The Algorithmic Verification of Reactive Systems

Lecture 4:

Temporal Logics and their Relation to Automata

A Language for Writing Specifications: Temporal Logic

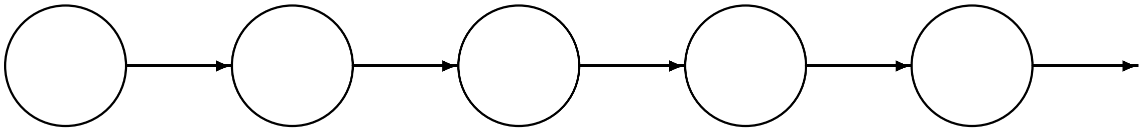
- Language for describing properties of infinite sequences,

- Extension of propositional logic (or first-order logic),

- Uses *Temporal Operators* to describe temporal (sequencing) properties.

Temporal Logic: Interpretations

- Temporal logic is interpreted on sequences of states:



- each state in the sequence gives a truth value to atomic propositions,
- a formula is given a meaning in a state (world, interpretation) of a sequence,
- *temporal operators* indicates in which states the formula (or parts thereof) should be interpreted.

Temporal Operators

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)

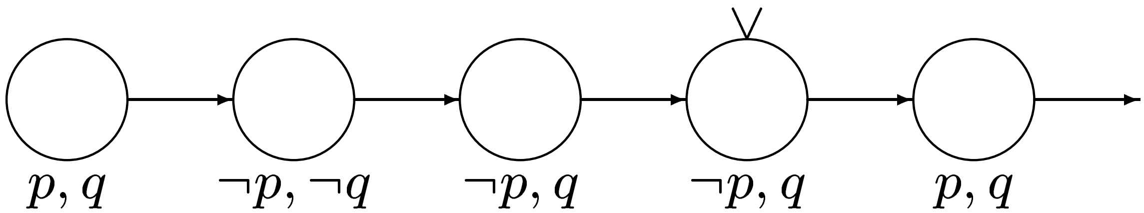
3. $\Diamond$ which is read "eventually" (in some future state of the sequence)

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

5. $\tilde{U}$ which is read "releases" (binary operator)

Examples

In the marked state of the sequence



the following formulas are *true*:

$$\neg p \wedge q$$

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

the following formula is *false*:

$$\Box\, p$$

More Examples

- $\Box(p \supset \bigcirc q)$ is satisfied by all sequences in which each state where p is true is immediately followed by a state in which q is true.

- $\Box(p \supset \bigcirc(\neg q\, U\, r))$ is satisfied by all sequences such that if p is true in a state, then q remains false from the next state on and until the first state where r is true, which must occur.

- The formula $\Box \Diamond p$ is satisfied by all sequences in which p is true infinitely often in the future

- The formula $p\, \tilde{U}\, q$ is satisfied by all sequences in which q is always true unless this obligation is *released* by p being true in a previous state.

Formal Syntax of Temporal Logic

The formulas of linear-time temporal logic (LTL) built from a set of atomic propositions P are the following.

- $\mathbf{true}$, $\mathbf{false}$, p, or $\neg p$, for all $p \in P$;
- $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are LTL formulas;
- $\bigcirc \varphi_1$, $\varphi_1 \, U \, \varphi_2$, or $\varphi_1 \, \tilde{U} \, \varphi_2$, where φ_1 and φ_2 are LTL formulas.

Two operators are then defined as abbreviations:

- $\Diamond \varphi = \mathbf{true} \, U \, \varphi$ ("eventually").
- $\Box \varphi = \mathbf{false} \, \tilde{U} \, \varphi$ ("always").

Semantics of Temporal Logic

The semantics of LTL is defined with respect to paths $\pi : \mathbf{N} \to 2^P$. For a path π, π^i represents the suffix of π obtained by removing its i first states, i.e. $\pi^i(j) = \pi(i+j)$. The rules given the truth of a formula in the first state of a path π are the following.

- For all π, we have $\pi \models \mathbf{true}$ and $\pi \not\models \mathbf{false}$.
- $\pi \models p$ for $p \in P$ iff $p \in \pi(0)$.
- $\pi \models \neg p$ for $p \in P$ iff $p \not\in \pi(0)$.
- $\pi \models \varphi_1 \wedge \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$.
- $\pi \models \varphi_1 \vee \varphi_2$ iff $\pi \models \varphi_1$ or $\pi \models \varphi_2$.
- $\pi \models \bigcirc\, \varphi_1$ iff $\pi^1 \models \varphi_1$.
- $\pi \models \varphi_1\, U\, \varphi_2$ iff there exists $i \geq 0$ such that $\pi^i \models \varphi_2$ and for all $0 \leq j < i$, we have $\pi^j \models \varphi_1$.
- $\pi \models \varphi_1\, \tilde{U}\, \varphi_2$ iff for all $i \geq 0$ such that $\pi^i \not\models \varphi_2$, there exists $0 \leq j < i$ such that $\pi^j \models \varphi_1$.

In the logic we have defined, negation is only applied to atomic propositions. This restriction can be lifted with the help of the following relations

$$\pi \not\models \varphi_1 \, U \, \varphi_2 \text{ iff } \pi \models (\neg\varphi_1) \, \tilde{U} (\neg\varphi_2)$$

$$\pi \not\models \varphi_1 \, \tilde{U} \, \varphi_2 \text{ iff } \pi \models (\neg\varphi_1) \, U (\neg\varphi_2)$$

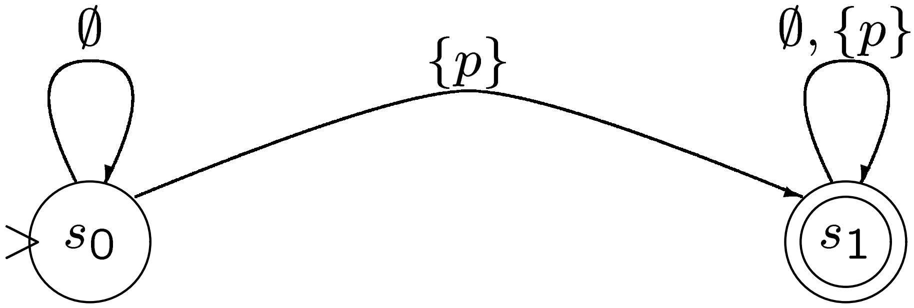
$$\pi \not\models \bigcirc \varphi_1 \text{ iff } \pi \models \bigcirc \neg\varphi_1$$

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

From LTL to Automata

Given a LTL formula φ 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 φ.

Example:



An automaton for $\Diamond\, p$

Idea of the Construction

To determine if a sequence $\pi : \mathbf{N} \rightarrow 2^P$ satisfies a formula φ, one can proceed by labeling it with subformulas of φ in a way that is compatible with LTL semantics. The subformulas of φ are the *closure* of φ ($cl(\varphi)$).

- $\varphi \in cl(\varphi)$,
- $\varphi_1 \wedge \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$,
- $\varphi_1 \vee \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$,
- $\bigcirc \varphi_1 \in cl(\varphi) \Rightarrow \varphi_1 \in cl(\varphi)$,
- $\varphi_1 \, U \, \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$.
- $\varphi_1 \, \tilde{U} \, \varphi_2 \in cl(\varphi) \Rightarrow \varphi_1, \varphi_2 \in cl(\varphi)$.

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

What are the labeling rules?

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

- For $p \in P$, if $p \in \tau(i)$ then $p \in \pi(i)$, and if $\neg p \in \tau(i)$ then $p \notin \pi(i)$

- if $\varphi_1 \wedge \varphi_2 \in \tau(i)$ then $\varphi_1 \in \tau(i)$ and $\varphi_2 \in \tau(i)$.

- if $\varphi_1 \vee \varphi_2 \in \tau(i)$ then $\varphi_1 \in \tau(i)$ or $\varphi_2 \in \tau(i)$.

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.

$$\begin{aligned} p\, U\, q &\equiv (q \vee (p \wedge \bigcirc(p\, U\, q))) \\ p\, \tilde{U}\, q &\equiv (q \wedge (p \vee \bigcirc(p\, \tilde{U}\, q))) \end{aligned}$$

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

- if $\bigcirc \varphi_1 \in \tau(i)$ then $\varphi_1 \in \tau(i+1)$.
- if $\varphi_1\, U\, \varphi_2 \in \tau(i)$ then either $\varphi_2 \in \tau(i)$, or $\varphi_1 \in \tau(i)$ and $\varphi_1\, U\, \varphi_2 \in \tau(i+1)$.
- if $\varphi_1\, \tilde{U}\, \varphi_2 \in \tau(i)$ then $\varphi_2 \in \tau(i)$, and either $\varphi_1 \in \tau(i)$ or $\varphi_1\, \tilde{U}\, \varphi_2 \in \tau(i+1)$

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

$$(\varphi_1 \, U \, \varphi_2) \;\equiv\; \varphi_1 \wedge (\varphi_2 \vee \bigcirc(\varphi_1 \, U \, \varphi_2))$$

φ_2, has to be true at some point!

Interpreting Labelings

A sequence π satisfies a formula φ if one can find a labeling τ satisfying

- The conditions above,
- $\varphi \in \tau(0)$, and
- If an eventuality formula $\varphi_1 \, U \, \varphi_2$ is in $\tau(i)$, then for some $j \geq i$, $\varphi_2 \in \tau(j)$.

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
$A_\varphi = (\Sigma, S, \delta, S_0, \mathcal{F})$ where

- $\Sigma = 2^P$,
- The set of states S is the set of possible labels, i.e. the subsets s of $2^{cl(\varphi)}$ that satisfy
  - if $\varphi_1 \wedge \varphi_2 \in \mathrm{s}$ then $\varphi_1 \in \mathrm{s}$ and $\varphi_2 \in \mathrm{s}$.
  - if $\varphi_1 \vee \varphi_2 \in \mathrm{s}$ then $\varphi_1 \in \mathrm{s}$ or $\varphi_2 \in \mathrm{s}$.

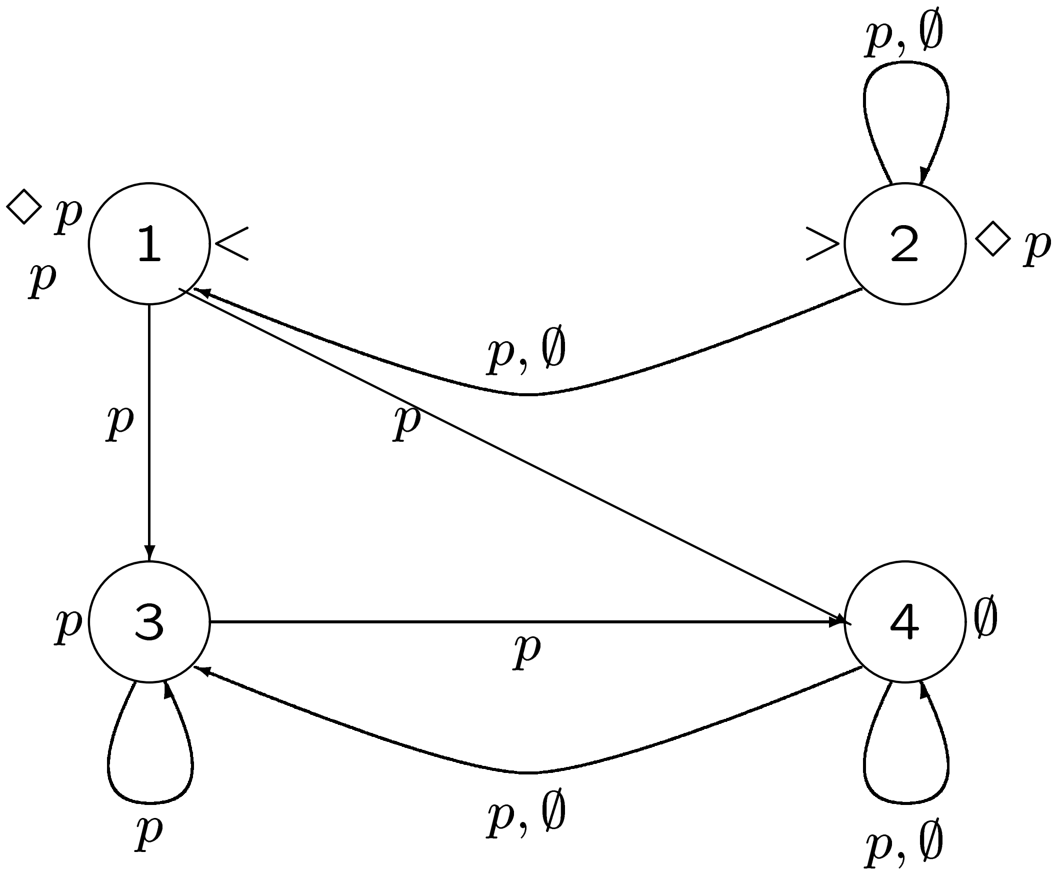
- The transition function δ checks that the propositional labeling matches the one in the sequence being considered and that the rules for the temporal operators are satisfied. Thus, $\mathbf{t} \in \delta(\mathbf{s}, \mathbf{a})$ iff
  - For all $p \in P$, if $p \in \mathbf{s}$ then $p \in \mathbf{a}$.
  - For all $p \in P$, if $\neg p \in \mathbf{s}$ then $p \notin \mathbf{a}$.
  - if $\bigcirc \varphi_1 \in \mathbf{s}$ then $\varphi_1 \in \mathbf{t}$.
  - if $\varphi_1 \, U \, \varphi_2 \in \mathbf{s}$ then either $\varphi_2 \in \mathbf{s}$, or $\varphi_1 \in \mathbf{s}$ and $\varphi_1 \, U \, \varphi_2 \in \mathbf{t}$.
  - if $\varphi_1 \, \tilde{U} \, \varphi_2 \in \mathbf{s}$ then $\varphi_2 \in \mathbf{s}$ and either $\varphi_1 \in \mathbf{s}$, or $\varphi_1 \, \tilde{U} \, \varphi_2 \in \mathbf{t}$.
- The initial states are those that contain φ, $S_0 = \{\mathbf{s} \in S \mid \varphi \in \mathbf{s}\}$.

- For the accepting condition, we need to impose that, for every eventuality formula $\varphi_1 \, U \, \varphi' \equiv e(\varphi') \in cl(\varphi)$, any state that contains that formula is followed by a state that contains φ'.

  - We have the property that, from every state in which $e(\varphi')$ appears, the transition relation ensures that this formula keeps appearing until the first state in which φ' appears.
  - It is thus sufficient to require that, for every eventuality $e(\varphi')$, one goes infinitely often either through a state in which it does not appear, or through a state in which the eventuality and φ' both appear.
  - The acceptance condition is thus a generalized Büchi condition. If the eventualities are $e_1(\varphi_1), \ldots e_m(\varphi_m)$, it is $\mathcal{F} = \{\Phi_1, \ldots, \Phi_m\}$ where $\Phi_i = \{\mathbf{s} \in S \mid e_i, \varphi_i \in \mathbf{s} \vee e_i \notin \mathbf{s}\}$.

Example

The automaton for $\diamond p$



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

Note: Transitions leading to states with more (nonatomic) elements of the closure than necessary were not included.

Improvements to the Construction

- Build the automaton incrementally from the formula φ by decomposing and expanding the temporal operators.
- Merge states which only differ by atomic propositions since these are determined by the labels of the transitions.
- When possible, detect inconsistent nodes without fully expanding to the propositional level.

Branching Time Temporal Logic

Branching time temporal logic are interpreted over *infinite trees* (varying degree) in which each node is a state assigning truth values to the atomic propositions.

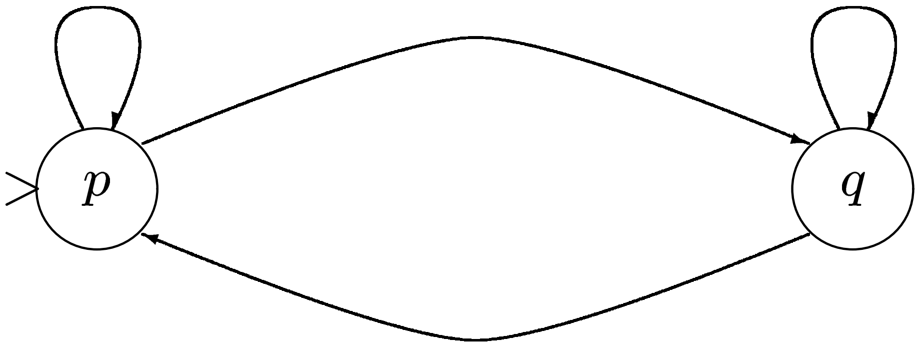
More generally, branching time temporal logic is interpreted over Kripke structures $K = \langle P, W, R, w^0, L \rangle$, where

- P is a set of atomic propositions,
- W is a set of states,
- $R \subseteq W \times W$ is a transition relation that must be total (i.e., for every $w \in W$ there exists $w' \in W$ such that $\langle w, w' \rangle \in R$),
- w^0 is an initial state, and
- $L : W \to 2^P$ maps each state to the set of atomic propositions true in that state.

From a state of a Kripke structure leave a number of *paths*, i.e. infinite sequence of states, $\pi = w_0, w_1, \ldots$ such that for every $i \geq 0$, $\langle w_i, w_{i+1} \rangle \in R$.

Note that by identifying $\pi(i)$ with $L(w_i)$, a path in a Kripke structure defines an LTL interpretation.

Example



Path Quantifiers

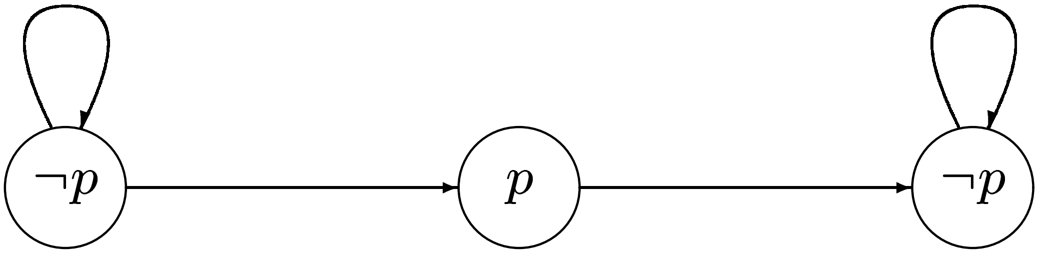
Branching-time temporal logic is linear-time temporal logic extended with *path quantifiers*:

- A for all paths,
- E for some paths.

Path quantifiers are applied to linear-time formulas which then become *state formulas* and can be used as atoms in a new path formula.

Examples

$A \Box p$, $E \Box p$, $A(p \supset \bigcirc E \Box p)$, $A \Box E \bigcirc p$, $A(\Box \Diamond p)$, $E(\Box \Diamond p)$, $E \Box E \Diamond p$.



Formal Syntax of Branching-time Temporal Logic

Syntax

We first define the more general logic CTL*.
A CTL* state formula is either:

- $\mathbf{true}$, $\mathbf{false}$, p, or $\neg p$, for all $p \in P$;
- $\varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$, where φ_1 and φ_2 are CTL* state formulas;
- $A\psi$ or $E\psi$, where ψ is a CTL* path formula.

A CTL* path formula is either:

- A CTL* state formula;
- $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\bigcirc \psi_1$, $\psi_1 U \psi_2$, or $\psi_1 \tilde{U} \psi_2$, where ψ_1 and ψ_2 are CTL* path formulas.

CTL* is the set of state formulas generated by the above rules.

A restricted form of Branching-time Temporal Logic

The logic *CTL* is a restricted subset of CTL* in which the temporal operators must be immediately preceded by a path quantifier.

Formally, it is the subset of CTL* obtained by defining the path formulas as follows.

A CTL path formula is of the form

- $\bigcirc \varphi_1$, $\varphi_1 \, U \, \varphi_2$, or $\varphi_1 \, \tilde{U} \, \varphi_2$, where φ_1 and φ_2 are CTL state formulas.

Examples

- CTL formulas: $A \Box p$, $E \Box p$, $A \Box E \bigcirc p$, $E \Box E \Diamond p$.
- CTL* formulas: $A(p \supset \bigcirc E \Box p)$, $A(\Box \Diamond p)$, $E(\Box \Diamond p)$.

Semantics

Given a Kripke structure $K = \langle P, W, R, w^0, L \rangle$, the truth of a CTL* formula in a state w is defined as follows.

- For all w, we have $w \models \mathbf{true}$ and $w \not\models \mathbf{false}$.
- $w \models p$ for $p \in P$ iff $p \in L(w)$.
- $w \models \neg p$ for $p \in P$ iff $p \notin L(w)$.
- $w \models \varphi_1 \wedge \varphi_2$ iff $w \models \varphi_1$ and $w \models \varphi_2$.
- $w \models \varphi_1 \vee \varphi_2$ iff $w \models \varphi_1$ or $w \models \varphi_2$.
- $w \models A\psi$ iff for every path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, we have $\pi \models \psi$.
- $w \models E\psi$ iff there exists a path $\pi = w_0, w_1, \ldots$, with $w_0 = w$, such that $\pi \models \psi$.

- $\pi \models \varphi$ for a state formula φ, iff $w_0 \models \varphi$ where $\pi = w_0, w_1, \ldots$
- $\pi \models \psi_1 \wedge \psi_2$ iff $\pi \models \psi_1$ and $\pi \models \psi_2$.
- $\pi \models \psi_1 \vee \psi_2$ iff $\pi \models \psi_1$ or $\pi \models \psi_2$.
- $\pi \models \bigcirc \psi$ iff $\pi^1 \models \psi$.
- $\pi \models \psi_1 \, U \, \psi_2$ iff there exists $i \geq 0$ such that $\pi^i \models \psi_2$ and for all $0 \leq j < i$, we have $\pi^j \models \psi_1$.
- $\pi \models \psi_1 \, \tilde{U} \, \psi_2$ iff for all $i \geq 0$ such that $\pi^i \not\models \psi_2$, there exists $0 \leq j < i$ such that $\pi^j \models \psi_1$.

From CTL to Alternating Tree Automata

As for linear-time the idea is to label the Kripke structure $K = \langle P, W, R, w^0, L\rangle$ with formulas according to the semantic rules. For a CTL formula φ, the labeling function is of the form $\tau : W \rightarrow 2^{cl(\varphi)}$ where $cl(\varphi)$ represents the set of subformulas of φ. The labeling has to satisfy the following rules for all $w \in W$.

- For $p \in P$, if $p \in \tau(w)$ then $p \in L(w)$, and if $\neg p \in \tau(w)$ then $p \notin L(w)$
- if $\varphi_1 \wedge \varphi_2 \in \tau(w)$ then $\varphi_1 \in \tau(w)$ and $\varphi_2 \in \tau(w)$.
- if $\varphi_1 \vee \varphi_2 \in \tau(w)$ then $\varphi_1 \in \tau(w)$ or $\varphi_2 \in \tau(w)$.

These are the rules for atomic propositions and boolean connectives. The temporal connectives are handled with the help of the following properties.

$$\begin{array}{rcl} A(p\,U\,q) & \equiv & (q \vee (p \wedge A\bigcirc A(p\,U\,q))) \\ E(p\,U\,q) & \equiv & (q \vee (p \wedge E\bigcirc E(p\,U\,q))) \\ A(p\,\tilde{U}\,q) & \equiv & (q \wedge (p \vee A\bigcirc A(p\,\tilde{U}\,q))) \\ E(p\,\tilde{U}\,q) & \equiv & (q \wedge (p \vee E\bigcirc E(p\,\tilde{U}\,q))) \end{array}$$

The rules are thus the following.

- if $A\bigcirc\varphi_1 \in \tau(w)$ then for all w' such that $(w, w') \in R$, $\varphi_1 \in \tau(w')$.
- if $E\bigcirc\varphi_1 \in \tau(w)$ then for some w' such that $(w, w') \in R$, $\varphi_1 \in \tau(w')$.
- if $A(\varphi_1\,U\,\varphi_2) \in \tau(w)$ then either $\varphi_2 \in \tau(w)$, or $\varphi_1 \in \tau(w)$ and for all w' such that $(w, w') \in R$ $A(\varphi_1\,U\,\varphi_2) \in \tau(w')$.
- if $E(\varphi_1\,U\,\varphi_2) \in \tau(w)$ then either $\varphi_2 \in \tau(w)$, or $\varphi_1 \in \tau(w)$ and for some w' such that $(w, w') \in R$ $E(\varphi_1\,U\,\varphi_2) \in \tau(w')$.
- if $A(\varphi_1\,\tilde{U}\,\varphi_2) \in \tau(w)$ then $\varphi_2 \in \tau(w)$, and either $\varphi_1 \in \tau(w)$ or for all w' such that $(w, w') \in R$ $A(\varphi_1\,\tilde{U}\,\varphi_2) \in \tau(w')$
- if $E(\varphi_1\,\tilde{U}\,\varphi_2) \in \tau(w)$ then $\varphi_2 \in \tau(w)$, and either $\varphi_1 \in \tau(w)$ or for some w' such that $(w, w') \in R$ $E(\varphi_1\,\tilde{U}\,\varphi_2) \in \tau(w')$

The only remaining condition is the one imposed by eventualities:

$$A(\varphi_1 \, U \, \varphi_2)$$
$$E(\varphi_1 \, U \, \varphi_2)$$

In the first case φ_2 has to occur on all paths from the state labeled by the eventuality; in the second case, it has to occur on some path on which φ_1 hold until then.

The Automaton for a CTL Formula

Automata on trees with degrees in a set D

We want to recognize the Kripke structures satisfying a formula φ. The automaton will thus operate on the corresponding infinite tree, but this tree has a varying branching degree. We thus adapt the definition of tree automata to cope with a set of possible branching degrees.

$\delta : S \times \Sigma \times \mathcal{D} \rightarrow \mathcal{B}^{+}(\mathrm{N} \times S)$ with $\delta(s, a, k) \in \mathcal{B}^{+}(\{0, \ldots, k-1\} \times S)$ for every $k \in \mathcal{D}$.

The automaton is a Büchi automaton on trees with varying degree $\mathcal{A}_{\mathcal{D},\varphi} = \langle \Sigma, \mathcal{D}, S, \delta, S_0, F \rangle$, where

- $\Sigma = 2^P$,
- The set of states S is $cl(\varphi)$ (this is sufficient since alternating automata allow the multiple labeling of nodes)
- $S_0 = \{\varphi\}$
- F consists of all $A\tilde{U}$ and $E\tilde{U}$ formulas in $cl(\varphi)$.
- For all $\mathbf{a} \in 2^P$ and $k \in \mathcal{D}$, δ is defined as follows.

- $\delta(p, \mathbf{a}, k) = \mathbf{true}$ if $p \in \mathbf{a}$.
- $\delta(p, \mathbf{a}, k) = \mathbf{false}$ if $p \notin \mathbf{a}$.
- $\delta(\neg p, \mathbf{a}, k) = \mathbf{true}$ if $p \notin \mathbf{a}$.
- $\delta(\neg p, \mathbf{a}, k) = \mathbf{false}$ if $p \in \mathbf{a}$.
- $\delta(\varphi_1 \wedge \varphi_2, \mathbf{a}, k) = \delta(\varphi_1, \mathbf{a}, k) \wedge \delta(\varphi_2, \mathbf{a}, k)$.
- $\delta(\varphi_1 \vee \varphi_2, \mathbf{a}, k) = \delta(\varphi_1, \mathbf{a}, k) \vee \delta(\varphi_2, \mathbf{a}, k)$.
- $\delta(A \bigcirc \varphi, \mathbf{a}, k) = \bigwedge_{c=0}^{k-1}(c, \varphi)$.
- $\delta(E \bigcirc \varphi, \mathbf{a}, k) = \bigvee_{c=0}^{k-1}(c, \varphi)$.
- $\delta(A(\varphi_1 \, U \, \varphi_2), \mathbf{a}, k) = \delta(\varphi_2, \mathbf{a}, k) \vee$
  $(\delta(\varphi_1, \mathbf{a}, k) \wedge \bigwedge_{c=0}^{k-1}(c, A(\varphi_1 \, U \, \varphi_2)))$.
- $\delta(E(\varphi_1 \, U \, \varphi_2), \mathbf{a}, k) = \delta(\varphi_2, \mathbf{a}, k) \vee$
  $(\delta(\varphi_1, \mathbf{a}, k) \wedge \bigvee_{c=0}^{k-1}(c, E(\varphi_1 \, U \, \varphi_2)))$.
- $\delta(A(\varphi_1 \, \tilde{U} \, \varphi_2), \mathbf{a}, k) = \delta(\varphi_2, \mathbf{a}, k) \wedge$
  $(\delta(\varphi_1, \mathbf{a}, k) \vee \bigwedge_{c=0}^{k-1}(c, A(\varphi_1 \, \tilde{U} \, \varphi_2)))$.
- $\delta(E(\varphi_1 \, \tilde{U} \, \varphi_2), \mathbf{a}, k) = \delta(\varphi_2, \mathbf{a}, k) \wedge$
  $(\delta(\varphi_1, \mathbf{a}, k) \vee \bigvee_{c=0}^{k-1}(c, E(\varphi_1 \, \tilde{U} \, \varphi_2)))$.

Example

$\varphi = A \Diamond A \Box p$.
$\mathcal{A}_{\mathcal{D},\varphi} = \langle \{\{p\}, \emptyset\}, \mathcal{D}, \{\varphi, A \Box p, p\}, \delta, \varphi, \{A \Box p\} \rangle$,
with δ given by

q	$\delta(q, \{p\}, k)$	$\delta(q, \emptyset, k)$
p	$\mathbf{true}$	$\mathbf{false}$
$A \Box p$	$\bigwedge_{c=0}^{k-1}(c, A \Box p)$	$\mathbf{false}$
φ	$\bigwedge_{c=0}^{k-1}(c, A \Box p) \vee \bigwedge_{c=0}^{k-1}(c, \varphi)$	$\bigwedge_{c=0}^{k-1}(c, \varphi)$

Conclusions

- Temporal logics can be translated into automata.
  - The size of the automaton for a formula φ is exponential in $|\varphi|$ for the nondeterministic automaton build for a LTL formula.
  - The size of the automaton for a formula φ is linear in $|\varphi|$ for the alternating automaton build for a CTL. formula.
- This can serve as the basis of algorithms for verifying properties expressed in a temporal logic.