Temporal Logics

and Automata

Pierre Wolper

Université de Liège

What is (Linear-Time) Temporal logic?

- Formulas built from a set of propositions P, Boolean connectives $\neg$, $\wedge$, $\vee$, and the temporal operators $\bigcirc$, $\Box$, $\Diamond$, and U.

- Interpretations are infinite sequences $\sigma : \mathbb{N} \rightarrow 2^P$.

Temporal Operators

The temporal operators indicate in which state(s) of an infinite sequence a formula should be interpreted.

1. $\bigcirc$ which is read "at the next time" (in the next state of the sequence)

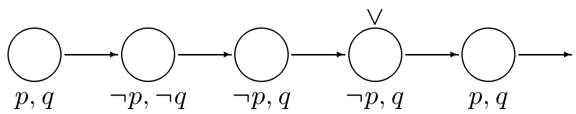
2. $\Box$ which is read "always in the future" (in all future states of the sequence)

3. $\Diamond$ which is read "eventually" (in some future state of the sequence)

4. U which is read "until" (binary operator)

Examples

In the marked state of the sequence



the following formulas are *true*:

$$\neg p \wedge q$$

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

$$\Diamond p$$

$$q \, U \, p$$

the following formula is *false*:

$$\Box p$$

Note:
A linear time temporal logic formula is a description of a set of infinite sequences (those that satisfy it)

Why is Temporal logic Used in Verification?

- The interpretations of temporal logic (infinite sequences of states) naturally coincide with a direct representations of computations.
- Temporal logic can easily express useful properties of computations, e.g. invariance ($\Box f$), qualitative progress ($\Diamond f$) . . .
- Propositional temporal logic is decidable and there are exploitable algorithms for checking that a system satisfies a temporal logic property.
- These algorithms work by reducing the verification problem to simple combinatorial problems for which a lot of machinery can be used for improving efficiency.
- The key is the close link between temporal logic and automata.

Words and Automata

- A finite word of length n over an alphabet Σ is a mapping $w : \{0, \ldots n-1\} \rightarrow \Sigma$
- An automaton on finite words is a description of a set of finite words (those that it accepts).
- An infinite word (ω-word) over an alphabet Σ is a mapping $w : \mathbb{N} \rightarrow \Sigma$
- An automaton on infinite words is a description of a set of infinite words (those that it accepts).

Relating Temporal Logic and Automata

Linear time temporal logic formulas and automata on infinite words describe exactly the same type of objects: sets of infinite sequences.

- There is a very natural translation from temporal logic to automata.
- An automaton can be viewed as a computationally oriented way of writing a temporal logic formula.
- Thus one can use the logic when ease of expression is the issue, and automata when algorithmic manipulation is needed.

More about Automata on Infinite Words

Same structure as automata on finite words:

$A = \{\Sigma, S, \delta, S_0, F\}$ where

- Σ is an alphabet,
- S is a set of states,
- $\delta : S \times \Sigma \rightarrow 2^S$ is a transition relation,
- $S_0 \subseteq S$ is a set of starting states, and
- $F \subseteq S$ is a set of accepting states.

What changes is the semantics.

When does an Automaton accept a Word?

Finite words $w : \{0, \ldots n-1\} \to \Sigma$

The word w is accepted by the automaton if there is a labeling of the word by states

$$\rho : \{0, \ldots n\} \to S$$

such that

- $\rho(0) \in S_0$ (the initial label is an initial state),
- $\forall 0 \leq i \leq n-1$, $\rho(i+1) \in \delta((\rho(i), w(i))$ (the labeling is compatible with the transition relation),
- $\rho(n) \in F$ (the last label is an accepting state).

Don't think of automata as programs, but as a formalism for defining sets of words that has a very simple semantics.

Infinite words $w : \mathbb{N} \to \Sigma$

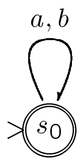
The word w is accepted by the automaton if there is a labeling of the word by states

$$\rho : \mathbb{N} \to S$$

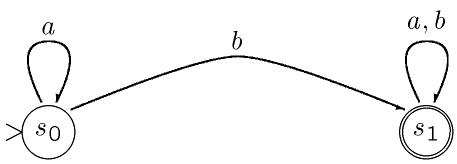
such that

- $\rho(0) \in S_0$ (the initial label is an initial state),
- $\forall i \geq 0$, $\rho(i+1) \in \delta((\rho(i), w(i))$ (the labeling is compatible with the transition relation),
- $inf(\rho) \cap F \neq \emptyset$ where $inf(\rho)$ is the set of states that appear infinitely often in ρ (the set of repeating states intersects F).

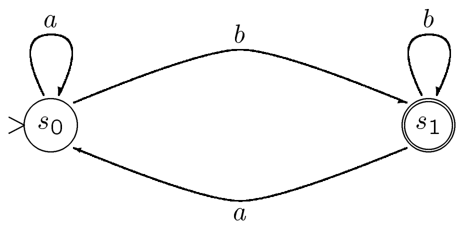
Examples



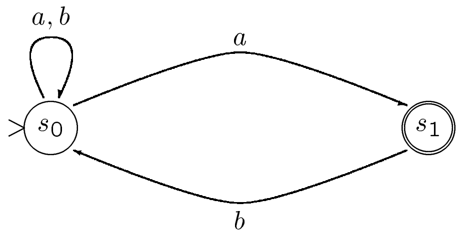
All words over $\Sigma = \{a, b\}$



An automaton accepting $a^\star b(a \cup b)^\omega$



All words containing b infinitely often ($\Box\Diamond b$)



Infinitely often ab

Other types of Acceptance conditions

Büchi: $F \subseteq S$,
$\inf(\rho) \cap F \neq \emptyset$.

Generalized Büchi: $\mathcal{F} \subseteq 2^S$,
i.e. $F = \{F_1, \ldots, F_m\}$
For each F_i, $\inf(\rho) \cap F_i \neq \emptyset$.

Rabin: $F \subseteq 2^Q \times 2^Q$,
i.e. $F = \{(G_1, B_1), (G_2, B_2), \ldots, (G_m, B_m)\}$
For some pair $(G_i, B_i) \in F$, $\inf(\rho) \cap G_i \neq \emptyset$ and $\inf(\rho) \cap B_i = \emptyset$

Streett: $F \subseteq 2^Q \times 2^Q$,
i.e. $F = \{(G_1, B_1), (G_2, B_2), \ldots, (G_m, B_m)\}$
$\inf(\rho) \cap G_i = \emptyset$ or $\inf(\rho) \cap B_i \neq \emptyset$

For nondeterministic automata, all define the ω-regular languages. For deterministic automata, Rabin and Street conditions are stronger.

Properties of Büchi Automata

- Closed under union, intersection, projection, and complementation.
- Non-emptiness easy to decide:
  - Check if some accepting state is accessible from an initial state and (nontrivially) from itself.
  - Linear time.
  - NLOGSPACE.

Closure under Union

Given $A_1 = (\Sigma, S_1, \delta_1, S_{01}, F_1)$ and $A_2 = (\Sigma, S_2, \delta_2, S_{02}, F_2)$,

the automaton $A = \{\Sigma, S, \delta, S_0, F\}$ where

- $S = S_1 \cup S_2, \quad S_0 = S_{01} \cup S_{02}, \quad F = F_1 \cup F_2$,
- $t \in \delta(s, a)$ if
$$\begin{cases} \text{either } t \in \delta_1(s,a) & \text{and } s \in S_1, \\ \text{or } t \in \delta_2(s,a) & \text{and } s \in S_2. \end{cases}$$

accepts $L(A_1) \cup L(A_2)$.

Closure under Intersection

Given $A_1 = (\Sigma, S_1, \delta_1, S_{01}, F_1)$
and $A_2 = (\Sigma, S_2, \delta_2, S_{02}, F_2)$,

The generalized Büchi automaton and
$A = \{\Sigma, S, \delta, S_0, \mathcal{F}\}$ where

- $S = S_1 \times S_2, \quad S_0 = S_{01} \times S_{02}$,
- $\mathcal{F} = \{F_1 \times S_2, F_2 \times S_1\}$,
- $(u, v) \in \delta((s, t), a)$ if $u \in \delta_1(s, a)$ and $v \in \delta_2(t, a)$.

accepts the language $L(A_1) \cap L(A_2)$.

Projection is also simple, but *complementation* is completely nontrivial (the Rabin-Scott subset construction is not sufficient!).

From Generalized Büchi to Büchi

Given $A = (\Sigma, S, \delta, S_0, \mathcal{F})$,
where $\mathcal{F} = \{F_1, \ldots, F_k\}$,

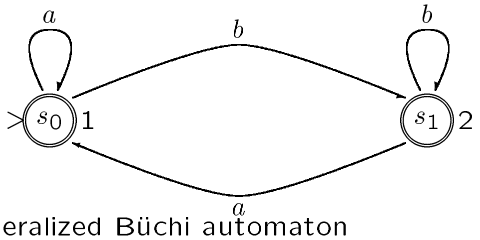
$A' = (\Sigma, S', \delta', S'_0, F')$ where

- $S' = S \times \{1, \ldots, k\}$.
- $S'_0 = S_0 \times \{1\}$.
- δ' is defined by $(t, i) \in \delta'((s, j), a)$ if

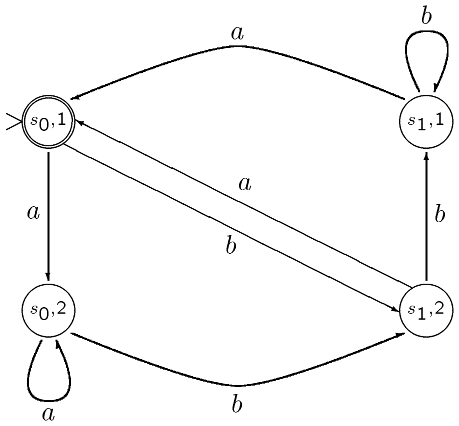
  $t \in \delta(s, a)$ and $\begin{cases} i = j & \text{if } s \notin F_j, \\ i = (j \bmod k) + 1 & \text{if } s \in F_j. \end{cases}$

- $F' = F_1 \times \{1\}$.

accepts $L(A)$.



A generalized Büchi automaton

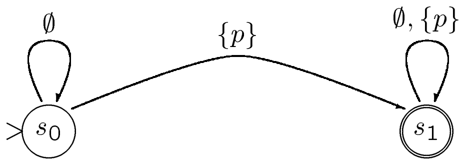


From generalized Büchi to Büchi

From TL to Automata

Given a TL formula f built from a set of atomic propositions P, construct an automaton on infinite words over the alphabet 2^P that accepts exactly the infinite sequences satisfying f.

Example:



An automaton for $\diamond p$

Idea of the Construction

To determine if a sequence $\sigma : \mathbb{N} \rightarrow 2^P$ satisfies a formula f, one can proceed by labeling it with subformulas and negations of subformulas of f in a way that is compatible with TL semantics. The (negated) subformulas of f are the *closure* of f ($cl(f)$).

- $f \in cl(f)$,
- $f_1 \wedge f_2 \in cl(f) \Rightarrow f_1, f_2 \in cl(f)$,
- $f_1 \vee f_2 \in cl(f) \Rightarrow f_1, f_2 \in cl(f)$,
- $f_1 \supset f_2 \in cl(f) \Rightarrow f_1, f_2 \in cl(f)$,
- $\neg f_1 \in cl(f) \Rightarrow f_1 \in cl(f)$,
- $f_1 \in cl(f) \Rightarrow \neg f_1 \in cl(f)$ if f_1 is not of the form $\neg f_1'$,
- $\bigcirc f_1 \in cl(f) \Rightarrow f_1 \in cl(f)$,
- $\Box f_1 \in cl(f) \Rightarrow f_1 \in cl(f)$,
- $\Diamond f_1 \in cl(f) \Rightarrow f_1 \in cl(f)$,
- $f_1 \, U \, f_2 \in cl(f) \Rightarrow f_1, f_2 \in cl(f)$.

$$cl(\Diamond p) = \{\Diamond p, \neg \Diamond p, p, \neg p\}$$

What are the labeling rules?

For every position $i \geq 0$ the labeling $\tau : \mathbb{N} \rightarrow 2^{cl(f)}$ has to satisfy the following:

- For $p \in P$, $p \in \tau(i)$ iff $p \in \sigma(i)$ and $\neg p \in \tau(i)$ iff $p \notin \sigma(i)$
- $f_1 \in \tau(i)$ if and only if $\neg f_1 \notin \tau(i)$
- $f_1 \wedge f_2 \in \tau(i)$ if and only if $f_1 \in \tau(i)$ and $f_2 \in \tau(i)$.
- $f_1 \vee f_2 \in$ s if and only if $f_1 \in \tau(i)$ or $f_2 \in \tau(i)$.

This takes care of propositions and boolean connectives. What about the temporal operators?

Labeling Rules for Temporal Operators

Even though the temporal operators, have a semantics that looks at the whole sequence, it can (mostly) be reduced to local conditions.

$$\begin{array}{rcl} \Box p & \equiv & (p \wedge \bigcirc \Box p) \\ \Diamond p & \equiv & (p \vee \bigcirc \Diamond p), \\ p \, U \, q & \equiv & (q \vee (p \wedge \bigcirc (p \, U \, q))). \end{array}$$

We thus add the following labeling rules for all positions $i \geq 0$:

- $\bigcirc f_1 \in \tau(i)$ if and only if $f_1 \in \tau(i+1)$.
- $\Box f_1 \in \tau(i)$ if and only if $f_1 \in \tau(i)$ and $\Box f_1 \in \tau(i+1)$.
- $\Diamond f_1 \in \tau(i)$ if and only if either $f_1 \in \tau(i)$, or $\Diamond f_1 \in \tau(i+1)$.
- $f_1 \, U \, f_2 \in \tau(i)$ if and only if either $f_2 \in \tau(i)$, or $f_1 \in \tau(i)$ and $f_1 \, U \, f_2 \in \tau(i+1)$.

There is only one missing condition that concerns formulas called *eventualities*

$$\begin{array}{rcl} \Diamond f_2 & \equiv & (f_2 \vee \bigcirc \Diamond f_2) \\ (f_1 \, U \, f_2) & \equiv & f_1 \wedge (f_2 \vee \bigcirc (f_1 \, U \, f_2)) \end{array}$$

f_2, has to be true at some point!

Interpreting Labelings

A sequence σ satisfies a formula f if one can find a labeling τ satisfying

- The conditions above,
- $f \in \tau(0)$, and
- If an eventuality formula $\diamond f_1$ ($f_1 \, U \, f_2$) is in $\tau(i)$, then for some $j \geq i$, $f_2 \in \tau(j)$.

An automaton on infinite words is just a labeling rule. Actually, we have defined an automaton on infinite words that accepts the models of a temporal logic formula!

The Automaton

It is a generalized Büchi automaton $A_f = (\Sigma, S, \delta, S_0, \mathcal{F})$ where

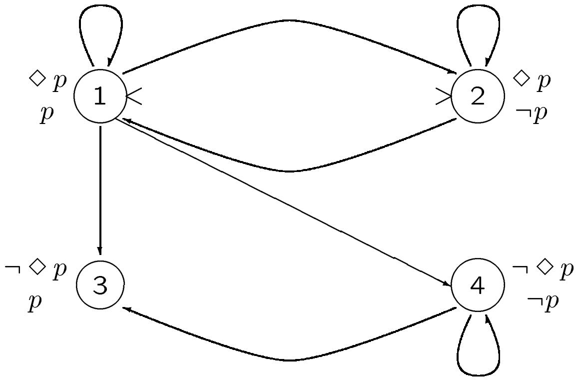
- $\Sigma = 2^P$,
- The set of states S is the set of possible labels, i.e. the elements s of $2^{cl(f)}$ that satisfy
  - $f_1 \in \mathrm{s}$ if and only if $\neg f_1 \notin \mathrm{s}$
  - $f_1 \wedge f_2 \in \mathrm{s}$ if and only if $f_1 \in \mathrm{s}$ and $f_2 \in \mathrm{s}$.
  - $f_1 \vee f_2 \in \mathrm{s}$ if and only if $f_1 \in \mathrm{s}$ or $f_2 \in \mathrm{s}$.

- The transition relation δ checks that the rules for the temporal operators are satisfied and that the propositional labeling matches the one in the sequence being considered. Thus, $\mathbf{t} \in \delta(\mathrm{s}, \mathbf{a})$ iff
  - $\bigcirc f_1 \in \mathrm{s}$ if and only if $f_1 \in \mathbf{t}$.
  - $\Box f_1 \in \mathrm{s}$ if and only if $f_1 \in \mathrm{s}$ and $\Box f_1 \in \mathbf{t}$.
  - $\diamond f_1 \in \mathrm{s}$ if and only if either $f_1 \in \mathrm{s}$, or $\diamond f_1 \in \mathbf{t}$.
  - $f_1 \, U \, f_2 \in \mathrm{s}$ if and only if either $f_2 \in \mathrm{s}$, or $f_1 \in \mathrm{s}$ and $f_1 \, U \, f_2 \in \mathbf{t}$.
  - For all $p \in P$, $p \in \mathbf{a}$ iff $p \in \mathrm{s}$
- The initial states are those that contain f, $S_0 = \{\mathrm{s} \in S \mid f \in \mathrm{s}\}$.

- For the accepting condition, we need to impose that, for every eventuality formula $e(f)$ ($\diamond f$ or $f_1 \, U \, f$) any state that contains that formula is followed by a state that contains f.
  - We have the property that, from every state in which $e(f)$ appears, the transition relation ensures that this formula keeps appearing until the first state in which f appears.
  - It is thus sufficient to require that, for every eventuality $e(f)$, one goes infinitely often either through a state in which it does not appear, or through a state in which the eventuality and f both appear
  - The acceptance condition is thus a generalized Büchi condition. If the eventualities are $e_1(f_1), \ldots e_m(f_m)$, it is $\mathcal{F} = \{F_1, \ldots, F_m\}$ where $F_i = \{\mathrm{s} \in S \mid e_i, f_i \in \mathrm{s} \vee e_i \notin \mathrm{s}\}$.

Example

The automaton for $\diamond p$



$\mathcal{F} = \{\{1, 3, 4\}\}$

Improvements to the Construction

- Build the automaton incrementally from the formula f by decomposing and expanding the temporal operators.
- Put the formula in positive normal form and only include positive formulas in the states (this allows missing formulas to be considered as "don't cares").
- Remove propositions from states and only include them in transitions.

Application: Model Checking

Verify that all the computations of a finite-state program P satisfy a temporal logic specification f

- Compute the global state graph of P
- Build the automaton for $\neg f$: $A_{\neg f}$
- Compute the product $P \times A_{\neg f}$
- Check that the product is empty (compute strongly connected components)

Complexity

- Linear in the size of the program (NLOGSPACE)
- Exponential in the size of the temporal specification (PSPACE)

How about Branching Time?

- Use alternating automata