Basics of model checking

Paul Gastin

LIAFA (Paris) and LSV (Cachan)

Paul.Gastin@liafa.jussieu.fr

Paul.Gastin@lsv.ens-cachan.fr

MOVEP, Dec. 2004

Outline

Need for formal verifications methods

Critical systems

- Transport
- Energy
- Medicine
- Communication
- Finance
- Embedded systems
- ...

Complementary approaches

- Theorem prover
- Model checking
- Test

Need for formal verifications methods

Critical systems

- Transport
- Energy
- Medicine
- Communication
- Finance
- Embedded systems
- ...

Complementary approaches

- Theorem prover
- Model checking
- Test

Model Checking

3 steps

- Constructing the model M (transition systems)
- Formalizing the specification φ (temporal logics)
- Checking whether $M \models \varphi$ (algorithmics)

Main difficulties

- Size of models (combinatorial explosion)
- Expressivity of models or logics
- Decidability and complexity of the model-checking problem
- Efficiency of tools

Challenges

- Extend models and algorithms to cope with more systems.
  Infinite systems, parameterized systems, probabilistic systems, concurrent systems, timed systems, hybrid systems, ...
- Scale current tools to cope with real-size systems.
  Needs for modularity, abstractions, symmetries, ...

Model Checking

3 steps

- Constructing the model M (transition systems)
- Formalizing the specification φ (temporal logics)
- Checking whether $M \models \varphi$ (algorithmics)

Main difficulties

- Size of models (combinatorial explosion)
- Expressivity of models or logics
- Decidability and complexity of the model-checking problem
- Efficiency of tools

Challenges

- Extend models and algorithms to cope with more systems.
  Infinite systems, parameterized systems, probabilistic systems, concurrent systems, timed systems, hybrid systems, ...
- Scale current tools to cope with real-size systems.
  Needs for modularity, abstractions, symmetries, ...

Model Checking

3 steps

- Constructing the model M (transition systems)
- Formalizing the specification φ (temporal logics)
- Checking whether $M \models \varphi$ (algorithmics)

Main difficulties

- Size of models (combinatorial explosion)
- Expressivity of models or logics
- Decidability and complexity of the model-checking problem
- Efficiency of tools

Challenges

- Extend models and algorithms to cope with more systems.
  Infinite systems, parameterized systems, probabilistic systems, concurrent systems, timed systems, hybrid systems, ...
- Scale current tools to cope with real-size systems.
  Needs for modularity, abstractions, symmetries, ...

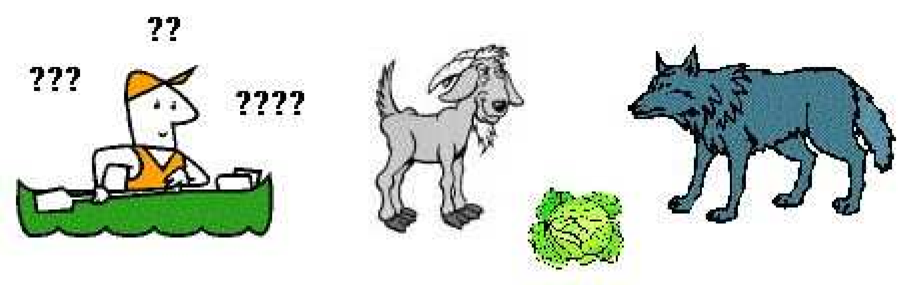
References

- The Temporal Logic of Reactive and Concurrent Systems: Specification. Z. Manna and A. Pnueli. Springer, 1991.
- Temporal Verification of Reactive Systems: Safety. Z. Manna and A. Pnueli. Springer, 1995.
- Model Checking. E.M. Clarke, O. Grumberg, D.A. Peled. MIT Press, 1999.
- Systems and Software Verification. Model-Checking Techniques and Tools. B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen. Springer, 2001.

Outline

Constructing the model

Example : Men, Wolf, Goat, Cabbage

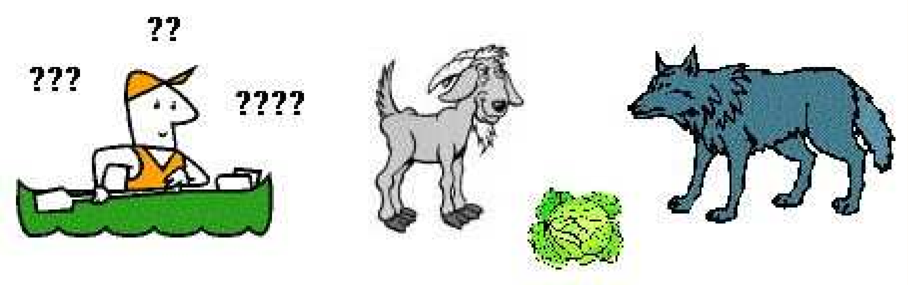


Model = Transition system

- State = who is on which side of the river
- Transition = crossing the river

Constructing the model

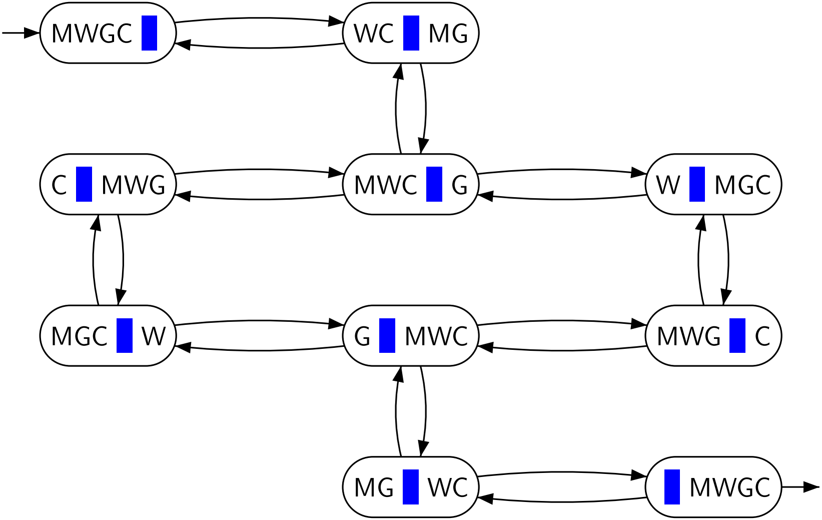
Example : Men, Wolf, Goat, Cabbage



Model = Transition system

- State = who is on which side of the river
- Transition = crossing the river

Transition system



Kripke structure

$M = (S, A, T, I, \mathrm{AP}, \ell)$

- S: set of states (often finite)
- $T \subseteq S \times A \times S$: set of transitions
- $I \subseteq S$: set of initial states
- AP: set of atomic propositions
- $\ell : S \to 2^{\mathrm{AP}}$: labelling function.

Digicode

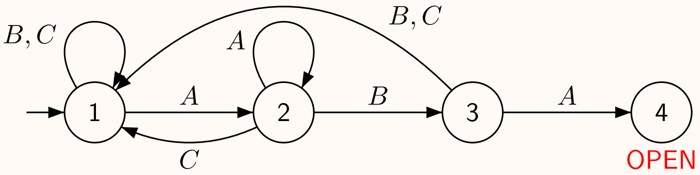
Pb: How can we easily describe big systems?

Kripke structure

$M = (S, A, T, I, \mathrm{AP}, \ell)$

- S: set of states (often finite)
- $T \subseteq S \times A \times S$: set of transitions
- $I \subseteq S$: set of initial states
- AP: set of atomic propositions
- $\ell : S \to 2^{\mathrm{AP}}$: labelling function.

Digicode



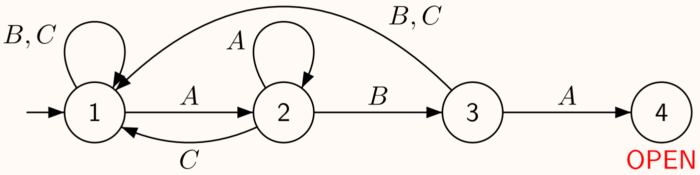
Pb: How can we easily describe big systems?

Kripke structure

$M = (S, A, T, I, \mathrm{AP}, \ell)$

- S: set of states (often finite)
- $T \subseteq S \times A \times S$: set of transitions
- $I \subseteq S$: set of initial states
- AP: set of atomic propositions
- $\ell : S \to 2^{\mathrm{AP}}$: labelling function.

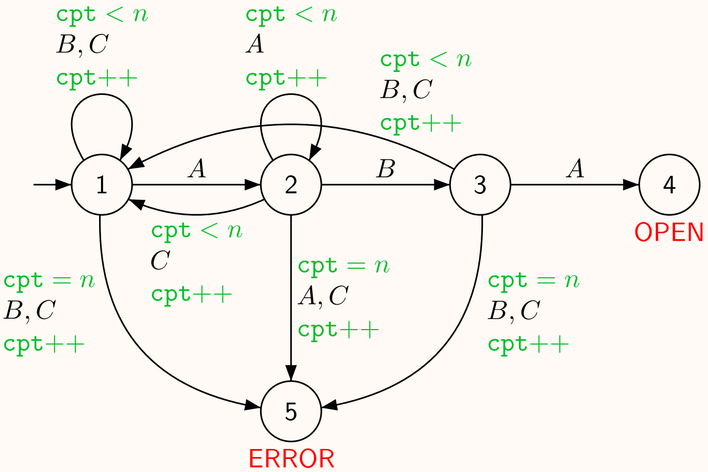
Digicode



Pb: How can we easily describe big systems?

Using variables

Digicode



Kripke structures with variables

$M = (S, A, \mathcal{V}, T, I, \mathrm{AP}, \ell)$

- $\mathcal{V}$: set of (typed) variables, e.g., boolean, [0..4], ...
- Condition: formula involving variables
- Update: modification of variables
- Transition: $p \xrightarrow{\text{condition,label,update}} q$

Programs = Kripke structures with variables

- Program counter = states
- Instructions = transitions
- Variables = variables

Kripke structures with variables

$M = (S, A, \mathcal{V}, T, I, \mathrm{AP}, \ell)$

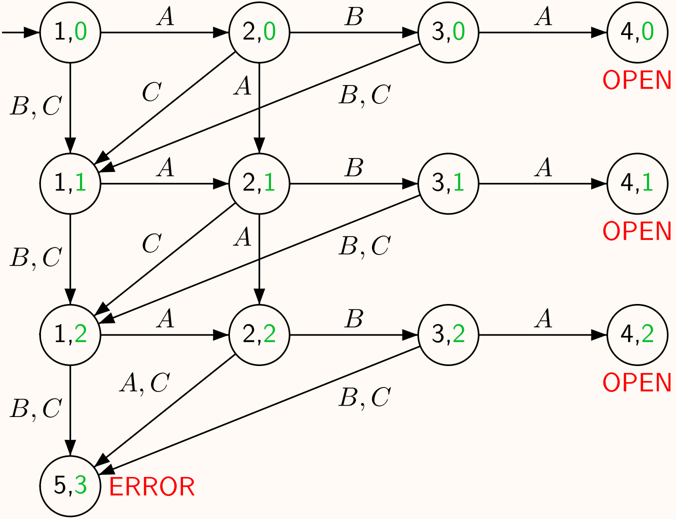
- $\mathcal{V}$: set of (typed) variables, e.g., boolean, [0..4], …
- Condition: formula involving variables
- Update: modification of variables
- Transition: $p \xrightarrow{\text{condition,label,update}} q$

Programs = Kripke structures with variables

- Program counter = states
- Instructions = transitions
- Variables = variables

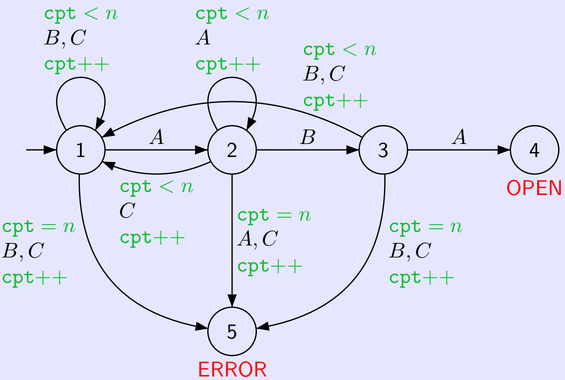
Expanding variables ($n = 2$)

Digicode



Symbolic representation

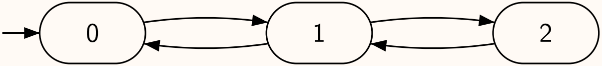
Logical representation

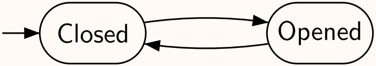


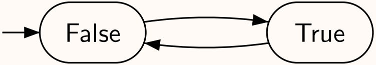
$$
\begin{array}{rll}
\delta_B = & & s = 1 \wedge \mathtt{cpt} < n \wedge s' = 1 \wedge \mathtt{cpt}' = \mathtt{cpt} + 1 \\
& \vee & s = 1 \wedge \mathtt{cpt} = n \wedge s' = 5 \wedge \mathtt{cpt}' = \mathtt{cpt} + 1 \\
& \vee & s = 2 \wedge s' = 3 \wedge \mathtt{cpt}' = \mathtt{cpt} \\
& \vee & s = 3 \wedge \mathtt{cpt} < n \wedge s' = 1 \wedge \mathtt{cpt}' = \mathtt{cpt} + 1 \\
& \vee & s = 3 \wedge \mathtt{cpt} = n \wedge s' = 5 \wedge \mathtt{cpt}' = \mathtt{cpt} + 1
\end{array}
$$

Modular description of concurrent systems

Elevator

- Cabin: 

- Door for level i: 

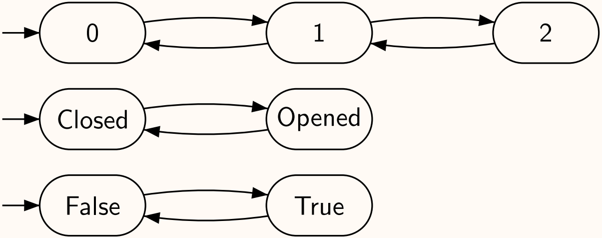
- Call for level i: 

The actual system is a synchronized product of all these automata.
It consists of (at most) $3 \times 2^3 \times 2^3 = 192$ states.

Modular description of concurrent systems

Elevator

- Cabin:
- Door for level i:
- Call for level i:



The actual system is a synchronized product of all these automata.
It consists of (at most) $3 \times 2^3 \times 2^3 = 192$ states.

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

$S = \prod_i S_i, \quad A = \prod_i (A_i \cup \{\varepsilon\}), \quad$ and $\quad I = \prod_i I_i$

$T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or $p_i = q_i$ and $a_i = \varepsilon\}$

$\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

  $S = \prod_i S_i, \quad A = \prod_i (A_i \cup \{\varepsilon\}), \quad$ and $\quad I = \prod_i I_i$

  $T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or $p_i = q_i$ and $a_i = \varepsilon\}$

  $\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

$S = \prod_i S_i$, $A = \prod_i (A_i \cup \{\varepsilon\})$, and $I = \prod_i I_i$

$$T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid \text{for all } i, (p_i, a_i, q_i) \in T_i \text{ or } p_i = q_i \text{ and } a_i = \varepsilon\}$$

$\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

  $S = \prod_i S_i$, $\quad A = \prod_i (A_i \cup \{\varepsilon\})$, $\quad$ and $\quad I = \prod_i I_i$

  $T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or $p_i = q_i$ and $a_i = \varepsilon\}$

  $\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

$S = \prod_i S_i, \quad A = \prod_i (A_i \cup \{\varepsilon\}), \quad$ and $\quad I = \prod_i I_i$

$T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or
$p_i = q_i$ and $a_i = \varepsilon\}$

$\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

$S = \prod_i S_i, \quad A = \prod_i (A_i \cup \{\varepsilon\}), \quad$ and $\quad I = \prod_i I_i$

$T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or $p_i = q_i$ and $a_i = \varepsilon\}$

$\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\mathrm{sync}} = \prod_i A_i$
- Asynchronous: $A_{\mathrm{sync}} = \biguplus_i A_i$
- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $A_{\mathrm{sync}} \subseteq A$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

Synchronized products

General product

- Components: $M_i = (S_i, A_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, A, T, I, \mathrm{AP}, \ell)$ with

  $S = \prod_i S_i, \quad A = \prod_i (A_i \cup \{\varepsilon\}), \quad$ and $\quad I = \prod_i I_i$

  $T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ for all $i, (p_i, a_i, q_i) \in T_i$ or $p_i = q_i$ and $a_i = \varepsilon\}$

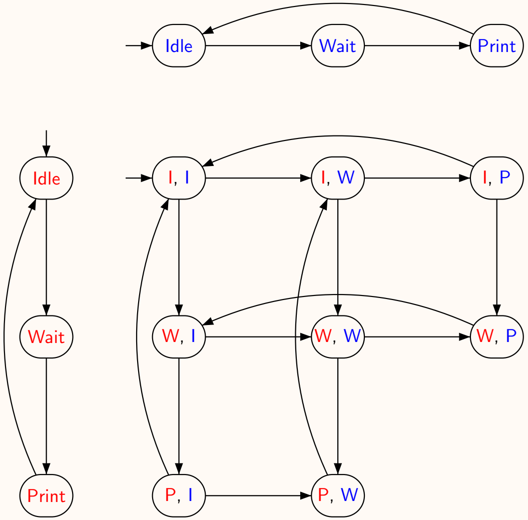
  $\mathrm{AP} = \biguplus_i \mathrm{AP}_i$ and $\ell(p_1, \ldots, p_n) = \bigcup_i \ell(p_i)$

Synchronized products are restrictions of the general product.

- Synchronous: $A_{\text{sync}} = \prod_i A_i$
- Asynchronous: $A_{\text{sync}} = \biguplus_i A_i$
- By states: $S_{\text{sync}} \subseteq S$
- By labels: $A_{\text{sync}} \subseteq A$
- By transitions: $T_{\text{sync}} \subseteq T$

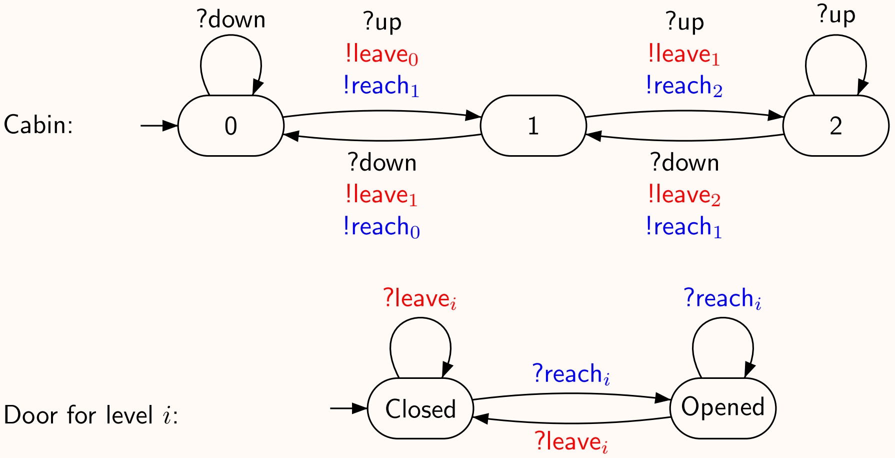
Example: Printer manager

Synchronization by states: (P, P) is forbidden



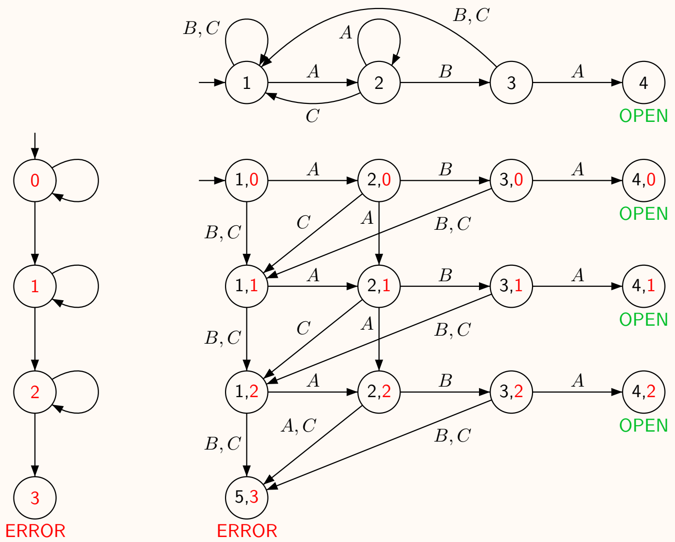
Example: Elevator

Synchronization by actions



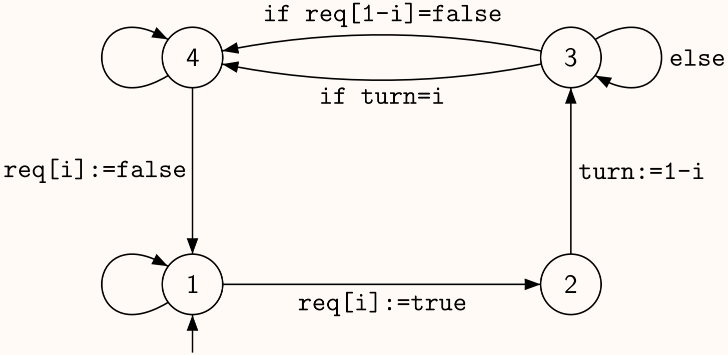
Example: digicode

Synchronization by transitions



Example: Peterson's algorithm (1981)

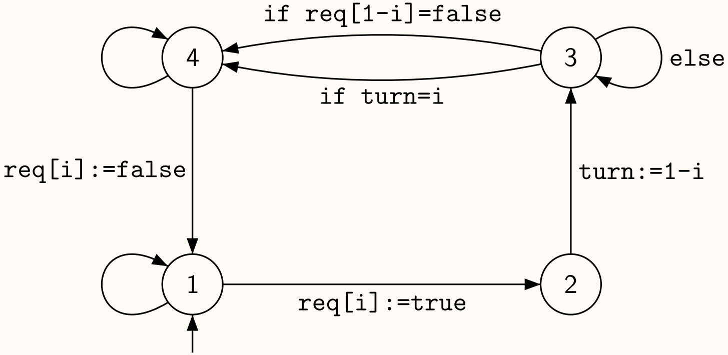
Synchronization by shared variables



The global state is a 5-tuple: ($\texttt{state}_0$, $\texttt{state}_1$, req[0], req[1], turn)

Example: Peterson's algorithm (1981)

Synchronization by shared variables



The global state is a 5-tuple: (state$_0$, state$_1$, req[0], req[1], turn)

High-level descriptions

- Sequential programs = transition system with variables
- Concurrent programs with shared variables
- Concurrent programs with Rendez-vous
- Concurrent programs with FIFO communication
- Petri net
- ...

Models: expressivity versus decidability

(Un)decidability

- Automata with 2 integer variables = Turing powerful
  Restriction to variables taking values in finite sets
- Asynchronous communication: unbounded fifo channels = Turing powerful
  Restriction to bounded channels

Some infinite state models are decidable

- Petri nets. Several unbounded integer variables but no zero-test.
- Pushdown automata. Model for recursive procedure calls.
- Timed automata.
- ...

Models: expressivity versus decidability

(Un)decidability

- Automata with 2 integer variables = Turing powerful
  Restriction to variables taking values in finite sets
- Asynchronous communication: unbounded fifo channels = Turing powerful
  Restriction to bounded channels

Some infinite state models are decidable

- Petri nets. Several unbounded integer variables but no zero-test.
- Pushdown automata. Model for recursive procedure calls.
- Timed automata.
- ...

Outline

Static and dynamic properties

Static properties

Example: Mutual exclusion

Most safety properties are static.

They can be reduced to reachability.

Dynamic properties

Example: Every request should be eventually granted.

$$\bigwedge_i \forall t, (\mathrm{Call}_i(t) \longrightarrow \exists t' \geq t, (\mathrm{atLevel}_i(t') \wedge \mathrm{openDoor}_i(t')))$$

The elevator should not cross a level for which a call is pending without stopping.

$$\bigwedge_i \forall t \forall t', (\mathrm{Call}_i(t) \wedge t \leq t' \wedge \mathrm{atLevel}_i(t')) \longrightarrow \exists t \leq t'' \leq t', (\mathrm{atLevel}_i(t'') \wedge \mathrm{openDoor}_i(t''))$$

Static and dynamic properties

Static properties

Example: Mutual exclusion

Most safety properties are static.

They can be reduced to reachability.

Dynamic properties

Example: Every request should be eventually granted.

$$\bigwedge_i \forall t, (\mathrm{Call}_i(t) \longrightarrow \exists t' \geq t, (\mathrm{atLevel}_i(t') \wedge \mathrm{openDoor}_i(t')))$$

The elevator should not cross a level for which a call is pending without stopping.

$$\bigwedge_i \forall t \forall t', (\mathrm{Call}_i(t) \wedge t \leq t' \wedge \mathrm{atLevel}_i(t')) \longrightarrow \exists t \leq t'' \leq t', (\mathrm{atLevel}_i(t'') \wedge \mathrm{openDoor}_i(t'')))$$

First Order specifications

First order logic

- These specifications can be written in $\mathrm{FO}(<)$.
- $\mathrm{FO}(<)$ has a good expressive power.
  ... but $\mathrm{FO}(<)$-formulas are not easy to write and to understand.
- $\mathrm{FO}(<)$ is decidable.
  ... but satisfiability and model checking are non elementary.

Temporal logics

- no variables: time is implicit.
- quantifications and variables are replaced by modalities.
- Usual specifications are easy to write and read.
- Good complexity for satisfiability and model checking problems.

First Order specifications

First order logic

- These specifications can be written in $\mathrm{FO}(<)$.
- $\mathrm{FO}(<)$ has a good expressive power.
  ... but $\mathrm{FO}(<)$-formulas are not easy to write and to understand.
- $\mathrm{FO}(<)$ is decidable.
  ... but satisfiability and model checking are non elementary.

Temporal logics

- no variables: time is implicit.
- quantifications and variables are replaced by modalities.
- Usual specifications are easy to write and read.
- Good complexity for satisfiability and model checking problems.

Linear versus Branching

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure.

Linear specifications

Example: The printer manager is fair.
On each run, whenever some process requests the printer, it eventually gets it.

Execution sequences (runs): $\sigma = s_0 \to s_1 \to s_2 \to \cdots$ with $s_i \to s_{i+1} \in T$

Two Kripke structures having the same execution sequences satisfy the same linear specifications.

Actually, linear specifications only depend on the label of the execution sequence

$$\ell(\sigma) = \ell(s_0) \to \ell(s_1) \to \ell(s_2) \to \cdots$$

Branching specifications

Example: Each process has the possibility to print first.

Such properties depend on the execution tree.

Execution tree = unfolding of the transition system

Linear versus Branching

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure.

Linear specifications

Example: The printer manager is fair.
On each run, whenever some process requests the printer, it eventually gets it.

Execution sequences (runs): $\sigma = s_0 \to s_1 \to s_2 \to \cdots$ with $s_i \to s_{i+1} \in T$

Two Kripke structures having the same execution sequences satisfy the same linear specifications.

Actually, linear specifications only depend on the label of the execution sequence

$$\ell(\sigma) = \ell(s_0) \to \ell(s_1) \to \ell(s_2) \to \cdots$$

Branching specifications

Example: Each process has the possibility to print first.

Such properties depend on the execution tree.

Execution tree $=$ unfolding of the transition system

Outline

Linear Temporal Logic (Pnueli 1977)

Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models p$	if	$p \in \lambda(x)$
$t, x \models \neg\varphi$	if	$t, x \not\models \varphi$
$t, x \models \varphi \vee \psi$	if	$t, x \models \varphi$ or $t, x \models \psi$
$t, x \models \mathsf{X}\,\varphi$	if	$\exists y.\ x \lessdot y \ \&\ t, y \models \varphi$
$t, x \models \varphi \,\mathsf{U}\, \psi$	if	$\exists z.\ x \leq z \ \&\ t, z \models \psi \ \&\ \forall y.\ (x \leq y < z) \to t, y \models \varphi$

Example

Linear Temporal Logic (Pnueli 1977)

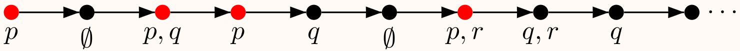
Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models p$	if	$p \in \lambda(x)$
$t, x \models \neg\varphi$	if	$t, x \not\models \varphi$
$t, x \models \varphi \vee \psi$	if	$t, x \models \varphi$ or $t, x \models \psi$
$t, x \models \mathsf{X}\,\varphi$	if	$\exists y.\ x \lessdot y \ \ \&\ \ t, y \models \varphi$
$t, x \models \varphi \,\mathsf{U}\, \psi$	if	$\exists z.\ x \leq z \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (x \leq y < z) \to t, y \models \varphi$

Example



Linear Temporal Logic (Pnueli 1977)

Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models p$	if	$p \in \lambda(x)$
$t, x \models \neg\varphi$	if	$t, x \not\models \varphi$
$t, x \models \varphi \vee \psi$	if	$t, x \models \varphi$ or $t, x \models \psi$
$t, x \models \mathsf{X}\,\varphi$	if	$\exists y.\ x \lessdot y \ \& \ t, y \models \varphi$
$t, x \models \varphi \,\mathsf{U}\, \psi$	if	$\exists z.\ x \leq z \ \& \ t, z \models \psi \ \& \ \forall y.\ (x \leq y < z) \to t, y \models \varphi$

Example

Linear Temporal Logic (Pnueli 1977)

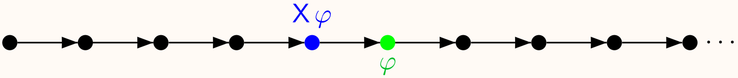
Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi$$

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models p$	if	$p \in \lambda(x)$
$t, x \models \neg\varphi$	if	$t, x \not\models \varphi$
$t, x \models \varphi \vee \psi$	if	$t, x \models \varphi$ or $t, x \models \psi$
$t, x \models \mathsf{X}\,\varphi$	if	$\exists y.\ x \lessdot y \ \&\ t, y \models \varphi$
$t, x \models \varphi\,\mathsf{U}\,\psi$	if	$\exists z.\ x \leq z \ \&\ t, z \models \psi \ \&\ \forall y.\ (x \leq y < z) \to t, y \models \varphi$

Example



Linear Temporal Logic (Pnueli 1977)

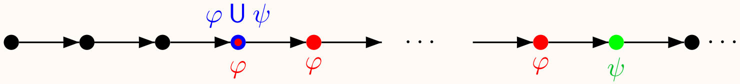
Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi$$

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models p$	if	$p \in \lambda(x)$
$t, x \models \neg\varphi$	if	$t, x \not\models \varphi$
$t, x \models \varphi \vee \psi$	if	$t, x \models \varphi$ or $t, x \models \psi$
$t, x \models \mathsf{X}\,\varphi$	if	$\exists y.\ x \lessdot y \ \ \& \ \ t, y \models \varphi$
$t, x \models \varphi \,\mathsf{U}\, \psi$	if	$\exists z.\ x \leq z \ \ \& \ \ t, z \models \psi \ \ \& \ \ \forall y.\ (x \leq y < z) \to t, y \models \varphi$

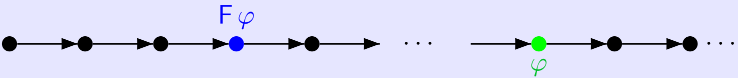
Example



Linear Temporal Logic (Pnueli 1977)

Macros:

- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$

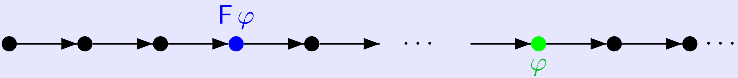


- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$
- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$
- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

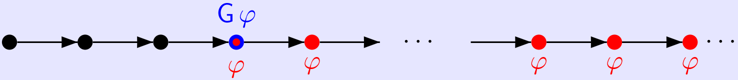
Linear Temporal Logic (Pnueli 1977)

Macros:

- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$

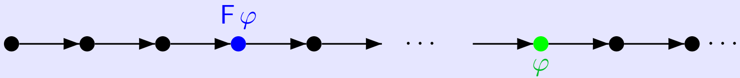


- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$
- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

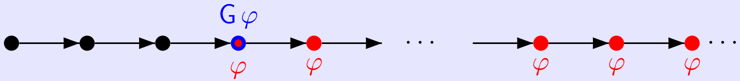
Linear Temporal Logic (Pnueli 1977)

Macros:

- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$

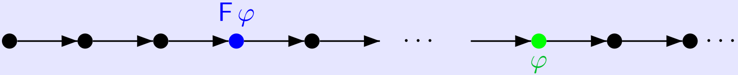


- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$
- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

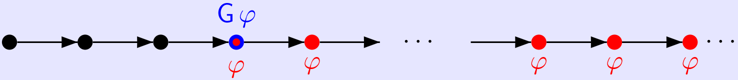
Linear Temporal Logic (Pnueli 1977)

Macros:

- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$

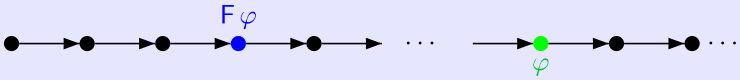


- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$
- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

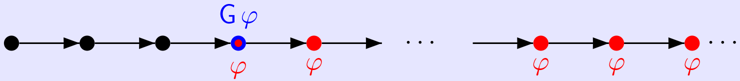
Linear Temporal Logic (Pnueli 1977)

Macros:

- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$

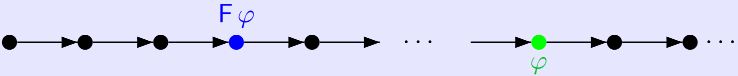


- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$
- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

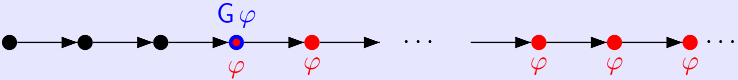
Linear Temporal Logic (Pnueli 1977)

Macros:

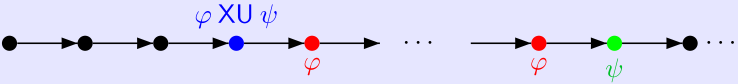
- Eventually: $\mathsf{F}\,\varphi = \top\;\mathsf{U}\;\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$



- Weak until: $\varphi\;\mathsf{W}\;\psi = \mathsf{G}\,\varphi \vee \varphi\;\mathsf{U}\;\psi$
- $\neg(\varphi\;\mathsf{U}\;\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\;\mathsf{U}\;(\neg\varphi \wedge \neg\psi)) = \neg\psi\;\mathsf{W}\;(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\;\mathsf{R}\;\psi = \psi\;\mathsf{W}\;(\varphi \wedge \psi) = \neg(\neg\varphi\;\mathsf{U}\;\neg\psi)$
- Next until: $\varphi\;\mathsf{XU}\;\psi = \mathsf{X}(\varphi\;\mathsf{U}\;\psi)$

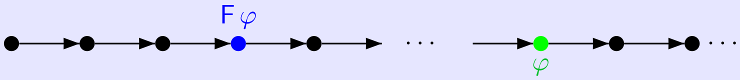


- $\mathsf{X}\,\psi = \bot\;\mathsf{XU}\;\psi$ and $\varphi\;\mathsf{U}\;\psi = \psi \vee (\varphi \wedge \varphi\;\mathsf{XU}\;\psi)$.

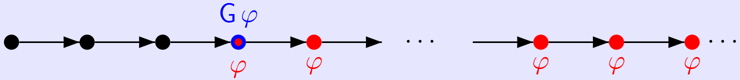
Linear Temporal Logic (Pnueli 1977)

Macros:

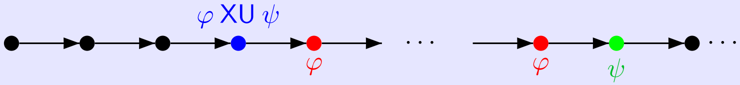
- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- Always: $\mathsf{G}\,\varphi = \neg\,\mathsf{F}\,\neg\varphi$



- Weak until: $\varphi\,\mathsf{W}\,\psi = \mathsf{G}\,\varphi \vee \varphi\,\mathsf{U}\,\psi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) = \neg\psi\,\mathsf{W}\,(\neg\varphi \wedge \neg\psi)$
- Release: $\varphi\,\mathsf{R}\,\psi = \psi\,\mathsf{W}\,(\varphi \wedge \psi) = \neg(\neg\varphi\,\mathsf{U}\,\neg\psi)$
- Next until: $\varphi\,\mathsf{XU}\,\psi = \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$



- $\mathsf{X}\,\psi = \bot\,\mathsf{XU}\,\psi$ and $\varphi\,\mathsf{U}\,\psi = \psi \vee (\varphi \wedge \varphi\,\mathsf{XU}\,\psi)$.

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\;\mathsf{U}\;\mathrm{grant}))$
- Release: $\mathrm{reset}\;\mathsf{R}\;\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\ \mathsf{U}\ \mathrm{grant}))$
- Release: $\mathrm{reset}\ \mathsf{R}\ \mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\;\mathsf{U}\;\mathrm{grant}))$
- Release: $\mathrm{reset}\;\mathsf{R}\;\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\;\mathsf{U}\;\mathrm{grant}))$
- Release: $\mathrm{reset}\;\mathsf{R}\;\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\,\mathsf{U}\,\mathrm{grant}))$
- Release: $\mathrm{reset}\,\mathsf{R}\,\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg\,\mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \rightarrow \mathsf{X}(\neg\mathrm{request}\;\mathsf{U}\;\mathrm{grant}))$
- Release: $\mathrm{reset}\;\mathsf{R}\;\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\text{good}$
- MutEx: $\neg\,\mathsf{F}(\text{crit}_1 \wedge \text{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\text{active}$
- Response: $\mathsf{G}(\text{request} \rightarrow \mathsf{F}\,\text{grant})$
- Response': $\mathsf{G}(\text{request} \rightarrow \mathsf{X}(\neg\text{request}\;\mathsf{U}\;\text{grant}))$
- Release: $\text{reset}\;\mathsf{R}\;\text{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\text{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\text{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\text{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\text{grant}$

Linear Temporal Logic (Pnueli 1977)

Specifications:

- Safety: $\mathsf{G}\,\text{good}$
- MutEx: $\neg\,\mathsf{F}(\text{crit}_1 \wedge \text{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\text{active}$
- Response: $\mathsf{G}(\text{request} \rightarrow \mathsf{F}\,\text{grant})$
- Response': $\mathsf{G}(\text{request} \rightarrow \mathsf{X}(\neg\text{request}\,\mathsf{U}\,\text{grant}))$
- Release: $\text{reset}\,\mathsf{R}\,\text{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\text{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\text{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\text{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\text{grant}$

Linear Temporal Logic (Pnueli 1977)

Examples

Every elevator request should be eventually satisfied.

$$\bigwedge_i \mathsf{G}(\mathrm{Call}_i \rightarrow \mathsf{F}(\mathrm{atLevel}_i \wedge \mathrm{openDoor}_i))$$

The elevator should not cross a level for which a call is pending without stopping.

$$\bigwedge_i \mathsf{G}(\mathrm{Call}_i \rightarrow \neg\mathrm{atLevel}_i \;\mathsf{W}\; (\mathrm{atLevel}_i \wedge \mathrm{openDoor}_i))$$

Linear Temporal Logic (Pnueli 1977)

Examples

Every elevator request should be eventually satisfied.

$$\bigwedge_i \mathsf{G}(\mathrm{Call}_i \to \mathsf{F}(\mathrm{atLevel}_i \wedge \mathrm{openDoor}_i))$$

The elevator should not cross a level for which a call is pending without stopping.

$$\bigwedge_i \mathsf{G}(\mathrm{Call}_i \to \neg \mathrm{atLevel}_i \; \mathsf{W} \; (\mathrm{atLevel}_i \wedge \mathrm{openDoor}_i)$$

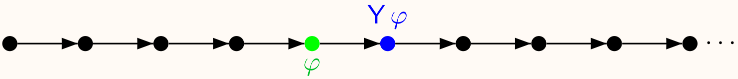
Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$t, x \models \mathsf{Y}\,\varphi$ if $\exists y.\ y \lessdot x\ \&\ t, y \models \varphi$

$t, x \models \varphi\,\mathsf{S}\,\psi$ if $\exists z.\ z \leq x\ \&\ t, z \models \psi\ \&\ \forall y.\ (z < y \leq x) \to t, y \models \varphi$

Example



LTL versus PLTL

$\mathsf{G}(\mathrm{grant} \to \mathsf{Y}(\neg\mathrm{grant}\,\mathsf{S}\,\mathrm{request}))$

$$= (\mathrm{request}\,\mathsf{R}\,\neg\mathrm{grant}) \wedge \mathsf{G}(\mathrm{grant} \to (\mathrm{request} \vee \mathsf{X}(\mathrm{request}\,\mathsf{R}\,\neg\mathrm{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

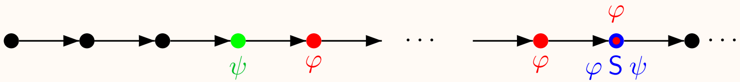
Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$t, x \models \mathsf{Y}\,\varphi \quad \text{if} \quad \exists y.\ y \lessdot x \ \ \&\ \ t, y \models \varphi$$

$$t, x \models \varphi\,\mathsf{S}\,\psi \quad \text{if} \quad \exists z.\ z \leq x \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi$$

Example



LTL versus PLTL

$$\mathsf{G}(\mathrm{grant} \to \mathsf{Y}(\neg\mathrm{grant}\,\mathsf{S}\,\mathrm{request}))$$

$$= (\mathrm{request}\,\mathsf{R}\,\neg\mathrm{grant}) \wedge \mathsf{G}(\mathrm{grant} \to (\mathrm{request} \vee \mathsf{X}(\mathrm{request}\,\mathsf{R}\,\neg\mathrm{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

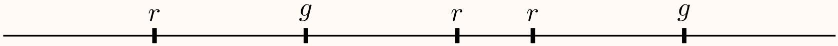
PLTL may be exponentially more succinct than LTL.

Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$t, x \models \mathsf{Y}\,\varphi \quad \text{if} \quad \exists y.\ y \lessdot x \ \ \&\ \ t, y \models \varphi$$
$$t, x \models \varphi\,\mathsf{S}\,\psi \quad \text{if} \quad \exists z.\ z \leq x \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi$$

Example



LTL versus PLTL

$\mathsf{G}(\text{grant} \to \mathsf{Y}(\neg\text{grant}\ \mathsf{S}\ \text{request}))$

$$= (\text{request}\ \mathsf{R}\ \neg\text{grant}) \wedge \mathsf{G}(\text{grant} \to (\text{request} \vee \mathsf{X}(\text{request}\ \mathsf{R}\ \neg\text{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

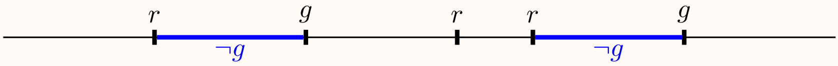
Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$t, x \models \mathsf{Y}\, \varphi \quad \text{if} \quad \exists y.\ y \lessdot x \ \ \&\ \ t, y \models \varphi$$

$$t, x \models \varphi \,\mathsf{S}\, \psi \quad \text{if} \quad \exists z.\ z \leq x \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi$$

Example



LTL versus PLTL

$\mathsf{G}(\mathrm{grant} \to \mathsf{Y}(\neg \mathrm{grant} \,\mathsf{S}\, \mathrm{request}))$

$$= (\mathrm{request} \,\mathsf{R}\, \neg \mathrm{grant}) \wedge \mathsf{G}(\mathrm{grant} \to (\mathrm{request} \vee \mathsf{X}(\mathrm{request} \,\mathsf{R}\, \neg \mathrm{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$
\begin{array}{lll}
t, x \models \mathsf{Y}\,\varphi & \text{if} & \exists y.\ y \lessdot x\ \ \&\ \ t, y \models \varphi \\
t, x \models \varphi\,\mathsf{S}\,\psi & \text{if} & \exists z.\ z \leq x\ \ \&\ \ t, z \models \psi\ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi
\end{array}
$$

Example

LTL versus PLTL

$\mathsf{G}(\text{grant} \to \mathsf{Y}(\neg\text{grant}\,\mathsf{S}\,\text{request}))$

$$= (\text{request}\,\mathsf{R}\,\neg\text{grant}) \wedge \mathsf{G}(\text{grant} \to (\text{request} \vee \mathsf{X}(\text{request}\,\mathsf{R}\,\neg\text{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$
\begin{array}{lll}
t, x \models \mathsf{Y}\,\varphi & \text{if} & \exists y.\ y \lessdot x \ \ \&\ \ t, y \models \varphi \\
t, x \models \varphi\,\mathsf{S}\,\psi & \text{if} & \exists z.\ z \leq x \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi
\end{array}
$$

Example

LTL versus PLTL

$\mathsf{G}(\mathrm{grant} \to \mathsf{Y}(\neg\mathrm{grant}\ \mathsf{S}\ \mathrm{request}))$

$$= (\mathrm{request}\ \mathsf{R}\ \neg\mathrm{grant}) \wedge \mathsf{G}(\mathrm{grant} \to (\mathrm{request} \vee \mathsf{X}(\mathrm{request}\ \mathsf{R}\ \neg\mathrm{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

Past LTL

Semantics: $t = [\mathbb{N}, \leq, \lambda]$ with $\lambda : \mathbb{N} \to \Sigma = 2^{\mathrm{AP}}$ and $x \in \mathbb{N}$

$$t, x \models \mathsf{Y}\,\varphi \quad \text{if} \quad \exists y.\ y \lessdot x \ \ \&\ \ t, y \models \varphi$$

$$t, x \models \varphi\,\mathsf{S}\,\psi \quad \text{if} \quad \exists z.\ z \leq x \ \ \&\ \ t, z \models \psi \ \ \&\ \ \forall y.\ (z < y \leq x) \to t, y \models \varphi$$

Example

LTL versus PLTL

$\mathsf{G}(\text{grant} \to \mathsf{Y}(\neg\text{grant}\ \mathsf{S}\ \text{request}))$

$$= (\text{request}\ \mathsf{R}\ \neg\text{grant}) \wedge \mathsf{G}(\text{grant} \to (\text{request} \vee \mathsf{X}(\text{request}\ \mathsf{R}\ \neg\text{grant})))$$

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

Expressivity

Theorem (Kamp 68)

$$\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$$

Separation Theorem (Gabbay, Pnueli, Shelah & Stavi 80)

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overleftarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{Y},\mathsf{S})$ and $\overrightarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$ and $k \geq 0$,

$$w,k \models \varphi \iff w,k \models \bigvee_i \overleftarrow{\varphi_i} \wedge \overrightarrow{\varphi_i}$$

Corollary: $\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{LTL}(\mathsf{X},\mathsf{U})$

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overrightarrow{\varphi} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$,

$$w,0 \models \varphi \iff w,0 \models \overrightarrow{\varphi}$$

Elegant algebraic proof of $\mathrm{LTL}(\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$ due to Wilke 98.

Expressivity

Theorem (Kamp 68)

$$\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$$

Separation Theorem (Gabbay, Pnueli, Shelah & Stavi 80)

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overleftarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{Y},\mathsf{S})$ and $\overrightarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$ and $k \geq 0$,

$$w,k \models \varphi \iff w,k \models \bigvee_i \overleftarrow{\varphi_i} \wedge \overrightarrow{\varphi_i}$$

Corollary: $\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{LTL}(\mathsf{X},\mathsf{U})$

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overrightarrow{\varphi} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$,

$$w,0 \models \varphi \iff w,0 \models \overrightarrow{\varphi}$$

Elegant algebraic proof of $\mathrm{LTL}(\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$ due to Wilke 98.

Expressivity

Theorem (Kamp 68)

$$\mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$$

Separation Theorem (Gabbay, Pnueli, Shelah & Stavi 80)

For all $\varphi \in \mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U})$ there exist $\overleftarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{Y}, \mathsf{S})$ and $\overrightarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{X}, \mathsf{U})$ such that for all $w \in \Sigma^\omega$ and $k \geq 0$,

$$w, k \models \varphi \iff w, k \models \bigvee_i \overleftarrow{\varphi_i} \wedge \overrightarrow{\varphi_i}$$

Corollary: $\mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U}) = \mathrm{LTL}(\mathsf{X}, \mathsf{U})$

For all $\varphi \in \mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U})$ there exist $\overrightarrow{\varphi} \in \mathrm{LTL}(\mathsf{X}, \mathsf{U})$ such that for all $w \in \Sigma^\omega$,

$$w, 0 \models \varphi \iff w, 0 \models \overrightarrow{\varphi}$$

Elegant algebraic proof of $\mathrm{LTL}(\mathsf{X}, \mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$ due to Wilke 98.

Expressivity

Theorem (Kamp 68)

$$\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$$

Separation Theorem (Gabbay, Pnueli, Shelah & Stavi 80)

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overleftarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{Y},\mathsf{S})$ and $\overrightarrow{\varphi_i} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$ and $k \geq 0$,

$$w, k \models \varphi \iff w, k \models \bigvee_i \overleftarrow{\varphi_i} \wedge \overrightarrow{\varphi_i}$$

Corollary: $\mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U}) = \mathrm{LTL}(\mathsf{X},\mathsf{U})$

For all $\varphi \in \mathrm{LTL}(\mathsf{Y},\mathsf{S},\mathsf{X},\mathsf{U})$ there exist $\overrightarrow{\varphi} \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$ such that for all $w \in \Sigma^\omega$,

$$w, 0 \models \varphi \iff w, 0 \models \overrightarrow{\varphi}$$

Elegant algebraic proof of $\mathrm{LTL}(\mathsf{X},\mathsf{U}) = \mathrm{FO}_\Sigma(\leq)$ due to Wilke 98.

Satisfiability for LTL

Let AP be the set of atomic propositions and $\Sigma = 2^{\mathrm{AP}}$.

(Initial) Satisfiability problem

Input: A formula $\varphi \in \mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U})$

Question: Existence of $w \in \Sigma^\omega$ such that $w, 0 \models \varphi$.

Theorem (Sistla & Clarke 85, Lichtenstein et. al 85)

The satisfiability problem for LTL is PSPACE-complete

Satisfiability for LTL

Let AP be the set of atomic propositions and $\Sigma = 2^{\mathrm{AP}}$.

(Initial) Satisfiability problem

Input: A formula $\varphi \in \mathrm{LTL}(\mathsf{Y}, \mathsf{S}, \mathsf{X}, \mathsf{U})$

Question: Existence of $w \in \Sigma^\omega$ such that $w, 0 \models \varphi$.

Theorem (Sistla & Clarke 85, Lichtenstein et. al 85)

The satisfiability problem for LTL is PSPACE-complete

Model checking for LTL

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{LTL}$

Question: Does $M \models \varphi$?

- Universal MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for all initial infinite run of M.
- Existential MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for some initial infinite run of M.

Theorem (Sistla & Clarke 85, Lichtenstein et. al 85)

The Model checking problem for LTL is PSPACE-complete

Model checking for LTL

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{LTL}$

Question: Does $M \models \varphi$?

- Universal MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for all initial infinite run of M.
- Existential MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for some initial infinite run of M.

Theorem (Sistla & Clarke 85, Lichtenstein et. al 85)

The Model checking problem for LTL is PSPACE-complete

Model checking for LTL

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{LTL}$

Question: Does $M \models \varphi$?

- Universal MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for all initial infinite run of M.
- Existential MC: $M \models \varphi$ if $\ell(\sigma), 0 \models \varphi$ for some initial infinite run of M.

Theorem (Sistla & Clarke 85, Lichtenstein et. al 85)

The Model checking problem for LTL is PSPACE-complete

$\mathrm{MC}(\mathsf{X},\mathsf{U}) \leq_P \overline{\mathrm{SAT}}(\mathsf{X},\mathsf{U})$ (Sistla & Clarke 85)

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$

Introduce new atomic propositions: $\mathrm{AP}_S = \{\mathrm{at}_s \mid s \in S\}$
Define $\mathrm{AP}' = \mathrm{AP} \uplus \mathrm{AP}_S$ $\quad \Sigma' = 2^{\mathrm{AP}'}$ $\quad \pi : \Sigma'^{\omega} \to \Sigma^{\omega}$ by $\pi(a) = a \cap \mathrm{AP}$.

Let $w \in \Sigma'^{\omega}$. We have $w \models \varphi$ iff $\pi(w) \models \varphi$

Define

$$\psi_M = \left(\bigvee_{s \in I} \mathrm{at}_s\right) \wedge \mathsf{G}\left(\bigvee_{s \in S}\left(\mathrm{at}_s \wedge \bigwedge_{t \neq s} \neg \mathrm{at}_t \wedge \bigwedge_{p \in \ell(s)} p \wedge \bigwedge_{p \notin \ell(s)} \neg p \wedge \bigvee_{t \in T(s)} \mathsf{X}\, \mathrm{at}_t\right)\right)$$

We have $w \models \psi_M$ iff $\pi(w) = \ell(\sigma)$ for some initial infinite run σ of M.

Therefore, $M \not\models \varphi$ iff $\ell(\sigma) \models \neg\varphi$ for some initial infinite run σ of M
iff $w \models \psi_M \wedge \neg\varphi$ for some $w \in \Sigma'^{\omega}$
iff $\psi_M \wedge \neg\varphi$ is satisfiable

$\mathrm{MC}(\mathsf{X}, \mathsf{U}) \leq_P \overline{\mathrm{SAT}}(\mathsf{X}, \mathsf{U})$ (Sistla & Clarke 85)

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{LTL}(\mathsf{X}, \mathsf{U})$

Introduce new atomic propositions: $\mathrm{AP}_S = \{\mathrm{at}_s \mid s \in S\}$
Define $\mathrm{AP}' = \mathrm{AP} \uplus \mathrm{AP}_S$ $\quad \Sigma' = 2^{\mathrm{AP}'}$ $\quad \pi : \Sigma'^\omega \to \Sigma^\omega$ by $\pi(a) = a \cap \mathrm{AP}$.

Let $w \in \Sigma'^\omega$. We have $w \models \varphi$ iff $\pi(w) \models \varphi$

Define

$$\psi_M = \left(\bigvee_{s \in I} \mathrm{at}_s\right) \wedge \mathsf{G}\left(\bigvee_{s \in S}\left(\mathrm{at}_s \wedge \bigwedge_{t \neq s} \neg \mathrm{at}_t \wedge \bigwedge_{p \in \ell(s)} p \wedge \bigwedge_{p \notin \ell(s)} \neg p \wedge \bigvee_{t \in T(s)} \mathsf{X}\, \mathrm{at}_t\right)\right)$$

We have $w \models \psi_M$ iff $\pi(w) = \ell(\sigma)$ for some initial infinite run σ of M.

Therefore, $M \not\models \varphi$ iff $\ell(\sigma) \models \neg\varphi$ for some initial infinite run σ of M
iff $w \models \psi_M \wedge \neg\varphi$ for some $w \in \Sigma'^\omega$
iff $\psi_M \wedge \neg\varphi$ is satisfiable

$\mathrm{MC}(\mathsf{X},\mathsf{U}) \leq_P \overline{\mathrm{SAT}}(\mathsf{X},\mathsf{U})$ **(Sistla & Clarke 85)**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{LTL}(\mathsf{X},\mathsf{U})$

Introduce new atomic propositions: $\mathrm{AP}_S = \{\mathrm{at}_s \mid s \in S\}$
Define $\mathrm{AP}' = \mathrm{AP} \uplus \mathrm{AP}_S$ $\quad \Sigma' = 2^{\mathrm{AP}'} \quad \pi : \Sigma'^\omega \to \Sigma^\omega$ by $\pi(a) = a \cap \mathrm{AP}$.

Let $w \in \Sigma'^\omega$. We have $w \models \varphi$ iff $\pi(w) \models \varphi$

Define

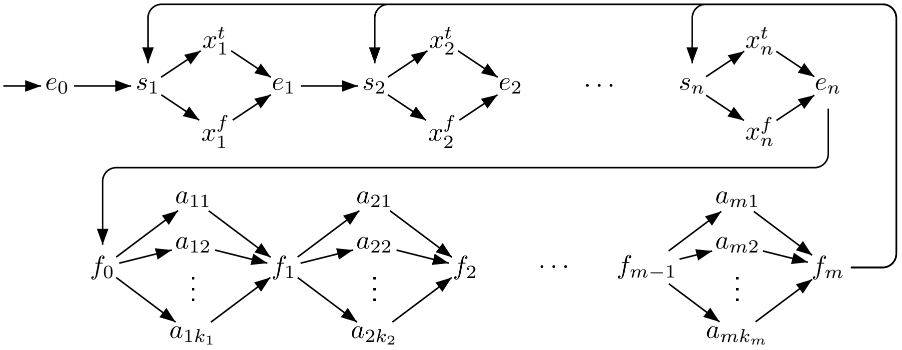
$$\psi_M = \left(\bigvee_{s\in I} \mathrm{at}_s\right) \wedge \mathsf{G}\left(\bigvee_{s\in S}\left(\mathrm{at}_s \wedge \bigwedge_{t\neq s} \neg\mathrm{at}_t \wedge \bigwedge_{p\in\ell(s)} p \wedge \bigwedge_{p\notin\ell(s)} \neg p \wedge \bigvee_{t\in T(s)} \mathsf{X}\,\mathrm{at}_t\right)\right)$$

We have $w \models \psi_M$ iff $\pi(w) = \ell(\sigma)$ for some initial infinite run σ of M.

Therefore, $M \not\models \varphi$ iff $\ell(\sigma) \models \neg\varphi$ for some initial infinite run σ of M
iff $w \models \psi_M \wedge \neg\varphi$ for some $w \in \Sigma'^\omega$
iff $\psi_M \wedge \neg\varphi$ is satisfiable

$\mathrm{QBF} \leq_P \overline{\mathrm{MC}}(\mathsf{X}, \mathsf{U})$ (Sistla & Clarke 85)

Let $\gamma = Q_1 x_1 \cdots Q_n x_n \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq k_i} a_{ij}$ with $Q_i \in \{\forall, \exists\}$ and consider the KS M:



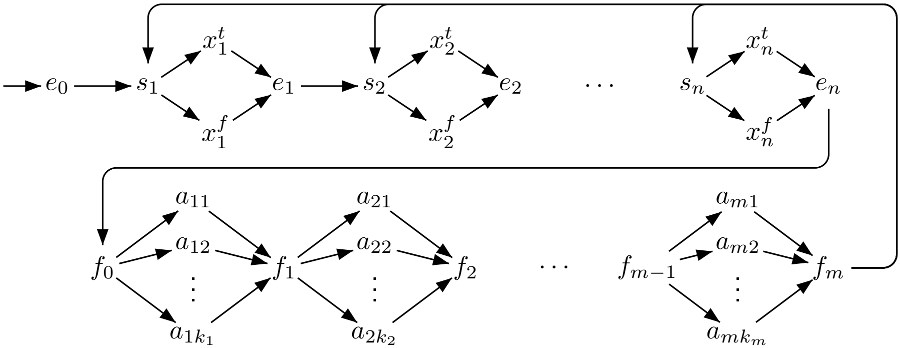
Let $\psi_{ij} = \begin{cases} \mathsf{G}(x_k^f \to \neg a_{ij} \mathbin{\mathsf{W}} s_k) & \text{if } a_{ij} = x_k \\ \mathsf{G}(x_k^t \to \neg a_{ij} \mathbin{\mathsf{W}} s_k) & \text{if } a_{ij} = \neg x_k \end{cases}$ and $\psi = \bigwedge_{i,j} \psi_{ij}$.

Let $\varphi_j = \mathsf{G}(e_{j-1} \to (\neg s_{j-1} \mathbin{\mathsf{U}} x_j^t) \wedge (\neg s_{j-1} \mathbin{\mathsf{U}} x_j^f))$ and $\varphi = \bigwedge_{j \mid Q_j = \forall} \varphi_j$.

Then, γ is valid iff $M \not\models \neg(\varphi \wedge \psi)$ iff $\sigma \models \varphi \wedge \psi$ for some run σ.

$\mathrm{QBF} \leq_P \overline{\mathrm{MC}}(\mathsf{X},\mathsf{U})$ (Sistla & Clarke 85)

Let $\gamma = Q_1 x_1 \cdots Q_n x_n \bigwedge_{1\leq i\leq m} \bigvee_{1\leq j\leq k_i} a_{ij}$ with $Q_i \in \{\forall, \exists\}$ and consider the KS M:



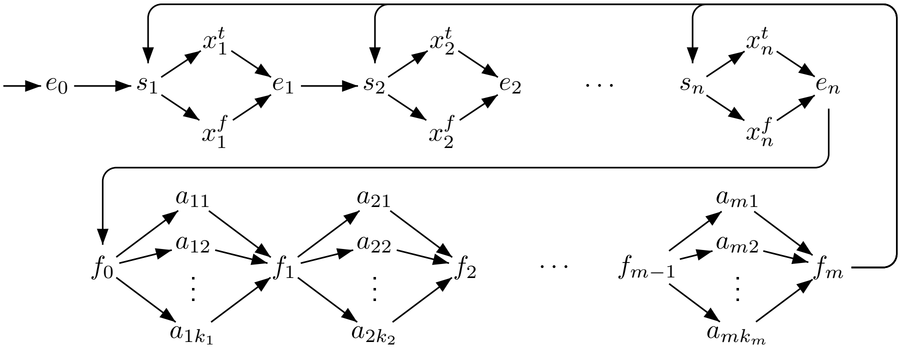
Let $\psi_{ij} = \begin{cases} \mathsf{G}(x_k^f \to \neg a_{ij} \mathsf{W} s_k) & \text{if } a_{ij} = x_k \\ \mathsf{G}(x_k^t \to \neg a_{ij} \mathsf{W} s_k) & \text{if } a_{ij} = \neg x_k \end{cases}$ and $\psi = \bigwedge_{i,j} \psi_{ij}$.

Let $\varphi_j = \mathsf{G}(e_{j-1} \to (\neg s_{j-1} \mathsf{U} x_j^t) \wedge (\neg s_{j-1} \mathsf{U} x_j^f))$ and $\varphi = \bigwedge_{j \mid Q_j = \forall} \varphi_j$.

Then, γ is valid iff $M \not\models \neg(\varphi \wedge \psi)$ iff $\sigma \models \varphi \wedge \psi$ for some run σ.

$\mathrm{QBF} \leq_P \overline{\mathrm{MC}}(\mathsf{X}, \mathsf{U})$ (Sistla & Clarke 85)

Let $\gamma = Q_1 x_1 \cdots Q_n x_n \bigwedge_{1 \leq i \leq m} \bigvee_{1 \leq j \leq k_i} a_{ij}$ with $Q_i \in \{\forall, \exists\}$ and consider the KS M:



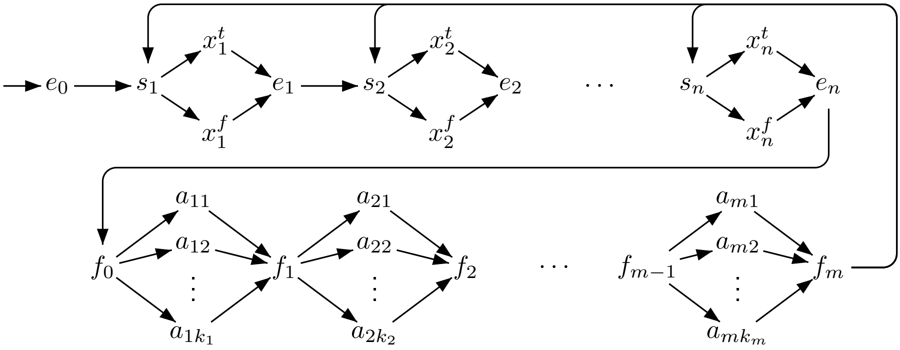
Let $\psi_{ij} = \begin{cases} \mathsf{G}(x_k^f \to \neg a_{ij} \mathsf{W} s_k) & \text{if } a_{ij} = x_k \\ \mathsf{G}(x_k^t \to \neg a_{ij} \mathsf{W} s_k) & \text{if } a_{ij} = \neg x_k \end{cases}$ and $\psi = \bigwedge_{i,j} \psi_{ij}$.

Let $\varphi_j = \mathsf{G}(e_{j-1} \to (\neg s_{j-1} \mathsf{U} x_j^t) \wedge (\neg s_{j-1} \mathsf{U} x_j^f))$ and $\varphi = \bigwedge_{j \mid Q_j = \forall} \varphi_j$.

Then, γ is valid iff $M \not\models \neg(\varphi \wedge \psi)$ iff $\sigma \models \varphi \wedge \psi$ for some run σ.

$\mathrm{QBF} \leq_P \overline{\mathrm{MC}}(\mathsf{X}, \mathsf{U})$ **(Sistla & Clarke 85)**

Let $\gamma = Q_1 x_1 \cdots Q_n x_n \bigwedge_{1\leq i\leq m} \bigvee_{1\leq j\leq k_i} a_{ij}$ with $Q_i \in \{\forall, \exists\}$ and consider the KS M:



Let $\psi_{ij} = \begin{cases} \mathsf{G}(x_k^f \to \neg a_{ij} \mathrel{\mathsf{W}} s_k) & \text{if } a_{ij} = x_k \\ \mathsf{G}(x_k^t \to \neg a_{ij} \mathrel{\mathsf{W}} s_k) & \text{if } a_{ij} = \neg x_k \end{cases}$ and $\psi = \bigwedge_{i,j} \psi_{ij}$.

Let $\varphi_j = \mathsf{G}(e_{j-1} \to (\neg s_{j-1} \mathrel{\mathsf{U}} x_j^t) \wedge (\neg s_{j-1} \mathrel{\mathsf{U}} x_j^f))$ and $\varphi = \bigwedge_{j \mid Q_j = \forall} \varphi_j$.

Then, γ is valid iff $M \not\models \neg(\varphi \wedge \psi)$ iff $\sigma \models \varphi \wedge \psi$ for some run σ.

Decision procedure for LTL

The core

From an LTL formula φ, construct a Büchi automaton $\mathcal{A}_\varphi$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w, 0 \models \varphi\}.$$

Satisfiability (initial)

Check the Büchi automaton $\mathcal{A}_\varphi$ for emptiness.

Model checking

Construct the product $\mathcal{B} = M \times \mathcal{A}_{\neg\varphi}$ so that the successful runs of $\mathcal{B}$ correspond to the successful run of $\mathcal{A}$ satisfying $\neg\varphi$.

Then, check $\mathcal{B}$ for emptiness.

Decision procedure for LTL

The core

From an LTL formula φ, construct a Büchi automaton $\mathcal{A}_\varphi$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w, 0 \models \varphi\}.$$

Satisfiability (initial)

Check the Büchi automaton $\mathcal{A}_\varphi$ for emptiness.

Model checking

Construct the product $\mathcal{B} = M \times \mathcal{A}_{\neg\varphi}$ so that the successful runs of $\mathcal{B}$ correspond to the successful run of $\mathcal{A}$ satisfying $\neg\varphi$.

Then, check $\mathcal{B}$ for emptiness.

Decision procedure for LTL

The core

From an LTL formula φ, construct a Büchi automaton $\mathcal{A}_\varphi$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w, 0 \models \varphi\}.$$

Satisfiability (initial)

Check the Büchi automaton $\mathcal{A}_\varphi$ for emptiness.

Model checking

Construct the product $\mathcal{B} = M \times \mathcal{A}_{\neg\varphi}$ so that the successful runs of $\mathcal{B}$ correspond to the successful run of $\mathcal{A}$ satisfying $\neg\varphi$.

Then, check $\mathcal{B}$ for emptiness.

Büchi automata

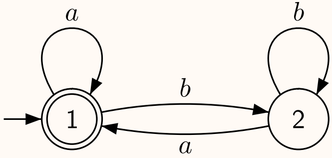
Definition

$\mathcal{A} = (Q, \Sigma, I, T, F)$ where

- Q: finite set of states
- Σ: finite set of labels
- $I \subseteq Q$: set of initial states
- $T \subseteq Q \times \Sigma \times Q$: transitions
- $F \subseteq Q$: set of accepting states (repeated, final)

Example

$\mathcal{A} =$



$$\mathcal{L}(\mathcal{A}) = \{w \in \{a,b\}^\omega \mid |w|_a = \omega\}$$

Büchi automata

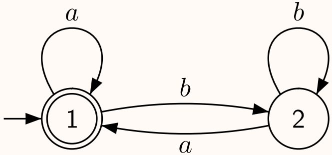
Definition

$\mathcal{A} = (Q, \Sigma, I, T, F)$ where

- Q: finite set of states
- Σ: finite set of labels
- $I \subseteq Q$: set of initial states
- $T \subseteq Q \times \Sigma \times Q$: transitions
- $F \subseteq Q$: set of accepting states (repeated, final)

Example

$\mathcal{A} =$



$$\mathcal{L}(\mathcal{A}) = \{w \in \{a,b\}^\omega \mid |w|_a = \omega\}$$

Büchi automata for some LTL formulas

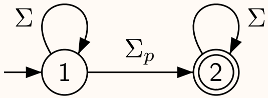
Definition

Recall that $\Sigma = 2^{\mathrm{AP}}$. For $p, q \in \mathrm{AP}$, we let

- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ $\quad\dots$

Examples

F p:



or

X X p:

G p:

Büchi automata for some LTL formulas

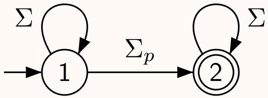
Definition

Recall that $\Sigma = 2^{\mathrm{AP}}$. For $p, q \in \mathrm{AP}$, we let

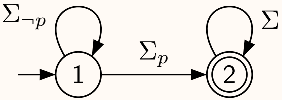
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ $\quad \ldots$

Examples

F p:



or



X X p:

G p:

Büchi automata for some LTL formulas

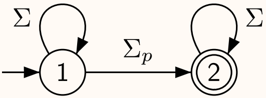
Definition

Recall that $\Sigma = 2^{\mathrm{AP}}$. For $p, q \in \mathrm{AP}$, we let

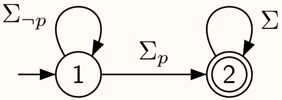
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ ...

Examples

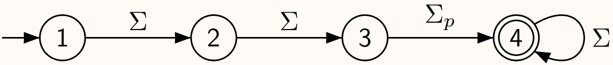
F p:



or



X X p:



G p:

Büchi automata for some LTL formulas

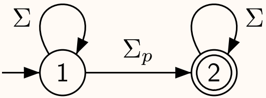
Definition

Recall that $\Sigma = 2^{\mathrm{AP}}$. For $p, q \in \mathrm{AP}$, we let

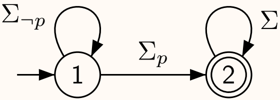
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ $\ldots$

Examples

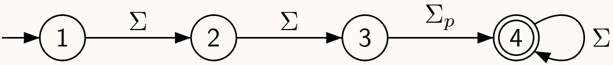
$\mathsf{F}\,p$:



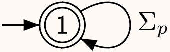
or



$\mathsf{X}\,\mathsf{X}\,p$:



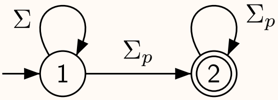
$\mathsf{G}\,p$:



Büchi automata for some LTL formulas

Examples

$\mathsf{F}\,\mathsf{G}\,p$:



no deterministic Büchi automaton.

$\mathsf{G}\,\mathsf{F}\,p$:

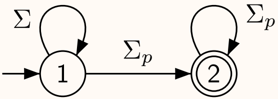
deterministic Büchi automaton
are not closed under complement.

$\mathsf{G}(p \to \mathsf{F}\,q)$:

Büchi automata for some LTL formulas

Examples

$\mathsf{F}\,\mathsf{G}\,p$:



no deterministic Büchi automaton.

$\mathsf{G}\,\mathsf{F}\,p$:

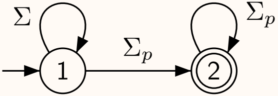
deterministic Büchi automaton
are not closed under complement.

$\mathsf{G}(p \to \mathsf{F}\,q)$:

Büchi automata for some LTL formulas

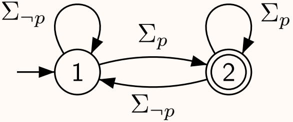
Examples

$\mathsf{F}\,\mathsf{G}\,p$:



no deterministic Büchi automaton.

$\mathsf{G}\,\mathsf{F}\,p$:



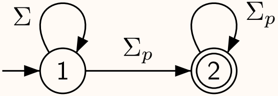
deterministic Büchi automaton
are not closed under complement.

$\mathsf{G}(p \to \mathsf{F}\,q)$:

Büchi automata for some LTL formulas

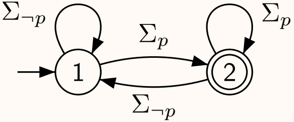
Examples

$\mathsf{F}\,\mathsf{G}\,p$:



no deterministic Büchi automaton.

$\mathsf{G}\,\mathsf{F}\,p$:



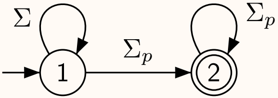
deterministic Büchi automaton
are not closed under complement.

$\mathsf{G}(p \to \mathsf{F}\,q)$:

Büchi automata for some LTL formulas

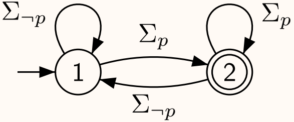
Examples

$\mathsf{F}\,\mathsf{G}\,p$:



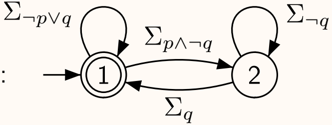
no deterministic Büchi automaton.

$\mathsf{G}\,\mathsf{F}\,p$:



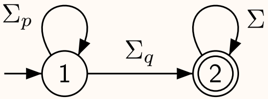
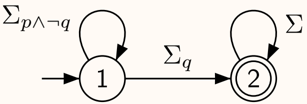
deterministic Büchi automaton
are not closed under complement.

$\mathsf{G}(p \to \mathsf{F}\,q)$:



Büchi automata for some LTL formulas

Examples

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

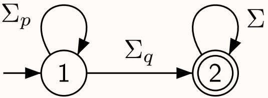
$p \,\mathsf{W}\, q$: or

$p \,\mathsf{R}\, q$: or

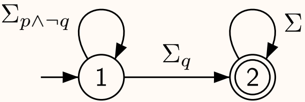
Büchi automata for some LTL formulas

Examples

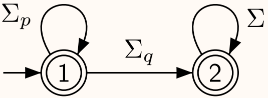
p U q:



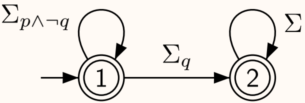
or



p W q:



or



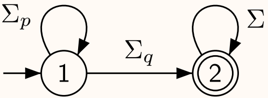
p R q:

or

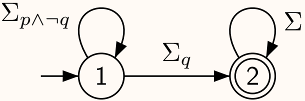
Büchi automata for some LTL formulas

Examples

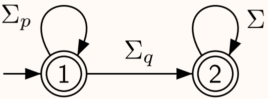
p U q:



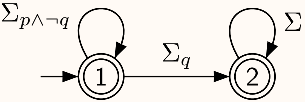
or



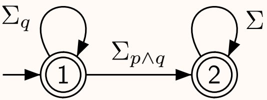
p W q:



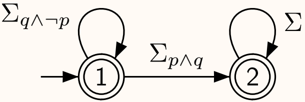
or



p R q:



or



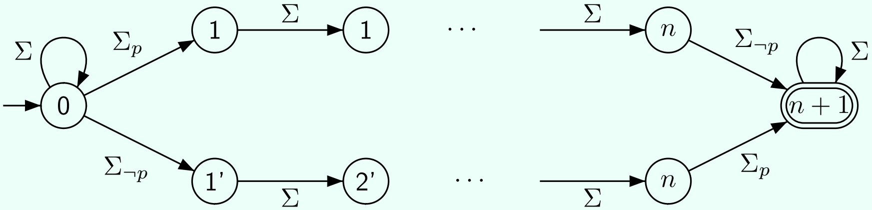
Büchi automata

Properties

Büchi automata are closed under union, intersection, complement.

- Union: trivial
- Intersection: easy (exercice)
- complement: hard

  Let $\varphi = \mathsf{F}((p \wedge \mathsf{X}^n \neg p) \vee (\neg p \wedge \mathsf{X}^n p))$



Any non deterministic Büchi automaton for $\neg\varphi$ has at least 2^n states.

Büchi automata

Exercice

Given Büchi automata for φ and ψ,

- Construct a Büchi automaton for $\mathsf{X}\,\varphi$ (trivial)
- Construct a Büchi automaton for $\varphi\,\mathsf{U}\,\psi$

This gives an inductive construction of $\mathcal{A}_\varphi$ from $\varphi \in \mathrm{LTL}(\mathrm{X}, \mathrm{U})$ ...

... but the size of $\mathcal{A}_\varphi$ might be non-elementary in the size of φ.

Büchi automata

Exercice

Given Büchi automata for φ and ψ,

- Construct a Büchi automaton for $\mathsf{X}\,\varphi$ (trivial)
- Construct a Büchi automaton for $\varphi\,\mathsf{U}\,\psi$

This gives an inductive construction of $\mathcal{A}_\varphi$ from $\varphi \in \mathrm{LTL}(\mathsf{X}, \mathsf{U})$ ...

... but the size of $\mathcal{A}_\varphi$ might be non-elementary in the size of φ.

Büchi automata

Exercice

Given Büchi automata for φ and ψ,

- Construct a Büchi automaton for $\mathsf{X}\,\varphi$ (trivial)
- Construct a Büchi automaton for $\varphi\,\mathsf{U}\,\psi$

This gives an inductive construction of $\mathcal{A}_\varphi$ from $\varphi \in \mathrm{LTL}(\mathsf{X}, \mathsf{U})$ ...

... but the size of $\mathcal{A}_\varphi$ might be non-elementary in the size of φ.

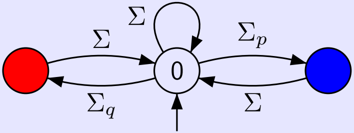
Generalized Büchi automata

Definition: acceptance on states

$\mathcal{A} = (Q, \Sigma, I, T, F_1, \ldots, F_n)$ with $F_i \subseteq Q$.

An infinite run σ is successful if it visits infinitely often each F_i.

$\mathsf{G}\,\mathsf{F}\,p \wedge \mathsf{G}\,\mathsf{F}\,q$:



Definition: acceptance on transitions

$\mathcal{A} = (Q, \Sigma, I, T, T_1, \ldots, T_n)$ with $T_i \subseteq T$.

An infinite run σ is successful if it uses infinitely many transitions from each T_i.

$\mathsf{G}\,\mathsf{F}\,p \wedge \mathsf{G}\,\mathsf{F}\,q$:

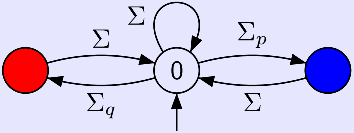
Generalized Büchi automata

Definition: acceptance on states

$\mathcal{A} = (Q, \Sigma, I, T, F_1, \ldots, F_n)$ with $F_i \subseteq Q$.

An infinite run σ is successful if it visits infinitely often each F_i.

$\mathsf{G\,F}\,p \wedge \mathsf{G\,F}\,q$:

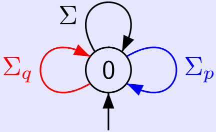


Definition: acceptance on transitions

$\mathcal{A} = (Q, \Sigma, I, T, T_1, \ldots, T_n)$ with $T_i \subseteq T$.

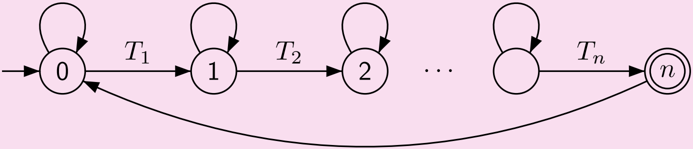
An infinite run σ is successful if it uses infinitely many transitions from each T_i.

$\mathsf{G\,F}\,p \wedge \mathsf{G\,F}\,q$:



GBA to BA

Synchronized product with



Negative normal form

Syntax ($p \in \mathrm{AP}$)

$$\varphi ::= \bot \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{R}\, \varphi$$

Any formula can be transformed in NNF

- $\neg \mathsf{X}\,\varphi = \mathsf{X}\,\neg\varphi$
- $\neg(\varphi \,\mathsf{U}\, \psi) = (\neg\varphi) \,\mathsf{R}\, (\neg\psi)$
- $\neg(\varphi \,\mathsf{R}\, \psi) = (\neg\varphi) \,\mathsf{U}\, (\neg\psi)$
- $\neg(\varphi \vee \psi) = (\neg\varphi) \wedge (\neg\psi)$
- $\neg(\varphi \wedge \psi) = (\neg\varphi) \vee (\neg\psi)$

Note that this does not increase the number of Temporal subformulas.

Negative normal form

Syntax ($p \in \mathrm{AP}$)

$$\varphi ::= \bot \mid p \mid \neg p \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \varphi\,\mathsf{R}\,\varphi$$

Any formula can be transformed in NNF

- $\neg\,\mathsf{X}\,\varphi = \mathsf{X}\,\neg\varphi$
- $\neg(\varphi\,\mathsf{U}\,\psi) = (\neg\varphi)\,\mathsf{R}\,(\neg\psi)$
- $\neg(\varphi\,\mathsf{R}\,\psi) = (\neg\varphi)\,\mathsf{U}\,(\neg\psi)$
- $\neg(\varphi \vee \psi) = (\neg\varphi) \wedge (\neg\psi)$
- $\neg(\varphi \wedge \psi) = (\neg\varphi) \vee (\neg\psi)$

Note that this does not increase the number of Temporal subformulas.

Reduction graph

Definition

$Z \subseteq \mathrm{NNF}$ is reduced if

- formulas in Z are of the form p, $\neg p$, or $\mathsf{X}\,\beta$,
- $\bot \notin Z$ and $\{p, \neg p\} \not\subseteq Z$ for all $p \in \mathrm{AP}$.

Reduction graph

- Vertices: subsets of NNF
- Edges: Let $Y \subseteq NNF$ and let $\alpha \in Y$ maximal not reduced.

If $\alpha = \alpha_1 \vee \alpha_2$:
$$Y \to Y \setminus \{\alpha\} \cup \{\alpha_1\}, \quad Y \to Y \setminus \{\alpha\} \cup \{\alpha_2\},$$

If $\alpha = \alpha_1 \wedge \alpha_2$:
$$Y \to Y \setminus \{\alpha\} \cup \{\alpha_1, \alpha_2\},$$

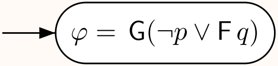
If $\alpha = \alpha_1 \mathbin{\mathsf{R}} \alpha_2$:
$$Y \to Y \setminus \{\alpha\} \cup \{\alpha_1, \alpha_2\}, \quad Y \to Y \setminus \{\alpha\} \cup \{\alpha_2, \mathsf{X}\,\alpha\},$$

If $\alpha = \alpha_1 \mathbin{\mathsf{U}} \alpha_2$:
$$Y \to Y \setminus \{\alpha\} \cup \{\alpha_2\}, \quad Y \xrightarrow{\alpha} Y \setminus \{\alpha\} \cup \{\alpha_1, \mathsf{X}\,\alpha\}.$$

Note the mark α on the last edge

Reduction graph

Example: $\varphi = \mathsf{G}(p \rightarrow \mathsf{F}\, q)$



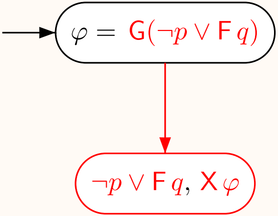
State = set of obligations.

Reduce obligations to litterals and next-formulas.

Note again the mark $\mathsf{F}\, q$ on the last edge

Reduction graph

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$



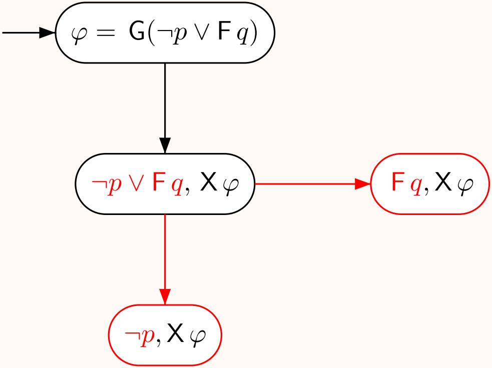
State = set of obligations.

Reduce obligations to litterals and next-formulas.

Note again the mark $\mathsf{F}\, q$ on the last edge

Reduction graph

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$



State = set of obligations.

Reduce obligations to litterals and next-formulas.

Note again the mark $\mathsf{F}\, q$ on the last edge

Reduction graph

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

$$\begin{tikzcd}
\arrow[r] & \varphi = \mathsf{G}(\neg p \vee \mathsf{F}\, q) \arrow[d] & q, \mathsf{X}\,\varphi \\
& \neg p \vee \mathsf{F}\, q,\ \mathsf{X}\,\varphi \arrow[r] \arrow[d] & \mathsf{F}\, q, \mathsf{X}\,\varphi \arrow[u] \arrow[d, "\mathsf{F}\, q"] \\
& \neg p, \mathsf{X}\,\varphi & \mathsf{X}\,\mathsf{F}\, q, \mathsf{X}\,\varphi
\end{tikzcd}$$

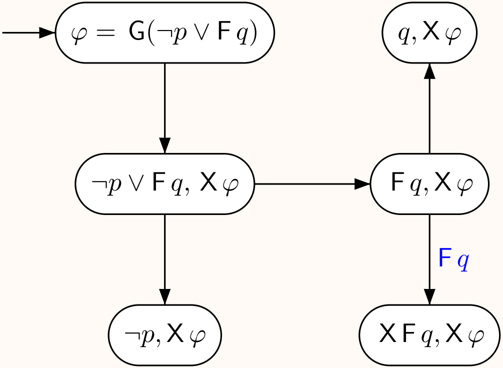
State = set of obligations.

Reduce obligations to litterals and next-formulas.

Note again the mark $\mathsf{F}\, q$ on the last edge

Reduction graph

Example: $\varphi = \mathsf{G}(p \rightarrow \mathsf{F}\, q)$



State = set of obligations.

Reduce obligations to litterals and next-formulas.

Note again the mark $\mathsf{F}\, q$ on the last edge

Automaton $\mathcal{A}_\varphi$

Definition: For $Y \subseteq \mathrm{NNF}$, let

- $\mathrm{Red}(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z\}$
- $\mathrm{Red}_\alpha(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z$ without using an edge marked with $\alpha\}$

Definition: For $Z \subseteq \mathrm{NNF}$ reduced, define

- $\mathrm{next}(Z) = \{\alpha \mid \mathsf{X}\,\alpha \in Z\}$
- $\Sigma_Z = \bigcap_{p \in Z} \Sigma_p \quad \cap \quad \bigcap_{\neg p \in Z} \Sigma_{\neg p}$

Automaton $\mathcal{A}_\varphi$

- States: $Q = 2^{\mathrm{sub}(\varphi)}$, $\qquad I = \{\varphi\}$
- Transitions: $T = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}(Y)\}$
- Acceptance: $T_\alpha = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}_\alpha(Y)\}$
  for each $\alpha = \alpha_1 \,\mathsf{U}\, \alpha_2 \in \mathrm{sub}(\varphi)$.

Automaton $\mathcal{A}_\varphi$

Definition: For $Y \subseteq \mathrm{NNF}$, let

- $\mathrm{Red}(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z\}$
- $\mathrm{Red}_\alpha(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z$ without using an edge marked with $\alpha\}$

Definition: For $Z \subseteq \mathrm{NNF}$ reduced, define

- $\mathrm{next}(Z) = \{\alpha \mid \mathsf{X}\,\alpha \in Z\}$
- $\Sigma_Z = \bigcap_{p \in Z} \Sigma_p \quad \cap \quad \bigcap_{\neg p \in Z} \Sigma_{\neg p}$

Automaton $\mathcal{A}_\varphi$

- States: $Q = 2^{\mathrm{sub}(\varphi)}, \qquad I = \{\varphi\}$
- Transitions: $T = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}(Y)\}$
- Acceptance: $T_\alpha = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}_\alpha(Y)\}$ for each $\alpha = \alpha_1 \,\mathsf{U}\, \alpha_2 \in \mathrm{sub}(\varphi)$.

Automaton $\mathcal{A}_\varphi$

Definition: For $Y \subseteq \mathrm{NNF}$, let

- $\mathrm{Red}(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z\}$
- $\mathrm{Red}_\alpha(Y) = \{Z$ reduced $\mid Y \xrightarrow{*} Z$ without using an edge marked with $\alpha\}$

Definition: For $Z \subseteq \mathrm{NNF}$ reduced, define

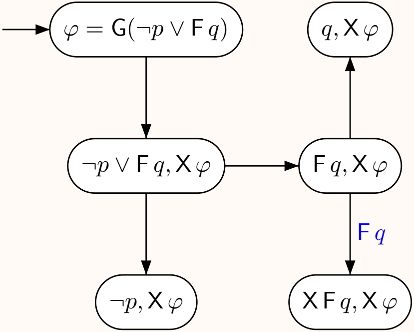
- $\mathrm{next}(Z) = \{\alpha \mid \mathsf{X}\,\alpha \in Z\}$
- $\Sigma_Z = \bigcap_{p \in Z} \Sigma_p \quad \cap \quad \bigcap_{\neg p \in Z} \Sigma_{\neg p}$

Automaton $\mathcal{A}_\varphi$

- States: $Q = 2^{\mathrm{sub}(\varphi)}$, $\qquad I = \{\varphi\}$
- Transitions: $T = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}(Y)\}$
- Acceptance: $T_\alpha = \{Y \xrightarrow{\Sigma_Z} \mathrm{next}(Z) \mid Y \in Q$ and $Z \in \mathrm{Red}_\alpha(Y)\}$
  for each $\alpha = \alpha_1 \,\mathsf{U}\, \alpha_2 \in \mathrm{sub}(\varphi)$.

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

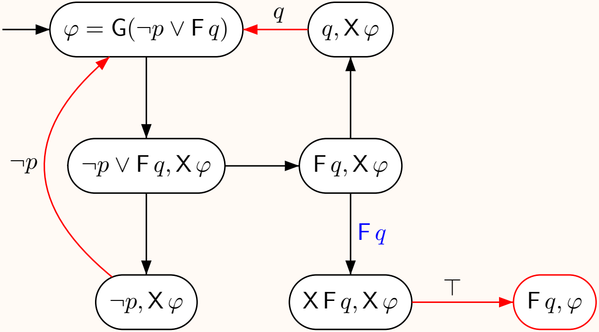


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

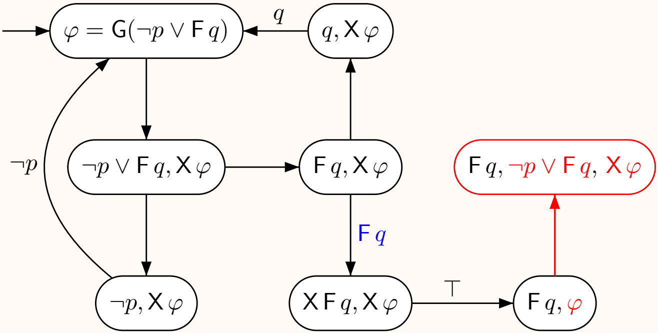


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

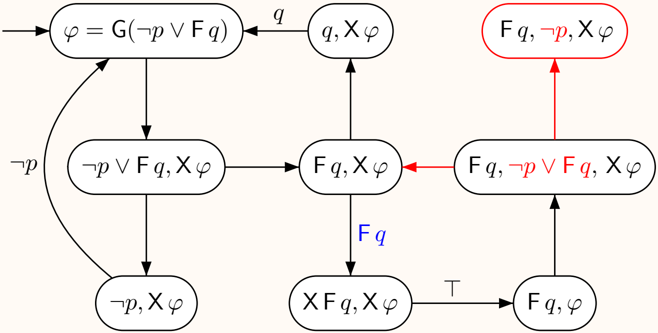


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

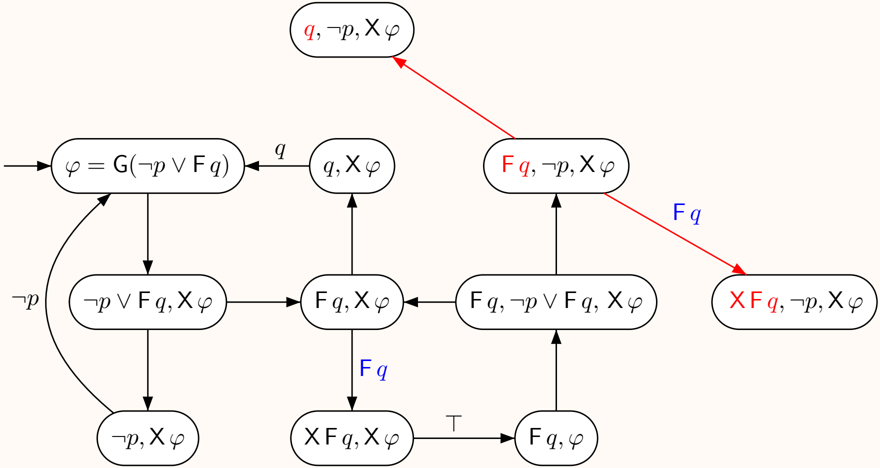


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \rightarrow \mathsf{F}\, q)$

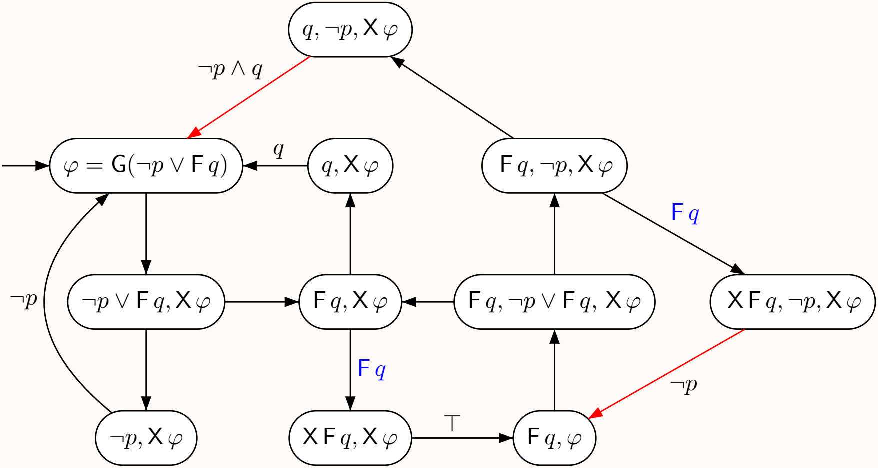


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

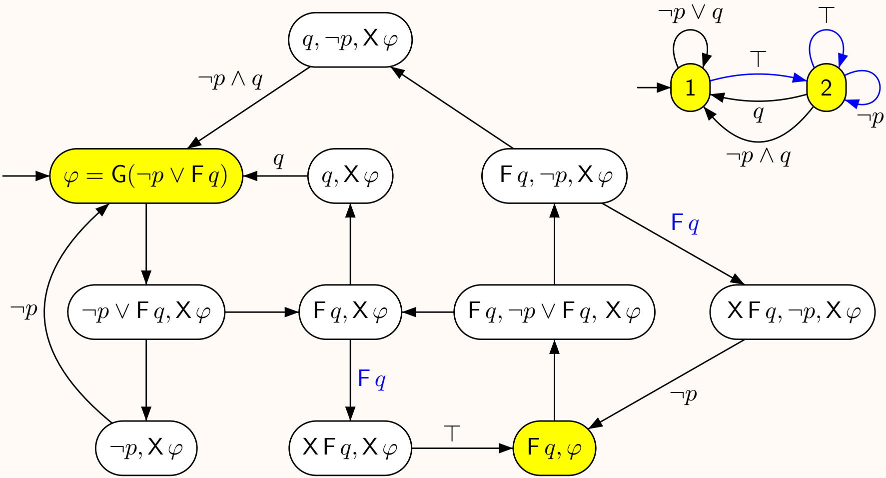


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \to \mathsf{F}\, q)$

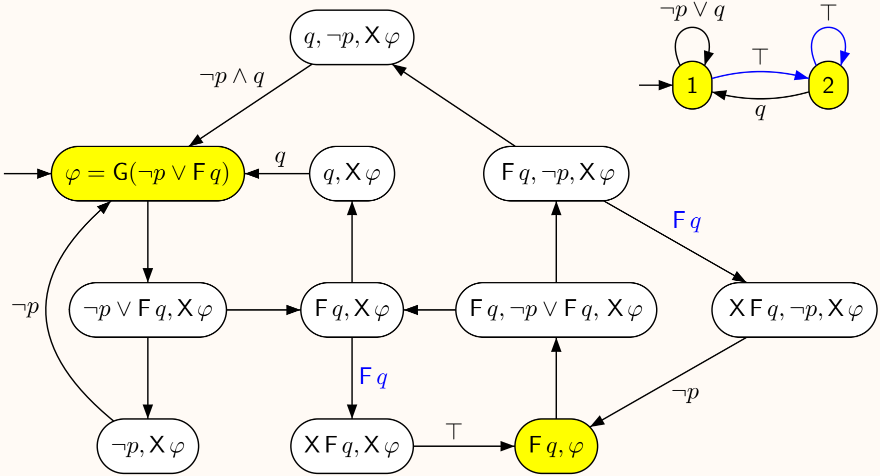


Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Example: $\varphi = \mathsf{G}(p \rightarrow \mathsf{F}\, q)$



Transition = check litterals and move forward.

Simplification

Automaton $\mathcal{A}_\varphi$

Theorem $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$

- $|Q| \leq 2^{|\varphi|}$
- number of acceptance tables $=$ number of until sub-formulas.

Corollary

Satisfiability and Model Checking are decidable in PSPACE.

Remark

An efficient construction is based on Very Weak Alternating Automata.
(Gastin & Oddoux, CAV'01)

The domain is still very active.

Automaton $\mathcal{A}_\varphi$

Theorem $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$

- $|Q| \leq 2^{|\varphi|}$
- number of acceptance tables $=$ number of until sub-formulas.

Corollary

Satisfiability and Model Checking are decidable in PSPACE.

Remark

An efficient construction is based on Very Weak Alternating Automata.
(Gastin & Oddoux, CAV'01)

The domain is still very active.

Automaton $\mathcal{A}_\varphi$

Theorem $\mathcal{L}(\mathcal{A}_\varphi) = \mathcal{L}(\varphi)$

- $|Q| \leq 2^{|\varphi|}$
- number of acceptance tables $=$ number of until sub-formulas.

Corollary

Satisfiability and Model Checking are decidable in PSPACE.

Remark

An efficient construction is based on Very Weak Alternating Automata.
(Gastin & Oddoux, CAV'01)

The domain is still very active.

Original References

- Sistla & Clarke 85. Complexity of propositional temporal logics. JACM 32(3), p. 733–749.
- Lichtenstein & Pnueli 85. Checking that finite state concurrent programs satisfy their linear specification. ACM Symp. PoPL'85, p. 97–107.
- Gabbay, Pnueli, Shelah & Stavi 80. On the temporal analysis of fairness. ACM Symp. PoPL'80, p. 163–173.
- Gabbay 87. The declarative past and imperative future: Executable temporal logics for interactive systems. conf. on Temporal Logics in Specifications, April 87. LNCS 398, p. 409–448, 1989.

Outline

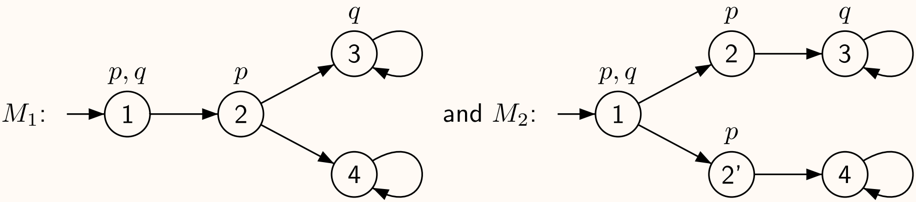
Possibility is not expressible in LTL

Example

φ: Whenever p holds, it is possible to reach a state where q holds.

φ cannot be expressed in LTL.

Consider the two models:



$M_1 \models \varphi$ but $M_2 \not\models \varphi$

M_1 and M_2 satisfy the same LTL formulas.

Quantification on runs

Example

φ: Whenever p holds, it is possible to reach a state where q holds.

$$\varphi = \mathsf{AG}(p \to \mathsf{EF}\, q)$$

- E: for some infinite run
- A: for all infinite run

Some specifications

- $\mathsf{EF}\,\varphi$: φ is possible
- $\mathsf{AG}\,\varphi$: φ is an invariant
- $\mathsf{AF}\,\varphi$: φ is unavoidable
- $\mathsf{EG}\,\varphi$: φ holds globally along some path

Quantification on runs

Example

φ: Whenever p holds, it is possible to reach a state where q holds.

$$\varphi = \mathsf{AG}(p \rightarrow \mathsf{EF}\, q)$$

- E: for some infinite run
- A: for all infinite run

Some specifications

- $\mathsf{EF}\,\varphi$: φ is possible
- $\mathsf{AG}\,\varphi$: φ is an invariant
- $\mathsf{AF}\,\varphi$: φ is unavoidable
- $\mathsf{EG}\,\varphi$: φ holds globally along some path

CTL^* (Emerson & Halpern 86)

Syntax: CTL^*: Computation Tree Logic

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$$

Semantics:

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and σ an infinte run of M.

$\sigma, i \models \mathsf{E}\varphi$ if $\sigma', 0 \models \varphi$ for some infinite run σ' such that $\sigma'(0) = \sigma(i)$

$\sigma, i \models \mathsf{A}\varphi$ if $\sigma', 0 \models \varphi$ for all infinite runs σ' such that $\sigma'(0) = \sigma(i)$

State formulas

A formula of the form p or $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ only depends on the current state.

State formulas are closed under boolean connectives.

If φ is a state formula, define $S(\varphi) = \{s \in S \mid s \models \varphi\}$

CTL^* (Emerson & Halpern 86)

Syntax: CTL^*: Computation Tree Logic

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$$

Semantics:

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and σ an infinte run of M.

$\sigma, i \models \mathsf{E}\varphi$ if $\sigma', 0 \models \varphi$ for some infinite run σ' such that $\sigma'(0) = \sigma(i)$

$\sigma, i \models \mathsf{A}\varphi$ if $\sigma', 0 \models \varphi$ for all infinite runs σ' such that $\sigma'(0) = \sigma(i)$

State formulas

A formula of the form p or $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ only depends on the current state.

State formulas are closed under boolean connectives.

If φ is a state formula, define $S(\varphi) = \{s \in S \mid s \models \varphi\}$

CTL^* (Emerson & Halpern 86)

Syntax: CTL^*: Computation Tree Logic

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{X}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$$

Semantics:

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and σ an infinte run of M.

$\sigma, i \models \mathsf{E}\varphi$ if $\sigma', 0 \models \varphi$ for some infinite run σ' such that $\sigma'(0) = \sigma(i)$

$\sigma, i \models \mathsf{A}\varphi$ if $\sigma', 0 \models \varphi$ for all infinite runs σ' such that $\sigma'(0) = \sigma(i)$

State formulas

A formula of the form p or $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ only depends on the current state.

State formulas are closed under boolean connectives.

If φ is a state formula, define $S(\varphi) = \{s \in S \mid s \models \varphi\}$

Model checking of CTL^*

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}^*$

Question: Does $M \models \varphi$?

Remark

$M \models \varphi$ iff $\ell(\sigma), 0 \models \varphi$ for all initial infinite run of M.

iff $I \subseteq S(\mathsf{A}\,\varphi)$

Theorem

The model checking problem for CTL^* is PSPACE-complete

Proof

PSPACE-hardness: follows from $\mathrm{LTL} \subseteq \mathrm{CTL}^*$.

PSPACE-easiness: inductively compute $S(\psi)$ for all state formulas.

Model checking of CTL^*

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}^*$

Question: Does $M \models \varphi$?

Remark

$M \models \varphi$ iff $\ell(\sigma), 0 \models \varphi$ for all initial infinite run of M.

iff $I \subseteq S(\mathsf{A}\,\varphi)$

Theorem

The model checking problem for CTL^* is PSPACE-complete

Proof

PSPACE-hardness: follows from $\mathrm{LTL} \subseteq \mathrm{CTL}^*$.

PSPACE-easiness: inductively compute $S(\psi)$ for all state formulas.

Computing $S(\psi)$

State formulas

- $S(p) = \{s \in S \mid p \in \ell(s)\}$,
- $S(\neg\psi) = S \setminus S(\psi)$,
- $S(\psi_1 \wedge \psi_2) = S(\psi_1) \cap S(\psi_2)$,
- $S(\psi_1 \vee \psi_2) = S(\psi_1) \cup S(\psi_2)$,
- $S(\mathsf{E}\,\psi) = ?$

  Compute $\mathcal{A}_\psi$, replacing state subformulas of ψ by new atomic propositions.

  To check whether $s \in S(\mathsf{E}\,\psi)$, check for emptiness the synchronized product of $\mathcal{A}_\psi$ and M with initial state s.
- $\mathsf{A}\,\psi = \neg\,\mathsf{E}\,\neg\psi$

Model checking

$$M \models \varphi \text{ iff } I \subseteq S(\mathsf{A}\,\varphi).$$

Computing $S(\psi)$

State formulas

- $S(p) = \{s \in S \mid p \in \ell(s)\}$,
- $S(\neg\psi) = S \setminus S(\psi)$,
- $S(\psi_1 \wedge \psi_2) = S(\psi_1) \cap S(\psi_2)$,
- $S(\psi_1 \vee \psi_2) = S(\psi_1) \cup S(\psi_2)$,
- $S(\mathsf{E}\,\psi) = ?$

  Compute $\mathcal{A}_\psi$, replacing state subformulas of ψ by new atomic propositions.

  To check whether $s \in S(\mathsf{E}\,\psi)$, check for emptiness the synchronized product of $\mathcal{A}_\psi$ and M with initial state s.
- $\mathsf{A}\,\psi = \neg\,\mathsf{E}\,\neg\psi$

Model checking

$$M \models \varphi \text{ iff } I \subseteq S(\mathsf{A}\,\varphi).$$

Computing $S(\psi)$

State formulas

- $S(p) = \{s \in S \mid p \in \ell(s)\}$,
- $S(\neg\psi) = S \setminus S(\psi)$,
- $S(\psi_1 \wedge \psi_2) = S(\psi_1) \cap S(\psi_2)$,
- $S(\psi_1 \vee \psi_2) = S(\psi_1) \cup S(\psi_2)$,
- $S(\mathsf{E}\,\psi) = ?$

  Compute $\mathcal{A}_\psi$, replacing state subformulas of ψ by new atomic propositions.

  To check whether $s \in S(\mathsf{E}\,\psi)$, check for emptiness the synchronized product of $\mathcal{A}_\psi$ and M with initial state s.
- $\mathsf{A}\,\psi = \neg\,\mathsf{E}\,\neg\psi$

Model checking

$$M \models \varphi \text{ iff } I \subseteq S(\mathsf{A}\,\varphi).$$

Computing $S(\psi)$

State formulas

- $S(p) = \{s \in S \mid p \in \ell(s)\}$,
- $S(\neg\psi) = S \setminus S(\psi)$,
- $S(\psi_1 \wedge \psi_2) = S(\psi_1) \cap S(\psi_2)$,
- $S(\psi_1 \vee \psi_2) = S(\psi_1) \cup S(\psi_2)$,
- $S(\mathsf{E}\,\psi) = ?$

  Compute $\mathcal{A}_\psi$, replacing state subformulas of ψ by new atomic propositions.

  To check whether $s \in S(\mathsf{E}\,\psi)$, check for emptiness the synchronized product of $\mathcal{A}_\psi$ and M with initial state s.
- $\mathsf{A}\,\psi = \neg\,\mathsf{E}\,\neg\psi$

Model checking

$$M \models \varphi \text{ iff } I \subseteq S(\mathsf{A}\,\varphi).$$

CTL (Clarke & Emerson 81)

Syntax: CTL: Computation Tree Logic

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{AX}\,\varphi \mid \mathsf{E}\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\varphi$$

Remarks

The semantics is inherited from CTL^*.

All CTL-formulas are state formulas. Hence, we have a simpler semantics.

Semantics: only state formulas

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and let $s \in S$.

$s \models p$	if	$p \in \ell(s)$
$s \models \mathsf{EX}\,\varphi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots$ with $s_1 \models \varphi$
$s \models \mathsf{AX}\,\varphi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots$, we have $s_1 \models \varphi$
$s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$
$s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$

CTL (Clarke & Emerson 81)

Syntax: CTL: Computation Tree Logic

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{AX}\,\varphi \mid \mathsf{E}\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\varphi$$

Remarks

The semantics is inherited from CTL^*.

All CTL-formulas are state formulas. Hence, we have a simpler semantics.

Semantics: only state formulas

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and let $s \in S$.

$s \models p$	if	$p \in \ell(s)$
$s \models \mathsf{EX}\,\varphi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots$ with $s_1 \models \varphi$
$s \models \mathsf{AX}\,\varphi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots$, we have $s_1 \models \varphi$
$s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$
$s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$

CTL (Clarke & Emerson 81)

Syntax: CTL: Computation Tree Logic

$$\varphi ::= \bot \mid p\ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{AX}\,\varphi \mid \mathsf{E}\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\varphi$$

Remarks

The semantics is inherited from CTL^*.

All CTL-formulas are state formulas. Hence, we have a simpler semantics.

Semantics: only state formulas

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and let $s \in S$.

$s \models p$	if	$p \in \ell(s)$
$s \models \mathsf{EX}\,\varphi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots$ with $s_1 \models \varphi$
$s \models \mathsf{AX}\,\varphi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots$, we have $s_1 \models \varphi$
$s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$
$s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$

CTL (Clarke & Emerson 81)

Semantics: only state formulas

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure without deadlocks and let $s \in S$.

$s \models p$	if	$p \in \ell(s)$
$s \models \mathsf{EX}\,\varphi$	if	$\exists s \to s'$ with $s' \models \varphi$
$s \models \mathsf{AX}\,\varphi$	if	$\forall s \to s'$ we have $s' \models \varphi$
$s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi$	if	$\exists s = s_0 \to s_1 \to s_2 \to \cdots s_j,$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$
$s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \geq 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \leq k < j$

Macros

- $\mathsf{EF}\,\varphi = \mathsf{E}\top\mathsf{U}\,\varphi$ and $\mathsf{AF}\,\varphi = \mathsf{A}\top\mathsf{U}\,\varphi$ $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$.
- $\mathsf{EG}\,\varphi = \neg\,\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi = \neg\,\mathsf{EF}\,\neg\varphi$

CTL (Clarke & Emerson 81)

Semantics: only state formulas

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure without deadlocks and let $s \in S$.

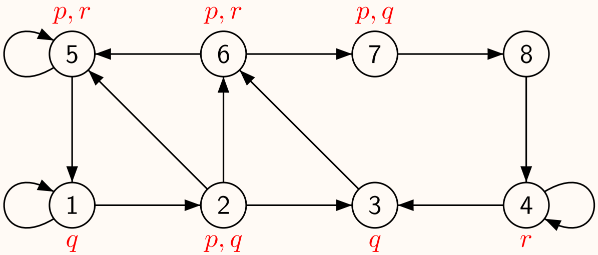
$$
\begin{array}{lll}
s \models p & \text{if} & p \in \ell(s) \\
s \models \mathsf{EX}\,\varphi & \text{if} & \exists s \to s' \text{ with } s' \models \varphi \\
s \models \mathsf{AX}\,\varphi & \text{if} & \forall s \to s' \text{ we have } s' \models \varphi \\
s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi & \text{if} & \exists s = s_0 \to s_1 \to s_2 \to \cdots s_j, \text{ with} \\
 & & \qquad s_j \models \psi \text{ and } s_k \models \varphi \text{ for all } 0 \le k < j \\
s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi & \text{if} & \forall s = s_0 \to s_1 \to s_2 \to \cdots, \exists j \ge 0 \text{ with} \\
 & & \qquad s_j \models \psi \text{ and } s_k \models \varphi \text{ for all } 0 \le k < j
\end{array}
$$

Macros

- $\mathsf{EF}\,\varphi = \mathsf{E}\top\mathsf{U}\,\varphi$ and $\mathsf{AF}\,\varphi = \mathsf{A}\top\mathsf{U}\,\varphi$ $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$.
- $\mathsf{EG}\,\varphi = \neg\,\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi = \neg\,\mathsf{EF}\,\neg\varphi$

CTL (Clarke & Emerson 81)

Example



Compute

$S(\mathsf{EX}\,p) =$

$S(\mathsf{AX}\,p) =$

$S(\mathsf{EF}\,p) =$

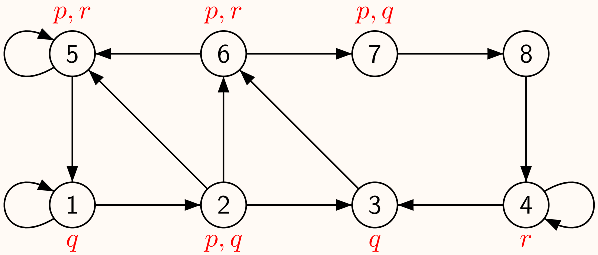
$S(\mathsf{AF}\,p) =$

$S(\mathsf{E}\,q\,\mathsf{U}\,r) =$

$S(\mathsf{A}\,q\,\mathsf{U}\,r) =$

CTL (Clarke & Emerson 81)

Example

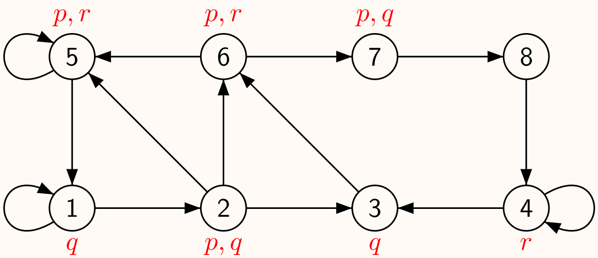


Compute

$$S(\mathsf{EX}\,p) = \{1, 2, 3, 5, 6\}$$
$$S(\mathsf{AX}\,p) =$$
$$S(\mathsf{EF}\,p) =$$
$$S(\mathsf{AF}\,p) =$$
$$S(\mathsf{E}\,q\,\mathsf{U}\,r) =$$
$$S(\mathsf{A}\,q\,\mathsf{U}\,r) =$$

CTL (Clarke & Emerson 81)

Example

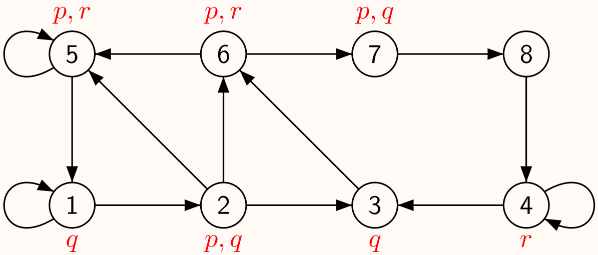


Compute

$$S(\mathsf{EX}\,p) = \{1,2,3,5,6\}$$
$$S(\mathsf{AX}\,p) = \{3,6\}$$
$$S(\mathsf{EF}\,p) =$$
$$S(\mathsf{AF}\,p) =$$
$$S(\mathsf{E}\,q\,\mathsf{U}\,r) =$$
$$S(\mathsf{A}\,q\,\mathsf{U}\,r) =$$

CTL (Clarke & Emerson 81)

Example

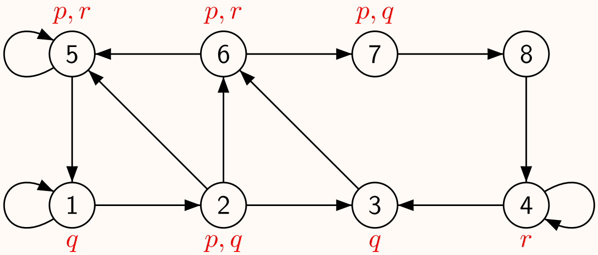


Compute

$$S(\mathsf{EX}\,p) = \{1,2,3,5,6\}$$
$$S(\mathsf{AX}\,p) = \{3,6\}$$
$$S(\mathsf{EF}\,p) = \{1,2,3,4,5,6,7,8\}$$
$$S(\mathsf{AF}\,p) =$$
$$S(\mathsf{E}\,q\,\mathsf{U}\,r) =$$
$$S(\mathsf{A}\,q\,\mathsf{U}\,r) =$$

CTL (Clarke & Emerson 81)

Example

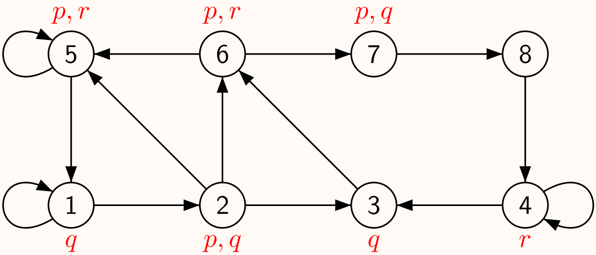


Compute

$$S(\mathsf{EX}\, p) = \{1,2,3,5,6\}$$
$$S(\mathsf{AX}\, p) = \{3,6\}$$
$$S(\mathsf{EF}\, p) = \{1,2,3,4,5,6,7,8\}$$
$$S(\mathsf{AF}\, p) = \{2,3,5,6,7\}$$
$$S(\mathsf{E}\, q\, \mathsf{U}\, r) =$$
$$S(\mathsf{A}\, q\, \mathsf{U}\, r) =$$

CTL (Clarke & Emerson 81)

Example

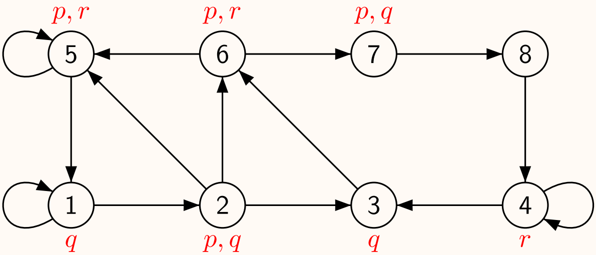


Compute

$$S(\mathsf{EX}\,p) = \{1,2,3,5,6\}$$
$$S(\mathsf{AX}\,p) = \{3,6\}$$
$$S(\mathsf{EF}\,p) = \{1,2,3,4,5,6,7,8\}$$
$$S(\mathsf{AF}\,p) = \{2,3,5,6,7\}$$
$$S(\mathsf{E}\,q\,\mathsf{U}\,r) = \{1,2,3,4,5,6\}$$
$$S(\mathsf{A}\,q\,\mathsf{U}\,r) =$$

CTL (Clarke & Emerson 81)

Example



Compute

$$S(\mathsf{EX}\, p) = \{1, 2, 3, 5, 6\}$$
$$S(\mathsf{AX}\, p) = \{3, 6\}$$
$$S(\mathsf{EF}\, p) = \{1, 2, 3, 4, 5, 6, 7, 8\}$$
$$S(\mathsf{AF}\, p) = \{2, 3, 5, 6, 7\}$$
$$S(\mathsf{E}\, q\, \mathsf{U}\, r) = \{1, 2, 3, 4, 5, 6\}$$
$$S(\mathsf{A}\, q\, \mathsf{U}\, r) = \{2, 3, 4, 5, 6\}$$

CTL (Clarke & Emerson 81)

Equivalent formulas

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $$\begin{aligned} \mathsf{A}\,\varphi\,\mathsf{U}\,\psi &= \neg\,\mathsf{E}\,\neg(\varphi\,\mathsf{U}\,\psi) \\ &= \neg\,\mathsf{E}(\mathsf{G}\,\neg\psi \wedge \neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) \\ &= \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi) \end{aligned}$$
- $\mathsf{A}\,\mathsf{G}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ — infinitely often
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi = \mathsf{EF}\,\mathsf{EG}\,\varphi$ — ultimately
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \neq \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi$

CTL (Clarke & Emerson 81)

Equivalent formulas

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $$\begin{aligned} \mathsf{A}\,\varphi\,\mathsf{U}\,\psi &= \neg\,\mathsf{E}\,\neg(\varphi\,\mathsf{U}\,\psi) \\ &= \neg\,\mathsf{E}(\mathsf{G}\,\neg\psi \wedge \neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) \\ &= \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi) \end{aligned}$$
- $\mathsf{A}\,\mathsf{G}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ — infinitely often
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi = \mathsf{EF}\,\mathsf{EG}\,\varphi$ — ultimately
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \neq \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi$

CTL (Clarke & Emerson 81)

Equivalent formulas

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $$\begin{aligned} \mathsf{A}\,\varphi\,\mathsf{U}\,\psi &= \neg\,\mathsf{E}\,\neg(\varphi\,\mathsf{U}\,\psi) \\ &= \neg\,\mathsf{E}(\mathsf{G}\,\neg\psi \wedge \neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) \\ &= \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi) \end{aligned}$$
- $\mathsf{A}\,\mathsf{G}(\mathrm{req} \rightarrow \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \rightarrow \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ infinitely often
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi = \mathsf{EF}\,\mathsf{EG}\,\varphi$ ultimately
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \neq \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi$

CTL (Clarke & Emerson 81)

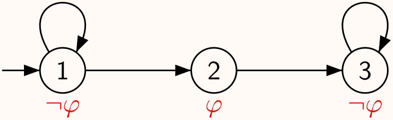
Equivalent formulas

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $$\begin{aligned} \mathsf{A}\,\varphi\,\mathsf{U}\,\psi &= \neg\,\mathsf{E}\,\neg(\varphi\,\mathsf{U}\,\psi) \\ &= \neg\,\mathsf{E}(\mathsf{G}\,\neg\psi \wedge \neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) \\ &= \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi) \end{aligned}$$
- $\mathsf{A}\,\mathsf{G}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ infinitely often
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi = \mathsf{EF}\,\mathsf{EG}\,\varphi$ ultimately
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \neq \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi$

CTL (Clarke & Emerson 81)

Equivalent formulas

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $$\begin{aligned} \mathsf{A}\,\varphi\,\mathsf{U}\,\psi &= \neg\,\mathsf{E}\,\neg(\varphi\,\mathsf{U}\,\psi) \\ &= \neg\,\mathsf{E}(\mathsf{G}\,\neg\psi \wedge \neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)) \\ &= \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi) \end{aligned}$$
- $\mathsf{A}\,\mathsf{G}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ infinitely often
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi = \mathsf{EF}\,\mathsf{EG}\,\varphi$ ultimately
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \neq \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \neq \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi$



Model checking of CTL

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}$

Question: Does $M \models \varphi$?

Remark

$M \models \varphi$ iff $I \subseteq S(\varphi)$

Theorem

The model checking problem for CTL is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Proof

Marking algorithm.

Model checking of CTL

Model checking problem

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}$

Question: Does $M \models \varphi$?

Remark

$M \models \varphi$ iff $I \subseteq S(\varphi)$

Theorem

The model checking problem for CTL is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Proof

Marking algorithm.

Model checking of CTL

procedure mark(φ)

case $\varphi = p \in \mathrm{AP}$
  for all $s \in S$ do $s.\varphi := (p \in \ell(s))$;

case $\varphi = \neg\varphi_1$
  mark(φ_1);
  for all $s \in S$ do $s.\varphi := \neg\, s.\varphi_1$;

case $\varphi = \varphi_1 \vee \varphi_2$
  mark(φ_1); mark(φ_2);
  for all $s \in S$ do $s.\varphi := s.\varphi_1 \vee s.\varphi_2$;

case $\varphi = EX\varphi_1$
  mark(φ_1);
  for all $s \in S$ do $s.\varphi := \mathrm{false}$;
  for all $(t, s) \in T$ do if $s.\varphi_1$ then $t.\varphi := \mathrm{true}$;

case $\varphi = AX\varphi_1$
  mark(φ_1);
  for all $s \in S$ do $s.\varphi := \mathrm{true}$;
  for all $(t, s) \in T$ do if $\neg\, s.\varphi_1$ then $t.\varphi := \mathrm{false}$;

Model checking of CTL

procedure mark(φ)

case $\varphi = E\varphi_1$ U φ_2

mark(φ_1); mark(φ_2);
$L := \emptyset$;
for all $s \in S$ do
  $s.\varphi := s.\varphi_2$;
  if $s.\varphi$ then $L := L \cup \{s\}$;
while $L \neq \emptyset$ do
  take $s \in L$;
  $L := L \setminus \{s\}$;
  for all $t \in S$ with $(t, s) \in T$ do
    if $t.\varphi_1 \wedge \neg\, t.\varphi$ then $t.\varphi := \mathrm{true}$; $L := L \cup \{t\}$;

Model checking of CTL

procedure mark(φ)

case $\varphi = A\varphi_1 \mathsf{U} \varphi_2$
  mark(φ_1); mark(φ_2);
  $L := \emptyset$;
  for all $s \in S$ do
    $s.\varphi := s.\varphi_2$; $s.nb :=$ degree(s);
    if $s.\varphi$ then $L := L \cup \{s\}$;
  while $L \neq \emptyset$ do
    take $s \in L$;
    $L := L \setminus \{s\}$;
    for all $t \in S$ with $(t, s) \in T$ do
      $t.nb := t.nb - 1$;
      if $t.nb = 0 \wedge t.\varphi_1 \wedge \neg t.\varphi$ then $t.\varphi := \mathrm{true}$; $L := L \cup \{t\}$;

fairness

Fairness

Only fair runs are of interest

- Each process is enabled infinitely often: $\bigwedge_i \mathsf{G}\,\mathsf{F}\,\mathrm{run}_i$
- No process stays ultimately in the critical section: $\bigwedge_i \neg\,\mathsf{F}\,\mathsf{G}\,\mathrm{CS}_i = \bigwedge_i \mathsf{G}\,\mathsf{F}\,\neg\mathrm{CS}_i$

Fair Kripke structure

$M = (S, T, I, \mathrm{AP}, \ell, \mathcal{F})$ where $\mathcal{F} = \{F_1, \ldots, F_n\}$ with $F_i \subseteq S$.

An infinite run σ is fair if it visits infinitely often each F_i

Fair quantifications

$$\mathsf{E}_f\,\varphi = \mathsf{E}(\mathrm{fair} \wedge \varphi) \qquad \text{and} \qquad \mathsf{A}_f\,\varphi = \mathsf{A}(\mathrm{fair} \rightarrow \varphi)$$

where

$$\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$$

fairness

Fairness

Only fair runs are of interest

- Each process is enabled infinitely often: $\bigwedge_i \mathsf{G}\,\mathsf{F}\,\mathrm{run}_i$
- No process stays ultimately in the critical section: $\bigwedge_i \neg\,\mathsf{F}\,\mathsf{G}\,\mathrm{CS}_i = \bigwedge_i \mathsf{G}\,\mathsf{F}\,\neg\mathrm{CS}_i$

Fair Kripke structure

$M = (S, T, I, \mathrm{AP}, \ell, \mathcal{F})$ where $\mathcal{F} = \{F_1, \ldots, F_n\}$ with $F_i \subseteq S$.

An infinite run σ is fair if it visits infinitely often each F_i

Fair quantifications

$$\mathsf{E}_f\,\varphi = \mathsf{E}(\mathrm{fair} \wedge \varphi) \qquad \text{and} \qquad \mathsf{A}_f\,\varphi = \mathsf{A}(\mathrm{fair} \rightarrow \varphi)$$

where

$$\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$$

fairness

Fairness

Only fair runs are of interest

- Each process is enabled infinitely often: $\bigwedge_i \mathsf{G}\,\mathsf{F}\,\mathrm{run}_i$
- No process stays ultimately in the critical section: $\bigwedge_i \neg\,\mathsf{F}\,\mathsf{G}\,\mathrm{CS}_i = \bigwedge_i \mathsf{G}\,\mathsf{F}\,\neg\mathrm{CS}_i$

Fair Kripke structure

$M = (S, T, I, \mathrm{AP}, \ell, \mathcal{F})$ where $\mathcal{F} = \{F_1, \ldots, F_n\}$ with $F_i \subseteq S$.

An infinite run σ is fair if it visits infinitely often each F_i

Fair quantifications

$$\mathsf{E}_f\,\varphi = \mathsf{E}(\mathrm{fair} \wedge \varphi) \qquad \text{and} \qquad \mathsf{A}_f\,\varphi = \mathsf{A}(\mathrm{fair} \rightarrow \varphi)$$

where

$$\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$$

fair CTL

Syntax of fair-CTL

$$\varphi ::= \bot \mid p \; (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}_f\,\mathsf{X}\,\varphi \mid \mathsf{A}_f\,\mathsf{X}\,\varphi \mid \mathsf{E}_f\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}_f\,\varphi\,\mathsf{U}\,\varphi$$

Lemma: CTL_f cannot be expressed in CTL

Consider the Kripke structure M_k defined by:

- $M_k, 2k \models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,p$ but $M_k, 2k-2 \not\models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,p$
- If $\varphi \in \mathrm{CTL}$ and $|\varphi| \leq m \leq k$ then $M_k, 2k \models \varphi$ iff $M_k, 2m \models \varphi$

If the fairness condition is $\ell^{-1}(p)$ then $\mathsf{E}_f\,\mathsf{F}\,\top$ cannot be expressed in CTL.

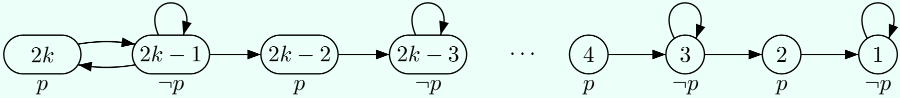
fair CTL

Syntax of fair-CTL

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}_f \, \mathsf{X} \, \varphi \mid \mathsf{A}_f \, \mathsf{X} \, \varphi \mid \mathsf{E}_f \, \varphi \, \mathsf{U} \, \varphi \mid \mathsf{A}_f \, \varphi \, \mathsf{U} \, \varphi$$

Lemma: CTL_f cannot be expressed in CTL

Consider the Kripke structure M_k defined by:



- $M_k, 2k \models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,p$ but $M_k, 2k-2 \not\models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,p$
- If $\varphi \in \mathrm{CTL}$ and $|\varphi| \leq m \leq k$ then $M_k, 2k \models \varphi$ iff $M_k, 2m \models \varphi$

If the fairness condition is $\ell^{-1}(p)$ then $\mathsf{E}_f \, \mathsf{F} \top$ cannot be expressed in CTL.

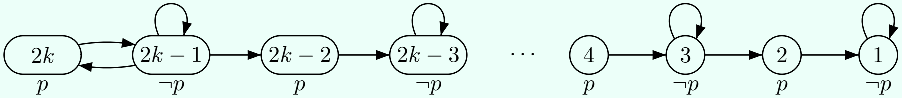
fair CTL

Syntax of fair-CTL

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}_f \mathsf{X}\, \varphi \mid \mathsf{A}_f \mathsf{X}\, \varphi \mid \mathsf{E}_f\, \varphi \,\mathsf{U}\, \varphi \mid \mathsf{A}_f\, \varphi \,\mathsf{U}\, \varphi$$

Lemma: CTL_f cannot be expressed in CTL

Consider the Kripke structure M_k defined by:



- $M_k, 2k \models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\, p$ but $M_k, 2k-2 \not\models \mathsf{E}\,\mathsf{G}\,\mathsf{F}\, p$
- If $\varphi \in \mathrm{CTL}$ and $|\varphi| \leq m \leq k$ then $M_k, 2k \models \varphi$ iff $M_k, 2m \models \varphi$

If the fairness condition is $\ell^{-1}(p)$ then $\mathsf{E}_f\, \mathsf{F} \top$ cannot be expressed in CTL.

Model checking of CTL_f

First step: Computation of $\mathrm{Fair} = \{s \in S \mid M, s \models \mathsf{E}_f\, \mathsf{F} \top\}$

Compute the SCC of M with Tarjan's algorithm (in linear time).

Let S' be the union of the SCCs which intersect each F_i.

Then, Fair is the set of states that can reach S'.

Note that reachability can be computed in linear time.

Reductions

$\mathsf{E}_f\, \mathsf{X}\, \varphi = \mathsf{E}\, \mathsf{X}(\mathrm{Fair} \wedge \varphi)$ and $\mathsf{E}_f\, \varphi\, \mathsf{U}\, \psi = \mathsf{E}\, \varphi\, \mathsf{U}\, (\mathrm{Fair} \wedge \psi)$

It remains to deal with $\mathsf{A}_f\, \varphi\, \mathsf{U}\, \psi$.

Recall that $\mathsf{A}\, \varphi\, \mathsf{U}\, \psi = \neg\, \mathsf{E}\mathsf{G}\, \neg\psi \vee \neg\, \mathsf{E}\, \neg\psi\, \mathsf{U}\, (\neg\varphi \wedge \neg\psi)$

This formula also holds for the fair quantifications.
Hence, we only need to compute the semantics of $\mathsf{E}_f\, \mathsf{G}\, \varphi$.

Model checking of CTL_f

First step: Computation of $\mathrm{Fair} = \{s \in S \mid M, s \models \mathsf{E}_f\,\mathsf{F}\,\top\}$

Compute the SCC of M with Tarjan's algorithm (in linear time).

Let S' be the union of the SCCs which intersect each F_i.

Then, Fair is the set of states that can reach S'.

Note that reachability can be computed in linear time.

Reductions

$\mathsf{E}_f\,\mathsf{X}\,\varphi = \mathsf{E}\,\mathsf{X}(\mathrm{Fair} \wedge \varphi)$ and $\mathsf{E}_f\,\varphi\,\mathsf{U}\,\psi = \mathsf{E}\,\varphi\,\mathsf{U}\,(\mathrm{Fair} \wedge \psi)$

It remains to deal with $\mathsf{A}_f\,\varphi\,\mathsf{U}\,\psi$.

Recall that $\mathsf{A}\,\varphi\,\mathsf{U}\,\psi = \neg\,\mathsf{EG}\,\neg\psi \vee \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)$

This formula also holds for the fair quantifications.
Hence, we only need to compute the semantics of $\mathsf{E}_f\,\mathsf{G}\,\varphi$.

Model checking of CTL_f

First step: Computation of $\mathrm{Fair} = \{s \in S \mid M, s \models \mathsf{E}_f \mathsf{F} \top\}$

Compute the SCC of M with Tarjan's algorithm (in linear time).

Let S' be the union of the SCCs which intersect each F_i.

Then, Fair is the set of states that can reach S'.

Note that reachability can be computed in linear time.

Reductions

$\mathsf{E}_f \mathsf{X} \varphi = \mathsf{E} \mathsf{X}(\mathrm{Fair} \wedge \varphi)$ and $\mathsf{E}_f \varphi \mathsf{U} \psi = \mathsf{E} \varphi \mathsf{U} (\mathrm{Fair} \wedge \psi)$

It remains to deal with $\mathsf{A}_f \varphi \mathsf{U} \psi$.

Recall that $\mathsf{A} \varphi \mathsf{U} \psi = \neg \mathsf{EG} \neg\psi \vee \neg \mathsf{E} \neg\psi \mathsf{U} (\neg\varphi \wedge \neg\psi)$

This formula also holds for the fair quantifications.
Hence, we only need to compute the semantics of $\mathsf{E}_f \mathsf{G} \varphi$.

Model checking of CTL_f

Computation of $\mathsf{E}_f\,\mathsf{G}\,\varphi$

Let M_φ be the restriction of M to $S_f(\varphi)$.

Compute the SCC of M_φ with Tarjan's algorithm (in linear time).

Let S' be the union of the SCCs of M_φ which intersect each F_i.

Then, $M, s \models \mathsf{E}_f\,\mathsf{G}\,\varphi$ iff $M, s \models \mathsf{E}\,\varphi\,\mathsf{U}\,S'$ iff $M_\varphi \models \mathsf{EF}\,S'$.

This is again a reachability problem which can be done in linear time.

Theorem

The model checking problem for CTL_f is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Model checking of CTL_f

Computation of $\mathsf{E}_f\,\mathsf{G}\,\varphi$

Let M_φ be the restriction of M to $S_f(\varphi)$.

Compute the SCC of M_φ with Tarjan's algorithm (in linear time).

Let S' be the union of the SCCs of M_φ which intersect each F_i.

Then, $M, s \models \mathsf{E}_f\,\mathsf{G}\,\varphi$ iff $M, s \models \mathsf{E}\,\varphi\,\mathsf{U}\,S'$ iff $M_\varphi \models \mathsf{EF}\,S'$.

This is again a reachability problem which can be done in linear time.

Theorem

The model checking problem for CTL_f is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Missing in this talk

- Symbolic model checking for CTL using BDDs.
- μ- calculus
- ...