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
2011 – 2012

Outline

1 Introduction

Models

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

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!
- Overbar missing in the mathematical specification:
  $\bar{\dot{R}}_n$: 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

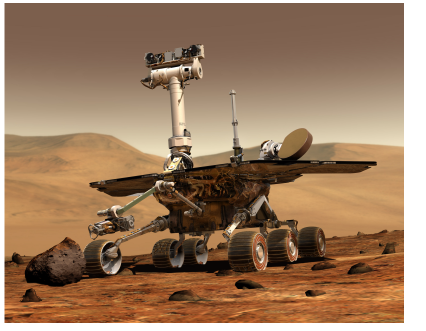
- Destroyed 37 seconds after launch (cost: 370 millions dollars).
- data conversion from a 64-bit floating point to 16-bit signed integer value caused a hardware exception (arithmetic overflow).
- Efficiency considerations had led to the disabling of the software handler (in Ada code) for this error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

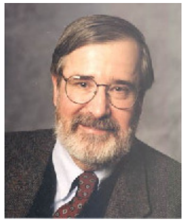
Formal verifications methods

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

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, 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

Introduction

2 Models
- Transition systems
- . . . with variables
- Concurrent systems
- Synchronization and communication

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

Outline

Introduction

2 Models
- Transition systems
- ... with variables
- Concurrent systems
- Synchronization and communication

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

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 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.

Outline

Introduction

2 Models

Transition systems

• ... with variables

Concurrent systems

Synchronization and communication

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

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 $x > 0$.

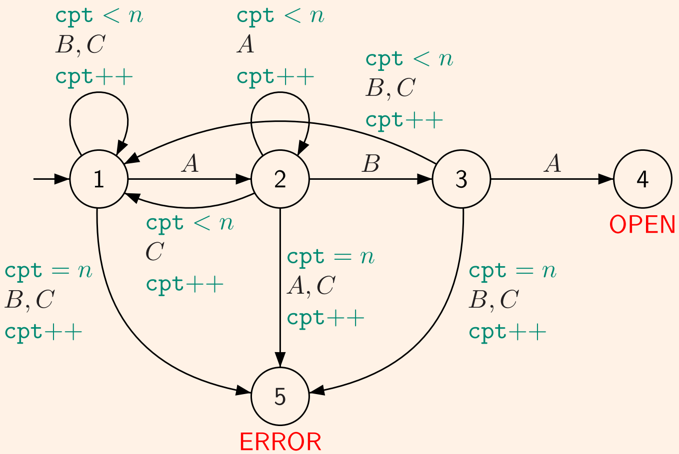
Programs = Kripke structures with variables

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

Example: GCD

TS with variables ...

Example: Digicode



Only variables

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

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

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

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

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

Outline

Introduction

2 Models
Transition systems
... with variables
Concurrent systems
Synchronization and communication

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

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

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

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
  $S = \prod_i S_i$, $\Sigma = \prod_i(\Sigma_i \cup \{\varepsilon\})$, and $I = \prod_i I_i$
  $T = \{(p_1, \ldots, p_n) \xrightarrow{(a_1,\ldots,a_n)} (q_1, \ldots, q_n) \mid$ 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: 2 special cases

- Synchronous: $\Sigma_{\mathrm{sync}} = \prod_i \Sigma_i$
- Asynchronous: $\Sigma_{\mathrm{sync}} = \biguplus_i \Sigma_i'$ 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: $T_{\mathrm{sync}} \subseteq T$

Outline

Introduction

2 Models

Transition systems

... with variables

Concurrent systems

- Synchronization and communication

Specifications

Satisfiability and Model Checking for LTL

Branching Time Specifications

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

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

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:

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

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

Atomicity

Definition: Atomic statements: atomic(ES)

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

$$ES ::= \text{skip} \mid \text{await } c \mid x := e \mid ES\, ;\, ES \mid ES \,\square\, ES$$
$$\mid \text{when } c \text{ do } ES \mid \text{if } c \text{ then } ES \text{ else } ES$$

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 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

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

Introduction

Models

3 Specifications
- Definitions
- Expressivity
- Separation
- Ehrenfeucht-Fraïssé games

Satisfiability and Model Checking for LTL

Branching Time Specifications

Some References

[7] 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.

[8] 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.

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

[16] 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

Outline

Introduction

Models

3 Specifications
- Definitions
- Expressivity
- Separation
- Ehrenfeucht-Fraïssé games

Satisfiability and Model Checking for LTL

Branching Time Specifications

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.
- 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 $(\mathbb{R}, <)$, $(\omega^2, <)$, ...

Definition: Temporal Structures

Let AP be a set of atoms (atomic propositions).

A *temporal structure* over a class $\mathcal{C}$ of time flows and AP is a triple $(\mathbb{T}, <, h)$ where $(\mathbb{T}, <)$ is a time flow in $\mathcal{C}$ and $h : \mathrm{AP} \to 2^{\mathbb{T}}$ is an assignment.

If $p \in \mathrm{AP}$ then $h(p) \subseteq \mathbb{T}$ gives the time points where p holds.

Linear behaviors and specifications

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

Definition: Runs as temporal structures

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

Such a temporal structure can be seen as an infinite word over $\Sigma = 2^{\mathrm{AP}}$:
$\ell(\sigma) = \ell(s_0)\ell(s_1)\ell(s_2)\cdots = (\mathbb{N}, <, w)$ with $w(i) = \ell(s_i) \in \Sigma$.

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

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

Definition: Run-trees as temporal strucutres

Run-tree = unfolding of the transition system.

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

Unordered tree $t : D^* \to \Sigma$ (partial map).

Associated temporal structure $(\mathrm{dom}(t), <, h)$ where
$<$ is the strict prefix relation over D^* and $h(p) = \{i \in \mathrm{dom}(t) \mid p \in t(i)\}$.

Example: Each process has the possibility to print first.

First-order Specifications

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

Let $P, Q, \ldots$ be unary predicates twinned with atoms $p, q, \ldots$ in AP.
Let $\mathrm{Var} = \{x, y, \ldots\}$ be first-order variables.

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

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

Let $w = (\mathbb{T}, <, h)$ be a temporal structure.
Precidates $P, Q, \ldots$ twinned with $p, q, \ldots$ are interpreded as $h(p), h(q), \ldots$
Let $\nu : \mathrm{Var} \to \mathbb{T}$ be an assignment of first-order variables.

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

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

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

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 $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$

$\varphi ::= \bot \mid p\ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{F}\,\varphi \mid \mathsf{P}\,\varphi \mid \mathsf{G}\,\varphi \mid \mathsf{H}\,\varphi \mid \varphi\,\mathsf{U}\,\varphi \mid \varphi\,\mathsf{S}\,\varphi \mid \mathsf{X}\,\varphi \mid \mathsf{Y}\,\varphi$

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

$$
\begin{array}{lll}
w, i \models p & \text{if} & i \in h(p) \\
\\
w, i \models \mathsf{F}\,\varphi & \text{if} & \exists j\ i < j \text{ and } w, j \models \varphi \\
w, i \models \mathsf{G}\,\varphi & \text{if} & \forall j\ i < j \rightarrow w, j \models \varphi \\
w, i \models \varphi\,\mathsf{U}\,\psi & \text{if} & \exists k\ i < k \text{ and } w, k \models \psi \text{ and } \forall j\ (i < j < k) \rightarrow w, j \models \varphi \\
w, i \models \mathsf{X}\,\varphi & \text{if} & \exists j\ i < j \text{ and } w, j \models \varphi \text{ and } \neg\exists k\ (i < k < j) \\
\\
w, i \models \mathsf{P}\,\varphi & \text{if} & \exists j\ i > j \text{ and } w, j \models \varphi \\
w, i \models \mathsf{H}\,\varphi & \text{if} & \forall j\ i > j \rightarrow w, j \models \varphi \\
w, i \models \varphi\,\mathsf{S}\,\psi & \text{if} & \exists k\ i > k \text{ and } w, k \models \psi \text{ and } \forall j\ (i > j > k) \rightarrow w, j \models \varphi \\
w, i \models \mathsf{Y}\,\varphi & \text{if} & \exists j\ i > j \text{ and } w, j \models \varphi \text{ and } \neg\exists k\ (i > k > j)
\end{array}
$$

Previous specifications can be written in $\mathrm{TL}(\mathrm{AP})$.

Temporal Specifications

Relations between modalities

$$
\begin{array}{rcl}
\mathsf{F}\,\varphi & = & \top\,\mathsf{U}\,\varphi \\
\mathsf{G}\,\varphi & = & \neg\,\mathsf{F}\,\neg\varphi \\
\mathsf{X}\,\varphi & = & \bot\,\mathsf{U}\,\varphi
\end{array}
$$

Definition: Derived modalities

$$
\begin{array}{rcll}
\varphi\,\mathsf{W}\,\psi & \stackrel{\text{def}}{=} & (\mathsf{G}\,\varphi) \vee (\varphi\,\mathsf{U}\,\psi) & \text{Weak Until} \\
\varphi\,\mathsf{R}\,\psi & \stackrel{\text{def}}{=} & (\mathsf{G}\,\psi) \vee (\psi\,\mathsf{U}\,(\varphi \wedge \psi)) & \text{Release}
\end{array}
$$

Definition: non-strict versions of modalities

$$
\begin{array}{rcl}
\mathsf{F}'\,\varphi & \stackrel{\text{def}}{=} & \varphi \vee \mathsf{F}\,\varphi \\
\mathsf{G}'\,\varphi & \stackrel{\text{def}}{=} & \varphi \wedge \mathsf{G}\,\varphi \\
\varphi\,\mathsf{U}'\,\psi & \stackrel{\text{def}}{=} & \psi \vee (\varphi \wedge \varphi\,\mathsf{U}\,\psi) \\
\varphi\,\mathsf{R}'\,\psi & \stackrel{\text{def}}{=} & \psi \wedge (\varphi \vee \varphi\,\mathsf{R}\,\psi)
\end{array}
$$

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: $\mathsf{G}'(\mathrm{request} \rightarrow \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}'(\mathrm{request} \rightarrow (\neg\mathrm{request}\,\mathsf{U}\,\mathrm{grant}))$
- Release: $\mathrm{reset}\,\mathsf{R}\,\mathrm{alarm}$
- Strong fairness: $\mathsf{G}\,\mathsf{F}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$
- Weak fairness: $\mathsf{F}\,\mathsf{G}\,\mathrm{request} \rightarrow \mathsf{G}\,\mathsf{F}\,\mathrm{grant}$

Outline

Introduction

Models

3 Specifications
- Definitions
- Expressivity
- Separation
- Ehrenfeucht-Fraïssé games

Satisfiability and Model Checking for LTL

Branching Time Specifications

Some References

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

[7] 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.

[8] 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.

[9] 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.

[17] 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

Temporal Specifications

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

$$\begin{aligned}
\mathsf{F}\,\varphi &= \mathsf{X}\,\mathsf{F}'\,\varphi \\
\mathsf{G}\,\varphi &= \mathsf{X}\,\mathsf{G}'\,\varphi \\
\varphi\,\mathsf{U}\,\psi &= \mathsf{X}(\varphi\,\mathsf{U}'\,\psi) \\
\neg\,\mathsf{X}\,\varphi &= \mathsf{X}\,\neg\varphi \vee \neg\,\mathsf{X}\,\top \\
\neg(\varphi\,\mathsf{U}\,\psi) &= (\mathsf{G}\,\neg\psi) \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi\wedge\neg\psi)) \\
&= \neg\psi\,\mathsf{W}\,(\neg\varphi\wedge\neg\psi) \\
&= \neg\varphi\,\mathsf{R}\,\neg\psi
\end{aligned}$$

Definition: discrete linear time flows

A linear time flow $(\mathbb{T},<)$ is discrete if $\mathsf{F}\,\top \to \mathsf{X}\,\top$ and $\mathsf{P}\,\top \to \mathsf{Y}\,\top$ are valid formulae.

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

Expressivity

Definition: Equivalence

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

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

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

Remark: $\mathrm{LTL}(\mathrm{AP},\mathsf{S},\mathsf{U}) \subseteq \mathrm{FO}_{\mathrm{AP}}(<)$

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

Expressivity

Theorem: Expressive completeness [6, Kamp 68]

For complete linear time flows,

$$\mathrm{TL}(\mathrm{AP},\mathsf{S},\mathsf{U}) = \mathrm{FO}_{\mathrm{AP}}(<)$$

Definition: complete linear time flows

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.

Remark:

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

Stavi connectives: Time flows with gaps

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

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

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

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

Theorem: [9, Gabbay, Hodkinson, Reynolds]

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

Exercise: Isolated gaps

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

Outline

Introduction

Models

3 Specifications
- Definitions
- Expressivity
- Separation
- Ehrenfeucht-Fraïssé games

Satisfiability and Model Checking for LTL

Branching Time Specifications

Some References

[7] 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.

[8] 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.

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

[11] 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.

Separation

Definition:

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

- w, w' agree on t if $h(t) = h'(t)$
- w, w' agree on the past of t if $h(s) = h'(s)$ for all $s < t$
- w, w' agree on the future of t if $h(s) = h'(s)$ 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 for all temporal structures $w = (\mathbb{T}, <, h)$ and $w' = (\mathbb{T}, <, h')$ 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) we have

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

Separation

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: [7, Gabbay, Pnueli, Shelah & Stavi 80]

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

- $(\mathbb{N}, <)$ 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 [10]

$\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ is separable over $(\mathbb{R}, <)$.

Separation and Expressivity

Theorem: [8, 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 F and P.

Then, $\mathcal{L}$ is separable over $\mathcal{C}$ iff it is expressively complete over $\mathcal{C}$.

Initial equivalence

Definition: Initial Equivalence

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

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

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

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

Corollary: of the separation theorem

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

Initial equivalence

Example: $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ versus $\mathrm{TL}(\mathrm{AP}, \mathsf{U})$

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

is initially equivalent to

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

Theorem: (Laroussinie & Markey & Schnoebelen 2002)

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

Outline

Introduction

Models

3 Specifications
- Definitions
- Expressivity
- Separation
- Ehrenfeucht-Fraïssé games

Satisfiability and Model Checking for LTL

Branching Time Specifications

Some References

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

[19] 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).

Temporal depth

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

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

Lemma:

Let $B \subseteq \mathrm{AP}$ be finite and $k \in \mathbb{N}$.
There are (up to equivalence) finitely many formulae in $\mathrm{TL}(B, \mathsf{S}, \mathsf{U})$ 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 $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ 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 $w_1 = baababaaba$.

$$
\begin{aligned}
(w_0, 3) &\equiv_0 (w_1, 4) \\
(w_0, 3) &\equiv_1 (w_1, 4) \ ? \\
(w_0, 3) &\equiv_1 (w_1, 6) \ ?
\end{aligned}
$$

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

EF-games for $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$

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 $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$: Until or Since (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.

Until and Since moves

Definition: (Strict) Until 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 (strict) Since 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:

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 $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$

Lemma: Determinacy

The k-round EF-game for $\mathrm{TL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ 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, $aaabbc, 0 \models p \,\mathsf{U}\, (q \,\mathsf{U}\, r)$ but $aababc, 0 \not\models p \,\mathsf{U}\, (q \,\mathsf{U}\, r)$.

Hence, $p \,\mathsf{U}\, (q \,\mathsf{U}\, r)$ cannot be expressed with a formula of temporal depth at most 1.

Exercise:

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

Moves for Future and Past modalities

Definition: (Strict) Future 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 (strict) Past move.

Example:

$p \, \mathsf{U} \, q$ is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{P}, \mathsf{F})$ 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 Future and Past moves.

Moves for Next and Yesterday modalities

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

Definition: Next 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 Yesterday move.

Exercise:

Show that $p \, \mathsf{U} \, q$ is not expressible in $\mathrm{TL}(\mathrm{AP}, \mathsf{Y}, \mathsf{P}, \mathsf{X}, \mathsf{F})$ over linear flows of time.

Non-strict Until and Since moves

Definition: non-strict Until move

- Spoiler chooses $\varepsilon \in \{0,1\}$ and $k_\varepsilon \in \mathbb{T}_\varepsilon$ such that $i_\varepsilon \le k_\varepsilon$.
- Duplicator chooses $k_{1-\varepsilon} \in \mathbb{T}_{1-\varepsilon}$ such that $i_{1-\varepsilon} \le k_{1-\varepsilon}$.
  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 $i_{1-\varepsilon} \le j_{1-\varepsilon} < k_{1-\varepsilon}$.
- Duplicator chooses $j_\varepsilon \in \mathbb{T}_\varepsilon$ with $i_\varepsilon \le j_\varepsilon < k_\varepsilon$.
  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 non-strict Since move.

Exercise:

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

Outline

Introduction

Models

Specifications

4 Satisfiability and Model Checking for LTL
Büchi automata
From LTL to BA
Decidability and Complexity

Branching Time Specifications

Some References

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

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

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

[15] 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`

[16] 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`

Outline

Introduction

Models

Specifications

4 Satisfiability and Model Checking for LTL
- Büchi automata
- From LTL to BA
- Decidability and Complexity

Branching Time Specifications

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 accepting (repeated, final) 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 accepting if $q_0 \in I$ and $q_i \in F$ for infinitely many i's.

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

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:

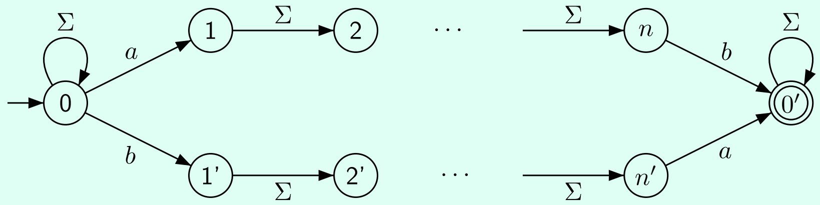
Büchi automata

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 $\varphi \in \mathrm{MSO}_\Sigma(\leq)_\Sigma(<)$ 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

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

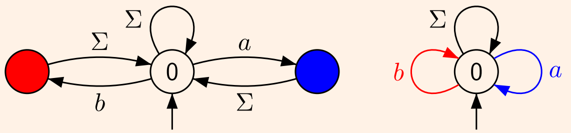
Generalized Büchi automata

Definition: acceptance 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 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 successful 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.

Büchi automata with output

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

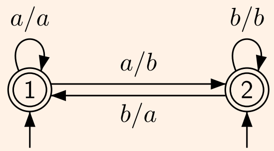
Let A and B be two alphabets.
A synchronous Büchi transducer from A to B is a tuple $\mathcal{A} = (Q, A, I, T, F, \mu)$ where (Q, A, I, T, F) is a Büchi automaton (input) and $\mu : T \to B$ is the output function.
It computes the relation

$$\begin{aligned} [\![\mathcal{A}]\!] = \{(u, v) \in A^\omega \times B^\omega \mid \exists\, \rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots & \text{ accepting run} \\ & \text{with } u = a_0 a_1 a_2 \cdots \\ & \text{and } v = \mu(q_0, a_0, q_1)\mu(q_1, a_1, q_2)\mu(q_2, a_2, q_3) \cdots\} \end{aligned}$$

If (Q, A, I, T, F) is unambiguous then $[\![\mathcal{A}]\!] : A^\omega \to B^\omega$ is a (partial) function.

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 $\mathcal{A} \cdot \mathcal{A}' = (Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j, \mu'')$ is defined by:

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

iff

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

$\mathcal{A} \cdot \mathcal{A}'$ 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 then $(Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j)$ is also unambiguous.
   Then, $\forall u \in A^\omega$ 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 $\mathcal{A} \times \mathcal{A}' = (Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j, \mu'')$ is defined by:

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

iff

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

$\mathcal{A} \times \mathcal{A}'$ 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 then $(Q \times Q', A, I \times I', T'', (F_i \times Q')_i, (Q \times F'_j)_j)$ is also unambiguous.
   Then, $\forall u \in A^\omega$ we have $[\![\mathcal{A} \times \mathcal{A}']\!](u) = ([\![\mathcal{A}]\!](u), [\![\mathcal{A}']\!](u))$.

Outline

Introduction

Models

Specifications

4 Satisfiability and Model Checking for LTL
- Büchi automata
- From LTL to BA
- Decidability and Complexity

Branching Time Specifications

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 $p, q \in \mathrm{AP}$,

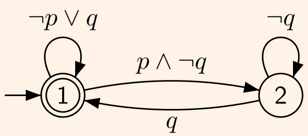
- $\Sigma_p = \{a \in \Sigma \mid p \in a\}$ and $\Sigma_{\neg p} = \Sigma \setminus \Sigma_p$
- $\Sigma_{p \wedge q} = \Sigma_p \cap \Sigma_q$ and $\Sigma_{p \vee q} = \Sigma_p \cup \Sigma_q$
- $\Sigma_{p \wedge \neg q} = \Sigma_p \setminus \Sigma_q$ ...

Notation:

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

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

Example:



Semantics of LTL with sequential functions

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

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 $b_i = \begin{cases} 1 & \text{if } u, i \models \varphi \\ 0 & \text{otherwise.} \end{cases}$

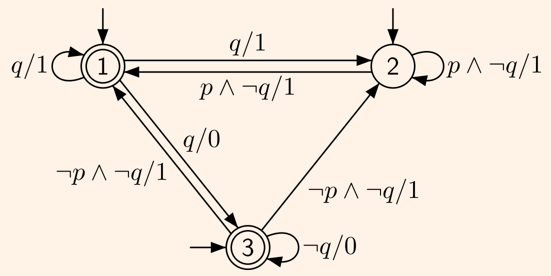
Example:

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

The aim is to compute $[\![\varphi]\!]$ with Büchi transducers.

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

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



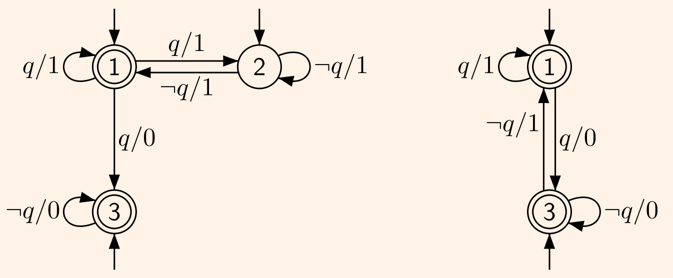
Lemma: The input BA is prophetic

For all $u = a_0 a_1 a_2 \cdots \in \Sigma^\omega$,
there is a unique accepting run $\rho = q_0, a_0, q_1, a_1, q_2, a_2, q_3, \ldots$ of $\mathcal{A}$ on u.

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

Special cases of Until: Future and Next

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

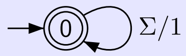


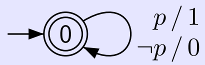
Exercise: Give SBT's for the following formulae:

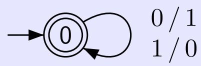
$p \,\mathsf{U}'\, q$, $\mathsf{F}'\, q$, $\mathsf{G}\, q$, $\mathsf{G}'\, q$, $p \,\mathsf{R}\, q$, $p \,\mathsf{R}'\, q$, $p \,\mathsf{S}\, q$, $p \,\mathsf{S}'\, q$, $\mathsf{G}(p \to \mathsf{F}\, q)$.

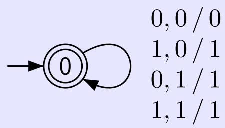
From LTL to Büchi automata

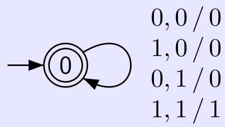
Definition: SBT for LTL modalities

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

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

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

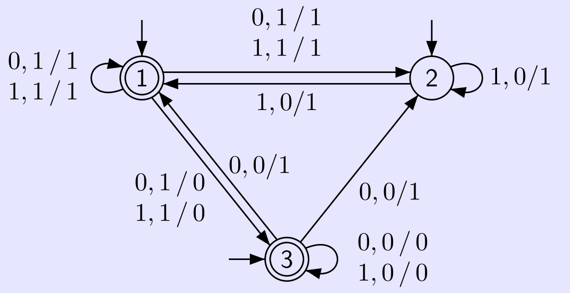
- $\mathcal{A}_\vee$ from $\mathbb{B}^2$ to $\mathbb{B}$: 

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

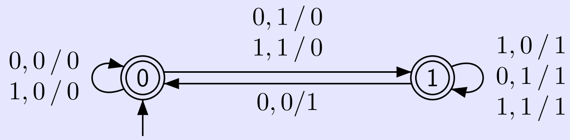
From LTL to Büchi automata

Definition: SBT for LTL modalities (cont.)

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



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



From LTL to Büchi automata

Definition: Translation from LTL to SGBT

For each $\xi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$ 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
- $\mathcal{A}_{\neg\varphi} = \mathcal{A}_\neg \circ \mathcal{A}_\varphi$
- $\mathcal{A}_{\varphi\vee\psi} = \mathcal{A}_\vee \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$
- $\mathcal{A}_{\varphi\mathsf{S}\psi} = \mathcal{A}_{\mathsf{S}} \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$
- $\mathcal{A}_{\varphi\mathsf{U}\psi} = \mathcal{A}_{\mathsf{U}} \circ (\mathcal{A}_\varphi \times \mathcal{A}_\psi)$

Theorem: Correctness of the translation

For each $\xi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$, we have $[\![\mathcal{A}_\xi]\!] = [\![\xi]\!]$.

Moreover, the number of states of $\mathcal{A}_\xi$ is at most $2^{|\xi|_{\mathsf{S}}} \cdot 3^{|\xi|_{\mathsf{U}}}$
where $|\xi|_{\mathsf{S}}$ (resp. $|\xi|_{\mathsf{U}}$) is the number of S (resp. U) 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: $\mathcal{A}_{\mathsf{X}}$, $\mathcal{A}_{\mathsf{U}'}$, ...

Useful simplifications

Reducing the number of temporal subformulae

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

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

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

Merging equivalent states

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

$$(s_1, a, s) \in T \iff (s_2, a, s) \in T$$

and $\quad (s_1, a, s) \in T_i \iff (s_2, a, s) \in T_i \quad$ for all $1 \leq i \leq n$.

Other constructions

- Tableau construction. See for instance [13, Wolper 85]
  - $+$: Easy definition, easy proof of correctness
  - $+$: Works both for future and past modalities
  - $-$: Inefficient without strong optimizations
- Using Very Weak Alternating Automata [15, Gastin & Oddoux 01].
  - $+$: Very efficient
  - $-$: Only for future modalities

  Online tool: `http://www.lsv.ens-cachan.fr/~gastin/ltl2ba/`
- Using reduction rules [16, 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.

Outline

Introduction

Models

Specifications

4 Satisfiability and Model Checking for LTL
- Büchi automata
- From LTL to BA
- Decidability and Complexity

Branching Time Specifications

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

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

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 [14, Sistla, Clarke 85], [12, 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 $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{S}, \mathsf{U})$

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 [14, Sistla, Clarke 85], [12, Lichtenstein & Pnueli 85]

The Model checking problem for LTL is PSPACE-complete

$\mathrm{MC}^\exists(\mathsf{U}) \leq_P \mathrm{SAT}(\mathsf{U})$ [14, Sistla & Clarke 85]

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

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 $\psi_M \in \mathrm{LTL}(\mathrm{AP}', \mathsf{X}, \mathsf{F}')$ 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 σ 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 satisfiable
$M \models_\forall \varphi$ iff $\psi_M \wedge \neg\varphi$ is not satisfiable

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

QBF Quantified Boolean Formulae

Definition: QBF

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

$Q_i \in \{\forall, \exists\}$ 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 $\{v_i, v'_i\} = \{0,1\}$.

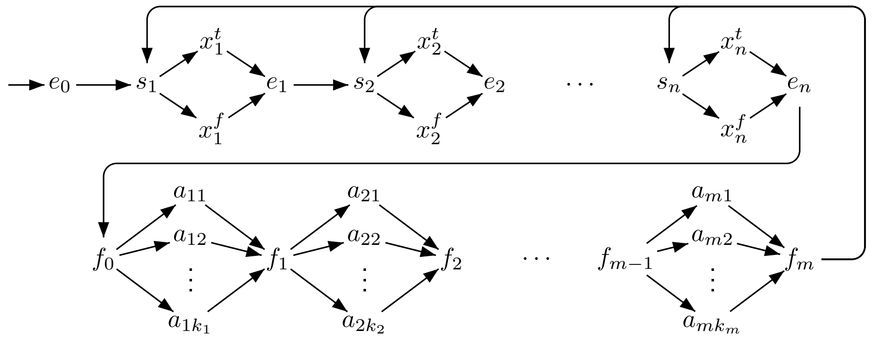
Proposition:

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

$\mathrm{QBF} \le_P \mathrm{MC}^{\exists}(\mathrm{U}')$ [14, Sistla & Clarke 85]

Let $\gamma = Q_1 x_1 \cdots Q_n x_n \bigwedge_{1 \le i \le m} \bigvee_{1 \le j \le k_i} a_{ij}$ with $Q_i \in \{\forall, \exists\}$ and a_{ij} literals.

Consider the KS M:



Let $\psi_{ij} = \begin{cases} \mathsf{G}'(x_k^f \to s_k \,\mathsf{R}'\, \neg a_{ij}) & \text{if } a_{ij} = x_k \\ \mathsf{G}'(x_k^t \to s_k \,\mathsf{R}'\, \neg a_{ij}) & \text{if } a_{ij} = \neg x_k \end{cases}$ and $\psi = \bigwedge_{i,j} \psi_{ij}$.

Let $\varphi_j = \mathsf{G}'(e_{j-1} \to (\neg s_{j-1} \,\mathsf{U}'\, x_j^t) \wedge (\neg s_{j-1} \,\mathsf{U}'\, x_j^f))$ and $\varphi = \bigwedge_{j \mid Q_j = \forall} \varphi_j$.

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

Complexity of LTL

Theorem: Complexity of LTL

The following problems are PSPACE-complete:

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

The following problems are NP-complete:

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

Outline

Introduction

Models

Specifications

Satisfiability and Model Checking for LTL

5 Branching Time Specifications
- CTL*
- CTL
- Fair CTL

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.

We need quantifications on runs: $\varphi = \mathsf{AG}(p \rightarrow \mathsf{EF}\, q)$

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

Outline

Introduction

Models

Specifications

Satisfiability and Model Checking for LTL

5 Branching Time Specifications
- CTL*
- CTL
- Fair CTL

CTL^* (Emerson & Halpern 86)

Definition: Syntax of the Computation Tree Logic CTL^*

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

In this chapter, temporal modalities U, F, G, . . . are non-strict.
We may also add past modalities Y and S

Definition: Semantics of CTL^*

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.

$M, \sigma, i \models \mathsf{E}\varphi$ 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:

- $\mathsf{A}\,\varphi \equiv \neg\,\mathsf{E}\,\neg\varphi$
- $\sigma'[i] = \sigma[i]$ 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

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

Atomic propositions are state formulae: $[\![p]\!] = \{s \in S \mid p \in \ell(s)\}$
State formulae are closed under boolean connectives.

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

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

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 $\quad \psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi$

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.

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

Remark:

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

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

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

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

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

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

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.

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

Proof:

For $\psi \in \mathrm{LTL}$, let $\mathrm{MC}^\exists_{\mathrm{LTL}}(M, t, \psi)$ be the function which computes in polynomial space whether $M, t \models_\exists \psi$, i.e., if $M, t \models \mathsf{E}\,\psi$.

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure, $s \in S$ and $\varphi \in \mathrm{CTL}^*$.
Replacing $\mathsf{A}\,\psi$ by $\neg\,\mathsf{E}\,\neg\psi$ we assume φ only contains the existential path quantifier.

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

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

Outline

Introduction

Models

Specifications

Satisfiability and Model Checking for LTL

5 Branching Time Specifications
- CTL^*
- CTL
- Fair CTL

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

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

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)

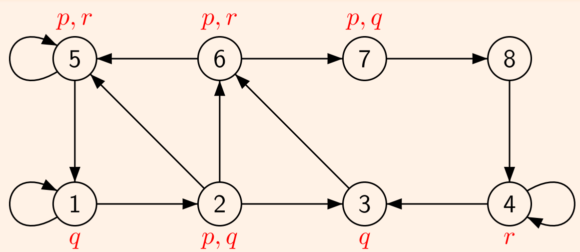
Definition: Semantics

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:



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

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

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

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

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

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

CTL (Clarke & Emerson 81)

Remark: Equivalent formulae

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

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 $p \in \ell(s)$.

Model checking of CTL

Definition: procedure semantics(φ)

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

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 $\varphi = E\varphi_1 \,\mathsf{U}\, \varphi_2$ — $\mathcal{O}(|S| + |T|)$
 semantics(φ_1); semantics(φ_2)
 $L := [\![\varphi_2]\!]$ // the "todo" set L is imlemented with a list — $\mathcal{O}(|S|)$
 $Z := [\![\varphi_2]\!]$ // the "result" is computed in the array Z — $\mathcal{O}(|S|)$
 while $L \neq \emptyset$ do — $|S|$ times
 Invariant: $L \subseteq Z$ and
 $[\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap T^{-1}(Z \setminus L)) \subseteq Z \subseteq [\![\mathsf{E}\,\varphi_1 \,\mathsf{U}\, \varphi_2]\!]$
  take $t \in L$; $L := L \setminus \{t\}$ — $\mathcal{O}(1)$
  for all $s \in T^{-1}(t)$ do — $|T|$ times
   if $s \in [\![\varphi_1]\!] \setminus Z$ then $L := L \cup \{s\}$; $Z := Z \cup \{s\}$ — $\mathcal{O}(1)$
 od
 $[\![\varphi]\!] := Z$ — $\mathcal{O}(|S|)$

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

Model checking of CTL

Definition: procedure semantics(φ)

case $\varphi = A\varphi_1 \,\mathsf{U}\, \varphi_2$ — $\mathcal{O}(|S| + |T|)$
 semantics(φ_1); semantics(φ_2)
 $L := [\![\varphi_2]\!]$ // the "todo" set L is imlemented with a list — $\mathcal{O}(|S|)$
 $Z := [\![\varphi_2]\!]$ // the "result" is computed in the array Z — $\mathcal{O}(|S|)$
 for all $s \in S$ do $c[s] := |T(s)|$ — $\mathcal{O}(|S|)$
 while $L \neq \emptyset$ do — $|S|$ times
 Invariant: $L \subseteq Z$ and
 $\forall s \in S,\ c[s] = |T(s) \setminus (Z \setminus L)|$ and
 $[\![\varphi_2]\!] \cup ([\![\varphi_1]\!] \cap \{s \in S \mid c[s] = 0\}) \subseteq Z \subseteq [\![\mathsf{A}\,\varphi_1 \,\mathsf{U}\, \varphi_2]\!]$
  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 $c[s] = 0 \wedge s \in [\![\varphi_1]\!] \setminus Z$ then $L := L \cup \{s\}$; $Z := Z \cup \{s\}$ — $\mathcal{O}(1)$
 od
 $[\![\varphi]\!] := Z$ — $\mathcal{O}(|S|)$

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

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.

Outline

Introduction

Models

Specifications

Satisfiability and Model Checking for LTL

5 Branching Time Specifications
- CTL*
- CTL
- Fair CTL

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

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

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

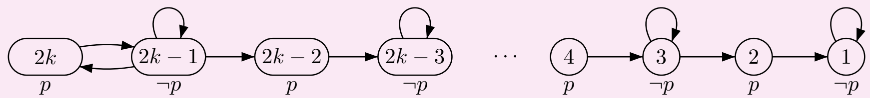
where $\mathrm{fair} = \bigwedge_i \mathsf{G}\,\mathsf{F}\,F_i$

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 $|\varphi| \leq m \leq k$ 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.