Initiation à la vérification
Basics of Verification

http://mpri.master.univ-paris7.fr/C-1-22.html

Paul Gastin

Paul.Gastin@lsv.ens-cachan.fr
http://www.lsv.ens-cachan.fr/~gastin/

M1 du MPRI
2009-2010

Outline

Need for formal verifications methods

Critical systems

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

Disastrous software bugs

Mariner 1 probe, 1962

See http://en.wikipedia.org/wiki/Mariner_1

- Destroyed 293 seconds after launch
- Missing hyphen in the data or program? No!
- Period instead of comma in FORTRAN? No!
- Overbar missing in the mathematical specification:
  $\overline{\dot{R}_n}$: nth smoothed value of the time derivative of a radius.
  Without the smoothing function indicated by the bar, the program treated normal minor variations of velocity as if they were serious, causing spurious corrections that sent the rocket off course.

Disastrous software bugs

Ariane 5 flight 501, 1996

See http://en.wikipedia.org/wiki/Ariane_5_Flight_501

- Destroyed 37 seconds after launch (cost: 370 millions dollars).
- data conversion from a 64-bit floating point to 16-bit signed integer value caused a hardware exception (arithmetic overflow).
- Efficiency considerations had led to the disabling of the software handler (in Ada code) for this error trap.
- The fault occured in the inertial reference system of Ariane 5. The software from Ariane 4 was re-used for Ariane 5 without re-testing.
- On the basis of those calculations the main computer commanded the booster nozzles, and somewhat later the main engine nozzle also, to make a large correction for an attitude deviation that had not occurred.
- The error occurred in a realignment function which was not useful for Ariane 5.

Disastrous software bugs

Other well-known bugs

- Therac-25, at least 3 death by massive overdoses of radiation.
  Race condition in accessing shared resources.
  See http://en.wikipedia.org/wiki/Therac-25
- Electricity blackout, USA and Canada, 2003, 55 millions people.
  Race condition in accessing shared resources.
  See http://en.wikipedia.org/wiki/Northeast_Blackout_of_2003
- Pentium FDIV bug, 1994.
  Flaw in the division algorithm, discovered by Thomas Nicely.
  See http://en.wikipedia.org/wiki/Pentium_FDIV_bug
- Needham-Schroeder, authentication protocol based on symmetric encryption.
  Published in 1978 by Needham and Schroeder
  Proved correct by Burrows, Abadi and Needham in 1989
  Flaw found by Lowe in 1995 (man in the middle)
  Automatically proved incorrect in 1996.
  See http://en.wikipedia.org/wiki/Needham-Schroeder_protocol

Formal verifications methods

Complementary approaches

- Theorem prover
- Model checking
- Static analysis
- Test

Model Checking

- Purpose 1: automatically finding software or hardware bugs.
- Purpose 2: prove correctness of abstract models.
- Should be applied during design.
- Real systems can be analysed with abstractions.

E.M. Clarke

E.A. Emerson

J. Sifakis

Prix Turing 2007.

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

Bibliography

[1] Christel Baier and Joost-Pieter Katoen.
Principles of Model Checking.
MIT Press, 2008.

[2] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and Ph. Schnoebelen.
Systems and Software Verification. Model-Checking Techniques and Tools.
Springer, 2001.

[3] E.M. Clarke, O. Grumberg, D.A. Peled.
Model Checking.
MIT Press, 1999.

[4] Z. Manna and A. Pnueli.
The Temporal Logic of Reactive and Concurrent Systems: Specification.
Springer, 1991.

[5] Z. Manna and A. Pnueli.
Temporal Verification of Reactive Systems: Safety.
Springer, 1995.

Outline

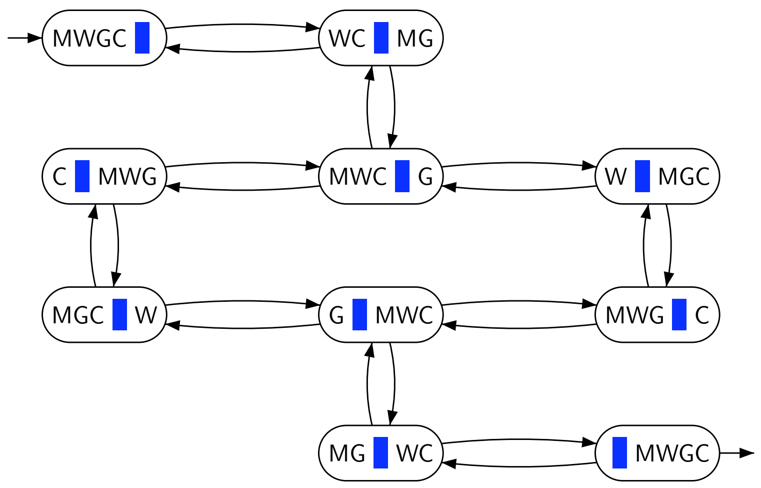
Constructing the model

Example: Men, Wolf, Goat, Cabbage

Model = Transition system

- State = who is on which side of the river
- Transition = crossing the river
- Specification
  Safety: Never leave WG or GC alone
  Liveness: Take everyone to the other side of the river.

Transition system

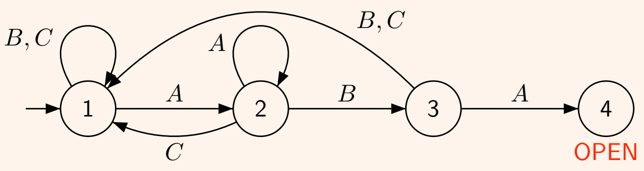


Transition system or Kripke structure

Definition: TS $M = (S, \Sigma, T, I, \mathrm{AP}, \ell)$

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

Example: Digicode



Every discrete system may be described with a TS.

Description Languages

Pb: How can we easily describe big systems?

Description Languages (high level)

- Programming languages
- Boolean circuits
- Modular description, e.g., parallel compositions
  problems: concurrency, synchronization, communication, atomicity, fairness, ...
- Petri nets (intermediate level)
- Transition systems (intermediate level)
  with variables, stacks, channels, ...
  synchronized products
- Logical formulae (low level)

Operational semantics

High level descriptions are translated (compiled) to low level (infinite) TS.

Transition systems with variables

Definition: TSV $M = (S, \Sigma, \mathcal{V}, (D_v)_{v \in \mathcal{V}}, T, I, \mathrm{AP}, \ell)$

- $\mathcal{V}$: set of (typed) variables, e.g., boolean, [0..4], ...
- Each variable $v \in \mathcal{V}$ has a domain D_v (finite or infinite)
- Guard or Condition: unary predicate over $D = \prod_{v \in \mathcal{V}} D_v$
  Symbolic descriptions: $x < 5$, $x + y = 10$, ...
- Instruction or Update: map $f : D \to D$
  Symbolic descriptions: $x := 0$, $x := (y+1)^2$, ...
- $T \subseteq S \times (2^D \times \Sigma \times D^D) \times S$
  Symbolic descriptions: $s \xrightarrow{x<50,?\mathrm{coin},x:=x+\mathrm{coin}} s'$
- $I \subseteq S \times 2^D$
  Symbolic descriptions: $(s_0, x := 0)$

Example: Vending machine

- coffee: 50 cents, orange juice: 1 euro, ...
- possible coins: 10, 20, 50 cents
- we may shuffle coin insertions and drink selection

Transition systems with variables

Semantics: low level TS

- $S' = S \times D$
- $I' = \{(s, \nu) \mid \exists (s, g) \in I \text{ with } \nu \models g\}$
- Transitions: $T' \subseteq (S \times D) \times \Sigma \times (S \times D)$

$$\frac{s \xrightarrow{g,a,f} s' \wedge \nu \models g}{(s, \nu) \xrightarrow{a} (s', f(\nu))}$$

SOS: Structural Operational Semantics

- AP': we may use atomic propositions in AP or guards in 2^D such as $x > 0$.

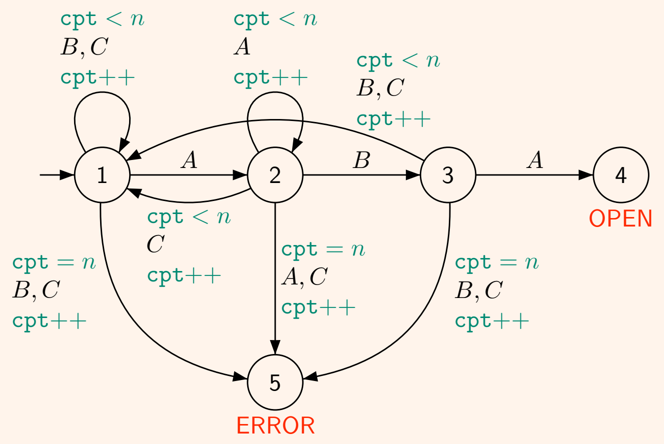
Programs = Kripke structures with variables

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

Example: GCD

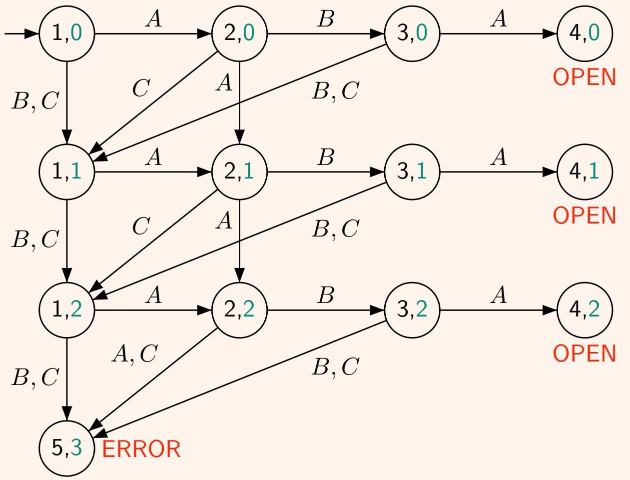
TS with variables ...

Example: Digicode



... and its semantics ($n = 2$)

Example: Digicode



Only variables

The state is nothing but a special variable: $s \in \mathcal{V}$ with domain $D_s = S$.

Definition: TSV $\quad M = (\mathcal{V}, (D_v)_{v \in \mathcal{V}}, T, I, \mathrm{AP}, \ell)$

- $D = \prod_{v \in \mathcal{V}} D_v$,
- $I \subseteq D$, $T \subseteq D \times D$

Symbolic representations with boolean functions

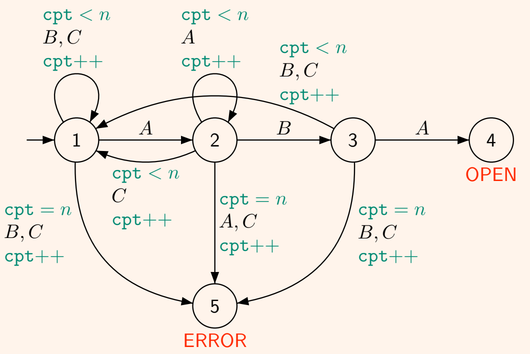
- I given by a formula $\psi(\nu)$
- T given by a formula $\varphi(\nu, \nu')$
  ν: values before the transition
  ν': values after the transition
- Often we use boolean variables only: $D_v = \{0, 1\}$
- Concise descriptions of boolean functions with Binary Decision Diagrams.

Example: Boolean circuit: modulo 8 counter

$$\begin{aligned} b'_0 &= \neg b_0 \\ b'_1 &= b_0 \oplus b_1 \\ b'_2 &= (b_0 \wedge b_1) \oplus b_2 \end{aligned}$$

Symbolic representation

Example: Logical representation



$$\begin{aligned}
\delta_B = \quad & s = 1 \land \mathsf{cpt} < n \land s' = 1 \land \mathsf{cpt}' = \mathsf{cpt} + 1 \\
\lor \quad & s = 1 \land \mathsf{cpt} = n \land s' = 5 \land \mathsf{cpt}' = \mathsf{cpt} + 1 \\
\lor \quad & s = 2 \land s' = 3 \land \mathsf{cpt}' = \mathsf{cpt} \\
\lor \quad & s = 3 \land \mathsf{cpt} < n \land s' = 1 \land \mathsf{cpt}' = \mathsf{cpt} + 1 \\
\lor \quad & s = 3 \land \mathsf{cpt} = n \land s' = 5 \land \mathsf{cpt}' = \mathsf{cpt} + 1
\end{aligned}$$

Modular description of concurrent systems

$$M = M_1 \parallel M_2 \parallel \cdots \parallel M_n$$

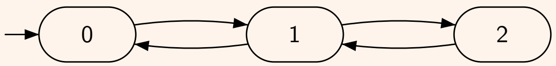
Semantics

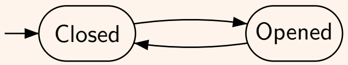
- Various semantics for the parallel composition $\parallel$
- Various communication mechanisms between components:
  Shared variables, FIFO channels, Rendez-vous, …
- Various synchronization mechanisms

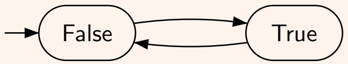
Example: Elevator with 1 cabin, 3 doors, 3 calling devices

Modular description of concurrent systems

Example: 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

Definition: General product

- Components: $M_i = (S_i, \Sigma_i, T_i, I_i, \mathrm{AP}_i, \ell_i)$
- Product: $M = (S, \Sigma, T, I, \mathrm{AP}, \ell)$ with
  $S = \prod_i S_i$, $\quad \Sigma = \prod_i (\Sigma_i \cup \{\varepsilon\})$, 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: restrictions of the general product.

Parallel compositions

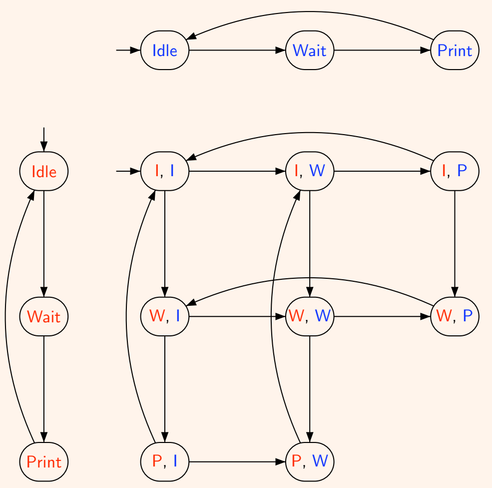
- Synchronous: $\Sigma_{\mathrm{sync}} = \prod_i \Sigma_i$
- Asynchronous: $\Sigma_{\mathrm{sync}} = \biguplus_i \Sigma'_i \qquad$ with $\Sigma'_i = \{\varepsilon\}^{i-1} \times \Sigma_i \times \{\varepsilon\}^{n-i}$

Synchronizations

- By states: $S_{\mathrm{sync}} \subseteq S$
- By labels: $\Sigma_{\mathrm{sync}} \subseteq \Sigma$
- By transitions: $T_{\mathrm{sync}} \subseteq T$

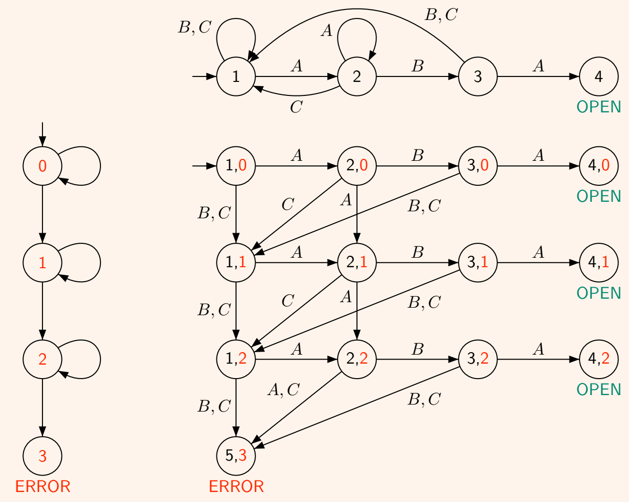
Example: Printer manager

Example: Asynchronous product
Synchronization by states: (P, P) is forbidden



Example: digicode

Example: Synchronous product
Synchronization by transitions



Synchronization by Rendez-vous

Synchronization by transitions is universal but too low-level.

Definition: Rendez-vous

- $!m$ sending message m
- $?m$ receiving message m
- SOS: Structural Operational Semantics

Local actions $\dfrac{s_1 \xrightarrow{a_1}_1 s_1'}{(s_1, s_2) \xrightarrow{a_1} (s_1', s_2)}$ $\dfrac{s_2 \xrightarrow{a_2}_1 s_2'}{(s_1, s_2) \xrightarrow{a_2} (s_1, s_2')}$

Rendez-vous $\dfrac{s_1 \xrightarrow{!m}_1 s_1' \wedge s_2 \xrightarrow{?m}_2 s_2'}{(s_1, s_2) \xrightarrow{m} (s_1', s_2')}$ $\dfrac{s_1 \xrightarrow{?m}_1 s_1' \wedge s_2 \xrightarrow{!m}_2 s_2'}{(s_1, s_2) \xrightarrow{m} (s_1', s_2')}$

- It is a kind of synchronization by actions.
- Essential feature of process algebra.

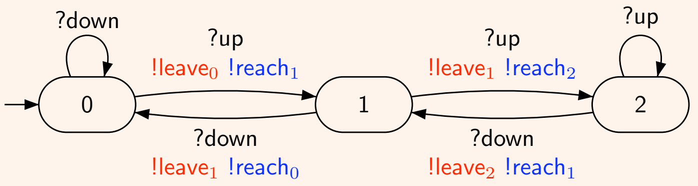
Example: Elevator with 1 cabin, 3 doors, 3 calling devices

- $?\mathrm{up}$ is uncontrollable for the cabin
- $?\mathrm{call}_0$ is uncontrollable for the system

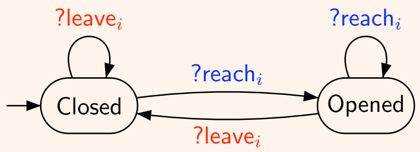
Example: Elevator

Example: Synchronization by Rendez-vous

Cabin:



Door for level i:



We should design the controller

Shared variables

Definition: Asynchronous product + shared variables

$\bar{s} = (s_1, \dots, s_n)$ denotes a tuple of states
$\nu \in D = \prod_{v \in \mathcal{V}} D_v$ is a valuation of variables.

Semantics (SOS) $$\frac{\nu \models g \wedge s_i \xrightarrow{g,a,f} s'_i \wedge s'_j = s_j \text{ for } j \neq i}{(\bar{s}, \nu) \xrightarrow{a} (\bar{s}', f(\nu))}$$

Example: Mutual exclusion for 2 processes satisfying

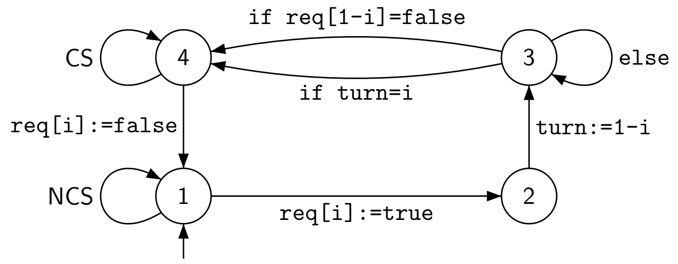
- Safety: never simultaneously in critical section (CS).
- Liveness: if a process wants to enter its CS, it eventually does.
- Fairness: if process 1 wants to enter its CS, then process 2 will enter its CS at most once before process 1 does.

using shared variables but no synchronization mechanisms: the atomicity is

- testing or reading or writing a single variable at a time
- no test and set: $x = 0; x := 1$

Peterson's algorithm (1981)

```
Process i:
   loop forever
      req[i] := true; turn := 1-i
      wait until (turn = i or req[1-i] = false)
      Critical section
      req[i] := false
```



Exercise:

- Is the algorithm still correct if we swape the first two assignments?
- Draw the concrete TS assuming the first two assignments are atomic.

Atomicity

Example:

Intially $x = 1 \wedge y = 2$
Program P_1: $x := x + y \parallel y := x + y$

Program P_2: $\begin{pmatrix} \texttt{Load}R_1, x \\ \texttt{Add}R_1, y \\ \texttt{Store}R_1, x \end{pmatrix} \parallel \begin{pmatrix} \texttt{Load}R_2, x \\ \texttt{Add}R_2, y \\ \texttt{Store}R_2, y \end{pmatrix}$

Assuming each instruction is atomic, what are the possible results of P_1 and P_2?

Atomicity

Definition: Atomic statements

Elementary statements (no loops, no communications, no synchronizations)

$$\begin{aligned} ES ::= & \text{ skip} \mid \text{await } c \mid x := e \mid ES \,;\, ES \mid ES \,\Box\, ES \\ & \mid \text{when } c \text{ do } ES \mid \text{if } c \text{ then } ES \text{ else } ES \end{aligned}$$

Atomic statements: if the ES can be fully executed then it is executed in one step.

$$\frac{(\bar{s}, \nu) \xrightarrow[*]{ES} (\bar{s}', \nu')}{(\bar{s}, \nu) \xrightarrow{\text{atomic}(ES)} (\bar{s}', \nu')}$$

Example: Atomic statements

- atomic$(x = 0; x := 1)$ (Test and set)
- atomic$(y := y - 1; \text{await}(y = 0); y := 1)$ is equivalent to await$(y = 1)$

Channels

Example: Leader election

We have n processes on a directed ring, each having a unique $\mathrm{id} \in \{1, \ldots, n\}$.

```
send(id)
loop forever
   receive(x)
   if (x = id) then STOP fi
   if (x > id) then send(x)
```


Channels

Definition: Channels

- Declaration:
  - c : channel [k] of bool   size k
  - c : channel [∞] of int   unbounded
  - c : channel [0] of colors   Rendez-vous
- Primitives:
  - empty(c)
  - $c!e$ add the value of expression e to channel c
  - $c?x$ read a value from c and assign it to x
- Domain: Let D_m be the domain for a single message.
  - $D_c = D_m^k$ size k
  - $D_c = D_m^*$ unbounded
  - $D_c = \{\varepsilon\}$ Rendez-vous
- Politics: FIFO, LIFO, BAG, ...

Channels

Semantics: (lossy) FIFO

Send $$\frac{s_i \xrightarrow{c!e} s_i' \wedge \nu'(c) = \nu(e) \cdot \nu(c)}{(\bar{s}, \nu) \xrightarrow{c!e} (\bar{s}', \nu')}$$

Receive $$\frac{s_i \xrightarrow{c?x} s_i' \wedge \nu(c) = \nu'(c) \cdot \nu'(x)}{(\bar{s}, \nu) \xrightarrow{c?e} (\bar{s}', \nu')}$$

Lossy send $$\frac{s_i \xrightarrow{c!e} s_i'}{(\bar{s}, \nu) \xrightarrow{c!e} (\bar{s}', \nu)}$$

Implicit assumption: all variables that do not occur in the premise are not modified.

Exercises:

1. Implement a FIFO channel using rendez-vous with an intermediary process.
2. Give the semantics of a LIFO channel.
3. Model the alternating bit protocol (ABP) using a lossy FIFO channel.
   Fairness assumption: For each channel, if infinitely many messages are sent, then infinitely many messages are delivered.

High-level descriptions

Summary

- 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

Definition: (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

Definition: 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

Definition: Static properties

Example: Mutual exclusion

Safety properties are often static.

They can be reduced to reachability.

Definition: 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 $\mathrm{FO}(<)$-formulae are not easy to write and to understand.
- $\mathrm{FO}(<)$ is decidable.
  ... but satisfiability and model checking are non elementary.

Definition: 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.

Definition: 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

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

Models are words in Σ^ω with $\Sigma = 2^{\mathrm{AP}}$.

Definition: 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

References

Bibliography

[6] S. Demri and P. Gastin.
Specification and Verification using Temporal Logics.
In Modern applications of automata theory, IISc Research Monographs 2. World Scientific, To appear.
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

A large list of references is given in this paper.

Outline

Linear Temporal Logic (Pnueli 1977)

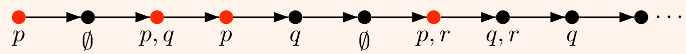
Definition: Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

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

Definition: Semantics: $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ with $\Sigma = 2^{\mathrm{AP}}$ and $i \in \mathbb{N}$

$w, i \models p$	if	$p \in a_i$
$w, i \models \neg\varphi$	if	$w, i \not\models \varphi$
$w, i \models \varphi \vee \psi$	if	$w, i \models \varphi$ or $w, i \models \psi$
$w, i \models \mathsf{X}\,\varphi$	if	$w, i+1 \models \varphi$
$w, i \models \varphi\,\mathsf{U}\,\psi$	if	$\exists k.\ i \leq k$ and $w, k \models \psi$ and $\forall j.\ (i \leq j < k) \to w, j \models \varphi$

Example:



Linear Temporal Logic (Pnueli 1977)

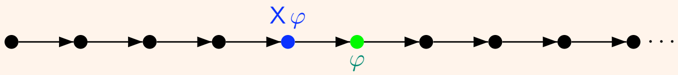
Definition: Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

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

Definition: Semantics: $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ with $\Sigma = 2^{\mathrm{AP}}$ and $i \in \mathbb{N}$

$w, i \models p$	if	$p \in a_i$
$w, i \models \neg\varphi$	if	$w, i \not\models \varphi$
$w, i \models \varphi \vee \psi$	if	$w, i \models \varphi$ or $w, i \models \psi$
$w, i \models \mathsf{X}\,\varphi$	if	$w, i+1 \models \varphi$
$w, i \models \varphi\,\mathsf{U}\,\psi$	if	$\exists k.\ i \leq k$ and $w, k \models \psi$ and $\forall j.\ (i \leq j < k) \rightarrow w, j \models \varphi$

Example:



Linear Temporal Logic (Pnueli 1977)

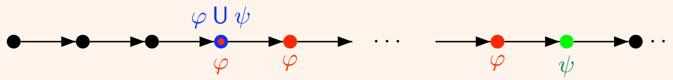
Definition: Syntax: $\mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

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

Definition: Semantics: $w = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ with $\Sigma = 2^{\mathrm{AP}}$ and $i \in \mathbb{N}$

$w, i \models p$	if	$p \in a_i$
$w, i \models \neg\varphi$	if	$w, i \not\models \varphi$
$w, i \models \varphi \vee \psi$	if	$w, i \models \varphi$ or $w, i \models \psi$
$w, i \models \mathsf{X}\,\varphi$	if	$w, i+1 \models \varphi$
$w, i \models \varphi\,\mathsf{U}\,\psi$	if	$\exists k.\ i \leq k$ and $w, k \models \psi$ and $\forall j.\ (i \leq j < k) \rightarrow w, j \models \varphi$

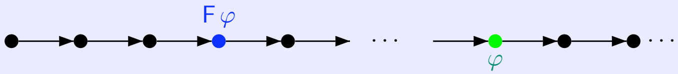
Example:



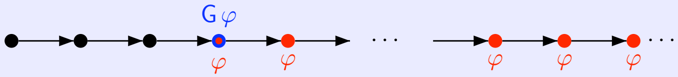
Linear Temporal Logic (Pnueli 1977)

Definition: Macros

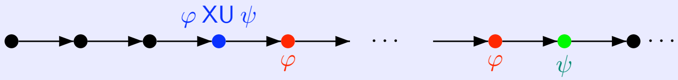
- Eventually: $\mathsf{F}\,\varphi = \top\,\mathsf{U}\,\varphi$



- 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)

Definition: 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)

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.

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

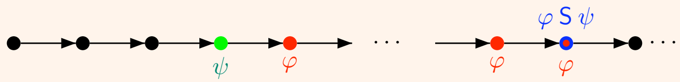
Past LTL

Definition: Semantics: $w = a_0a_1a_2\cdots \in \Sigma^\omega$ with $\Sigma = 2^{\mathrm{AP}}$ and $i \in \mathbb{N}$

$w, i \models \mathsf{Y}\,\varphi$ if $i > 0$ and $w, i-1 \models \varphi$

$w, i \models \varphi\,\mathsf{S}\,\psi$ if $\exists k.\ k \le i$ and $w, k \models \psi$ and $\forall j.\ (k < j \le i) \to w, y \models \varphi$

Example:



Example: LTL versus PLTL

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

Theorem (Laroussinie & Markey & Schnoebelen 2002)

PLTL may be exponentially more succinct than LTL.

Past LTL

Definition: Semantics: $w = a_0a_1a_2\cdots \in \Sigma^\omega$ with $\Sigma = 2^{\mathrm{AP}}$ and $i \in \mathbb{N}$

$w, i \models \mathsf{Y}\,\varphi$ if $i > 0$ and $w, i-1 \models \varphi$

$w, i \models \varphi\,\mathsf{S}\,\psi$ if $\exists k.\ k \le i$ and $w, k \models \psi$ and $\forall j.\ (k < j \le i) \to w, y \models \varphi$

Example:

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.

Expressivity

Theorem (Kamp 68)

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

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 $k \ge 0$,

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

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 $\mathrm{LTL}(\mathsf{X}, \mathsf{U}) = \mathrm{FO}_\Sigma(\le)$ due to Wilke 98.

Model checking for LTL

Definition: Model checking problem

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

Question: Does $M \models \varphi$?

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

$$M \models_\forall \varphi \quad \text{iff} \quad M \not\models_\exists \neg\varphi$$

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

The Model checking problem for LTL is PSPACE-complete

Satisfiability for LTL

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

Definition: Satisfiability problem

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

Question: Existence of $w \in \Sigma^\omega$ and $i \in \mathbb{N}$ such that $w, i \models \varphi$.

Definition: Initial Satisfiability problem

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

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

Definition: (Initial) validity

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

Decision procedure for LTL

Definition: The core

From a formula $\varphi \in \mathrm{LTL}(\mathrm{AP}, \ldots)$, 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 a synchronized product $\mathcal{B} = M \otimes \mathcal{A}_{\neg\varphi}$ so that
the successful runs of $\mathcal{B}$ correspond to the initial runs of M satisfying $\neg\varphi$.

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

Theorem:

Checking Büchi automata for emptiness is NLOGSPACE-complete.

Büchi automata

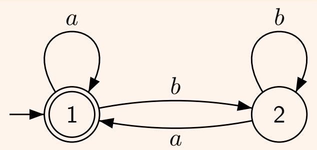
Definition:

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

$\mathcal{A} =$



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

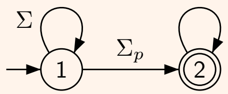
Büchi automata for some LTL formulae

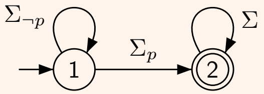
Definition:

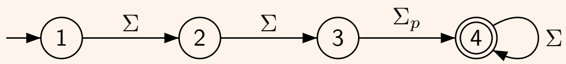
Recall that $\Sigma = 2^{\mathrm{AP}}$. For $\psi \in \mathbb{B}(\mathrm{AP})$ we let $\Sigma_\psi = \{a \in \Sigma \mid a \models \psi\}$.
For instance, for $p, q \in \mathrm{AP}$,

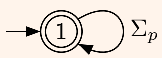
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ ...

Examples:

F p: 

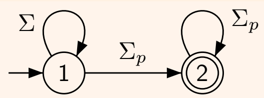
or 

X X p: 

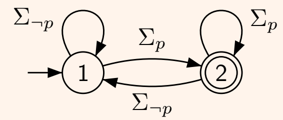
G p: 

Büchi automata for some LTL formulae

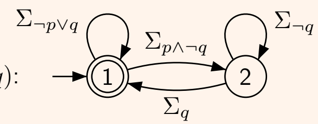
Examples:

F G p: 

no deterministic Büchi automaton.

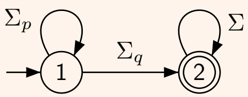
G F p: 

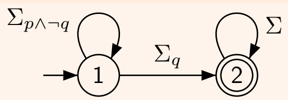
deterministic Büchi automaton are not closed under complement.

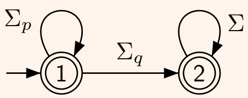
G$(p \to$ F $q)$: 

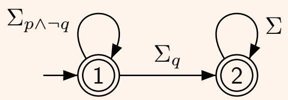
Büchi automata for some LTL formulae

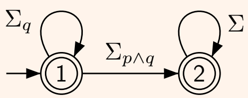
Examples:

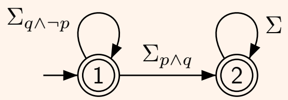
p U q: 

or 

p W q: 

or 

p R q: 

or 

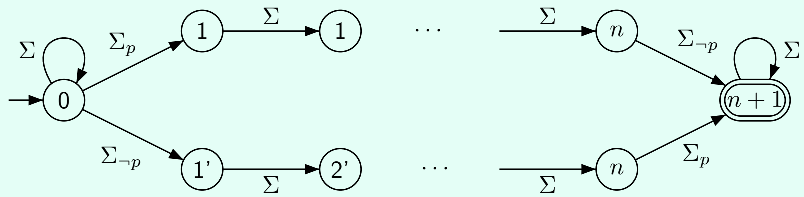
Büchi automata

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

Exercise:

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 $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$ ...

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

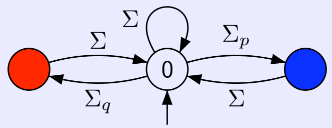
Generalized Büchi automata

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.

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

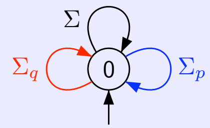


Definition: acceptance on transitions

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

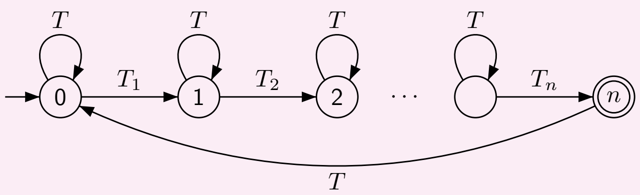
An infinite run σ is successful if it uses infinitely many transitions from each T_i.

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



GBA to BA

Proof: Synchronized product with $\mathcal{B}$



Transitions: $\dfrac{t = s_1 \xrightarrow{a} s_1' \in \mathcal{A} \wedge s_2 \xrightarrow{t} s_2' \in \mathcal{B}}{(s_1, s_2) \xrightarrow{a} (s_1', s_2')}$

Accepting states: $Q \times \{n\}$

Negative normal form

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

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

Proposition: Any formula can be transformed in NNF

$$\begin{array}{cc}
\neg(\varphi \vee \psi) \equiv (\neg\varphi) \wedge (\neg\psi) & \neg(\varphi \wedge \psi) \equiv (\neg\varphi) \vee (\neg\psi) \\
\neg(\varphi\,\mathsf{U}\,\psi) \equiv (\neg\varphi)\,\mathsf{R}\,(\neg\psi) & \neg(\varphi\,\mathsf{R}\,\psi) \equiv (\neg\varphi)\,\mathsf{U}\,(\neg\psi) \\
\neg\,\mathsf{X}\,\varphi \equiv \mathsf{X}\,\neg\varphi & \neg\neg\varphi \equiv \varphi
\end{array}$$

This does not increase the number of Temporal subformulae.

Temporal formulae

Definition: Temoral formulae

- literals
- formulae with outermost connective X, U or R.

Reducing the number of temporal subformulae

$$(\mathsf{X}\,\varphi) \wedge (\mathsf{X}\,\psi) \equiv \mathsf{X}(\varphi \wedge \psi) \qquad (\mathsf{X}\,\varphi)\,\mathsf{U}\,(\mathsf{X}\,\psi) \equiv \mathsf{X}(\varphi\,\mathsf{U}\,\psi)$$
$$(\varphi\,\mathsf{R}\,\psi_1) \wedge (\varphi\,\mathsf{R}\,\psi_2) \equiv \varphi\,\mathsf{R}\,(\psi_1 \wedge \psi_2) \qquad (\varphi_1\,\mathsf{R}\,\psi) \vee (\varphi_2\,\mathsf{R}\,\psi) \equiv (\varphi_1 \vee \varphi_2)\,\mathsf{R}\,\psi$$
$$(\mathsf{G}\,\varphi) \wedge (\mathsf{G}\,\psi) \equiv \mathsf{G}(\varphi \wedge \psi) \qquad \mathsf{G}\,\mathsf{F}\,\varphi \vee \mathsf{G}\,\mathsf{F}\,\psi \equiv \mathsf{G}\,\mathsf{F}(\varphi \vee \psi)$$

From LTL to BA (See [6])

Definition:

- $Z \subseteq \mathrm{NNF}$ is consistent if $\bot \notin Z$ and $\{p, \neg p\} \not\subseteq Z$ for all $p \in \mathrm{AP}$.
- For $Z \subseteq \mathrm{NNF}$, we define $\bigwedge Z = \bigwedge_{\psi \in Z} \psi$.
  Note that $\bigwedge \emptyset = \top$ and if Z is inconsistent then $\bigwedge Z \equiv \bot$.

Intuition for the BA $\mathcal{A}_\varphi = (Q, \Sigma, I, T, (T_\alpha)_{\alpha \in \mathsf{U}(\varphi)})$

Let $\varphi \in \mathrm{NNF}$ be a formula.

- $\mathrm{sub}(\varphi)$ is the set of sub-formulae of φ.
- $\mathsf{U}(\varphi)$ the set of until sub-formulae of φ.
- We construct a BA $\mathcal{A}_\varphi$ with $Q = 2^{\mathrm{sub}(\varphi)}$ and $I = \{\varphi\}$.
- A state $Z \subseteq \mathrm{sub}(\varphi)$ is a set of obligations.
- If $Z \subseteq \mathrm{sub}(\varphi)$, we want $\mathcal{L}(\mathcal{A}_\varphi^Z) = \{u \in \Sigma^\omega \mid u, 0 \models \bigwedge Z\}$
  where $\mathcal{A}_\varphi^Z$ is $\mathcal{A}_\varphi$ using Z as unique initial state.

Reduced formulae

Definition: Reduced formulae

- A formula is reduced if it is a literal (p or $\neg p$) or a next-formula ($\mathsf{X}\,\beta$).
- $Z \subseteq \mathrm{NNF}$ is reduced if all formulae in Z are reduced,

For $Z \subseteq \mathrm{NNF}$ consistent and reduced, we define

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

Lemma: Next step

Let $Z \subseteq \mathrm{NNF}$ be consistent and reduced.
Let $u = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ and $n \geq 0$. Then

$$u, n \models \bigwedge Z \qquad \text{iff} \qquad u, n+1 \models \bigwedge \mathrm{next}(Z) \text{ and } a_n \in \Sigma_Z$$

- $\mathcal{A}_\varphi$ will have transitions $Z \xrightarrow{\Sigma_Z} \mathrm{next}(Z)$.
  Note that $\emptyset \xrightarrow{\Sigma} \emptyset$.
- Problem: $\mathrm{next}(Z)$ is not reduced in general (it may even be inconsistent).

Reduction rules

Definition: Reduction of obligations to literals and next-formulae

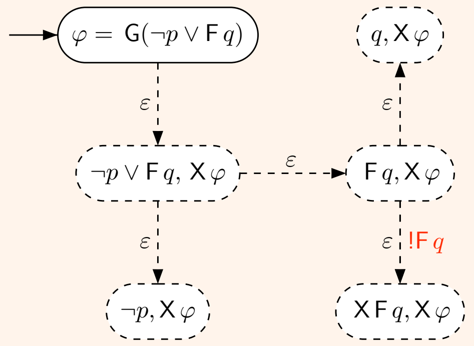
Let $Y \subseteq \mathrm{NNF}$ and let $\psi \in Y$ maximal not reduced.

$$\begin{array}{lll}
\text{If } \psi = \psi_1 \wedge \psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_1, \psi_2\} \\
\text{If } \psi = \psi_1 \vee \psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_1\} \\
 & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_2\} \\
\text{If } \psi = \psi_1\,\mathsf{R}\,\psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_1, \psi_2\} \\
 & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_2, \mathsf{X}\,\psi\} \\
\text{If } \psi = \mathsf{G}\,\psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_2, \mathsf{X}\,\psi\} \\
\text{If } \psi = \psi_1\,\mathsf{U}\,\psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_2\} \\
 & Y \xrightarrow[!\psi]{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_1, \mathsf{X}\,\psi\} \\
\text{If } \psi = \mathsf{F}\,\psi_2: & Y \xrightarrow{\varepsilon} & Y \setminus \{\psi\} \cup \{\psi_2\} \\
 & Y \xrightarrow[!\psi]{\varepsilon} & Y \setminus \{\psi\} \cup \{\mathsf{X}\,\psi\}
\end{array}$$

Note the mark $!\psi$ on the second transitions for U and F.

Reduction rules

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



State = set of obligations.

Reduce obligations to literals and next-formulae.

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

Reduction

Lemma: Soundness

- if there is only one rule $Y \xrightarrow{\varepsilon} Y_1$ then $\bigwedge Y \equiv \bigwedge Y_1$
- if there are two rules $Y \xrightarrow{\varepsilon} Y_1$ and $Y \xrightarrow{\varepsilon} Y_2$ then $\bigwedge Y \equiv \bigwedge Y_1 \vee \bigwedge Y_2$

Definition:

For $Y \subseteq \mathrm{NNF}$ and $\alpha \in \mathsf{U}(\varphi)$, let

$$\mathrm{Red}(Y) = \{Z \text{ consistent and reduced} \mid \exists\, Y \xrightarrow[*]{\varepsilon} Z\}$$
$$\mathrm{Red}_\alpha(Y) = \{Z \text{ consistent and reduced} \mid \exists\, Y \xrightarrow[*]{\varepsilon} Z \text{ without using an edge marked with } !\alpha\}$$

Lemma: Soundness

- Let $Y \subseteq \mathrm{NNF}$, then $\bigwedge Y \equiv \bigvee_{Z \in \mathrm{Red}(Y)} \bigwedge Z$
- Let $u = a_0 a_1 a_2 \cdots \in \Sigma^\omega$ and $n \geq 0$ with $u, n \models \bigwedge Y$.
  Then, $\exists Z \in \mathrm{Red}(Y)$ such that $u, n \models \bigwedge Z$
  and $Z \in \mathrm{Red}_\alpha(Y)$ for all $\alpha = \alpha_1 \,\mathsf{U}\, \alpha_2 \in \mathsf{U}(\varphi)$ such that $u, n \models \alpha_2$.

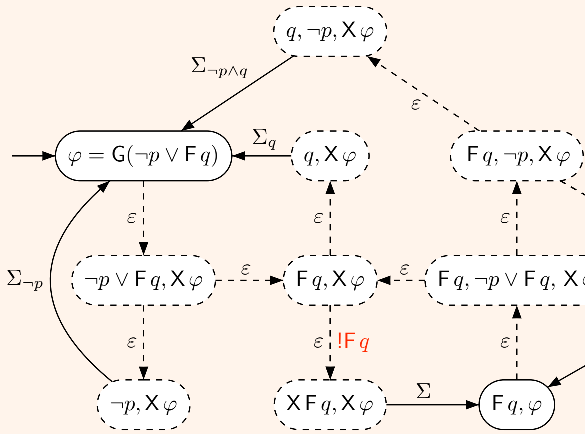
Automaton $\mathcal{A}_\varphi$

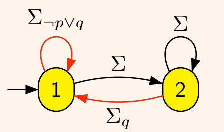
Definition: Automaton $\mathcal{A}_\varphi$

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

Automaton $\mathcal{A}_\varphi$

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





Transition = check literals and move forward.

Simplification

Correctness of $\mathcal{A}_\varphi$

Proposition: $\mathcal{L}(\varphi) \subseteq \mathcal{L}(\mathcal{A}_\varphi)$

Lemma:

Let $\rho = Y_0 \xrightarrow{a_0} Y_1 \xrightarrow{a_1} Y_2 \cdots$ be an accepting run of $\mathcal{A}_\varphi$ on $u = a_0 a_1 a_2 \cdots \in \Sigma^\omega$.

Then, $\forall \psi \in \mathrm{sub}(\varphi)$, $\forall n \geq 0$, $\forall Y_n \xrightarrow[*]{\varepsilon} Y \xrightarrow[*]{\varepsilon} Z$ with $a_n \in \Sigma_Z$, $Y_{n+1} = \mathrm{next}(Z)$

$$\psi \in Y \quad \Longrightarrow \quad u, n \models \psi$$

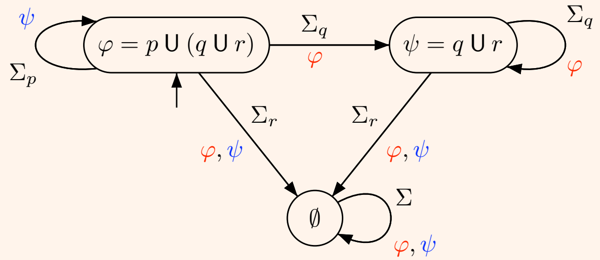
Corollary: $\mathcal{L}(\mathcal{A}_\varphi) \subseteq \mathcal{L}(\varphi)$

Example with two until sub-formulae

Example: Nested until: $\varphi = p \,\mathsf{U}\, \psi$ with $\psi = q \,\mathsf{U}\, r$

$\mathrm{Red}(\{\varphi\}) = \{\{p, \mathsf{X}\,\varphi\}, \{q, \mathsf{X}\,\psi\}, \{r\}\}$
$\mathrm{Red}_\varphi(\{\varphi\}) = \{\{q, \mathsf{X}\,\psi\}, \{r\}\}$
$\mathrm{Red}_\psi(\{\varphi\}) = \{\{p, \mathsf{X}\,\varphi\}, \{r\}\}$

$\mathrm{Red}(\{\psi\}) = \{\{q, \mathsf{X}\,\psi\}, \{r\}\}$
$\mathrm{Red}_\varphi(\{\psi\}) = \{\{q, \mathsf{X}\,\psi\}, \{r\}\}$
$\mathrm{Red}_\psi(\{\psi\}) = \{\{r\}\}$



Satisfiability and Model Checking

Corollary: PSPACE upper bound for satisfiability and model checking

- Let $\varphi \in \mathrm{LTL}$, we can check whether φ is satisfiable (or valid) in space polynomial in $|\varphi|$.
- Let $\varphi \in \mathrm{LTL}$ and $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure. We can check whether $M \models_\forall \varphi$ (or $M \models_\exists \varphi$) in space polynomial in $|\varphi| + \log |M|$.

Proof:

For $M \models_\forall \varphi$ we construct a synchronized product $M \otimes \mathcal{A}_{\neg\varphi}$:

Transitions: $\dfrac{s \xrightarrow{a} s' \in M \quad \wedge \quad Y \xrightarrow{\ell(s)} Y' \in \mathcal{A}_{\neg\varphi}}{(s, Y) \xrightarrow{a} (s', Y')}$

Acceptance conditions: inherited from $\mathcal{A}_{\neg\varphi}$.

Check $M \otimes \mathcal{A}_{\neg\varphi}$ for emptiness.

On the fly simplifications $\mathcal{A}_\varphi$

Built-in: reduction of a maximal formula.

Definition: Additional reduction rules

If $\bigwedge Y \equiv \bigwedge Y'$ then we may use $Y \xrightarrow{\varepsilon} Y'$.

Remark: checking equivalence is as hard as building the automaton.
Hence we only use syntactic equivalences.

If $\psi = \psi_1 \vee \psi_2$ and $\psi_1 \in Y$ or $\psi_2 \in Y$: $\quad Y \xrightarrow{\varepsilon} Y \setminus \{\psi\}$

If $\psi = \psi_1 \,\mathsf{U}\, \psi_2$ and $\psi_2 \in Y$: $\quad Y \xrightarrow{\varepsilon} Y \setminus \{\psi\}$

If $\psi = \psi_1 \,\mathsf{R}\, \psi_2$ and $\psi_1 \in Y$: $\quad Y \xrightarrow{\varepsilon} Y \setminus \{\psi\} \cup \{\psi_2\}$

On the fly simplifications $\mathcal{A}_\varphi$

Definition: Merging equivalent states

Let A $= (Q, \Sigma, I, T, T_1, \ldots, T_n)$ and $s_1, s_2 \in Q$.
We can merge s_1 and s_2 if they have the same outgoing transitions:
$\forall a \in \Sigma$, $\forall s \in Q$,

$$\begin{array}{rll} & (s_1, a, s) \in T \iff (s_2, a, s) \in T & \\ \text{and} & (s_1, a, s) \in T_i \iff (s_2, a, s) \in T_i & \text{for all } 1 \le i \le n. \end{array}$$

Remark: Sufficient condition

Two states Y, Y' of $\mathcal{A}_\varphi$ have the same outgoing transition if

$$\begin{array}{rll} & \mathrm{Red}(Y) = \mathrm{Red}(Y') & \\ \text{and} & \mathrm{Red}_\alpha(Y) = \mathrm{Red}_\alpha(Y') & \text{for all } \alpha \in \mathsf{U}(\varphi). \end{array}$$

Example: Let $\varphi = \mathsf{G}\,\mathsf{F}\,p \wedge \mathsf{G}\,\mathsf{F}\,q$.

Without merging states $\mathcal{A}_\varphi$ has 4 states.
These 4 states have the same outgoing transitions.
The simplified automaton has only one state.

Other constructions

- Tableau construction. See for instance [8]
  - $+$: Easy definition, easy proof of correctness
  - $+$: Works both for future and past modalities
  - $-$: Inefficient without optimizations
- Using Very Weak Alternating Automata [7].
  - $+$: Very efficient
  - $-$: Only for future modalities
- The domain is still very active.
- See other references in [6].

[7] P. Gastin and D. Oddoux.
Fast LTL to Büchi automata translation.
In *CAV'01*, vol. 2102, *Lecture Notes in Computer Science*, pp. 53–65. Springer, (2001).
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

[8] P. Wolper.
The tableau method for temporal logic: An overview,
Logique et Analyse. **110–111**, 119–136, (1985).

$\mathrm{MC}^{\exists}(\mathsf{X},\mathsf{U}) \le_P \mathrm{SAT}(\mathsf{X},\mathsf{U})$ (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, $M \models_\exists \varphi$ iff $\psi_M \wedge \varphi$ is satisfiable
$M \models_\forall \varphi$ iff $\psi_M \wedge \neg\varphi$ is not satisfiable

Remark: we also have $\mathrm{MC}^{\exists}(\mathsf{X},\mathsf{F}) \le_P \mathrm{SAT}(\mathsf{X},\mathsf{F})$.

QBF Quantified Boolean Formulae

Definition: QBF

Input: A formula $\gamma = Q_1 x_1 \cdots Q_n x_n \gamma'$ with $\gamma' = \bigwedge_{1\le i\le m} \bigvee_{1\le j\le k_i} a_{ij}$
$Q_i \in \{\forall, \exists\}$ and $a_{ij} \in \{x_1, \neg x_1, \ldots, x_n, \neg x_n\}$.

Question: Is γ valid?

Definition:

An assignment of the variables $\{x_1, \ldots, x_n\}$ is a word $v = v_1 \cdots v_n \in \{0,1\}^\omega$.
We write $v[i]$ for the prefix of length i.
Let $V \subseteq \{0,1\}^\omega$ be a set of assignments.

- V is valid (for γ') if $v \models \gamma'$ for all $v \in V$,
- V is closed (for γ) if $\forall v \in V$, $\forall 1 \le i \le n$ s.t. $Q_i = \forall$,
  $\exists v' \in V$ s.t. $v[i-1] = v'[i-1]$ and $\{v_i, v'_i\} = \{0,1\}$.

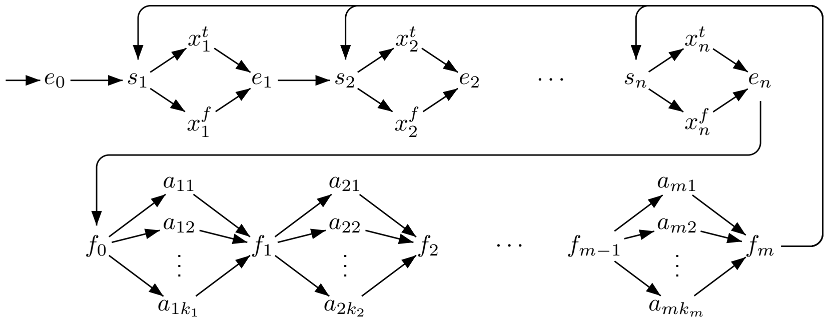
Proposition:

γ is valid iff $\exists V \subseteq \{0,1\}^n$ s.t. V is nonempty valid and closed

$\mathrm{QBF} \leq_P \mathrm{MC}^\exists(\mathsf{U})$ (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 a_{ij} literals.

Consider the KS M:



Let $\psi_{ij} = \begin{cases} \mathsf{G}(x_k^f \to s_k \mathrel{\mathsf{R}} \neg a_{ij}) & \text{if } a_{ij} = x_k \\ \mathsf{G}(x_k^t \to s_k \mathrel{\mathsf{R}} \neg a_{ij}) & \text{if } a_{ij} = \neg x_k \end{cases}$ 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 $M \models_\exists \psi \wedge \varphi$.

$\mathrm{QBF} \leq_P \mathrm{MC}^\exists(\mathsf{U})$ (Sistla & Clarke 85)

Proof: If $M \models_\exists \psi \wedge \varphi$ then γ is valid

Each finite path $\tau = e_0 \xrightarrow{*} f_m$ in M defines a valuation v^τ by:

$$v_k^\tau = \begin{cases} 1 & \text{if } \tau, |\tau| \models \neg s_k \mathrel{\mathsf{S}} x_k^t \\ 0 & \text{if } \tau, |\tau| \models \neg s_k \mathrel{\mathsf{S}} x_k^f \end{cases}$$

Claim: if $\tau \models \psi$ then $v^\tau \models \gamma'$.

Let σ be an initial infinite path of M s.t. $\sigma, 0 \models \psi \wedge \varphi$.
Let $V = \{v^\tau \mid \tau = e_0 \xrightarrow{*} f_m \text{ is a prefix of } \sigma\}$.

Claim: V is nonempty, valid and closed.

$\mathrm{QBF} \leq_P \mathrm{MC}^\exists(\mathsf{U})$ (Sistla & Clarke 85)

Proof: If γ is valid then $M \models_\exists \psi \wedge \varphi$

Let $V \subseteq \{0,1\}^\omega$ be nonempty, valid and closed.

First ingredient: extension of a run.
Assume $\tau = e_0 \xrightarrow{*} f_m$ satisfies $v^\tau \in V$ and $\tau, 0 \models \psi$.
Let $1 \leq i \leq n$ with $Q_i = \forall$.
Let $v' \in V$ s.t. $v'[i-1] = v[i-1]$ and $\{v_i, v'_i\} = \{0,1\}$.
We can extend τ in $\tau' = \tau \to e_i \xrightarrow{*} e_n \to f_0 \xrightarrow{*} f_m$ with $v^{\tau'} = v'$ and $\tau', 0 \models \psi$.
We say that τ' is the extension of τ wrt. i

Second step: the sequence of indices for the extensions.
Let $1 \leq i_\ell < \cdots < i_1 \leq n$ be the indices of universal quantifications ($Q_{i_j} = \forall$).
Define by induction $w_1 = i_1$ and if $k < \ell$, $w_{k+1} = w_k i_{k+1} w_k$. Let $w = (w_\ell 1)^\omega$.

Final step: the infinite run.
Let $v \in V \neq \emptyset$ and let $\tau = e_0 \xrightarrow{*} f_m$ with $v^\tau \in V$ and $\tau, 0 \models \psi$.
We build an infinite run σ by extending τ inductively wrt. the sequence of indices defined by w.

Claim: $\sigma, 0 \models \psi \wedge \varphi$.

Complexity of LTL

Theorem: Complexity of LTL

The following problems are PSPACE-complete:

- $\mathrm{SAT}(\mathrm{LTL}(\mathsf{X},\mathsf{U},\mathsf{Y},\mathsf{S}))$, $\mathrm{MC}^\forall(\mathrm{LTL}(\mathsf{X},\mathsf{U},\mathsf{Y},\mathsf{S}))$, $\mathrm{MC}^\exists(\mathrm{LTL}(\mathsf{X},\mathsf{U},\mathsf{Y},\mathsf{S}))$
- $\mathrm{SAT}(\mathrm{LTL}(\mathsf{X},\mathsf{F}))$, $\mathrm{MC}^\forall(\mathrm{LTL}(\mathsf{X},\mathsf{F}))$, $\mathrm{MC}^\exists(\mathrm{LTL}(\mathsf{X},\mathsf{F}))$
- $\mathrm{SAT}(\mathrm{LTL}(\mathsf{U}))$, $\mathrm{MC}^\forall(\mathrm{LTL}(\mathsf{U}))$, $\mathrm{MC}^\exists(\mathrm{LTL}(\mathsf{U}))$
- The restriction of the above problems to a unique propositional variable

The following problems are NP-complete:

- $\mathrm{SAT}(\mathrm{LTL}(\mathsf{F}))$, $\mathrm{MC}^\exists(\mathrm{LTL}(\mathsf{F}))$

Some original References

[9] A. Sistla and E. Clarke.
The complexity of propositional linear temporal logic.
Journal of the Association for Computing Machinery. **32** (3), 733–749, (1985).

[10] O. Lichtenstein and A. Pnueli.
Checking that finite state concurrent programs satisfy their linear specification.
In *ACM Symposium PoPL'85*, 97–107.

[11] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi.
On the temporal analysis of fairness.
In *7th Annual ACM Symposium PoPL'80*, 163–173. ACM Press.

[12] D. Gabbay.
The declarative past and imperative future: Executable temporal logics for interactive systems.
In *Temporal Logics in Specifications, April 87.* LNCS 398, 409–448, 1989.

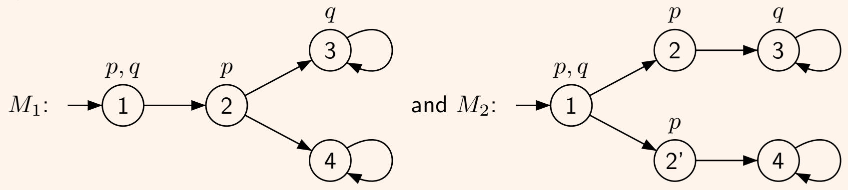
Outline

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 formulae.

We need quantifications on runs: $\varphi = \mathsf{AG}(p \to \mathsf{EF}\, q)$
- E: for some infinite run
- A: for all infinite runs

CTL^* (Emerson & Halpern 86)

Definition: Syntax of the Computation Tree Logic CTL^*

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

Definition: Semantics:

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

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

Example: 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

Remark: $\mathsf{A}\,\varphi \equiv \neg\,\mathsf{E}\,\neg\varphi$

State formulae and path formulae

Definition: State formulae

$\varphi \in \mathrm{CTL}^*$ is a state formula if $\forall M, \sigma, \sigma', i, j$ such that $\sigma(i) = \sigma'(j)$ we have

$$M, \sigma, i \models \varphi \iff M, \sigma', j \models \varphi$$

If φ is a state formula and $M = (S, T, I, \mathrm{AP}, \ell)$, define

$$[\![\varphi]\!]^M = \{s \in S \mid M, s \models \varphi\}$$

Example: State formulae

Formulae of the form p or $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ are state formulae.
State formulae are closed under boolean connectives.

$$[\![p]\!] = \{s \in S \mid p \in \ell(s)\} \qquad [\![\neg\varphi]\!] = S \setminus [\![\varphi]\!] \qquad [\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$$

Definition: Alternative syntax

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

Model checking of CTL^*

Definition: Existential and universal model checking

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

$M \models_\exists \varphi$ if $M, \sigma, 0 \models \varphi$ for some initial infinite run σ of M.
$M \models_\forall \varphi$ if $M, \sigma, 0 \models \varphi$ for all initial infinite run σ of M.

Remark:

$M \models_\exists \varphi$ iff $I \cap [\![\mathsf{E}\,\varphi]\!] \neq \emptyset$

$M \models_\forall \varphi$ iff $I \subseteq [\![\mathsf{A}\,\varphi]\!]$

$M \models_\forall \varphi$ iff $M \not\models_\exists \neg\varphi$

Definition: Model checking problems $\mathrm{MC}^\forall_{\mathrm{CTL}^*}$ and $\mathrm{MC}^\exists_{\mathrm{CTL}^*}$

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

Question: Does $M \models_\forall \varphi$? or Does $M \models_\exists \varphi$?

Complexity of CTL^*

Theorem

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

Proof:

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

PSPACE-easiness: reduction to LTL-model checking by inductive eliminations of path quantifications.

$\mathrm{MC}^\forall_{\mathrm{CTL}^*}$ in PSPACE

Proof:

For $\mathcal{Q} \in \{\exists, \forall\}$, let $\mathrm{MC}^{\mathcal{Q}}_{\mathrm{LTL}}(M, s, \varphi)$ be the function which computes in polynomial space whether $M, s \models_{\mathcal{Q}} \varphi$.

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

$\mathrm{MC}^\forall_{\mathrm{CTL}^*}(M, s, \varphi)$
  If E, A do not occur in φ then return $\mathrm{MC}^\forall_{\mathrm{LTL}}(M, s, \varphi)$ fi
  Let $\mathcal{Q}\psi$ be a subformula of φ with $\psi \in \mathrm{LTL}$ and $\mathcal{Q} \in \{\mathsf{E}, \mathsf{A}\}$
  Let $p_{\mathcal{Q}\psi}$ be a new propositional variable
  Define $\ell' : S \to 2^{\mathrm{AP}'}$ with $\mathrm{AP}' = \mathrm{AP} \uplus \{p_{\mathcal{Q}\psi}\}$ by
    $\ell'(t) \cap \mathrm{AP} = \ell(t)$ and $p_{\mathcal{Q}\psi} \in \ell'(t)$ iff $\mathrm{MC}^{\mathcal{Q}}_{\mathrm{LTL}}(M, t, \psi)$
  Let $M' = (S, T, I, \mathrm{AP}', \ell')$
  Let $\varphi' = \varphi[p_{\mathcal{Q}\psi}/\mathcal{Q}\psi]$ be obtained from φ by replacing each $\mathcal{Q}\psi$ by $p_{\mathcal{Q}\psi}$
  Return $\mathrm{MC}^\forall_{\mathrm{CTL}^*}(M', s, \varphi')$

Satisfiability for CTL^*

Definition: $\mathrm{SAT}(\mathrm{CTL}^*)$

Input: A formula $\varphi \in \mathrm{CTL}^*$

Question: Existence of a model M and a run σ such that $M, \sigma, 0 \models \varphi$?

Theorem

The satisfiability problem for CTL^* is 2-EXPTIME-complete

CTL (Clarke & Emerson 81)

Definition: Computation Tree Logic (CTL)

Syntax:

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

The semantics is inherited from CTL^*.

Remark: All CTL formulae are state formulae

Examples: Macros

- $\mathsf{EF}\,\varphi = \mathsf{E}\top\mathsf{U}\,\varphi$ and $\mathsf{AF}\,\varphi = \mathsf{A}\top\mathsf{U}\,\varphi$
- $\mathsf{EG}\,\varphi = \neg\,\mathsf{AF}\,\neg\varphi$ and $\mathsf{AG}\,\varphi = \neg\,\mathsf{EF}\,\neg\varphi$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{EF}\,\mathrm{grant})$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$

CTL (Clarke & Emerson 81)

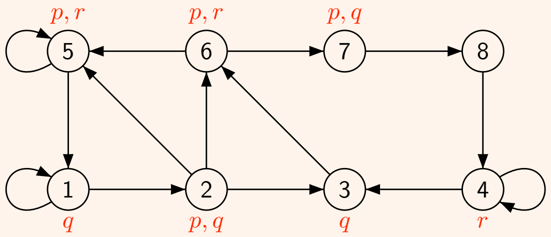
Definition: Semantics

All CTL-formulae are state formulae. Hence, we have a simpler semantics.
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	$\exists s = s_0 \to s_1 \to s_2 \to \cdots s_j$ finite path, with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \le k < j$
$s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$	if	$\forall s = s_0 \to s_1 \to s_2 \to \cdots$ infinite path, $\exists j \ge 0$ with $s_j \models \psi$ and $s_k \models \varphi$ for all $0 \le k < j$

CTL (Clarke & Emerson 81)

Example:

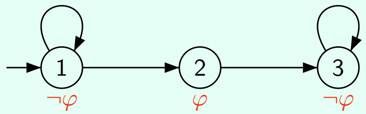


$$[\![\mathsf{EX}\,p]\!] = \{1,2,3,5,6\}$$
$$[\![\mathsf{AX}\,p]\!] = \{3,6\}$$
$$[\![\mathsf{EF}\,p]\!] = \{1,2,3,4,5,6,7,8\}$$
$$[\![\mathsf{AF}\,p]\!] = \{2,3,5,6,7\}$$
$$[\![\mathsf{E}\,q\,\mathsf{U}\,r]\!] = \{1,2,3,4,5,6\}$$
$$[\![\mathsf{A}\,q\,\mathsf{U}\,r]\!] = \{2,3,4,5,6\}$$

CTL (Clarke & Emerson 81)

Remark: Equivalent formulae

- $\mathsf{AX}\,\varphi = \neg\,\mathsf{EX}\,\neg\varphi$,
- $\neg(\varphi\,\mathsf{U}\,\psi) = \mathsf{G}\,\neg\psi \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$
- $\mathsf{A}\,\varphi\,\mathsf{U}\,\psi = \neg\,\mathsf{EG}\,\neg\psi \wedge \neg\,\mathsf{E}\,\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi)$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) = \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi = \mathsf{AG}\,\mathsf{AF}\,\varphi$ — 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

Definition: Existential and universal model checking

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

$M \models_{\exists} \varphi$ if $M, s \models \varphi$ for some $s \in I$.
$M \models_{\forall} \varphi$ if $M, s \models \varphi$ for all $s \in I$.

Remark:

$M \models_{\exists} \varphi$ iff $I \cap [\![\varphi]\!] \neq \emptyset$

$M \models_{\forall} \varphi$ iff $I \subseteq [\![\varphi]\!]$

$M \models_{\forall} \varphi$ iff $M \not\models_{\exists} \neg\varphi$

Definition: Model checking problems $\mathrm{MC}^{\forall}_{\mathrm{CTL}}$ and $\mathrm{MC}^{\exists}_{\mathrm{CTL}}$

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

Question: Does $M \models_{\forall} \varphi$? or Does $M \models_{\exists} \varphi$?

Model checking of CTL

Theorem

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

Proof:

Compute $[\![\varphi]\!]$ by induction on the formula.
The set $[\![\varphi]\!]$ is represented by a boolean array.

Definition: procedure semantics(φ)

case $\varphi = p \in \mathrm{AP}$
$[\![\varphi]\!] := \{s \in S \mid p \in \ell(s)\}$ — $\mathcal{O}(|S|)$

case $\varphi = \neg\varphi_1$
semantics(φ_1)
$[\![\varphi]\!] := S \setminus [\![\varphi_1]\!]$ — $\mathcal{O}(|S|)$

case $\varphi = \varphi_1 \vee \varphi_2$
semantics(φ_1); semantics(φ_2)
$[\![\varphi]\!] := [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$ — $\mathcal{O}(|S|)$

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = EX\,\varphi_1$
semantics(φ_1)
$[\![\varphi]\!] := \emptyset$ — $\mathcal{O}(|S|)$
for all $(s, t) \in T$ do if $t \in [\![\varphi_1]\!]$ then $[\![\varphi]\!] := [\![\varphi]\!] \cup \{s\}$ — $\mathcal{O}(|T|)$

case $\varphi = AX\,\varphi_1$
semantics(φ_1)
$[\![\varphi]\!] := S$ — $\mathcal{O}(|S|)$
for all $(s, t) \in T$ do if $t \notin [\![\varphi_1]\!]$ then $[\![\varphi]\!] := [\![\varphi]\!] \setminus \{s\}$ — $\mathcal{O}(|T|)$

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = E\varphi_1$ U φ_2 — $\mathcal{O}(|S|+|T|)$
semantics(φ_1); semantics(φ_2)
$L := [\![\varphi_2]\!]$ // the set L is the "todo" list — $\mathcal{O}(|S|)$
$Z := \emptyset$ // the set Z is the "done" list — $\mathcal{O}(|S|)$
while $L \neq \emptyset$ do — $|S|$ times
Invariant: $[\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap T^{-1}(Z)) \subseteq Z \cup L \subseteq [\![\mathsf{E}\,\varphi_1\,\mathsf{U}\,\varphi_2]\!]$
take $t \in L$; $L := L \setminus \{t\}$; $Z := Z \cup \{t\}$ — $\mathcal{O}(1)$
for all $s \in T^{-1}(t)$ do — $|T|$ times
if $s \in [\![\varphi_1]\!] \setminus (Z \cup L)$ then $L := L \cup \{s\}$
$[\![\varphi]\!] := Z$ // Z is only used to make the invariant clear

Model checking of CTL

Definition: procedure semantics(φ)

Replacing $Z \cup L$ by $[\![\varphi]\!]$

case $\varphi = E\varphi_1$ U φ_2 — $\mathcal{O}(|S|+|T|)$
semantics(φ_1); semantics(φ_2)
$L := [\![\varphi_2]\!]$ // the set L is imlemented with a list — $\mathcal{O}(|S|)$
$[\![\varphi]\!] := [\![\varphi_2]\!]$ — $\mathcal{O}(|S|)$
while $L \neq \emptyset$ do — $|S|$ times
take $t \in L$; $L := L \setminus \{t\}$ — $\mathcal{O}(1)$
for all $s \in T^{-1}(t)$ do — $|T|$ times
if $s \in [\![\varphi_1]\!] \setminus [\![\varphi]\!]$ then $L := L \cup \{s\}$; $[\![\varphi]\!] := [\![\varphi]\!] \cup \{s\}$ — $\mathcal{O}(1)$

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = A\varphi_1$ U φ_2 — $\mathcal{O}(|S|+|T|)$
semantics(φ_1); semantics(φ_2)
$L := [\![\varphi_2]\!]$ // the set L is the "todo" list — $\mathcal{O}(|S|)$
$Z := \emptyset$ // the set Z is the "done" list — $\mathcal{O}(|S|)$
for all $s \in S$ do $c[s] := |T(s)|$ — $\mathcal{O}(|S|)$
while $L \neq \emptyset$ do — $|S|$ times
Invariant: $\forall s \in S,\ c[s] = |T(s) \setminus Z|$ and
$[\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap \{s \in S \mid T(s) \subseteq Z\}) \subseteq Z \cup L \subseteq [\![\mathsf{A}\,\varphi_1\,\mathsf{U}\,\varphi_2]\!]$
take $t \in L$; $L := L \setminus \{t\}$; $Z := Z \cup \{t\}$ — $\mathcal{O}(1)$
for all $s \in T^{-1}(t)$ do — $|T|$ times
$c[s] := c[s] - 1$ — $\mathcal{O}(1)$
if $c[s] = 0 \wedge s \in [\![\varphi_1]\!] \setminus (Z \cup L)$ then $L := L \cup \{s\}$
$[\![\varphi]\!] := Z$ // Z is only used to make the invariant clear

Model checking of CTL

Definition: procedure semantics(φ)

Replacing $Z \cup L$ by $[\![\varphi]\!]$

case $\varphi = A\varphi_1$ U φ_2 — $\mathcal{O}(|S|+|T|)$
semantics(φ_1); semantics(φ_2)
$L := [\![\varphi_2]\!]$ // the set L is imlemented with a list — $\mathcal{O}(|S|)$
$[\![\varphi]\!] := [\![\varphi_2]\!]$ — $\mathcal{O}(|S|)$
for all $s \in S$ do $c[s] := |T(s)|$ — $\mathcal{O}(|S|)$
while $L \neq \emptyset$ do — $|S|$ times
take $t \in L$; $L := L \setminus \{t\}$ — $\mathcal{O}(1)$
for all $s \in T^{-1}(t)$ do — $|T|$ times
$c[s] := c[s] - 1$ — $\mathcal{O}(1)$
if $c[s] = 0 \wedge s \in [\![\varphi_1]\!] \setminus [\![\varphi]\!]$ then — $\mathcal{O}(1)$
$L := L \cup \{s\}$; $[\![\varphi]\!] := [\![\varphi]\!] \cup \{s\}$ — $\mathcal{O}(1)$

Complexity of CTL

Definition: SAT(CTL)

Input: A formula $\varphi \in \mathrm{CTL}$

Question: Existence of a model M and a state s such that $M, s \models \varphi$?

Theorem: Complexity

- The model checking problem for CTL is PTIME-complete.
- The satisfiability problem for CTL is EXPTIME-complete.

fairness

Example: 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: $\bigwedge_i \neg\,\mathsf{F}\,\mathsf{G}\,\mathrm{CS}_i = \bigwedge_i \mathsf{G}\,\mathsf{F}\,\neg\mathrm{CS}_i$

Definition: Fair Kripke structure

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

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

fair CTL

Definition: Syntax of fair-CTL

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

Definition: Semantics as a fragment of CTL*

Let $M = (S, T, I, \mathrm{AP}, \ell, F_1, \ldots, F_n)$ be a fair Kripke structure.

Then, $\mathsf{E}_f\,\varphi = \mathsf{E}(\mathrm{fair} \wedge \varphi)$ and $\mathsf{A}_f\,\varphi = \mathsf{A}(\mathrm{fair} \to \varphi)$

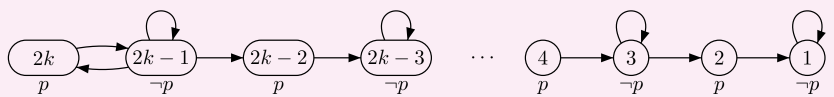
where $\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$

Lemma: CTL_f cannot be expressed in CTL

fair CTL

Proof: 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 $|\varphi| \le m \le k$ then

$$M_k, 2k \models \varphi \text{ iff } M_k, 2m \models \varphi$$

$$M_k, 2k-1 \models \varphi \text{ iff } M_k, 2m-1 \models \varphi$$

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

Model checking of CTL_f

Theorem

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

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

Compute the SCC of M with Tarjan's algorithm (in time $\mathcal{O}(|M|)$).

Let S' be the union of the (non trivial) 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.

Model checking of CTL_f

Proof: 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 $\mathsf{A} \varphi \mathsf{U} \psi = \neg \mathsf{EG} \neg\psi \wedge \neg \mathsf{E} \neg\psi \mathsf{U} (\neg\varphi \wedge \neg\psi)$

This formula also holds for fair quantifications A_f and E_f.

Hence, we only need to compute the semantics of $\mathsf{E}_f \mathsf{G} \varphi$.

Proof: Computation of $\mathsf{E}_f \mathsf{G} \varphi$

Let M_φ be the restriction of M to $[\![\varphi]\!]_f$.

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

Let S' be the union of the (non trivial) 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 $M_\varphi, s \models \mathsf{EF} S'$.

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