P = T_\psi \times M$ (reachable states only)
- compute [**EG***true*] (e.g. by Emerson-Lei):

  $\Longrightarrow$ states $(4,4)$, $(4,7)$, $(6,3)$, $(6,5)$, $(6,6)$, $(7,1)$, $(7,2)$ are not part of a (fair) infinite path

  $\Longrightarrow$ no initial states in [**EG***true*] ((7.1) has been removed).

  $\Longrightarrow$ $T_\psi \times M \not\models$ **EG***true*

  $\Longrightarrow$ Property verified!
- N.B.: fairness condition $\top$ irrelevent here

Product $P = T_\psi \times M$: symbolic representation

- Initial states: $I(s,c,h,e,x) = (\neg s \wedge \neg h \wedge \neg e) \wedge \neg(c \vee (\neg h \wedge x)) = \neg s \wedge \neg h \wedge \neg e \wedge \neg c \wedge \neg x$
- Transition relation: $R(s,c,h,e,x,s',c',h',e',x') =$ (an OBDD for)

$$
\begin{array}{ll}
(x \leftrightarrow (c' \vee (\neg h' \wedge x'))) \wedge (& \\
(\ \neg s \wedge \neg c \wedge \neg h \wedge \neg e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (close\_door,\ no\ error) \\
(\ s \wedge \neg c \wedge \neg h \wedge e \wedge s' \wedge c' \wedge \neg h' \wedge e') \vee & (close\_door,\ error) \\
(\ \neg s \wedge c \wedge \neg e \wedge \neg s' \wedge \neg c' \wedge \neg h' \wedge \neg e') \vee & (open\_door,\ no\ error) \\
(\ s \wedge c \wedge \neg h \wedge e \wedge s' \wedge \neg c' \wedge \neg h' \wedge e') \vee & (open\_door,\ error) \\
(\ \neg s \wedge c \wedge \neg h \wedge \neg e \wedge s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (start\_oven,\ no\ error) \\
(\ \neg s \wedge \neg c \wedge \neg h \wedge \neg e \wedge s' \wedge \neg c' \wedge \neg h' \wedge e') \vee & (start\_oven,\ error) \\
(\ s \wedge c \wedge \neg h \wedge e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') \vee & (reset) \\
(\ s \wedge c \wedge \neg h \wedge \neg e \wedge s' \wedge c' \wedge h' \wedge \neg e') \vee & (warmup) \\
(\ s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge h' \wedge \neg e') \vee & (start\_cooking) \\
(\ \neg s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge h' \wedge \neg e') \vee & (cook) \\
(\ \neg s \wedge c \wedge h \wedge \neg e \wedge \neg s' \wedge c' \wedge \neg h' \wedge \neg e') & (done) \\
) &
\end{array}
$$

[**EG***true*]: symbolic representation

- Emerson-Lei returns (an OBDD equivalent to):

$$
\begin{array}{ll}
\mathbf{EG}\,true = & \\
(\ \neg s \wedge \neg c \wedge \neg h \wedge \neg e \wedge\ x) \vee & (3,1) \\
(\ \ s \wedge \neg c \wedge \neg h \wedge\ e \wedge\ x) \vee & (3,2) \\
(\ \neg s \wedge\ c \wedge \neg h \wedge \neg e \wedge\ x) \vee & (1,3) \\
(\ \neg s \wedge\ c \wedge\ h \wedge \neg e \wedge\ x) \vee & (2,4) \\
(\ \ s \wedge\ c \wedge \neg h \wedge\ e \wedge\ x) \vee & (1,5) \\
(\ \ s \wedge\ c \wedge \neg h \wedge \neg e \wedge\ x) \vee & (1,5) \\
(\ \ s \wedge\ c \wedge\ h \wedge \neg e \wedge\ x) \vee & (2,7) \\
\quad \ldots & \text{(other unreachables states)}
\end{array}
$$

- Initial states: $I(s,c,h,e,x) = \neg s \wedge \neg h \wedge \neg e \wedge \neg c \wedge \neg x$

$\Longrightarrow$ $I(s,c,h,e,x) \not\models \mathbf{EG}\,true$

$\Longrightarrow$ $I \not\subseteq [\mathbf{EG}\,true]$

$\Longrightarrow$ $T_\psi \times M \not\models \mathbf{EG}\,true$

$\Longrightarrow$ Property verified!

The property verified is...

Outline

Ex: Symbolic CTL Model Checking

Given the following finite state machine expressed in NuSMV input language:

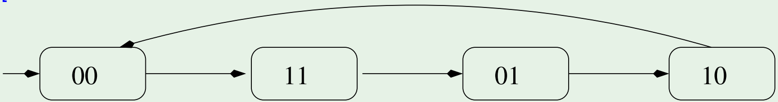
```
MODULE main
VAR v1 : boolean; v2 : boolean;
INIT (!v1 & !v2)
TRANS (next(v1) <-> !v1) & (next(v2) <-> (v1<->v2))
```

and consider the property $P \stackrel{\text{def}}{=} (v_1 \wedge v_2)$. Write:

- the Boolean formulas $I(v_1, v_2)$ and $T(v_1, v_2, v_1', v_2')$ representing respectively the initial states and the transition relation of M.

  [Solution: $I(v_1, v_2)$ is $(\neg v_1 \wedge \neg v_2)$, $T(v_1, v_2, v_1', v_2')$ is $(v_1' \leftrightarrow \neg v_1) \wedge (v_2' \leftrightarrow (v_1 \leftrightarrow v_2))$]
- the graph representing the FSM. (Assume the notation "$v_1 v_2$" for labeling the states: e.g. "10" means "$v_1 = 1, v_2 = 0$".)

  [Solution:



]

Ex: Symbolic CTL Model Checking (cont.)

- the Boolean formula representing symbolically **EX**P. [The formula must be computed symbolically, not simply inferred from the graph of the previous question!]

  [Solution:

$$
\begin{array}{rcl}
\mathbf{EX}(P) & = & \exists v_1', v_2'.(T(v_1, v_2, v_1', v_2') \wedge P(v_1', v_2')) \\
& = & \exists v_1', v_2'.((v_1' \leftrightarrow \neg v_1) \wedge (v_2' \leftrightarrow (v_1 \leftrightarrow v_2)) \wedge \underbrace{(v_1' \wedge v_2')}_{\Longrightarrow v_1'=\top, v_2'=\top}) \\
& = & \overbrace{(\neg v_1 \wedge \neg v_2)}^{v_1'=\top, v_2'=\top} \vee \bot \vee \bot \vee \bot \\
& = & (\neg v_1 \wedge \neg v_2)
\end{array}
$$

  .]

Ex: Symbolic CTL Model Checking

Given the following finite state machine expressed in NuSMV input language:

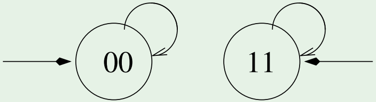
```
VAR     v1 : boolean;  v2 : boolean;
INIT    init(v1) <-> init(v2)
TRANS   (v1 <-> next(v2)) &   (v2 <-> next(v1));
```

write:

- the Boolean formulas $I(v_1, v_2)$ and $T(v_1, v_2, v_1', v_2')$ representing the initial states and the transition relation of M respectively.

  [Solution: $I(v_1, v_2)$ is $(v_1 \leftrightarrow v_2)$, $T(v_1, v_2, v_1', v_2')$ is $(v_1 \leftrightarrow v_2') \wedge (v_2 \leftrightarrow v_1')$]

- the graph representing the FSM. (Assume the notation "$v_1 v_2$" for labeling the states. E.g., "10" means "$v_1 = 1, v_2 = 0$".)

  [Solution: ]

Ex: Symbolic CTL Model Checking (cont.)

- the Boolean formula $R^1(v'_1, v'_2)$ representing the set of states which can be reached after exactly 1 step. NOTE: this must be computed symbolically, not simply deduced from the graph of question b).

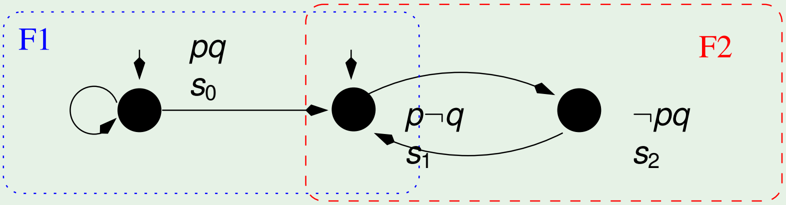
  [Solution:

$$\begin{array}{rcl} R^1(v'_1, v'_2) & = & \exists v_1, v_2.(I(v_1, v_2) \wedge T(v_1, v_2, v'_1, v'_2)) \\ & = & \exists v_1, v_2.((v_1 \leftrightarrow v_2) \wedge (v_1 \leftrightarrow v'_2) \wedge (v_2 \leftrightarrow v'_1)) \\ & = & ((v_1 \leftrightarrow v_2) \wedge (v_1 \leftrightarrow v'_2) \wedge (v_2 \leftrightarrow v'_1))[v_1 = \bot, v_2 = \bot] \vee \\ & & ((v_1 \leftrightarrow v_2) \wedge (v_1 \leftrightarrow v'_2) \wedge (v_2 \leftrightarrow v'_1))[v_1 = \bot, v_2 = \top] \vee \\ & & ((v_1 \leftrightarrow v_2) \wedge (v_1 \leftrightarrow v'_2) \wedge (v_2 \leftrightarrow v'_1))[v_1 = \top, v_2 = \bot] \vee \\ & & ((v_1 \leftrightarrow v_2) \wedge (v_1 \leftrightarrow v'_2) \wedge (v_2 \leftrightarrow v'_1))[v_1 = \top, v_2 = \top] \\ & = & (\neg v'_1 \wedge \neg v'_2) \vee \bot \vee \bot \vee (v'_1 \wedge v'_2) \\ & = & (\neg v'_1 \wedge \neg v'_2) \vee (v'_1 \wedge v'_2) \\ & = & (v'_1 \leftrightarrow v'_2) \end{array}$$

  .]

Ex: Fair CTL Model Checking

Consider the following *fair* Kripke Model M:



For each of the following facts, say if it is true or false in CTL.

(*a*) $M \models \mathbf{AF}\neg p$
[Solution: true]

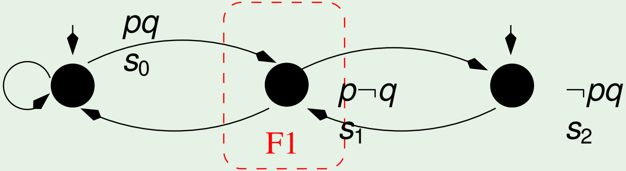
(*b*) $M \models \mathbf{A}(p\mathbf{U}\neg q)$
[Solution: true]

(*c*) $M \models \mathbf{AX}\neg q$
[Solution: false]

(*d*) $M \models \mathbf{AGAF}\neg p$
[Solution: true]

Ex: Fair CTL Model Checking

Consider the following *fair* Kripke Model M:



For each of the following facts, say if it is true or false in CTL.

(a) $M \models \mathbf{EF}(p \wedge q)$
[Solution: true]

(b) $M \models \mathbf{AGAF}p$
[Solution: true]

(c) $M \models \mathbf{AF}\neg q$
[Solution: true]

(d) $M \models \mathbf{AG}(\neg p \vee \neg q)$
[Solution: false]

Ex: Symbolic LTL Model Checking

Given the following LTL formula: $\varphi \stackrel{\text{def}}{=} \neg((\mathbf{GF}p \wedge \mathbf{GF}q) \rightarrow \mathbf{GF}r)$

(a) Compute the Negative Normal Form of φ ($NNF(\varphi)$).

[Solution:
$$\begin{array}{lll} \varphi & \Longleftrightarrow & \neg((\mathbf{GF}p \wedge \mathbf{GF}q) \rightarrow \mathbf{GF}r) \\ & \Longleftrightarrow & \neg(\neg(\mathbf{GF}p \wedge \mathbf{GF}q) \vee \mathbf{GF}r) \\ & \Longleftrightarrow & (\mathbf{GF}p \wedge \mathbf{GF}q \wedge \neg\mathbf{GF}r) \\ & \Longleftrightarrow & (\mathbf{GF}p \wedge \mathbf{GF}q \wedge \mathbf{FG}\neg r) \Longleftrightarrow NNF(\varphi) \end{array}$$
]

(b) Compute the set of elementary subformulas of φ.

[Solution: First write the formula in terms of **X** and **U**'s (write "$\mathbf{F}\psi$" for "$\top\mathbf{U}\psi$"):

$$\begin{array}{lll} \varphi & \Longleftrightarrow & \neg((\mathbf{GF}p \wedge \mathbf{GF}q) \rightarrow \mathbf{GF}r) \\ & \Longleftrightarrow & \neg((\neg\mathbf{F}\neg\mathbf{F}p \wedge \neg\mathbf{F}\neg\mathbf{F}q) \rightarrow \neg\mathbf{F}\neg\mathbf{F}r) \end{array}$$

$el(\mathbf{F}\neg\mathbf{F}p) = \{\mathbf{XF}\neg\mathbf{F}p\} \cup el(\neg\mathbf{F}p) = \{\mathbf{XF}\neg\mathbf{F}p\} \cup \{\mathbf{XF}p\} \cup el(p) = \{\mathbf{XF}\neg\mathbf{F}p, \mathbf{XF}p, p\}$.

Hence:
$$\begin{array}{lll} el(\varphi) & = & el(\neg((\neg\mathbf{F}\neg\mathbf{F}p \wedge \neg\mathbf{F}\neg\mathbf{F}q) \rightarrow \neg\mathbf{F}\neg\mathbf{F}r)) \\ & = & el(\mathbf{F}\neg\mathbf{F}p) \cup el(\mathbf{F}\neg\mathbf{F}q) \cup el(\mathbf{F}\neg\mathbf{F}r) \\ & = & \{\mathbf{XF}\neg\mathbf{F}p, \mathbf{XF}p, p, \mathbf{XF}\neg\mathbf{F}q, \mathbf{XF}q, q, \mathbf{XF}\neg\mathbf{F}r, \mathbf{XF}r, r\} \end{array}$$
]

(c) What is the (maximum) number of states of a fair Kripke Model representing φ?

[Solution: By definition it is $2^{|el(\varphi)|} = 2^9 = 512$.]

Ex: Symbolic LTL Model Checking

Given the following LTL formula $\psi \stackrel{\text{def}}{=} \neg\mathbf{F}\neg p$, compute and draw the tableau $\mathcal{T}_\psi$ of ψ. [Solution:

(i) The set of elementary subformulas of ψ is $el(\psi) \stackrel{\text{def}}{=} \{p, \mathbf{XF}\neg p\}$. Hence, the set of states is

$$\{s_1 : (p, \neg\mathbf{XF}\neg p),\ s_2 : (p, \mathbf{XF}\neg p),\ s_3 : (\neg p, \neg\mathbf{XF}\neg p),\ s_4 : (\neg p, \mathbf{XF}\neg p)\}$$

(ii) The set of initial states of $\mathcal{T}_\psi$ is $sat(\psi) \stackrel{\text{def}}{=} S \setminus (sat(\neg p) \cup sat(\mathbf{XF}\neg p)) = \{s_1\}$.

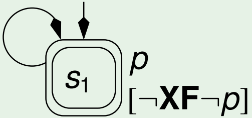
(iii) Since s_1 is the only state in $sat(\neg\mathbf{F}\neg p)$, then s_1 is the only successor of itself, so that the only relevant transition is a self-loop over s_1.
(One can also —un-necessarily— draw all transitions from states where $\neg\mathbf{XF}\neg p$ holds into $\{s_1\}$ and from from states where $\mathbf{XF}\neg p$ holds into $\{s_2, s_3, s_4\}$.)

(iv) There is one **U**-subformula, $\mathbf{F}\neg p$, so that there is one fairness condition defined as $sat(\neg\mathbf{F}\neg p \vee \neg p)$. Since $\mathbf{F}\neg p$ is false in s_1, then s_1 is part of the fairness condition. [Alternatively: there is no positive **U**-subformula, so that we must add a $\mathbf{AGAF}\top$ fairness condition, which is equivalent to say that all states belong to the fairness condition.]

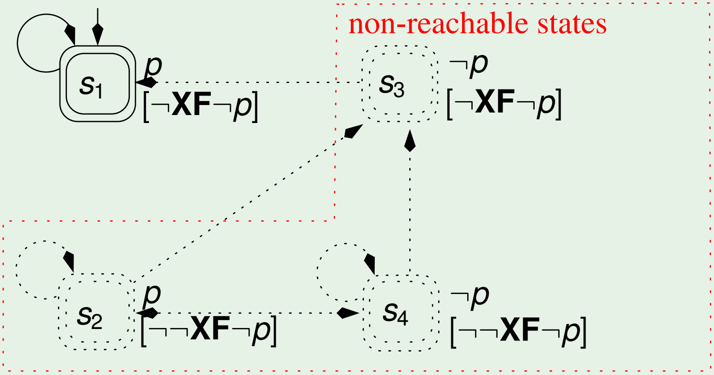
]

Ex: Symbolic LTL Model Checking (cont.)

[Solution:



or, alternatively without simplifications:



]

Ex: Symbolic LTL Model Checking

Given the following LTL formula $\psi \stackrel{\text{def}}{=} \mathbf{G}p$, compute and draw the tableau $\mathcal{T}_\psi$ of ψ. [Without converting anything into **X**, **U**].

[Solution:

(i) The set of elementary subformulas of ψ is $el(\psi) \stackrel{\text{def}}{=} \{p, \mathbf{XG}p\}$. Hence, the set of states is

$$\{s_1 : (p, \mathbf{XG}p),\ s_2 : (p, \neg\mathbf{XG}p),\ s_3 : (\neg p, \mathbf{XG}p),\ s_4 : (\neg p, \neg\mathbf{XG}p)\}$$

(ii) The set of initial states of $\mathcal{T}_\psi$ is $sat(\psi) \stackrel{\text{def}}{=} sat(p) \cap sat(\mathbf{XG}p) = \{s_1\}$.

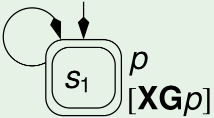
(iii) Since s_1 is the only state in $sat(\mathbf{G}p)$, then s_1 is the only successor of itself, so that the only relevant transition is a self-loop over s_1.
(One can also —un-necessarily— draw all transitions from states where $\mathbf{XG}p$ holds into $\{s_1\}$ and from from states where $\neg\mathbf{XG}p$ holds into $\{s_2, s_3, s_4\}$.)

(iv) Since there is no "**U**" subformula, we must add a $\mathbf{AGAF}\top$ fairness condition, which is equivalent to say that all states belong to the fairness condition.

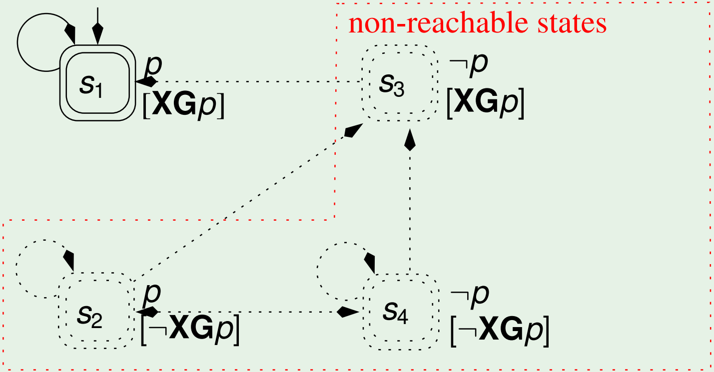
]

Ex: Symbolic LTL Model Checking (cont.)

[Solution:



or, alternatively without simplifications:



]