Initiation à la vérification
Basics of Verification

https://wikimpri.dptinfo.ens-cachan.fr/doku.php?id=cours:c-1-22

Paul Gastin

Paul.Gastin@lsv.ens-cachan.fr

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

MPRI – M1
2017 – 2018

Outline

Need for formal verification methods

Critical systems

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

Disastrous software bugs

https://en.wikipedia.org/wiki/List_of_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!
- 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

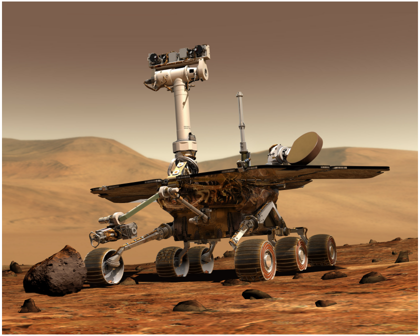
- Destroyed 37 seconds after launch (cost: 370 millions dollars).
- data conversion from a 64-bit floating point to 16-bit signed integer value caused a hardware exception (arithmetic overflow).
- Efficiency considerations had led to the disabling of the software handler (in Ada code) for this error 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

Spirit Rover (Mars Exploration), 2004

See `http://en.wikipedia.org/wiki/Spirit_rover`

- Landed on January 4, 2004.
- Ceased communicating on January 21.
- Flash memory management anomaly: too many files on the file system
- Resumed to working condition on February 6.



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

Based on

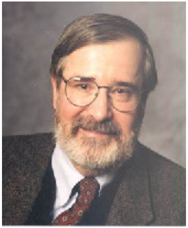
- A formal model of the system
- A formal semantics of the modelling language
- A formal specification

Complementary approaches

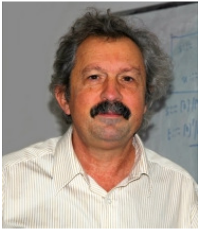
- Theorem prover
- Model checking
- Static analysis
- Test

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, ... See Modules 2.8 & 2.9
- Scale current tools to cope with real-size systems.
  Needs for modularity, abstractions, symmetries, ...

References

[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, 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.

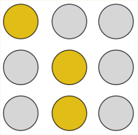
[5] Z. Manna and A. Pnueli.
Temporal Verification of Reactive Systems: Safety.
Springer, 1995.

[6] Ph. Schnoebelen.
The Complexity of Temporal Logic Model Checking.
In *AiML'02*, 393–436. King's College Publication, 2003.

Outline

Model and abstractions

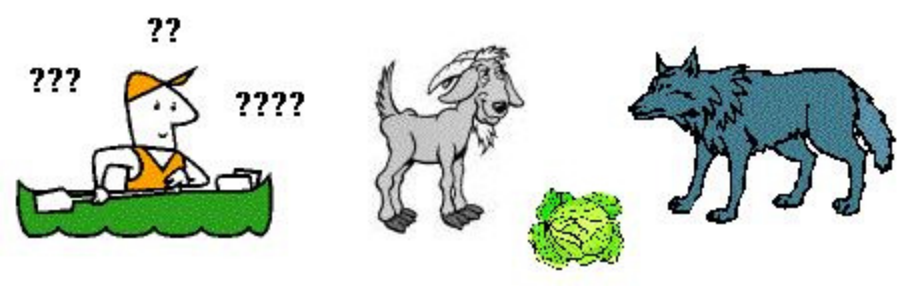
Example: Golden face



Each coin has a golden face and a silver face.
At each step, we may flip simultaneously the 3 coins of a line, column or diagonal.
Is it possible to have all coins showing its golden face ?
If yes, what is the smallest number of steps.

Model and Specification

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 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.

Every discrete system may be described with a TS.

Example: Digicode ABA

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

- Finite description with S, Σ, AP, ℓ as before
- $\mathcal{V}$: set of (typed) variables, e.g., boolean, [0..4], $\mathbb{N}$, ...
- Each variable $v \in \mathcal{V}$ has a domain D_v (finite or infinite). Let $D = \prod_{v\in\mathcal{V}} D_v$.
- Guard or Condition g with semantics $[\![g]\!] \subseteq D$ (predicate)
  Symbolic descriptions: $x < 5$, $x + y = 10$, ...
- Instruction or Update f with semantics $[\![f]\!] : D \to D$
  Symbolic descriptions: $x := 0$, $x := (y + 1)^2$, ...
- $T \subseteq S \times (\mathtt{Guard} \times \Sigma \times \mathtt{Update}) \times S$
  Symbolic descriptions: $s \xrightarrow{x<50,?\mathrm{coin},x:=x+\mathrm{coin}} s'$
- $I \subseteq S \times \mathtt{Guard}$
  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 such as $x > 0$.

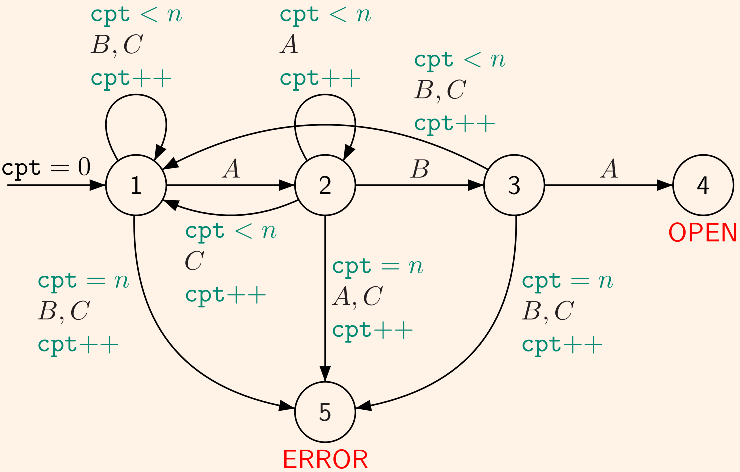
Programs = Kripke structures with variables

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

Example: GCD

TS with variables . . .

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 logic formulae

- 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 formulae with Binary Decision Diagrams.

Example: Boolean circuit: modulo 8 counter

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

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 restrictions

Atomic propositions are inherited from the local systems.

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

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

Synchronized products (semantics)

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

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

  T defined by $\dfrac{\forall i,\ p_i \xrightarrow{a_i} q_i \in T_i \vee (a_i = \varepsilon \wedge p_i = q_i)}{(p_1, \ldots, p_n) \xrightarrow{(a_1, \ldots, a_n)} (q_1, \ldots, q_n)}$

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

Synchronized products: restrictions of the general product.

Parallel compositions: 2 special cases

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

Restrictions

- on states: $S_{\text{restrict}} \subseteq S$
- on labels: $\Sigma_{\text{restrict}} \subseteq \Sigma$
- on transitions: $T_{\text{restrict}} \subseteq T$

Shared variables

Definition: Asynchronous product + shared variables

$\bar{s} = (s_1, \ldots, s_n)$ 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 without further restrictions: 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:                // i is not a variable
   loop forever
      req[i] := true; turn := 1-i
      wait until (turn = i or req[1-i] = false)
      Critical section
      req[i] := false
```

Exercise:

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

Atomicity

Example:

Intially $x = 1 \wedge y = 2$

Program P_1: $x := x + y \parallel y := x + y$

Program P_2: $\left(\begin{array}{c} \texttt{Load} R_1, x \\ \texttt{Add} R_1, y \\ \texttt{Store} R_1, x \end{array}\right) \parallel \left(\begin{array}{c} \texttt{Load} R_2, x \\ \texttt{Add} R_2, y \\ \texttt{Store} R_2, y \end{array}\right)$

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

Atomicity

Definition: Atomic statements: atomic(ES)

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

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

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

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

Example: Atomic statements

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

Communication by Rendez-vous

Restriction on transitions is universal but too low-level.

Definition: Rendez-vous

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

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

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

- It is a restriction on actions.
- Essential feature of process algebra.

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

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

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 variable 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 program = transition system with variables
- Concurrent program with shared variables
- Concurrent program with Rendez-vous
- Concurrent program with FIFO communication
- Petri net
- ...

Models: expressivity versus decidability

Remark: (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 or lossy channels

Remark: 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

Example: Static properties

Mutual exclusion

Safety properties are often static.

They can be reduced to reachability.

Example: Dynamic properties

Every elevator request should be eventually granted.

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

Temporal Structures

Definition: Flows of time

A *flow of time* is a strict order $(\mathbb{T}, <)$ where $\mathbb{T}$ is the nonempty set of *time points* and $<$ is an irreflexive transitive relation on $\mathbb{T}$.

Example: Flows of time

- $(\{0, \ldots, n\}, <)$: Finite runs of sequential systems.
- $(\mathbb{N}, <)$: Infinite runs of sequential systems.
- $(\mathbb{R}, <)$: runs of real-time sequential systems.
- Trees: Finite or infinite run-trees of sequential systems.
- Mazurkiewicz traces: runs of distributed systems (partial orders).
- and also $(\mathbb{Z}, <)$ or $(\mathbb{Q}, <)$ or $(\omega^2, <)$, ...

Definition: Temporal Structures

Let AP be a set of atoms (atomic propositions) and let $\mathcal{C}$ be a class of time flows.

A *temporal structure* over $(\mathcal{C}, \mathrm{AP})$ is a triple $(\mathbb{T}, <, \lambda)$ where $(\mathbb{T}, <)$ is a time flow in $\mathcal{C}$ and $\lambda \colon \mathbb{T} \to 2^{\mathrm{AP}}$ labels time points with atomic propositions.

The temporal structure $(\mathbb{T}, <, \lambda)$ is also denoted $(\mathbb{T}, <, h)$ where $h : \mathrm{AP} \to 2^{\mathbb{T}}$ assigns time points to atomic propositions: $h(p) = \{t \in \mathbb{T} \mid p \in \lambda(t)\}$ for $p \in \mathrm{AP}$.

Linear behaviors and specifications

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

Definition: Runs as temporal structures

An infinite run $\sigma = s_0 s_1 s_2 \cdots$ of M with $(s_i, s_{i+1}) \in T$ for all $i \geq 0$ defines a *linear* temporal structure $\ell(\sigma) = (\mathbb{N}, <, \lambda)$ where $\lambda(i) = \ell(s_i)$ for $i \in \mathbb{N}$.

Such a temporal structure can be seen as an infinite word over $\Sigma = 2^{\mathrm{AP}}$:
$\ell(\sigma) = \ell(s_0)\ell(s_1)\ell(s_2) \cdots \in \Sigma^\omega$

Linear specifications only depend on runs.

Example: The printer manager is fair.

On each run, whenever some process requests the printer, it eventually gets it.

Remark:

Two Kripke structures having the same linear temporal structures satisfy the same linear specifications.

Branching behaviors and specifications

The system has an infinite active run, but it may always reach an inactive state.

Definition: Computation-tree or run-tree : unfolding of the TS

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure. Wlog. $I = \{s_0\}$ is a singleton.

Let D be a finite set with $|D|$ the outdegree of the transition relation T.

The computation-tree of M is an unordered tree $t : D^* \to S$ (partial map) s.t.

- $t(\varepsilon) = s_0$,
- For every node $u \in \mathrm{dom}(t)$ labelled $s = t(u)$, if $T(s) = \{s_1, \ldots, s_k\}$ then u has exactly k children which are labelled s_1,. . . ,s_k

Associated temporal structure $\ell(t) = (\mathrm{dom}(t), <, \lambda)$ where

- $<$ is the strict prefix relation over D^*,
- and $\lambda(u) = \ell(t(u))$ for $u \in \mathrm{dom}(t)$.

(Linear) runs of M are branches of the computation-tree t.

First-order Specifications

Definition: Syntax of $\mathrm{FO}(\mathrm{AP},<)$

Let $\mathrm{Var} = \{x, y, \ldots\}$ be first-order variables.

$$\varphi ::= \bot \mid p(x) \mid x = y \mid x < y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi$$

where $p \in \mathrm{AP}$.

Definition: Semantics of $\mathrm{FO}(\mathrm{AP},<)$

Let $w = (\mathbb{T}, <, \lambda)$ be a temporal structure over AP.
Let $\nu : \mathrm{Var} \to \mathbb{T}$ be an assignment of first-order variables to time points.

$$\begin{array}{lll} w, \nu \models p(x) & \text{if} & p \in \lambda(\nu(x)) \\ w, \nu \models x = y & \text{if} & \nu(x) = \nu(y) \\ w, \nu \models x < y & \text{if} & \nu(x) < \nu(y) \\ w, \nu \models \exists x\, \varphi & \text{if} & w, \nu[x \mapsto t] \models \varphi \text{ for some } t \in \mathbb{T} \end{array}$$

where $\nu[x \mapsto t]$ maps x to t and $y \neq x$ to $\nu(y)$.

Previous specifications can be written in $\mathrm{FO}(<)$ (except the branching one).

First-order vs Temporal

First-order logic

- $\mathrm{FO}(<)$ 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.

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.
- Good expressive power.

Linear Temporal Logic (LTL) over $(\mathbb{N}, <)$ introduced by Pnueli (1977) as a convenient specification language for verification of systems.

Temporal Specifications

Definition: Syntax of $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

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

Definition: Semantics: $w = (\mathbb{T}, <, \lambda)$ temporal structure and $i \in \mathbb{T}$

$w, i \models p$	if	$p \in \lambda(i)$
$w, i \models \neg\varphi$	if	$w, i \not\models \varphi$
$w, i \models \varphi \vee \psi$	if	$w, i \models \varphi$ or $w, i \models \psi$
$w, i \models \varphi \,\mathsf{SU}\, \psi$	if	$\exists k \ i < k$ and $w, k \models \psi$ and $\forall j \ (i < j < k \to w, j \models \varphi)$
$w, i \models \varphi \,\mathsf{SS}\, \psi$	if	$\exists k \ i > k$ and $w, k \models \psi$ and $\forall j \ (i > j > k \to w, j \models \varphi)$

Previous specifications can be written in $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$
(except the branching one).

Temporal Specifications

Definition: non-strict versions of until and since

$$\varphi\,\mathsf{U}\,\psi \overset{\text{def}}{=} \psi \vee (\varphi \wedge \varphi\,\mathsf{SU}\,\psi) \qquad \varphi\,\mathsf{S}\,\psi \overset{\text{def}}{=} \psi \vee (\varphi \wedge \varphi\,\mathsf{SS}\,\psi)$$

$$w,i \models \varphi\,\mathsf{U}\,\psi \quad \text{if} \quad \exists k\ i \leq k \text{ and } w,k \models \psi \text{ and } \forall j\ (i \leq j < k \to w,j \models \varphi)$$

$$w,i \models \varphi\,\mathsf{S}\,\psi \quad \text{if} \quad \exists k\ i \geq k \text{ and } w,k \models \psi \text{ and } \forall j\ (i \geq j > k \to w,j \models \varphi)$$

Definition: Derived modalities

$\mathsf{X}\,\varphi \overset{\text{def}}{=} \bot\,\mathsf{SU}\,\varphi$ Next $\qquad$ $\mathsf{Y}\,\varphi \overset{\text{def}}{=} \bot\,\mathsf{SS}\,\varphi$ Yesterday

$$w,i \models \mathsf{X}\,\varphi \quad \text{if} \quad \exists k\ i < k \text{ and } w,k \models \varphi \text{ and } \neg\exists j\ (i < j < k)$$

$$w,i \models \mathsf{Y}\,\varphi \quad \text{if} \quad \exists k\ i > k \text{ and } w,k \models \varphi \text{ and } \neg\exists j\ (i > j > k)$$

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{SF}\,\varphi & \overset{\text{def}}{=} & \top\,\mathsf{SU}\,\varphi & \mathsf{SP}\,\varphi & \overset{\text{def}}{=} & \top\,\mathsf{SS}\,\varphi \\
\mathsf{F}\,\varphi & \overset{\text{def}}{=} & \top\,\mathsf{U}\,\varphi & \mathsf{P}\,\varphi & \overset{\text{def}}{=} & \top\,\mathsf{S}\,\varphi \\
\mathsf{G}\,\varphi & \overset{\text{def}}{=} & \neg\,\mathsf{F}\,\neg\varphi & \mathsf{H}\,\varphi & \overset{\text{def}}{=} & \neg\,\mathsf{P}\,\neg\varphi
\end{array}$$

$\varphi\,\mathsf{W}\,\psi \overset{\text{def}}{=} (\mathsf{G}\,\varphi) \vee (\varphi\,\mathsf{U}\,\psi)$ Weak Until

$\varphi\,\mathsf{R}\,\psi \overset{\text{def}}{=} (\mathsf{G}\,\psi) \vee (\psi\,\mathsf{U}\,(\varphi \wedge \psi))$ Release

Temporal Specifications

Example: Specifications on the time flow $(\mathbb{N}, <)$

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

Discrete linear time flows

Definition: discrete linear time flows $(\mathbb{T}, <)$

A linear time flow is discrete if $\mathsf{SF}\top \rightarrow \mathsf{X}\top$ and $\mathsf{SP}\top \rightarrow \mathsf{Y}\top$ are valid formulae.

$(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ are discrete.

$(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$ are not discrete.

Exercise: For discrete linear time flows $(\mathbb{T}, <)$

$$
\begin{aligned}
\varphi \,\mathsf{SU}\, \psi &\equiv \mathsf{X}(\varphi \,\mathsf{U}\, \psi) \\
\varphi \,\mathsf{SS}\, \psi &\equiv \mathsf{Y}(\varphi \,\mathsf{S}\, \psi) \\[1ex]
\neg \mathsf{X}\, \varphi &\equiv \neg \mathsf{X} \top \vee \mathsf{X} \neg\varphi \\
\neg \mathsf{Y}\, \varphi &\equiv \neg \mathsf{Y} \top \vee \mathsf{Y} \neg\varphi \\[1ex]
\neg(\varphi \,\mathsf{U}\, \psi) &\equiv (\mathsf{G} \neg\psi) \vee (\neg\psi \,\mathsf{U}\, (\neg\varphi \wedge \neg\psi)) \\
&\equiv \neg\psi \,\mathsf{W}\, (\neg\varphi \wedge \neg\psi) \\
&\equiv \neg\varphi \,\mathsf{R}\, \neg\psi
\end{aligned}
$$

Model checking for linear behaviors

Definition: Model checking problem

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

Question: Does $M \models \varphi$?

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

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

Theorem [11, Sistla, Clarke 85], [10, Lichtenstein & Pnueli 85]

The Model checking problem for LTL is PSPACE-complete. Proof later

Weaknesses of linear behaviors

Example:

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

φ cannot be checked on linear behaviors.

We need to consider the computation-trees.

Weaknesses of FO specifications

Example:

ψ: The system has an infinite active run, but it may always reach an inactive state.

ψ cannot be expressed in FO.

We need quantifications on runs: $\psi = \mathsf{EG}(\mathrm{Active} \wedge \mathsf{EF}\, \neg \mathrm{Active})$

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

MSO Specifications

Definition: Syntax of $\mathrm{MSO}(\mathrm{AP}, <)$

$$\varphi ::= \bot \mid p(x) \mid x = y \mid x < y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

where $p \in \mathrm{AP}$, x, y are first-order variables and X is a second-order variable.

Definition: Semantics of $\mathrm{MSO}(\mathrm{AP}, <)$

Let $w = (\mathbb{T}, <, \lambda)$ be a temporal structure over AP.
An assignment ν maps first-order variables to time points in $\mathbb{T}$
and second-order variables to sets of time points.

The semantics of first-order constructs is unchanged.

$$\begin{array}{lll} w, \nu \models x \in X & \text{if} & \nu(x) \in \nu(X) \\ w, \nu \models \exists X\, \varphi & \text{if} & w, \nu[X \mapsto T] \models \varphi \text{ for some } T \subseteq \mathbb{T} \end{array}$$

where $\nu[X \mapsto T]$ maps X to T and keeps unchanged the other assignments.

MSO vs Temporal

MSO logic

- $\mathrm{MSO}(<)$ has a good expressive power
  ... but $\mathrm{MSO}(<)$-formulae are not easy to write and to understand.
- $\mathrm{MSO}(<)$ is decidable on computation trees
  ... but satisfiability and model checking are non elementary.

We need a temporal logic

- with no explicit variables,
- allowing quantifications over runs,
- usual specifications should be easy to write and read,
- with good complexity for satisfiability and model checking problems,
- with good expressive power.

Computation Tree Logic CTL^* introduced by Emerson & Halpern (1986).

CTL^* (Emerson & Halpern 86)

Definition: Syntax of the Computation Tree Logic $\mathrm{CTL}^*(\mathrm{AP}, \mathsf{SU})$

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

We may also add the past modality SS

Definition: Semantics of $\mathrm{CTL}^*(\mathrm{AP}, \mathsf{SU})$

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure.
Let $\sigma = s_0 s_1 s_2 \cdots$ be an infinte run of M.

$M, \sigma, i \models p$	if	$p \in \ell(s_i)$
$M, \sigma, i \models \varphi\,\mathsf{SU}\,\psi$	if	$\exists k > i,\ M, \sigma, k \models \psi$ and $\forall i < j < k,\ M, \sigma, j \models \varphi$
$M, \sigma, i \models \mathsf{E}\varphi$	if	$M, \sigma', i \models \varphi$ for some infinite run σ' such that $\sigma'[i] = \sigma[i]$
$M, \sigma, i \models \mathsf{A}\varphi$	if	$M, \sigma', i \models \varphi$ for all infinite runs σ' such that $\sigma'[i] = \sigma[i]$

where $\sigma[i] = s_0 \cdots s_i$.

Remark:

- $\sigma'[i] = \sigma[i]$ means that future is branching but past is not.

CTL* (Emerson & Halpern 86)

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: Some equivalences

- $\mathsf{A}\,\varphi \equiv \neg\,\mathsf{E}\,\neg\varphi$
- $\mathsf{E}(\varphi \vee \psi) \equiv \mathsf{E}\,\varphi \vee \mathsf{E}\,\psi$
- $\mathsf{A}(\varphi \wedge \psi) \equiv \mathsf{A}\,\varphi \wedge \mathsf{A}\,\psi$

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 runs σ of M.

Remark: $M \models_\forall \varphi$ 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 $M \models_\exists \varphi$?

Theorem:

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

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

$M, s \models \varphi$ if $M, \sigma, 0 \models \varphi$ for some infinite run σ of M with $\sigma(0) = s$

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

Example: State formulae

Atomic propositions are state formulae: $[\![p]\!] = \{s \in S \mid p \in \ell(s)\}$

State formulae are closed under boolean connectives.

$[\![\neg\varphi]\!] = S \setminus [\![\varphi]\!]$ $\quad$ $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$

Formulae of the form $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ are state formulae, provided φ is future.

Remark: $M \models_\exists \varphi$ iff $I \cap [\![\mathsf{E}\,\varphi]\!] \neq \emptyset$ $\quad$ $M \models_\forall \varphi$ iff $I \subseteq [\![\mathsf{A}\,\varphi]\!]$

Definition: Alternative syntax

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

Path formulae $\quad \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \psi\ \mathsf{SU}\ \psi$

CTL (Clarke & Emerson 81)

Definition: Computation Tree Logic $\mathrm{CTL}(\mathrm{AP}, \mathsf{X}, \mathsf{U})$

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

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

Examples: Macros

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

CTL (Clarke & Emerson 81)

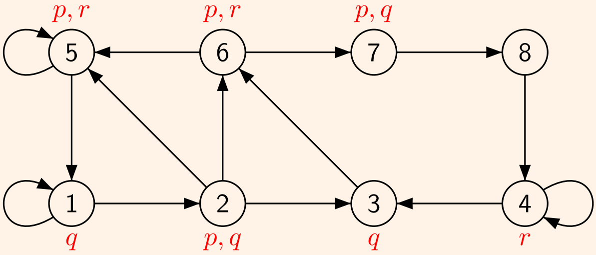
Definition: Semantics

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 $s \in S$.

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

CTL (Clarke & Emerson 81)

Example:



$$[\![\mathsf{EX}\,p]\!] =$$

$$[\![\mathsf{AX}\,p]\!] =$$

$$[\![\mathsf{EF}\,p]\!] =$$

$$[\![\mathsf{AF}\,p]\!] =$$

$$[\![\mathsf{E}\,q\,\mathsf{U}\,r]\!] =$$

$$[\![\mathsf{A}\,q\,\mathsf{U}\,r]\!] =$$

CTL (Clarke & Emerson 81)

Remark: Equivalent formulae

- $\mathsf{AX}\,\varphi \equiv \neg\,\mathsf{EX}\,\neg\varphi$,

- $\neg(\varphi\,\mathsf{U}\,\psi) \equiv \mathsf{G}\,\neg\psi \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$
- $\mathsf{A}\,\varphi\,\mathsf{U}\,\psi \equiv \neg\,\mathsf{EG}\,\neg\psi \wedge \neg\,\mathsf{E}(\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$

- $\mathsf{AG}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) \equiv \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$

- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi \equiv \mathsf{AG}\,\mathsf{AF}\,\varphi$
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi \equiv \mathsf{EF}\,\mathsf{EG}\,\varphi$

- $\mathsf{EG}\,\mathsf{EF}\,\varphi \not\equiv \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi \not\equiv \mathsf{EG}\,\mathsf{AF}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \not\equiv \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi \not\equiv \mathsf{AF}\,\mathsf{EG}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \not\equiv \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi \not\equiv \mathsf{EG}\,\mathsf{AX}\,\varphi$

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 $M \models_\exists \varphi$?

Theorem:

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}$ a formula.
The model checking problem $M \models_\exists \varphi$ is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

References

[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, 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.

[6] Ph. Schnoebelen.
The Complexity of Temporal Logic Model Checking.
In *AiML'02*, 393–436. King's College Publication, 2003.

References

[7] S. Demri and P. Gastin.
Specification and Verification using Temporal Logics.
In Modern applications of automata theory, IISc Research Monographs 2. World Scientific, 2012.
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

[8] D. Gabbay, I. Hodkinson and M. Reynolds.
Temporal logic: mathematical foundations and computational aspects.
Vol 1, Clarendon Press, Oxford, 1994.

[9] 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.

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

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

Outline

Model checking of CTL

Theorem

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}$ a formula.
The model checking problem $M \models_{\exists} \varphi$ is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$

Proof:

Compute $[\![\varphi]\!] = \{s \in S \mid M, s \models \varphi\}$ by induction on the formula.

The set $[\![\varphi]\!]$ is represented by a boolean array: $L[s][\varphi] = \top$ if $s \in [\![\varphi]\!]$.

The labelling ℓ is encoded in L: for $p \in \mathrm{AP}$ we have $L[s][p] = \top$ if $p \in \ell(s)$.

For each $t \in S$, the set $T^{-1}(t)$ is represented as a *list*.

`for all` $t \in S$ `do for all` $s \in T^{-1}(t)$ `do ... od` takes time $\mathcal{O}(|T|)$.

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = \neg\varphi_1$

semantics(φ_1)

$[\![\varphi]\!] := S \setminus [\![\varphi_1]\!]$ $\qquad \mathcal{O}(|S|)$

case $\varphi = \varphi_1 \vee \varphi_2$

semantics(φ_1); semantics(φ_2)

$[\![\varphi]\!] := [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$ $\qquad \mathcal{O}(|S|)$

case $\varphi = EX\varphi_1$

semantics(φ_1)

$[\![\varphi]\!] := \emptyset$ $\qquad \mathcal{O}(|S|)$

for all $t \in [\![\varphi_1]\!]$ do for all $s \in T^{-1}(t)$ do $[\![\varphi]\!] := [\![\varphi]\!] \cup \{s\}$ $\qquad \mathcal{O}(|T|)$

case $\varphi = AX\varphi_1$

semantics(φ_1)

$[\![\varphi]\!] := S$ $\qquad \mathcal{O}(|S|)$

for all $t \notin [\![\varphi_1]\!]$ do for all $s \in T^{-1}(t)$ do $[\![\varphi]\!] := [\![\varphi]\!] \setminus \{s\}$ $\qquad \mathcal{O}(|T|)$

Model checking of CTL

Definition: procedure semantics(φ)

```
case φ = E φ1 U φ2                                                          O(|S| + |T|)
   semantics(φ1); semantics(φ2)
   Todo := ⟦φ2⟧   // the "todo" set Todo is imlemented with a list          O(|S|)
   Good := ⟦φ2⟧   // the "result" is computed in the array Good             O(|S|)
   while Todo ≠ ∅ do                                                        |S| times
   Invariant 1:      ⟦φ2⟧ ∪ Todo ⊆ Good ⊆ ⟦E φ1 U φ2⟧
      take t ∈ Todo; Todo := Todo \ {t}                                     O(1)
      for all s ∈ T^{-1}(t) do                                              |T| times
         if s ∈ ⟦φ1⟧ \ Good then
                  Todo := Todo ∪ {s}; Good := Good ∪ {s}                    O(1)
   od
   ⟦φ⟧ := Good                                                              O(|S|)
```

Good is only used to make the invariant clear. It can be replaced by $[\![\varphi]\!]$.

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = \mathsf{A}\,\varphi_1\,\mathsf{U}\,\varphi_2$ — $\mathcal{O}(|S| + |T|)$

semantics(φ_1); semantics(φ_2)

$\text{Todo} := [\![\varphi_2]\!]$ // the "todo" set Todo is imlemented with a list — $\mathcal{O}(|S|)$

$\text{Good} := [\![\varphi_2]\!]$ // the "result" is computed in the array Good — $\mathcal{O}(|S|)$

for all $s \in S$ do $c[s] := |T(s)|$ — $\mathcal{O}(|S|)$

while $\text{Todo} \neq \emptyset$ do — $|S|$ times

Invariant 1: $[\![\varphi_2]\!] \cup \text{Todo} \subseteq \text{Good} \subseteq [\![\mathsf{A}\,\varphi_1\,\mathsf{U}\,\varphi_2]\!]$

Invariant 2: $\forall s \in S,\ c[s] = |T(s) \setminus (\text{Good} \setminus \text{Todo})|$

take $t \in \text{Todo}$; $\text{Todo} := \text{Todo} \setminus \{t\}$ — $\mathcal{O}(1)$

for all $s \in T^{-1}(t)$ do — $|T|$ times

$c[s] := c[s] - 1$ — $\mathcal{O}(1)$

if $c[s] = 0 \wedge s \in [\![\varphi_1]\!] \setminus \text{Good}$ then

$\text{Todo} := \text{Todo} \cup \{s\}$; $\text{Good} := \text{Good} \cup \{s\}$ — $\mathcal{O}(1)$

od

$[\![\varphi]\!] := \text{Good}$ — $\mathcal{O}(|S|)$

Good is only used to make the invariant clear. It can be replaced by $[\![\varphi]\!]$.

Complexity of CTL

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

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 $\mathsf{A}_f\, \varphi = \mathsf{A}(\mathrm{fair} \to \varphi)$

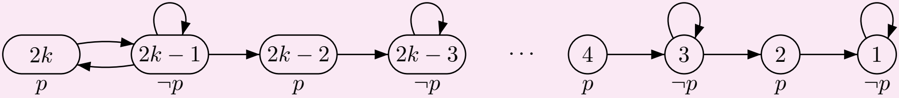
where $\mathrm{fair} = \bigwedge_i \mathsf{G}\, \mathsf{F}\, F_i$

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:



- $M_k, 2k \models \mathsf{E\,G\,F}\,p$ but $M_k, 2k-2 \not\models \mathsf{E\,G\,F}\,p$
- If $\varphi \in \mathrm{CTL}$ and $|\varphi| \leq m \leq k$ 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 $\mathsf{A}_f\,\varphi\,\mathsf{U}\,\psi$.

We have $\mathsf{A}_f\,\varphi\,\mathsf{U}\,\psi = \neg\,\mathsf{E}_f\,\mathsf{G}\,\neg\psi \wedge \neg\,\mathsf{E}_f(\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$

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.

Büchi automata

Definition:

A Büchi automaton (BA) is a tuple $\mathcal{A} = (Q, \Sigma, I, T, F)$ where

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

Run: $\rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \dots$ with $(q_i, a_i, q_{i+1}) \in T$ for all $i \geq 0$.

ρ is initial if $q_0 \in I$.

ρ is final (successful) if $q_i \in F$ for infinitely many i's.

ρ is accepting if it is both initial and final.

$$\mathcal{L}(\mathcal{A}) = \{a_0 a_1 a_2 \cdots \in \Sigma^\omega \mid \exists\, \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \dots \text{ accepting run}\}$$

A language $L \subseteq \Sigma^\omega$ is ω-regular if it can be accepted by some Büchi automaton.

Büchi automata

Examples:

Infinitely many a's:

Finitely many a's:

Whenever a then later b:

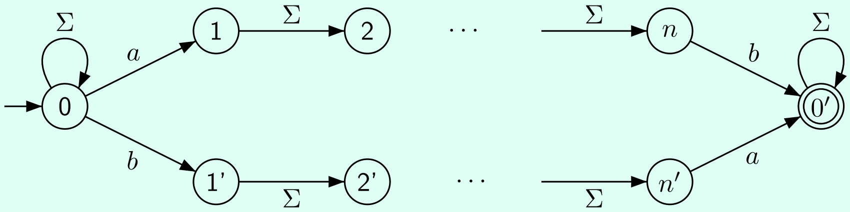
Büchi automata

Properties

Büchi automata are closed under union, intersection, complement.

- Union: trivial
- Intersection: easy (exercise)
- complement: difficult

Let $L = \Sigma^*(a\Sigma^{n-1}b \cup b\Sigma^{n-1}a)\Sigma^\omega$



Any non deterministic Büchi automaton for $\Sigma^\omega \setminus L$ has at least 2^n states.

Büchi automata

Theorem: Büchi

Let $L \subseteq \Sigma^\omega$ be a language. The following are equivalent:

- L is ω-regular
- L is ω-rational, i.e., L is a finite union of languages of the form $L_1 \cdot L_2^\omega$ where $L_1, L_2 \subseteq \Sigma^+$ are rational.
- L is MSO-definable, i.e., there is a sentence $\varphi \in \mathrm{MSO}_\Sigma(<)$ such that $L = \mathcal{L}(\varphi) = \{w \in \Sigma^\omega \mid w \models \varphi\}$.

Exercises:

1. Construct a BA for $\mathcal{L}(\varphi)$ where φ is the $\mathrm{FO}_\Sigma(<)$ sentence

$$(\forall x, (P_a(x) \to \exists y > x, P_a(y))) \to (\forall x, (P_b(x) \to \exists y > x, P_c(y)))$$

2. Given BA for $L_1 \subseteq \Sigma^\omega$ and $L_2 \subseteq \Sigma^\omega$, construct BA for

$$\begin{aligned} \mathrm{next}(L_1) &= \Sigma \cdot L_1 \\ \mathrm{until}(L_1, L_2) &= \{uv \in \Sigma^\omega \mid u \in \Sigma^+ \wedge v \in L_2 \wedge \\ &\qquad u''v \in L_1 \text{ for all } u', u'' \in \Sigma^+ \text{ with } u = u'u''\} \end{aligned}$$

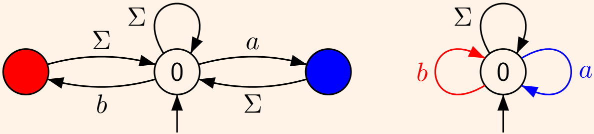
Generalized Büchi automata

Definition: final condition on states or on transitions

$\mathcal{A} = (Q, \Sigma, I, T, F_1, \ldots, F_n)$ with $F_i \subseteq Q$.
An infinite run σ is final (successful) if it visits infinitely often each F_i.

$\mathcal{A} = (Q, \Sigma, I, T, T_1, \ldots, T_n)$ with $T_i \subseteq T$.
An infinite run σ is final if it uses infinitely many transitions from each T_i.

Example: Infinitely many a's and infinitely many b's



Theorem:

1. GBA and BA have the same expressive power.
2. Checking whether a BA or GBA has an accepting run is NLOGSPACE-complete.

Unambiguous or prophetic Büchi automata

Definition: Unambiguous Büchi automata

A BA or GBA $\mathcal{A}$ is unambiguous if every word has at most one accepting run in $\mathcal{A}$.

Definition: Prophetic Büchi automata

A BA or GBA $\mathcal{A}$ is prophetic if every word has exactly one final run in $\mathcal{A}$.

Examples: UBA and PBA

- Finitely many a's.
- $\mathsf{G}(a \to \mathsf{F}\, b)$ with $\Sigma = \{a, b, c\}$.

Proposition: Closure properties

Prophetic BA or GBA are closed under boolean operations (union, intersection, complement).

Theorem: Prophetic Büchi automata (Carton-Michel 2003)

Every ω-regular language can be accepted by a prophetic BA.

Büchi automata with output

Definition: SBT: Synchronous (letter to letter) Büchi transducer

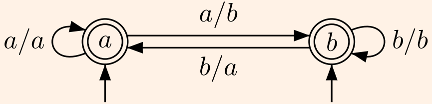
Let A and B be two alphabets.
A synchronous Büchi transducer from A to B is a tuple $\mathcal{A} = (Q, A, I, T, F, \mu)$ where (Q, A, I, T, F) is a Büchi automaton (input) and $\mu : T \to B$ is the output function.
It computes the relation

$$\begin{aligned}[\![\mathcal{A}]\!] = \{(u, v) \in A^\omega \times B^\omega \mid\ & \exists\, \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots \text{ accepting run} \\ & \text{with } u = a_0 a_1 a_2 \cdots \text{ and } v = \mu(\rho), \\ & \text{i.e., } v = b_0 b_1 b_2 \cdots \text{ with } b_i = \mu(q_i, a_i, q_{i+1}) \text{ for } i \geq 0\}\end{aligned}$$

If (Q, A, I, T, F) is unambiguous then $[\![\mathcal{A}]\!] : A^\omega \to B^\omega$ is a (partial) function, in which case we also write $[\![\mathcal{A}]\!](u) = v$ for $(u, v) \in [\![\mathcal{A}]\!]$.

We will also use SGBT: synchronous transducers with generalized Büchi acceptance.

Example: Left shift with $A = B = \{a, b\}$



Composition of Büchi transducers

Definition: Composition

Let A, B, C be alphabets.
Let $\mathcal{A} = (Q, A, I, T, (F_i)_i, \mu)$ be an SGBT from A to B.
Let $\mathcal{A}' = (Q', B, I', T', (F'_j)_j, \mu')$ be an SGBT from B to C.
Then $\mathcal{A} \cdot \mathcal{A}' = (Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j, \mu'')$ defined by:

$$\tau'' = (p, p') \xrightarrow{a} (q, q') \in T'' \text{ and } \mu''(\tau'') = c$$

iff

$$\tau = p \xrightarrow{a} q \in T \text{ and } \tau' = p' \xrightarrow{\mu(\tau)} q' \in T' \text{ and } c = \mu'(\tau')$$

is an SGBT from A to C.
When the transducers define functions, we also denote the composition by $\mathcal{A}' \circ \mathcal{A}$.

Proposition: Composition

1. We have $[\![\mathcal{A} \cdot \mathcal{A}']\!] = [\![\mathcal{A}]\!] \cdot [\![\mathcal{A}']\!]$.
2. If $(Q, A, I, T, (F_i)_i)$ and $(Q', B, I', T', (F'_j)_j)$ are unambiguous (resp. prophetic) then $(Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j)$ is also unambiguous (resp. prophetic), and
   $\forall u \in A^\omega$ we have $[\![\mathcal{A}' \circ \mathcal{A}]\!](u) = [\![\mathcal{A}']\!]([\![\mathcal{A}]\!](u))$.

Product of Büchi transducers

Definition: Product

Let A, B, C be alphabets.
Let $\mathcal{A} = (Q, A, I, T, (F_i)_i, \mu)$ be an SGBT from A to B.
Let $\mathcal{A}' = (Q', A, I', T', (F'_j)_j, \mu')$ be an SGBT from A to C.
Then $\mathcal{A} \times \mathcal{A}' = (Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j, \mu'')$ defined by:

$$\tau'' = (p, p') \xrightarrow{a} (q, q') \in T'' \text{ and } \mu''(\tau'') = (b, c)$$

iff

$$\tau = p \xrightarrow{a} q \in T \text{ and } b = \mu(\tau) \text{ and } \tau' = p' \xrightarrow{a} q' \in T' \text{ and } c = \mu'(\tau')$$

is an SGBT from A to $B \times C$.

Proposition: Product

We identify $(B \times C)^\omega$ with $B^\omega \times C^\omega$.

1. We have $[\![\mathcal{A} \times \mathcal{A}']\!] = \{(u, v, v') \mid (u, v) \in [\![\mathcal{A}]\!] \text{ and } (u, v') \in [\![\mathcal{A}']\!]\}$.
2. If $(Q, A, I, T, (F_i)_i)$ and $(Q', A, I', T', (F'_j)_j)$ are unambiguous (resp. prophetic) then $(Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j)$ is also unambiguous (resp. prophetic), and
$\forall u \in A^\omega$ we have $[\![\mathcal{A} \times \mathcal{A}']\!](u) = ([\![\mathcal{A}]\!](u), [\![\mathcal{A}']\!](u))$.

Subalphabets of $\Sigma = 2^{\mathrm{AP}}$

Definition:

For a propositional formula ξ over AP, we let $\Sigma_\xi = \{a \in \Sigma \mid a \models \xi\}$.
For instance, for $p, q \in \mathrm{AP}$,

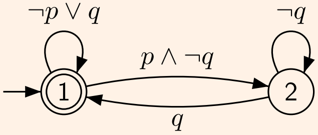
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ $\ldots$

Notation:

In automata, $s \xrightarrow{\Sigma_\xi} s'$ stands for the set of transitions $\{s\} \times \Sigma_\xi \times \{s'\}$.

To simplify the pictures, we use $s \xrightarrow{\xi} s'$ instead of $s \xrightarrow{\Sigma_\xi} s'$.

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



Semantics of LTL with sequential functions

Definition: Semantics of $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

Let $\Sigma = 2^{\mathrm{AP}}$ and $\mathbb{B} = \{0, 1\}$.

Define $[\![\varphi]\!] : \Sigma^\omega \to \mathbb{B}^\omega$ by $[\![\varphi]\!](u) = b_0 b_1 b_2 \cdots$ with $b_i = \begin{cases} 1 & \text{if } u, i \models \varphi \\ 0 & \text{otherwise.} \end{cases}$

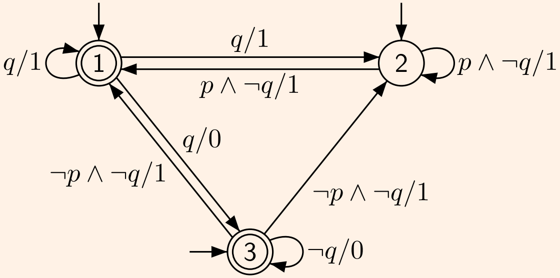
Example:

$$[\![p \,\mathsf{SU}\, q]\!](\emptyset\{q\}\{p\}\emptyset\{p\}\{p\}\{q\}\emptyset\{p\}\{p,q\}\emptyset^\omega) = 1001110110^\omega$$
$$[\![\mathsf{X}\, p]\!](\emptyset\{q\}\{p\}\emptyset\{p\}\{p\}\{q\}\emptyset\{p\}\{p,q\}\emptyset^\omega) = 0101100110^\omega$$
$$[\![\mathsf{F}\, p]\!](\emptyset\{q\}\{p\}\emptyset\{p\}\{p\}\{q\}\emptyset\{p\}\{p,q\}\emptyset^\omega) = 1111111110^\omega$$

The aim is to compute $[\![\varphi]\!]$ with synchronous Büchi transducers (actually, SGBT).

Synchronous Büchi transducer for $p \,\mathsf{SU}\, q$

Example: An SBT for $[\![p \,\mathsf{SU}\, q]\!]$



Lemma: The input BA is prophetic

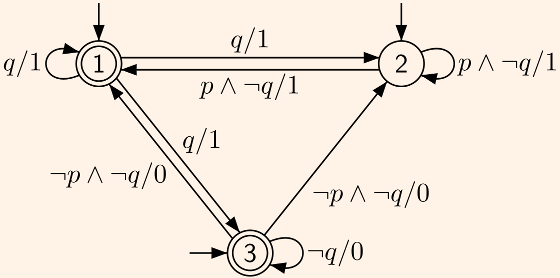
For all $u = a_0 a_1 a_2 \cdots \in \Sigma^\omega$,
there is a unique final run $\rho = s_0, a_0, s_1, a_1, s_2, a_2, s_3, \ldots$ of $\mathcal{A}$ on u.

The run ρ satisfies for all $i \geq 0$, $s_i = \begin{cases} 1 & \text{if } u, i \models q \\ 2 & \text{if } u, i \models \neg q \wedge (p \,\mathsf{U}\, q) \\ 3 & \text{if } u, i \models \neg(p \,\mathsf{U}\, q) \end{cases}$

Hence, the SBT computes $[\![p \,\mathsf{SU}\, q]\!]$.

Synchronous Büchi transducer for $p\,\mathsf{U}\,q$

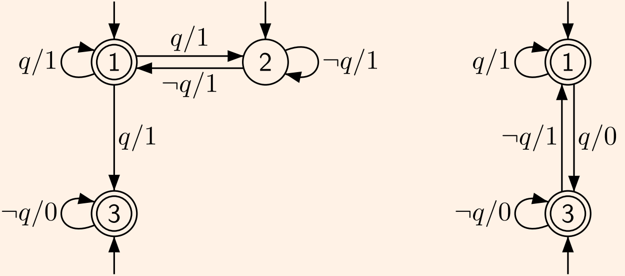
Example: An SBT for $[\![p\,\mathsf{U}\,q]\!]$



The automaton is prophetic (same input BA as for $p\,\mathsf{SU}\,q$).
This SBT computes $[\![p\,\mathsf{U}\,q]\!]$.

Special cases of Until: Future and Next

Example: $\mathsf{F}\,q = \top\,\mathsf{U}\,q$ and $\mathsf{X}\,q = \bot\,\mathsf{SU}\,q$

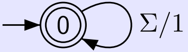


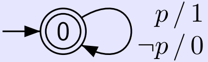
Exercise: Give SBT's for the following formulae:

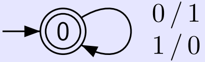
$\mathsf{SF}\,q$, $\mathsf{SG}\,q$, $p\,\mathsf{SR}\,q$, $p\,\mathsf{SS}\,q$, $\mathsf{Y}\,q$, $\mathsf{G}\,q$, $p\,\mathsf{R}\,q$, $p\,\mathsf{S}\,q$, $\mathsf{G}(p \to \mathsf{F}\,q)$.

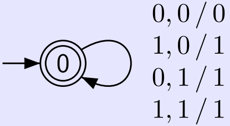
From LTL to Büchi automata

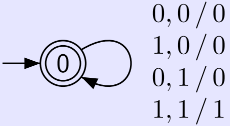
Definition: SBT for LTL modalities

- $\mathcal{A}_\top$ from Σ to $\mathbb{B} = \{0,1\}$: 

- $\mathcal{A}_p$ from Σ to $\mathbb{B} = \{0,1\}$: 

- $\mathcal{A}_\neg$ from $\mathbb{B}$ to $\mathbb{B}$: 

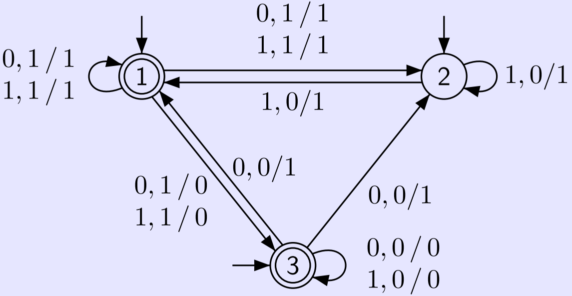
- $\mathcal{A}_\vee$ from $\mathbb{B}^2$ to $\mathbb{B}$: 

- $\mathcal{A}_\wedge$ from $\mathbb{B}^2$ to $\mathbb{B}$: 

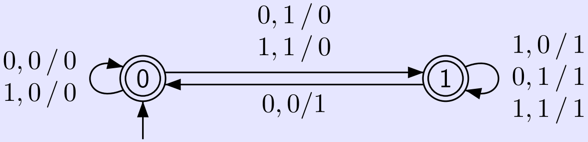
From LTL to Büchi automata

Definition: SBT for LTL modalities (cont.)

- $\mathcal{A}_{\mathsf{SU}}$ from $\mathbb{B}^2$ to $\mathbb{B}$: Prophetic



- $\mathcal{A}_{\mathsf{SS}}$ from $\mathbb{B}^2$ to $\mathbb{B}$: Deterministic Not prophetic



From LTL to Büchi automata

Definition: Translation from LTL to SGBT

For each $\xi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ we define inductively an SGBT $\mathcal{A}_\xi$ as follows:

- $\mathcal{A}_\top$ and $\mathcal{A}_p$ for $p \in \mathrm{AP}$ are already defined
- $\mathcal{A}_{\neg\varphi} = \mathcal{A}_\neg \circ \mathcal{A}_\varphi$
- $\mathcal{A}_{\varphi\vee\psi} = \mathcal{A}_\vee \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$
- $\mathcal{A}_{\varphi \mathsf{SS} \psi} = \mathcal{A}_{\mathsf{SS}} \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$
- $\mathcal{A}_{\varphi \mathsf{SU} \psi} = \mathcal{A}_{\mathsf{SU}} \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$

Theorem: Correctness of the translation

For each $\xi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$, we have $[\![\mathcal{A}_\xi]\!] = [\![\xi]\!]$ and $\mathcal{A}_\xi$ is unambiguous.

Moreover, the number of states of $\mathcal{A}_\xi$ is at most $2^{|\xi|_{\mathsf{SS}}} \cdot 3^{|\xi|_{\mathsf{SU}}}$

the number of acceptance conditions is $|\xi|_{\mathsf{SU}}$

where $|\xi|_{\mathsf{SS}}$ (resp. $|\xi|_{\mathsf{SU}}$) is the number of SS (resp. SU) occurring in ξ.

Remark:

- If a subformula φ occurs serveral time in ξ, we only need one copy of $\mathcal{A}_\varphi$.
- We may also use automata for other modalities: $\mathcal{A}_{\mathsf{X}}$ (2 states), $\mathcal{A}_{\mathsf{U}}$, ...

Useful simplifications

Reducing the number of temporal subformulae

$$(\mathsf{X}\,\varphi) \wedge (\mathsf{X}\,\psi) \equiv \mathsf{X}(\varphi \wedge \psi) \qquad (\mathsf{X}\,\varphi)\,\mathsf{SU}\,(\mathsf{X}\,\psi) \equiv \mathsf{X}(\varphi\,\mathsf{SU}\,\psi)$$

$$(\mathsf{G}\,\varphi) \wedge (\mathsf{G}\,\psi) \equiv \mathsf{G}(\varphi \wedge \psi) \qquad \mathsf{G}\,\mathsf{F}\,\varphi \vee \mathsf{G}\,\mathsf{F}\,\psi \equiv \mathsf{G}\,\mathsf{F}(\varphi \vee \psi)$$

$$(\varphi_1\,\mathsf{SU}\,\psi) \wedge (\varphi_2\,\mathsf{SU}\,\psi) \equiv (\varphi_1 \wedge \varphi_2)\,\mathsf{SU}\,\psi \qquad (\varphi\,\mathsf{SU}\,\psi_1) \vee (\varphi\,\mathsf{SU}\,\psi_2) \equiv \varphi\,\mathsf{SU}\,(\psi_1 \vee \psi_2)$$

Merging equivalent states

Let $\mathcal{A} = (Q, \Sigma, I, T, (F_i)_i, \mu)$ be an SGBT and $s_1, s_2 \in Q$.

We can merge s_1 and s_2 if they satisfy the same final conditions:

$$s_1 \in F_i \iff s_2 \in F_i \qquad \text{for all } i$$

and they have the same outgoing transitions: $\forall a \in \Sigma$, $\forall s \in Q$,

$$\tau_1 = (s_1, a, s) \in T \iff \tau_2 = (s_2, a, s) \in T \qquad \text{and} \qquad \mu(\tau_1) = \mu(\tau_2)$$

Other constructions

- Tableau construction. See for instance [16, Wolper 85]
  $+$: Easy definition, easy proof of correctness
  $+$: Works both for future and past modalities
  $-$: Inefficient without strong optimizations
- Using Very Weak Alternating Automata [17, Gastin & Oddoux 01].
  $+$: Very efficient
  $-$: Only for future modalities
  Online tool: `http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/`
- Using reduction rules [7, Demri & Gastin 10].
  $+$: Efficient and produces small automata
  $+$: Can be used by hand on real examples
  $-$: Only for future modalities
- The domain is still very active.

Some References

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

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

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

[17] 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

[7] S. Demri and P. Gastin.
Specification and Verification using Temporal Logics.
In Modern applications of automata theory, IISc Research Monographs 2.
World Scientific, 2012.
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

Satisfiability for LTL over $(\mathbb{N}, <)$

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

Definition: Satisfiability problem

Input: A formula $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

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 $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

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

Remark: φ is satisfiable iff $\mathsf{F}\,\varphi$ is *initially* satisfiable.

Definition: (Initial) validity

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

Theorem [11, Sistla, Clarke 85], [10, Lichtenstein & Pnueli 85]

The satisfiability problem for LTL is PSPACE-complete.

Model checking for LTL

Definition: Model checking problem

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

Question: Does $M \models \varphi$?

- Universal MC: $M \models_{\forall} \varphi$ if $\ell(\sigma), 0 \models \varphi$ for all initial infinite runs 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 [11, Sistla, Clarke 85], [10, Lichtenstein & Pnueli 85]

The Model checking problem for LTL is PSPACE-complete

$\mathrm{MC}^\exists(\mathsf{SU}) \leq_P \mathrm{SAT}(\mathsf{SU})$
[11, Sistla & Clarke 85]

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

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 $\psi_M \in \mathrm{LTL}(\mathrm{AP}', \mathsf{X}, \mathsf{F})$ of size $\mathcal{O}(|M|^2)$ by

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

Let $w = a_0 a_1 a_2 \cdots \in \Sigma'^\omega$. Then, $w \models \psi_M$ iff there exists an initial infinite run $\sigma = s_0, s_1, s_2, \ldots$ of M such that $\pi(w) = \ell(\sigma)$ and $a_i \cap \mathrm{AP}_S = \{\mathrm{at}_{s_i}\}$ for all $i \geq 0$.

Therefore, $M \models_\exists \varphi$ iff $\psi_M \wedge \varphi$ is initially satisfiable
$M \models_\forall \varphi$ iff $\psi_M \wedge \neg\varphi$ is not initially satisfiable

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

QBF **Quantified Boolean Formulae**

Definition: QBF

Input: A formula $\gamma = Q_1 x_1 \cdots Q_n x_n \gamma'$ with $\gamma' = \bigwedge_{1 \le i \le m} \bigvee_{1 \le j \le k_i} a_{ij}$ (CNF)

$Q_i \in \{\forall, \exists\}$ 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 $v = v_1 \cdots v_n \in \{0,1\}^n$.
We write $v[i]$ for the prefix of length i.
Let $V \subseteq \{0,1\}^n$ 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 $v'_i = 1 - v_i$.

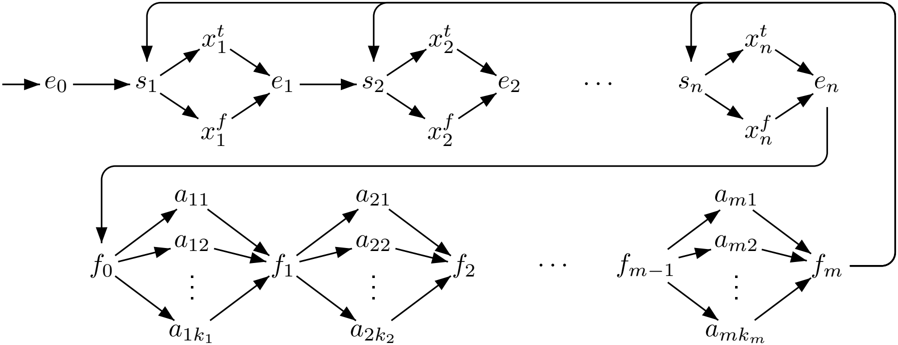
Proposition:

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

$\mathrm{QBF} \leq_P \mathrm{MC}^{\exists}(\mathsf{U})$ [11, 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 $\varphi_i = \mathsf{G}(e_{i-1} \to (\neg s_{i-1} \mathrel{\mathsf{U}} x_i^t) \wedge (\neg s_{i-1} \mathrel{\mathsf{U}} x_i^f))$ and $\varphi = \bigwedge_{i \mid Q_i = \forall} \varphi_i$.

Then, γ is valid iff $M \models_{\exists} \psi \wedge \varphi$.

Complexity of LTL

Theorem: Complexity of LTL

The following problems are PSPACE-complete:

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

The following problems are NP-complete:

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

Complexity of CTL^*

Definition: Syntax of the Computation Tree Logic CTL^*

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

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.

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

Outline

Expressivity

Definition: Equivalence

Let $\mathcal{C}$ be a class of time flows.

Two formulae $\varphi, \psi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ are equivalent over $\mathcal{C}$ if
for all temporal structures $w = (\mathbb{T}, <, \lambda)$ over $\mathcal{C}$ and all time points $t \in \mathbb{T}$ we have

$$w, t \models \varphi \quad \text{iff} \quad w, t \models \psi$$

Two formulae $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ and $\psi(x) \in \mathrm{FO}_{\mathrm{AP}}(<)$ are equivalent over $\mathcal{C}$ if
for all temporal structures $w = (\mathbb{T}, <, \lambda)$ over $\mathcal{C}$ and all time points $t \in \mathbb{T}$ we have

$$w, t \models \varphi \quad \text{iff} \quad w, x \mapsto t \models \psi$$

We also write $w \models \psi(t)$.

Remark: $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS}) \subseteq \mathrm{FO}^3_{\mathrm{AP}}(<) \subseteq \mathrm{FO}_{\mathrm{AP}}(<)$

$\forall \varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$, $\exists \psi(x) \in \mathrm{FO}^3_{\mathrm{AP}}(<)$ such that φ and $\psi(x)$ are equivalent.

Expressivity problem: $\mathrm{LTL} = \mathrm{FO}?$

Expressivity

Definition: complete linear time flows

A time flow $(\mathbb{T}, <)$ is linear if $<$ is a total strict order.

A linear time flow $(\mathbb{T}, <)$ is complete if every nonempty and bounded subset of $\mathbb{T}$ has a least upper bound and a greatest lower bound.

$(\mathbb{N}, <)$, $(\mathbb{Z}, <)$ and $(\mathbb{R}, <)$ are complete.
$(\mathbb{Q}, <)$ and $(\mathbb{R} \setminus \{0\}, <)$ are not complete.

Theorem: Expressive completeness [12, Kamp 68]

For complete linear time flows, $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS}) = \mathrm{FO}_{\mathrm{AP}}(<)$

Elegant algebraic proof of $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}) = \mathrm{FO}_{\mathrm{AP}}(<)$ over $(\mathbb{N}, <)$ due to Wilke 98.

See also Diekert-Gastin [18]: $\mathrm{TL} = \mathrm{FO} = \mathrm{SF} = \mathrm{AP} = \mathrm{CFBA} = \mathrm{VWAA}$.

Example: Translate in $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ (1)

$$\psi(x) = \neg P_a(x) \wedge \neg P_b(x) \wedge \forall y \forall z \, (P_a(y) \wedge P_b(z) \wedge y < z) \rightarrow$$
$$\exists v \; y < v < z \wedge \begin{pmatrix} & P_c(v) \wedge x < y \\ \vee & P_d(v) \wedge z < x \\ \vee & P_e(v) \wedge y < x < z \end{pmatrix}$$

Stavi connectives: Time flows with gaps

Definition: Stavi Until: $\overline{\mathsf{U}}$

Let $w = (\mathbb{T}, <, \lambda)$ be a temporal structure and $i \in \mathbb{T}$. Then, $w, i \models \varphi \,\overline{\mathsf{U}}\, \psi$ if

$$
\begin{aligned}
&\exists k \;\, i < k \\
&\;\wedge \exists j \,(i < j < k \wedge w, j \models \neg\varphi) \\
&\;\wedge \exists j \,(i < j < k \wedge \forall \ell \,(i < \ell < j \to w, \ell \models \varphi)) \\
&\;\wedge \forall j \left[i < j < k \to \left[\begin{array}{l} \quad \exists k' \,[j < k' \wedge \forall j' \,(i < j' < k' \to w, j' \models \varphi)] \\ \vee \,[\forall \ell \,(j < \ell < k \to w, \ell \models \psi) \wedge \exists \ell \,(i < \ell < j \wedge w, \ell \models \neg\varphi)] \end{array} \right] \right]
\end{aligned}
$$

Similar definition for the Stavi Since $\overline{\mathsf{S}}$.

Example: (2)

- Let $w = (\mathbb{R} \setminus \{0\}, <, h)$ with $h(p) = \mathbb{R}_-$ and $h(q) = \mathbb{R}_+$.
  Then, $w, -1 \not\models p \,\mathsf{SU}\, q$ but $w, -1 \models p \,\overline{\mathsf{U}}\, q$.
- Let $w' = (\mathbb{R} \setminus \{0\}, <, h')$ with $h'(p) = \mathbb{R} \setminus \{1, \frac{1}{2}, \frac{1}{4}, \ldots, 0\}$ and $h'(q) = \mathbb{R}_+$.
  Then, $w', -1 \models p \,\overline{\mathsf{U}}\, q$.

Stavi connectives: Time flows with gaps

Theorem: [14, Gabbay, Hodkinson, Reynolds]

$\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS}, \overline{\mathsf{S}}, \overline{\mathrm{U}})$ is expressively complete for $\mathrm{FO}_{\mathrm{AP}}(<)$ over the class of all linear time flows.

Exercise: Isolated gaps (3)

Let $\varphi_p = p\,\mathsf{SU}\,p \wedge \mathsf{SF}\,\neg p \wedge \neg(p\,\mathsf{SU}\,\neg p) \wedge \neg(p\,\mathsf{SU}\,\neg(p\,\mathsf{SU}\,\top))$.

Let $w = (\mathbb{T}, <, \lambda)$ with $\mathbb{T} \subseteq \mathbb{R}$ and $t \in \mathbb{T}$.

Show that if $w, t \models \varphi_p$ then $\mathbb{T}$ has a gap.

Let $\psi_{p,q} = \varphi_p \wedge (q \vee \varphi_p)\,\mathsf{SU}\,(q \wedge \neg p)$.

Show that $\psi_{p,q}$ is equivalent to $p\,\overline{\mathrm{U}}\,q$ over the time flow $(\mathbb{R} \setminus \{0\}, <)$.

Show that $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ is $\mathrm{FO}_{\mathrm{AP}}(<)$-complete over the time flow $(\mathbb{R} \setminus \mathbb{Z}, <)$.

Temporal depth

Definition: Temporal depth of $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

$$\begin{aligned}
\mathrm{td}(p) &= 0 && \text{if } p \in \mathrm{AP} \\
\mathrm{td}(\neg\varphi) &= \mathrm{td}(\varphi) \\
\mathrm{td}(\varphi \vee \psi) &= \max(\mathrm{td}(\varphi), \mathrm{td}(\psi)) \\
\mathrm{td}(\varphi \,\mathsf{SS}\, \psi) &= \max(\mathrm{td}(\varphi), \mathrm{td}(\psi)) + 1 \\
\mathrm{td}(\varphi \,\mathsf{SU}\, \psi) &= \max(\mathrm{td}(\varphi), \mathrm{td}(\psi)) + 1
\end{aligned}$$

Lemma:

Let $B \subseteq \mathrm{AP}$ be finite and $k \in \mathbb{N}$.
There are (up to equivalence) finitely many formulae in $\mathrm{TL}(B, \mathsf{SU}, \mathsf{SS})$ of temporal depth at most k.

k-equivalence

Definition:

Let $w_0 = (\mathbb{T}_0, <, h_0)$ and $w_1 = (\mathbb{T}_1, <, h_1)$ be two temporal structures.
Let $i_0 \in \mathbb{T}_0$ and $i_1 \in \mathbb{T}_1$. Let $k \in \mathbb{N}$.

We say that (w_0, i_0) and (w_1, i_1) are k-equivalent, denoted $(w_0, i_0) \equiv_k (w_1, i_1)$, if they satisfy the same formulae in $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ of temporal depth at most k.

Lemma: $\equiv_k$ is an equivalence relation of finite index.

Example:

Let $a = \{p\}$ and $b = \{q\}$. Let $w_0 = babaababaa$ and $w_1 = baababaaba$.

$$(w_0, 3) \equiv_0 (w_1, 4)$$
$$(w_0, 3) \equiv_1 (w_1, 4) \ ?$$
$$(w_0, 3) \equiv_1 (w_1, 6) \ ?$$

Here, $\mathbb{T}_0 = \mathbb{T}_1 = \{0, 1, 2, \ldots, 9\}$.

EF-games for $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

The EF-game has two players: Spoiler (Player I) and Duplicator (Player II).

The game board consists of 2 temporal structures:
$w_0 = (\mathbb{T}_0, <, h_0)$ and $w_1 = (\mathbb{T}_1, <, h_1)$.

There are two tokens, one on each structure: $i_0 \in \mathbb{T}_0$ and $i_1 \in \mathbb{T}_1$.

A configuration is a tuple (w_0, i_0, w_1, i_1)
or simply (i_0, i_1) if the game board is understood.

Let $k \in \mathbb{N}$.

The k-round EF-game from a configuration proceeds with (at most) k moves.

There are 2 available moves for $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$: SU-move or SS-move (see below).

Spoiler chooses which move is played in each round.

Spoiler wins if

- Either duplicator cannot answer during a move (see below).
- Or a configuration such that $(w_0, i_0) \not\equiv_0 (w_1, i_1)$ is reached.

Otherwise, duplicator wins.

Strict Until and Since moves

Definition: SU-move

- Spoiler chooses $\varepsilon \in \{0,1\}$ and $k_\varepsilon \in \mathbb{T}_\varepsilon$ such that $i_\varepsilon < k_\varepsilon$.
- Duplicator chooses $k_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ such that $i_{1-\varepsilon} < k_{1-\varepsilon}$.
  Spoiler wins if there is no such $k_{1-\varepsilon}$.
  Either spoiler chooses (k_0, k_1) as next configuration of the EF-game,
  or the move continues as follows
- Spoiler chooses $j_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ with $i_{1-\varepsilon} < j_{1-\varepsilon} < k_{1-\varepsilon}$.
- Duplicator chooses $j_\varepsilon \in \mathbb{T}_\varepsilon$ with $i_\varepsilon < j_\varepsilon < k_\varepsilon$.
  Spoiler wins if there is no such j_ε.
  The next configuration is (j_0, j_1).

Similar definition for the SS-move.

Winning strategy

Definition: Winning strategy

Duplicator has a winning strategy in the k-round EF-game starting from (w_0, i_0, w_1, i_1) if he can win all plays starting from this configuration.
This is denoted by $(w_0, i_0) \sim_k (w_1, i_1)$.

Spoiler has a winning strategy in the k-round EF-game starting from (w_0, i_0, w_1, i_1) if she can win all plays starting from this configuration.

Example: (4)

Let $a = \{p\}$, $b = \{q\}$, $c = \{r\}$. Let $w_0 = aaabbc$ and $w_1 = aababc$.

$$(w_0, 0) \sim_1 (w_1, 0)$$
$$(w_0, 0) \not\sim_2 (w_1, 0)$$

Here, $\mathbb{T}_0 = \mathbb{T}_1 = \{0, 1, 2, \ldots, 5\}$.

EF-games for $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

Lemma: Determinacy

The k-round EF-game for $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ is determined:
For each initial configuration, either spoiler or duplicator has a winning strategy.

Theorem: Soundness and completeness of EF-games

For all $k \in \mathbb{N}$ and all configurations (w_0, i_0, w_1, i_1), we have

$$(w_0, i_0) \sim_k (w_1, i_1) \text{ iff } (w_0, i_0) \equiv_k (w_1, i_1)$$

Example:

Let $a = \{p\}$, $b = \{q\}$, $c = \{r\}$.
Then, $aaaabbc, 0 \models p \,\mathsf{SU}\, (q \,\mathsf{SU}\, r)$ but $aaababc, 0 \not\models p \,\mathsf{SU}\, (q \,\mathsf{SU}\, r)$.
$p \,\mathsf{SU}\, (q \,\mathsf{SU}\, r)$ cannot be expressed with a formula of temporal depth at most 1.
$p \,\mathsf{SU}\, (q \wedge \mathsf{X}\, q)$ cannot be expressed with a formula of temporal depth at most 1.

Exercise:

On finite linear time flows, "even length" cannot be expressed in $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$.

Moves for Strict Future and Past modalities

Definition: SF-move

- Spoiler chooses $\varepsilon \in \{0,1\}$ and $j_\varepsilon \in \mathbb{T}_\varepsilon$ such that $i_\varepsilon < j_\varepsilon$.
- Duplicator chooses $j_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ such that $i_{1-\varepsilon} < j_{1-\varepsilon}$.
  Spoiler wins if there is no such $j_{1-\varepsilon}$.
  The new configuration is (j_0, j_1).

Similar definition for the SP-move.

Example:

$p \,\mathsf{SU}\, q$ is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{SP}, \mathsf{SF})$ over linear flows of time.

Let $a = \emptyset$, $b = \{p\}$ and $c = \{q\}$.

Let $w_0 = (abc)^n a (abc)^n$ and $w_1 = (abc)^n (abc)^n$.

If $n > k$ then, starting from $(w_0, 3n, w_1, 3n)$, duplicator has a winning strategy in the k-round EF-game using SF-moves and SP-moves.

Moves for Next and Yesterday modalities

Notation: $i \lessdot j \stackrel{\text{def}}{=} i < j \wedge \neg \exists k \, (i < k < j)$.

Definition: X-move

- Spoiler chooses $\varepsilon \in \{0, 1\}$ and $j_\varepsilon \in \mathbb{T}_\varepsilon$ such that $i_\varepsilon \lessdot j_\varepsilon$.
- Duplicator chooses $j_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ such that $i_{1-\varepsilon} \lessdot j_{1-\varepsilon}$.
  Spoiler wins if there is no such $j_{1-\varepsilon}$.
  The new configuration is (j_0, j_1).

Similar definition for the Y-move.

Exercise:

Show that $p \, \mathsf{SU} \, q$ is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{Y}, \mathsf{SP}, \mathsf{X}, \mathsf{SF})$ over linear time flows.

Non-strict Until and Since moves

Definition: U-move

- Spoiler chooses $\varepsilon \in \{0,1\}$ and $k_\varepsilon \in \mathbb{T}_\varepsilon$ such that $i_\varepsilon \leq k_\varepsilon$.
- Duplicator chooses $k_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ such that $i_{1-\varepsilon} \leq k_{1-\varepsilon}$.
  Either spoiler chooses (k_0, k_1) as new configuration of the EF-game,
  or the move continues as follows
- Spoiler chooses $j_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ with $i_{1-\varepsilon} \leq j_{1-\varepsilon} < k_{1-\varepsilon}$.
- Duplicator chooses $j_\varepsilon \in \mathbb{T}_\varepsilon$ with $i_\varepsilon \leq j_\varepsilon < k_\varepsilon$.
  Spoiler wins if there is no such j_ε.
  The new configuration is (j_0, j_1).

- If duplicator chooses $k_{1-\varepsilon} = i_{1-\varepsilon}$ then the new configuration must be (k_0, k_1).
- If spoiler chooses $k_\varepsilon = i_\varepsilon$ then duplicator must choose $k_{1-\varepsilon} = i_{1-\varepsilon}$, otherwise he loses.

Similar definition for the S-move.

Exercise:

1. Show that SU is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ over $(\mathbb{R}, <)$.
2. Show that SU is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ over $(\mathbb{N}, <)$.

Syntactic Separation

Definition: Syntactically pure formulae and separation

A formula $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ is

- syntactically pure present if it is a boolean combination of formulae in AP,
- syntactically pure future if it is a boolean combination of formulae of the form $\alpha \,\mathsf{SU}\, \beta$ where $\alpha, \beta \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU})$,
- syntactically pure past if it is a boolean combination of formulae of the form $\alpha \,\mathsf{SS}\, \beta$ where $\alpha, \beta \in \mathrm{TL}(\mathrm{AP}, \mathsf{SS})$.
- syntactically separated if it is a boolean combination of syntactically pure formulae.

A logic $\mathcal{L}$ is syntactically separable over a class $\mathcal{C}$ of time flows if each formula $\varphi \in \mathcal{L}$ is equivalent to some (finite) boolean combination of syntactically pure formulae.

Example: (5)

The formulae $\varphi_1 = \mathsf{SF}(q \wedge \mathsf{SP}\, p)$ and $\varphi_2 = \mathsf{SF}(q \wedge \neg \mathsf{SP} \neg p)$ are not separated but we can find equivalent syntactically separated formulae.

Separation

Theorem: [9, Gabbay, Pnueli, Shelah & Stavi 80]

$\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ is syntactically separable over discrete and complete linear orders.

Definition: Discrete linear order

A linear time flow $(\mathbb{T}, <)$ is discrete if every non-maximal element has an immediate successor and every non-minimal element has an immediate predecessor.

- $(\mathbb{N}, <)$ is the unique (up to isomorphism) discrete and complete linear order with a first point and no last point.
- $(\mathbb{Z}, <)$ is the unique (up to isomorphism) discrete and complete linear order with no first point and no last point.
- Any discrete and complete linear order is isomorphic to a sub-flow of $(\mathbb{Z}, <)$.

Theorem: Gabbay, Reynolds, see [8]

$\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ is syntactically separable over $(\mathbb{R}, <)$.

Semantic Separation

Definition:

Let $w = (\mathbb{T}, <, \lambda)$ and $w' = (\mathbb{T}, <, \lambda')$ be temporal structures over the same time flow, and let $t \in \mathbb{T}$ be a time point.

- w, w' agree on t if $\lambda(t) = \lambda'(t)$
- w, w' agree on the past of t if $\lambda(s) = \lambda'(s)$ for all $s < t$
- w, w' agree on the future of t if $\lambda(s) = \lambda'(s)$ for all $s > t$

Definition: Pure formulae

Let $\mathcal{C}$ be a class of time flows. A formula φ over some logic $\mathcal{L}$ is pure past (resp. pure present, pure future) over $\mathcal{C}$ if

$$w, t \models \varphi \quad \text{iff} \quad w', t \models \varphi$$

for all temporal structures $w = (\mathbb{T}, <, \lambda)$ and $w' = (\mathbb{T}, <, \lambda')$ over $\mathcal{C}$ and all time points $t \in \mathbb{T}$ such that

w, w' agree on the past of t (resp. on t, on the future of t).

Separation

Remark: Syntax versus semantic

Every formula $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ which is syntactically pure present (resp. future, past) is also semantically pure present (resp. future, past).

Definition: Separation

A logic $\mathcal{L}$ is separable over a class $\mathcal{C}$ of time flows if each formula $\varphi \in \mathcal{L}$ is equivalent to some (finite) boolean combination of pure formulae.

Theorem: [13, Gabbay 89] (already stated by Gabbay in 81)

Let $\mathcal{C}$ be a class of linear time flows.

Let $\mathcal{L}$ be a temporal logic able to express SF and SP.

Then, $\mathcal{L}$ is separable over $\mathcal{C}$ iff it is expressively complete for $\mathrm{FO}_{\mathrm{AP}}(<)$ over $\mathcal{C}$.

Exercise: Checking semantically pure

Is the following problem decidable? If yes, what is his complexity?

Input: A formula $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

Question: Is the formula φ *semantically* pure future?

Initial equivalence

Definition: Initial Equivalence

Let $\mathcal{C}$ be a class of time flows having a least element (denoted 0).
Two formulae $\varphi, \psi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ are initially equivalent over $\mathcal{C}$ if for all temporal structures $w = (\mathbb{T}, <, \lambda)$ over $\mathcal{C}$ we have

$$w, 0 \models \varphi \quad \text{iff} \quad w, 0 \models \psi$$

Two formulae $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ and $\psi(x) \in \mathrm{FO}_{\mathrm{AP}}(<)$ are initially equivalent over $\mathcal{C}$ if for all temporal structures $w = (\mathbb{T}, <, \lambda)$ over $\mathcal{C}$ we have

$$w, 0 \models \varphi \quad \text{iff} \quad w \models \psi(0)$$

Corollary: of the separation theorem

For each $\varphi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ there exists $\psi \in \mathrm{TL}(\mathrm{AP}, \mathsf{SU})$ such that φ and ψ are initially equivalent over $(\mathbb{N}, <)$.

Initial equivalence

Example: $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ versus $\mathrm{TL}(\mathrm{AP}, \mathsf{SU})$

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

is initially equivalent to

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

Theorem: (Laroussinie & Markey & Schnoebelen 2002)

$\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$ may be exponentially more succinct than $\mathrm{TL}(\mathrm{AP}, \mathsf{SU})$ over $(\mathbb{N}, <)$.

Some References

[12] J. Kamp.
Tense Logic and the Theory of Linear Order.
PhD thesis, UCLA, USA, (1968).

[9] 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.

[13] 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.

[14] D. Gabbay, I. Hodkinson and M. Reynolds.
Temporal expressive completeness in the presence of gaps.
In *Logic Colloquium '90*, Springer Lecture Notes in Logic 2, pp. 89-121, 1993.

[15] I. Hodkinson and M. Reynolds.
Separation — Past, Present and Future.
In "We Will Show Them: Essays in Honour of Dov Gabbay".
Vol 2, pages 117–142, College Publications, 2005.
Great survey on separation properties.

Some References

[8] D. Gabbay, I. Hodkinson and M. Reynolds.
Temporal logic: mathematical foundations and computational aspects.
Vol 1, Clarendon Press, Oxford, 1994.

[18] V. Diekert and P. Gastin.
First-order definable languages.
In *Logic and Automata: History and Perspectives*, vol. 2, *Texts in Logic and Games*, pp. 261–306. Amsterdam University Press, (2008).
Overview of formalisms expressively equivalent to First-Order for words.
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

[19] H. Straubing.
Finite automata, formal logic, and circuit complexity.
In *Progress in Theoretical Computer Science*, Birkhäuser, (1994).

[20] K. Etessami and Th. Wilke.
An until hierarchy and other applications of an Ehrenfeucht-Fraïssé game for temporal logic.
In *Information and Computation*, vol. 106, pp. 88–108, (2000).