Introduction to Formal Methods
Chapter 03: Temporal Logics

Roberto Sebastiani

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

CDLM in Informatica, academic year 2019-2020

last update: Thursday 20th February, 2020, 19:04

Outline

Boolean logic

Basic notation & definitions

- Boolean formula
  - $\top, \bot$ are formulas
  - A propositional atom $A_1, A_2, A_3, ...$ is a formula;
  - if φ_1 and φ_2 are formulas, then
    $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$
    are formulas.
- *Atoms*(φ): the set $\{A_1, ..., A_N\}$ of atoms occurring in φ.
- Literal: a propositional atom A_i (positive literal) or its negation $\neg A_i$ (negative literal)
  - Notation: if $l := \neg A_i$, then $\neg l := A_i$
- Clause: a disjunction of literals $\bigvee_j l_j$ (e.g., $(A_1 \vee \neg A_2 \vee A_3 \vee ...)$)
- Cube: a conjunction of literals $\bigwedge_j l_j$ (e.g., $(A_1 \wedge \neg A_2 \wedge A_3 \wedge ...)$)

Semantics of Boolean operators

- Truth table:

φ_1	φ_2	$\neg\varphi_1$	$\varphi_1 \wedge \varphi_2$	$\varphi_1 \vee \varphi_2$	$\varphi_1 \rightarrow \varphi_2$	$\varphi_1 \leftarrow \varphi_2$	$\varphi_1 \leftrightarrow \varphi_2$
$\bot$	$\bot$	$\top$	$\bot$	$\bot$	$\top$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$
$\top$	$\bot$	$\bot$	$\bot$	$\top$	$\bot$	$\top$	$\bot$
$\top$	$\top$	$\bot$	$\top$	$\top$	$\top$	$\top$	$\top$

Note

- $\wedge$, $\vee$ and $\leftrightarrow$ are commutative:
  $$(\varphi_1 \wedge \varphi_2) \iff (\varphi_2 \wedge \varphi_1)$$
  $$(\varphi_1 \vee \varphi_2) \iff (\varphi_2 \vee \varphi_1)$$
  $$(\varphi_1 \leftrightarrow \varphi_2) \iff (\varphi_2 \leftrightarrow \varphi_1)$$
- $\wedge$ and $\vee$ are associative:
  $$((\varphi_1 \wedge \varphi_2) \wedge \varphi_3) \iff (\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \iff (\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$$
  $$((\varphi_1 \vee \varphi_2) \vee \varphi_3) \iff (\varphi_1 \vee (\varphi_2 \vee \varphi_3)) \iff (\varphi_1 \vee \varphi_2 \vee \varphi_3)$$

Syntactic Properties of Boolean Operators

$$
\begin{array}{rcl}
\neg\neg\varphi_1 & \Longleftrightarrow & \varphi_1 \\
(\varphi_1 \vee \varphi_2) & \Longleftrightarrow & \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\neg(\varphi_1 \vee \varphi_2) & \Longleftrightarrow & (\neg\varphi_1 \wedge \neg\varphi_2) \\
(\varphi_1 \wedge \varphi_2) & \Longleftrightarrow & \neg(\neg\varphi_1 \vee \neg\varphi_2) \\
\neg(\varphi_1 \wedge \varphi_2) & \Longleftrightarrow & (\neg\varphi_1 \vee \neg\varphi_2) \\
(\varphi_1 \rightarrow \varphi_2) & \Longleftrightarrow & (\neg\varphi_1 \vee \varphi_2) \\
\neg(\varphi_1 \rightarrow \varphi_2) & \Longleftrightarrow & (\varphi_1 \wedge \neg\varphi_2) \\
(\varphi_1 \leftarrow \varphi_2) & \Longleftrightarrow & (\varphi_1 \vee \neg\varphi_2) \\
\neg(\varphi_1 \leftarrow \varphi_2) & \Longleftrightarrow & (\neg\varphi_1 \wedge \varphi_2) \\
(\varphi_1 \leftrightarrow \varphi_2) & \Longleftrightarrow & ((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_1 \leftarrow \varphi_2)) \\
 & \Longleftrightarrow & ((\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2)) \\
\neg(\varphi_1 \leftrightarrow \varphi_2) & \Longleftrightarrow & (\neg\varphi_1 \leftrightarrow \varphi_2) \\
 & \Longleftrightarrow & (\varphi_1 \leftrightarrow \neg\varphi_2) \\
 & \Longleftrightarrow & ((\varphi_1 \vee \varphi_2) \wedge (\neg\varphi_1 \vee \neg\varphi_2))
\end{array}
$$

Boolean logic can be expressed in terms of $\{\neg, \wedge\}$ (or $\{\neg, \vee\}$) only

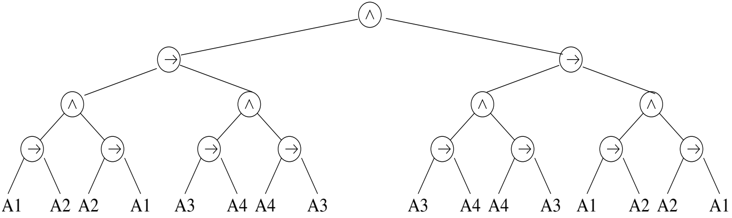
TREE and DAG representation of formulas: example

Formulas can be represented either as trees or as DAGS:

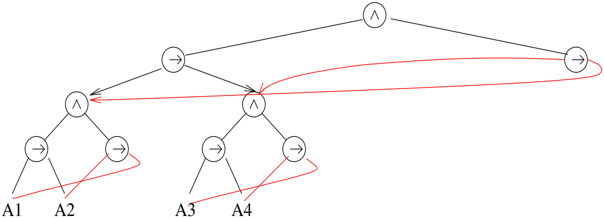
- DAG representation can be up to exponentially smaller

$$(A_1 \leftrightarrow A_2) \leftrightarrow (A_3 \leftrightarrow A_4)$$
$$\Downarrow$$
$$(((A_1 \leftrightarrow A_2) \rightarrow (A_3 \leftrightarrow A_4))\wedge$$
$$((A_3 \leftrightarrow A_4) \rightarrow (A_1 \leftrightarrow A_2)))$$
$$\Downarrow$$
$$(((A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_1)) \rightarrow ((A_3 \rightarrow A_4) \wedge (A_4 \rightarrow A_3)))\wedge$$
$$(((A_3 \rightarrow A_4) \wedge (A_4 \rightarrow A_3)) \rightarrow (((A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_1))))$$

TREE and DAG representation of formulas: example (cont)



Tree Representation



DAG Representation

Basic notation & definitions (cont)

- Total truth assignment μ for φ:
  $\mu : \mathit{Atoms}(\varphi) \longmapsto \{\top, \bot\}$.
- Partial Truth assignment μ for φ:
  $\mu : \mathcal{A} \longmapsto \{\top, \bot\}$, $\mathcal{A} \subset \mathit{Atoms}(\varphi)$.
- Set and formula representation of an assignment:
  - μ can be represented as a set of literals:
    EX: $\{\mu(A_1) := \top, \mu(A_2) := \bot\} \implies \{A_1, \neg A_2\}$
  - μ can be represented as a formula (cube):
    EX: $\{\mu(A_1) := \top, \mu(A_2) := \bot\} \implies (A_1 \land \neg A_2)$

Basic notation & definitions (cont)

- a total truth assignment μ satisfies φ ($\mu \models \varphi$):
  - $\mu \models A_i \iff \mu(A_i) = \top$
  - $\mu \models \neg\varphi \iff$ *not* $\mu \models \varphi$
  - $\mu \models \varphi_1 \wedge \varphi_2 \iff \mu \models \varphi_1$ *and* $\mu \models \varphi_2$
  - $\mu \models \varphi_1 \vee \varphi_2 \iff \mu \models \varphi_1$ *or* $\mu \models \varphi_2$
  - $\mu \models \varphi_1 \rightarrow \varphi_2 \iff$ *if* $\mu \models \varphi_1$, *then* $\mu \models \varphi_2$
  - $\mu \models \varphi_1 \leftrightarrow \varphi_2 \iff \mu \models \varphi_1$ *iff* $\mu \models \varphi_2$
- a partial truth assignment μ satisfies φ iff it makes φ evaluate to true (Ex: $\{A_1\} \models (A_1 \vee A_2)$)

  $\Longrightarrow$ if μ satisfies φ, then all its total extensions satisfy φ (Ex: $\{A_1, A_2\} \models (A_1 \vee A_2)$ and $\{A_1, \neg A_2\} \models (A_1 \vee A_2)$)
- φ is satisfiable iff $\mu \models \varphi$ for some μ
- φ_1 entails φ_2 ($\varphi_1 \models \varphi_2$): $\varphi_1 \models \varphi_2$ iff $\mu \models \varphi_1 \Longrightarrow \mu \models \varphi_2$ for every μ
- φ is valid ($\models \varphi$): $\models \varphi$ iff $\mu \models \varphi$ for every μ

Property

φ is valid $\iff \neg\varphi$ is not satisfiable

Equivalence and equi-satisfiability

- φ_1 and φ_2 are equivalent iff, for every μ, $\mu \models \varphi_1$ iff $\mu \models \varphi_2$
- φ_1 and φ_2 are equi-satisfiable iff
  exists μ_1 s.t. $\mu_1 \models \varphi_1$ iff exists μ_2 s.t. $\mu_2 \models \varphi_2$
- φ_1, φ_2 equivalent
  $\Downarrow \quad \not\Uparrow$
  φ_1, φ_2 equi-satisfiable
- EX: $\varphi_1 \stackrel{\text{def}}{=} \psi_1 \vee \psi_2$ and $\varphi_2 \stackrel{\text{def}}{=} (\psi_1 \vee \neg A_3) \wedge (A_3 \vee \psi_2)$ s.t. A_3 not in $\psi_1 \vee \psi_2$, are equi-satisfiable but not equivalent:
  - $\mu \models (\psi_1 \vee \neg A_3) \wedge (A_3 \vee \psi_2) \Longrightarrow \mu \models \psi_1 \vee \psi_2$
  - $\mu' \models \psi_1 \vee \psi_2 \Longrightarrow \mu' \wedge A_3 \models (\psi_1 \vee \neg A_3) \wedge (A_3 \vee \psi_2)$ or $\mu' \wedge \neg A_3 \models (\psi_1 \vee \neg A_3) \wedge (A_3 \vee \psi_2)$ [φ_1, φ_2 equi-satisfiable]
  - $\mu' \not\models \psi_1$ and $\mu' \models \psi_2 \Longrightarrow \mu' \wedge A_3 \models \psi_1 \vee \psi_2$ and $\mu' \wedge A_3 \not\models (\psi_1 \vee \neg A_3) \wedge (A_3 \vee \psi_2)$ [φ_1, φ_2 not equivalent]
- Typically used when φ_2 is the result of applying some transformation T to φ_1: $\varphi_2 \stackrel{\text{def}}{=} T(\varphi_1)$:
  we say that T is validity-preserving [satisfiability-preserving] iff $T(\varphi_1)$ and φ_1 are equivalent [equi-satisfiable]

Complexity

- For N variables, there are up to 2^N truth assignments to be checked.
- The problem of deciding the satisfiability of a propositional formula is NP-complete
- The most important logical problems (validity, inference, entailment, equivalence, ...) can be straightforwardly reduced to satisfiability, and are thus (co)NP-complete.

$$\Downarrow$$

No existing worst-case-polynomial algorithm.

POLARITY of subformulas

- Positive/negative occurrences
  - φ occurs positively in φ;
  - if $\neg\varphi_1$ occurs positively [negatively] in φ, then φ_1 occurs negatively [positively] in φ
  - if $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$ occur positively [negatively] in φ, then φ_1 and φ_2 occur positively [negatively] in φ;
  - if $\varphi_1 \rightarrow \varphi_2$ occurs positively [negatively] in φ, then φ_1 occurs negatively [positively] in φ and φ_2 occurs positively [negatively] in φ;
  - if $\varphi_1 \leftrightarrow \varphi_2$ occurs in φ, then φ_1 and φ_2 occur positively and negatively in φ;
- EX:
  - φ_1 occurs positively in $\neg(\varphi_1 \rightarrow \varphi_2)$
  - φ_2 occurs negatively in $\neg(\varphi_1 \rightarrow \varphi_2)$
- intuition: φ_1 occurs positively [negatively] in φ iff it occurs under the scope of an (implicit) even [odd] number of negations.

$\Longrightarrow$ Polarity: the number of nested negations modulo 2.

Substitution

Properties

- If φ_1 is equivalent to φ_2, then $\varphi[\varphi_1|\varphi_2]$ is equivalent to φ:

$$\models (\varphi_1 \leftrightarrow \varphi_2)$$
$$\Downarrow$$
$$\models \varphi[\varphi_1|\varphi_2] \leftrightarrow \varphi$$

- If φ_2 entails φ_1 and φ_1 occurs only positively in φ, then $\varphi[\varphi_1|\varphi_2]$ entails φ:

$$\varphi_2 \models \varphi_1$$
$$\Downarrow$$
$$\varphi[\varphi_1|\varphi_2] \models \varphi$$

- dual case for negative occurrence

Negative normal form (NNF)

- φ is in Negative normal form iff it is given only by the recursive applications of $\wedge, \vee$ to literals.
- every φ can be reduced into NNF:
  (i) substituting all $\rightarrow$'s and $\leftrightarrow$'s:

$$\begin{array}{lcl} \varphi_1 \rightarrow \varphi_2 & \Longrightarrow & \neg\varphi_1 \vee \varphi_2 \\ \varphi_1 \leftrightarrow \varphi_2 & \Longrightarrow & (\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2) \end{array}$$

  (ii) pushing down negations recursively:

$$\begin{array}{lcl} \neg(\varphi_1 \wedge \varphi_2) & \Longrightarrow & \neg\varphi_1 \vee \neg\varphi_2 \\ \neg(\varphi_1 \vee \varphi_2) & \Longrightarrow & \neg\varphi_1 \wedge \neg\varphi_2 \\ \neg\neg\varphi_1 & \Longrightarrow & \varphi_1 \end{array}$$

- The reduction is linear if a DAG representation is used.
- Preserves the equivalence of formulas.

NNF: example

$$(A_1 \leftrightarrow A_2) \leftrightarrow (A_3 \leftrightarrow A_4)$$

$$\Downarrow$$

$$\begin{array}{c}((((A_1 \rightarrow A_2) \wedge (A_1 \leftarrow A_2)) \rightarrow ((A_3 \rightarrow A_4) \wedge (A_3 \leftarrow A_4)))\wedge \\ (((A_1 \rightarrow A_2) \wedge (A_1 \leftarrow A_2)) \leftarrow ((A_3 \rightarrow A_4) \wedge (A_3 \leftarrow A_4))))\end{array}$$

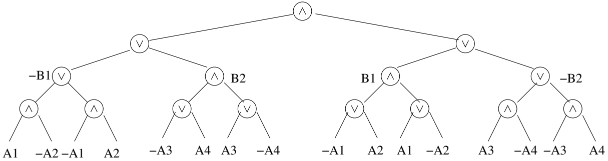
$$\Downarrow$$

$$\begin{array}{c}((\neg((\neg A_1 \vee A_2) \wedge (A_1 \vee \neg A_2)) \vee ((\neg A_3 \vee A_4) \wedge (A_3 \vee \neg A_4)))\wedge \\ (((\neg A_1 \vee A_2) \wedge (A_1 \vee \neg A_2)) \vee \neg((\neg A_3 \vee A_4) \wedge (A_3 \vee \neg A_4))))\end{array}$$

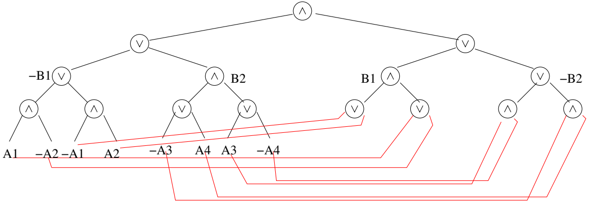
$$\Downarrow$$

$$\begin{array}{c}((((A_1 \wedge \neg A_2) \vee (\neg A_1 \wedge A_2)) \vee ((\neg A_3 \vee A_4) \wedge (A_3 \vee \neg A_4)))\wedge \\ (((\neg A_1 \vee A_2) \wedge (A_1 \vee \neg A_2)) \vee ((A_3 \wedge \neg A_4) \vee (\neg A_3 \wedge A_4))))\end{array}$$

NNF: example (cont)



Tree Representation



DAG Representation

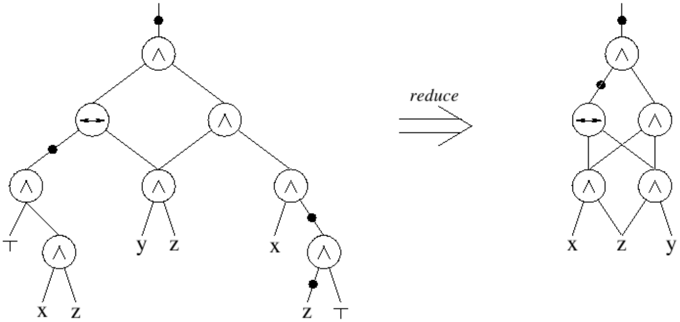
Note

For each non-literal subformula φ, φ and $\neg\varphi$ have different representations $\Longrightarrow$ they are not shared.

Optimized polynomial representations

And-Inverter Graphs, Reduced Boolean Circuits, Boolean Expression Diagrams

- Maximize the sharing in DAG representations:
  $\{\wedge, \leftrightarrow, \neg\}$-only, negations on arcs, sorting of subformulae, lifting of $\neg$'s over $\leftrightarrow$'s,...



Conjunctive Normal Form (CNF)

- φ is in Conjunctive normal form iff it is a conjunction of disjunctions of literals:

$$\bigwedge_{i=1}^{L} \bigvee_{j_i=1}^{K_i} l_{j_i}$$

- the disjunctions of literals $\bigvee_{j_i=1}^{K_i} l_{j_i}$ are called clauses
- Easier to handle: list of lists of literals.
  $\Longrightarrow$ no reasoning on the recursive structure of the formula

Classic CNF Conversion $CNF(\varphi)$

- Every φ can be reduced into CNF by, e.g.,
  (i) converting it into NNF (not indispensible);
  (ii) applying recursively the DeMorgan's Rule:
  $$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \implies (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3)$$
- Worst-case exponential.
- $Atoms(CNF(\varphi)) = Atoms(\varphi)$.
- $CNF(\varphi)$ is equivalent to φ.
- Rarely used in practice.

Labeling CNF conversion $CNF_{label}(\varphi)$

- Every φ can be reduced into CNF by, e.g., applying recursively bottom-up the rules:

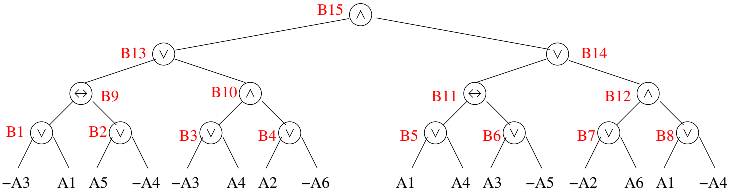
  $$\varphi \implies \varphi[(l_i \vee l_j)|B] \wedge CNF(B \leftrightarrow (l_i \vee l_j))$$
  $$\varphi \implies \varphi[(l_i \wedge l_j)|B] \wedge CNF(B \leftrightarrow (l_i \wedge l_j))$$
  $$\varphi \implies \varphi[(l_i \leftrightarrow l_j)|B] \wedge CNF(B \leftrightarrow (l_i \leftrightarrow l_j))$$

  l_i, l_j being literals and B being a "new" variable.
- Worst-case linear.
- $Atoms(CNF_{label}(\varphi)) \supseteq Atoms(\varphi)$.
- $CNF_{label}(\varphi)$ is equi-satisfiable w.r.t. φ.
- More used in practice.

Labeling CNF conversion $CNF_{label}(\varphi)$ (cont.)

$CNF(B \leftrightarrow (l_i \vee l_j))$	$\Longleftrightarrow$	$(\neg B \vee l_i \vee l_j) \wedge$ $(B \vee \neg l_i) \wedge$ $(B \vee \neg l_j)$
$CNF(B \leftrightarrow (l_i \wedge l_j))$	$\Longleftrightarrow$	$(\neg B \vee l_i) \wedge$ $(\neg B \vee l_j) \wedge$ $(B \vee \neg l_i \neg l_j)$
$CNF(B \leftrightarrow (l_i \leftrightarrow l_j))$	$\Longleftrightarrow$	$(\neg B \vee \neg l_i \vee l_j) \wedge$ $(\neg B \vee l_i \vee \neg l_j) \wedge$ $(B \vee l_i \vee l_j) \wedge$ $(B \vee \neg l_i \vee \neg l_j)$

Labeling CNF conversion CNF_{label} – example



$$
\begin{array}{ll}
CNF(B_1 \leftrightarrow (\neg A_3 \vee A_1)) & \wedge \\
\ldots & \wedge \\
CNF(B_8 \leftrightarrow (A_1 \vee \neg A_4)) & \wedge \\
CNF(B_9 \leftrightarrow (B_1 \leftrightarrow B_2)) & \wedge \\
\ldots & \wedge \\
CNF(B_{12} \leftrightarrow (B_7 \wedge B_8)) & \wedge \\
CNF(B_{13} \leftrightarrow (B_9 \vee B_{10})) & \wedge \\
CNF(B_{14} \leftrightarrow (B_{11} \vee B_{12})) & \wedge \\
CNF(B_{15} \leftrightarrow (B_{13} \wedge B_{14})) & \wedge \\
B_{15} &
\end{array}
$$

Labeling CNF conversion CNF_{label} (variant)

- As in the previous case, applying instead the rules:

$$
\begin{array}{llll}
\varphi & \Longrightarrow & \varphi[(l_i \vee l_j)|B] & \wedge\, CNF(B \rightarrow (l_i \vee l_j)) & \textit{if } (l_i \vee l_j) \textit{ pos.} \\
\varphi & \Longrightarrow & \varphi[(l_i \vee l_j)|B] & \wedge\, CNF((l_i \vee l_j) \rightarrow B) & \textit{if } (l_i \vee l_j) \textit{ neg.} \\
\varphi & \Longrightarrow & \varphi[(l_i \wedge l_j)|B] & \wedge\, CNF(B \rightarrow (l_i \wedge l_j)) & \textit{if } (l_i \wedge l_j) \textit{ pos.} \\
\varphi & \Longrightarrow & \varphi[(l_i \wedge l_j)|B] & \wedge\, CNF((l_i \wedge l_j) \rightarrow B) & \textit{if } (l_i \wedge l_j) \textit{ neg.} \\
\varphi & \Longrightarrow & \varphi[(l_i \leftrightarrow l_j)|B] & \wedge\, CNF(B \rightarrow (l_i \leftrightarrow l_j)) & \textit{if } (l_i \leftrightarrow l_j) \textit{ pos.} \\
\varphi & \Longrightarrow & \varphi[(l_i \leftrightarrow l_j)|B] & \wedge\, CNF((l_i \leftrightarrow l_j) \rightarrow B) & \textit{if } (l_i \leftrightarrow l_j) \textit{ neg.}
\end{array}
$$

- Pro: smaller in size:

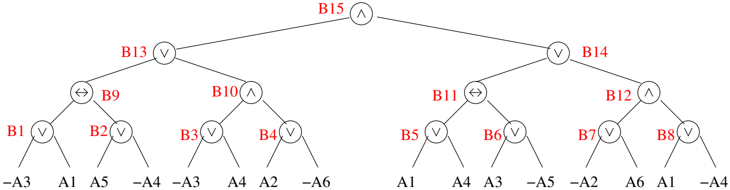
$$
\begin{array}{ll}
CNF(B \rightarrow (l_i \vee l_j)) & = (\neg B \vee l_i \vee l_j) \\
CNF(((l_i \vee l_j) \rightarrow B)) & = (\neg l_i \vee B) \wedge (\neg l_j \vee B)
\end{array}
$$

- Con: looses backward propagation:
  unlike with $CNF(B \leftrightarrow (l_i \vee l_j))$, with $CNF(B \rightarrow (l_i \vee l_j))$ we can no more infer that B is true from the fact that l_i is true or l_j is true.

Labeling CNF conversion $CNF_{label}(\varphi)$ (cont.)

$CNF(B \rightarrow (l_i \vee l_j))$	$\Longleftrightarrow$	$(\neg B \vee l_i \vee l_j)$
$CNF(B \leftarrow (l_i \vee l_j))$	$\Longleftrightarrow$	$(B \vee \neg l_i) \wedge$ $(B \vee \neg l_j)$
$CNF(B \rightarrow (l_i \wedge l_j))$	$\Longleftrightarrow$	$(\neg B \vee l_i) \wedge$ $(\neg B \vee l_j)$
$CNF(B \leftarrow (l_i \wedge l_j))$	$\Longleftrightarrow$	$(B \vee \neg l_i \neg l_j)$
$CNF(B \rightarrow (l_i \leftrightarrow l_j))$	$\Longleftrightarrow$	$(\neg B \vee \neg l_i \vee l_j) \wedge$ $(\neg B \vee l_i \vee \neg l_j)$
$CNF(B \leftarrow (l_i \leftrightarrow l_j))$	$\Longleftrightarrow$	$(B \vee l_i \vee l_j) \wedge$ $(B \vee \neg l_i \vee \neg l_j)$

Labeling CNF conversion CNF_{label} – example



Basic

$$
\begin{array}{ll}
CNF(B_1 \leftrightarrow (\neg A_3 \vee A_1)) & \wedge \\
... & \wedge \\
CNF(B_8 \leftrightarrow (A_1 \vee \neg A_4)) & \wedge \\
CNF(B_9 \leftrightarrow (B_1 \leftrightarrow B_2)) & \wedge \\
... & \wedge \\
CNF(B_{12} \leftrightarrow (B_7 \wedge B_8)) & \wedge \\
CNF(B_{13} \leftrightarrow (B_9 \vee B_{10})) & \wedge \\
CNF(B_{14} \leftrightarrow (B_{11} \vee B_{12})) & \wedge \\
CNF(B_{15} \leftrightarrow (B_{13} \wedge B_{14})) & \wedge \\
B_{15} &
\end{array}
$$

Improved

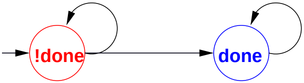
$$
\begin{array}{ll}
CNF(B_1 \leftrightarrow (\neg A_3 \vee A_1)) & \wedge \\
... & \wedge \\
CNF(B_8 \rightarrow (A_1 \vee \neg A_4)) & \wedge \\
CNF(B_9 \rightarrow (B_1 \leftrightarrow B_2)) & \wedge \\
... & \wedge \\
CNF(B_{12} \rightarrow (B_7 \wedge B_8)) & \wedge \\
CNF(B_{13} \rightarrow (B_9 \vee B_{10})) & \wedge \\
CNF(B_{14} \rightarrow (B_{11} \vee B_{12})) & \wedge \\
CNF(B_{15} \rightarrow (B_{13} \wedge B_{14})) & \wedge \\
B_{15} &
\end{array}
$$

Labeling CNF conversion CNF_{label} – further optimizations

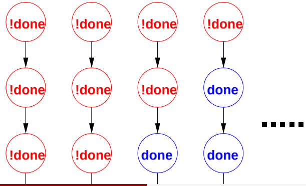
- Do not apply CNF_{label} when not necessary: (e.g., $CNF_{label}(\varphi_1 \wedge \varphi_2) \Longrightarrow CNF_{label}(\varphi_1) \wedge \varphi_2$, if φ_2 already in CNF)
- Apply Demorgan's rules where it is more effective: (e.g., $CNF_{label}(\varphi_1 \wedge (A \rightarrow (B \wedge C))) \Longrightarrow CNF_{label}(\varphi_1) \wedge (\neg A \vee B) \wedge (\neg A \vee C)$
- exploit the associativity of $\wedge$'s and $\vee$'s: $\ldots \underbrace{(A_1 \vee (A_2 \vee A_3))}_{B} \ldots \Longrightarrow \ldots CNF(B \leftrightarrow (A_1 \vee A_2 \vee A_3))\ldots$
- before applying CNF_{label}, rewrite the initial formula so that to maximize the sharing of subformulas (RBC, BED)
- ...

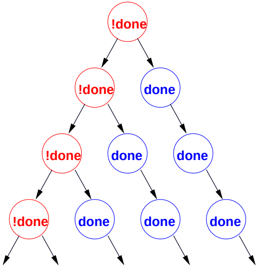
Computation tree vs. computation paths

- 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
- Computation Tree Logic (CTL)
  - interpreted over computation tree of Kripke model
  - branching model of time
  - temporal operators plus path quantifiers

Linear Temporal Logic (LTL): Syntax

- An atomic proposition is a LTL formula;
- if φ_1 and φ_2 are LTL formulae, then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$ 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.

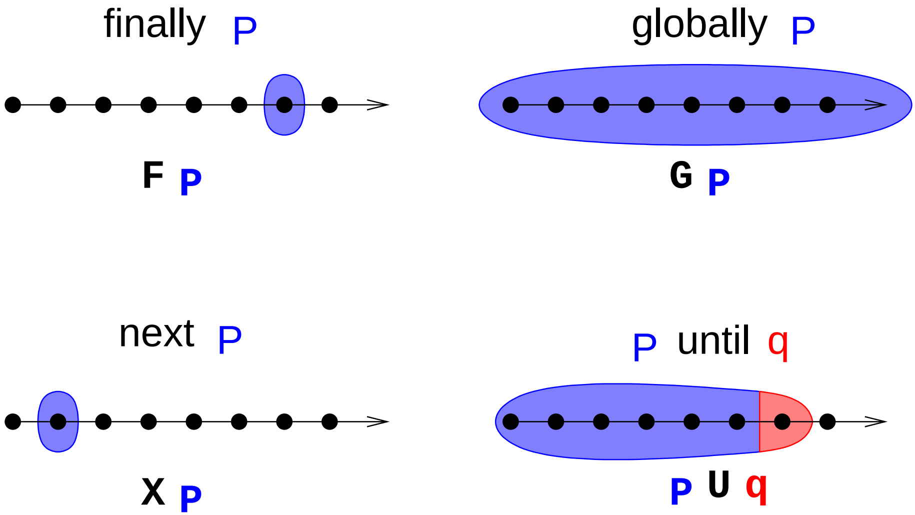
LTL semantics: intuitions

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."

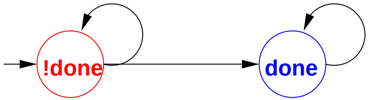
$$\mathbf{GF}\text{send} \rightarrow \mathbf{GF}\text{recv}$$

LTL Formal Semantics

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

LTL Formal Semantics (cont.)

- LTL properties are evaluated over paths, i.e., over infinite, linear sequences of states: $\pi = s_0 \to s_1 \to \cdots \to s_t \to s_{t+1} \to \cdots$
- 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

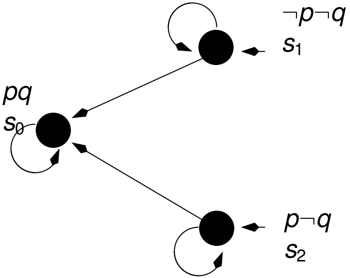
$\mathcal{M} \not\models \phi \not\Longrightarrow \mathcal{M} \models \neg\phi$ (!!)

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

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

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

Note

LTL can be defined in terms of $\wedge$, $\neg$, $\mathbf{X}$, $\mathbf{U}$ only

Exercise

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

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

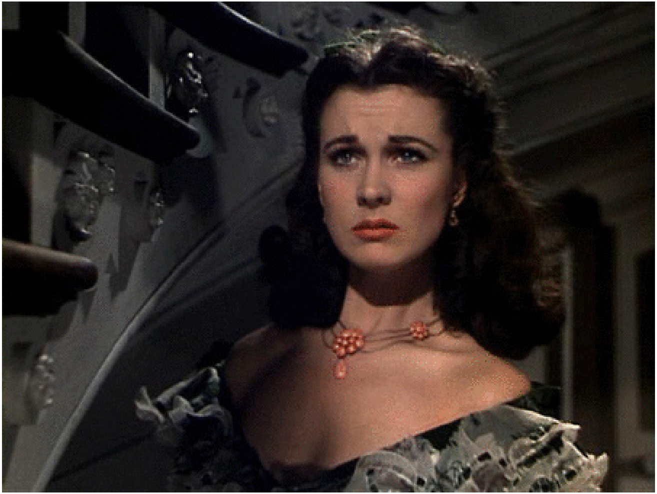
- Let φ_1 and φ_2 be LTL formulae:

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

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

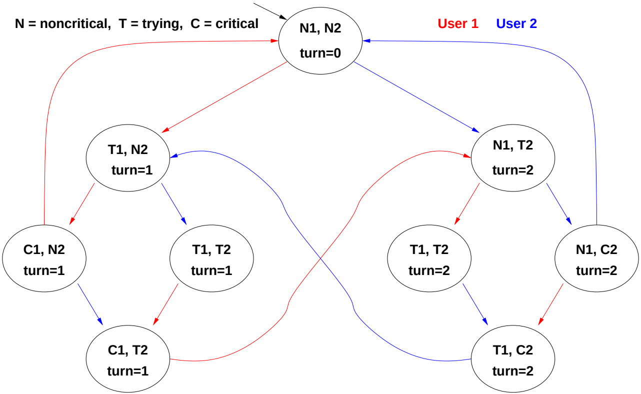
$$(p\mathbf{U}q) \wedge (\mathbf{G}\neg p) \Longrightarrow (q \vee (p \wedge \mathbf{X}(p\mathbf{U}q))) \wedge (\neg p \wedge \mathbf{XG}\neg p)$$

Tableaux rules: a quote



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

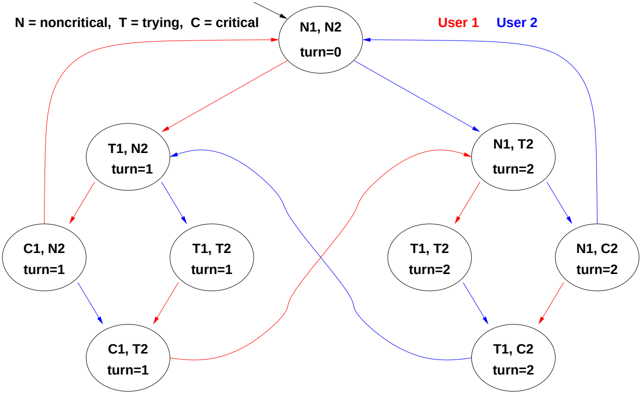
Example 1: mutual exclusion (safety)



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

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

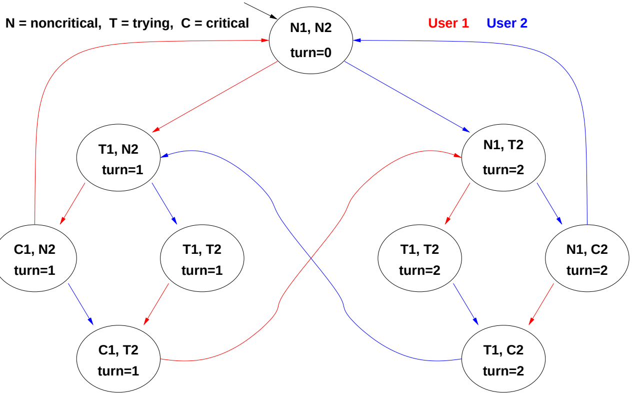
Example 2: liveness



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

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

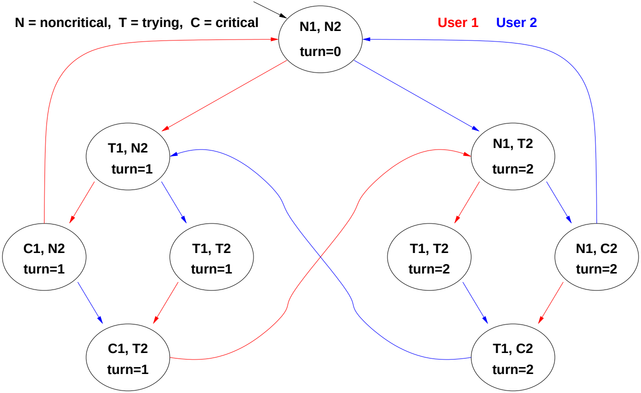
Example 3: liveness



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

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

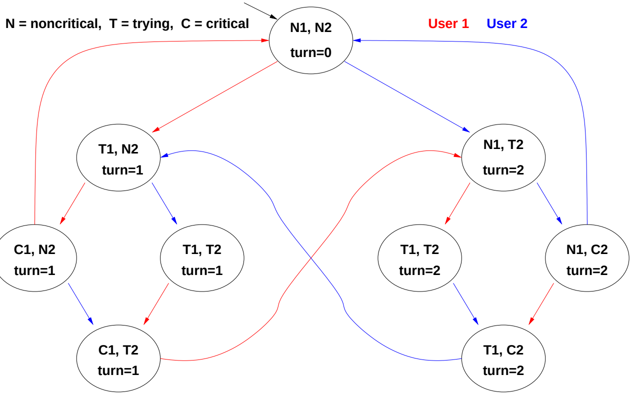
Example 4: fairness



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

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

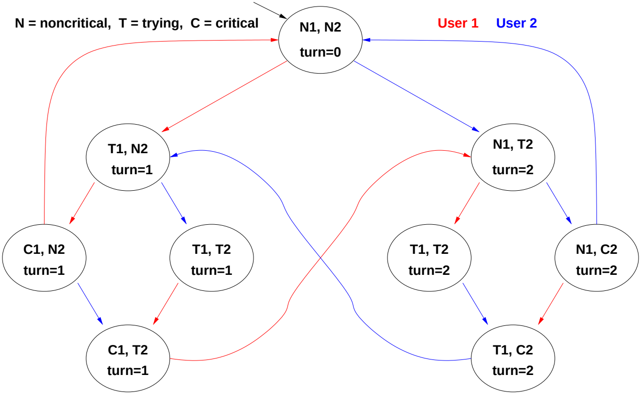
Example 5: strong fairness



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

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

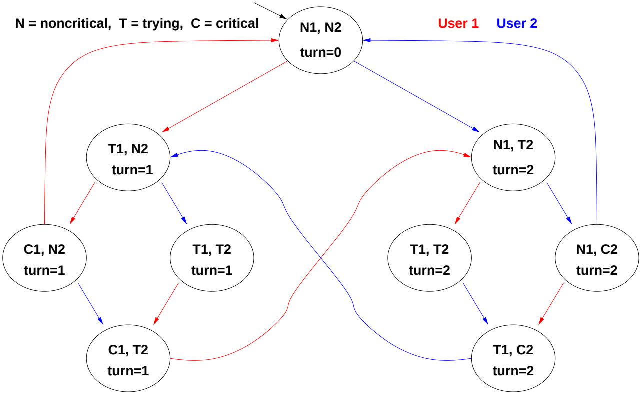
Example 6: Releases



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

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

Example 7: **XF**



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

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. $\mathbf{GF}T \rightarrow \mathbf{GF}C$

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

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

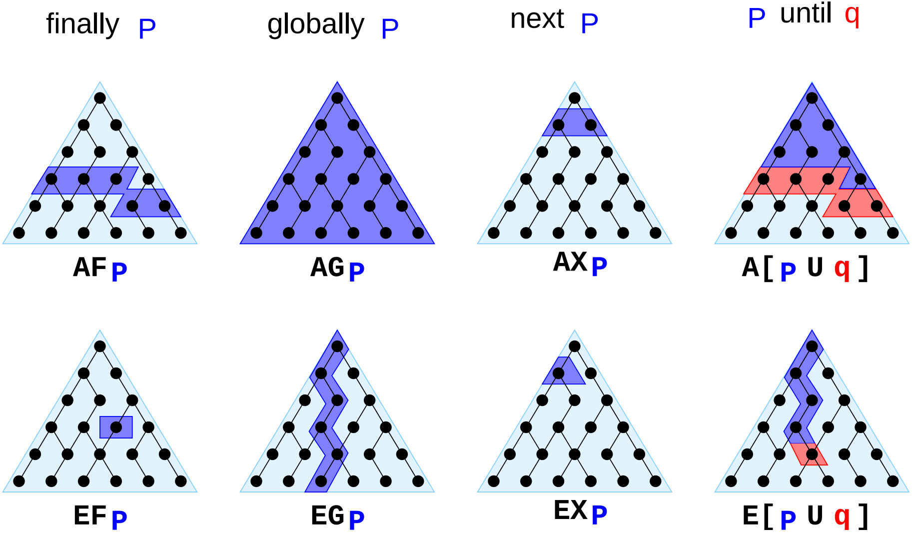
CTL is given by the standard boolean logic enhanced with the operators **AX**, **AG**, **AF**, **AU**, **EX**, **EG**, **EF**, **EU**:

- "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.]



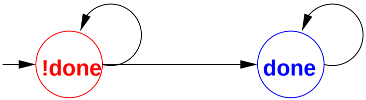
CTL Formal Semantics

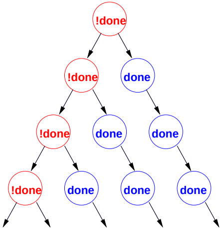
Let $(s_i, s_{i+1}, \ldots)$ be a path outgoing from state s_i in M

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

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

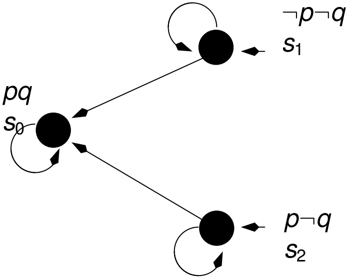
Important Remark

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

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

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



Syntactic properties of CTL operators

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

Note

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

Exercise:

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

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

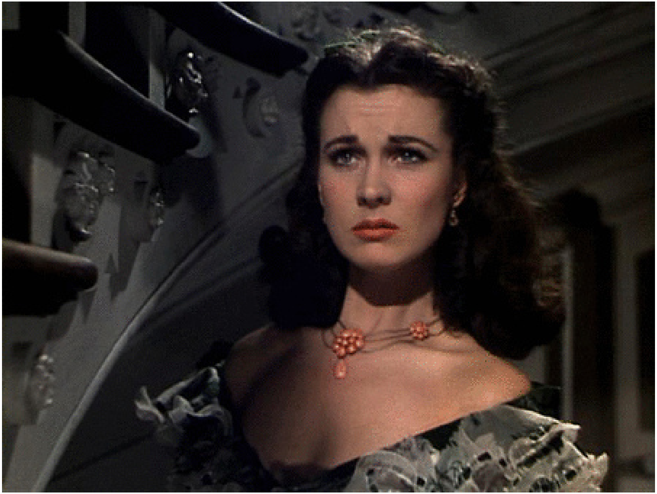
- Let φ_1 and φ_2 be CTL formulae:

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

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

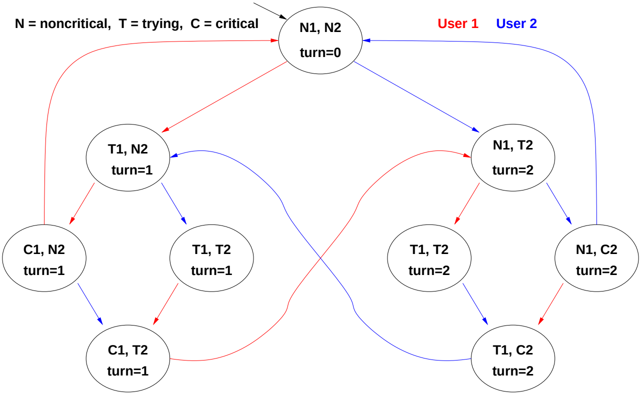
$$\mathbf{A}(p\mathbf{U}q) \wedge (\mathbf{EG}\neg p) \Longrightarrow (q \vee (p \wedge \mathbf{AXA}(p\mathbf{U}q))) \wedge (\neg p \wedge \mathbf{EXEG}\neg p)$$

Tableaux rules: a quote



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

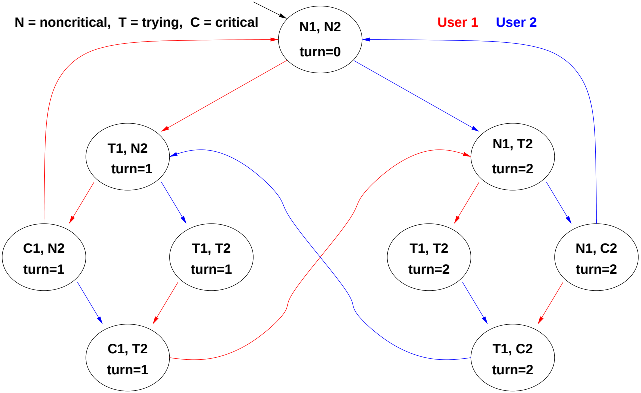
Example 1: mutual exclusion (safety)



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

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.)

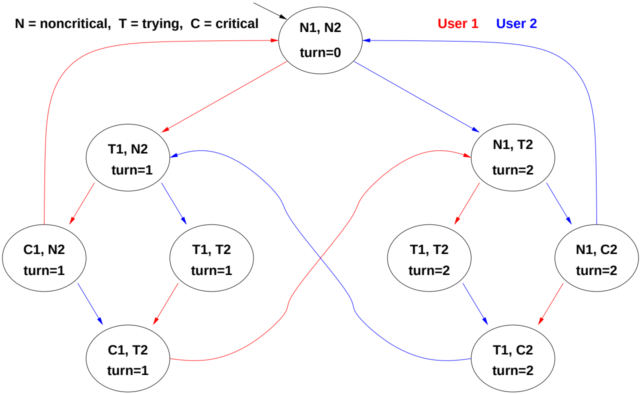
Example 2: liveness



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

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.)

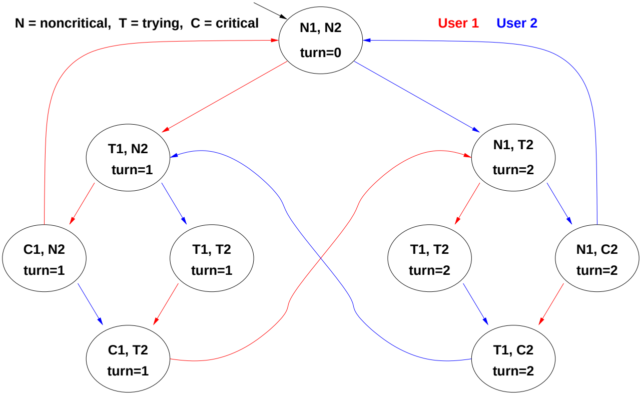
Example 3: fairness



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

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.)

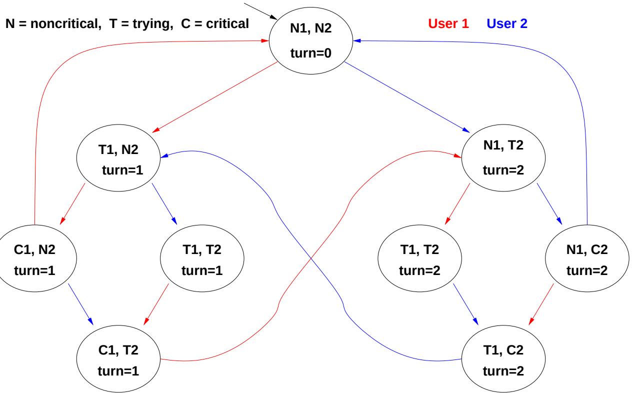
Example 3: fairness (2)



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

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

Example 4: blocking

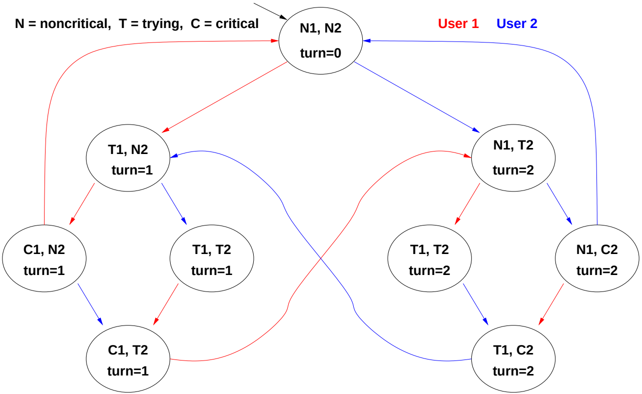


$$M \models \mathbf{AG}(N_1 \rightarrow \mathbf{EF}\, T_1) \ ?$$

YES: from each state where N_1 holds there is a path leading to a state where T_1 holds

(No corresponding LTL formula.)

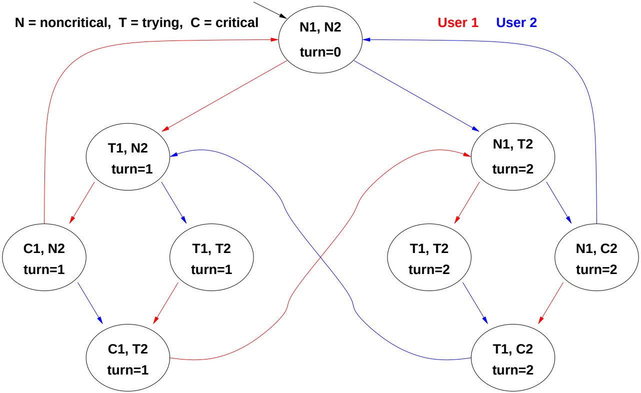
Example 5: blocking (2)



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

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 $\mathbf{G}(N_1 \rightarrow \mathbf{F} T_1)$.)

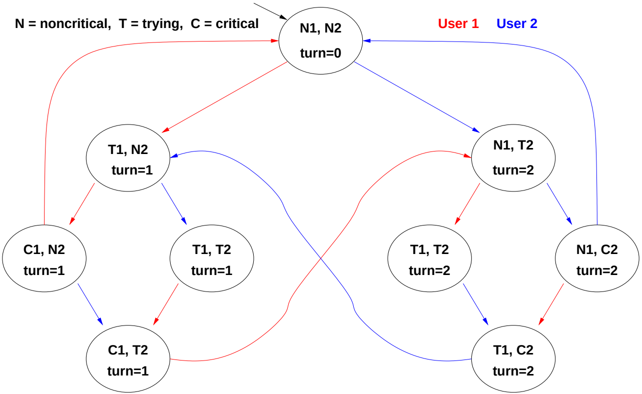
Example 6:



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

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

Example 7:



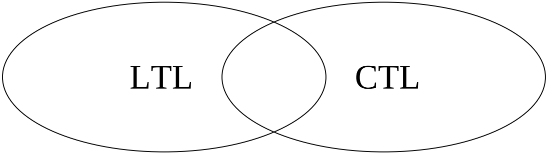
$$M \models \mathbf{AFEG}N_1 \ ?$$

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.)

LTL vs. CTL: expressiveness

- many CTL formulas cannot be expressed in LTL (e.g., those containing existentially quantified subformulas)
  E.g., $\mathbf{AG}(N_1 \rightarrow \mathbf{EF}T_1)$, $AFAG\varphi$
- many LTL formulas cannot be expressed in CTL (e.g. fairness LTL formulas)
  E.g., $\mathbf{GF}T_1 \rightarrow \mathbf{GF}C_1$, $FG\varphi$
- 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., $\mathbf{G}\neg(C_1 \wedge C_2)$, $\mathbf{F}C_1$, $\mathbf{G}(T_1 \rightarrow \mathbf{F}C_1)$, $\mathbf{GF}C_1$



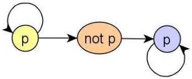
Example: *AFAGp* vs. *FGp*

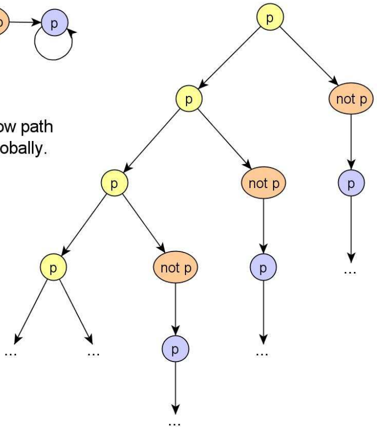
(Example developed by the students Andrea Mattioli and Mirko Boniatti, 2005.)

AFAGp != FGp

Example:

Kripke model ------------> Infinite tree





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

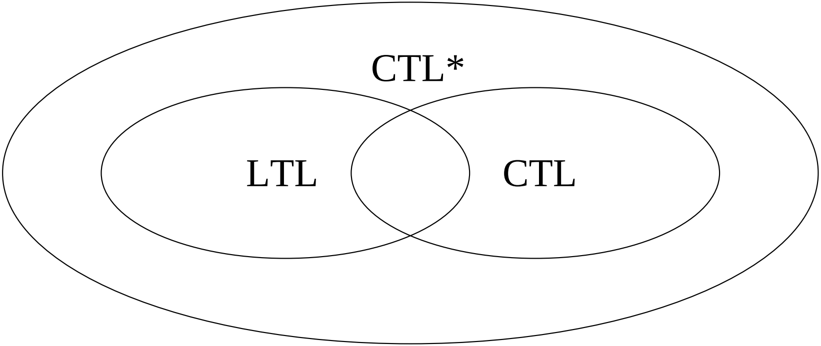
In principle in CTL* one may have sequences of nested path quantifiers. In such case, the most internal one dominates:

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

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")

The need for fairness conditions: intuition

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

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

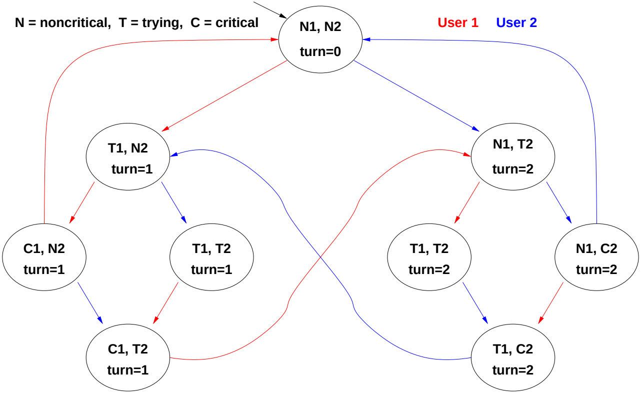
$\Longrightarrow$ it is reasonable enough to assume the protocol suitable under the condition that each user is infinitely often outside the restroom

- such a condition is called fairness condition

The need for fairness conditions: an example

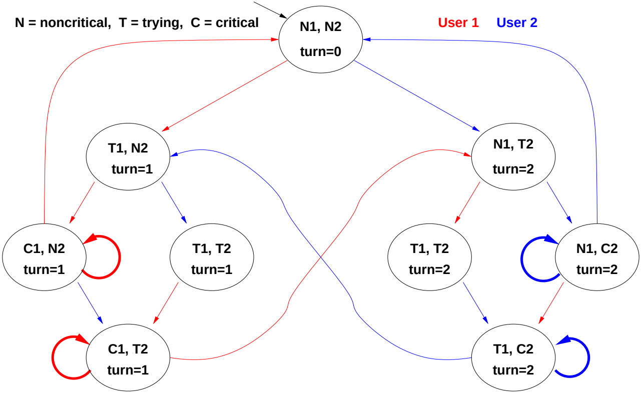
- Consider a variant of the mutual exclusion in which one process can stay permanently in the critical zone
- Do $M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1)$, $M \models \mathbf{AG}(T_2 \rightarrow \mathbf{AF}C_2)$ still hold?

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



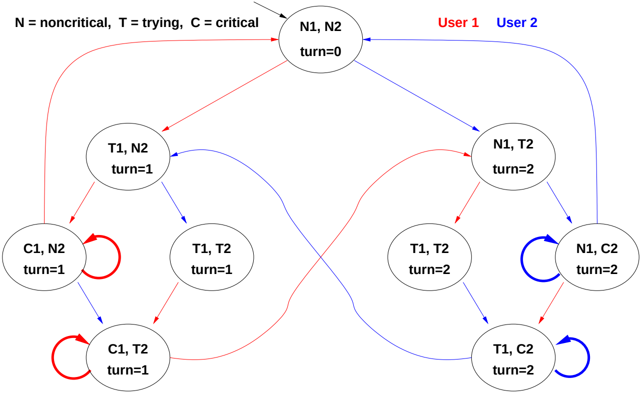
$$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1) \qquad M \models \mathbf{AG}(T_2 \rightarrow \mathbf{AF}C_2)$$

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



$M \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1)?$ $M \models \mathbf{AG}(T_2 \rightarrow \mathbf{AF}C_2)?$

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



$\mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1)$? $\mathbf{AG}(T_2 \rightarrow \mathbf{AF}C_2)$?

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

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

Fairness conditions

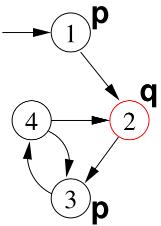
- It is desirable that certain (typically Boolean) conditions φ's hold infinitely often: $\mathbf{AGAF}\varphi$ ($\mathbf{GF}\varphi$ in LTL)
- $\mathbf{AGAF}\varphi$ ($\mathbf{GF}\varphi$) is called fairness conditions
- Intuitively, fairness conditions are used to eliminate behaviours in which a certain condition φ never holds:

$$\neg\mathbf{EFEG}\neg\varphi$$

  ("it is never reached a state from which φ is forever false")
- Example: it is not desirable that, once a process is in the critical section, it never exits: $\mathbf{AGAF}\neg C_1$ ($\neg\mathbf{EFEG}C_1$)
- A fair condition φ_i can be represented also by the set f_i of states where φ_i holds ($f_i := \{s : M, s \models \varphi_i\}$)

Fair Kripke models

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



  E.g., $\{\{2\}\} := \{\{s : M, s \models q\}\} = \{\mathbf{GF}q\}$ is the set of fairness conditions of the Kripke model above

- Fair path π: at least one state for each f_i occurs infinitely often in π (φ_i holds infinitely often in π: $\pi \models \mathbf{GF}\varphi_i$)
  E.g., every path visiting infinitely often state 2 is a fair path.

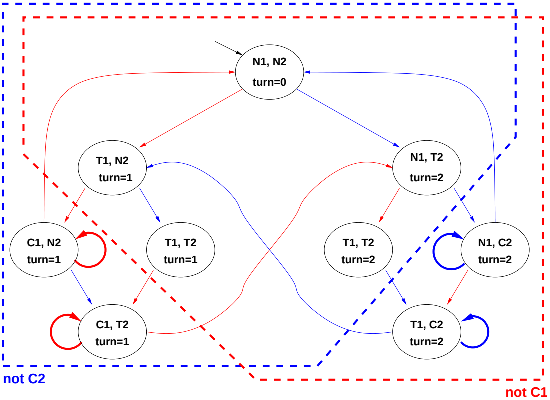
CTL M.C. with Fair Kripke Models

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

- Path quantifiers apply only to fair paths:
  - $M_F \models \mathbf{A}\varphi$ iff $\pi \models \varphi$ for every fair path π
  - $M_F \models \mathbf{E}\varphi$ iff $\pi \models \varphi$ for some fair path π
- Fair state: a state from which at least one fair path originates, that is, a state s is a fair state in M_F iff $M_F, s \models \mathbf{EG}\mathit{true}$.

Fairness: example

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



$M_F \models \mathbf{AG}(T_1 \rightarrow \mathbf{AF}C_1)$? $\qquad$ $M_F \models \mathbf{AG}(T_2 \rightarrow \mathbf{AF}C_2)$?

YES: every fair path satisfies the conditions

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

Remark: fair CTL M.C.

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

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

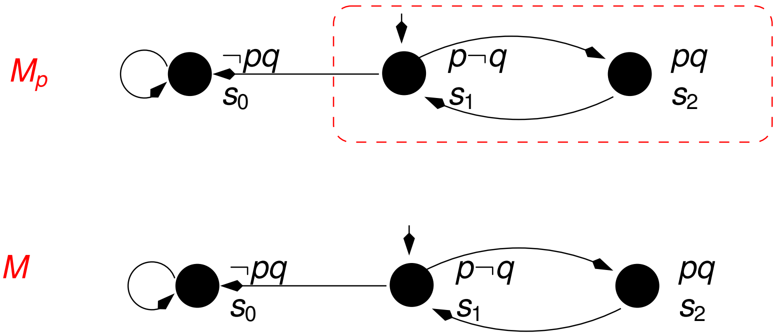
Remark: fair LTL M.C.

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

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

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

[Example provided by the student Davide Kirchner, 2014]



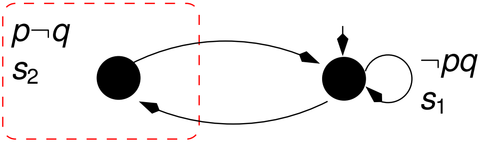
- $M_p \not\models \mathbf{AG}q$
- $M \models (\mathbf{AGAF}p) \rightarrow \mathbf{AG}q$

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

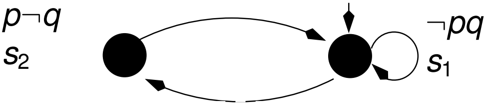
Ex. CTL: $M_{\{f_1,\dots,f_n\}} \models \psi \not\Longleftrightarrow M \models (\bigwedge_{i=1}^n \mathbf{EGEF} f_i) \to \psi$.

[Example provided by the student Daniele Giuliani, 2019]

$M_{p!q}$



M

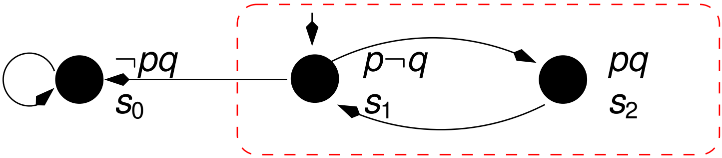


- $M_{p!q} \not\models \mathbf{EFEG} q$
- $M \models (\mathbf{EGEF} p) \to \mathbf{EFEG} q$

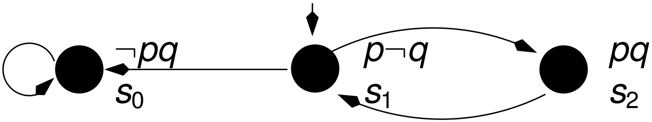
Exercise: show that $M_{\{f_1,\dots,f_n\}} \models \psi \not\Longleftrightarrow M \models (\bigwedge_{i=1}^n \mathbf{EGEF} f_i) \to \psi$.

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

M_p



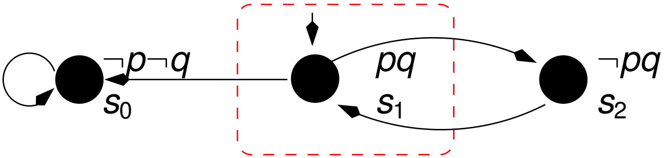
M



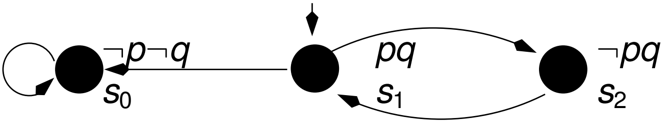
- $M_p \not\models \mathbf{G}q$
- $M \not\models (\mathbf{GF}p) \rightarrow \mathbf{G}q$

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

M_p



M



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

Ex: Labeled CNF-ization

Consider the following Boolean formula φ:

$$((\neg A_1 \wedge \neg A_2) \vee (A_7 \wedge A_4) \vee (\neg A_3 \wedge A_2) \vee (A_5 \wedge \neg A_4))$$

Using the *improved* CNF_{label} conversion, produce the CNF formula $CNF_{label}(\varphi)$.

[Solution: we introduce fresh Boolean variables naming the subformulas of φ:

$$\overbrace{(\overbrace{(\neg A_1 \wedge \neg A_2)}^{B_1} \vee \overbrace{(A_7 \wedge A_4)}^{B_2} \vee \overbrace{(\neg A_3 \wedge A_2)}^{B_3} \vee \overbrace{(A_5 \wedge \neg A_4)}^{B_4})}^{B}$$

from which we obtain:

$$\begin{array}{ll} (B) & \wedge \\ (\neg B \vee B_1 \vee B_2 \vee B_3 \vee B_4) & \wedge \\ (\neg B_1 \vee \neg A_1) \wedge (\neg B_1 \vee \neg A_2) & \wedge \\ (\neg B_2 \vee A_7) \wedge (\neg B_2 \vee A_4) & \wedge \\ (\neg B_3 \vee \neg A_3) \wedge (\neg B_3 \vee A_2) & \wedge \\ (\neg B_4 \vee A_5) \wedge (\neg B_4 \vee \neg A_4) & \end{array} .$$

Ex: NNF conversion

Consider the following Boolean formula φ:

$$\neg(((\neg A_1 \to \neg A_2) \wedge (\neg A_3 \to A_4)) \vee ((A_5 \to A_6) \wedge (A_7 \to \neg A_8)))$$

Compute the Negative Normal Form of φ, called φ'.

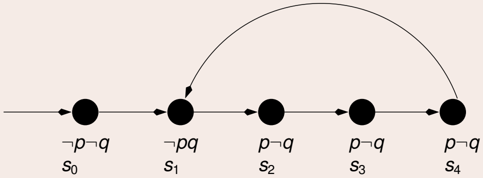
[Solution:

$$\begin{array}{ll}
 & \varphi \\
\Rightarrow & \neg(((\neg A_1 \to \neg A_2) \wedge (\neg A_3 \to A_4)) \vee ((A_5 \to A_6) \wedge (A_7 \to \neg A_8))) \\
\Rightarrow & (\neg((\neg A_1 \to \neg A_2) \wedge (\neg A_3 \to A_4)) \wedge \neg((A_5 \to A_6) \wedge (A_7 \to \neg A_8))) \\
\Rightarrow & ((\neg(\neg A_1 \to \neg A_2) \vee \neg(\neg A_3 \to A_4)) \wedge (\neg(A_5 \to A_6) \vee \neg(A_7 \to \neg A_8))) \\
\Rightarrow & (((\neg A_1 \wedge A_2) \vee (\neg A_3 \wedge \neg A_4)) \wedge ((A_5 \wedge \neg A_6) \vee (A_7 \wedge A_8))) \\
= & \varphi'
\end{array}$$

]

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]

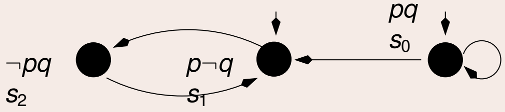
(b) $\pi, s_0 \models \mathbf{FG}(q \leftrightarrow \neg p)$
[Solution: true]

(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]

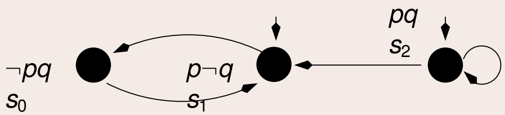
(b) $M \models \mathbf{G}(\neg p \rightarrow F\neg q)$
[Solution: true]

(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]

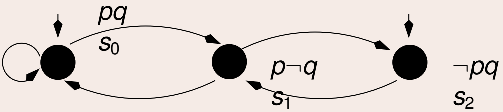
(b) $M \models \mathbf{EG}p$
[Solution: false]

(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]

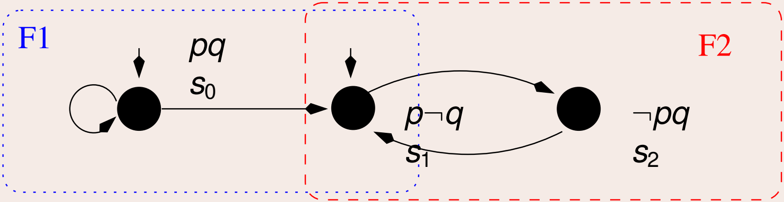
(b) $M \models \mathbf{EG}q$
[Solution: false]

(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]

Ex: Fair CTL Model Checking

Consider the following *fair* Kripke Model M:



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

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

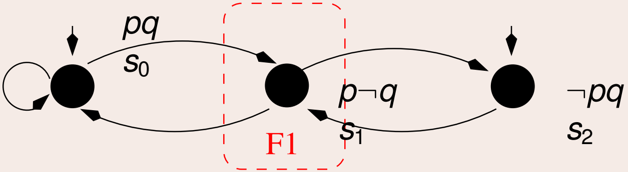
(b) $M \models \mathbf{A}(p\mathbf{U}\neg q)$
[Solution: true]

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

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

Ex: Fair CTL Model Checking

Consider the following *fair* Kripke Model M:



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

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

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

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

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