Formal Methods:
Module I: Automated Reasoning
Ch. 03: **Temporal Logics**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: http://disi.unitn.it/rseba/DIDATTICA/fm2022/
Teaching assistant: Giuseppe Spallitta – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2021-2022

last update: Wednesday 6th April, 2022, 12:58

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, C. Tinelli, and S.Tonetta, who detain its copyright. Some exampes 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. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Outline

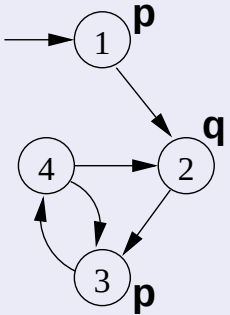
1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. 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 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)

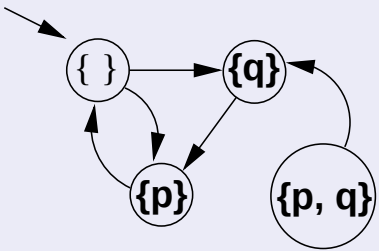


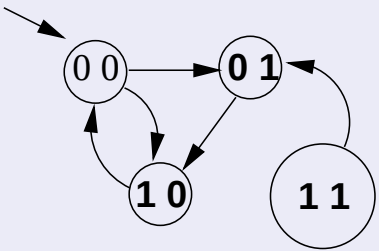
Remark

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

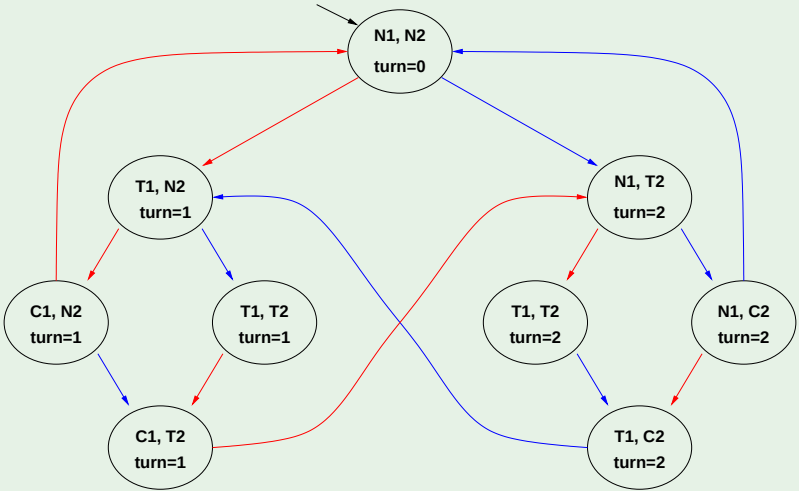
Kripke Structures: Two Alternative Representations:

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





Example: a Kripke model for mutual exclusion

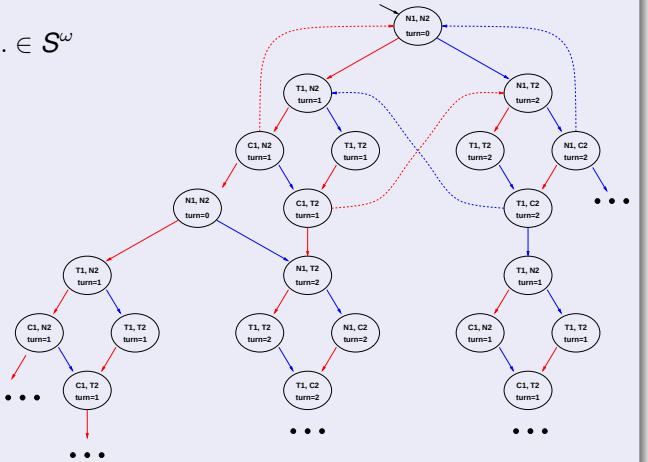


Path in a Kripke Model

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 $(s_i, s_{i+1}) \in R$.



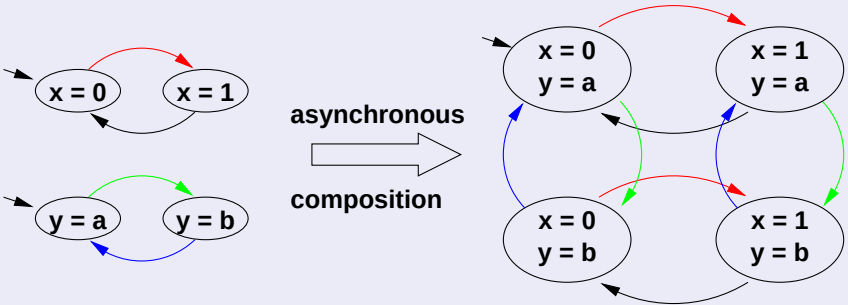
A state s is reachable in M if there is a path from the initial states to s.

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

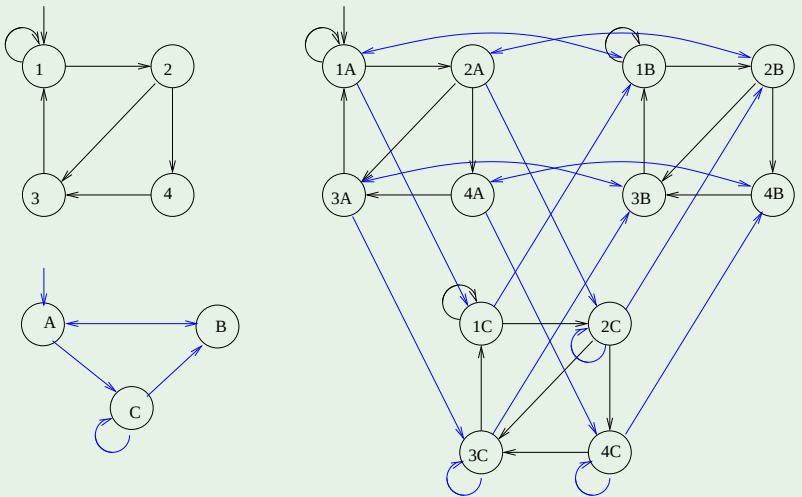
Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$. Then the asynchronous product $M \stackrel{\text{def}}{=} M_1 || M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L \rangle$, where

- $S \subseteq S_1 \times S_2$ s.t., $\forall \langle s_1, s_2 \rangle \in S,\ \forall I \in AP_1 \cap AP_2, I \in L_1(s_1)$ *iff* $I \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ 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. $L(\langle s_1, s_2 \rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

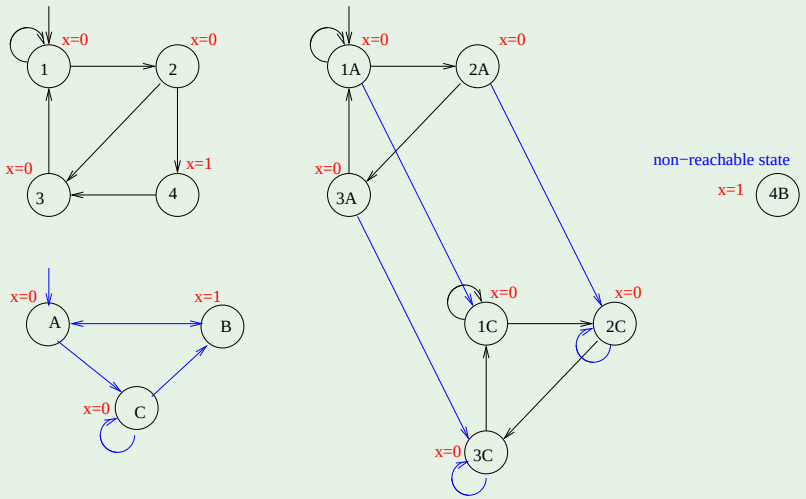
Note: combined states must agree on the values of Boolean variables.

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

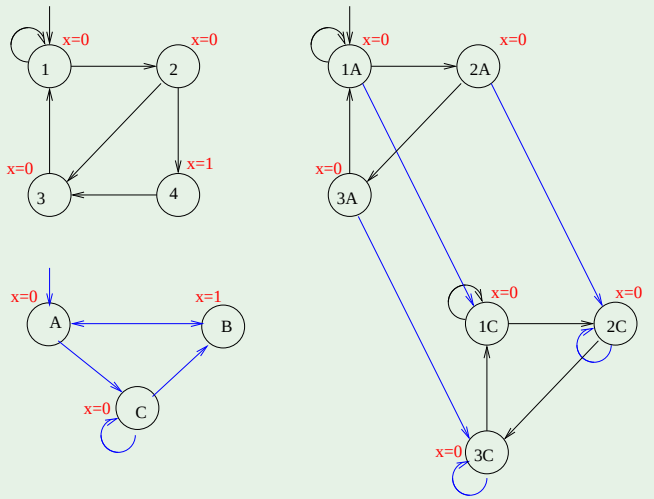
Asynchronous Composition: Example 1



Asynchronous Composition: Example 2

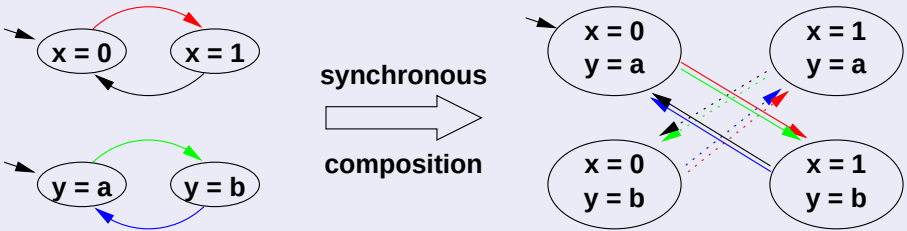


Asynchronous Composition: Example 2



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

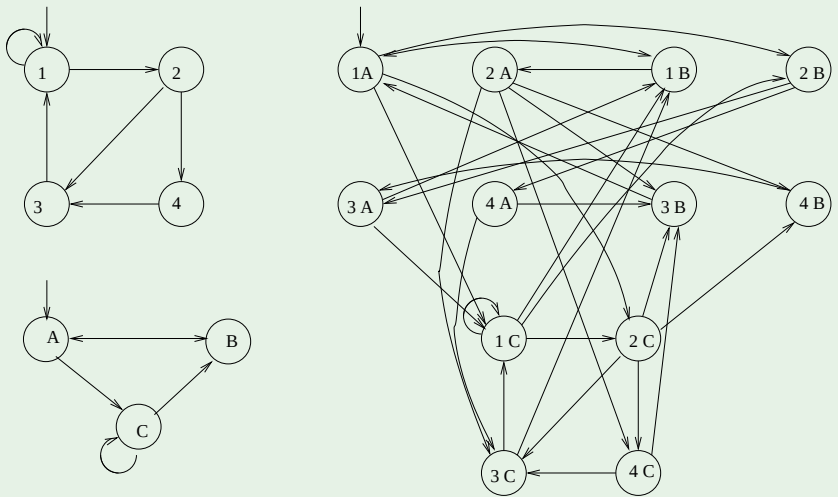
Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$, $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$. Then the synchronous product $M \stackrel{\text{def}}{=} M_1 \times M_2$ is $M \stackrel{\text{def}}{=} \langle S, I, R, AP, L \rangle$, where

- $S \subseteq S_1 \times S_2$ s.t., $\forall \langle s_1, s_2 \rangle \in S,\ \forall I \in AP_1 \cap AP_2, I \in L_1(s_1)$ *iff* $I \in L_2(s_2)$
- $I \subseteq I_1 \times I_2$ 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. $L(\langle s_1, s_2 \rangle) \stackrel{\text{def}}{=} L_1(s_1) \cup L_2(s_2)$.

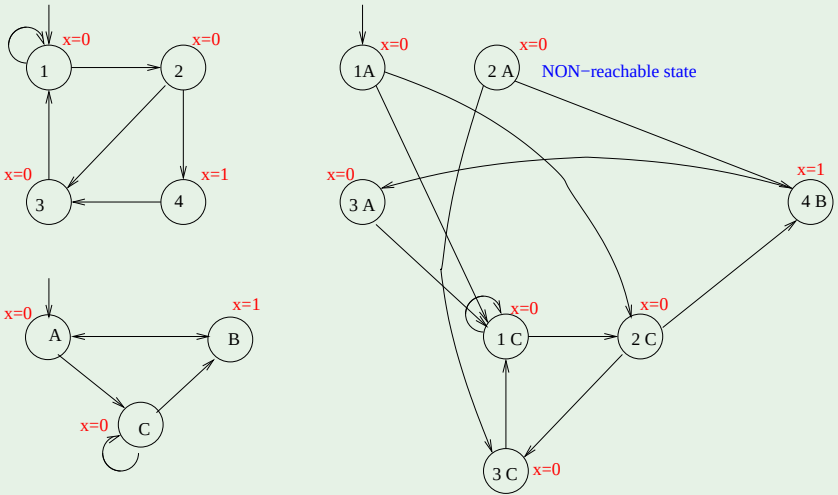
Note: combined states must agree on the values of Boolean variables.

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

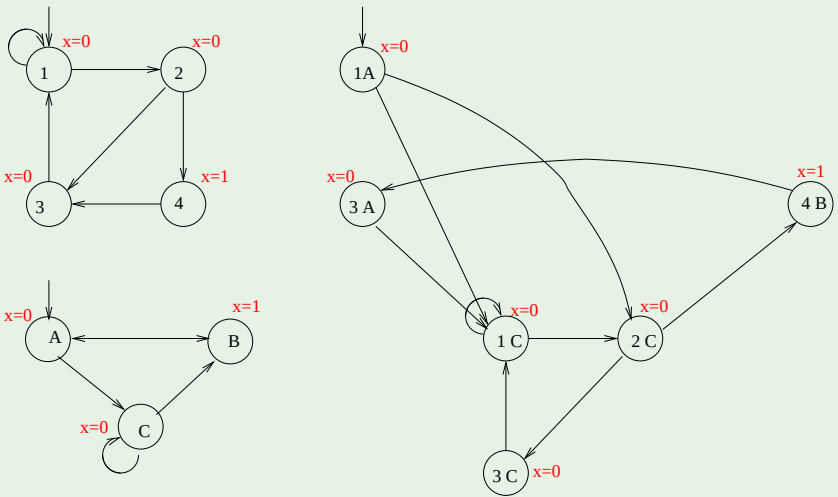
Synchronous Composition: Example 1



Synchronous Composition: Example 2



Synchronous Composition: Example 2 (cont.)



Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

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.

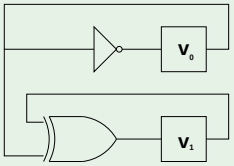
The SMV language

- 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)

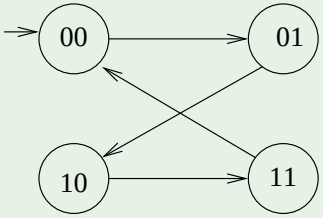
Example: a Simple Counter Circuit

```
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



$$I(V) = (\neg v_0 \wedge \neg v_1)$$
$$R(V, V') = (v_0' \leftrightarrow \neg v_0) \wedge (v_1' \leftrightarrow v_0 \bigoplus v_1)$$

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. }
...
```

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

Outline

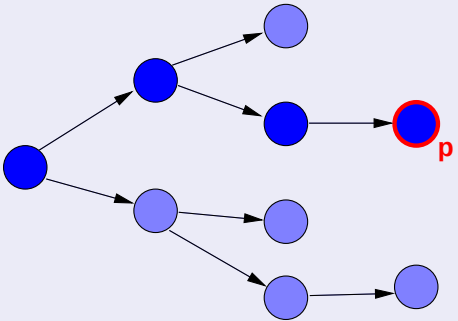
1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

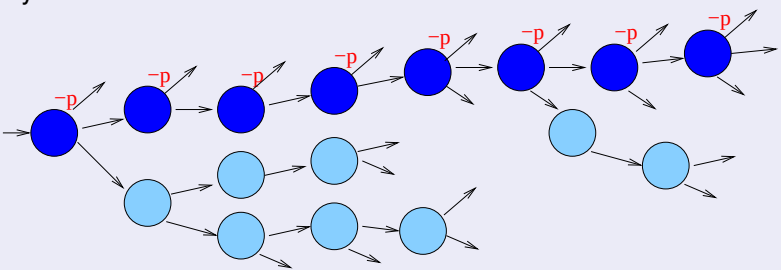
Safety Properties

- 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

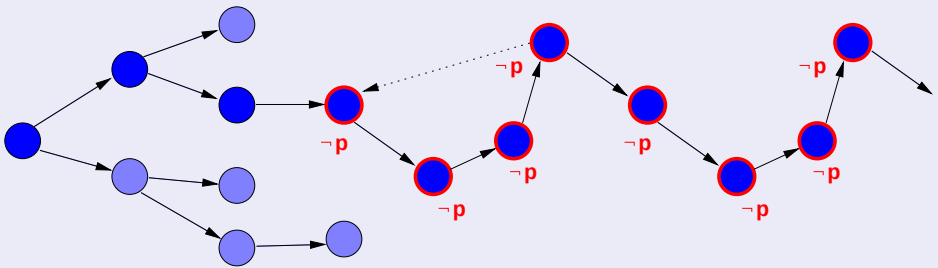
- Something desirable will eventually happen
  - sooner or later this will happen
- Can be refuted by infinite behaviour



  - an infinite behaviour can be typically presented as a loop

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



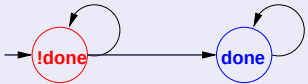
  - an infinite behaviour can be typically presented as a loop

Outline

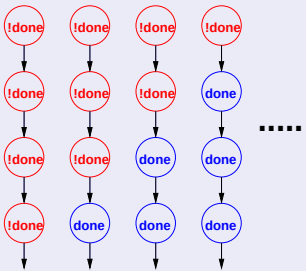
1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Computation tree vs. computation paths

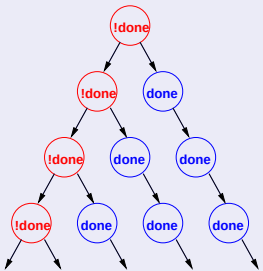
- Consider the following Kripke structure:



- 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”.

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

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 $\mathbf{X}\varphi_1$, $\mathbf{G}\varphi_1$, $\mathbf{F}\varphi_1$, $\varphi_1\mathbf{U}\varphi_2$ 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.

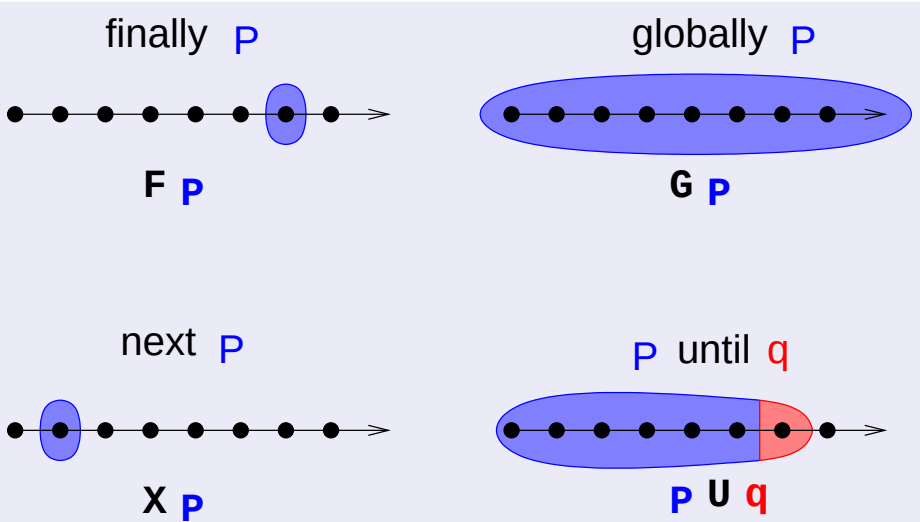
LTL semantics: intuitions

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.”

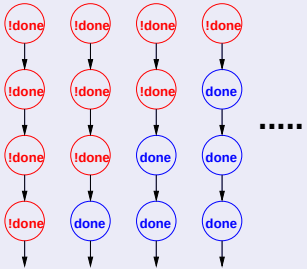
$$\mathbf{GF}\text{send} \rightarrow \mathbf{GF}\text{recv}$$

LTL Formal Semantics

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

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., $\phi = \mathbf{F}done$)



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

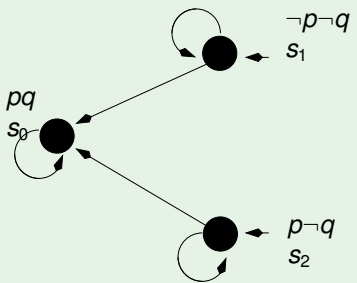
$\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$ (!!)

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

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

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

Note

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

Exercise

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

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

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

LTL tableaux rules

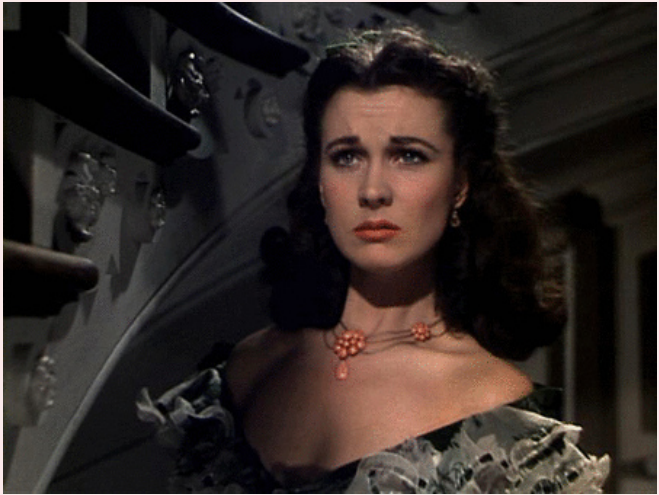
- Let φ_1 and φ_2 be LTL formulae:

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

- 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

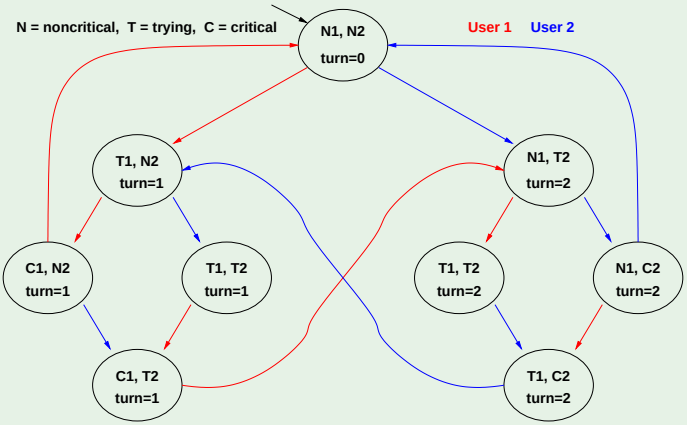


“After all... tomorrow is another day."
[Scarlett O’Hara, “Gone with the Wind”]

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

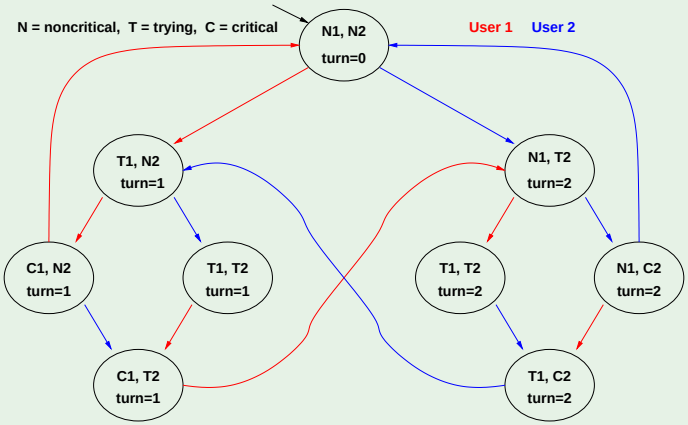
Example 1: mutual exclusion (safety)



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

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

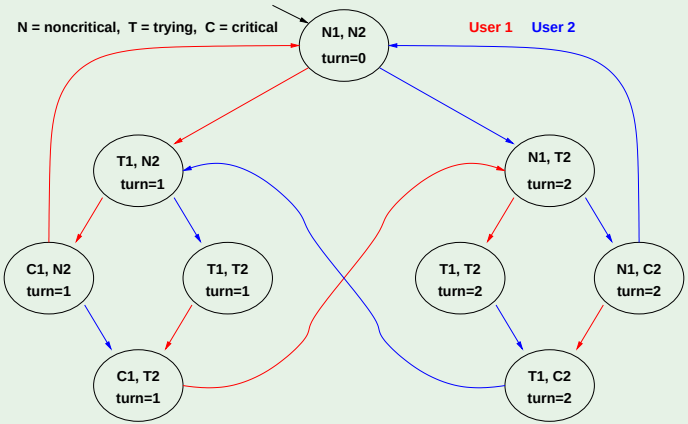
Example 2: liveness



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

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

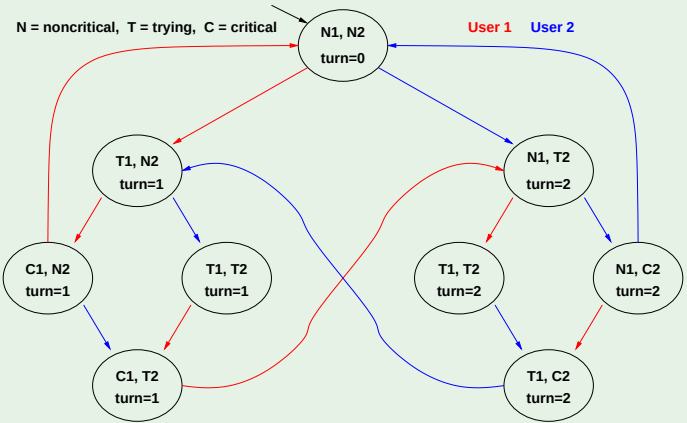
Example 3: liveness



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

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

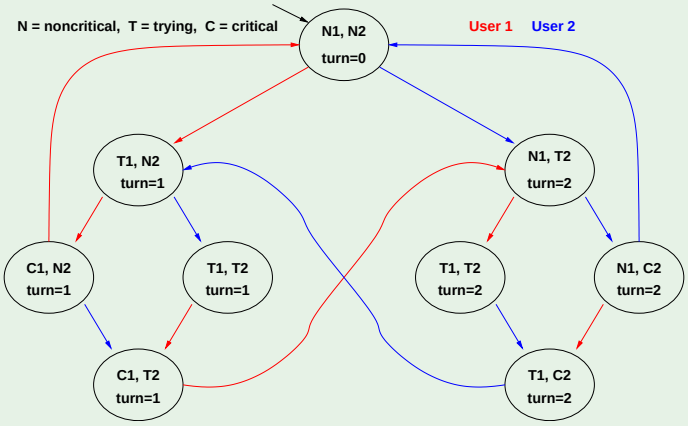
Example 4: fairness



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

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

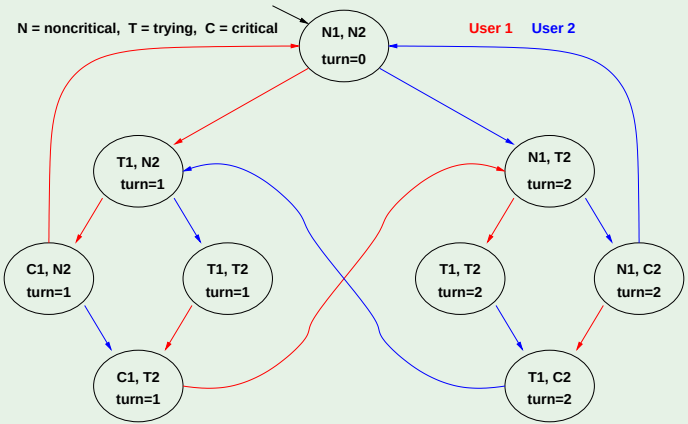
Example 5: strong fairness



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

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

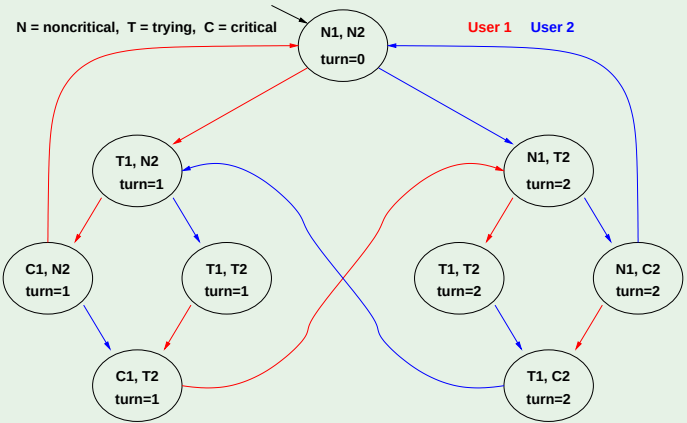
Example 6: blocking



$M \models \mathbf{G}(N_1 \rightarrow \mathbf{F}\, T_1)$?

NO: e.g., in the initial state, there is an infinite cyclic solution in which N_1 holds and T_1 never holds!

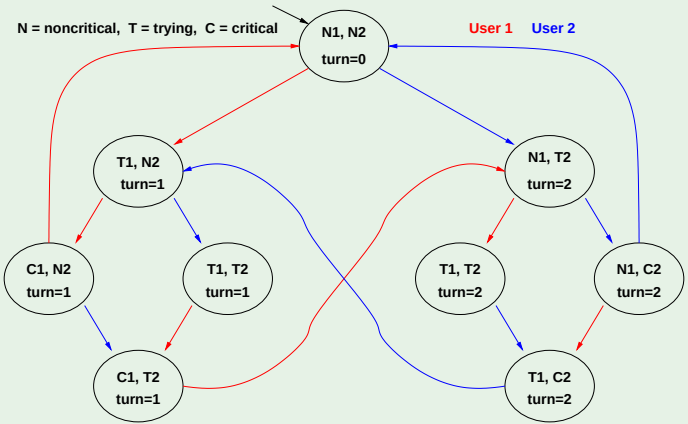
Example 7: Releases



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

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

Example 8: **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. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

- $\mathbf{G}(T \rightarrow \mathbf{F}C) \impliedby \mathbf{GF}T \rightarrow \mathbf{GF}C$?
- 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)

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

Computational Tree Logic (CTL): Syntax

- An atomic proposition is a CTL formula;
- if φ_1 and φ_2 are CTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$ are CTL formulae;
- if φ_1 and φ_2 are CTL formulae, then $\mathbf{AX}\varphi_1$, $\mathbf{A}(\varphi_1 \mathbf{U} \varphi_2)$, $\mathbf{AG}\varphi_1$, $\mathbf{AF}\varphi_1$, $\mathbf{EX}\varphi_1$, $\mathbf{E}(\varphi_1 \mathbf{U} \varphi_2)$, $\mathbf{EG}\varphi_1$, $\mathbf{EF}\varphi_1$,, are CTL formulae.
($\mathbf{E}(\varphi_1 \mathbf{R} \varphi_2)$ and $\mathbf{A}(\varphi_1 \mathbf{R} \varphi_2)$ never used in practice.)

CTL semantics: intuitions

CTL is given by the standard boolean logic enhanced with the operators **AX**, **AG**, **AF**, **AU**, **EX**, **EG**, **EF**, **EU**:

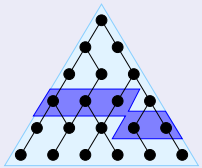
- "Necessarily Next" **AX**: **AX**φ is true in s_t iff φ is true in every successor state s_{t+1}
- "Possibly Next" **EX**: **EX**φ is true in s_t iff φ is true in one successor state s_{t+1}
- "Necessarily in the future" (or "Inevitably") **AF**: **AF**φ is true in s_t iff φ is inevitably true in **some** $s_{t'}$ with $t' \geq t$
- "Possibly in the future" (or "Possibly") **EF**: **EF**φ is true in s_t iff φ may be true in **some** $s_{t'}$ with $t' \geq t$

CTL semantics: intuitions [cont.]

- "Globally" (or "always") **AG**: **AG**φ is true in s_t iff φ is true in **all** $s_{t'}$ with $t' \geq t$
- "Possibly henceforth" **EG**: **EG**φ is true in s_t iff φ is possibly true henceforth
- "Necessarily Until" **AU**: **A**(φ**U**ψ) is true in s_t iff necessarily φ holds until ψ holds.
- "Possibly Until" **EU**: **E**(φ**U**ψ) is true in s_t iff possibly φ holds until ψ holds.

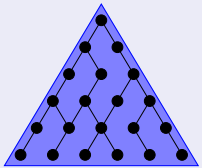
CTL semantics: intuitions [cont.]

finally P



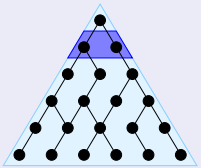
AF P

globally P



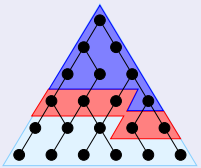
AG P

next P

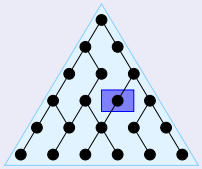


AX P

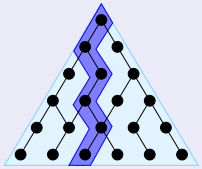
P until q



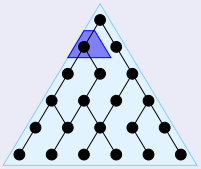
A[P U q]



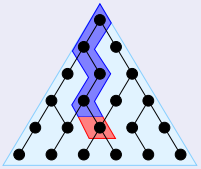
EF P



EG P



EX P



E[P U q]

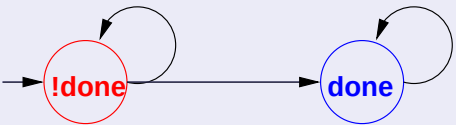
CTL Formal Semantics

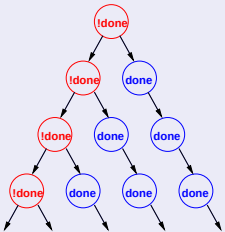
Let $(s_i, s_{i+1}, \ldots)$ be a path outgoing from state s_i in M

M, s_i	$\models$	a	iff	$a \in L(s_i)$	
M, s_i	$\models$	$\neg\varphi$	iff	$M, s_i \not\models \varphi$	
M, s_i	$\models$	$\varphi \vee \psi$	iff	$M, s_i \models \varphi$ *or* $M, s_i \models \psi$	
M, s_i	$\models$	$AX\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots)$,	$M, s_{i+1} \models \varphi$
M, s_i	$\models$	$EX\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots)$,	$M, s_{i+1} \models \varphi$
M, s_i	$\models$	$AG\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots)$,	*for all* $j \geq i. M, s_j \models \varphi$
M, s_i	$\models$	$EG\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots)$,	*for all* $j \geq i. M, s_j \models \varphi$
M, s_i	$\models$	$AF\varphi$	iff	*for all* $(s_i, s_{i+1}, \ldots)$,	*for some* $j \geq i. M, s_j \models \varphi$
M, s_i	$\models$	$EF\varphi$	iff	*for some* $(s_i, s_{i+1}, \ldots)$,	*for some* $j \geq i. M, s_j \models \varphi$
M, s_i	$\models$	$A(\varphi U \psi)$	iff	*for all* $(s_i, s_{i+1}, \ldots)$,	*for some* $j \geq i.$ $(M, s_j \models \psi$ *and* *forall* k *s.t.* $i \leq k < j. M, s_k \models \varphi)$
M, s_i	$\models$	$E(\varphi U \psi)$	iff	*for some* $(s_i, s_{i+1}, \ldots)$,	*for some* $j \geq i.$ $(M, s_j \models \psi$ *and* *forall* k *s.t.* $i \leq k < j. M, s_k \models \varphi)$

Formal Semantics (cont.)

- CTL properties (e.g. **AF***done*) are evaluated over trees.





- Every temporal operator (**F**, **G**, **X**, **U**) is preceded by a path quantifier (**A** or **E**).
- Universal modalities (**AF**, **AG**, **AX**, **AU**): the temporal formula is true in **all** the paths starting in the current state.
- Existential modalities (**EF**, **EG**, **EX**, **EU**): the temporal formula is true in **some** path starting in the current state.

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

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

$\mathcal{M}, s \models \phi$ for every initial state $s \in I$ of the Kripke structure

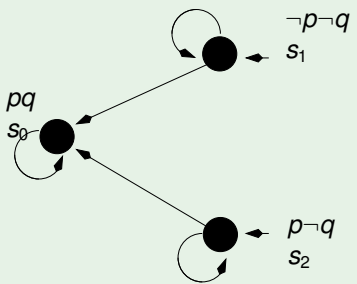
Important Remark

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

- E.g. if ϕ is a universal formula **A**... and two initial states s_0, s_1 are s.t. $\mathcal{M}, s_0 \models \phi$ and $\mathcal{M}, s_1 \not\models \phi$
- $\mathcal{M} \not\models \phi \Longrightarrow \mathcal{M} \models \neg\phi$ if $\mathcal{M}$ has only one initial state

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

- $\mathcal{M} \not\models \mathbf{AG}p$, in fact:
  - $\mathcal{M}, s_1 \not\models \mathbf{AG}p$
    (e.g., $\{s_1, ...\}$ is a counter-example)
  - $\mathcal{M}, s_2 \models \mathbf{AG}p$
- $\mathcal{M} \not\models \neg\mathbf{AG}p$, in fact:
  - $\mathcal{M}, s_1 \models \neg\mathbf{AG}p$
    (i.e., $\mathcal{M}, s_1 \models \mathbf{EF}\neg p$)
  - $\mathcal{M}, s_2 \not\models \neg\mathbf{AG}p$
    (i.e., $\mathcal{M}, s_2 \not\models \mathbf{EF}\neg p$)



Syntactic properties of CTL operators

$$
\begin{array}{lcl}
\varphi_1 \vee \varphi_2 & \iff & \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\ldots & & \\
\mathbf{A}(\varphi_1 \mathbf{U} \varphi_2) & \iff & \neg\mathbf{E}(\neg\varphi_2 \mathbf{U}(\neg\varphi_1 \wedge \neg\varphi_2)) \wedge \neg\mathbf{EG}\neg\varphi_2 \\
\mathbf{EF}\ \varphi_1 & \iff & \mathbf{E}(\top \mathbf{U} \varphi_1) \\
\mathbf{AG}\varphi_1 & \iff & \neg\mathbf{EF}\neg\varphi_1 \\
\mathbf{AF}\ \varphi_1 & \iff & \neg\mathbf{EG}\neg\varphi_1 \\
\mathbf{AX}\varphi_1 & \iff & \neg\mathbf{EX}\neg\varphi_1
\end{array}
$$

Note

CTL can be defined in terms of $\wedge$, $\neg$, **EX**, **EG**, **EU** only

Exercise:

prove that $\mathbf{A}(\varphi_1 \mathbf{U} \varphi_2) \iff \neg\mathbf{EG}\neg\varphi_2 \wedge \neg\mathbf{E}(\neg\varphi_2 \mathbf{U}(\neg\varphi_1 \wedge \neg\varphi_2))$

Strength of CTL operators

- $\mathbf{A}[\mathbf{OP}]\varphi \models \mathbf{E}[\mathbf{OP}]\varphi$, s.t. $[\mathbf{OP}] \in \{\mathbf{X}, \mathbf{F}, \mathbf{G}, \mathbf{U}\}$
- $\mathbf{AG}\varphi \models \varphi \models \mathbf{AF}\varphi$, $\mathbf{EG}\varphi \models \varphi \models \mathbf{EF}\varphi$
- $\mathbf{AG}\varphi \models \mathbf{AX}\varphi \models \mathbf{AF}\varphi$, $\mathbf{EG}\varphi \models \mathbf{EX}\varphi \models \mathbf{EF}\varphi$
- $\mathbf{AG}\varphi \models \mathbf{AX}...\mathbf{AX}\varphi \models \mathbf{AF}\varphi$, $\mathbf{EG}\varphi \models \mathbf{EX}...\mathbf{EX}\varphi \models \mathbf{EF}\varphi$
- $\mathbf{A}(\varphi \mathbf{U} \psi) \models \mathbf{AF}\psi$, $\mathbf{E}(\varphi \mathbf{U} \psi) \models \mathbf{EF}\psi$

CTL tableaux rules

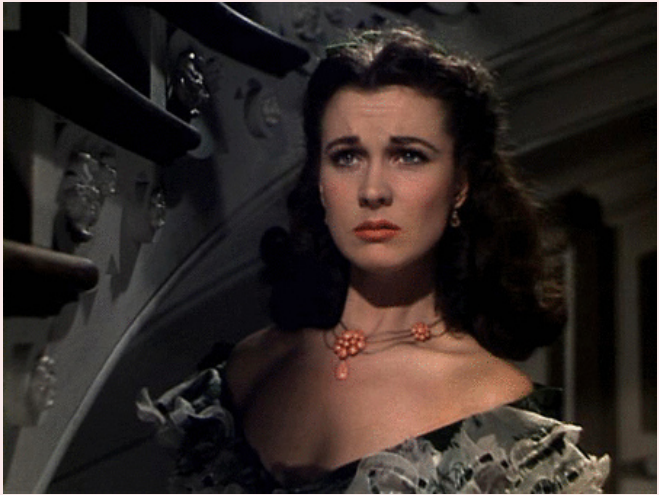
- Let φ_1 and φ_2 be CTL formulae:

$$\begin{aligned}
\mathbf{AF}\varphi_1 &\iff (\varphi_1 \vee \mathbf{AXAF}\varphi_1) \\
\mathbf{AG}\varphi_1 &\iff (\varphi_1 \wedge \mathbf{AXAG}\varphi_1) \\
\mathbf{A}(\varphi_1 \mathbf{U} \varphi_2) &\iff (\varphi_2 \vee (\varphi_1 \wedge \mathbf{AXA}(\varphi_1 \mathbf{U} \varphi_2))) \\
\mathbf{EF}\varphi_1 &\iff (\varphi_1 \vee \mathbf{EXEF}\varphi_1) \\
\mathbf{EG}\varphi_1 &\iff (\varphi_1 \wedge \mathbf{EXEG}\varphi_1) \\
\mathbf{E}(\varphi_1 \mathbf{U} \varphi_2) &\iff (\varphi_2 \vee (\varphi_1 \wedge \mathbf{EXE}(\varphi_1 \mathbf{U} \varphi_2)))
\end{aligned}$$

- Recursive definitions of **AF**, **AG**, **AU**, **EF**, **EG**, **EU**.
- If applied recursively, rewrite a CTL formula in terms of atomic, **AX**- and **EX**-formulas:

$$\mathbf{A}(p \mathbf{U} q) \wedge (\mathbf{EG} \neg p) \Longrightarrow (q \vee (p \wedge \mathbf{AXA}(p \mathbf{U} q))) \wedge (\neg p \wedge \mathbf{EXEG} \neg p)$$

Tableaux Rules: a Quote

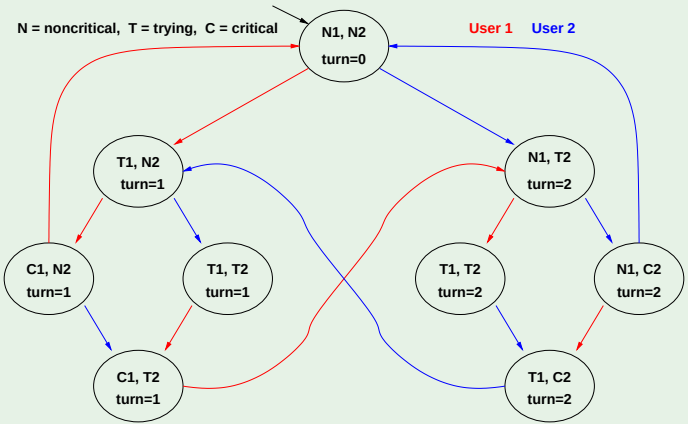


“After all... tomorrow is another day."
[Scarlett O’Hara, “Gone with the Wind”]

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

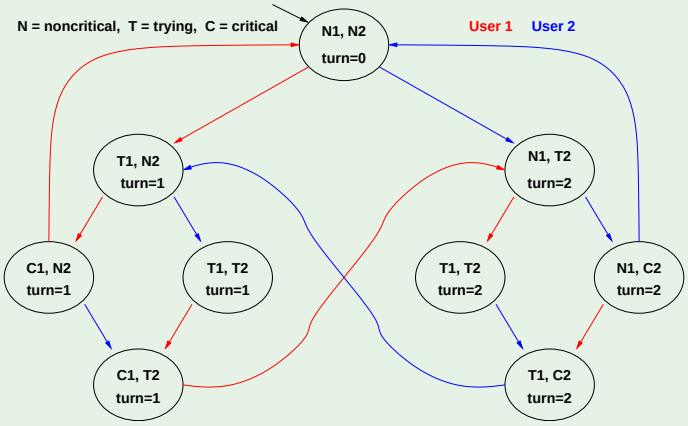
Example 1: mutual exclusion (safety)



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

YES: There is no reachable state in which $(C_1 \wedge C_2)$ holds!
(Same as the $\mathbf{G}\neg(C_1 \wedge C_2)$ in LTL.)

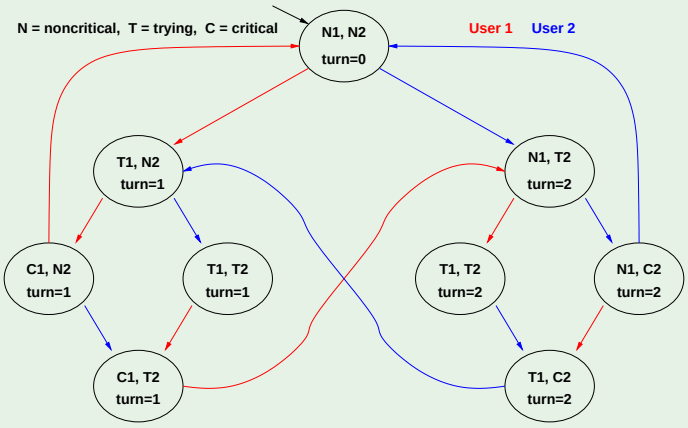
Example 2: liveness



$M \models \mathbf{AF}\, C_1$?

No: there is an infinite cyclic solution in which C_1 never holds!
(Same as $\mathbf{F}C_1$ in LTL.)

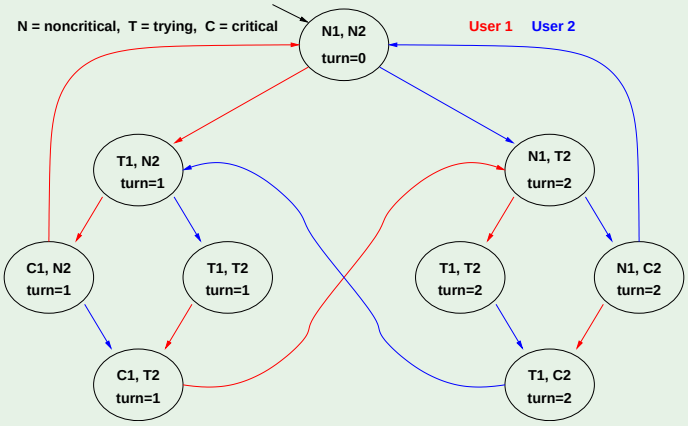
Example 3: liveness



$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}\, C_1)$?

YES: every path starting from each state where T_1 holds passes through a state where C_1 holds (Same as $\mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$ in LTL.)

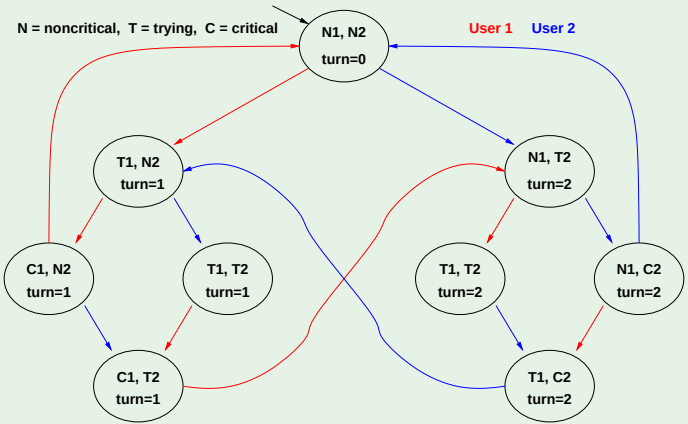
Example 4: fairness



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

NO: e.g., in the initial state, there is an infinite cyclic solution in which C_1 never holds!
(Same as $\mathbf{GF}C_1$ in LTL.)

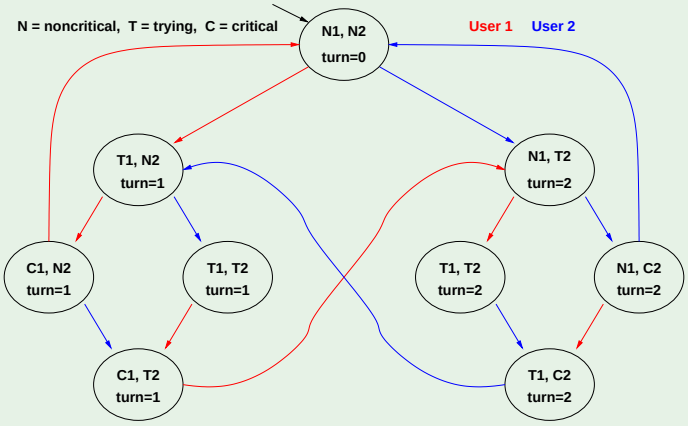
Example 5: fairness (2)



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

NO: there is an infinite 8-shaped cyclic solution in which $(turn = 0)$ never holds!

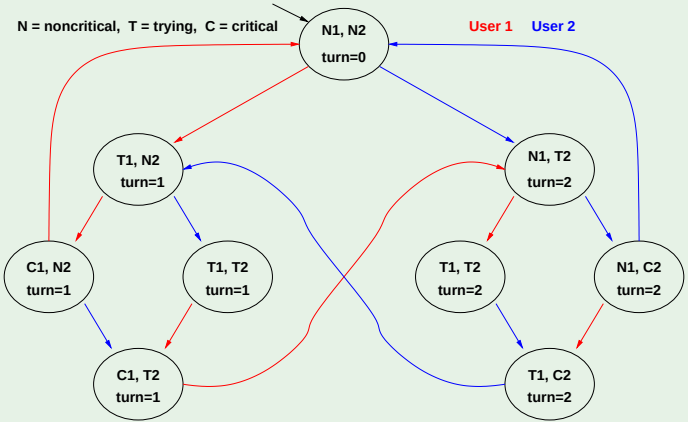
Example 6: blocking



$M \models \mathbf{AG}(N_1 \rightarrow \mathbf{EF}\ T_1)$?

YES: from each state where N_1 holds there is a path leading to a state where T_1 holds
(No corresponding LTL formula.)

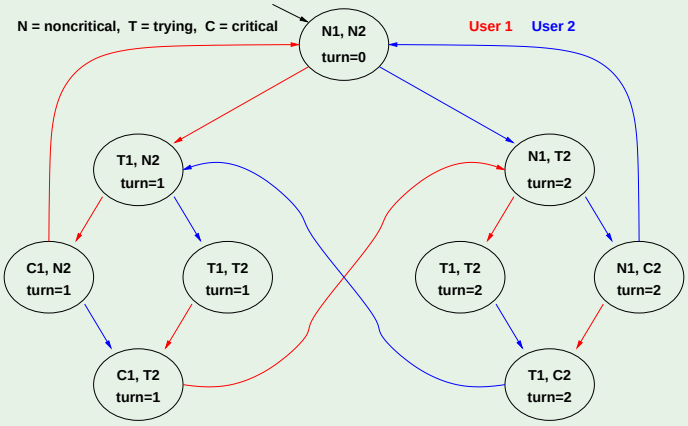
Example 7: blocking (2)



$M \models \mathbf{AG}(N_1 \rightarrow \mathbf{AF}\, T_1)$?

NO: e.g., in the initial state, there is an infinite cyclic solution in which N_1 holds and T_1 never holds!
(Same as LTL formula $\mathbf{G}(N_1 \rightarrow \mathbf{F}T_1)$.)

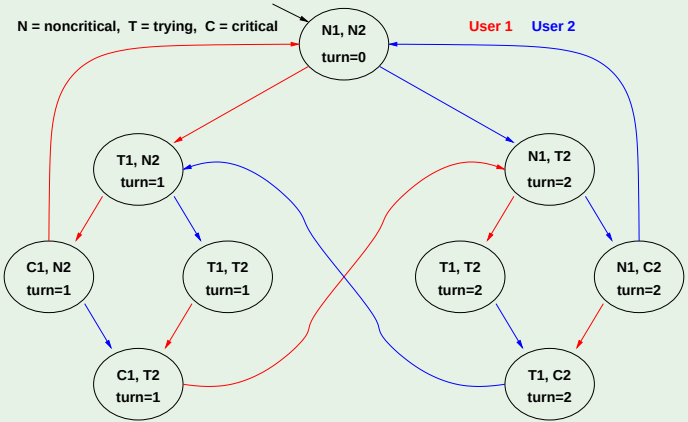
Example 8:



$M \models \mathbf{EG} N_1$?

YES: there is an infinite cyclic solution where N_1 always holds
(No corresponding LTL formula.)

Example 9:



$M \models \mathbf{AFEG}N_1$?

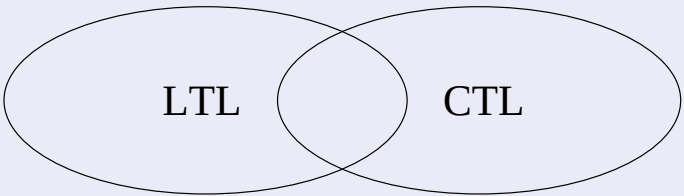
YES: there is an infinite cyclic solution where N_1 always holds, and from every state you necessarily reach one state of such cycle
(No corresponding LTL formula.)

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

LTL vs. CTL: expressiveness

- many CTL formulas cannot be expressed in LTL
  (e.g., those containing existentially quantified subformulas)
  E.g., $\mathbf{AG}(N_1 \rightarrow \mathbf{EF}T_1)$, $\mathbf{AFAG}\varphi$
- many LTL formulas cannot be expressed in CTL
  (e.g. fairness LTL formulas)
  E.g., $\mathbf{GF}T_1 \rightarrow \mathbf{GF}C_1$, $\mathbf{FG}\varphi$
- some formulas can be expressed both in LTL and in CTL (typically LTL formulas with operators of nesting depth 1, and/or with operators occurring positively)
  E.g., $\mathbf{G}\neg(C_1 \wedge C_2)$, $\mathbf{F}C_1$, $\mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$, $\mathbf{GF}C_1$

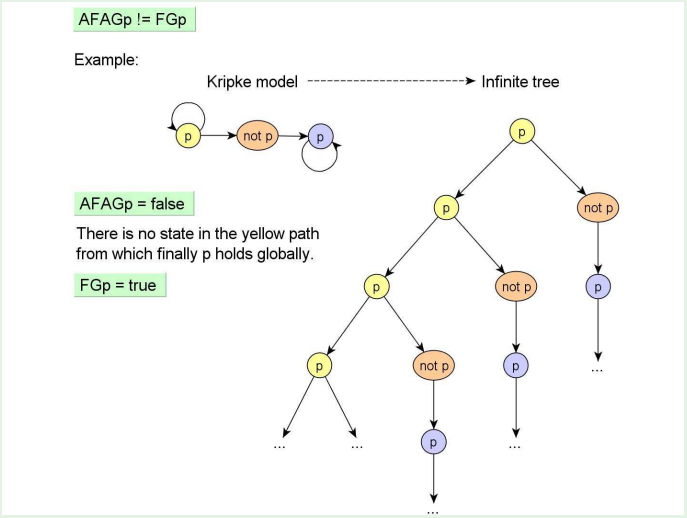


Example: *AFAGp* vs. *FGp*

(Example developed by the students Andrea Mattioli and Mirko Boniatti, 2005.)

AFAGp != FGp

Example:



AFAGp = false

There is no state in the yellow path from which finally p holds globally.

FGp = true

LTL vs. CTL: M.C. Algorithms

- LTL M.C. problems are typically handled with automata- based M.C. approaches (Wolper & Vardi)
- CTL M.C. problems are typically handled with symbolic M.C. approaches (Clarke & McMillan)
- LTL M.C. problems can be reduced to CTL M.C. problems under fairness constraints (Clarke et al.)

CTL*

- Syntax: let p's, φ's, ψ's being propositions, state formulae and path formulae respectively:
  - p, $\neg\varphi$, $\varphi_1 \wedge \varphi_2$, $\mathbf{A}\psi$, $\mathbf{E}\psi$ are state formulae
    (properties of the set of paths starting from a state)
  - φ, $\neg\psi$, $\psi_1 \wedge \psi_2$, $\mathbf{X}\psi$, $\mathbf{G}\psi$, $\mathbf{F}\psi$, $\psi_1\mathbf{U}\psi_2$ are path formulae
    (properties of a path)
- Semantics: **A**, **E**, **X**, **G**, **F**, **U** as in CTL
  - **A**, **E**: quantify on paths (as in CTL)
  - **X**, **G**, **F**, **U**: (as in LTL)
  - as in CTL, but **X**, **G**, **F**, **U** not necessarily preceded by **A**,**E**

Remark

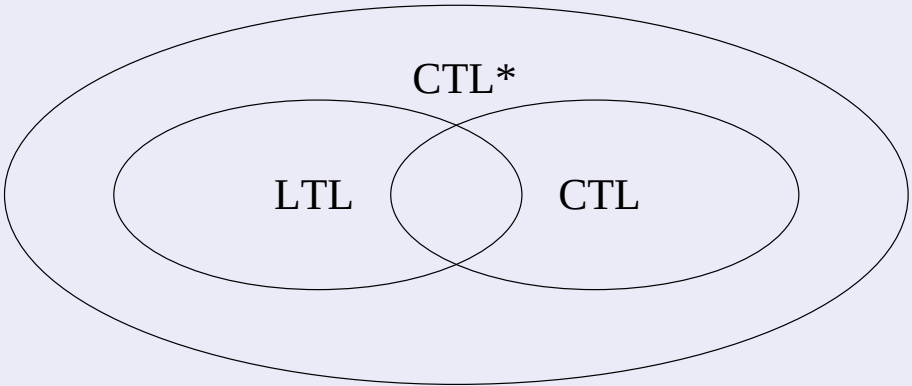
In principle in CTL* one may have sequences of nested path quantifiers.
In such case, the most internal one dominates:

$$M, s \models \mathbf{AE}\psi \text{ iff } M, s \models \mathbf{E}\psi, \qquad M, s \models \mathbf{EA}\psi \text{ iff } M, s \models \mathbf{A}\psi.$$

CTL* vs LTL & CTL

CTL* subsumes both CTL and LTL

- φ in CTL $\Longrightarrow$ φ in CTL* (e.g., $\mathbf{AG}(N_1 \rightarrow \mathbf{EF}T_1)$
- φ in LTL $\Longrightarrow$ $\mathbf{A}\varphi$ in CTL* (e.g., $\mathbf{A}(\mathbf{GF}T_1 \rightarrow \mathbf{GF}C_1)$
- LTL $\cup$ CTL $\subset$ CTL* (e.g., $\mathbf{E}(\mathbf{GF}p \rightarrow \mathbf{GF}q)$)



“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

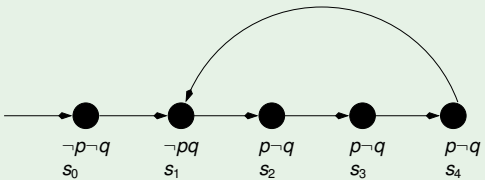
(Arnold Schwarzenegger in “Twins”)

Outline

1. Transition Systems as Kripke Models
   - Kripke Models
   - Languages for Transition Systems (hints)
2. Properties and Temporal Logics
   - Properties
   - Temporal Logics
3. Linear Temporal Logic – LTL
   - LTL: Syntax and Semantics
   - Some LTL Model Checking Examples
4. Computation Tree Logic – CTL
   - CTL: Syntax and Semantics
   - Some CTL Model Checking Examples
5. LTL vs. CTL
6. Exercises

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]

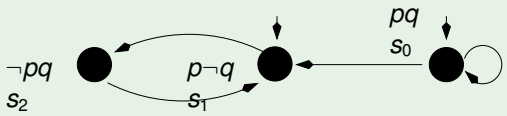
(b) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$
[Solution: true]

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

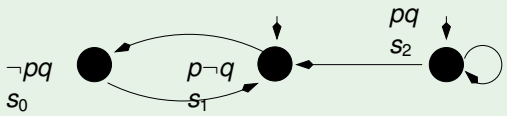
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$
[Solution: true]

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

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

Ex: CTL Model Checking

Consider the following Kripke Model M:



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

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

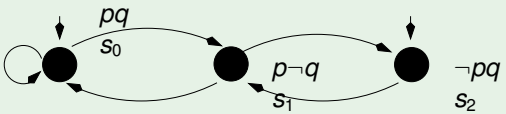
(b) $M \models \mathbf{EG}p$
[Solution: false]

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

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

Ex: CTL Model Checking

Consider the following Kripke Model M:



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

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

(b) $M \models \mathbf{EG}q$
[Solution: false]

(c) $M \models ((\mathbf{AGAF}p \vee \mathbf{AGAF}q) \wedge (\mathbf{AGAF}\neg p \vee \mathbf{AGAF}\neg q)) \rightarrow q$
[Solution: true]

(d) $M \models \mathbf{AFEG}(p \wedge q)$
[Solution: false]