Formal Methods:
Module I: Automated Reasoning
Ch. 04: **Linear Temporal Logic**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: http://disi.unitn.it/rseba/DIDATTICA/fm2021/
Teaching assistant: Giuseppe Spallitta – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2020-2021

last update: Tuesday 13th April, 2021, 13:56

Outline

Outline

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems
   - Properties
2. Linear Temporal Logic – LTL
   - Generalities on Temporal Logics
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
3. Exercises

Kripke Models

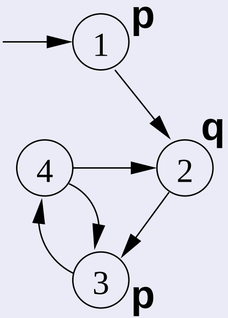
- Theoretical role: the semantic framework for a variety of logics
  - Modal Logics
  - Description Logics
  - Temporal Logics
  - ...
- Practical role: used to describe reactive systems:
  - nonterminating systems with infinite behaviors (e.g. communication protocols, hardware circuits);
  - represent the dynamic evolution of modeled systems;
  - a state includes values to state variables, program counters, content of communication channels.
  - can be animated and validated before their actual implementation

Kripke Models

- Theoretical role: the semantic framework for a variety of logics
  - Modal Logics
  - Description Logics
  - Temporal Logics
  - ...
- Practical role: used to describe reactive systems:
  - nonterminating systems with infinite behaviors (e.g. communication protocols, hardware circuits);
  - represent the dynamic evolution of modeled systems;
  - a state includes values to state variables, program counters, content of communication channels.
  - can be animated and validated before their actual implementation

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

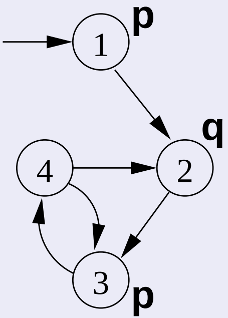


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

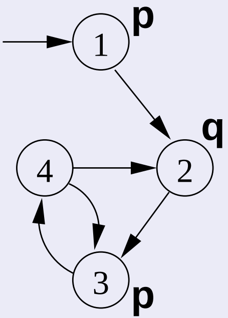


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

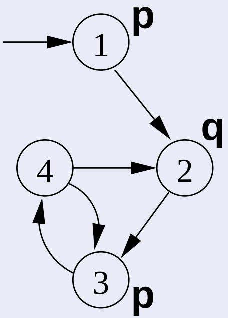


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

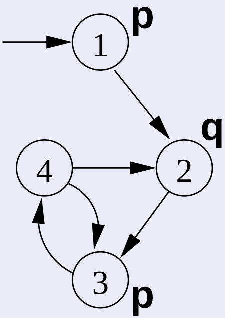


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

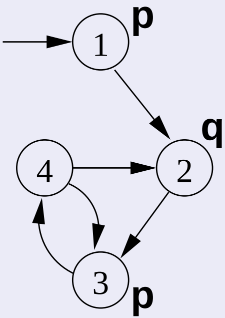


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L \rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

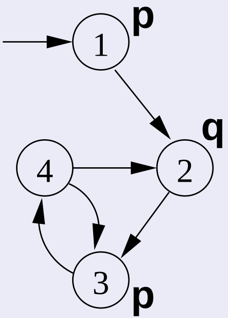


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L\rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

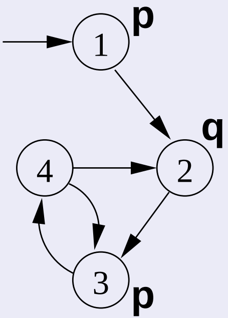


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L\rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

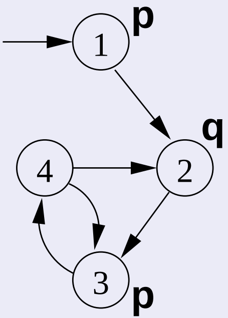


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

Kripke Model: Formal Definition

- A Kripke model $\langle S, I, R, AP, L\rangle$ consists of
  - a finite 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 $L : S \longmapsto 2^{AP}$.
- We assume R total: for every state s, there exists (at least) one state s' s.t. $(s, s') \in R$
- Sometimes we use variables with discrete bounded values $v_i \in \{d_1, ..., d_k\}$ (can be encoded with $\lceil log(k) \rceil$ Boolean variables)

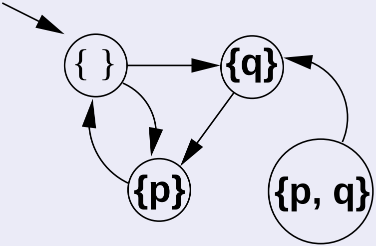


Remark

Unlike with other types of Automata (e.g., Buechi), in Kripke models the values of all variables are always assigned in each state.

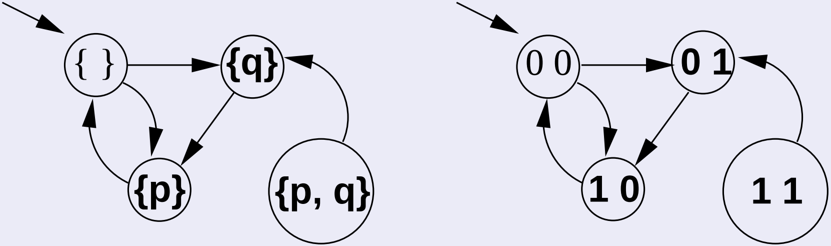
Kripke Structures: Two Alternative Representations:

- each state identifies univocally the values of the atomic propositions which hold there
- each state is labeled by a bit vector

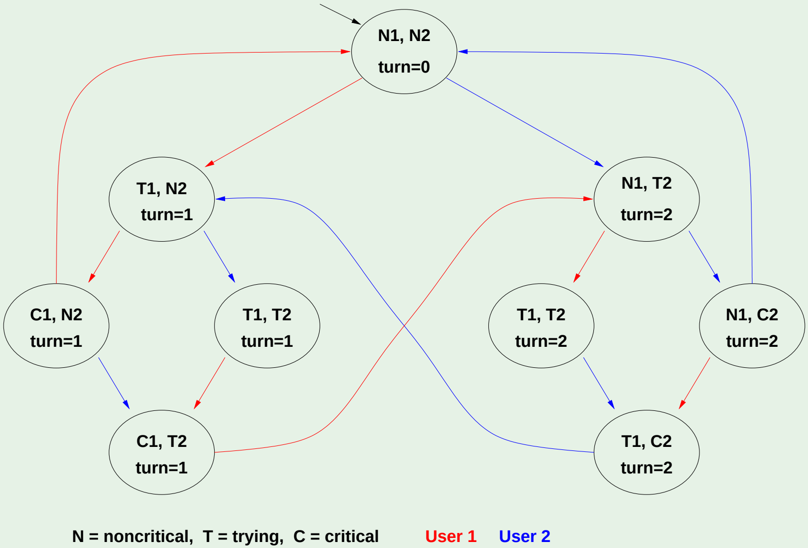


Kripke Structures: Two Alternative Representations:

- each state identifies univocally the values of the atomic propositions which hold there
- each state is labeled by a bit vector



Example: a Kripke model for mutual exclusion

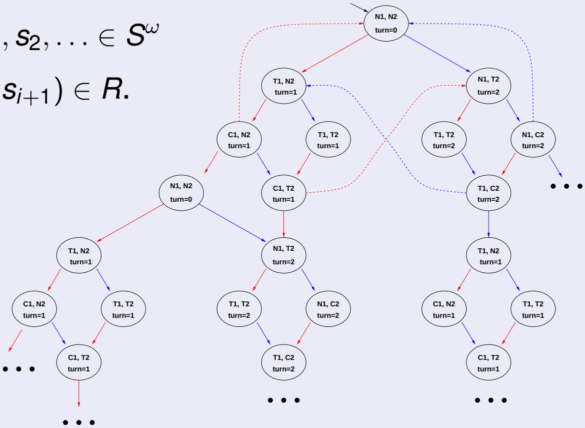


Path in a Kripke Model

A path in a Kripke model M is an infinite sequence of states

$$\pi = s_0, s_1, s_2, \ldots \in S^\omega$$

such that $s_0 \in I$ and $(s_i, s_{i+1}) \in R$.



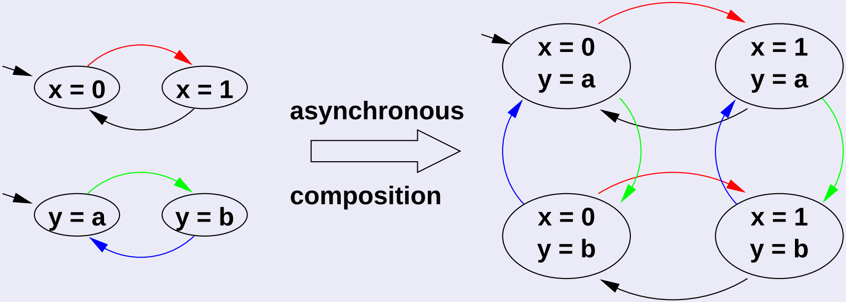
A state s is reachable in M if there is a path from the initial states to s.

Composing Kripke Models

- Complex Kripke Models are tipically obtained by composition of smaller ones
- Components can be combined via
  - asynchronous composition.
  - synchronous composition,

Asynchronous Composition

- Interleaving of evolution of components.
- At each time instant, one component is selected to perform a transition.



- Typical example: communication protocols.

Asynchronous Composition/Product: formal definition

Asynchronous product of Kripke models

Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$. Then the asynchronous product $M \stackrel{\text{def}}{=} M_1 || M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L \rangle$, where

- $S \subseteq S_1 \times S_2$ s.t., $\forall \langle s_1, s_2 \rangle \in S,\ \forall l \in AP_1 \cap AP_2, l \in L_1(s_1)$ *iff* $l \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ s.t. $I \subseteq S$
- $R(\langle s_1, s_2 \rangle, \langle t_1, t_2 \rangle)$ *iff* $(R_1(s_1, t_1)$ **and** $s_2 = t_2)$ **or** $(s_1 = t_1$ **and** $R_2(s_2, t_2))$
- $AP = AP_1 \cup AP_2$
- $L : S \longmapsto 2^{AP}$ s.t. $L(\langle s_1, s_2 \rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

Note: combined states must agree on the values of Boolean variables.

Asynchronous composition is associative:
$(...(M_1||M_2)||...)||M_n) = (M1||(M_2||(...||M_n)...) = M_1||M_2||...||M_n$

Asynchronous Composition/Product: formal definition

Asynchronous product of Kripke models

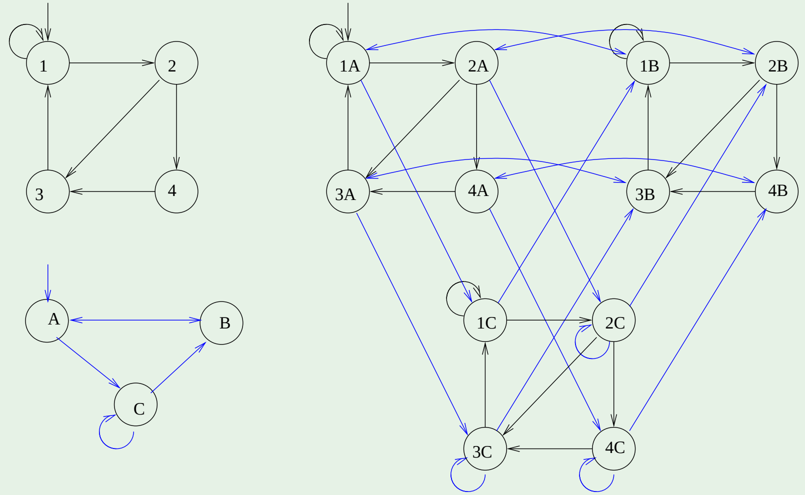
Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$. Then the asynchronous product $M \stackrel{\text{def}}{=} M_1 || M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L \rangle$, where

- $S \subseteq S_1 \times S_2$ s.t.,
  $\forall \langle s_1, s_2 \rangle \in S, \ \forall l \in AP_1 \cap AP_2, l \in L_1(s_1) \textit{ iff } l \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ s.t. $I \subseteq S$
- $R(\langle s_1, s_2 \rangle, \langle t_1, t_2 \rangle) \textit{ iff } \begin{array}{l} (R_1(s_1, t_1) \textbf{ and } s_2 = t_2) \textbf{ or} \\ (s_1 = t_1 \textbf{ and } R_2(s_2, t_2)) \end{array}$
- $AP = AP_1 \cup AP_2$
- $L : S \longmapsto 2^{AP}$ s.t. $L(\langle s_1, s_2 \rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

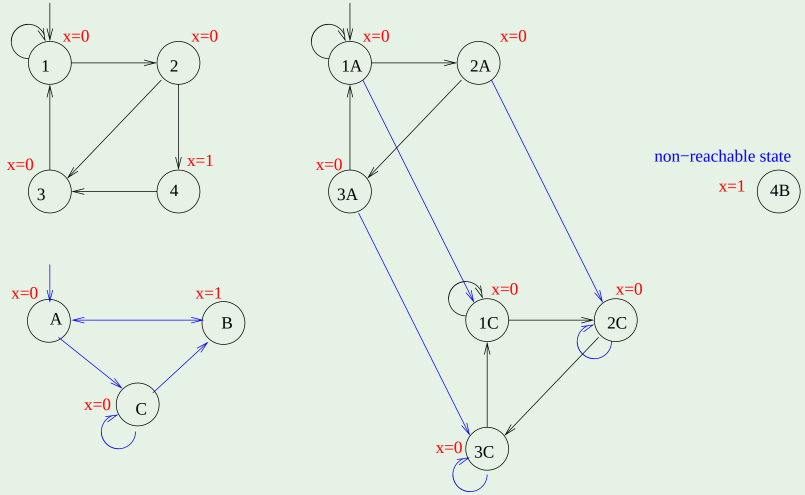
Note: combined states must agree on the values of Boolean variables.

Asynchronous composition is associative:
$(...(M_1||M_2)||...)||M_n) = (M1||(M_2||(...||M_n)...) = M_1||M_2||...||M_n$

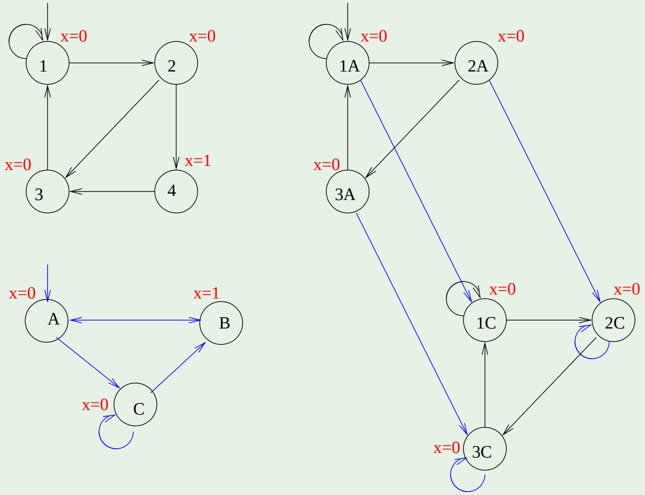
Asynchronous Composition: Example 1



Asynchronous Composition: Example 2

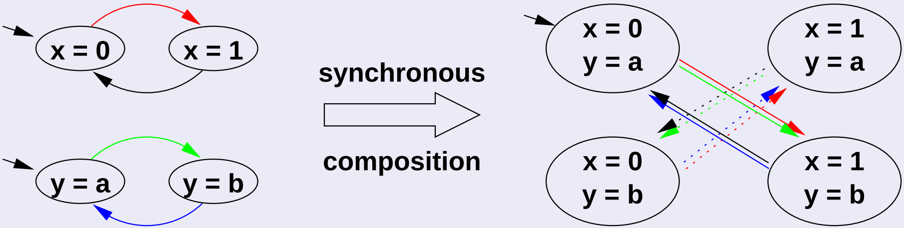


Asynchronous Composition: Example 2



Synchronous Composition

- Components evolve in parallel.
- At each time instant, every component performs a transition.



- Typical example: sequential hardware circuits.

Synchronous Composition/Product: formal definition

Synchronous product of Kripke models

Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$. Then the synchronous product $M \stackrel{\text{def}}{=} M_1 \times M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L \rangle$, where

- $S \subseteq S_1 \times S_2$ s.t.,
  $\forall \langle s_1, s_2 \rangle \in S, \forall l \in AP_1 \cap AP_2, l \in L_1(s_1)$ *iff* $l \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ s.t. $I \subseteq S$
- $R(\langle s_1, s_2 \rangle, \langle t_1, t_2 \rangle)$ *iff* $(R_1(s_1, t_1)$ **and** $R_2(s_2, t_2))$
- $AP = AP_1 \cup AP_2$
- $L : S \longmapsto 2^{AP}$ s.t. $L(\langle s_1, s_2 \rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

Note: combined states must agree on the values of Boolean variables.

Synchronous composition is associative:
$(...(M_1 \times M_2) \times ...) \times M_n) = (M1 \times (M_2 \times (... \times M_n)...) = M_1 \times M_2 \times ... \times M_n$

Synchronous Composition/Product: formal definition

Synchronous product of Kripke models

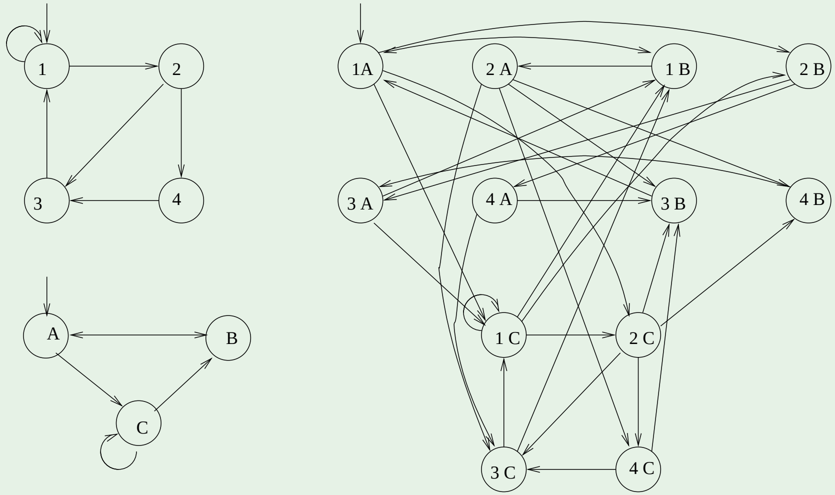
Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1\rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2\rangle$. Then the synchronous product $M \stackrel{\text{def}}{=} M_1 \times M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L\rangle$, where

- $S \subseteq S_1 \times S_2$ s.t., $\forall \langle s_1, s_2\rangle \in S,\ \forall l \in AP_1 \cap AP_2, l \in L_1(s_1)$ *iff* $l \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ s.t. $I \subseteq S$
- $R(\langle s_1, s_2\rangle, \langle t_1, t_2\rangle)$ *iff* $(R_1(s_1, t_1)$ **and** $R_2(s_2, t_2))$
- $AP = AP_1 \cup AP_2$
- $L : S \longmapsto 2^{AP}$ s.t. $L(\langle s_1, s_2\rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

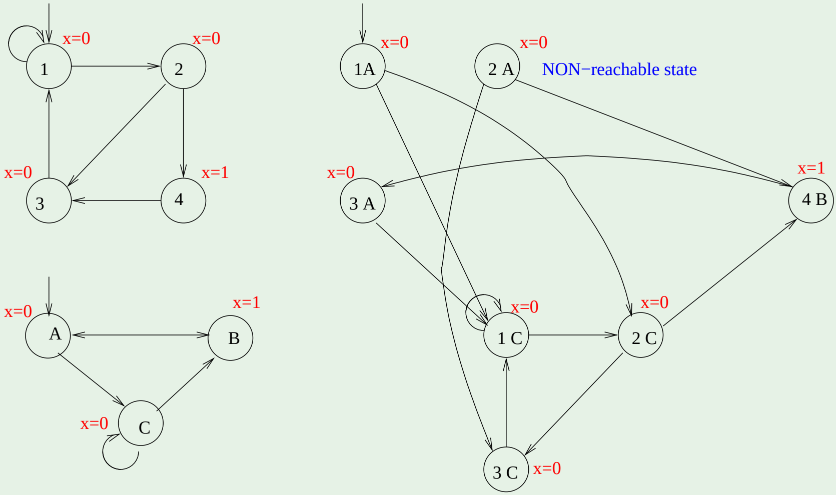
Note: combined states must agree on the values of Boolean variables.

Synchronous composition is associative:
$(...(M_1 \times M_2) \times ...) \times M_n) = (M1 \times (M_2 \times (... \times M_n)...) = M_1 \times M_2 \times ... \times M_n$

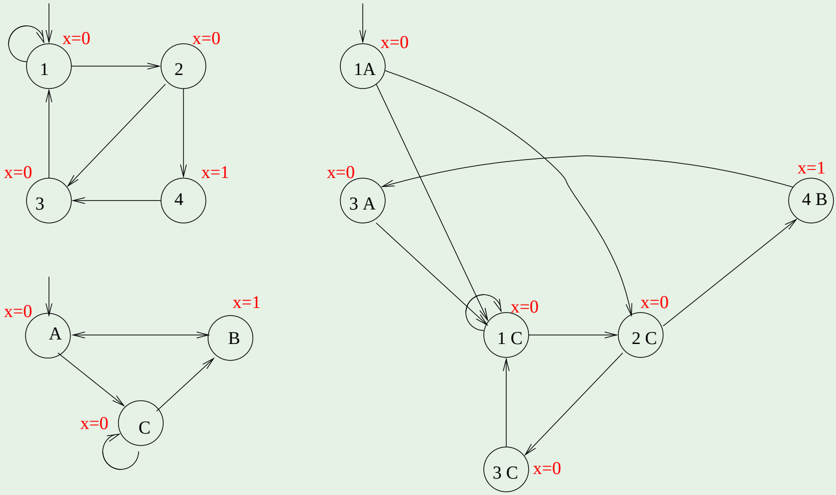
Synchronous Composition: Example 1



Synchronous Composition: Example 2



Synchronous Composition: Example 2 (cont.)



Outline

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

Description languages for Kripke Model

- Most often a Kripke model is not given explicitly (states, arcs),...
- ... rather it is usually presented in a structured language (e.g., SMV, PROMELA, StateCharts, VHDL, ...)
  - even a piece of SW can be seen as a Kripke model!
- Each component is presented by specifying
  - state variables: determine the set of atomic propositions AP, the state space S and the labeling L.
  - initial values of variables V: determine the set of initial states I.
    - described as a relation $I(V_0)$ in terms of state variables at step 0
  - instructions: determine the transition relation R.
    - described as a relation $R(V, V')$ in terms of current state variables V and next state variables V'
- Aka as symbolic representation of a Kripke model

Remark

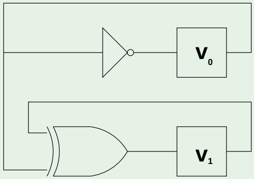
Tipically symbolic description are much more compact (and intuitive) than the explicit representation of the Kripke model.

The SMV language

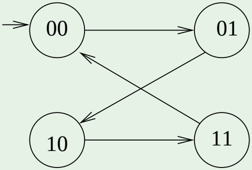
- The input language of the SMV M.C. (and NuSMV)
- Booleans, enumerative and bounded integers as data types
- now enriched with other constructs, e.g. in NuXMV language
- An SMV program consists of:
  - Declarations of the state variables (e.g., `b0`);
  - Assignments that define the initial states (e.g., `init(b0) := 0`).
  - Assignments that define the transition relation (e.g., `next(b0) := !b0`).
- Allows for both synchronous and asyncronous composition of modules (though synchronous interaction more natural)

Example: a Simple Counter Circuit

```
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);
```

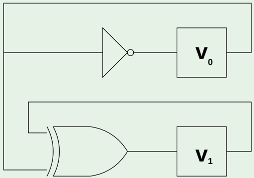


v_1	v_0	v_1'	v_0'
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

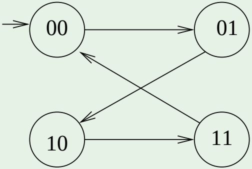


Example: a Simple Counter Circuit

```
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



$$
\begin{aligned}
I(V) &= (\neg v_0 \land \neg v_1) \\
R(V, V') &= (v_0' \leftrightarrow \neg v_0) \land (v_1' \leftrightarrow v_0 \bigoplus v_1)
\end{aligned}
$$

Standard Programming Languages

- Standard programming languages are typically sequential

$\Longrightarrow$ Transition relation defined in terms also of the program counter

- Numbers & values Booleanized

```
...
10. i = 0;
11. acc = 0.0;
12. while (i<dim) {
13.   acc += V[i];
14.   i++;
15. }
...
```

....

$(pc = 10) \rightarrow ((i' = 0) \land (pc' = 11))$
$(pc = 11) \rightarrow ((acc' = 0.0) \land (pc' = 12))$
$(pc = 12) \rightarrow ((i < dim) \rightarrow (pc' = 13))$
$(pc = 12) \rightarrow (\neg(i < dim) \rightarrow (pc' = 16))$
$(pc = 13) \rightarrow ((acc' = acc + read(V, i)) \land (pc' = 14))$
$(pc = 14) \rightarrow (i' = i + 1) \land (pc' = 15))$
$(pc = 15) \rightarrow (pc' = 16))$

...

Standard Programming Languages

- Standard programming languages are typically sequential

$\Longrightarrow$ Transition relation defined in terms also of the program counter

- Numbers & values Booleanized

```
...
10. i = 0;
11. acc = 0.0;
12. while (i<dim) {
13.   acc += V[i];
14.   i++;
15. }
...
```

....

$(pc = 10) \rightarrow ((i' = 0) \wedge (pc' = 11))$
$(pc = 11) \rightarrow ((acc' = 0.0) \wedge (pc' = 12))$
$(pc = 12) \rightarrow ((i < dim) \rightarrow (pc' = 13))$
$(pc = 12) \rightarrow (\neg(i < dim) \rightarrow (pc' = 16))$
$(pc = 13) \rightarrow ((acc' = acc + read(V, i)) \wedge (pc' = 14))$
$(pc = 14) \rightarrow (i' = i + 1) \wedge (pc' = 15))$
$(pc = 15) \rightarrow (pc' = 16))$

...

Standard Programming Languages

- Standard programming languages are typically sequential

$\Longrightarrow$ Transition relation defined in terms also of the program counter

- Numbers & values Booleanized

```
...
10. i = 0;
11. acc = 0.0;
12. while (i<dim) {
13.   acc += V[i];
14.   i++;
15. }
...
```

....

$(pc = 10) \rightarrow ((i' = 0) \land (pc' = 11))$

$(pc = 11) \rightarrow ((acc' = 0.0) \land (pc' = 12))$

$(pc = 12) \rightarrow ((i < dim) \rightarrow (pc' = 13))$

$(pc = 12) \rightarrow (\neg(i < dim) \rightarrow (pc' = 16))$

$(pc = 13) \rightarrow ((acc' = acc + read(V, i)) \land (pc' = 14))$

$(pc = 14) \rightarrow (i' = i + 1) \land (pc' = 15))$

$(pc = 15) \rightarrow (pc' = 16))$

...

Standard Programming Languages

- Standard programming languages are typically sequential

$\Longrightarrow$ Transition relation defined in terms also of the program counter

- Numbers & values Booleanized

```
...
10. i = 0;
11. acc = 0.0;
12. while (i<dim) {
13.   acc += V[i];
14.   i++;
15. }
...
```

....

$(pc = 10) \rightarrow ((i' = 0) \land (pc' = 11))$

$(pc = 11) \rightarrow ((acc' = 0.0) \land (pc' = 12))$

$(pc = 12) \rightarrow ((i < dim) \rightarrow (pc' = 13))$

$(pc = 12) \rightarrow (\neg(i < dim) \rightarrow (pc' = 16))$

$(pc = 13) \rightarrow ((acc' = acc + read(V, i)) \land (pc' = 14))$

$(pc = 14) \rightarrow (i' = i + 1) \land (pc' = 15))$

$(pc = 15) \rightarrow (pc' = 16))$

...

Outline

Safety Properties

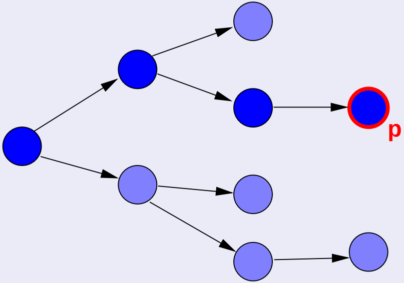
- Bad events never happen
  - deadlock: two processes waiting for input from each other, the system is unable to perform a transition.
  - no reachable state satisfies a “bad” condition, e.g. never two processes in critical section at the same time
- can be refuted by a finite behaviour
- Ex.: it is never the case that p.

Safety Properties

- Bad events never happen
  - deadlock: two processes waiting for input from each other, the system is unable to perform a transition.
  - no reachable state satisfies a “bad” condition, e.g. never two processes in critical section at the same time
- can be refuted by a finite behaviour
- Ex.: it is never the case that p.

Safety Properties

- Bad events never happen
  - deadlock: two processes waiting for input from each other, the system is unable to perform a transition.
  - no reachable state satisfies a “bad” condition, e.g. never two processes in critical section at the same time
- can be refuted by a finite behaviour
- Ex.: it is never the case that *p*.

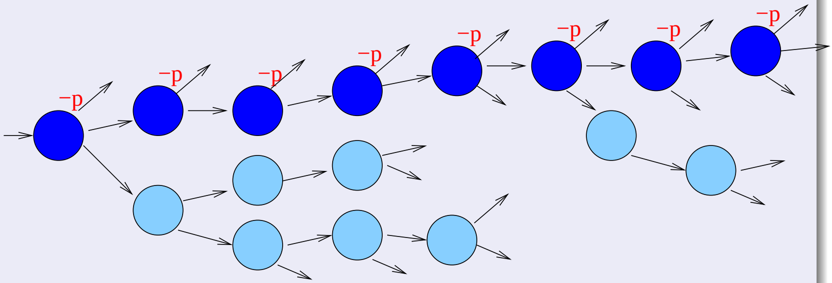


Liveness Properties

- Something desirable will eventually happen
  - sooner or later this will happen
- can be refuted by infinite behaviour
  - an infinite behaviour can be typically presented as a loop

Liveness Properties

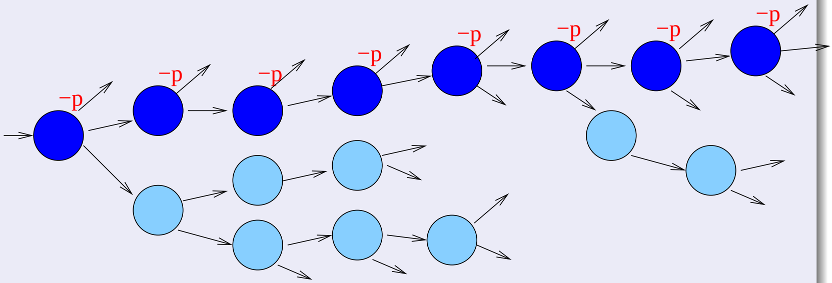
- Something desirable will eventually happen
  - sooner or later this will happen
- can be refuted by infinite behaviour



  - an infinite behaviour can be typically presented as a loop

Liveness Properties

- Something desirable will eventually happen
  - sooner or later this will happen
- can be refuted by infinite behaviour



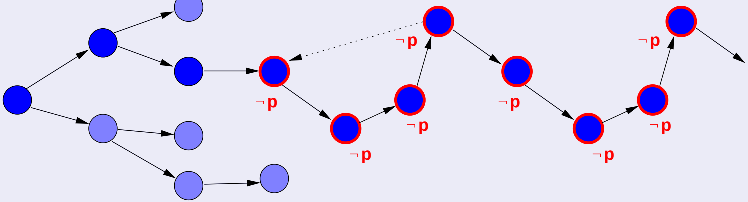
  - an infinite behaviour can be typically presented as a loop

Fairness Properties

- Something desirable will happen infinitely often
  - important subcase of liveness
  - whenever a subroutine takes control, it will always return it (sooner or later)
- can be refuted by infinite behaviour
  - a subroutine takes control and never returns it
  - an infinite behaviour can be typically presented as a loop

Fairness Properties

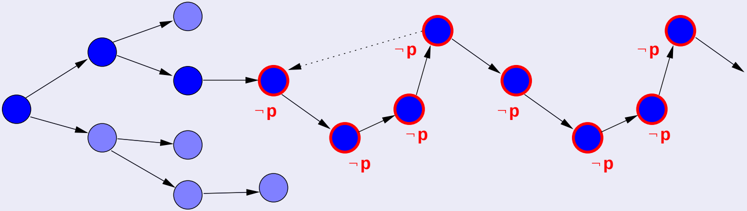
- Something desirable will happen infinitely often
  - important subcase of liveness
  - whenever a subroutine takes control, it will always return it (sooner or later)
- can be refuted by infinite behaviour
  - a subroutine takes control and never returns it



  - an infinite behaviour can be typically presented as a loop

Fairness Properties

- Something desirable will happen infinitely often
  - important subcase of liveness
  - whenever a subroutine takes control, it will always return it (sooner or later)
- can be refuted by infinite behaviour
  - a subroutine takes control and never returns it



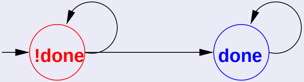
  - an infinite behaviour can be typically presented as a loop

Outline

Outline

Computation tree vs. computation paths

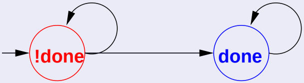
- Consider the following Kripke structure:



- Its execution can be seen as:

Computation tree vs. computation paths

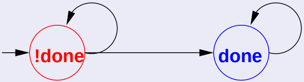
- Consider the following Kripke structure:



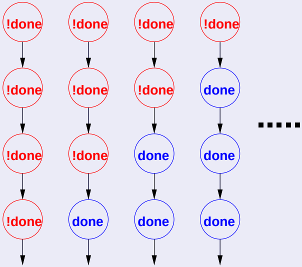
- Its execution can be seen as:

Computation tree vs. computation paths

- Consider the following Kripke structure:

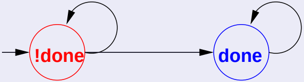


- Its execution can be seen as:
  - an infinite set of computation paths
  - an infinite computation tree

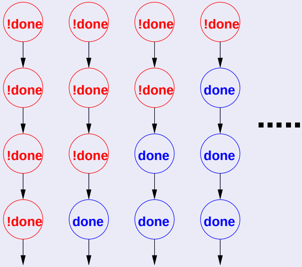


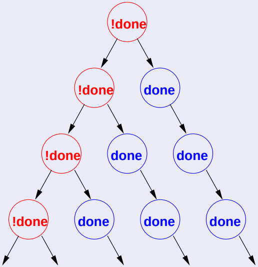
Computation tree vs. computation paths

- Consider the following Kripke structure:



- Its execution can be seen as:
  - an infinite set of computation paths
  - an infinite computation tree





Temporal Logics

- Express properties of “Reactive Systems”
  - nonterminating behaviours,
  - without explicit reference to time.
- Linear Temporal Logic (LTL)
  - interpreted over each path of the Kripke structure
  - linear model of time
  - temporal operators
  - “Medieval”: “since birth, one’s destiny is set”.
- Computation Tree Logic (CTL)
  - interpreted over computation tree of Kripke model
  - branching model of time
  - temporal operators plus path quantifiers
  - “Humanistic”: “one makes his/her own destiny step-by-step”.

Temporal Logics

- Express properties of “Reactive Systems”
  - nonterminating behaviours,
  - without explicit reference to time.
- Linear Temporal Logic (LTL)
  - interpreted over each path of the Kripke structure
  - linear model of time
  - temporal operators
  - “Medieval”: “since birth, one’s destiny is set”.
- Computation Tree Logic (CTL)
  - interpreted over computation tree of Kripke model
  - branching model of time
  - temporal operators plus path quantifiers
  - “Humanistic”: “one makes his/her own destiny step-by-step”.

Temporal Logics

- Express properties of “Reactive Systems”
  - nonterminating behaviours,
  - without explicit reference to time.
- Linear Temporal Logic (LTL)
  - interpreted over each path of the Kripke structure
  - linear model of time
  - temporal operators
  - “Medieval”: “since birth, one’s destiny is set”.
- Computation Tree Logic (CTL)
  - interpreted over computation tree of Kripke model
  - branching model of time
  - temporal operators plus path quantifiers
  - “Humanistic”: “one makes his/her own destiny step-by-step”.

Outline

Linear Temporal Logic (LTL): Syntax

- An atomic proposition is a LTL formula;
- if φ_1 and φ_2 are LTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, $\varphi_1 \oplus \varphi_2$ are LTL formulae;
- if φ_1 and φ_2 are LTL formulae, then $\mathbf{X}\varphi_1$, $\varphi_1\mathbf{U}\varphi_2$, $\mathbf{G}\varphi_1$, $\mathbf{F}\varphi_1$ are LTL formulae, where **X**, **G**, **F**, **U** are the "next", "globally", "eventually","until" temporal operators respectively.
- Another operator **R** "releases" (the dual of **U**) is used sometimes.

Linear Temporal Logic (LTL): Syntax

- An atomic proposition is a LTL formula;
- if φ_1 and φ_2 are LTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, $\varphi_1 \oplus \varphi_2$ are LTL formulae;
- if φ_1 and φ_2 are LTL formulae, then $\mathbf{X}\varphi_1$, $\varphi_1\mathbf{U}\varphi_2$, $\mathbf{G}\varphi_1$, $\mathbf{F}\varphi_1$ are LTL formulae, where **X**, **G**, **F**, **U** are the “next”, “globally”, “eventually”,“until” temporal operators respectively.
- Another operator **R** “releases” (the dual of **U**) is used sometimes.

Linear Temporal Logic (LTL): Syntax

- An atomic proposition is a LTL formula;
- if φ_1 and φ_2 are LTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, $\varphi_1 \oplus \varphi_2$ are LTL formulae;
- if φ_1 and φ_2 are LTL formulae, then $\mathbf{X}\varphi_1$, $\varphi_1\mathbf{U}\varphi_2$, $\mathbf{G}\varphi_1$, $\mathbf{F}\varphi_1$ are LTL formulae, where **X**, **G**, **F**, **U** are the "next", "globally", "eventually","until" temporal operators respectively.
- Another operator **R** "releases" (the dual of **U**) is used sometimes.

Linear Temporal Logic (LTL): Syntax

- An atomic proposition is a LTL formula;
- if φ_1 and φ_2 are LTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$, $\varphi_1 \oplus \varphi_2$ are LTL formulae;
- if φ_1 and φ_2 are LTL formulae, then $\mathbf{X}\varphi_1$, $\varphi_1\mathbf{U}\varphi_2$, $\mathbf{G}\varphi_1$, $\mathbf{F}\varphi_1$ are LTL formulae, where **X**, **G**, **F**, **U** are the "next", "globally", "eventually","until" temporal operators respectively.
- Another operator **R** "releases" (the dual of **U**) is used sometimes.

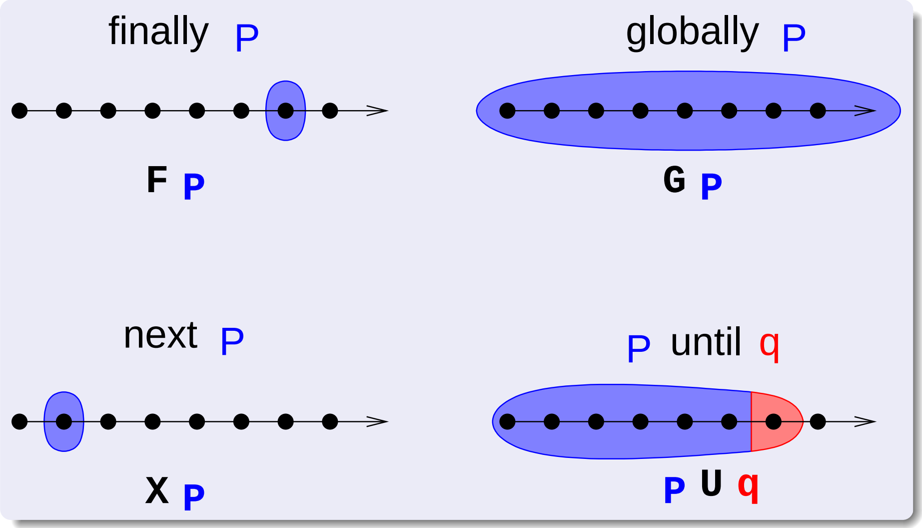
LTL semantics: intuitions

LTL is given by the standard boolean logic enhanced with the following temporal operators, which operate through paths $\langle s_0, s_1, ..., s_k, ...\rangle$:

- "Next" **X**: $\mathbf{X}\varphi$ is true in s_t iff φ is true in s_{t+1}
- "Finally" (or "eventually") **F**: $\mathbf{F}\varphi$ is true in s_t iff φ is true in **some** $s_{t'}$ with $t' \geq t$
- "Globally" (or "henceforth") **G**: $\mathbf{G}\varphi$ is true in s_t iff φ is true in **all** $s_{t'}$ with $t' \geq t$
- "Until" **U**: $\varphi\mathbf{U}\psi$ is true in s_t iff, for some state $s_{t'}$ s.t $t' \geq t$:
  - ψ is true in $s_{t'}$ **and**
  - φ is true in all states $s_{t''}$ s.t. $t \leq t'' < t'$
- "Releases" **R**: $\varphi\mathbf{R}\psi$ is true in s_t iff, for all states $s_{t'}$ s.t. $t' \geq t$:
  - ψ is true **or**
  - φ is true in some states $s_{t''}$ with $t \leq t'' < t'$

  "ψ can become false only if φ becomes true first"

LTL semantics: intuitions



LTL: Some Noteworthy Examples

- Safety: “it never happens that a train is arriving and the bar is up”

$$\mathbf{G}(\neg(\text{train\_arriving} \wedge \text{bar\_up}))$$

- Liveness: “if input, then eventually output”

$$\mathbf{G}(\text{input} \rightarrow \mathbf{F}\text{output})$$

- Releases: “the device is not working if you don’t first repair it”

$$(\text{repair\_device}\ \mathbf{R}\ \neg\text{working\_device})$$

- Fairness: “infinitely often send ”

$$\mathbf{GF}\text{send}$$

- Strong fairness: “infinitely often send implies infinitely often recv.”

$$\mathbf{GF}\text{send} \rightarrow \mathbf{GF}\text{recv}$$

LTL Formal Semantics

$$
\begin{array}{llllrll}
\pi, s_i & \models & a & \text{iff} & & a \in L(s_i) & \\
\pi, s_i & \models & \neg\varphi & \text{iff} & \pi, s_i & \not\models & \varphi \\
\pi, s_i & \models & \varphi \wedge \psi & \text{iff} & \pi, s_i & \models & \varphi \text{ and} \\
 & & & & \pi, s_i & \models & \psi \\
\pi, s_i & \models & \mathbf{X}\varphi & \text{iff} & \pi, s_{i+1} & \models & \varphi \\
\pi, s_i & \models & \mathbf{F}\varphi & \text{iff} & \text{for some } j \geq i \ : \pi, s_j & \models & \varphi \\
\pi, s_i & \models & \mathbf{G}\varphi & \text{iff} & \text{for all } j \geq i \ : \pi, s_j & \models & \varphi \\
\pi, s_i & \models & \varphi\mathbf{U}\psi & \text{iff} & \text{for some } j \geq i \ : (\pi, s_j & \models & \psi \ \text{ and} \\
 & & & & \text{for all } k \text{ s.t. } i \leq k < j \ : \pi, s_k & \models & \varphi) \\
\pi, s_i & \models & \varphi\mathbf{R}\psi & \text{iff} & \text{for all } j \geq i \ : (\pi, s_j & \models & \psi \ \text{ or} \\
 & & & & \text{for some } k \text{ s.t. } i \leq k < j \ : \pi, s_k & \models & \varphi)
\end{array}
$$

LTL Formal Semantics (cont.)

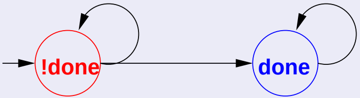
- LTL properties are evaluated over paths, i.e., over infinite, linear sequences of states: $\pi = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_t \rightarrow s_{t+1} \rightarrow \cdots$
- Given an infinite sequence $\pi = s_0, s_1, s_2, \ldots$
  - $\pi, s_i \models \phi$ if ϕ is true in state s_i of π.
  - $\pi \models \phi$ if ϕ is true in the initial state s_0 of π.
- The LTL model checking problem $\mathcal{M} \models \phi$
  - check if $\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$ (e.g., $\phi = \mathbf{F}\mathit{done}$)

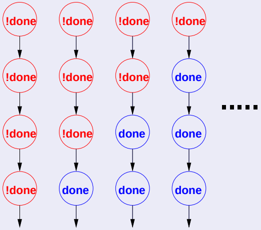
LTL Formal Semantics (cont.)

- LTL properties are evaluated over paths, i.e., over infinite, linear sequences of states: $\pi = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_t \rightarrow s_{t+1} \rightarrow \cdots$
- Given an infinite sequence $\pi = s_0, s_1, s_2, \ldots$
  - $\pi, s_i \models \phi$ if ϕ is true in state s_i of π.
  - $\pi \models \phi$ if ϕ is true in the initial state s_0 of π.
- The LTL model checking problem $\mathcal{M} \models \phi$
  - check if $\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$ (e.g., $\phi = \mathbf{F} done$)

LTL Formal Semantics (cont.)

- LTL properties are evaluated over paths, i.e., over infinite, linear sequences of states: $\pi = s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_t \rightarrow s_{t+1} \rightarrow \cdots$
- Given an infinite sequence $\pi = s_0, s_1, s_2, \ldots$
  - $\pi, s_i \models \phi$ if ϕ is true in state s_i of π.
  - $\pi \models \phi$ if ϕ is true in the initial state s_0 of π.
- The LTL model checking problem $\mathcal{M} \models \phi$
  - check if $\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$ (e.g., $\phi = \mathbf{F}\mathit{done}$)





The LTL model checking problem $\mathcal{M} \models \phi$

$\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$

Important Remark

$\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$ (!!)

- E.g. if ϕ is a LTL formula and two paths π_1 and π_2 are s.t. $\pi_1 \models \phi$ and $\pi_2 \models \neg\phi$.

The LTL model checking problem $\mathcal{M} \models \phi$: remark

The LTL model checking problem $\mathcal{M} \models \phi$

$\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$

Important Remark

$\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$ (!!)

- E.g. if ϕ is a LTL formula and two paths π_1 and π_2 are s.t. $\pi_1 \models \phi$ and $\pi_2 \models \neg\phi$.

The LTL model checking problem $\mathcal{M} \models \phi$: remark

The LTL model checking problem $\mathcal{M} \models \phi$

$\pi \models \phi$ for every path π of the Kripke structure $\mathcal{M}$

Important Remark

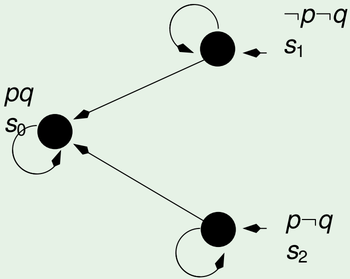
$\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$ (!!)

- E.g. if ϕ is a LTL formula and two paths π_1 and π_2 are s.t. $\pi_1 \models \phi$ and $\pi_2 \models \neg\phi$.

Example: $\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$

Let $\pi_1 \stackrel{\text{def}}{=} \{s_1\}^\omega$, $\pi_2 \stackrel{\text{def}}{=} \{s_2\}^\omega$.

- $\mathcal{M} \not\models \mathbf{G}p$, in fact:
  - $\pi_1 \not\models \mathbf{G}p$
  - $\pi_2 \models \mathbf{G}p$
- $\mathcal{M} \not\models \neg\mathbf{G}p$, in fact:
  - $\pi_1 \models \neg\mathbf{G}p$
  - $\pi_2 \not\models \neg\mathbf{G}p$



Syntactic properties of LTL operators

$$
\begin{aligned}
\varphi_1 \lor \varphi_2 &\iff \neg(\neg\varphi_1 \land \neg\varphi_2) \\
\ldots & \\
\mathbf{F}\ \varphi_1 &\iff \top \mathbf{U} \varphi_1 \\
\mathbf{G}\ \varphi_1 &\iff \bot \mathbf{R} \varphi_1 \\
\mathbf{F}\varphi_1 &\iff \neg\mathbf{G}\neg\varphi_1 \\
\mathbf{G}\varphi_1 &\iff \neg\mathbf{F}\neg\varphi_1 \\
\neg\mathbf{X}\varphi_1 &\iff \mathbf{X}\neg\varphi_1 \\
\varphi_1\mathbf{R}\varphi_2 &\iff \neg(\neg\varphi_1\mathbf{U}\neg\varphi_2) \\
\varphi_1\mathbf{U}\varphi_2 &\iff \neg(\neg\varphi_1\mathbf{R}\neg\varphi_2)
\end{aligned}
$$

Note

LTL can be defined in terms of $\land$, $\neg$, **X**, **U** only

Exercise

Prove that $\varphi_1\mathbf{R}\varphi_2 \iff \mathbf{G}\varphi_2 \lor \varphi_2\mathbf{U}(\varphi_1 \land \varphi_2)$

Syntactic properties of LTL operators

$$
\begin{array}{lcl}
\varphi_1 \vee \varphi_2 & \iff & \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\dots & & \\
\mathbf{F}\ \varphi_1 & \iff & \top\mathbf{U}\varphi_1 \\
\mathbf{G}\ \varphi_1 & \iff & \bot\mathbf{R}\varphi_1 \\
\mathbf{F}\varphi_1 & \iff & \neg\mathbf{G}\neg\varphi_1 \\
\mathbf{G}\varphi_1 & \iff & \neg\mathbf{F}\neg\varphi_1 \\
\neg\mathbf{X}\varphi_1 & \iff & \mathbf{X}\neg\varphi_1 \\
\varphi_1\mathbf{R}\varphi_2 & \iff & \neg(\neg\varphi_1\mathbf{U}\neg\varphi_2) \\
\varphi_1\mathbf{U}\varphi_2 & \iff & \neg(\neg\varphi_1\mathbf{R}\neg\varphi_2)
\end{array}
$$

Note

LTL can be defined in terms of $\wedge$, $\neg$, **X**, **U** only

Exercise

Prove that $\varphi_1\mathbf{R}\varphi_2 \iff \mathbf{G}\varphi_2 \vee \varphi_2\mathbf{U}(\varphi_1 \wedge \varphi_2)$

Syntactic properties of LTL operators

$$\varphi_1 \vee \varphi_2 \iff \neg(\neg\varphi_1 \wedge \neg\varphi_2)$$
$$\ldots$$
$$\mathbf{F}\ \varphi_1 \iff \top \mathbf{U} \varphi_1$$
$$\mathbf{G}\ \varphi_1 \iff \bot \mathbf{R} \varphi_1$$
$$\mathbf{F}\varphi_1 \iff \neg\mathbf{G}\neg\varphi_1$$
$$\mathbf{G}\varphi_1 \iff \neg\mathbf{F}\neg\varphi_1$$
$$\neg\mathbf{X}\varphi_1 \iff \mathbf{X}\neg\varphi_1$$
$$\varphi_1 \mathbf{R} \varphi_2 \iff \neg(\neg\varphi_1 \mathbf{U} \neg\varphi_2)$$
$$\varphi_1 \mathbf{U} \varphi_2 \iff \neg(\neg\varphi_1 \mathbf{R} \neg\varphi_2)$$

Note

LTL can be defined in terms of $\wedge$, $\neg$, **X**, **U** only

Exercise

Prove that $\varphi_1 \mathbf{R} \varphi_2 \iff \mathbf{G}\varphi_2 \vee \varphi_2 \mathbf{U}(\varphi_1 \wedge \varphi_2)$

Proof of $\varphi \mathbf{R} \psi \Leftrightarrow (\mathbf{G}\psi \vee \psi \mathbf{U}(\varphi \wedge \psi))$

[Solution proposed by the student Samuel Valentini, 2016]

(All state indexes below are implicitly assumed to be ≥ 0.)

$\Rightarrow$: Let π be s.t. $\pi, s_0 \models \varphi \mathbf{R} \psi$

- If $\forall j, \pi, s_j \models \psi$, then $\pi, s_0 \models \mathbf{G}\psi$.
- Otherwise, let s_k be the first state s.t. $\pi, s_k \not\models \psi$.
- Since $\pi, s_0 \models \varphi \mathbf{R} \psi$, then $k > 0$ and exists $k' < k$ s.t. $\pi, S_{k'} \models \varphi$
- By construction, $\pi, s_{k'} \models \varphi \wedge \psi$ and, for every $w < k'$, $\pi, s_w \models \psi$, so that $\pi, s_0 \models \psi \mathbf{U}(\varphi \wedge \psi)$.
- Thus, $\pi, s_0 \models \mathbf{G}\psi \vee \psi \mathbf{U}(\varphi \wedge \psi)$

$\Leftarrow$: Let π be s.t. $\pi, s_0 \models \mathbf{G}\psi \vee \psi \mathbf{U}(\varphi \wedge \psi)$

- If $\pi, s_0 \models \mathbf{G}\psi$, then $\forall j, \pi, s_j \models \psi$, so that $\pi, s_0 \models \varphi \mathbf{R} \psi$.
- Otherwise, $\pi, s_0 \models \psi \mathbf{U}(\varphi \wedge \psi)$.
- Let s_k be the first state s.t. $\pi, s_k \not\models \psi$.
- by construction, $\exists k'$ such that $\pi, S_{k'} \models \varphi \wedge \psi$
- by the definition of k, we have that $k' < k$ and $\forall w < k$, $\pi, S_w \models \psi$.
- Thus $\pi, s_0 \models \varphi \mathbf{R} \psi$

Strength of LTL operators

- $\mathbf{G}\varphi \models \varphi \models \mathbf{F}\varphi$
- $\mathbf{G}\varphi \models \mathbf{X}\varphi \models \mathbf{F}\varphi$
- $\mathbf{G}\varphi \models \mathbf{XX...X}\varphi \models \mathbf{F}\varphi$
- $\varphi\mathbf{U}\psi \models \mathbf{F}\psi$
- $\mathbf{G}\psi \models \varphi\mathbf{R}\psi$

LTL tableaux rules

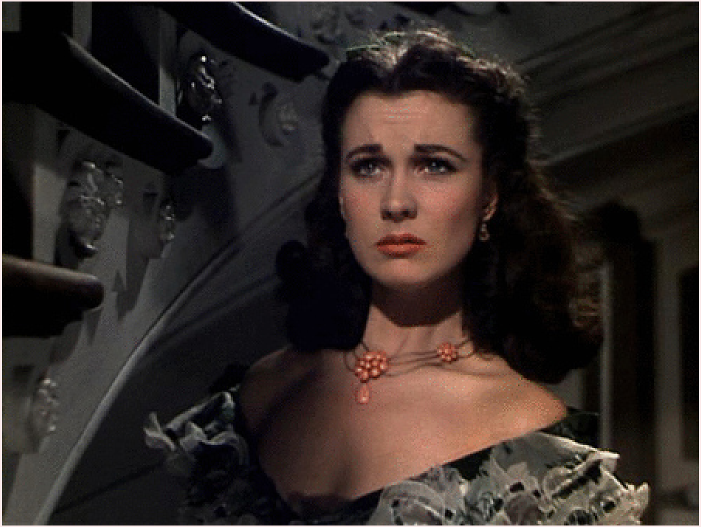
- Let φ_1 and φ_2 be LTL formulae:

$$\begin{aligned}
\mathbf{F}\varphi_1 &\iff (\varphi_1 \vee \mathbf{XF}\varphi_1) \\
\mathbf{G}\varphi_1 &\iff (\varphi_1 \wedge \mathbf{XG}\varphi_1) \\
\varphi_1\mathbf{U}\varphi_2 &\iff (\varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1\mathbf{U}\varphi_2))) \\
\varphi_1\mathbf{R}\varphi_2 &\iff (\varphi_2 \wedge (\varphi_1 \vee \mathbf{X}(\varphi_1\mathbf{R}\varphi_2)))
\end{aligned}$$

- If applied recursively, rewrite an LTL formula in terms of atomic and **X**-formulas:

$$(p\mathbf{U}q) \wedge (\mathbf{G}\neg p) \Longrightarrow (q \vee (p \wedge \mathbf{X}(p\mathbf{U}q))) \wedge (\neg p \wedge \mathbf{XG}\neg p)$$

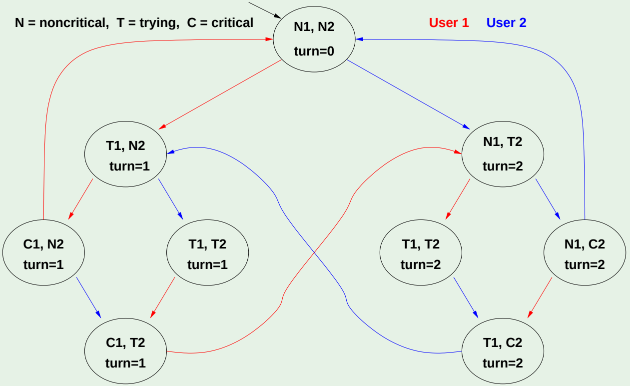
Tableaux Rules: a Quote



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

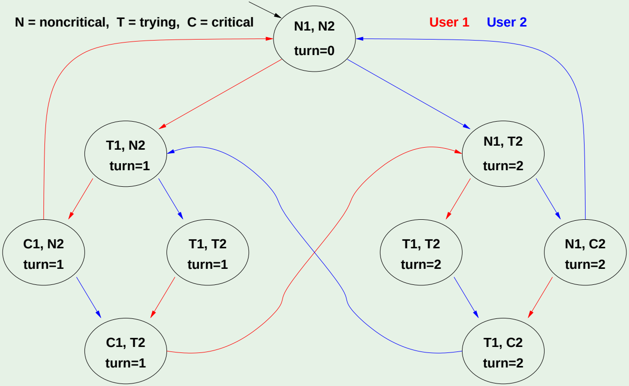
Outline

Example 1: mutual exclusion (safety)



$$M \models \mathbf{G}\neg(C_1 \wedge C_2)\ ?$$

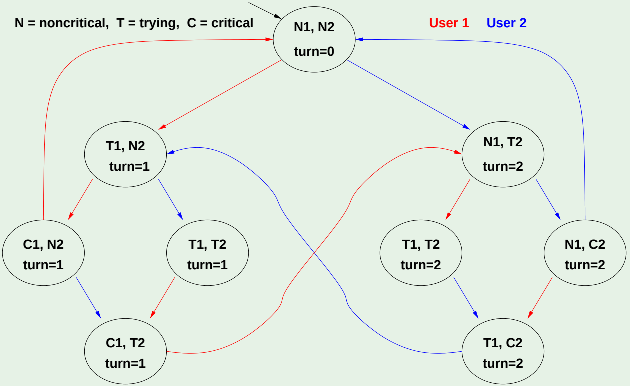
Example 1: mutual exclusion (safety)



$$M \models \mathbf{G}\neg(C_1 \wedge C_2)\ ?$$

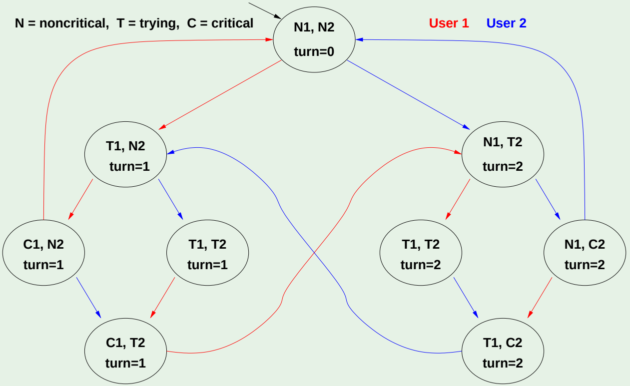
YES: There is no reachable state in which $(C_1 \wedge C_2)$ holds!

Example 2: liveness



$M \models \mathbf{F} C_1$?

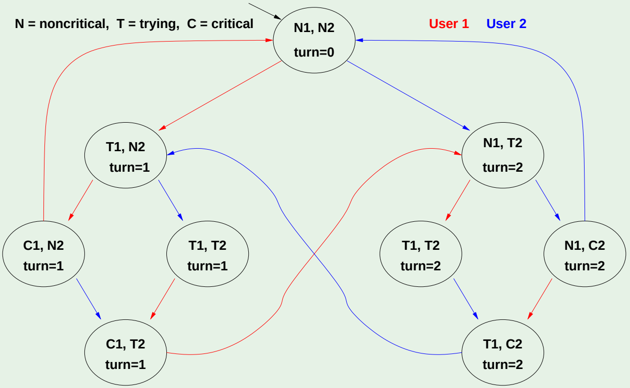
Example 2: liveness



$M \models \mathbf{F} C_1$?

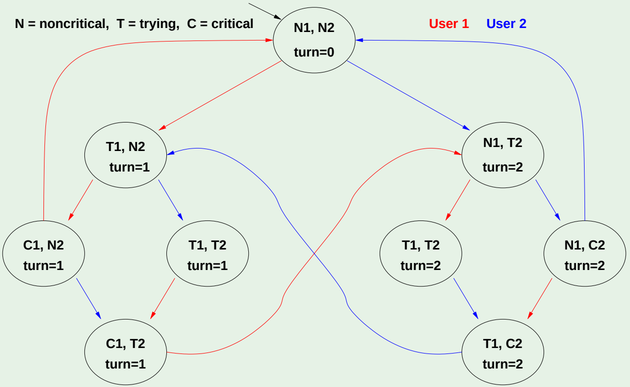
NO: there is an infinite cyclic solution in which C_1 never holds!

Example 3: liveness



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

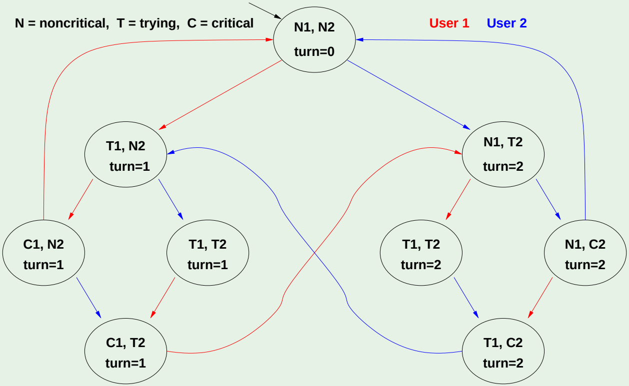
Example 3: liveness



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

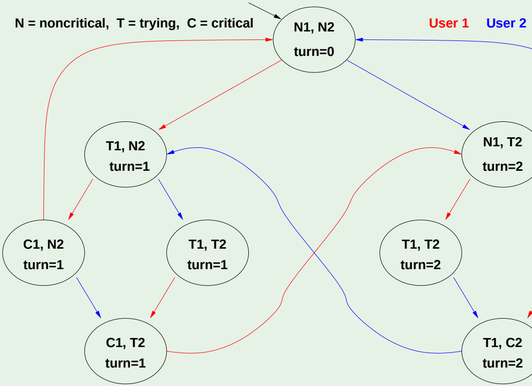
YES: every path starting from each state where T_1 holds passes through a state where C_1 holds.

Example 4: fairness



$$M \models \mathbf{GF} C_1 \ ?$$

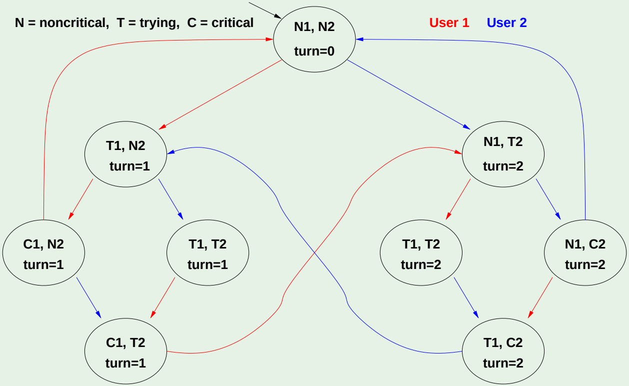
Example 4: fairness



$M \models \mathbf{GF} C_1$?

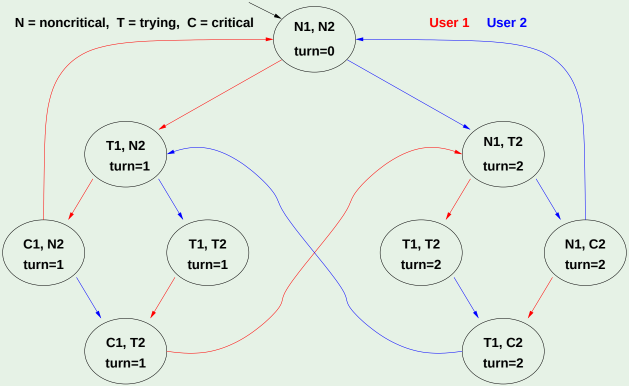
NO: e.g., in the initial state, there is an infinite cyclic solution in which C_1 never holds!

Example 5: strong fairness



$$M \models \mathbf{GF} T_1 \rightarrow \mathbf{GF} C_1 \ ?$$

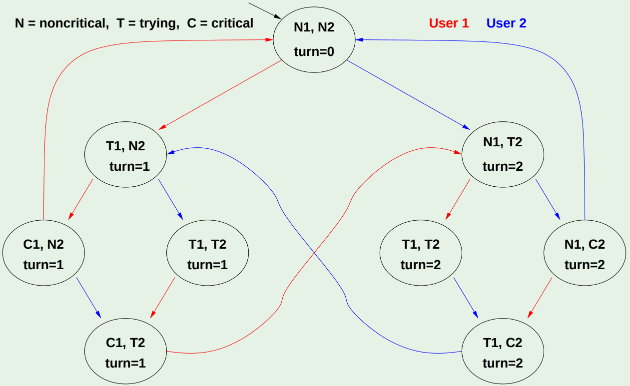
Example 5: strong fairness



$$M \models \mathbf{GF}T_1 \rightarrow \mathbf{GF}C_1 \text{ ?}$$

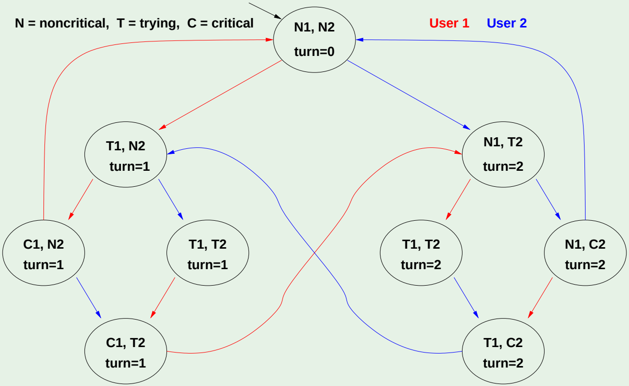
YES: every path which visits T_1 infinitely often also visits C_1 infinitely often (see liveness property of previous example).

Example 6: Releases



$M \models T_1 \mathbf{R} \neg C_1$?

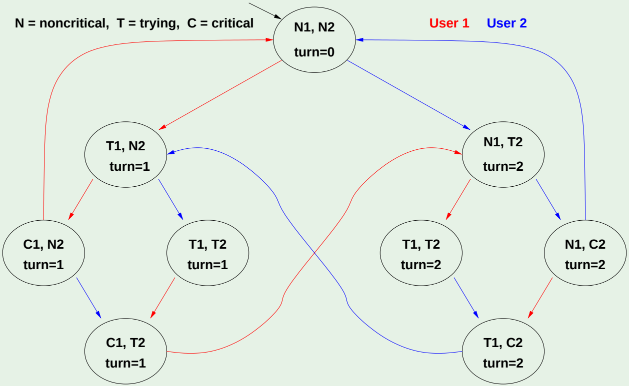
Example 6: Releases



$M \models T_1 \mathbf{R} \neg C_1$?

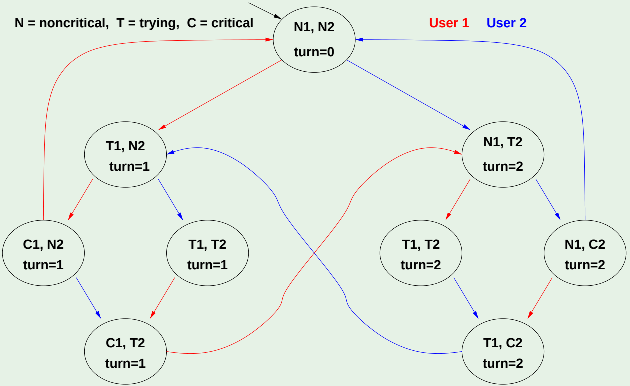
YES: C_1 in paths only strictly after T_1 has occured.

Example 7: **XF**



$$M \models \mathbf{XF}(turn = 0)\ ?$$

Example 7: **XF**



$M \models \mathbf{XF}(turn = 0)$?

NO: a counter-example is the ∞-shaped loop:
$(N1, N2), \{(T1, N2), (C1, N2), (C1, T2), (N1, T2), (N1, C2), (T1, C2)\}^{\omega}$

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ s.t. $j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ s.t. $j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ s.t. $j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.

  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$

  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$

  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$

  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$

  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$

  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$

  $\implies \pi \models \mathbf{GF}C$

  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models \mathbf{F}C$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models \mathbf{F}C$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models \mathbf{F}C$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t.j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t.j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t.j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \implies \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- YES: if $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$, then $M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$!
- let $M \models \mathbf{G}(T \rightarrow \mathbf{F}C)$.
  let $\pi \in M$ s.t. $\pi \models \mathbf{GF}T$
  $\implies \pi, s_i \models \mathbf{F}T$ for each $s_i \in \pi$
  $\implies \pi, s_j \models T$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_j \models FC$ for each $s_i \in \pi$ and for some $s_j \in \pi$ $s.t. j \geq i$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$, for some $s_j \in \pi$ $s.t. j \geq i$ and for some $k \geq j$
  $\implies \pi, s_k \models C$ for each $s_i \in \pi$ and for some $k \geq i$
  $\implies \pi \models \mathbf{GF}C$
  $\implies M \models \mathbf{GF}T \rightarrow \mathbf{GF}C$.

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \Longleftarrow \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- NO!.
- Counter example:
    - $\mathbf{GF}T \rightarrow \mathbf{GF}C$ is satisfied
    - $\mathbf{G}(T \rightarrow \mathbf{F}C)$ is not satisfied

(Counter-example proposed by the student Vaishak Belle, 2008)

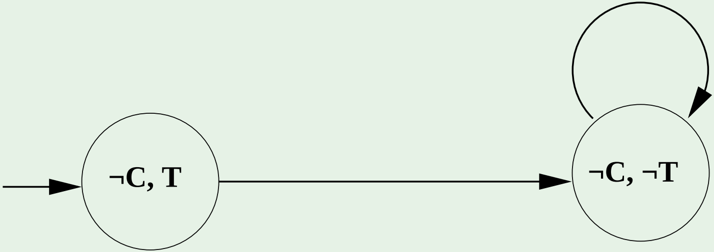
Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \Longleftarrow \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- NO!.
- Counter example:
  - $\mathbf{GF}T \rightarrow \mathbf{GF}C$ is satisfied
  - $\mathbf{G}(T \rightarrow \mathbf{F}C)$ is not satisfied

(Counter-example proposed by the student Vaishak Belle, 2008)

Example: $\mathbf{G}(T \rightarrow \mathbf{F}C)$ vs. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \quad \Longleftarrow \quad \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- NO!.
- Counter example:



  - $\mathbf{GF}T \rightarrow \mathbf{GF}C$ is satisfied
  - $\mathbf{G}(T \rightarrow \mathbf{F}C)$ is not satisfied

(Counter-example proposed by the student Vaishak Belle, 2008)

“You have no respect for logic. (...)
I have no respect for those who have no respect for logic.”

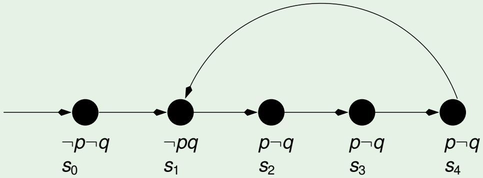
https://www.youtube.com/watch?v=uGstM8QMCjQ

(Arnold Schwarzenegger in “Twins”)

Outline

Exercise: LTL Model Checking (path)

Consider the following path π:



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

(*a*) $\pi, s_0 \models \mathbf{GF}q$

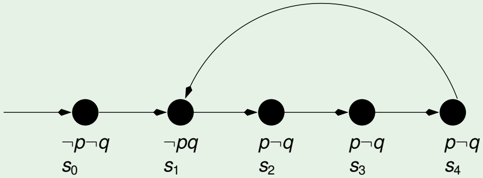
(*b*) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$

(*c*) $\pi, s_2 \models \mathbf{G}p$

(*d*) $\pi, s_2 \models p\mathbf{U}q$

Exercise: LTL Model Checking (path)

Consider the following path π:



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

(*a*) $\pi, s_0 \models \mathbf{GF}q$
[Solution: true]

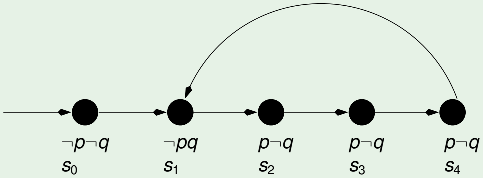
(*b*) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$

(*c*) $\pi, s_2 \models \mathbf{G}p$

(*d*) $\pi, s_2 \models p\mathbf{U}q$

Exercise: LTL Model Checking (path)

Consider the following path π:



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

(*a*) $\pi, s_0 \models \mathbf{GF}q$
[Solution: true]

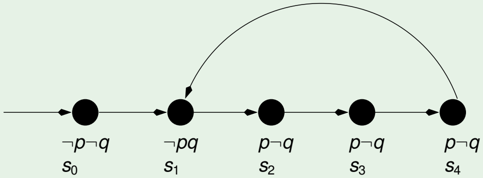
(*b*) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$
[Solution: true]

(*c*) $\pi, s_2 \models \mathbf{G}p$

(*d*) $\pi, s_2 \models p\mathbf{U}q$

Exercise: LTL Model Checking (path)

Consider the following path π:



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

(*a*) $\pi, s_0 \models \mathbf{GF}q$
[Solution: true]

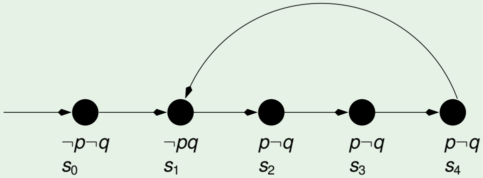
(*b*) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$
[Solution: true]

(*c*) $\pi, s_2 \models \mathbf{G}p$
[Solution: false]

(*d*) $\pi, s_2 \models p\mathbf{U}q$

Exercise: LTL Model Checking (path)

Consider the following path π:



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

(a) $\pi, s_0 \models \mathbf{GF}q$
[Solution: true]

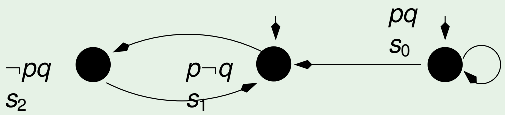
(b) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$
[Solution: true]

(c) $\pi, s_2 \models \mathbf{G}p$
[Solution: false]

(d) $\pi, s_2 \models p\mathbf{U}q$
[Solution: true]

Ex: LTL Model Checking

Consider the following Kripke Model M:



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

(a) $M \models (p\mathbf{U}q)$

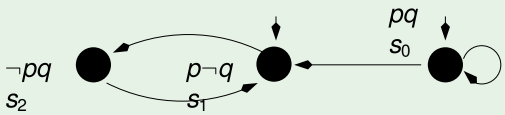
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$

(c) $M \models \mathbf{G}p \rightarrow \mathbf{G}q$

(d) $M \models \mathbf{FG}p$

Ex: LTL Model Checking

Consider the following Kripke Model M:



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

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

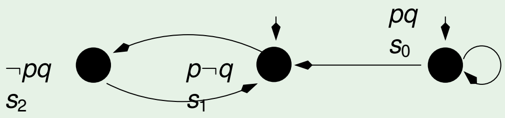
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$

(c) $M \models \mathbf{G}p \rightarrow \mathbf{G}q$

(d) $M \models \mathbf{FG}p$

Ex: LTL Model Checking

Consider the following Kripke Model M:



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

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

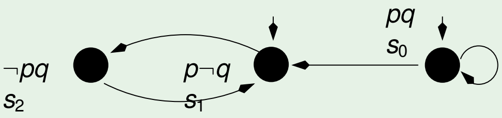
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$
[Solution: true]

(c) $M \models \mathbf{G}p \rightarrow \mathbf{G}q$

(d) $M \models \mathbf{FG}p$

Ex: LTL Model Checking

Consider the following Kripke Model M:



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

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

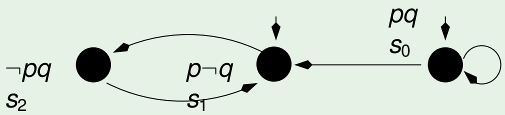
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$
[Solution: true]

(c) $M \models \mathbf{G}p \rightarrow \mathbf{G}q$
[Solution: true]

(d) $M \models \mathbf{FG}p$

Ex: LTL Model Checking

Consider the following Kripke Model M:



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

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

(*b*) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$
[Solution: true]

(*c*) $M \models \mathbf{G}p \rightarrow \mathbf{G}q$
[Solution: true]

(*d*) $M \models \mathbf{FG}p$
[Solution: false]