Computer aided verification

Lecture 2:

Expressing properties of systems in Linear Temporal Logic (LTL)

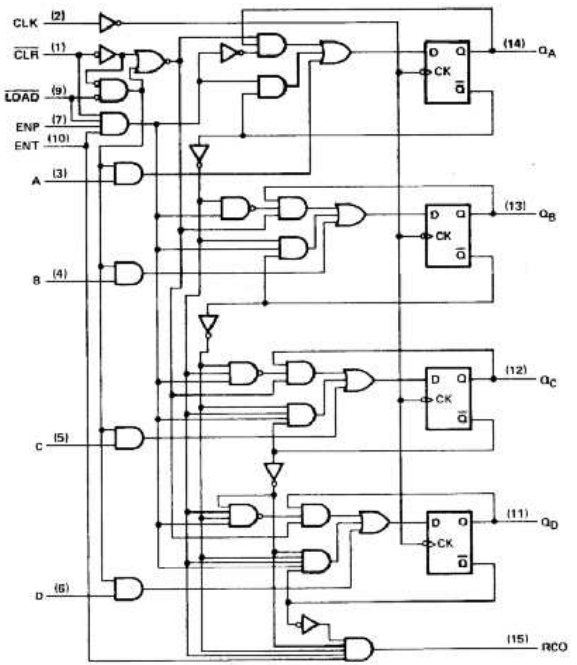
Sławomir Lasota
University of Warsaw

Plan

- Modeling systems as Kripke structures
- LTL
- Expressivity
- Model checking

Modeling systems as Kripke structures

```
01523    Dim lQuoteCount As Long
01524    Dim lcount      As Long
01525    Dim sChar       As String
01526    Dim sPrevChar   As String
01527
01528    ' Starts with Rem it is a comment
01529    sLine = Trim(sLine)
01530    If Left(sLine, 3) = "Rem" Then
01531       CleanUpLine = ""
01532       Exit Function
01533    End If
01534
01535    ' Starts with ' it is a comment
01536    If Left(sLine, 1) = "'" Then
01537       CleanUpLine = ""
01538       Exit Function
01539    End If
01540
01541    ' Contains ' may end in a comment, so test if it is a comment or in the
01542    ' body of a string
01543    If InStr(sLine, " '") > 0 Then
01544       sPrevChar = " "
01545       lQuoteCount = 0
01546
01547       For lcount = 1 To Len(sLine)
01548          sChar = Mid(sLine, lcount, 1)
01549
01550          ' If we found " '" then an even number of " characters in front
01551          ' means it is the start of a comment, and odd number means it is
01552          ' part of a string
01553          If sChar = "'" And sPrevChar = " " Then
01554             If lQuoteCount Mod 2 = 0 Then
01555                sLine = Trim(Left(sLine, lcount - 1))
01556                Exit For
01557             End If
01558          ElseIf sChar = """" Then
01559             lQuoteCount = lQuoteCount + 1
01560          End If
01561          sPrevChar = sChar
01562       Next lcount
01563    End If
01564
```



Kripke structure (model)

Def.: Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, P - atomic properties (propositional variables)

Kripke structure (model)

Def.: Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, P - atomic properties (propositional variables)

Often we assume that $\rightarrow$ is total: no deadlock!

$$\forall s \in S.\ \exists s' \in S.\ s \rightarrow s'$$

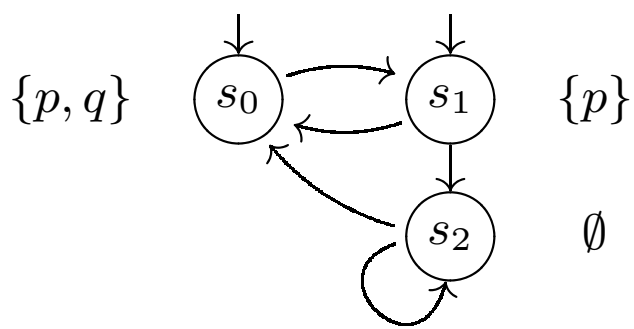
Kripke structure (model)

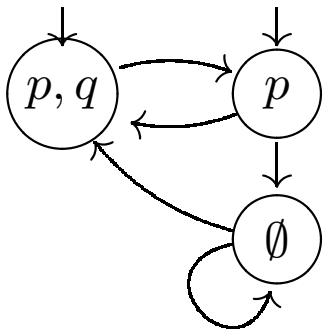
Def.: Kripke structure $M = \langle S, S_{\text{init}}, \rightarrow, L \rangle$

- $S_{\text{init}} \subseteq S$ nonempty set of initial states
- $\rightarrow \subseteq S \times S$ transition relation
- $L : S \rightarrow \mathcal{P}(P)$, P - atomic properties (propositional variables)

Often we assume that $\rightarrow$ is total: no deadlock!

$$\forall s \in S.\ \exists s' \in S.\ s \rightarrow s'$$





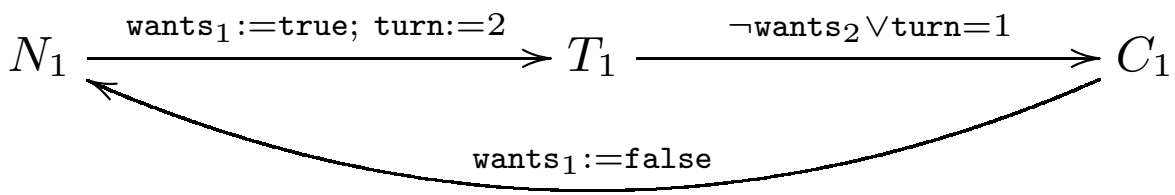
Abstraction: program $\mapsto$ Kripke structure

```
Proc1

private:                              N1
       wants1 := true;
       turn := 2;

       /* attempts */                T1

       (!wants2 ∨ turn = 1);

       /* critical section */        C1

       wants1 := false;
       goto private
```

Abstraction: program $\mapsto$ Kripke structure

```
Proc1

private:                              N1
       wants1 := true;
       turn := 2;

       /* attempts */                T1

       (!wants2 ∨ turn = 1);

       /* critical section */        C1

       wants1 := false;
       goto private
```

Proc$_1$:



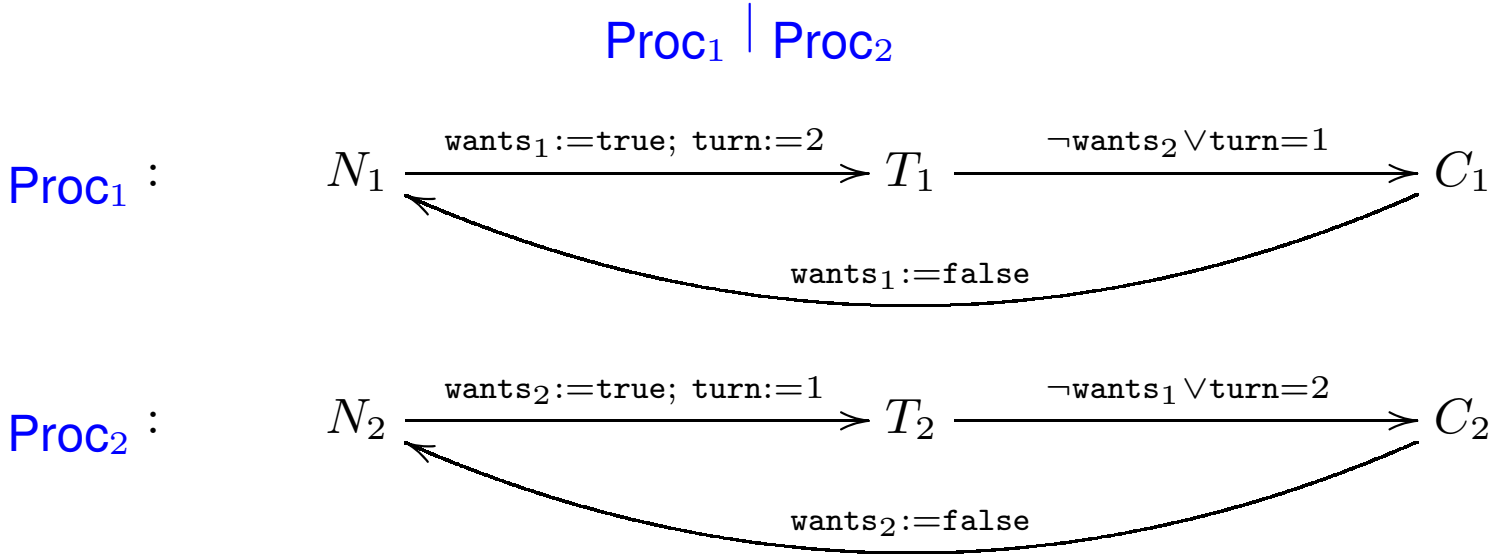
Abstraction: program $\mapsto$ Kripke structure

```
Proc1

private:
        wants1 := true;
        turn := 2;

        /* attempts */

        (!wants2 ∨ turn = 1);

        /* critical section */

        wants1 := false;
        goto private
```

```
Proc2

private:
        wants2 := true;
        turn := 1;

        /* attempts */

        (!wants1 ∨ turn = 2);

        /* critical section */

        wants2 := false;
        goto private
```

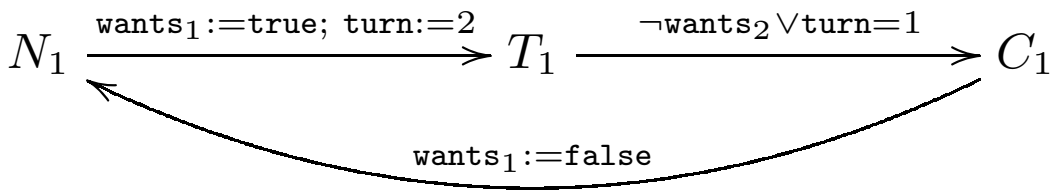
Abstraction: program $\mapsto$ Kripke structure

- N_i private section
- T_i attempt to enter critical section
- C_i critical section

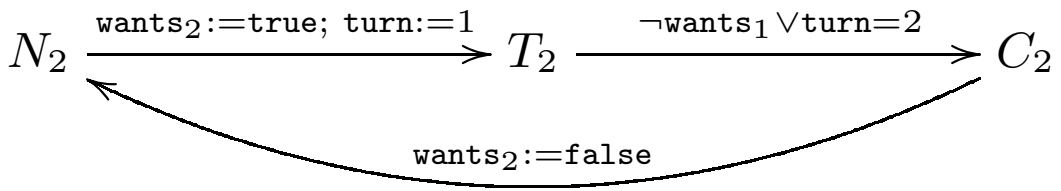


Abstraction: program $\mapsto$ Kripke structure

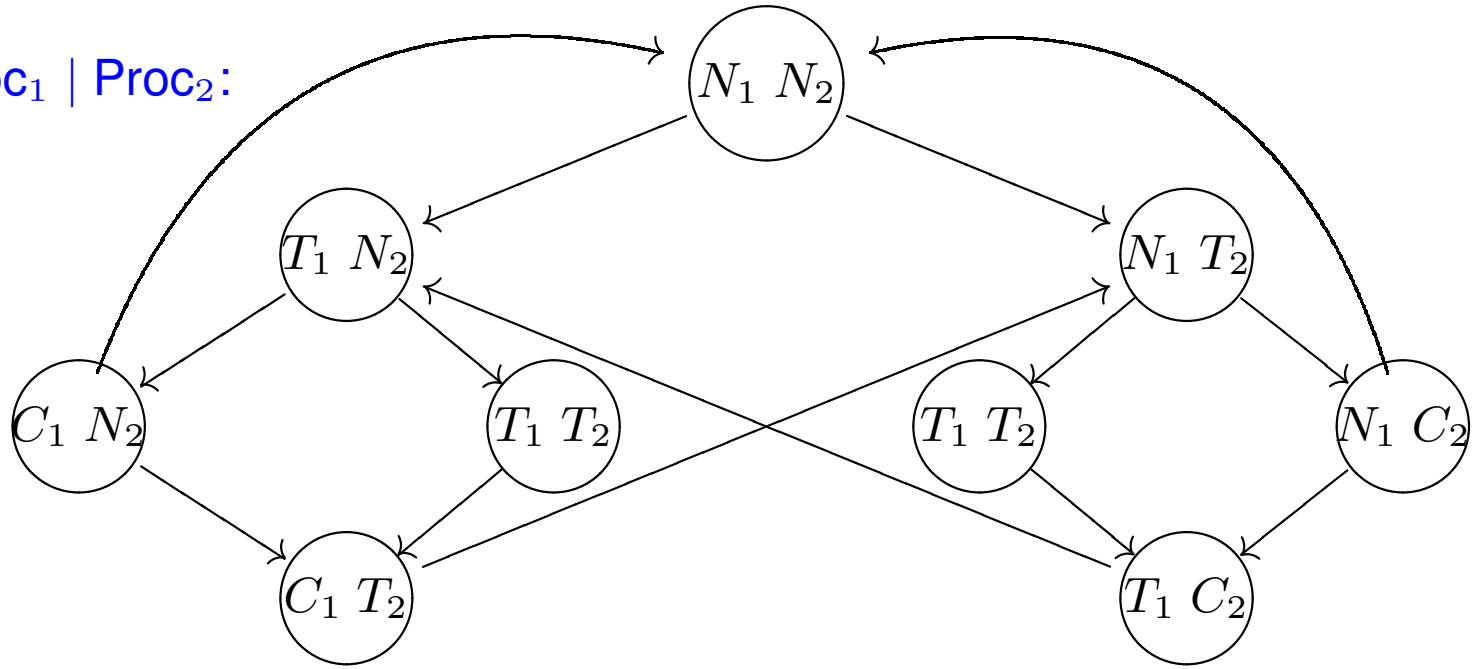
Proc_1 :



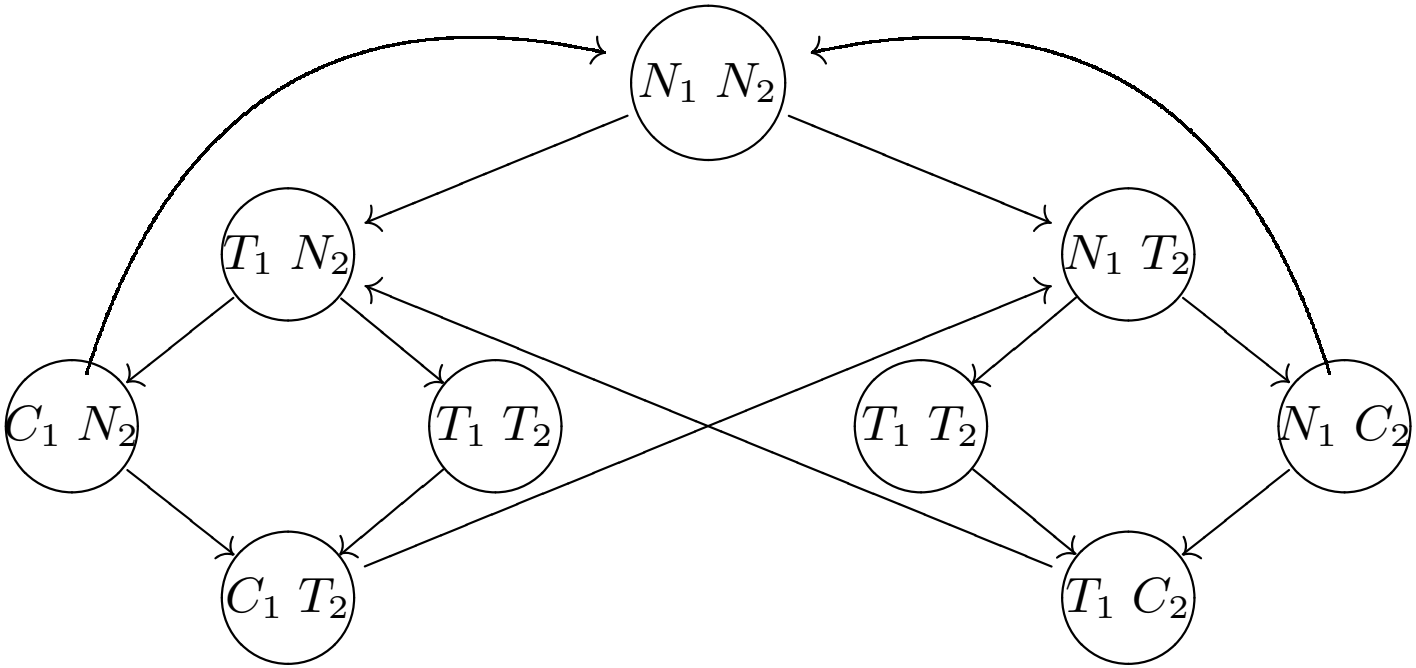
Proc_2 :



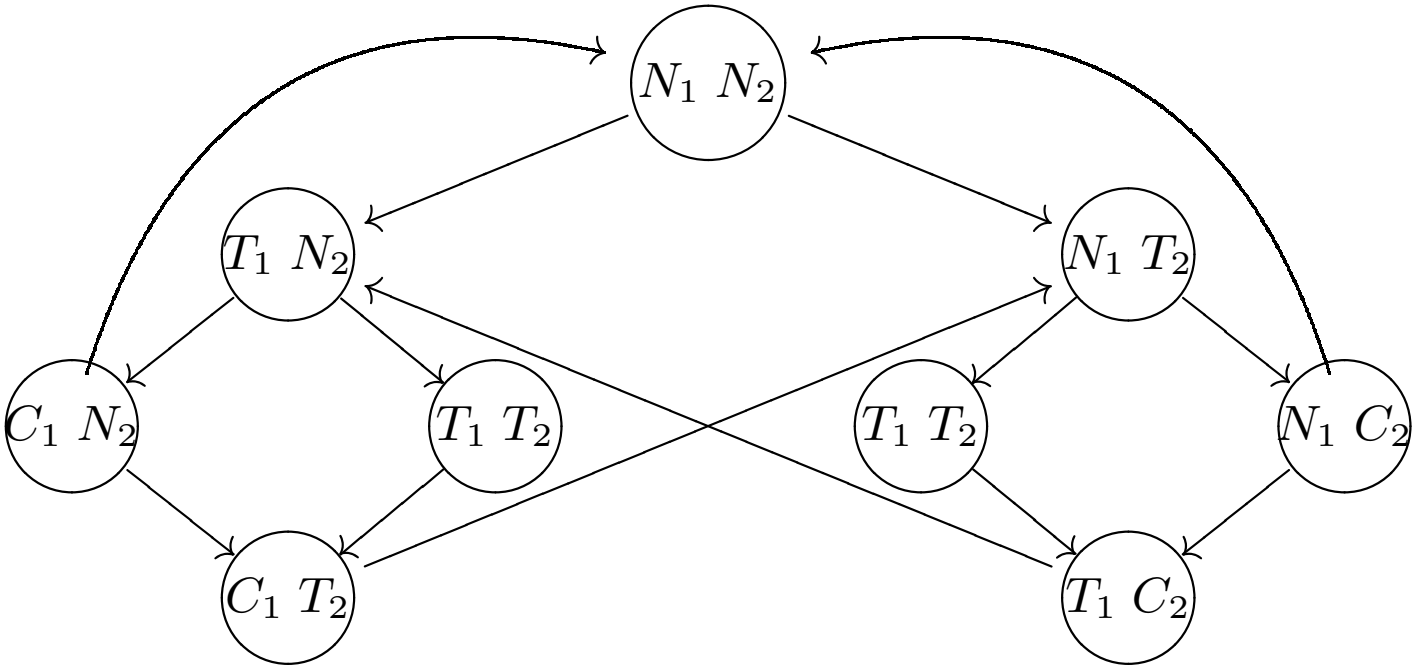
$\text{Proc}_1 \mid \text{Proc}_2$:



Abstraction: program $\mapsto$ Kripke structure

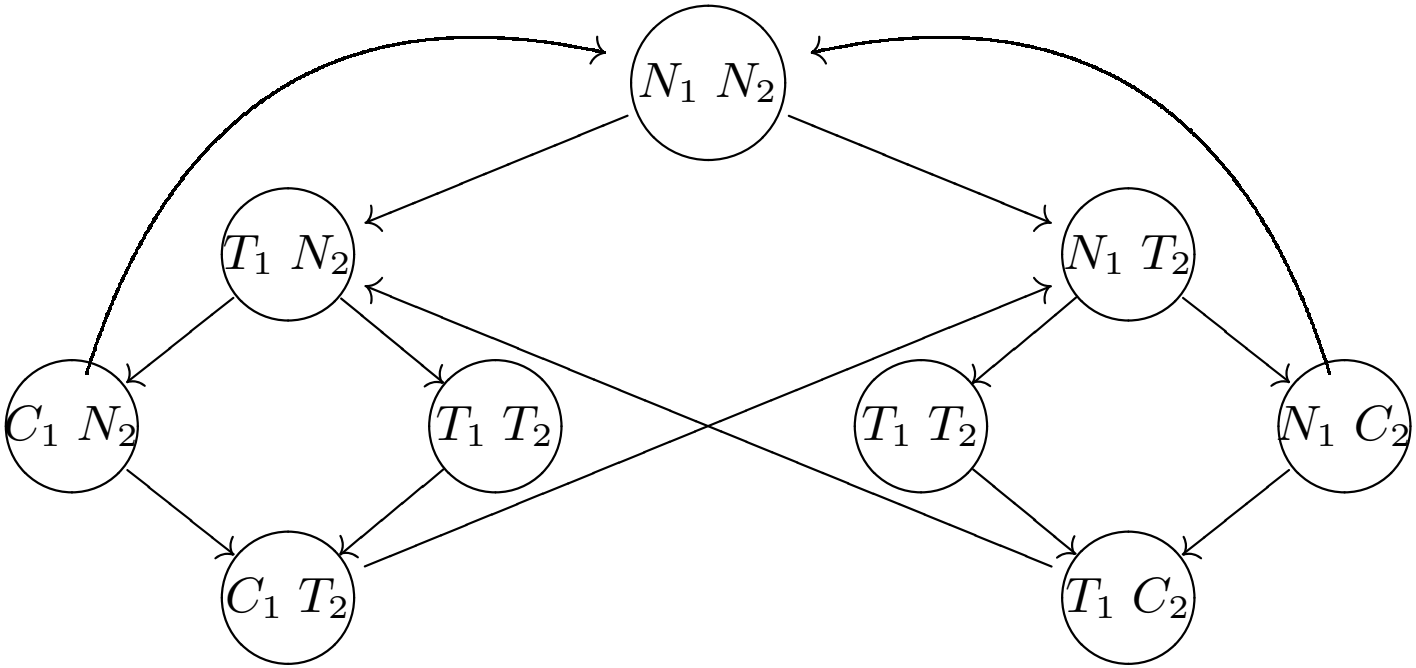


Abstraction: program $\mapsto$ Kripke structure

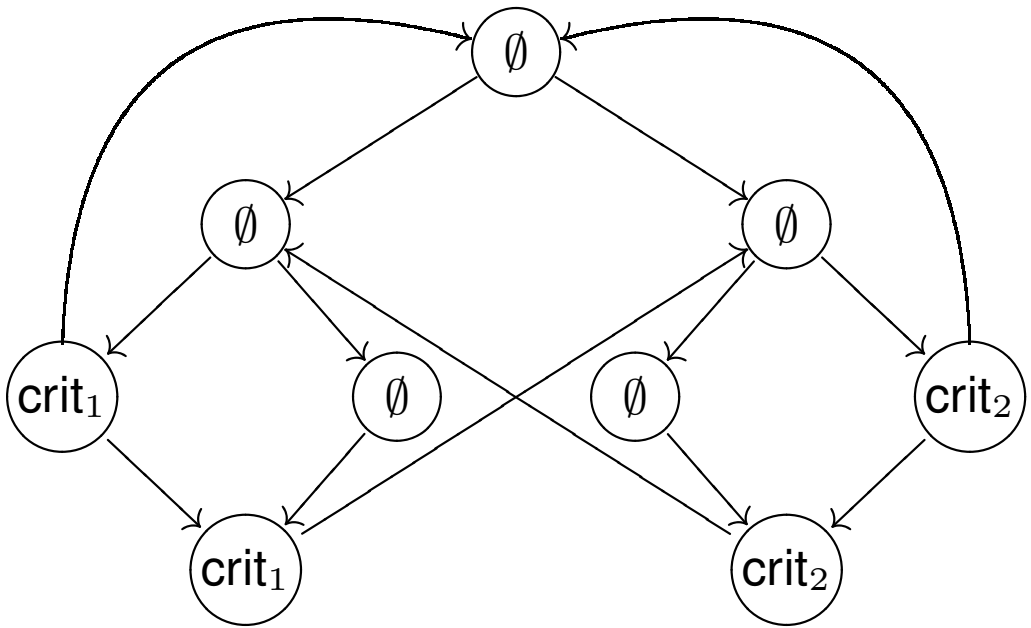


$P = \{\mathsf{crit}_1, \mathsf{crit}_2\}$

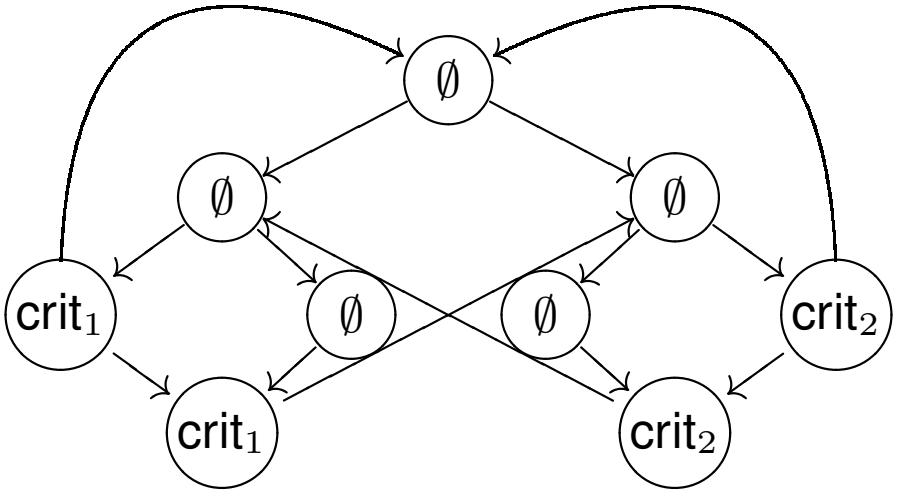
Abstraction: program $\mapsto$ Kripke structure

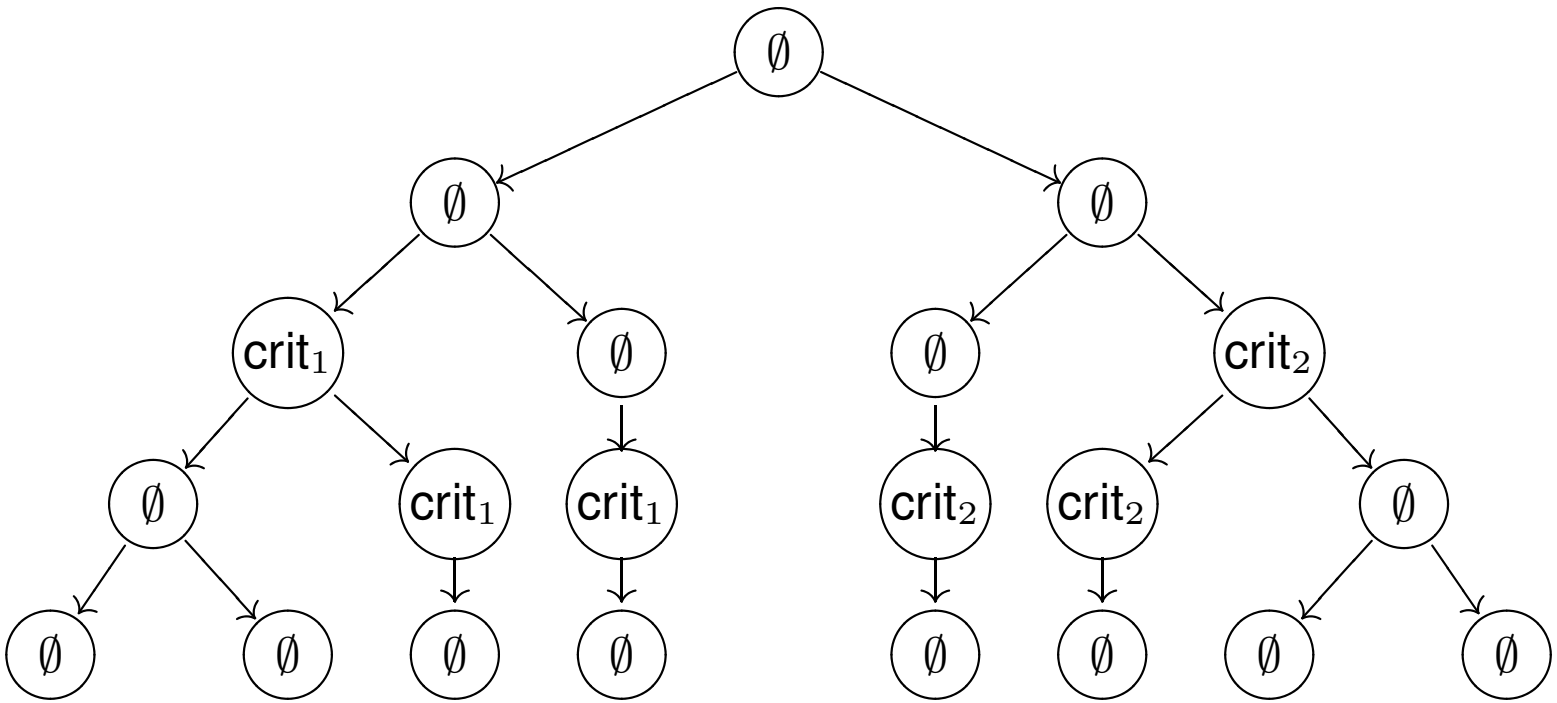


$P = \{\mathsf{crit}_1, \mathsf{crit}_2\}$



Kripke structure $\mapsto$ tree

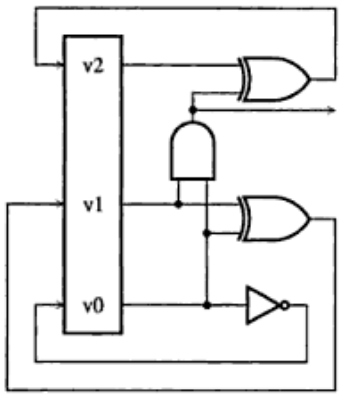




. . .

Hardware

$$
\begin{aligned}
v_0' &:= \neg v_0 \\
v_1' &:= v_0 \oplus v_1 \\
v_2' &:= (v_0 \wedge v_1) \oplus v_2
\end{aligned}
$$



Question: What is the induced Kripke structure?

Paths

Def.: Path (run) is a maximal sequence

$$\Pi = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots$$

Notation: $|\Pi|$ – number of states in Π

Paths

Def.: Path (run) is a maximal sequence

$$\Pi = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots$$

Notation: $|\Pi|$ – number of states in Π

LTL says about paths.

In a Kripke structure M, formula $\phi \in$ LTL is interpreted as follows:
for every path Π such that $s_0 \in S_{\text{init}}$, ϕ holds.

Paths

Def.: Path (run) is a maximal sequence

$$\Pi = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots$$

Notation: $|\Pi|$ – number of states in Π

LTL says about paths.

In a Kripke structure M, formula $\phi \in$ LTL is interpreted as follows:
for every path Π such that $s_0 \in S_{\text{init}}$, ϕ holds.

Notation: $M \models \phi$, $\Pi \models \phi$

LTL

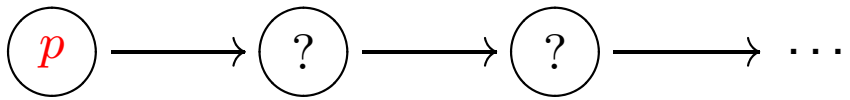
Def.: LTL (Linear Temporal Logic)

$$\phi \quad := \quad p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2$$

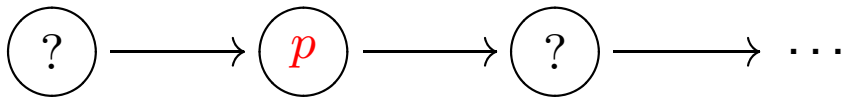
Def.: LTL (Linear Temporal Logic)

$$\phi \quad := \quad p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2$$

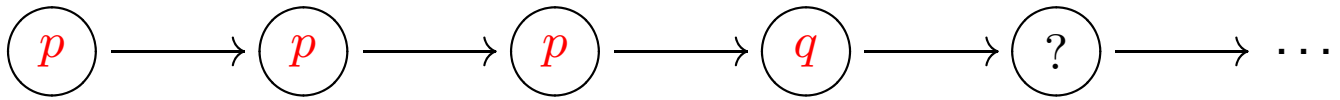
p



$\mathsf{X}\,p$



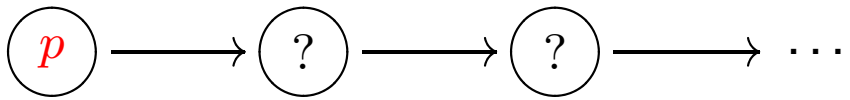
$p \,\mathsf{U}\, q$



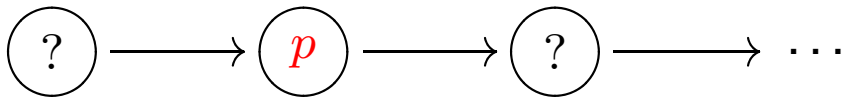
Def.: LTL (Linear Temporal Logic)

$$\phi \;:=\; p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{X}\,\phi \mid \phi_1 \,\mathsf{U}\, \phi_2$$

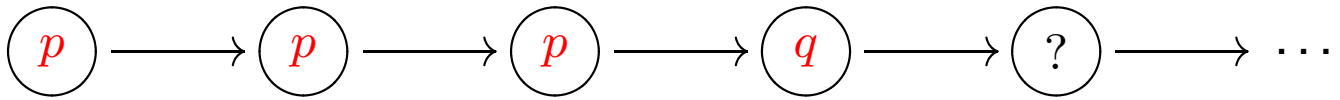
p



$\mathsf{X}\,p$



$p\,\mathsf{U}\,q$

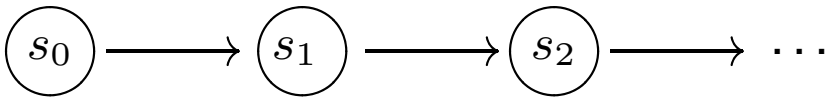


Example:

$$\neg\texttt{starts}\ \mathsf{U}\ \texttt{key}, \qquad \neg\texttt{starts}\ \mathsf{U}\ \neg\texttt{starts} \wedge \texttt{key}$$

LTL – semantics

Semantics:

$\Pi \; = \;$ 

$\Pi \models p$ iff $p \in L(s_0)$

$\Pi \models \neg\phi$ iff $\Pi \not\models \phi$

$\Pi \models \phi_1 \wedge \phi_2$ iff $\Pi \models \phi_1$ and $\Pi \models \phi_2$

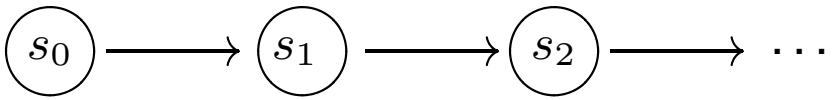
$\Pi \models \mathsf{X}\,\phi$ iff $\Pi^1 \models \phi$, where $\Pi^i = s_i \rightarrow s_{i+1} \rightarrow s_{i+2} \rightarrow \ldots$

$\Pi \models \phi_1 \,\mathsf{U}\, \phi_2$ iff $\exists i < |\Pi|.\ \Pi^i \models \phi_2 \ \wedge\ \forall j < i.\ \Pi^j \models \phi_1$

$M \models \phi$ iff $\Pi \models \phi$ for all paths Π starting in an initial state

LTL – semantics

Alternative semantics:

$\Pi \;=\;$ 

$\Pi \models \phi$ iff $\Pi, 0 \models \phi$

$\Pi, n \models p$ iff $p \in L(s_n)$

$\Pi, n \models \neg\phi$ iff $\Pi, n \not\models \phi$

$\Pi, n \models \phi_1 \wedge \phi_2$ iff $\Pi, n \models \phi_1$ and $\Pi, n \models \phi_2$

$\Pi, n \models \mathsf{X}\,\phi$ iff $\Pi, n+1 \models \phi$

$\Pi, n \models \phi_1 \,\mathsf{U}\, \phi_2$ iff $\exists i \geq n.\ \Pi, i \models \phi_2 \ \wedge\ \forall j \in [n \ldots i-1].\ \Pi, j \models \phi_1$

$M \models \phi$ iff $\Pi \models \phi$ for all paths Π starting in an initial state

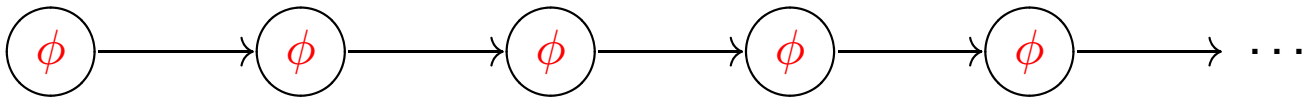
Linear time

An LTL formula essentially specifies a subset of $\mathcal{P}(P)^\omega$.

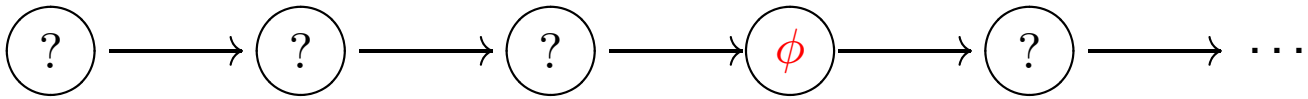
LTL - always, finally

Question: How to write

always ϕ



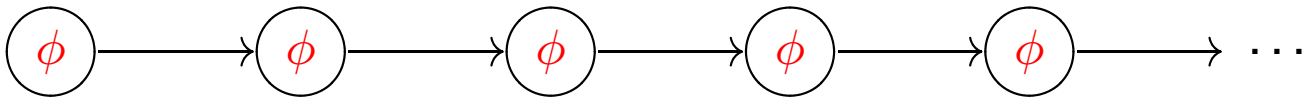
eventually ϕ



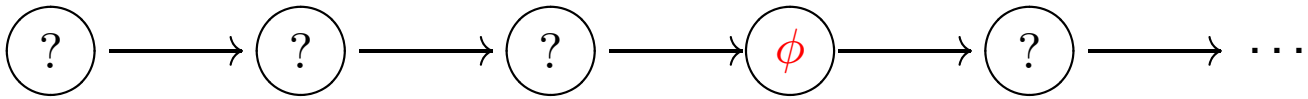
LTL - always, finally

Question: How to write

always ϕ



eventually ϕ



Notation:

$$\mathsf{F}\,\phi \equiv \mathtt{true}\,\mathsf{U}\,\phi$$

$$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$$

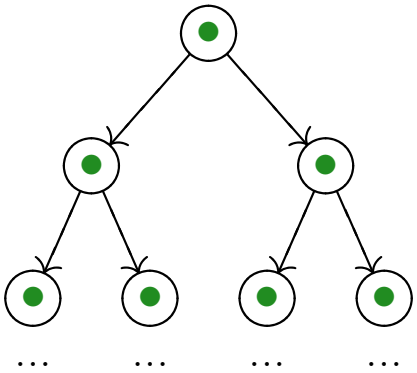
$$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$$

$$\mathsf{F}\,\phi \equiv \Diamond\phi$$

$$\mathsf{G}\,\phi \equiv \Box\phi$$

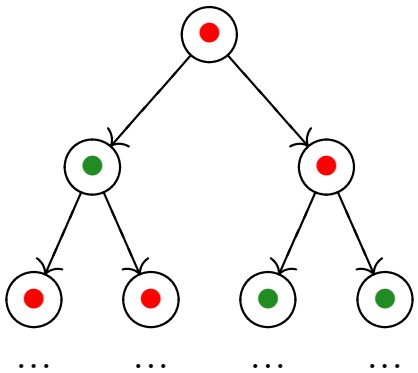
$$\mathsf{X}\,\phi \equiv \bigcirc\phi$$

Typical properties



safety

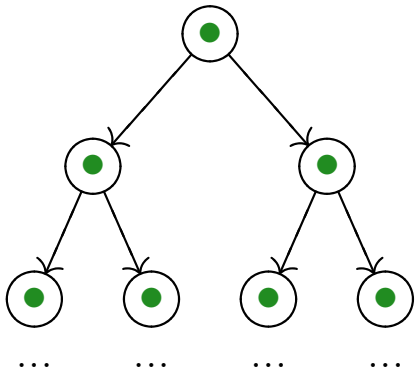
Bad will never happen



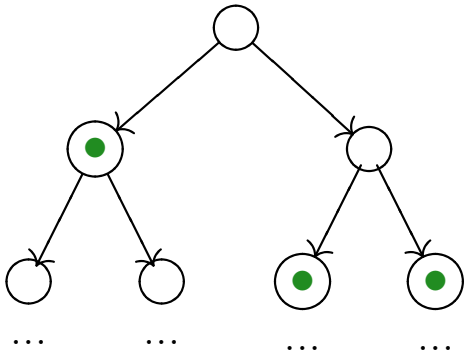
liveness

Good will eventually happen

Typical properties

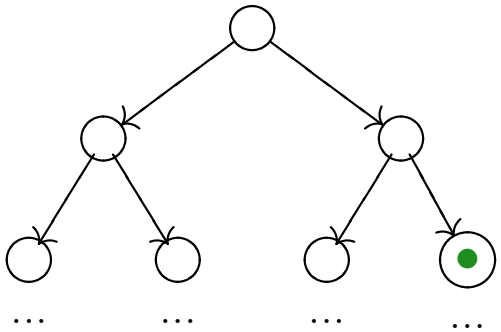


safety



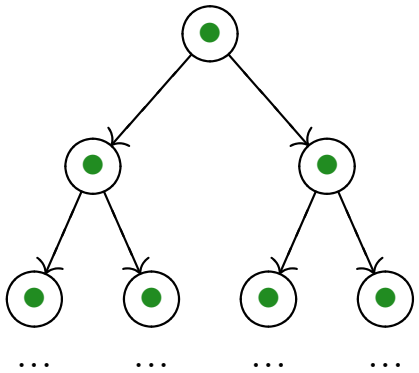
liveness

?



possibility

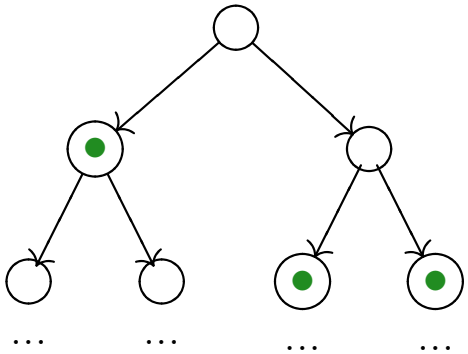
Typical properties



safety

$\mathsf{G}\,\phi$

$\mathsf{G}\,\neg(\mathtt{cr}_1 \wedge \mathtt{cr}_2)$

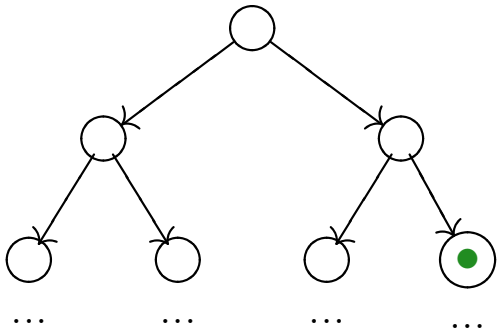


liveness

?

$\mathsf{F}\,\phi$

$\mathsf{F}\,\mathtt{granted}$



possibility

$\mathsf{G}\,\neg\phi$

$\neg\,\mathsf{G}\,\neg\phi$

$\mathsf{G}\,\neg\mathtt{occ}$

Classification of properties

Def.: Property = subset of $\mathcal{P}(P)^\omega$

Safety properties X

negative decision always after finitely many steps

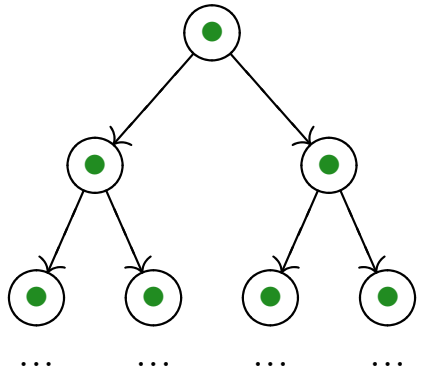
Classification of properties

Def.: Property = subset of $\mathcal{P}(P)^\omega$

Safety properties X

negative decision always after finitely many steps

if $\pi \notin X$ then there is a prefix $\rho < \pi$ such that $\rho < \pi'$ implies $\pi' \notin X$



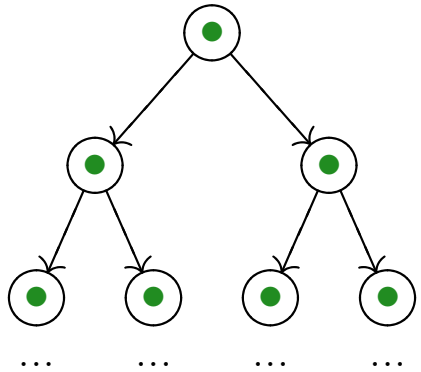
Classification of properties

Def.: Property = subset of $\mathcal{P}(P)^\omega$

Safety properties X

negative decision always after finitely many steps

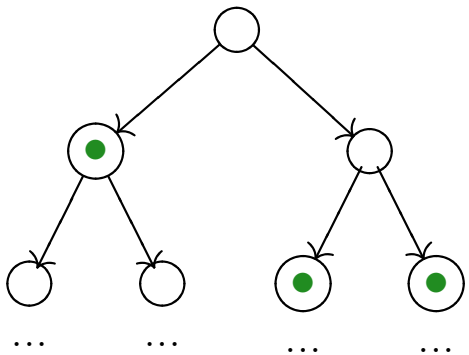
if $\pi \notin X$ then there is a prefix $\rho < \pi$ such that $\rho < \pi'$ implies $\pi' \notin X$



Liveness properties X

negative decision never after finitely many steps

for every ρ exists $\pi > \rho$ such that $\pi \in X$



Example properties

- infinitely often ϕ ?
- almost always ϕ ?
- "weak" $\phi_1 \,\mathsf{U}\, \phi_2$: ϕ_2 not necessarily ?
- if `req` then `granted` later ?

Example properties

- infinitely often ϕ — $\mathsf{G}\,\mathsf{F}\,\phi$
- almost always ϕ — $\mathsf{F}\,\mathsf{G}\,\phi$
- "weak" $\phi_1\,\mathsf{U}\,\phi_2$: ϕ_2 not necessarily — $\mathsf{G}\,\phi_1 \vee \phi_1\,\mathsf{U}\,\phi_2$
- if `req` then `granted` later

  $\texttt{req} \implies \mathsf{X}\,\mathsf{F}\,\texttt{granted}$

- fairness: if stubbornly `req` then `granted`

  "weak": stubbornly = almost always — ?

  "strong": stubbornly = infinitely often — ?

Example properties

- infinitely often ϕ — $\mathsf{G}\,\mathsf{F}\,\phi$
- almost always ϕ — $\mathsf{F}\,\mathsf{G}\,\phi$
- "weak" $\phi_1\,\mathsf{U}\,\phi_2$: ϕ_2 not necessarily — $\mathsf{G}\,\phi_1 \vee \phi_1\,\mathsf{U}\,\phi_2$
- if `req` then `granted` later

  $\texttt{req} \implies \mathsf{X}\,\mathsf{F}\,\texttt{granted}$

- fairness: if stubbornly `req` then `granted`

  "weak": stubbornly = almost always — $\mathsf{F}\,\mathsf{G}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted}$

  "strong": stubbornly = infinitely often — $\mathsf{G}\,\mathsf{F}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted}$

Example properties

- infinitely often ϕ $\quad \mathsf{G}\,\mathsf{F}\,\phi$
- almost always ϕ $\quad \mathsf{F}\,\mathsf{G}\,\phi$
- "weak" $\phi_1\,\mathsf{U}\,\phi_2$: ϕ_2 not necessarily $\quad \mathsf{G}\,\phi_1 \vee \phi_1\,\mathsf{U}\,\phi_2$
- if `req` then `granted` later

  $\texttt{req} \implies \mathsf{X}\,\mathsf{F}\,\texttt{granted}$

- fairness: if stubbornly `req` then `granted`

  "weak": stubbornly = almost always $\quad \mathsf{F}\,\mathsf{G}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted}$

  "strong": stubbornly = infinitely often $\quad \mathsf{G}\,\mathsf{F}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted}$

- fairness (variant):

  "weak": $\quad \mathsf{F}\,\mathsf{G}\,\texttt{req} \implies \mathsf{G}\,\mathsf{F}\,\texttt{granted} = \mathsf{G}\,(\mathsf{F}\,\mathsf{G}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted})$

  "strong": $\quad \mathsf{G}\,\mathsf{F}\,\texttt{req} \implies \mathsf{G}\,\mathsf{F}\,\texttt{granted} = \mathsf{G}\,(\mathsf{G}\,\mathsf{F}\,\texttt{req} \implies \mathsf{F}\,\texttt{granted})$

De Morgan's laws

$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$

$? \equiv \neg\,\mathsf{X}\,\neg\phi$

De Morgan's laws

$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$

$? \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$

$\mathsf{X}\,\phi \equiv \neg\,\mathsf{X}\,\neg\phi$

De Morgan's laws

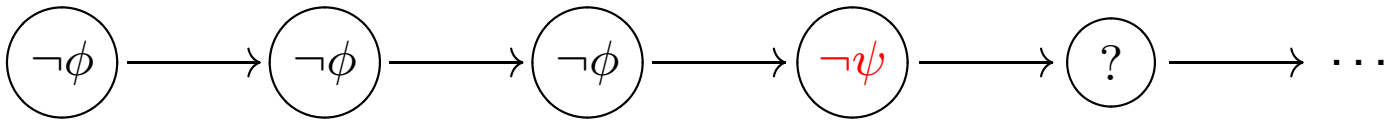
$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\mathsf{X}\,\phi \equiv \neg\,\mathsf{X}\,\neg\phi$

$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$

$\phi\,\mathsf{R}\,\psi \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$

$\neg\phi\,\mathsf{U}\,\neg\psi$



$\Pi \models \phi\,\mathsf{R}\,\psi$ iff ?

De Morgan's laws

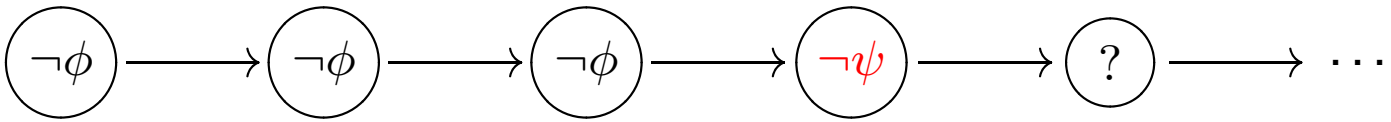
$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\mathsf{X}\,\phi \equiv \neg\,\mathsf{X}\,\neg\phi$

$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$

$\phi\,\mathsf{R}\,\psi \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$

$\neg\phi\,\mathsf{U}\,\neg\psi$



$\Pi \models \phi\,\mathsf{R}\,\psi$ iff $\forall i < |\Pi|.\ (\forall j < i.\ \Pi^j \models \neg\phi) \implies \Pi^i \models \psi$

De Morgan's laws

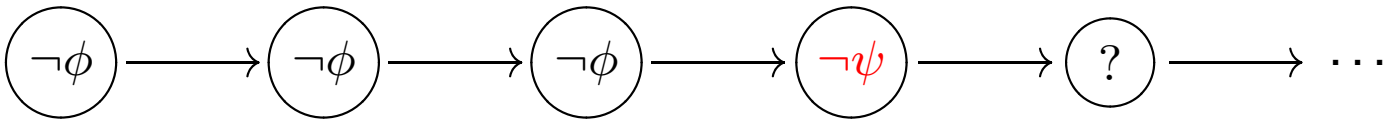
$\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$

$\mathsf{X}\,\phi \equiv \neg\,\mathsf{X}\,\neg\phi$

$\mathsf{G}\,\phi \equiv \neg\,\mathsf{F}\,\neg\phi$

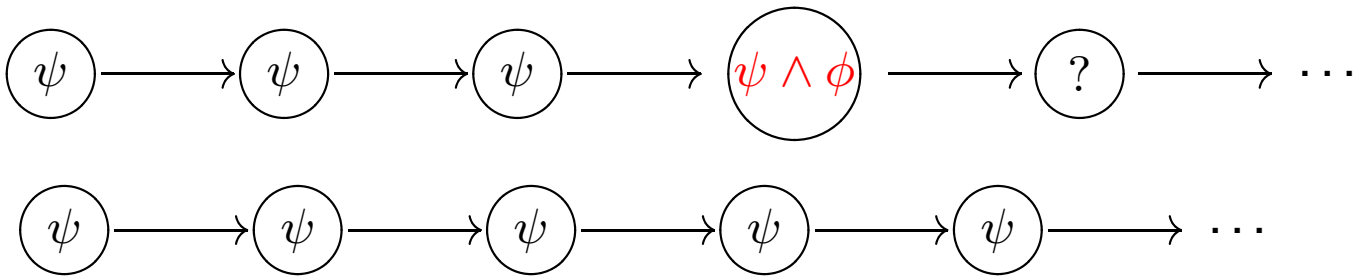
$\phi\,\mathsf{R}\,\psi \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi)$

$\neg\phi\,\mathsf{U}\,\neg\psi$



$\Pi \models \phi\,\mathsf{R}\,\psi$ iff $\forall i < |\Pi|.\ (\forall j < i.\ \Pi^j \models \neg\phi) \implies \Pi^i \models \psi$

$\phi\,\mathsf{R}\,\psi$



$\phi\,\mathsf{R}\,\psi \equiv \neg(\neg\phi\,\mathsf{U}\,\neg\psi) \equiv \psi\,\mathsf{U}\,(\psi \wedge \phi) \vee \mathsf{G}\,\psi \equiv \psi\,\mathsf{W}\,(\psi \wedge \phi)$

U and R as fixed points

$$\phi \,\mathsf{U}\, \psi \quad \equiv \quad \psi \;\vee\; (\phi \wedge \mathsf{X}\,(\phi \,\mathsf{U}\, \psi))$$

U and R as fixed points

$$\phi \,\mathsf{U}\, \psi \quad \equiv \quad \psi \ \vee \ (\phi \wedge \mathsf{X}(\phi \,\mathsf{U}\, \psi))$$

$$\phi \,\mathsf{R}\, \psi \quad \equiv \quad \psi \ \wedge \ (\phi \vee \mathsf{X}(\phi \,\mathsf{R}\, \psi))$$

Pushing negation down

$$\neg(\phi_1 \wedge \phi_2) \equiv \neg\phi_1 \vee \neg\phi_2$$

$$\neg \mathsf{F}\,\phi \equiv \mathsf{G}\,\neg\phi$$

$$\neg \mathsf{G}\,\phi \equiv \mathsf{F}\,\neg\phi$$

$$\neg \mathsf{X}\,\phi \equiv \mathsf{X}\,\neg\phi$$

Pushing negation down

$\neg(\phi_1 \wedge \phi_2) \equiv \neg\phi_1 \vee \neg\phi_2$

$\neg \mathsf{F}\,\phi \equiv \mathsf{G}\,\neg\phi$

$\neg \mathsf{G}\,\phi \equiv \mathsf{F}\,\neg\phi$

$\neg \mathsf{X}\,\phi \equiv \mathsf{X}\,\neg\phi$

$\neg(\phi\,\mathsf{U}\,\psi) \equiv (\neg\psi)\,\mathsf{W}\,(\neg\phi \wedge \neg\psi)$ why not in this way?

$\neg(\phi\,\mathsf{U}\,\psi) \equiv \neg\phi\,\mathsf{R}\,\neg\psi$

Expressivity

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand ?

(1') . . . strictly beforehand ?

(2) every b is proceeded by a that appears after last b,if any before ?

(3) alternating blocks of a and b ("relay") ?

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

$$\equiv \neg b\, \mathsf{W}\, (a \wedge \neg b) \equiv a\, \mathsf{R}\, \neg b \equiv SPr(a, b)$$

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

$$\equiv \neg b\, \mathsf{W}\, (a \wedge \neg b) \equiv a\, \mathsf{R}\, \neg b \equiv SPr(a, b)$$

(2) every b is proceeded by a that appears after last b, if any before

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

$$\equiv \neg b\, \mathsf{W}\, (a \wedge \neg b) \equiv a\, \mathsf{R}\, \neg b \equiv SPr(a, b)$$

(2) every b is proceeded by a that appears after last b, if any before

$$Pr(a, b) \wedge \mathsf{G}\, (b \implies \mathsf{X}\, Pr(a, b))$$

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

$$\equiv \neg b\, \mathsf{W}\, (a \wedge \neg b) \equiv a\, \mathsf{R}\, \neg b \equiv SPr(a, b)$$

(2) every b is proceeded by a that appears after last b, if any before

$$Pr(a, b) \wedge \mathsf{G}\, (b \implies \mathsf{X}\, Pr(a, b))$$

(3) alternating blocks of a and b ("relay")

Write a formula . . .

$P = \{a, b, \ldots\}$

(1) if b then some a was beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, a)$$

$$\equiv \neg b\, \mathsf{W}\, a \equiv Pr(a, b)$$

(1') . . . strictly beforehand

$$\mathsf{F}\, b \implies (\neg b\, \mathsf{U}\, (a \wedge \neg b))$$

$$\equiv \neg b\, \mathsf{W}\, (a \wedge \neg b) \equiv a\, \mathsf{R}\, \neg b \equiv SPr(a, b)$$

(2) every b is proceeded by a that appears after last b, if any before

$$Pr(a, b) \wedge \mathsf{G}\, (b \implies \mathsf{X}\, Pr(a, b))$$

(3) alternating blocks of a and b ("relay")

$$\mathsf{G}\, (\, (a \implies a\, \mathsf{W}\, (\neg a \wedge b)) \wedge (b \implies \ldots)\,)$$

What is inexpressible?

(1) on every path a state appears such that

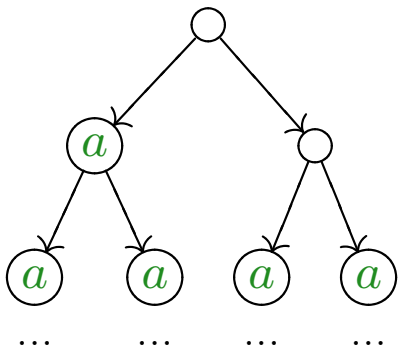
in every successor state ?

(on every path) a holds

(1') on some path ... ?

(2) on every path a state appears such that

in every following state a holds ?



What is inexpressible?

(1) on every path a state appears such that

in every successor state F X a ?

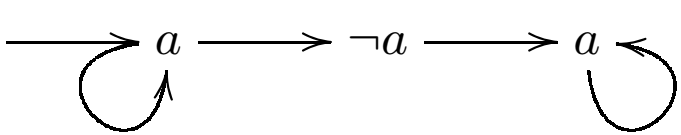
(on every path) a holds

(1') on some path . . . ?

(2) on every path a state appears such that

in every following state a holds F G a ?

too weak!



$\models$ F G a

What is inexpressible? (cont.)

(3) even(a): on every even position a ?

What is inexpressible? (cont.)

(3) even(a): on every even position a ?

(3') oddeven(a): on every even position a

and on every odd position $\neg a$

What is inexpressible? (cont.)

(3) even(a): on every even position a ?

(3') oddeven(a): on every even position a

and on every odd position $\neg a$

$$\mathsf{G}\,(\,(a \implies \mathsf{X}\,\neg a) \wedge (\neg a \implies \mathsf{X}\,a)$$

What is inexpressible? (cont.)

(3) even(a): on every even position a ?

(3') oddeven(a): on every even position a

and on every odd position $\neg a$

$$\mathsf{G}\,(\,(a \implies \mathsf{X}\,\neg a) \wedge (\neg a \implies \mathsf{X}\,a)$$

(4) from every reachable state some initial state is

reachable ?

Expressivity

Thm.: LTL = LTL(X, U) is more expressive than LTL(X, F)

Thm.: LTL = FO($\leq$)

Thm: Past temporal connectives:

$$X^{-1},\ U^{-1},\ F^{-1},\ G^{-1}$$

do not increase expressive power.

Thm: LTL(F, G, F^{-1}, G^{-1}) = ?

Model checking

Decision problems

Model checking

PSPACE-complete

- input: M, ϕ
- question: $M \models \phi$?

Satisfiability

PSPACE-complete

- input: ϕ
- question: $\exists M.\ M \models \phi$? ($\exists \Pi.\ \Pi \models \phi$?)

Complexity of model checking

$$|M| \cdot 2^{\mathcal{O}(|\phi|)}$$

$2^{\mathcal{O}(|\phi|)}$ OK

$|M|$ too much!

Complexity of model checking

$$|M| \cdot 2^{\mathcal{O}(|\phi|)}$$

$2^{\mathcal{O}(|\phi|)}$ OK

$|M|$ too much!

When the state space is huge, or even infinite:

- abstraction
- symbolic approaches
- on-the-fly state-space exploration

Algorithm

(1) $M \mapsto \mathcal{A}_M$

(2) $\neg\phi \mapsto \mathcal{A}_{\neg\phi}$

LTL $\rightarrow$ ω-automata

(3) $L(\mathcal{A}_M \times \mathcal{A}_{\neg\phi}) = \emptyset$?

yes $\rightarrow$ $M \models \phi$

no $\rightarrow$ $\neg(M \models \phi)$, counterexample = a path in M

Algorithm

(1) $M \mapsto \mathcal{A}_M$

(2) $\neg\phi \mapsto \mathcal{A}_{\neg\phi}$

LTL $\rightarrow$ ω-automata

(3) $L(\mathcal{A}_M \times \mathcal{A}_{\neg\phi}) = \emptyset$?

yes $\rightarrow$ $M \models \phi$

no $\rightarrow$ $\neg(M \models \phi)$, counterexample = a path in M

$\phi = \mathsf{G}\,(p \implies \mathsf{X}\,\mathsf{F}\,q)$

$A_{\neg\phi} =$