Introduction to Formal Methods
Chapter 06: Fair CTL Model Checking

Roberto Sebastiani and Stefano Tonetta

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: http://disi.unitn.it/rseba/DIDATTICA/fm2020/
Teaching assistant: Enrico Magnago – enrico.magnago@unitn.it

CDLM in Informatica, academic year 2019-2020

last update: Monday 18th May, 2020, 14:48

Outline

Outline

1. Fair CTL Model Checking: Generalities

2. Checking Fair **EG** (Explicit-State)

3. Checking Fair **EG** (Symbolic)

Fairness/Justice

An event E occurs infinitely often. Example:

- Let R_i be true iff the process i is running.
- Fairness: every process runs infinitely often.

$$\bigwedge_i \mathbf{GF} R_i$$

  (It is often used as condition for other properties.)
- You cannot express the condition in CTL:
  - $\mathbf{GF}\phi = \mathbf{AGAF}\phi$,
  - but $(\mathbf{GF}\phi) \rightarrow \psi \neq (\mathbf{AGAF}\phi) \rightarrow \psi$,
  - because $\mathbf{FG}\phi' \neq \mathbf{EFEG}\phi'$.
  - There is no CTL formula equivalent to $\mathbf{FG}\phi$ ($\mathbf{FG}\phi \neq \mathbf{AFAG}\phi$ and $\mathbf{FG}\phi \neq \mathbf{EFEG}\phi$).

Fairness/Justice

An event E occurs infinitely often. Example:

- Let R_i be true iff the process i is running.
- Fairness: every process runs infinitely often.

$$\bigwedge_i \mathbf{GF} R_i$$

(It is often used as condition for other properties.)

- You cannot express the condition in CTL:
  - $\mathbf{GF}\phi = \mathbf{AGAF}\phi$,
  - but $(\mathbf{GF}\phi) \rightarrow \psi \neq (\mathbf{AGAF}\phi) \rightarrow \psi$,
  - because $\mathbf{FG}\phi' \neq \mathbf{EFEG}\phi'$.
  - There is no CTL formula equivalent to $\mathbf{FG}\phi$ ($\mathbf{FG}\phi \neq \mathbf{AFAG}\phi$ and $\mathbf{FG}\phi \neq \mathbf{EFEG}\phi$).

Fairness/Justice

An event E occurs infinitely often. Example:

- Let R_i be true iff the process i is running.
- Fairness: every process runs infinitely often.

$$\bigwedge_i \mathbf{GF} R_i$$

(It is often used as condition for other properties.)

- You cannot express the condition in CTL:
  - $\mathbf{GF}\phi = \mathbf{AGAF}\phi$,
  - but $(\mathbf{GF}\phi) \rightarrow \psi \neq (\mathbf{AGAF}\phi) \rightarrow \psi$,
  - because $\mathbf{FG}\phi' \neq \mathbf{EFEG}\phi'$.
  - There is no CTL formula equivalent to $\mathbf{FG}\phi$ ($\mathbf{FG}\phi \neq \mathbf{AFAG}\phi$ and $\mathbf{FG}\phi \neq \mathbf{EFEG}\phi$).

Fair CTL Model Checking

- Let M be the KS $M = \langle S, S_0, R, L, AP \rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, ..., F_n\}$.
- A path is fair iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
  - $M_f, s_i \models \mathbf{A}\phi$ iff $\pi \models \phi$ for every fair path $\pi = s_i, s_{i+1}, \ldots$
  - $M_f, s_i \models \mathbf{E}\phi$ iff $\pi \models \phi$ for some fair path $\pi = s_i, s_{i+1}, \ldots$
- We introduce $\mathbf{A}_f$ and $\mathbf{E}_f$, the restricted versions of the quantifiers $\mathbf{A}$ and $\mathbf{E}$:
  - $M_f, s \models \mathbf{A}\phi$ iff $M, s \models \mathbf{A}_f\phi$
  - $M_f, s \models \mathbf{E}\phi$ iff $M, s \models \mathbf{E}_f\phi$
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_f iff $M_f, s \models \mathbf{EG}\mathit{true}$.

Fair CTL Model Checking

- Let M be the KS $M = \langle S, S_0, R, L, AP \rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, ..., F_n\}$.
- A path is fair iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
  - $M_f, s_i \models \mathbf{A}\phi$ iff $\pi \models \phi$ for every fair path $\pi = s_i, s_{i+1}, \dots$
  - $M_f, s_i \models \mathbf{E}\phi$ iff $\pi \models \phi$ for some fair path $\pi = s_i, s_{i+1}, \dots$
- We introduce $\mathbf{A}_f$ and $\mathbf{E}_f$, the restricted versions of the quantifiers **A** and **E**:
  - $M_f, s \models \mathbf{A}\phi$ iff $M, s \models \mathbf{A}_f\phi$
  - $M_f, s \models \mathbf{E}\phi$ iff $M, s \models \mathbf{E}_f\phi$
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_f iff $M_f, s \models \mathbf{EG}\mathit{true}$.

Fair CTL Model Checking

- Let M be the KS $M = \langle S, S_0, R, L, AP\rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT\rangle$, $FT = \{F_1, ..., F_n\}$.
- A path is fair iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
  - $M_f, s_i \models \mathbf{A}\phi$ iff $\pi \models \phi$ for every fair path $\pi = s_i, s_{i+1}, ...$
  - $M_f, s_i \models \mathbf{E}\phi$ iff $\pi \models \phi$ for some fair path $\pi = s_i, s_{i+1}, ...$
- We introduce $\mathbf{A}_f$ and $\mathbf{E}_f$, the restricted versions of the quantifiers $\mathbf{A}$ and $\mathbf{E}$:
  - $M_f, s \models \mathbf{A}\phi$ iff $M, s \models \mathbf{A}_f\phi$
  - $M_f, s \models \mathbf{E}\phi$ iff $M, s \models \mathbf{E}_f\phi$
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_f iff $M_f, s \models \mathbf{EG}\mathit{true}$.

Fair CTL Model Checking

- Let M be the KS $M = \langle S, S_0, R, L, AP \rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, ..., F_n\}$.
- A path is fair iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
  - $M_f, s_i \models \mathbf{A}\phi$ iff $\pi \models \phi$ for every fair path $\pi = s_i, s_{i+1}, ...$
  - $M_f, s_i \models \mathbf{E}\phi$ iff $\pi \models \phi$ for some fair path $\pi = s_i, s_{i+1}, ...$
- We introduce $\mathbf{A}_f$ and $\mathbf{E}_f$, the restricted versions of the quantifiers **A** and **E**:
  - $M_f, s \models \mathbf{A}\phi$ iff $M, s \models \mathbf{A}_f\phi$
  - $M_f, s \models \mathbf{E}\phi$ iff $M, s \models \mathbf{E}_f\phi$
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_f iff $M_f, s \models \mathbf{EG}\mathit{true}$.

Fair CTL Model Checking

- Let M be the KS $M = \langle S, S_0, R, L, AP \rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, ..., F_n\}$.
- A path is fair iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
  - $M_f, s_i \models \mathbf{A}\phi$ iff $\pi \models \phi$ for every fair path $\pi = s_i, s_{i+1}, ...$
  - $M_f, s_i \models \mathbf{E}\phi$ iff $\pi \models \phi$ for some fair path $\pi = s_i, s_{i+1}, ...$
- We introduce $\mathbf{A}_f$ and $\mathbf{E}_f$, the restricted versions of the quantifiers **A** and **E**:
  - $M_f, s \models \mathbf{A}\phi$ iff $M, s \models \mathbf{A}_f\phi$
  - $M_f, s \models \mathbf{E}\phi$ iff $M, s \models \mathbf{E}_f\phi$
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_f iff $M_f, s \models \mathbf{EG}\mathit{true}$.

Fair CTL Model Checking

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

Fair CTL Model Checking

- In order to solve the fair CTL model checking problem, we must be able to compute:
  - $[\mathbf{E}_f\mathbf{X}(\varphi)]$
  - $[\mathbf{E}_f(\varphi\mathbf{U}\psi)]$
  - $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Suppose we have a procedure `Check_FairEG` to compute $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Let $fair = \mathbf{E}_f\mathbf{G}true$. ($M, s \models \mathbf{E}_f\mathbf{G}true$ 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

- In order to solve the fair CTL model checking problem, we must be able to compute:
  - $[\mathbf{E}_f\mathbf{X}(\varphi)]$
  - $[\mathbf{E}_f(\varphi\mathbf{U}\psi)]$
  - $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Suppose we have a procedure `Check_FairEG` to compute $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Let $fair = \mathbf{E}_f\mathbf{G}true$. ($M, s \models \mathbf{E}_f\mathbf{G}true$ 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

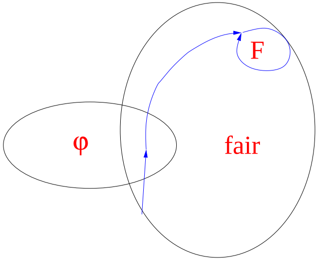
- In order to solve the fair CTL model checking problem, we must be able to compute:
  - $[\mathbf{E}_f\mathbf{X}(\varphi)]$
  - $[\mathbf{E}_f(\varphi\mathbf{U}\psi)]$
  - $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Suppose we have a procedure `Check_FairEG` to compute $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Let $fair = \mathbf{E}_f\mathbf{G}true$. ($M, s \models \mathbf{E}_f\mathbf{G}true$ 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:
  - $\mathbf{E}_f\mathbf{X}(\varphi) = \mathbf{EX}(\varphi \wedge fair)$
  - $\mathbf{E}_f(\varphi\mathbf{U}\psi) = \mathbf{E}(\varphi\mathbf{U}(\psi \wedge fair))$

Fair CTL Model Checking

- In order to solve the fair CTL model checking problem, we must be able to compute:
  - $[\mathbf{E}_f\mathbf{X}(\varphi)]$
  - $[\mathbf{E}_f(\varphi\mathbf{U}\psi)]$
  - $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Suppose we have a procedure `Check_FairEG` to compute $[\mathbf{E}_f\mathbf{G}\varphi]$.
- Let $fair = \mathbf{E}_f\mathbf{G}true$. ($M, s \models \mathbf{E}_f\mathbf{G}true$ 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

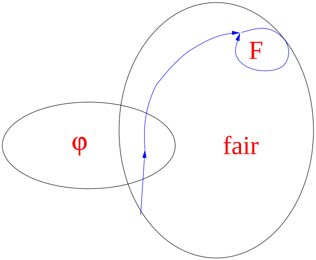
- $\mathbf{E}_f\mathbf{X}(\varphi) \equiv \mathbf{EX}(\varphi \wedge \mathit{fair})$:



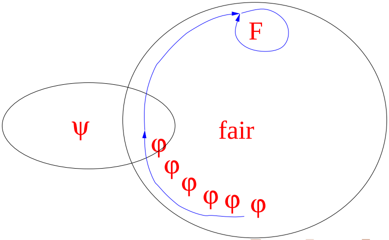
- $\mathbf{E}_f(\varphi \mathbf{U} \psi) \equiv \mathbf{E}(\varphi \mathbf{U}(\psi \wedge \mathit{fair}))$:

Fair CTL Model Checking

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



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



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.

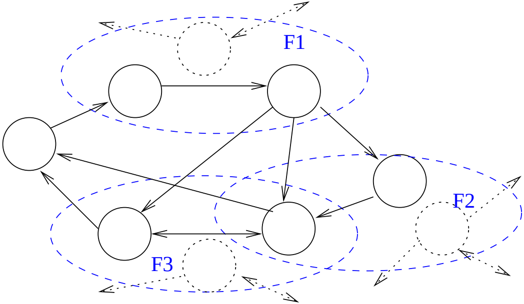
Given a fair Kripke model M, a fair non-trivial SCC is an SCC with at least one edge that contains at least one state for every fair condition

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

SCC-based Check_FairEG

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

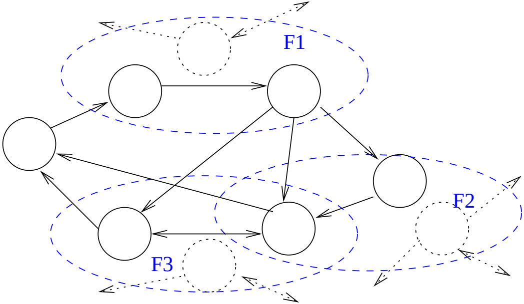
Given a fair Kripke model M, a fair non-trivial SCC is an SCC with at least one edge that contains at least one state for every fair condition
$\Longrightarrow$ all states in a fair (non-trivial) SCC are fair states



SCC-based Check_FairEG

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`)`).

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`)`).

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(`$[\phi], C$`)`).

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`$([\phi], C)$).

SCC-based Check_FairEG (cont.)

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

(i) restrict the graph of M to $[\phi]$;

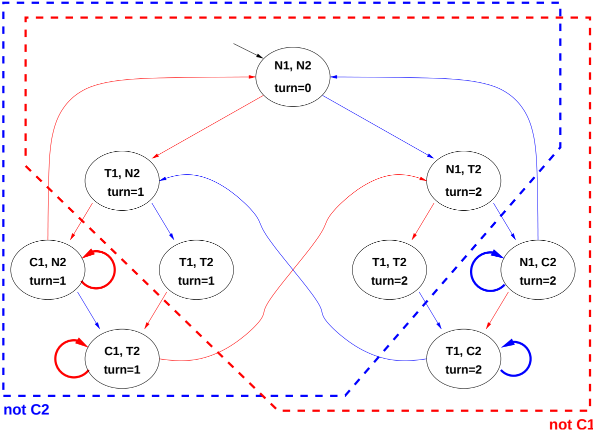
(ii) find all fair non-trivial SCCs C_i

(iii) build $C := \cup_i C_i$;

(iv) compute the states that can reach C (`Check_EU`($[\phi]$, C)).

Example: Check_FairEG

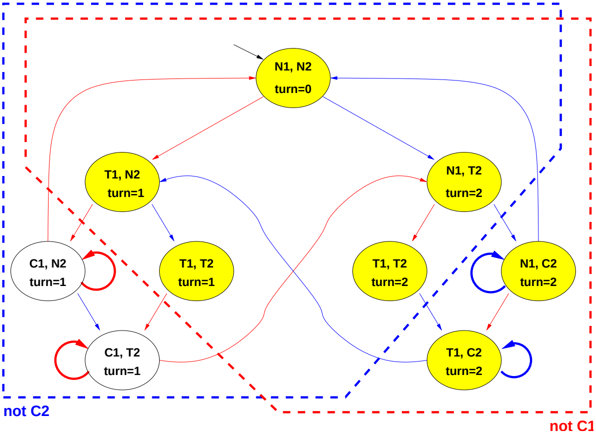
F := {{ not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

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

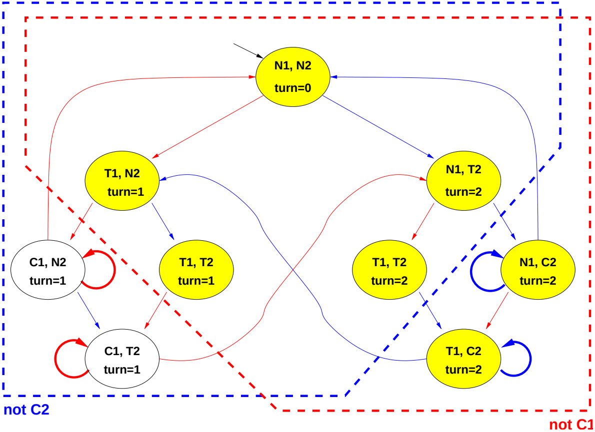


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

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

Example: Check_FairEG

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

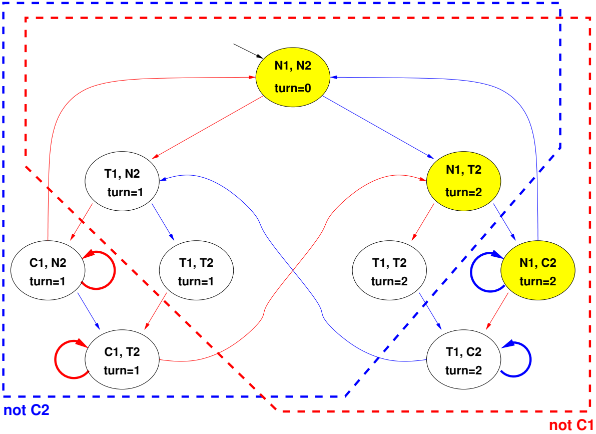


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

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

Example: Check_FairEG

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

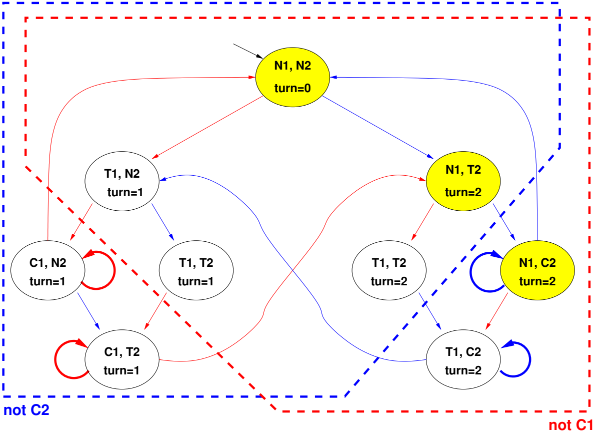


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

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

Example: Check_FairEG

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

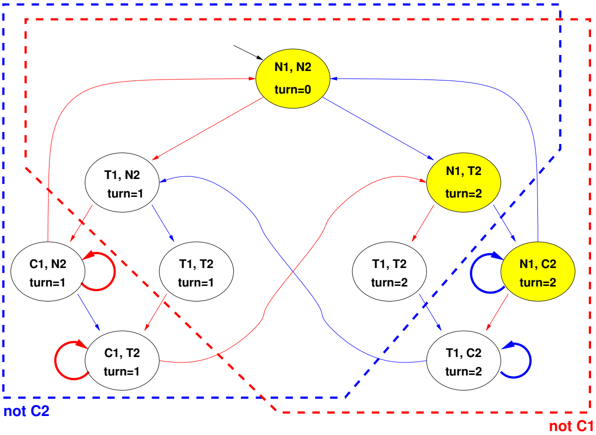


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

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

Example: Check_FairEG

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

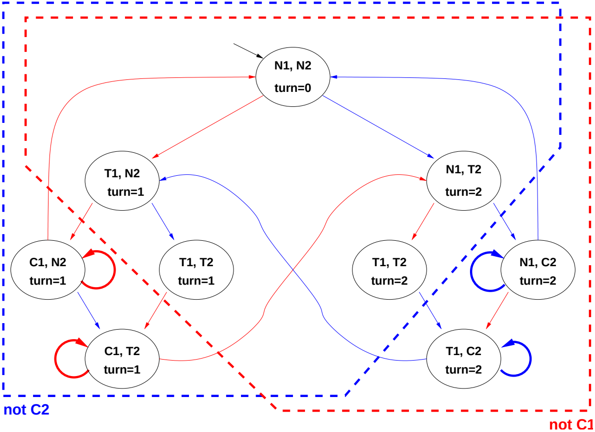


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

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

Example: Check_FairEG

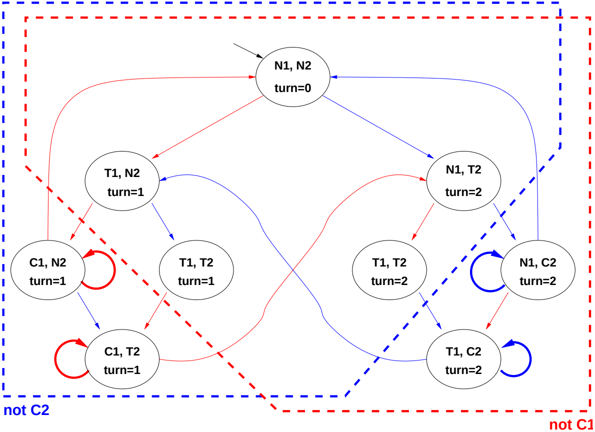
F := {{ not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

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

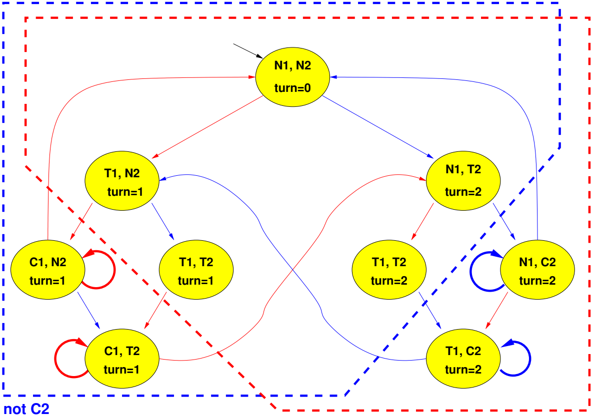


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

Check_FairEU($\top, \varphi$) = Check_EU($\top, \varphi \wedge$ *fair*)

Example: Check_FairEG

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

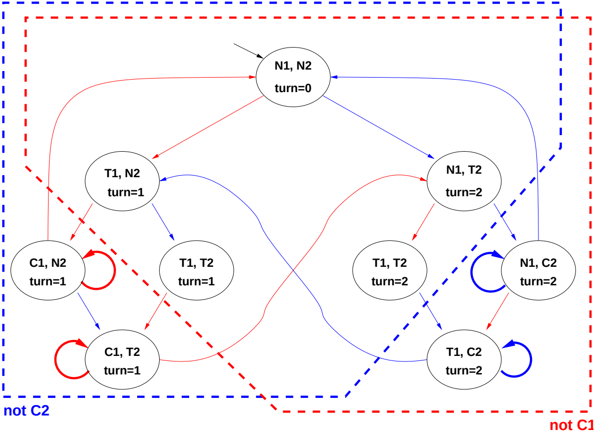


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

fair=Check_FairEG($\top$)

Example: Check_FairEG

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

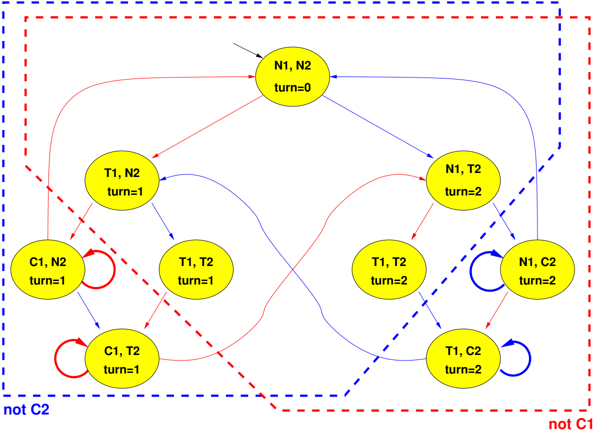


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

Check_FairEU($\top, \varphi$) = Check_EU($\top, \varphi \wedge$ *fair*)

Example: Check_FairEG

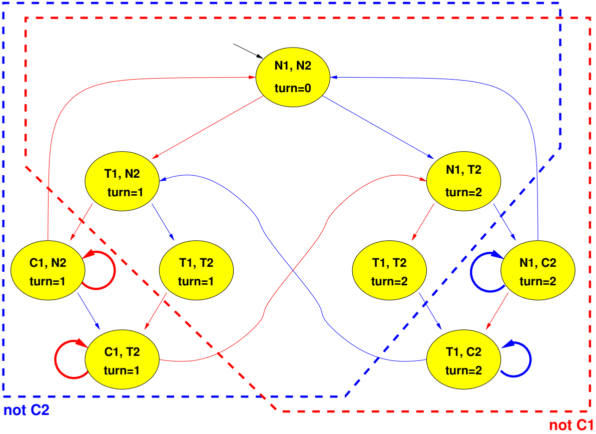
F := {{ not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

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



$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

$\Longrightarrow$ Property Verified

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.
- We want an algorithm based on the preimage computation.

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.
- We want an algorithm based on the preimage computation.

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.
- We want an algorithm based on the preimage computation.

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.
- We want an algorithm based on the preimage computation.

Outline

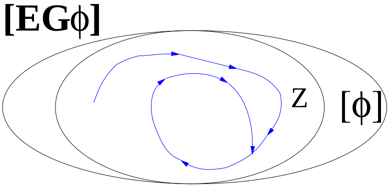
Emerson-Lei Algorithm

Recall the fixpoint characterization of **EG**:

- $[\mathbf{EG}\phi] = \nu Z.([\phi] \cap [\mathbf{EX}Z])$
  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 $\mathbf{E}_f\mathbf{G}$ similarly:

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



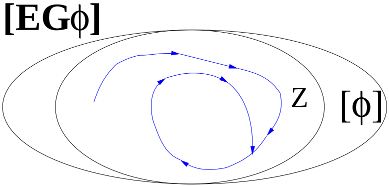
Emerson-Lei Algorithm

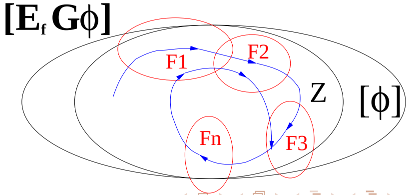
Recall the fixpoint characterization of **EG**:

- $[\mathbf{EG}\phi] = \nu Z.([\phi] \cap [\mathbf{EX}Z])$
  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 $\mathbf{E}_f\mathbf{G}$ similarly:

- $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \cap F_i))])$
  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 Fi in FT
        Y:= Check_EU(Z,Fi∩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 Fi in FT
        Y:= Check_EU(Z',Fi∩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)

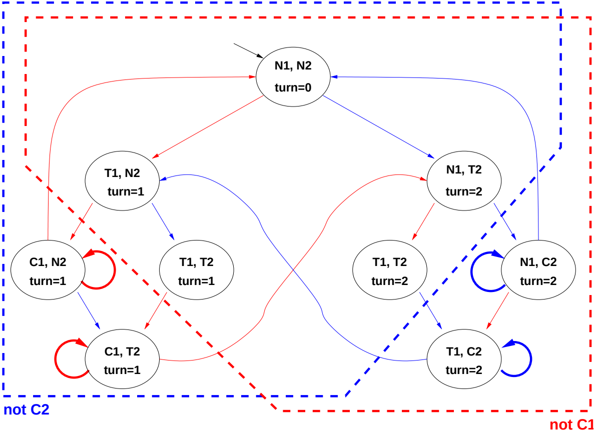
Recall: $[\mathbf{E}_f\mathbf{G}\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT}[\mathbf{EX}\ \mathbf{E}(Z\mathbf{U}(Z \wedge F_i))])$

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

Symbolic version.

Example: normal Check_EG

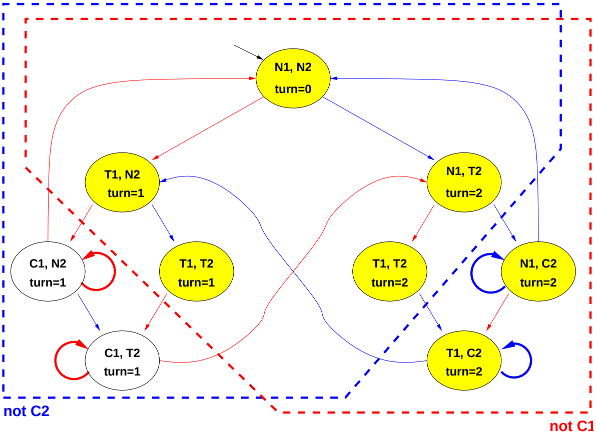
F := { { not C1},{not C2}}



$$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$$

Example: normal Check_EG

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

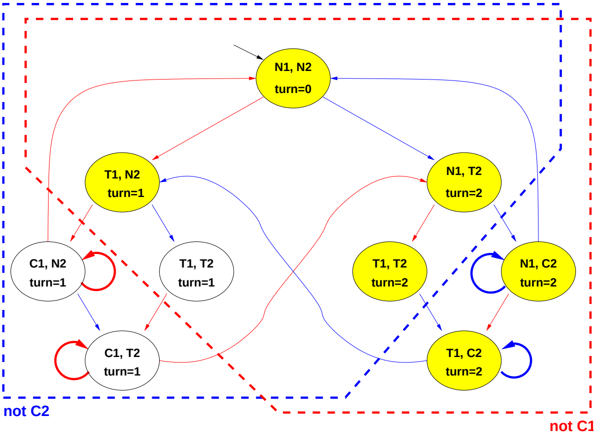


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EG}g = \nu Z.g \wedge \mathbf{EX}Z$

Example: normal Check_EG

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

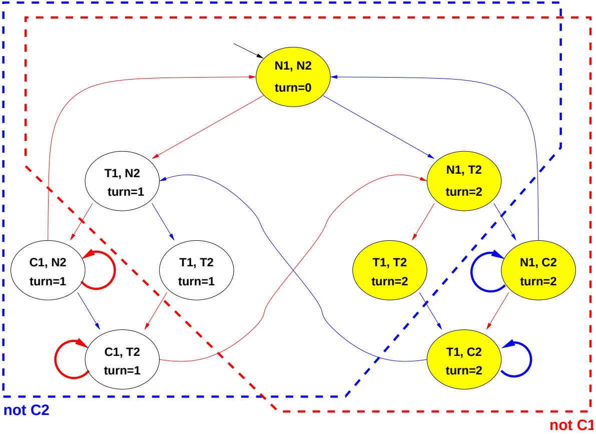


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EG}g = \nu Z.g \wedge \mathbf{EX}Z$

Example: normal Check_EG

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

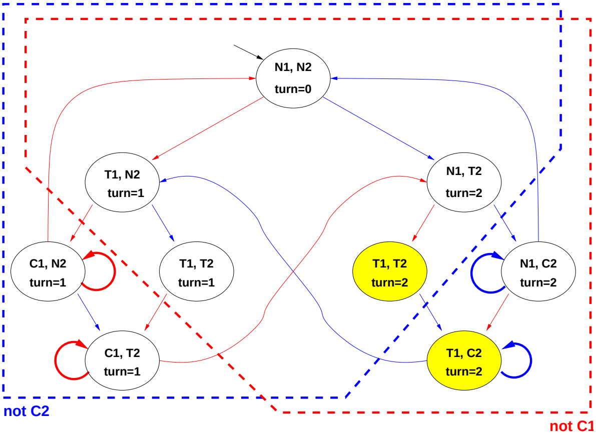


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EG}g = \nu Z.g \wedge \mathbf{EX}Z$

Example: normal Check_EG

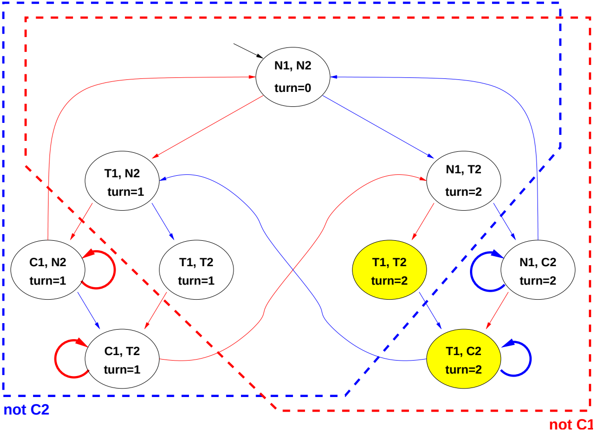
F := { { not C1},{not C2}}



$$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$$

Example: normal Check_EG

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

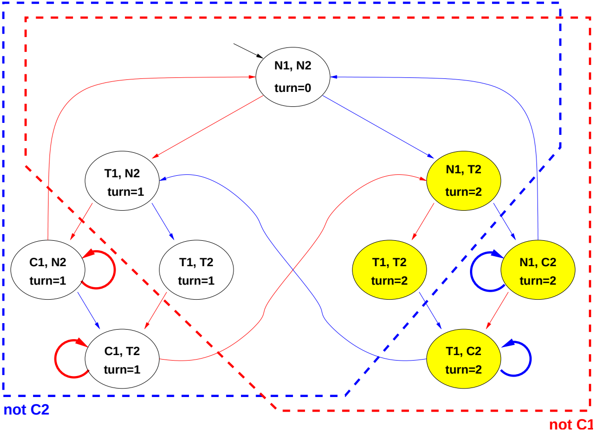


$$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$$

$$\mathbf{EF}g = \mathbf{E}(\top \mathbf{U} g) = \mu Z.g \vee (\top \wedge \mathbf{EX}Z) = \mu Z.g \vee \mathbf{EX}Z$$

Example: normal Check_EG

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

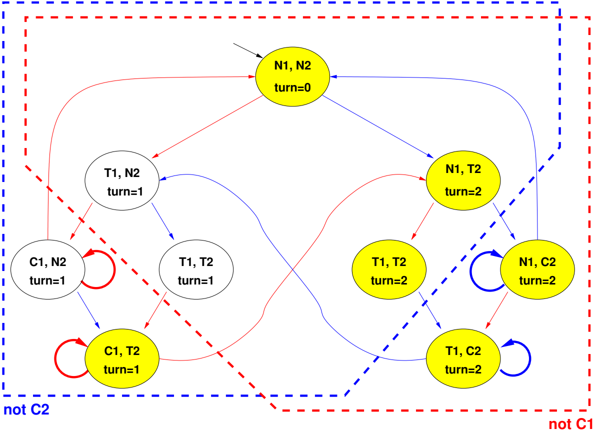


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EF}g = \mathbf{E}(\top \mathbf{U} g) = \mu Z.g \vee (\top \wedge \mathbf{EX}Z) = \mu Z.g \vee \mathbf{EX}Z$

Example: normal Check_EG

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

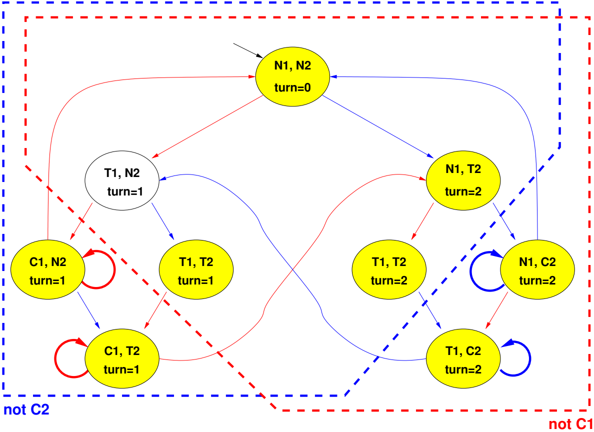


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EF}g = \mathbf{E}(\top\mathbf{U}g) = \mu Z.g \vee (\top \wedge \mathbf{EX}Z) = \mu Z.g \vee \mathbf{EX}Z$

Example: normal Check_EG

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

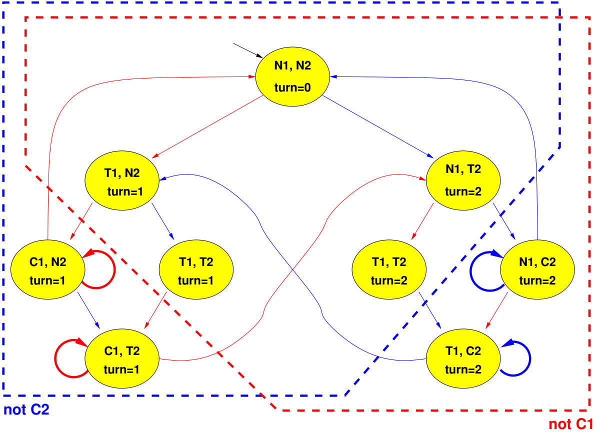


$$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$$

$$\mathbf{EF}g = \mathbf{E}(\top \mathbf{U} g) = \mu Z.g \vee (\top \wedge \mathbf{EX}Z) = \mu Z.g \vee \mathbf{EX}Z$$

Example: normal Check_EG

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

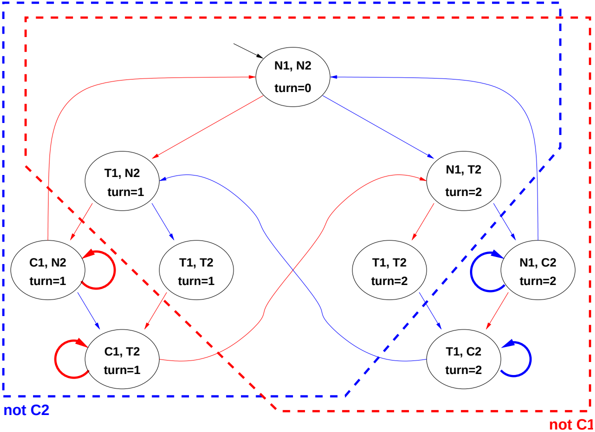


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\mathbf{EF}g = \mathbf{E}(\top\mathbf{U}g) = \mu Z.g \vee (\top \wedge \mathbf{EX}Z) = \mu Z.g \vee \mathbf{EX}Z$

Example: normal Check_EG

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

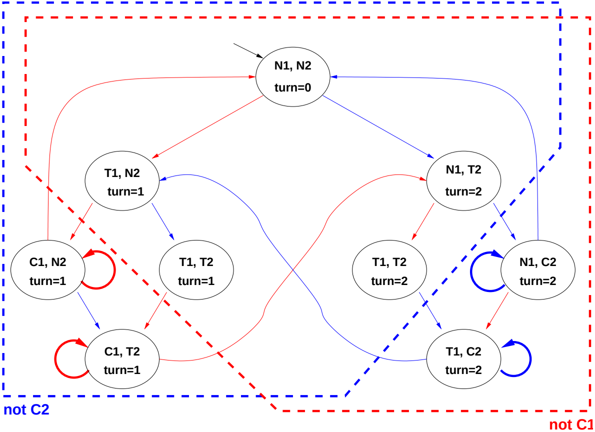


$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) = \neg\mathbf{EF}(T_1 \wedge \mathbf{EG}\neg C_1)$

$\Longrightarrow$ Property not verified

Example: Check_FairEG

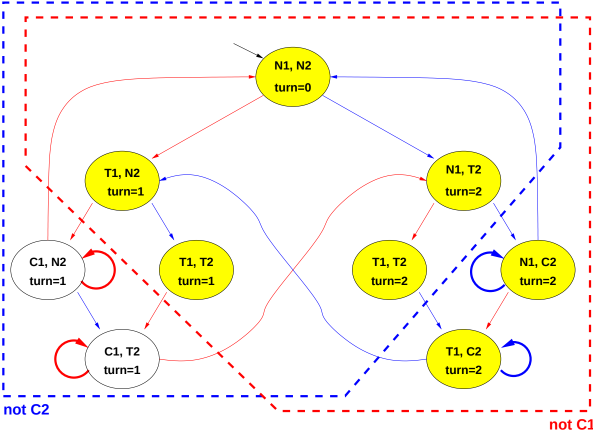
F := { { not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

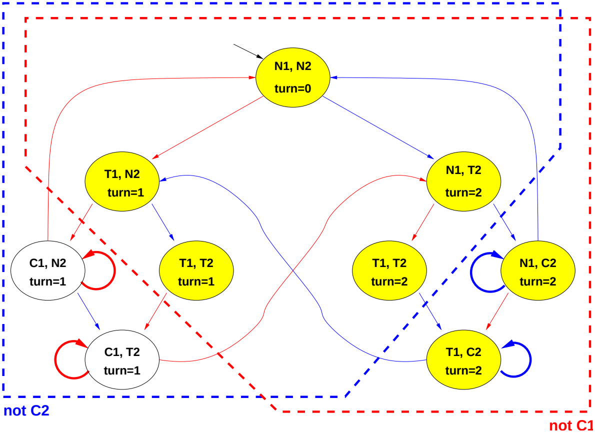
F := { { not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

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

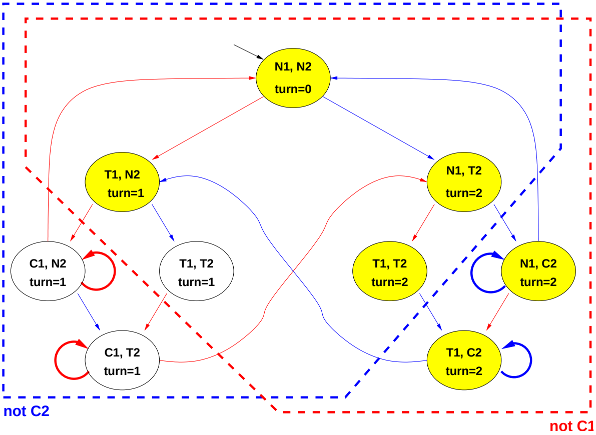


$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

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

Example: Check_FairEG

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

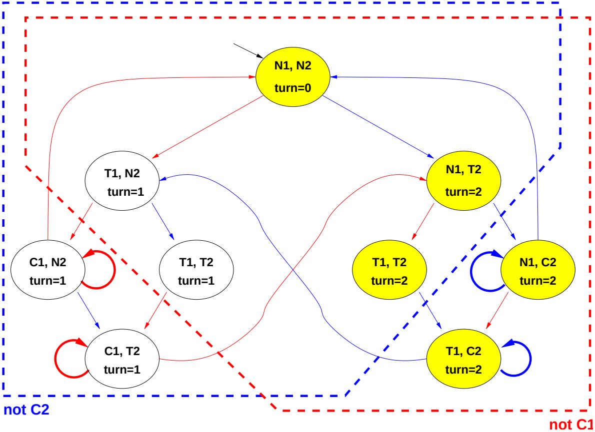


$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

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

Example: Check_FairEG

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

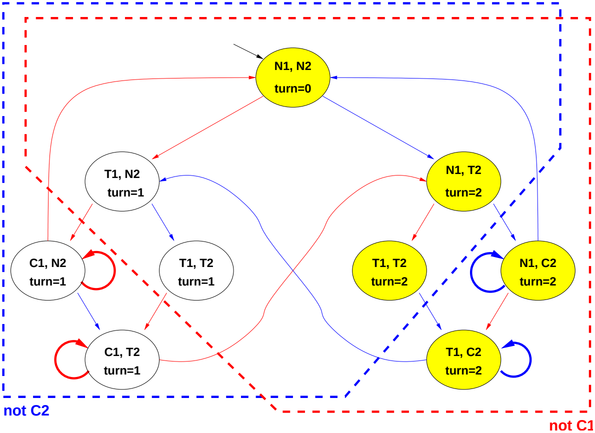


$$M \models \mathbf{A}_f\mathbf{G}(T_1 \to \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

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

Example: Check_FairEG

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

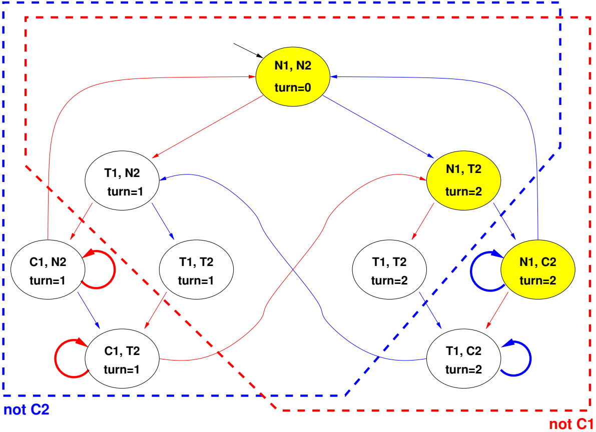


$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

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

Example: Check_FairEG

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

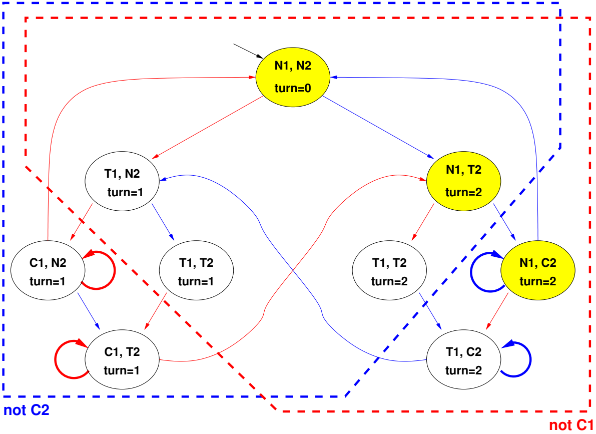


$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

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

Example: Check_FairEG

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

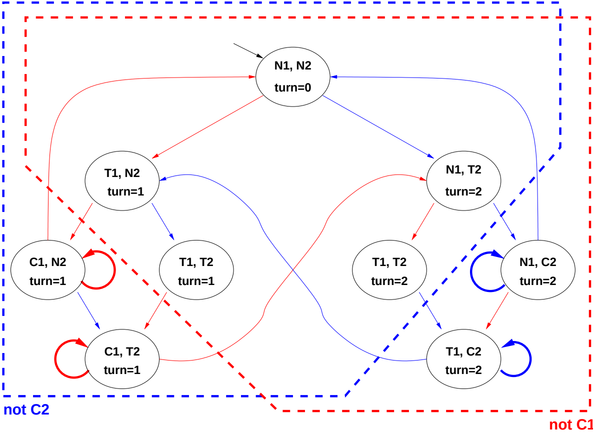


$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

Fixpoint reached

Example: Check_FairEG

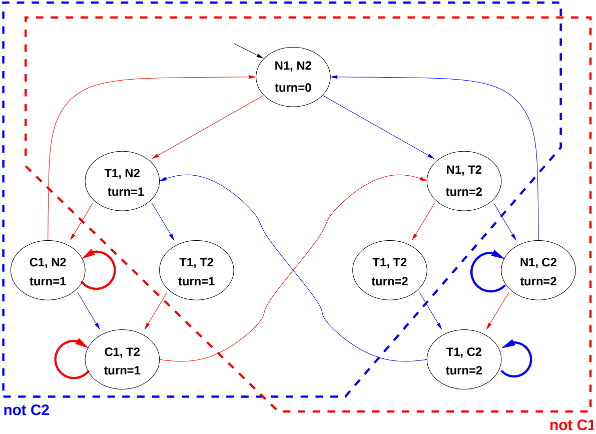
F := { { not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

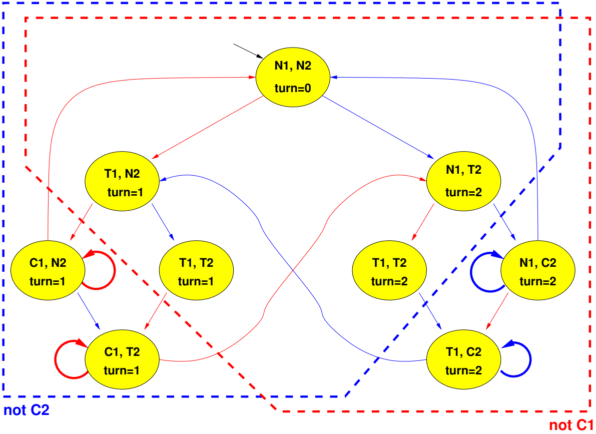
F := { { not C1},{not C2}}



$$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$$

Example: Check_FairEG

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



$M \models \mathbf{A}_f\mathbf{G}(T_1 \rightarrow \mathbf{A}_f\mathbf{F}C_1) = \neg\mathbf{E}_f\mathbf{F}(T_1 \wedge \mathbf{E}_f\mathbf{G}\neg C_1)$

$\Longrightarrow$ Property verified