

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

Copyright notice: *some material (text, figures) displayed in these slides is courtesy of R. Alur, M. Benerecetti, A. Cimatti, M. Di Natale, P. Pandya, M. Pistore, M. Roveri, and S. Tonetta, who detain its copyright. Some examples displayed in these slides are taken from [Clarke, Grunberg & Peled, "Model Checking", MIT Press], and their copyright is detained by the authors. All the other material is copyrighted by Roberto Sebastiani. Every commercial use of this material is strictly forbidden by the copyright laws without the authorization of the authors. No copy of these slides can be displayed in public without containing this copyright notice.*

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 $\mathbf{FG}\phi \neq \mathbf{EFEG}\phi$).

Fair CTL Model Checking

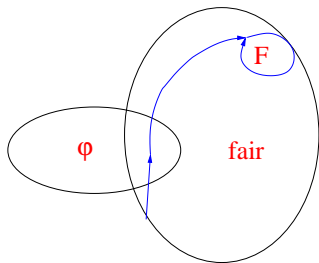
- Let M be the KS $M = \langle S, S_0, R, L, AP \rangle$ and M_f the fair version $M_f = \langle S, S_0, R, L, AP, FT \rangle$, $FT = \{F_1, \dots, F_n\}$.
- A path is **fair** iff it visits every F_i infinitely often.
- Fair CTL model checking restricts the path quantifiers to *fair paths*:
 - $M_f, s_i \models \mathbf{A}\phi$ 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 \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 $M_f, s \models \mathbf{EG}true$.

Fair CTL Model Checking

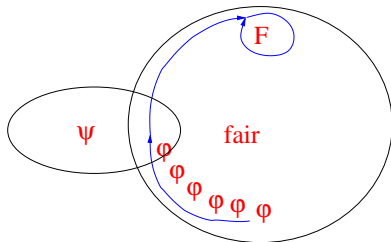
- In order to solve the fair CTL model checking problem, we must be able to compute:
 - $[E_f X(\varphi)]$
 - $[E_f(\varphi U \psi)]$
 - $[E_f G\varphi]$.
- Suppose we have a procedure `Check_FairEG` to compute $[E_f G\varphi]$.
- Let $fair = E_f Gtrue$. ($M, s \models E_f Gtrue$ if s is a fair state.)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

Fair CTL Model Checking

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



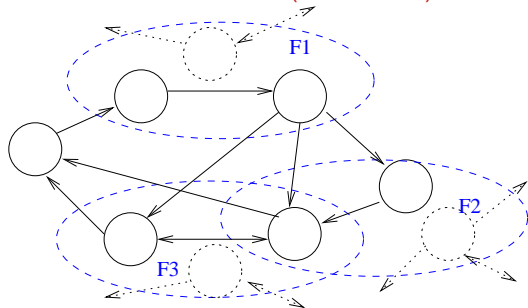
- $\mathbf{E}_f(\varphi \mathbf{U} \psi) \equiv \mathbf{E}(\varphi \mathbf{U}(\psi \wedge \text{fair}))$:



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

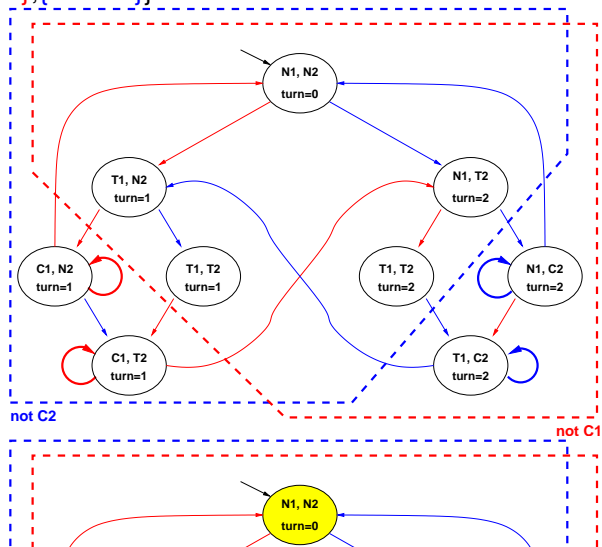


SCC-based Check_FairEG (cont.)

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

Example: Check_FairEG

$$F := \{ \{ \text{not } C1 \}, \{ \text{not } C2 \} \}$$


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.

Emerson-Lei Algorithm

Recall the fixpoint characterization of **EG**:

- $[EG\phi] = \nu Z.([\phi] \cap [EXZ])$

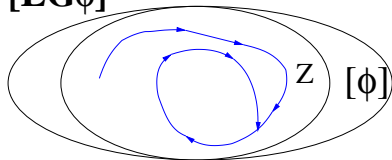
The greatest set Z s.t. every state z in Z satisfies ϕ and reaches another state in Z in one step.

We can characterize E_fG similarly:

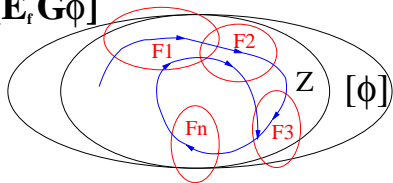
- $[E_fG\phi] = \nu Z.([\phi] \cap \bigcap_{F_i \in FT} [EX E(ZU(Z \cap F_i))])$

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.

$[EG\phi]$



$[E_fG\phi]$



Emerson-Lei Algorithm

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

```

state_set Check_FairEG( state_set [ $\phi$ ] ) {
  Z' := [ $\phi$ ];
  repeat
    Z := Z';
    for each Fi in FT
      Y := Check_EU(Z, Fi  $\cap$  Z);
      Z' := Z'  $\cap$  PreImage(Y);
    end for;
  until (Z' = Z);
  return Z;
}

```

Implementation of the above formula

Emerson-Lei Algorithm

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

```

state_set Check_FairEG( state_set [ $\phi$ ] ) {
   $Z' := [\phi]$ ;
  repeat
     $Z := Z'$  ;
    for each  $F_i$  in FT
       $Y := \text{Check\_EU}(Z', F_i \cap Z')$  ;
       $Z' := Z' \cap \text{PreImage}(Y)$  ;
    end for;
  until ( $Z' = Z$ ) ;
  return  $Z$ ;
}

```

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

Emerson-Lei Algorithm (symbolic version)

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

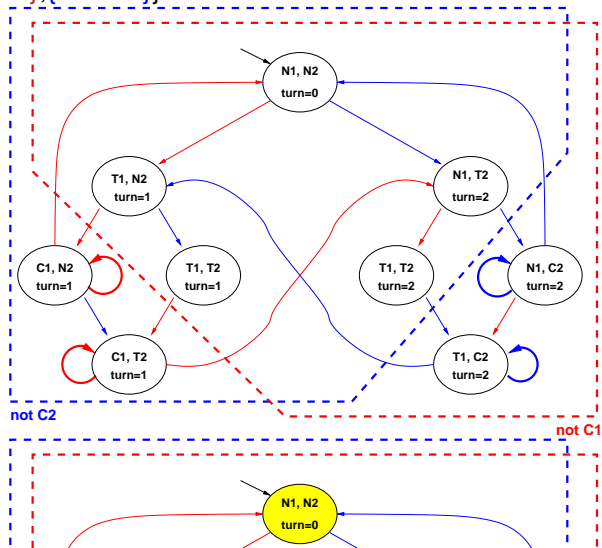
```

Obdd Check_FairEG ( Obdd  $\phi$  ) {
   $Z' := \phi$ ;
  repeat
     $Z := Z'$ ;
    for each  $Fi$  in FT
       $Y := \text{Check\_EU}(Z', Fi \wedge Z')$ ;
       $Z' := Z' \wedge \text{PreImage}(Y)$ ;
    end for;
  until ( $Z' \leftrightarrow Z$ );
  return  $Z$ ;
}

```

Symbolic version.

Example: normal Check_EG

$$F := \{ \{ \text{not } C1 \}, \{ \text{not } C2 \} \}$$


Example: Check_FairEG

$$F := \{ \{ \text{not } C1 \}, \{ \text{not } C2 \} \}$$
