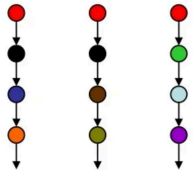
Linear Temporal Logic

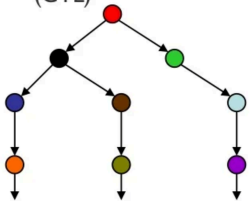
Giuseppe De Giacomo

Temporal logic

- Linear Time
  - Every moment has a unique successor
  - Infinite sequences (words)
  - Linear Time Temporal Logic (LTL)



- Branching Time
  - Every moment has several successors
  - Infinite tree
  - Computation Tree Logic (CTL)



Courtesy of Carlo Gezzi

Linear Temporal Logic (LTL)

A standard language for talking about infinite state sequences.
[Amir Pnueli - The Temporal Logic of Programs - FOCS'77]

$\top$	truth constant	$\bigcirc\varphi$	in the next state...
p	primitive propositions	$\Diamond\varphi$	will **eventually** be the case
$\neg\varphi$	classical negation	$\Box\varphi$	is **always** the case
$\varphi \vee \psi$	classical disjunction	$\varphi \mathsf{U} \psi$	φ **until** ψ
$\varphi \wedge \psi$	classical conjunction	$\varphi \mathsf{R} \psi$	φ **release** ψ

Minimal syntax

$$\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathsf{U} \varphi$$

$$\Diamond\varphi \equiv \top \mathsf{U} \varphi \qquad \Box\varphi \equiv \neg\Diamond\neg\varphi \qquad \varphi \mathsf{R} \psi \equiv \neg(\neg\varphi \mathsf{U} \neg\psi)$$

Alternative syntax in the literature

you may encounter the following notations:

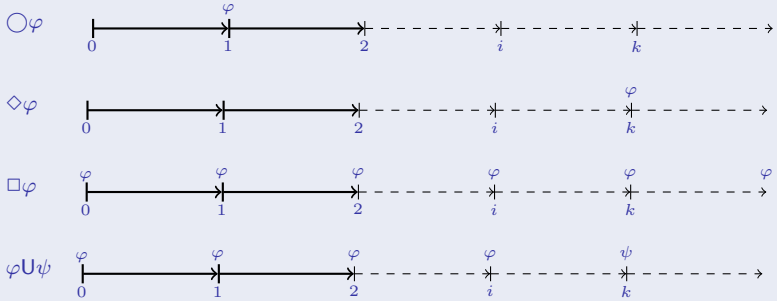
$$\begin{array}{lcl} \mathsf{X}\varphi & : & \bigcirc\varphi \\ \mathsf{F}\varphi & : & \Diamond\varphi \\ \mathsf{G}\varphi & : & \Box\varphi \end{array}$$

past operators are possible (though not strictly necessary)

Example LTL formulae

Eventually I will graduate	$\Diamond \texttt{degree}$
The plane will never crash	$\Box \neg \texttt{crash}$
I will eat pizza infinitely often	$\Box \Diamond \texttt{eatPizza}$
. . . and they all lived happily ever after	$\Diamond \Box \texttt{happy}$
We are not friends until you apologise	$(\neg \texttt{friends}) \mathsf{U} \texttt{youApologise}$
Every time it is requested, a document will be printed	$\Box(\texttt{print\_req} \rightarrow \Diamond \texttt{print})$
The two processes are never active at the same time	$\Box \neg (\texttt{proc}_1 \wedge \texttt{proc}_2)$

Semantics of LTL



LTL formulas are evaluated on infinite traces.

The language defined by an LTL formula φ is $\mathscr{L}(\varphi) = \{w \in \Sigma^\omega : w \models \varphi\}$.

LTL properties for the traffic light model

how to express
"the light is infinitely often red"
by an LTL formula?

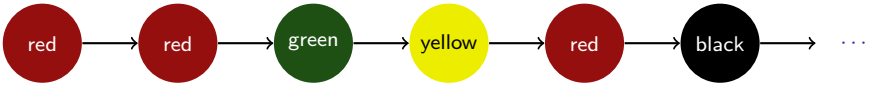
$\Box\Diamond \text{red}$

how to express
"once green, the light cannot become immediately red after"
by an LTL formula?

$\Box(\text{green} \supset \neg \bigcirc \text{red})$

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

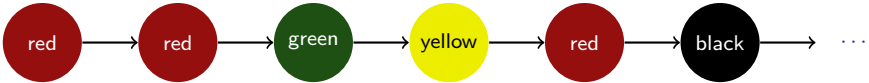


question: $\pi \models \bigcirc \bigcirc$ red?

answer: no

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

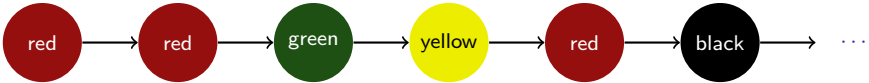


question: $\pi \models \text{red} \mathsf{U} \text{green}$?

answer: yes

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

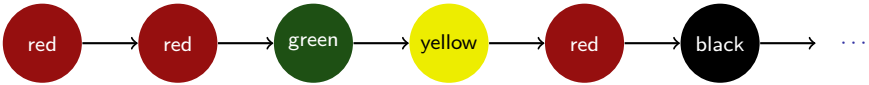


question: $\pi \models \Diamond$black?

answer: yes

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

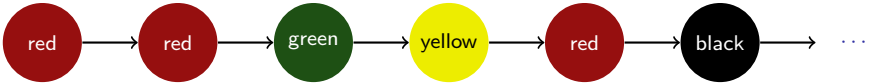


question: $\pi \models \Box\neg\text{red}$?

answer: no

Verification of LTL specs is over linear-time paths

back to the traffic light model, consider the following path:

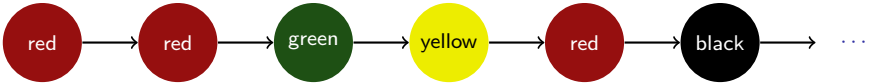


question: $\pi \models (\Diamond \text{green}) \mathsf{U} (\bigcirc \bigcirc \text{red})$?

answer: yes

Verification of LTL specs is over linear-time paths

A tricker example, consider again the path:



question: $\pi \models (\Diamond \text{black}) \mathsf{U} (\bigcirc \text{yellow})$?

answer: yes

Interesting equivalences

Duality:

$$\begin{array}{lcl} \neg\Box\varphi & \equiv & \Diamond\neg\varphi \\ \neg\Diamond\varphi & \equiv & \Box\neg\varphi \\ \neg \bigcirc \varphi & \equiv & \bigcirc\neg\varphi \end{array}$$

Absorption

$$\begin{array}{lcl} \Box\Box\varphi & \equiv & \Box\varphi \\ \Diamond\Diamond\varphi & \equiv & \Diamond\varphi \end{array}$$

$$\begin{array}{lcl} \Diamond\Box\Diamond\varphi & \equiv & \Box\Diamond\varphi \\ \Box\Diamond\Box\varphi & \equiv & \Diamond\Box\varphi \end{array}$$

Distribution:

$$\begin{array}{lcl} \Diamond(\varphi \vee \psi) & \equiv & \Diamond\varphi \vee \Diamond\psi \\ \Box(\varphi \wedge \psi) & \equiv & \Box\varphi \wedge \Box\psi \end{array}$$

But:

$$\begin{array}{lcl} \Diamond(\varphi \wedge \psi) & \not\equiv & \Diamond\varphi \wedge \Diamond\psi \\ \Box(\varphi \vee \psi) & \not\equiv & \Box\varphi \vee \Box\psi \end{array}$$

Weak Until and Release

Weak Until

Weak Until, denoted by $\varphi \mathsf{W} \psi$ says that "φ holds until ψ holds, however it is fine for ψ not to hold at all, and in that case φ holds forever". Note this is the typical interpretation of the word "until" in English. Formally it is defined as:

$$\varphi \mathsf{W} \psi \doteq \varphi \mathsf{U} \psi \vee \Box \varphi$$

Release

Release denoted by $\varphi \mathsf{R} \psi$ says that "φ releases ψ from holding forever". It can be defined as:

$$\varphi \mathsf{R} \psi \doteq \psi \mathsf{W} (\varphi \wedge \psi)$$

The following holds:

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

it also holds that

$\varphi \mathsf{U} \psi \equiv \neg(\neg\varphi \mathsf{R} \neg\psi)$ (Release is dual of Until)

Fixpoint equations

Describe temporal modalities recursively

- $\Diamond\psi \equiv \psi \vee \bigcirc\Diamond\psi$ — $\Diamond\psi$ is a the least fixpoint of $\Psi = \psi \vee \bigcirc\Psi$
- $\Box\psi \equiv \psi \wedge \bigcirc\Box\psi$ — $\Box\psi$ is a the greatest fixpoint of $\Psi = \psi \wedge \bigcirc\Psi$
- $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc\varphi \mathsf{U} \psi)$ — $\varphi \mathsf{U} \psi$ is the least fixpoint of $\Psi = \psi \vee (\varphi \wedge \bigcirc\Psi)$
- $\varphi \mathsf{R} \psi \equiv \psi \wedge (\varphi \vee \bigcirc\varphi \mathsf{R} \psi)$ — $\varphi \mathsf{U} \psi$ is the greatest fixpoint of $\Psi = \psi \wedge (\varphi \vee \bigcirc\Psi)$

Negation Normal Form for LTL

NNF

Negation Normal Form for LTL: for $a \in AP$

$$\varphi ::= \mathsf{true} \mid \mathsf{false} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \bigcirc \varphi \mid \varphi \mathsf{U} \varphi \mid \varphi \mathsf{R} \varphi$$

Theorem

Each LTL formula φ admits an equivalent in NNF sometimes denoted $\mathsf{nnf}(\varphi)$

Classes of LTL specifications

Question: what class of LTL formulas capture invariants?

Answer: $\Box\varphi$, where $\varphi ::= \text{true} \mid a \mid \varphi \wedge \varphi \mid \neg\varphi$

Example: $\Box\neg\text{red}$

Classes of LTL specifications

Question: how is the class of safety properties characterized?

answer: "nothing bad ever happens"

example: "every red light is immediately preceded by amber"

Question: how can we express this property in LTL?

answer: $\neg \text{red} \wedge \Box(\bigcirc \text{red} \supset \text{yellow})$

Classes of LTL specifications

Question: how is the class of liveness properties characterized?

answer: "something good eventually happens"

Example: "The light is infinitely often red"

Question: how can we express this property in LTL?

answer: $\Box\Diamond$red

Fairness properties in LTL

Unconditional fairness: "every transition is infinitely often taken"

$$\Box\Diamond\Psi$$

Strong fairness: "if a transition is infinitely often enabled, then it is infinitely often taken"

$$\Box\Diamond\Phi \supset \Box\Diamond\Psi$$

Weak fairness: "if a transition is continuously enabled from a certain point in time, then it is infinitely often taken"

$$\Diamond\Box\Phi \supset \Box\Diamond\Psi$$

Equivalence of LTL and CTL formulas

- CTL-formula Φ and LTL-formula φ (both over *AP*) are *equivalent*, denoted $\Phi \equiv \varphi$, if for any transition system *TS* (over *AP*):

$$TS \models \Phi \quad \text{if and only if} \quad TS \models \varphi$$

- Let Φ be a CTL-formula, and φ the LTL-formula obtained by eliminating all path quantifiers in Φ. Then: [Clarke & Draghicescu]

$\Phi \equiv \varphi$ or there does not exist any LTL-formula that is equivalent to Φ

From Joost-Pieter Katoen's slides.

LTL and CTL are incomparable

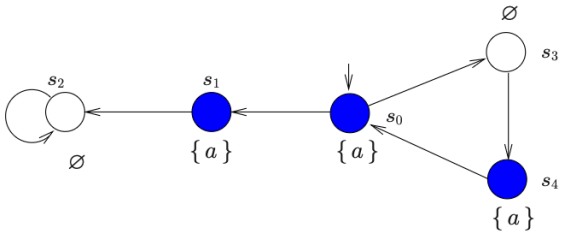
- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box a$
  - $\Diamond (a \ \wedge \ \bigcirc a)$
- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall\Diamond \forall\Box a$
  - $\forall\Diamond (a \wedge \forall\bigcirc a)$
  - $\forall\Box \exists\Diamond a$

$\Rightarrow$ Cannot be expressed = there does not exist an equivalent formula

From Joost-Pieter Katoen's slides.

Comparing LTL and CTL (1)

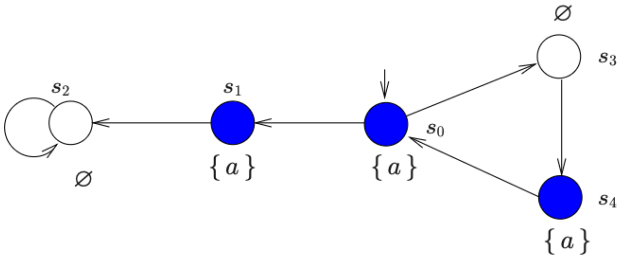
$\Diamond(a \wedge \bigcirc a)$ is not equivalent to $\forall\Diamond(a \wedge \forall\bigcirc a)$



From Joost-Pieter Katoen's slides.

Comparing LTL and CTL (1)

$\Diamond\,(a \;\wedge\; \bigcirc\, a)$ is not equivalent to $\forall\Diamond\,(a \;\wedge\; \forall\bigcirc\, a)$

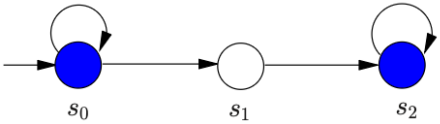


$s_0 \models \Diamond\,(a \;\wedge\; \bigcirc\, a)$ but $\underbrace{s_0 \not\models \forall\Diamond\,(a \;\wedge\; \forall\bigcirc\, a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$

From Joost-Pieter Katoen's slides.

Comparing LTL and CTL (2)

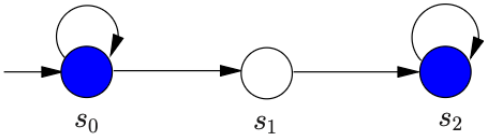
$\forall\Diamond\forall\Box\, a$ is not equivalent to $\Diamond\Box\, a$



From Joost-Pieter Katoen's slides.

Comparing LTL and CTL (2)

$\forall\Diamond\forall\Box\, a$ is not equivalent to $\Diamond\Box\, a$



$s_0 \models \Diamond\Box\, a$ **but** $\underbrace{s_0 \not\models \forall\Diamond\forall\Box\, a}_{\text{path } s_0^\omega \text{ violates it}}$

From Joost-Pieter Katoen's slides.

Test yourself!

https://brown.co1.qualtrics.com/jfe/form/SV_cGw7j9ppU9bz4P4

Curtesy of Ben Greenman – https://users.cs.utah.edu/~blg/publications/publications.html#gpdzdkmnz-fm-2024

Transition systems and their traces

Transitions Systems with Unlabelled Transitions

Consider the following variant of transition systems: $TS = (\mathrm{Prop}, S, I, \rightarrow, L)$, where

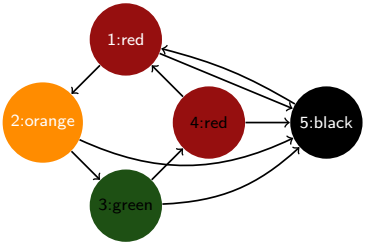
- Prop is a set of propositions (e.g., fluents, possibly including additional fluents to denote actions just executed),
- S is the set of states,
- $\rightarrow$ is the (unlabelled) transition relation,
- L is the function assigning to each state s the set $L(s) \in 2^{\mathrm{Prop}}$ of propositions p in Prop that are true in s.

Traces of a transition system

- An infinite path is an infinite state sequence $s_0 s_1 \ldots$ such that $s_0 \in I$ and $s_i \rightarrow s_{i+1}$ for all $i \geq 0$
- An infinite trace is the "output" of an infinite path: $L(s_0)L(s_1)\ldots$
- $Traces(TS)$ is the set of infinite path of TS

An example

Consider traffic lights model



Consider the (liveness) property: $\psi := \Box(\text{black} \supset \Diamond\text{red})$

Does it hold $TS \models \psi$?

answer: yes

Model-Checking

Model Checking

Verifying that a system **satisfies** a given temporal specification.

Later we will look also at "reactive synthesis"!

Reactive Synthesis

Synthesizing a system that **satisfies** a given temporal specification by construction.

Model Checking

Industry-strength approach to automated verification.

Idea: view state transition graph of a program P as a model M_P, and express correctness criteria as logic formula φ

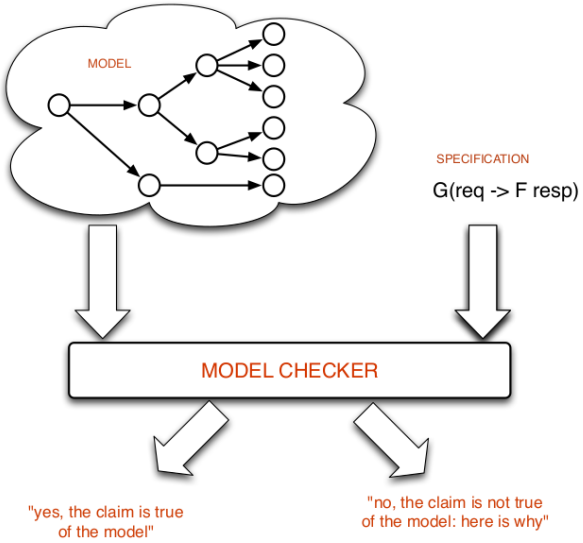
Verification then **reduces** to a model checking problem: $M_P \models \varphi$

Most widely used logical specification languages: **LTL** and **CTL**

[Amir Pnueli - The Temporal Logic of Programs - FOCS'77]]
[Clarke and Emerson - Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic - LP'81]
[Christel Baier and Joost-Pieter Katoen - Principles of Model Checking - MIT Presss 2008]

Can be (reasonably) efficiently automated, leading to many tools (SPIN, SMV, PRISM, MOCHA, MCMAS, EVE, ...).

Model Checking



LTL model checking

Main idea

LTL model checking algorithm takes:

- a model $\mathcal{T}$ and
- a formula φ

and returns

- Yes if $\mathcal{T} \models \varphi$
- No and a counter-example if $\mathcal{T} \not\models \varphi$

$\mathcal{T} \rightarrow \mathcal{L}(\mathcal{T})$

$\phi \rightarrow \mathcal{L}(\phi)$

$\downarrow$

$\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\phi)$

$\neg\neg\forall \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \rightarrow \pi \in \mathcal{L}(\phi)$

$\neg \exists \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \wedge \pi \notin \mathcal{L}(\phi)$

$\exists \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \wedge \pi \in \mathcal{L}(\neg\phi)$

$\mathcal{L}(\mathcal{T}) \cap \mathcal{L}(\neg\phi) = \phi$?

LTL model checking

Essential ideas

- Consider a model $\mathscr{T}$ and an LTL property φ
- $\mathscr{T} \models \varphi$ if for all the paths π of $\mathscr{T}$, it holds that $\pi \models \varphi$, namely if $\pi \in \mathscr{L}(\varphi)$.
- Equivalently, $\mathscr{T}$ admits no path π such that $\pi \models \neg\varphi$ (no counterexample)
- More formally

$$\begin{aligned} \mathscr{T} \models \varphi &\Leftrightarrow \mathscr{L}(\mathscr{T}) \subseteq \mathscr{L}(\varphi) \\ &\Leftrightarrow \mathscr{L}(\mathscr{T}) \cap \overline{\mathscr{L}(\varphi)} = \emptyset \\ &\Leftrightarrow \mathscr{L}(\mathscr{T}) \cap \mathscr{L}(\neg\varphi) = \emptyset \end{aligned}$$

Automata-based LTL model checking

- Input:
  - a model $\mathscr{T}$ and
  - a formula φ
- Construction:
  - Construct the automaton $A_{\mathscr{T}}$ from the LTS
  - Construct the automaton $A_{\neg\varphi}$ from the LTL formula
  - Construct the product automaton $A_{\mathscr{T},\neg\varphi} = A_{\mathscr{T}} \wedge A_{\neg\varphi}$
- Solve nonemptiness problem:

$$\mathscr{L}(A_{\mathscr{T},\neg\varphi}) \stackrel{?}{\neq} \emptyset$$

Output:

- Yes if $\mathscr{L}(A_{\mathscr{T},\neg\varphi}) = \emptyset$
- No if otherwise

(and show a counterexample path $\pi \models \neg\varphi$)

$\mathcal{T} \rightarrow \mathcal{L}(\mathcal{T})$

$\phi \rightarrow \mathcal{L}(\phi)$

$\downarrow$

$\mathcal{L}(\mathcal{T}) \subseteq \mathcal{L}(\phi)$

$\neg\neg\forall \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \rightarrow \pi \in \mathcal{L}(\phi)$

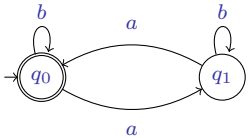
$\neg\exists \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \wedge \pi \notin \mathcal{L}(\phi)$

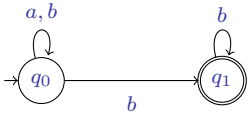
$\exists \pi.\ \pi \in \mathcal{L}(\mathcal{T}) \wedge \pi \in \mathcal{L}(\neg\phi)$

$\mathcal{L}(\mathcal{T}) \cap \mathcal{L}(\neg\phi) = \phi$?

Büchi automata

Deterministic (DBA) and nondeterministic (NBA) Büchi automata are of the similar to DFA and NFA for regular languages.





However, they read infinite words $w \in \Sigma^\omega$.

As there is no last state in the corresponding runs ρ, the acceptance condition is to visit a final state in F infinitely many times.

What are the languages recognized by the DBA and NBA depicted above?

NBA vs. DBA

Theorem

The language $L = \{w \in \Sigma^\omega : w \text{ contains finitely many } a's\}$ can be recognized by a NBA but not by any DBA.

Corollary

NBAs are strictly more expressive than DBAs.

ω-regular languages

A language is called ω-regular if it the union of expressions of the form $\alpha \cdot \beta^\omega$ with α and β being regular languages.

Theorem

1. For every ω-regular language L, there exists a NBA A_L such that $\mathscr{L}(A) = L$.
2. For every NBA A, the language $\mathscr{L}(A)$ is ω-regular.

From LTL to NBA

Theorem

For an LTL formula φ, we can construct a nondeterministic Büchi automaton $A_\varphi = \langle Q, \Sigma, I, \delta, F \rangle$ such that $\mathscr{L}(A_\varphi) = \mathscr{L}(\varphi)$.

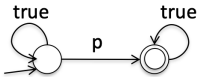
We do not look into the details on the construction of A_φ. However there are several available tools that can do it for us, including:

- `http://www.lsv.fr/~gastin/ltl2ba`
- `https://owl.model.in.tum.de/try/`
- `https://spot.lrde.epita.fr/app/`

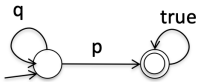
The resulting NBA can be exponential in the size for the formula in the worst case, but typically it is not.

Examples

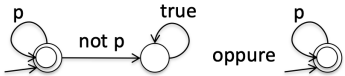
- **F** p
  L = true* p true$^\omega$



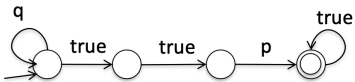
- q **U** p
  L = q* p true$^\omega$



- **G** p
  L = p$^\omega$



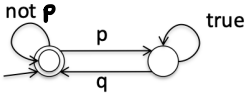
- q **U XX** p
  L = q* true true p true$^\omega$



Examples

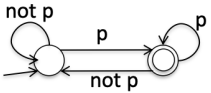
- **G** (p -> **F** q)

  L = (not p* p true q true)$^{\omega}$ +
  (not p* p true q true)* not p$^{\omega}$



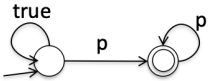
- **FG** p

  L = true*p$^{\omega}$



- **GF** p

  L = (true*p)$^{\omega}$



Exercise

From LTL to NBA in practice –*use one of the tools above!*

- $p \mathsf{U} q$
- $\Diamond p$
- $\Box p$
- $q \mathsf{U} (\bigcirc \bigcirc p)$
- $\Box (p \rightarrow \Diamond q)$
- $\Box \Diamond p$
- $\Diamond \Box p$
- $\Box \Diamond p \wedge \Box \Diamond q$

From Labeled Transition Systems to NBA

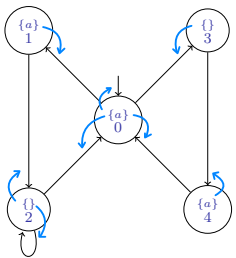
Formal definition

A labeled transition system $\mathscr{T} = \langle S, S_0, E, \lambda \rangle$ with $E \subseteq S \times S$ and $\lambda : S \to 2^{\mathrm{Prop}}$ is turned into a NBA $A_{\mathscr{T}} = \langle \Sigma, Q, Q_0, \delta, F \rangle$ with:

- $\Sigma = 2^{\mathrm{Prop}}$
- $Q = S \cup \{\epsilon\}$
- $Q_0 = \{\epsilon\}$
- $\delta(\epsilon, \sigma) = \{s \in S_0 : \sigma = \lambda(s_0)\}$
  $\delta(s, \sigma) = \{s' \in S : (s, s') \in E \text{ and } \sigma = \lambda(s')\}$
- $F = Q$

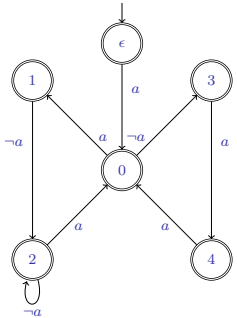
The labeling of states is pushed backward to the incoming edges.
A root state is included to push the initial state labels backward.
Every state is accepting.

From Labeled Transition Systems to NBA



$\mathcal{T} \rightarrow A_{\mathcal{T}}$

$\mathcal{L}(\mathcal{T}) \rightarrow \mathcal{L}(A_{\mathcal{T}})$



Theorem

For every labeled transition system $\mathcal{T}$, the automaton $A_{\mathcal{T}}$ recognizes all and only those infinite words that are generated by $\mathcal{T}$.

Intersection $A_{\mathscr{T},\neg\varphi}$ of $A_{\mathscr{T}}$ and $A_{\neg\varphi}$

Intuitively the intersection $A_{\mathscr{T},\neg\varphi}$ of $A_{\mathscr{T}}$ and $A_{\neg\varphi}$ is obtaining by running in parallel (synchronously) the two automata and accepting the infinite traces that are recognized by both.

Intersection of $A_{\mathscr{T}}$ and $A_{\neg\varphi}$

Consider the NBA $A_{\mathscr{T}} = \langle Q_1, \Sigma, I_1, \delta_1, Q_1 \rangle$ and $A_{\neg\varphi} = \langle Q_2, \Sigma, I_2, \delta_2, F_2 \rangle$.
The product automaton $A_{\mathscr{T}} \wedge A_{\neg\varphi}$ is an NBA $A_{\mathscr{T},\neg\varphi} = \langle Q, \Sigma, I, \delta, F \rangle$ is defined as:

- $Q = Q_1 \times Q_2$
- $I = I_1 \times I_2$
- $F = [illegible] \; Q_1 \times F_2$
- $\delta((q_1, q_2), \sigma) = (q_1', q_2')$ where $q_1' = \delta_1(q_1, \sigma)$ and $q_2' = \delta_2(q_2, \sigma)$

$$\mathscr{L}(A_{\mathscr{T}} \wedge A_{\neg\varphi}) = \mathscr{L}(A_{\mathscr{T}}) \cap \mathscr{L}(A_{\neg\varphi}).$$

LTL model checking summary

Problem

For a given LTS $\mathscr{T}$ and an LTL formula φ, Model Checking is the problem of verifying that all the executions of $\mathscr{T}$ satisfy φ. Equivalently

$$\mathscr{T} \models \varphi \iff \mathscr{L}(\mathscr{T}) \subseteq \mathscr{L}(\varphi)$$

Automata-Theoretic Approach

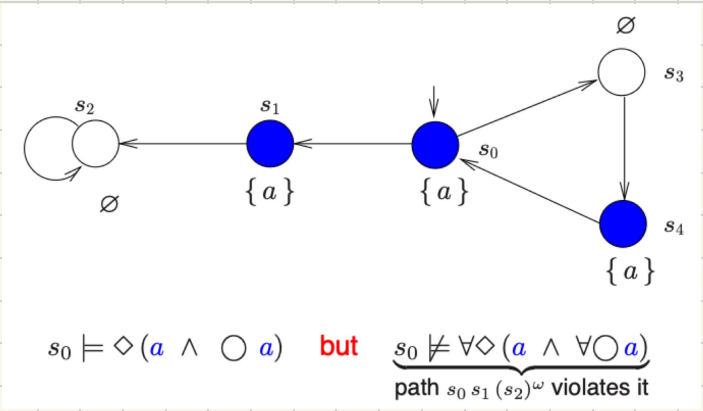
$\mathscr{T} \to A_{\mathscr{T}}$ $\qquad \mathscr{L}(\mathscr{T}) = \mathscr{L}(A_{\mathscr{T}})$

$\neg\varphi \to A_{\neg\varphi}$ $\qquad \mathscr{L}(\neg\varphi) = \mathscr{L}(A_{\neg\varphi})$

$\mathscr{L}(\mathscr{T}) \subseteq \mathscr{L}(\varphi) \iff \mathscr{L}(A_{\mathscr{T}}) \subseteq \mathscr{L}(A_{\varphi}) \iff \mathscr{L}(A_{\mathscr{T}}) \cap \mathscr{L}(A_{\neg\varphi}) = \mathscr{L}(A_{\mathscr{T},\neg\varphi}) = \emptyset$ $\qquad A_{\mathscr{T},\neg\varphi} = A_{\mathscr{T}} \wedge A_{\neg\varphi}$

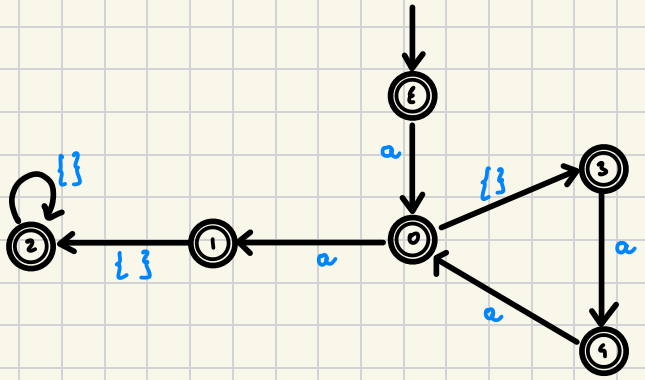
Can we complement the NBA A_φ to get $\overline{A_\varphi}$ instead?

Better not! Complementing NBAs is very costly and no good algorithms are known!

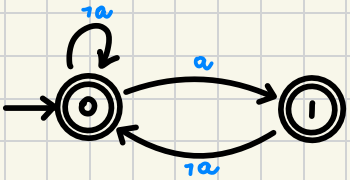


$s_0 \models \Diamond(a \wedge \bigcirc a)$ but $\underbrace{s_0 \not\models \forall\Diamond(a \wedge \forall\bigcirc a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$

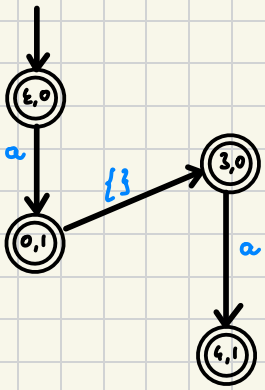
A_τ:



$A_{\neg\phi}$:



$A_\tau \wedge A_{\neg\phi}$:

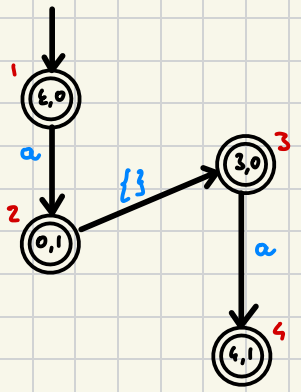


∄ UN PATH CHE RIMANE ALL' ∞ SU UNO STATO, QUINDI NON RICONOSCE NULLA

$\mathcal{L}(A_\tau \wedge A_{\neg\phi}) = \phi$

$\mathcal{L}(\tau) \subseteq \mathcal{L}(\phi) \rightarrow \tau \models \phi$

$\nu X.\, \mu Y\ (F \wedge \langle NEXT \rangle X \vee \langle NEXT \rangle Y)$



$[X_0] = \{1,2,3,4\}$

$[X_1] = [\mu Y\ (F \wedge \langle NEXT \rangle X_0 \vee \langle NEXT \rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \wedge PREE(NEXT, X_0) \vee PREE(NEXT, Y_0)$

$= \{1,2,3,4\} \wedge \{1,2,3\} \vee \phi = \{1,2,3\}$

$[Y_2] = [F] \wedge PREE(NEXT, X_0) \vee PREE(NEXT, Y_1)$

$= \{1,2,3,4\} \wedge \{1,2,3\} \vee \{1,2,3\} = \{1,2,3\}$

$[Y_1] = [Y_2] = [X_1] = \{1,2,3\}$

$[X_2] = [\mu Y\ (F \wedge \langle NEXT \rangle X_1 \vee \langle NEXT \rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \wedge PREE(NEXT, X_1) \vee PREE(NEXT, Y_0)$

$= \{1,2,3,4\} \wedge \{1,2\} \vee \phi = \{1,2\}$

$[Y_2] = [F] \wedge PREE(NEXT, X_1) \vee PREE(NEXT, Y_1)$

$= \{1,2,3,4\} \wedge \{1,2\} \vee \{1\} = \{1,2\}$

$[Y_1] = [Y_2] = [X_2] = \{1,2\}$

$[X_3] = [\mu Y\ (F \wedge \langle NEXT \rangle X_2 \vee \langle NEXT \rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \wedge PREE(NEXT, X_2) \vee PREE(NEXT, Y_0)$

$= \{1,2,3,4\} \wedge \{1\} \vee \phi = \{1\}$

$[Y_2] = [F] \wedge PREE(NEXT, X_2) \vee PREE(NEXT, Y_1)$

$= \{1,2,3,4\} \wedge \{1\} \vee \phi = \{1\}$

$[Y_1] = [Y_2] = [X_3] = \{1\}$

$[X_4] = [\mu Y\ (F \wedge \langle NEXT \rangle X_3 \vee \langle NEXT \rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \wedge PREE(NEXT, X_3) \cup PREE(NEXT, Y_0)$

$= \{1,2,3,4\} \wedge \phi \cup \phi = \phi$

$[Y_0] = [Y_1] = [X_4] = \phi$

$[X_5] = [\mu Y\ (F \wedge \langle NEXT \rangle X_4 \vee \langle NEXT \rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \wedge PREE(NEXT, X_4) \cup PREE(NEXT, Y_0)$

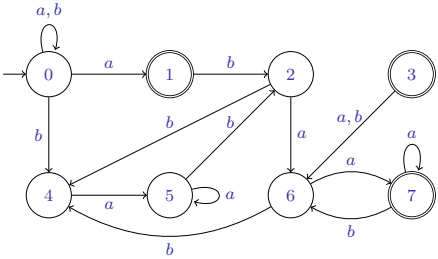
$= \{1,2,3,4\} \wedge \phi \cup \phi = \phi$

$[Y_0] = [Y_1] = [X_5] = \phi$

$[X_4] = [X_5] = \phi$

$s_1 \in [\cup X.\ \mu Y\ (F \wedge \langle NEXT \rangle X \vee \langle NEXT \rangle Y)] = \phi$? NO!

The nonemptiness problem for NFA



Question

Given a NFA A, decide whether

$$\mathscr{L}(A) \stackrel{?}{\neq} \emptyset$$

Does a word w accepted by A exist?

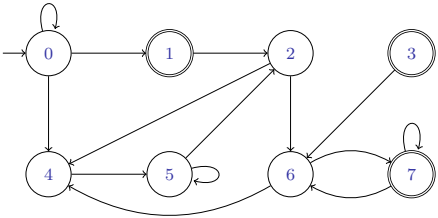
Observation

A word w is accepted by A iff there exists a run whose path starts in 0 and ends in a final state F.

Solution

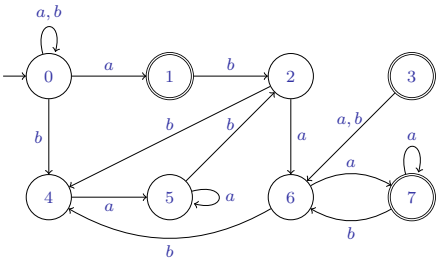
Nonemptiness of NFAs reduces to reachability over graphs.

Reachability with fix-point theory



$\mathsf{Reach}(F) = \mu\mathscr{Z}(F \vee \langle \mathsf{next} \rangle \mathscr{Z})$

The nonemptiness problem for NBA



Question

Given a NBA A, decide whether

$$\mathscr{L}(A) \stackrel{?}{\neq} \emptyset$$

Does a word w accepted by A exist?

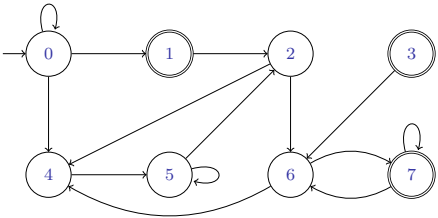
Observation

A word w is accepted by A iff there exists a run whose path starts in 0 and visits a final state in F infinitely many times.

Solution

Nonemptiness of NBAs reduces to recurrent reachability over graphs.

Recurrent reachability with fix-point theory



$$\begin{aligned}\mathsf{Buchi}(F) &= \nu\mathscr{Y}.(\mathsf{Reach}(F \wedge \langle\mathsf{next}\rangle\mathscr{Y})) \\ &= \nu\mathscr{Y}.(\mu\mathscr{Z}.((F \wedge \langle\mathsf{next}\rangle\mathscr{Y}) \vee \langle\mathsf{next}\rangle\mathscr{Z}))\end{aligned}$$

NBA nonemptiness revisited

The **nonemptiness** problem for an automaton is to decide whether there is at least one word for which there is an accepting run.

For NFA (i.e., standard nondeterministic finite automata), nonemptiness algorithms are based on reachability

In Datalog notation:

$$\begin{aligned} \mathsf{nonempty} &:- \mathsf{initial}(X),\ \mathsf{cn}(X,Y),\ \mathsf{final}(Y). \\ \mathsf{cn}(X,Y) &:- \mathsf{r}(X,A,Y). \\ \mathsf{cn}(X,Y) &:- \mathsf{r}(X,A,Z),\ \mathsf{cn}(Z,Y). \end{aligned}$$

where `initial(X)` denotes that `X` is an initial state; `final(X)` denotes that `X` is a final state `r(X,A,Y)` denotes that a transition from `X` to `Y` reading `A`; and `cn(·,·)` is the **transitive closure** of `r(X,A,Y)` projected on `X, Y`.

Notice that `cn(·,·)` is not expressible in FOL.

Reachability is a well-known problem on graphs, its complexity is NLOGSPACE-complete.

Thm: Nonemptiness for NFA a is **NLOGSPACE-complete**.

NBA nonemptiness revisited

For NBA, nonemptiness is based on **fair reachability**.

In Datalog notation:

$$\begin{aligned} \mathsf{nonempty} &:- \mathsf{initial}(X),\ \mathsf{cn}(X,Y),\ \mathsf{final}(Y),\ \mathsf{cn}(Y,Y). \\ \mathsf{cn}(X,Y) &:- \mathsf{r}(X,A,Y). \\ \mathsf{cn}(X,Y) &:- \mathsf{r}(X,A,Z),\ \mathsf{cn}(Z,Y). \end{aligned}$$

where again `initial(X)` denotes that `X` is an initial state; `final(X)` denotes that `X` is a final state `r(X,A,Y)` denotes that a transition from `X` to `Y` reading `A`; and `cn(·,·)` is the **transitive closure** of `r(X,A,Y)` projected on `X,Y`.

Fair reachability amounts to two separate reachability problems:

(1) Reach a final state from an initial state;

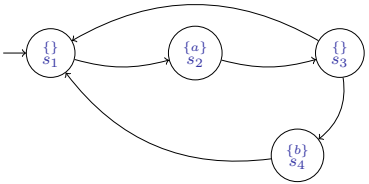
(2) From that final state reach itself via a loop.

Fair reachability has the same complexity as reachability: NLOGSPACE-complete.

Thm: Nonemptiness for NBA a is **NLOGSPACE-complete**.

Exercise

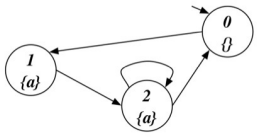
$\mathscr{T}$:

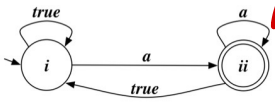


φ: $\quad \bigcirc a \wedge (\Box(b \rightarrow \bigcirc a)) \wedge \Diamond a$

Exercise 6 (optional). Model check the LTL formula $\Diamond\Box\neg a$ against the following transition system, by considering that the Büchi automaton for $\neg(\Diamond\Box\neg a)$ is the one below:

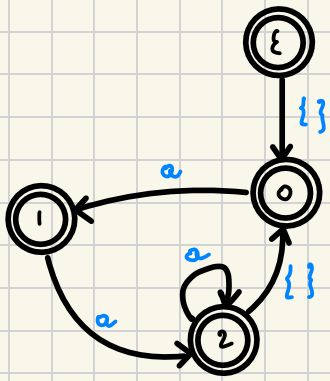
$\mathcal{T}$



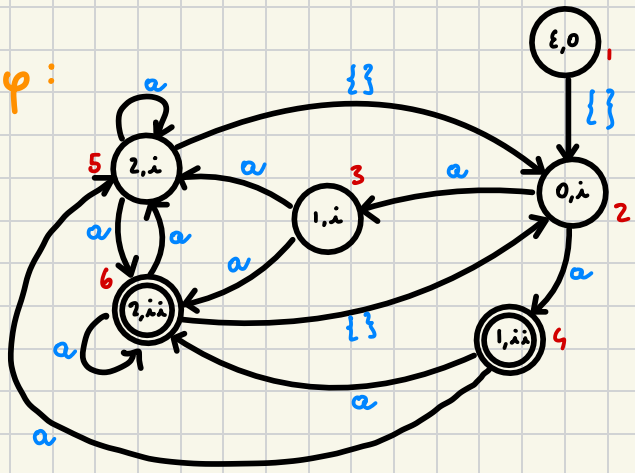


$A_{\neg\varphi}$

$A_{\mathcal{T}}$:



$A_{\mathcal{T}} \wedge A_{\neg\varphi}$:



$\nu X.\, \mu Y\, (F \wedge \langle NEXT\rangle X \vee \langle NEXT\rangle Y)$

$[X_0] = \{1,2,3,4,5,6\}$

$[X_1] = [\mu Y\, (F \wedge \langle NEXT\rangle X_0 \vee \langle NEXT\rangle Y)]$

$[Y_0] = \phi$

$[Y_1] = [F] \cap PREE(NEXT, X_0) \cup PREE(NEXT, Y_0) =$

$= \{4,6\} \cap \{1,2,3,4,5,6\} \cup \phi = \{4,6\}$

$[Y_2] = [F] \cap PREE(NEXT, X_0) \cup PREE(NEXT, Y_1) =$

$= \{4,6\} \cap \{1,2,3,4,5,6\} \cup \{2,3,4,5,6\} = \{2,3,4,5,6\}$

$[Y_3] = [F] \cap PREE(NEXT, X_0) \cup PREE(NEXT, Y_2) =$

$= \{4,6\} \cap \{1,2,3,4,5,6\} \cup \{1,2,3,4,5,6\} = \{1,2,3,4,5,6\}$

$[Y_4] = [F] \cap PREE(NEXT, X_0) \cup PREE(NEXT, Y_3) =$

$= \{4,6\} \cap \{1,2,3,4,5,6\} \cup \{1,2,3,4,5,6\} = \{1,2,3,4,5,6\}$

$[Y_3] = [Y_4] = [X_1] = \{1,2,3,4,5,6\}$

$[X_0] = [X_1] = \{1,2,3,4,5,6\}$

$s_1 \in [\nu X.\, \mu Y\, (F \wedge \langle NEXT\rangle X \vee \langle NEXT\rangle Y)] = \{1,2,3,4,5,6\}$? YES!

So $(A_{\mathcal{T}} \wedge A_{\neg\varphi}) \neq \phi \rightarrow \mathcal{T} \not\models \varphi$