Linear-time Temporal Logic

Pedro Cabalar

Department of Computer Science
University of Corunna, SPAIN
cabalar@udc.es

2015/2016

Propositional Linear-time Temporal Logic (LTL)

Syntax

- Σ = set of atoms or propositions. Example: $\Sigma = \{p, q, r\}$
- usual propositional operators $\bot, \top, \wedge, \vee, \neg, \rightarrow, \leftrightarrow$ (or $\equiv$)
- plus modal operators to talk about (linear) time

Modal operators:

- unary operators:

  $\Box$ = "*forever*"
  $\Diamond$ = "*eventually*"
  $\bigcirc$ = "*next*"

- binary operators:

  $\mathcal{U}$ = "*until*"
  $\mathcal{W}$ = "*until*" (weak version)
  $\mathcal{V}$ = "*release*" (dual of $\mathcal{U}$)

Semantics

Definition 1 (State)

Given a set of propositions Σ, a state s is a truth valuation $s : \Sigma \longrightarrow \{True, False\}$.

It can be represented as the set of (true) atoms. Example: if $\Sigma = \{p, q, r\}$ state $s = \{p, r\}$ means $s(p) = True, \quad s(q) = False, \quad s(r) = True$.

Definition 2 (Interpretation)

An *interpretation* M is an infinite sequence of states $s_0, s_1, s_2, \ldots$

Example:

$$\begin{array}{ccccccccccc} \{p,q\} & & \{p,r\} & & \{q\} & & \{q,r\} & & \emptyset & & \\ \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \ldots \\ s_0 & & s_1 & & s_2 & & s_3 & & s_4 & & \end{array}$$

Definition 3 (Satisfaction)

Let $M = s_0, s_1, \ldots$ with $i \geq 0$. We say that $M, i \models \alpha$ when:

- $M, i \models p$ if $p \in s_i$ (for $p \in \Sigma$)
- $M, i \models \Box\alpha$ if $M, j \models \alpha$ for all $j \geq i$
- $M, i \models \Diamond\alpha$ if $M, j \models \alpha$ for some $j \geq i$
- $M, i \models \bigcirc\alpha$ if $M, i+1 \models \alpha$
- $M, i \models \alpha\, \mathcal{U}\, \beta$ if there exists $n \geq i$, $M, n \models \beta$ and $M, j \models \alpha$ for all $i \leq j < n$.
- $M, i \models \alpha\, \mathcal{W}\, \beta$ if $M, i \models \Box\alpha$ or $M, i \models \alpha\, \mathcal{U}\, \beta$

Semantics

- $\bigcirc\varphi$

$$\bullet \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \bullet \longrightarrow \ldots \longrightarrow \bullet \longrightarrow \ldots$$

- $\Box\varphi$

$$\overset{\varphi}{\bullet} \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \ldots \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \ldots$$

- $\Diamond\varphi$

$$\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \ldots \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \ldots$$

Semantics

- $\varphi \, \mathcal{U} \, \psi$ = repeat φ until (mandatorily) ψ

$$\overset{\varphi}{\bullet} \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \ldots \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \overset{\psi}{\bullet} \longrightarrow \ldots$$

- $\varphi \, \mathcal{V} \, \psi$ = there is a φ before any state in which $\neg\psi$

$$\bullet \longrightarrow \overset{\varphi}{\bullet} \longrightarrow \bullet \longrightarrow \ldots \longrightarrow \bullet \longrightarrow \overset{\neg\psi}{\bullet} \longrightarrow \ldots$$

Semantics

- $\top\ \mathcal{U}\ \psi$ = repeat $\top$ until (mandatorily) ψ

$$\begin{array}{ccccccccccccc} \top & & \top & & \top & & & & \top & & \psi & & \\ \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \dots & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \dots \end{array}$$

  This is equivalent to $\Diamond\psi$.

- $\bot\ \mathcal{V}\ \psi$ = there is a $\bot$ before any state with $\neg\psi$.
  That is, we cannot have $\neg\phi$, i.e., ψ must hold forever $\Box\psi$

$$\begin{array}{ccccccccccccc} \psi & & \psi & & \psi & & & & \psi & & \psi & & \\ \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \dots & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \dots \end{array}$$

Some standard logical terminology

- Interpretation M is a *model* of theory Γ, written $M \models \Gamma$, iff $M, 0 \models \alpha$ for each formula $\alpha \in \Gamma$.
- Formula α is inconsistent or unsatisfiable iff it has no models. α is a tautology or is valid iff any interpretation is a model of α.
- α is a "logical consequence of" or "is entailed by" Γ, written $\Gamma \models \alpha$, iff any model of Γ satisfies α. Therefore, when $\Gamma = \emptyset$, what does $\models \alpha$ mean?
- Two formulas are equivalent iff they have the same models.
- LTL satisfies $\{\alpha\} \models \beta$ iff $\models \alpha \rightarrow \beta$

  In particular, α and β are equivalent iff $\models \alpha \leftrightarrow \beta$.

Some interesting equivalences

$$\Diamond\alpha \leftrightarrow \top\,\mathcal{U}\,\alpha \tag{1}$$

$$\Box\alpha \leftrightarrow \bot\,\mathcal{V}\,\alpha \tag{2}$$

$$\Box\alpha \leftrightarrow \neg\Diamond\neg\alpha \tag{3}$$

$$\Diamond\alpha \leftrightarrow \neg\Box\neg\alpha \tag{4}$$

$$\Box\alpha \leftrightarrow \alpha \wedge \bigcirc\Box\alpha \tag{5}$$

$$\Diamond\alpha \leftrightarrow \alpha \vee \bigcirc\Diamond\alpha \tag{6}$$

$$\alpha\,\mathcal{U}\,\beta \leftrightarrow (\alpha\,\mathcal{W}\,\beta) \wedge \Diamond\beta \tag{7}$$

$$\alpha\,\mathcal{W}\,\beta \leftrightarrow (\alpha\,\mathcal{U}\,\beta) \vee \Box\alpha \tag{8}$$

$$\alpha\,\mathcal{U}\,\beta \leftrightarrow \beta \vee \alpha \wedge \bigcirc(\alpha\,\mathcal{U}\,\beta) \tag{9}$$

$$\alpha\,\mathcal{V}\,\beta \leftrightarrow \neg(\neg\alpha\,\mathcal{U}\,\neg\beta) \tag{10}$$

$$\alpha\,\mathcal{V}\,\beta \leftrightarrow \beta\,\mathcal{W}\,(\beta \wedge \alpha) \tag{11}$$

(Monadic) First Order Logic

- LTL can be seen as a fragment of Predicate Calculus
- $MFO(<)$ = Monadic First Order Logic with $<$ relation
  - All predicates are monadic (1 argument) $p(x), q(y), \ldots$
  - excepting a linear order predicate $x \leq y$, binary and infix
  - arguments $x, y, \ldots$ represent time points
  - constant 0 represents the initial time point
- Example $\Box p$ can be translated as $\forall x(x \geq 0 \rightarrow p(x))$

(Monadic) First Order Logic, MFO($<$)

- We adopt some abbreviations

$$
\begin{aligned}
x = y &\stackrel{def}{=} x \leq y \ \wedge \ y \leq x \\
x < y &\stackrel{def}{=} x \leq y \ \wedge \ \neg(y \leq x) \\
x \leq y \leq z &\stackrel{def}{=} x \leq y \ \wedge \ y \leq z \\
\forall x \geq i \colon \alpha(x) &\stackrel{def}{=} \forall x(i \leq x \rightarrow \alpha(x)) \\
\exists x \geq i \colon \alpha(x) &\stackrel{def}{=} \exists x(i \leq x \wedge \alpha(x)) \\
\forall x \in i..j \colon \alpha(x) &\stackrel{def}{=} \forall x(i \leq x \leq j \rightarrow \alpha(x)) \\
\exists x \in i..j \colon \alpha(x) &\stackrel{def}{=} \exists x(i \leq x \leq j \wedge \alpha(x))
\end{aligned}
$$

- The meaning of function '$+1$' can be defined with the axiom

$$x + 1 = y \quad \leftrightarrow \quad x < y \wedge \neg\exists z(x < z \wedge z < y)$$

Kamp's translation

Temporal formula α at state i becomes MFO($<$) formula $\alpha(i)$.

$$
\begin{aligned}
(p)(i) &\stackrel{def}{=} p(i) \\
(\neg\alpha)(i) &\stackrel{def}{=} \neg\alpha(i) \\
(\alpha \wedge \beta)(i) &\stackrel{def}{=} \alpha(i) \wedge \beta(i) \\
(\alpha \vee \beta)(i) &\stackrel{def}{=} \alpha(i) \vee \beta(i) \\
(\bigcirc\alpha)(i) &\stackrel{def}{=} \alpha(i+1) \\
(\Box\alpha)(i) &\stackrel{def}{=} \forall j \geq i : \alpha(j) \\
(\Diamond\alpha)(i) &\stackrel{def}{=} \exists j \geq i : \alpha(j) \\
(\alpha\, \mathcal{U}\, \beta)(i) &\stackrel{def}{=} \exists j \geq i : (\beta(j) \wedge \forall k \in i..j-1 : \alpha(k)) \\
(\alpha\, \mathcal{V}\, \beta)(i) &\stackrel{def}{=} \forall j \geq i : (\beta(j) \vee \exists k \in i..j-1 : \alpha(k))
\end{aligned}
$$

Theorem: $M, i \models \alpha$ in LTL iff $M \models \alpha(i)$ in First-Order Logic.

Kamp's translation

- Example of use: prove the tautology $\neg(\alpha\,\mathcal{U}\,\beta) \leftrightarrow \neg\alpha\,\mathcal{V}\,\neg\beta$

$$
\begin{aligned}
(\neg(\alpha\,\mathcal{U}\,\beta))(i) \quad &\leftrightarrow \quad \neg\exists j \geq i\colon (\beta(j) \wedge \forall k \in i..j-1\colon \alpha(k)) \\
&\leftrightarrow \quad \forall j \geq i\colon \neg(\beta(j) \wedge \forall k \in i..j-1\colon \alpha(k)) \\
&\leftrightarrow \quad \forall j \geq i\colon (\neg\beta(j) \vee \neg\forall k \in i..j-1\colon \alpha(k)) \\
&\leftrightarrow \quad \forall j \geq i\colon (\neg\beta(j) \vee \exists k \in i..j-1\colon \neg\alpha(k)) \\
&\leftrightarrow \quad \forall j \geq i\colon ((\neg\beta)(j) \vee \exists k \in i..j-1\colon (\neg\alpha)(k)) \\
&\leftrightarrow \quad (\neg\alpha\,\mathcal{V}\,\neg\beta)(i)
\end{aligned}
$$

Kamp's translation

- Another example: prove the tautology $\Box\alpha \leftrightarrow \alpha \wedge \bigcirc\Box\alpha$

$$
\begin{aligned}
(\Box\alpha)(i) &= \forall j \geq i : \alpha(j) \\
&\leftrightarrow \alpha(i) \wedge \forall j \geq i+1 : \alpha(j) \\
&\leftrightarrow \alpha(i) \wedge (\Box\alpha)(i+1) \\
&\leftrightarrow \alpha(i) \wedge (\bigcirc\Box\alpha)(i) \\
&\leftrightarrow (\alpha \wedge \bigcirc\Box\alpha)(i)
\end{aligned}
$$

Exercises

Exercise 1

Prove validity of (6) *and* (9).

Exercise 2

Prove the validity of the following formulas:

$$\begin{array}{rcl} \beta & \rightarrow & \Diamond\beta \\ \beta & \rightarrow & \alpha\,\mathcal{U}\,\beta \\ \alpha\,\mathcal{U}\,\beta & \rightarrow & \Diamond\beta \end{array}$$

Kamp's theorem

Theorem 4 (Kamp 1968)

LTL is exactly as expressive as MFO($<$).

- As we saw, any LTL formula can be interpreted in MFO($<$) in a natural way.
- The real interest of Kamp's proof is the other direction: any MFO($<$) formula can be "rearranged" as an equivalent modal LTL formula.

Exercises

Exercise 3

Which are the models of $\bot\, \mathcal{U} p$*? Which are the models of* $(\bigcirc p)\, \mathcal{U} \neg p$ *?*

Exercise 4

Define an operator $\mathcal{B}$ *("before") so that* $\alpha\ \mathcal{B}\ \beta$ *means for any state in which* β *will occur, then some* α *will occur before.*

Exercise 5

Try to express the formula whose models satisfy: p *is true in all even states* $0, 2, 4, \ldots$ *leaving all the rest free.*

Outline

Examples of properties specification

Figure out the meaning of these example formulas:

- $\Box((\neg passport \vee \neg ticket) \rightarrow \bigcirc\neg board))$
- $\Box(requested \rightarrow \Diamond received)$
- $\Box(received \rightarrow \bigcirc processed)$
- $\Box(processed \rightarrow \Diamond\Box done)$
- "It can't be that we continually resend a request that is never done." The statement: $\Box requested \wedge \Box\neg done$ should be inconsistent.
  That is, we should be able to derive $\Box requested \rightarrow \Diamond done$.

An example: trains crossing

- Railroad, single rail and a road level-crossing.
- Goal: specifying properties to be satisfied.
- Propositions representing events
  - a = “A train is *a*pproaching"
  - c = “A train is *c*rossing"
  - l = “The *l*ight is blinking"
  - b = “The *b*arrier is down"

Safety properties

Safety property = something *bad* never happens = $\Box\neg bad$.

- When a train is crossing, the barrier must be down
  Solution: $\Box(c \rightarrow b) \equiv \Box\neg(c \wedge \neg b)$
- If a train is approaching or crossing, the light must be blinking
  Solution: $\Box(a \vee c \rightarrow l) \equiv \Box\neg((a \vee c) \wedge \neg l)$
- If the barrier is up and the light is off, then no train is coming or crossing. Solution: $\Box(\neg b \wedge \neg l \rightarrow \neg a \wedge \neg c) \equiv \Box\neg(\neg b \wedge \neg l \wedge (a \vee c))$

Liveness properties

Liveness property = something *initiated* eventually *terminates* = $\Box(\textit{initiated} \rightarrow \Diamond \textit{terminates})$

- When a train is approaching, a train will eventually cross
  Solution: $\Box(a \rightarrow \Diamond c)$
- Sometimes we can use $\mathcal{U}$, $\mathcal{W}$ or $\mathcal{V}$ to propagate a condition until termination.
- When a train is approaching (and nobody is crossing), the barrier will be eventually down before it crosses (if it does so)
  Solution: $\Box(a \wedge \neg c \rightarrow \neg c \; \mathcal{W} \; b)$
- If a train finishes crossing, the barrier will be eventually risen
  Solution: $\Box(c \wedge \bigcirc \neg c \rightarrow \bigcirc \Diamond \neg b)$

Infinitely often vs latching condition

- Something happens infinitely often = $\Box\Diamond$*something*.
  Example: The barrier is risen infinitely often = $\Box\Diamond\neg b$
- The dual is a latching condition = $\Diamond\Box\alpha$.
  Example: at a given point, no more trains are approaching = $\Diamond\Box\neg a$

Fairness

Fairness means that if a choice holds sufficiently often, then it is taken sufficiently often. Some examples:

- Unconditional or absolute fairness (a.k.a. impartiality)
  every process should be executed infinitely often $\Box\Diamond executed_i$
- Strong fairness every process enabled infinitely often should be executed infinitely often $\Box\Diamond enabled_i \rightarrow \Box\Diamond executed_i$
- Weak fairness every process permanently enabled after some point should be executed infinitely often
  $\Diamond\Box enabled_i \rightarrow \Box\Diamond executed_i$

Outline

A simple example of concurrent program

- An example (from “A primer on Model Checking”, M. Ben-Ari 2010).
- Two processes `P` and `Q` may increment the value of a memory cell `n` using a register `regP`, `regQ` respectively.
- They run concurrently. An interruption may arbitrarily stop the execution, store the register values, and then restore them on return.

```
byte n=0;
active proctype P() {
     byte regP=0;
 p1: regP=n;
 p2: regP++;
 p3: n=regP;
 p4: skip
}
```

```
active proctype Q() {
     byte regQ=0;
 q1: regQ=n;
 q2: regQ++;
 q3: n=regQ;
 q4: skip
}
```

A simple example of concurrent program

- A state has `(IPP,IPQ,regP,regQ,n)` where `IPP` and `IPQ` are the respective instruction pointers.
- Initial state is always `(p1,q1,0,0,0)`. Thus, each variable can take values `0,1,2` (at most, two increments are made).
- There exist $4 \times 4 \times 3 \times 3 \times 3 = 432$ states.
- We may build a non-deterministic finite automaton (NFDA) with all the transitions.

A simple example of concurrent program

- Example of computation:

```
(p1,q1,0,0,0) → (p2,q1,0,0,0)
(p3,q1,1,0,0) → (p4,q1,1,0,1)
(p4,q2,1,1,1) → (p4,q3,1,2,1)
(p4,q4,1,2,2)
```

```
p1: regP=n;   |   q1: regQ=n;
p2: regP++;   |   q2: regQ++;
p3: n=regP;   |   q3: n=regQ;
p4: skip      |   q4: skip
```

A simple example of concurrent program

- We want to check the assertion: after termination, `n=2`. That is, $p4 \wedge q4 \Rightarrow n = 2$.
- We check whether there exists an execution satisfying the negation: $p4 \wedge q4 \wedge n \neq 2$.
- After building the automaton, we obtain a counterexample path:

```
(p1,q1,0,0,0) → (p2,q1,0,0,0)
(p2,q2,0,0,0) → (p3,q2,1,0,0)
(p3,q3,1,1,0) → (p4,q3,1,1,1)
(p4,q4,1,1,1)
```

```
p1: regP=n;   |   q1: regQ=n;
p2: regP++;   |   q2: regQ++;
p3: n=regP;   |   q3: n=regQ;
p4: skip      |   q4: skip
```

Model checking algorithms

- Real problems have finite number of states (computers deal with a finite number of bits).
- But still, we deal with an unfeasible, astronomical number of cases: possible values in the memory $\times$ possible transitions in a path the NFDA $\times$ number of possible paths in the NFDA.
- Keypoint: not all the states are reachable. In our example, from 432, fixing initial state `(p1,q1,0,0,0)` only 22 are reachable.
- Model checking algorithms generate reachable states on-the-fly from initial state and property to check.
- To check whether a generated state was obtained before, a hash table storing the states is used.

Automata construction

- Model checking algorithms try to build an automaton $\mathcal{A}_P$ that captures the program behaviour.
- Given the property α to check, we generate a second automaton $\mathcal{A}_{\neg\alpha}$ for its negation.
- We take the intersection automaton $\mathcal{A}_P \cap \mathcal{A}_{\neg\alpha}$.
  1. If no path, the property is satisfied
  2. If we find a path, it is a counterexample
- "On-the-fly" techniques [Gerth,Peled,Vardi & Wolper 95] allow detecting that a property does not hold before completely constructing the full automaton.

Explicit vs Symbolic

Two possibilities:

- Explicit model checking: each automaton node is an individual state. A hash table indexes all the expanded states. SPIN uses this method.
- Symbolic model checking each node actually represents a set of states. Typically, each set of states is represented with a Binary Decision Diagram (BDD). SMV uses this method.

Model checking techniques

- Partial order reduction: the keypoint is detecting when the ordering of interleavings is irrelevant.

  Example: n processes can execute instructions l_1, l_2, ... l_n in any ordering. We have $n!$ combinations, but we can fix an arbitrary one when ordering is irrelevant for the property to check.

- Bounded model checking: when we want to check if property α is violated in k or fewer steps ($k \geq 0$ finite).

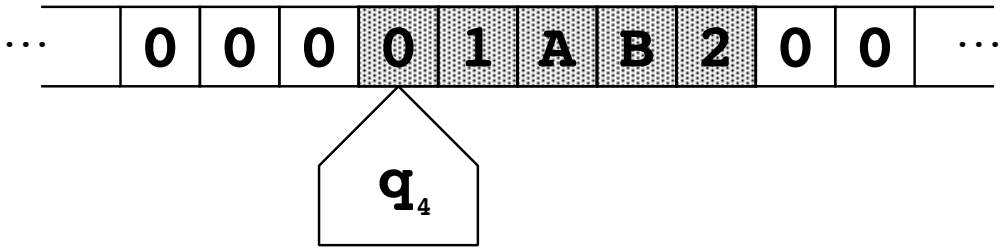
  Fixing the path length $i \leq k$ we can translate the problem to SAT (propositional satisfiability). Iterative deepening goes increasing $i = 1, 2, \ldots, k$ until a counterexample is found or k reached.

Outline

Complexity

- In complexity theory, solving a decision problem means building an algorithm that, in a finite number of steps, answers yes or no to a given input query.
- For instance, *SAT* (propositional satisfiability, i.e., "does a formula α have any model?") is a decision problem, and its complexity class is **NP**-complete.
- Other examples of **NP**-complete problems are: the Travelling Salesman problem, the Graph Coloring problem, Subset Sum problem (find non-empty subset of integers that sum 0).

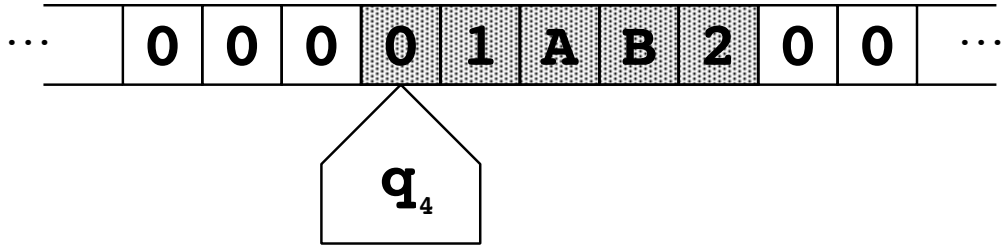
Meaning of NP-completeness



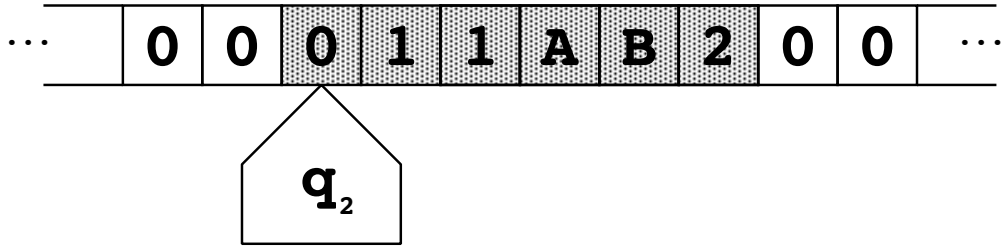
- A Turing Machine (TM) is a theoretical device that operates on an infinite tape with cells containing symbols in a finite alphabet (including the blank or '0')
- The TM has a current state S_i among a finite set of states (including '*Halt*'), and a head pointing to the "current" cell in the tape.
- It has an associated transition function that describes the next step.

Meaning of NP-completeness

- Example: with scanned symbol 0 and state q_4, write 1, move *Left* and go to state q_2. That is:



$$t(0, q_4) = (1, Left, q_2)$$

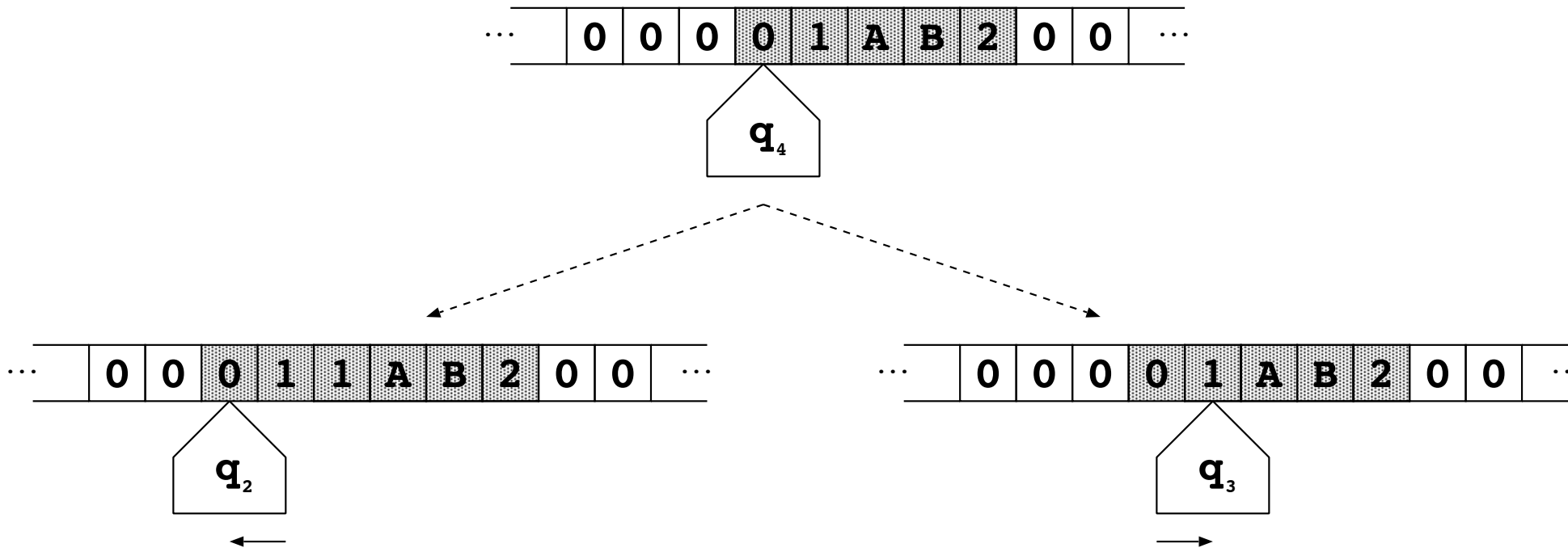


Meaning of NP-completeness

- A decision problem consists in providing a given tape input and asking the Turing Machine for a final output symbol answering *Yes* or *No*.
- Example: *SAT* = given (an encoding of) a propositional formula, does it have at least one model?
- A decision problem is in complexity class **P** iff the number of steps carried out by the TM is polynomial on the size n of the input.

Meaning of NP-completeness

- Now, a non-deterministic Turing Machine (NDTM) is such that the transition function is replaced by a transition relation.
- We may have different possibilities for the next step.
- Example: $t(0, q_4, 1, Left, q_2)$, $t(0, q_4, 0, Right, q_3)$



Meaning of NP-completeness

- Keypoint: an NDTM provides an affirmative answer to a decision problem when at least one of the executions for the same input answers *Yes*.
- A decision problem is in class **NP** iff the number of steps carried out by the NDTM is polynomial on the size n of the input.
- For *SAT*, we can build an NDTM that performs two steps:
  1. For each atom, generate 1 or 0 nondeterministically. This provides an arbitrary interpretation in linear time.
  2. Test whether the current interpretation is a model or not.

  The sequence of these two steps takes polynomial time.

Meaning of NP-completeness

- Unsolved problem

$$\boxed{\mathbf{P} \stackrel{?}{=} \mathbf{NP}}$$

- The most accepted conjecture is that $\mathbf{P} \subset \mathbf{NP}$. But remains unproved.

- It is one of the 7 Millenium Prize Problems
  http://www.claymath.org/millennium/P_vs_NP/
  The Clay Mathematics Institute designated $1 million prize for its solution!

Meaning of NP-completeness

- A problem X is **C**-complete, for some complexity class **C**, iff any problem Y in **C** is reducible to X in polynomial-time.
- A complete problem is a representative of the class. Example: if an **NP**-complete problem happened to be in **P** then **P** $=$ **NP**.
- *SAT* was the first problem to be identified as **NP**-complete (Cook's theorem, 1971).
- *SAT* is commonly used nowadays for showing that a problem X is at least as complex as **NP**. To this aim, just encode *SAT* into X.

LTL-satisfiability is PSPACE-complete

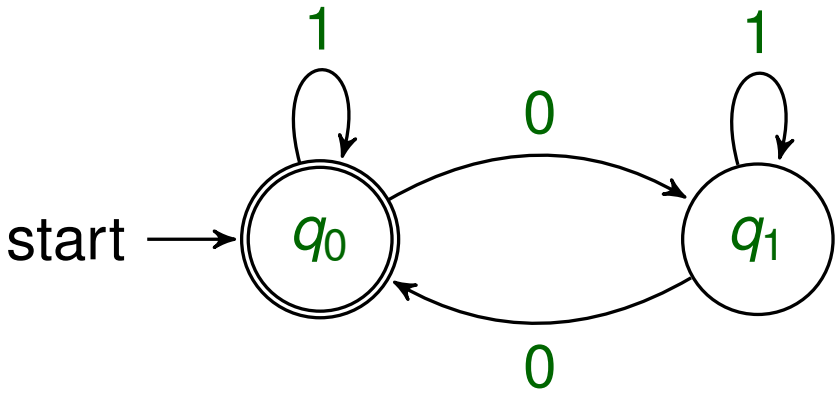
Theorem 5

[Halpern & Reif 1981], [Sistla & Clarke, 1982]
LTL-satisfiability is **PSPACE***-complete.*

- **PSPACE** is the set of decision problems that can be solved by a Turing Machine using a polynomial amount of space (for a finite, unlimited time).
- There is no difference when the machine is non-deterministic **NPSPACE** $=$ **PSPACE** [Savitch 1970].
- On the other hand, **NP** $\subseteq$ **PSPACE**. Again, unsolved question **NP** $\stackrel{?}{=}$ **PSPACE** but strongly suspected to be $\neq$.
- Other **PSPACE**-complete problems are: Quantified Boolean Formula satisfiability, AI-Planning (STRIPS) existence of plan.

LTL and automata

- An finite state machine or finite automaton is a tuple $(Q, A, \delta, q_0, \mathbf{F})$ where
  - Q is a finite set of states
  - A is a finite set called the alphabet
  - $\delta : Q \times A \to Q$ is the transition function
  - q_0 is the initial state
  - $\mathbf{F}$ is the set of accepting or final states
- Example: this automaton recognizes words containing an even



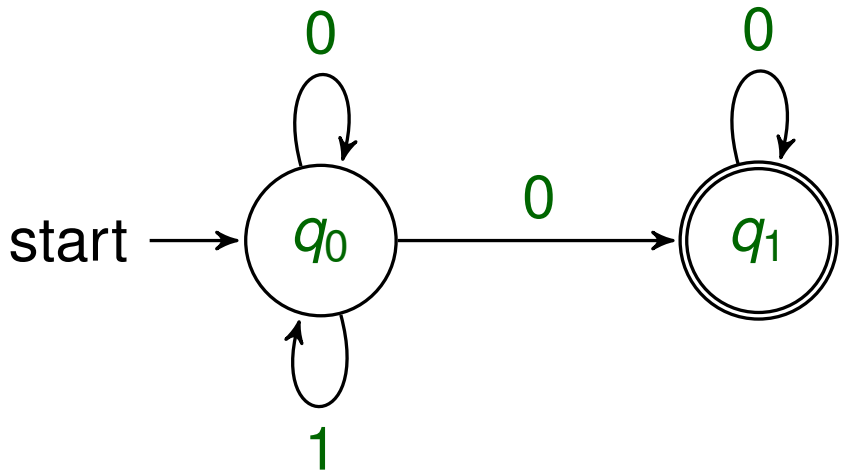
number of 0's

LTL and automata

- ω-automata are a variation where the accepted language consists of words of infinite length. They define different acceptance conditions (when we consider a word to be "accepted")
- A Büchi automaton (BA) is an ω-automaton with the acceptance condition:

  There is some run that visits (at least) one of the states in **F** *infinitely often*

- Example: this automaton recognizes the language $(0 + 1)^*0^\omega$

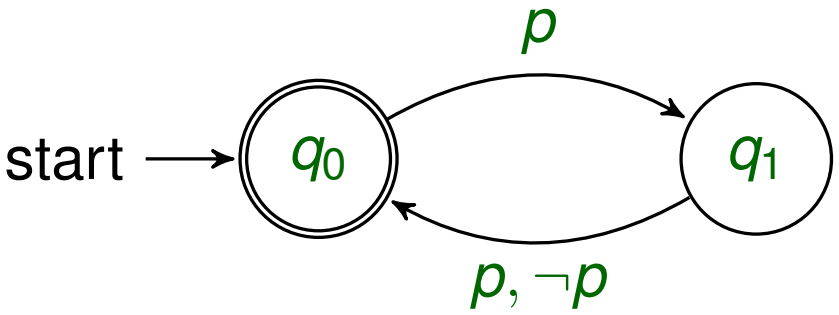


LTL and automata

- During model checking, LTL properties are translated into "equivalent" BA's
- By equivalent we mean they recognize the same language. The BA alphabet A corresponds to the set of possible LTL states.
- Example: if the formula uses atoms $\Sigma = \{p, q\}$ then $A = 2^{\Sigma} = \{\emptyset, \{p\}, \{q\}, \{p, q\}\}$
- Usually, each BA arc is labelled with a set of states that yield the same transition. This set of states is actually represented as an LTL formula.

LTL and automata

- A language accepted by a non-deterministic BA is called regular ω-language.
- An important restriction: LTL is less expressive than Büchi automata.
- For instance, Exercise 4 (make p true in even states and free in all the rest) cannot be represented in *LTL* whereas it is accepted by the Büchi automaton:



- Other temporal logics do cover regular ω-languages.

Outline

Inference Methods

Inference or formal proof: we make syntactic manipulation of formulae. To do so, we use:

- An initial set of formulae: axioms.
- Syntactic manipulation rules: inference rules.
- As a result of applying these rules, we go obtaining new formulae: theorems

Inference methods

- Notation: $\Gamma \vdash \alpha$ means that formula α can be derived or inferred from theory Γ.
- Usually, axioms are not represented inside Γ. Thus, $\vdash \alpha$ means that α is a theorem (from logic **L**).
- Given a language $\mathcal{L}$, a logic **L** is a subset of $\mathcal{L}$. It can be defined:
  - Semantically: $\mathbf{L} = \{\alpha \in \mathcal{L} \mid \models \alpha\}$.
  - Syntactically: $\mathbf{L} = \{\alpha \in \mathcal{L} \mid \vdash \alpha\}$.
- What should we expect from an inference method?
  - Soundness (or correctness): if $\vdash \alpha$ then $\models \alpha$
  - Completeness: if $\models \alpha$ then $\vdash \alpha$

A deductive system

We define the *LTL* deductive system as follows.

Axioms:

Ax0	*PC*	Any substitution instance of any Propositional Calculus tautology
Ax1	$\vdash \Box(\alpha \to \beta) \to (\Box\alpha \to \Box\beta)$	Distribution of $\Box$ over $\to$
Ax2	$\vdash \bigcirc(\alpha \to \beta) \to (\bigcirc\alpha \to \bigcirc\beta)$	Distribution of $\bigcirc$ over $\to$
Ax3	$\vdash \Box\alpha \to (\alpha \wedge \bigcirc\alpha \wedge \bigcirc\Box\alpha)$	Expansion of $\Box$
Ax4	$\vdash \Box(\alpha \to \bigcirc\alpha) \to (\alpha \to \Box\alpha)$	Induction
Ax5	$\vdash \bigcirc\alpha \leftrightarrow \neg\bigcirc\neg\alpha$	Linearity

Inference rules:

MP	$\frac{\vdash\alpha,\ \vdash\alpha\to\beta}{\vdash\beta}$	Modus Ponens
N	$\frac{\vdash\alpha}{\vdash\Box\alpha}$	Necessitation

A deductive system

An example of a proof

Theorem 6 (transitivity)

$\vdash \Box\Box p \leftrightarrow \Box p$

Proof:

1.	$\vdash \Box\Box p \rightarrow \Box p$	Expansion
2.	$\vdash \Box p \rightarrow \bigcirc\Box p$	Expansion
3.	$\vdash \Box(\Box p \rightarrow \bigcirc\Box p)$	Necessitation on 2
4.	$\vdash \Box(\Box p \rightarrow \bigcirc\Box p) \rightarrow (\Box p \rightarrow \Box\Box p)$	Induction
5.	$\vdash \Box p \rightarrow \Box\Box p$	Modus Ponens on $3, 4$
6.	$\vdash \Box\Box p \leftrightarrow \Box p$	P.C. $1, 5$

Q.E.D.

A deductive system

Derived inference rules:

G$_\Box$	$\frac{\vdash \alpha \rightarrow \beta}{\vdash \Box\alpha \rightarrow \Box\beta}$	$\Box$ – Generalization
G$_\bigcirc$	$\frac{\vdash \alpha \rightarrow \beta}{\vdash \bigcirc\alpha \rightarrow \bigcirc\beta}$	$\bigcirc$ – Generalization
Ind	$\frac{\vdash \alpha \rightarrow \bigcirc\alpha}{\vdash \alpha \rightarrow \Box\alpha}$	Induction

These rules can be derived from previous axioms and rules.

A deductive system

Exercises

Exercise 6

Prove the following theorems:

$$\vdash \Box(p \wedge q) \leftrightarrow \Box p \wedge \Box q$$
$$\vdash \Diamond(p \vee q) \leftrightarrow \Diamond p \vee \Diamond q$$

Exercise 7

Prove the theorem

$$\vdash \Box p \vee \Box q \rightarrow \Box(p \vee q)$$

and find a counterexample for:

$$\Box(p \vee q) \rightarrow \Box p \vee \Box q$$

Outline

Semantic tableaux

- For simplicity, we assume $\alpha \rightarrow \beta \stackrel{def}{=} \neg\alpha \vee \beta$ and $\alpha \leftrightarrow \beta \stackrel{def}{=} (\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$
- With respect to Propositional Calculus tableaux, we add unfolding rules for modal operators as follows:

Propositional Calculus rules

Formula	Branch 1	Branch 2
$\alpha \vee \beta$	α	β
$\alpha \wedge \beta$	α, β	
$\neg(\alpha \vee \beta)$	$\neg\alpha, \neg\beta$	
$\neg(\alpha \wedge \beta)$	$\neg\alpha$	$\neg\beta$
$\neg\neg\alpha$	α	

Modal rules

Formula	Branch 1	Branch 2
$\Box\alpha$	$\alpha, \bigcirc\Box\alpha$	
$\neg\Diamond\alpha$	$\neg\alpha, \neg\bigcirc\Diamond\alpha$	
$\Diamond\alpha$	α	$\bigcirc\Diamond\alpha$
$\neg\Box\alpha$	$\neg\alpha$	$\neg\bigcirc\Box\alpha$

Semantic tableaux

- When these rules are exhausted, each tableau leaf is boxed and (partially) represents a state
- The state usually contains $\bigcirc$-formulas like $\bigcirc\alpha$ or $\neg\bigcirc\alpha$. In such a case, we generate a transition to a next state whose content is fixed with the new rules:

Formula	Next state
$\bigcirc\alpha$	α
$\neg\bigcirc\alpha$	$\neg\alpha$

- We can reach a state repeated in previous tableau node. If so, we just label the previous node and reuse it

Semantic tableaux

- Example: take $(p \vee q) \wedge \bigcirc(\neg p \wedge \neg q)$

$$(p \vee q) \wedge \bigcirc(\neg p \wedge \neg q)$$

$$(p \vee q), \bigcirc(\neg p \wedge \neg q)$$

$$\boxed{p, \bigcirc(\neg p \wedge \neg q)} \qquad \boxed{q, \bigcirc(\neg p \wedge \neg q)}$$

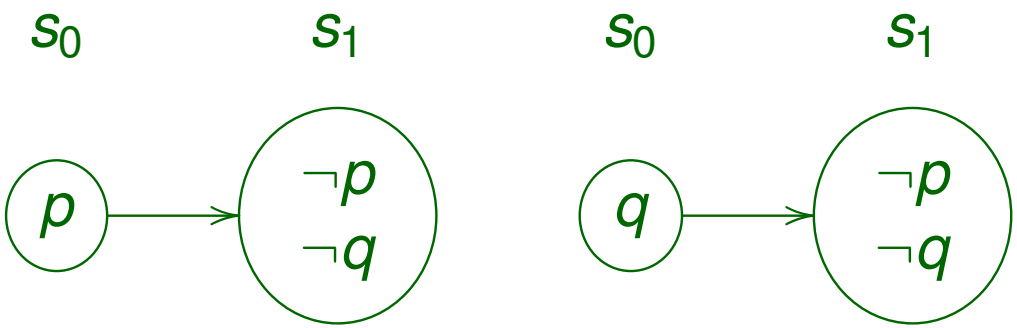
- Both open branches yield to a transition to a new state where:

$$\neg p \wedge \neg q$$

$$\boxed{\neg p, \neg q}$$

Semantic tableaux

- That is, any model of $(p \vee q) \wedge \bigcirc(\neg p \wedge \neg q)$ must contain one of the following structures:



- These are called Hintikka structures. They can be expanded to interpretations (arbitrarily completing the truth of the rest of atoms)

Semantic tableaux

- Example 2: is $\Box(p \wedge q) \rightarrow \Box p$ valid?
- We negate the formula and check if we obtain a closed tableau

$$\neg(\Box(p \wedge q) \rightarrow \Box p)$$

$$l_0 : \Box(p \wedge q), \Diamond\neg p$$

$$p \wedge q, \bigcirc\Box(p \wedge q), \Diamond\neg p$$

$$p, q, \bigcirc\Box(p \wedge q), \Diamond\neg p$$

$$p, q, \bigcirc\Box(p \wedge q), \neg p$$

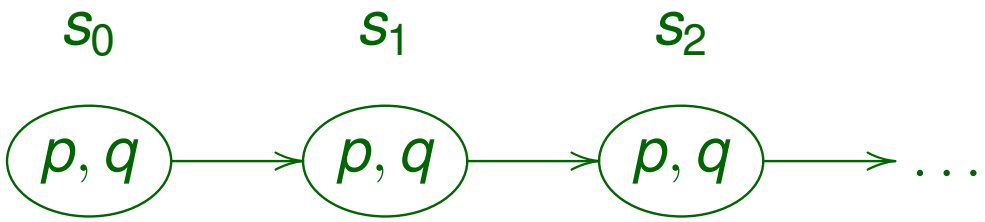
$$\times$$

$$\boxed{p, q, \bigcirc\Box(p \wedge q), \bigcirc\Diamond\neg p}$$

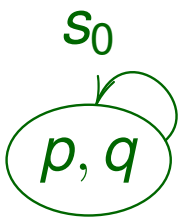
We would create a new state with $\Box(p \wedge q), \Diamond\neg p = l_0$

Semantic tableaux

- The tableau is open but generates the following Hintikka structure:



or simply

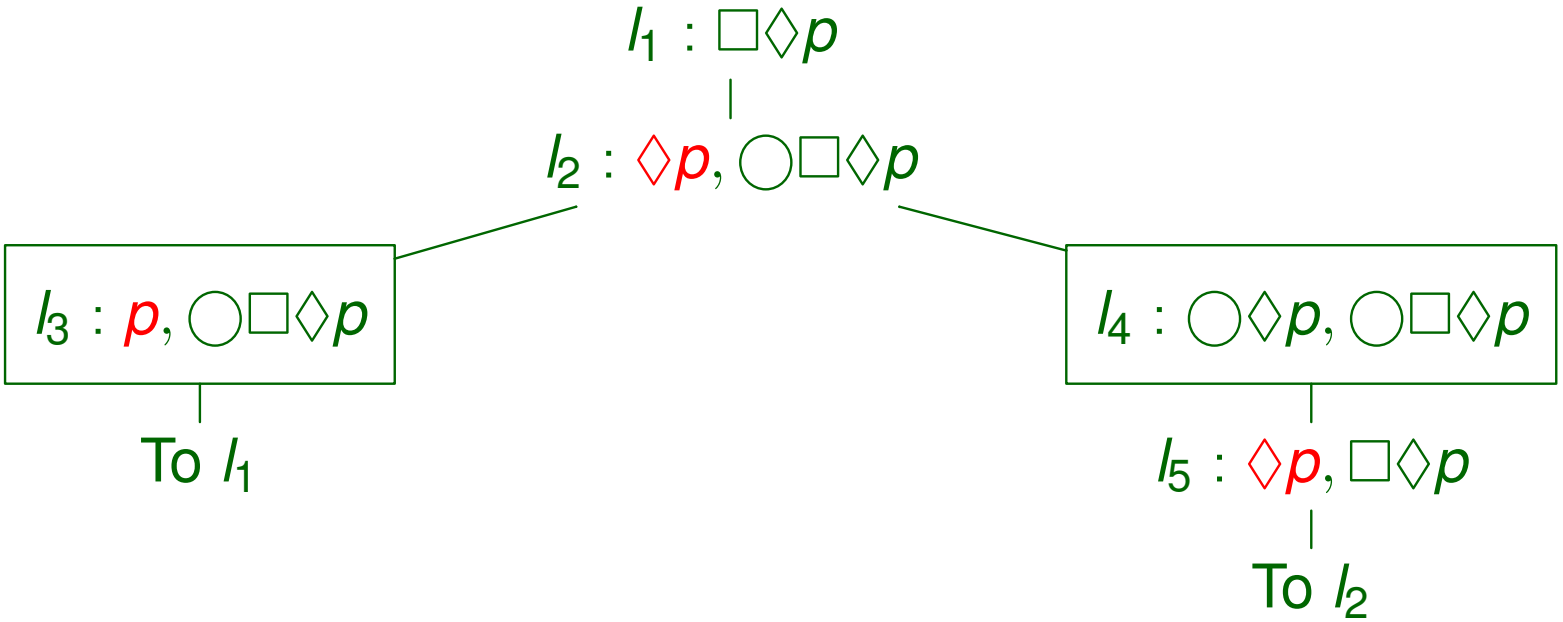


which is never a model because $\Diamond\neg p$ is never fulfilled

- For open tableaux, we will have to check fulfillment of $\Diamond\alpha$ formulas

Semantic tableaux

- Example $\Box\Diamond p$



$\Diamond\alpha$ formulas are fulfilled, so the Hintikka structure represents possible models: