Automata and Logic

- There is an intimate connection between automata and logic
- Logic
  - a temporal logic formula $\varphi$ is identified with all models that satisfy it
- Automata
  - a language of an automaton is the set of all words accepted by it
- The language of an automaton for a logical formula $\varphi$ is the set of all models that satisfy $\varphi$
  - strings for linear logic
  - trees for branching logic
Automata-Theoretic Approach

- Automata-theoretic approach gives a uniform solution to both satisfiability and model-checking.

- For a given logical formula $\varphi$ and a model $K$:
  - $\varphi$ is satisfiable iff there exists a model that satisfies $\varphi$
  - $\square p$ is satisfiable
  - $\square (p \land \neg p)$ is not
  - model-checking is deciding if $\varphi$ is satisfied by a given model

- Automata-theoretic solution
  - build an automaton $A_\varphi$ for the formula $\varphi$
  - $\varphi$ is satisfiable iff $A_\varphi$ is non-empty
  - combine $A_{\neg \varphi}$ and $K$ into an automaton $A_{\neg \varphi, K}$
  - $K \models \varphi$ iff $A_{\neg \varphi, K}$ is empty

Automata-Theoretic Approach

- Automata provide a clean separation between logic and algorithms
- Constructing an automaton for a formula
  - what does that mean for a model to satisfy the formula
- Solving non-emptiness problem for an automaton
  - how to decide if a given model satisfies the formula
Outline

- Automata on infinite words
  - refresher
  - acceptance conditions
  - computational tree of an automaton
  - alternation – a powerful extension of nondeterminism
- Constructing an Alternating Word Automaton for LTL
- Automata on infinite trees
  - deterministic automata
  - nondeterministic automata
  - alternating automata
- Constructing an Alternating Tree Automaton for CTL

Finite-state Automata

- A finite state automaton $A$ is a tuple $(\Sigma, Q, \delta, q_0, \mathcal{F})$, where
  - $\Sigma$ is a finite alphabet
  - $Q$ is a finite set of states
  - $\delta \subseteq Q \times \Sigma \times Q$ is a transition relation
  - $q_0 \in Q$ is a designated initial state
  - $\mathcal{F} \subseteq Q^\omega$ is an acceptance condition
- A $D$-labeled infinite string $s$ is a function $\mathbb{N} \rightarrow D$
- A $\Sigma$-word $w$ is a $\Sigma$-labeled infinite string
  - $w = ababaa\omega$
  - $w(0) = a$, $w(1) = b$, $w(3) = a$, etc.
Finite-state Automata

- A run $r$ of an automaton over a word $w$ is a $N \times Q$ labeled string, where
  - a node of $r$ labeled with $(n, q)$ indicates that the automaton reads letter $n$ of $w$ while at state $q$

  $b, c \rightarrow a $  
  $q_0 \rightarrow b \leftarrow a \rightarrow q_1$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, a)$</th>
<th>$\delta(q, b)$</th>
<th>$\delta(q, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_1$</td>
<td>$q_0$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_0$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

- a run on $w = ab\alpha$ is

$$(0, q_0), (1, q_1), (2, q_0), (3, q_1), (4, q_1), (5, q_1), \ldots$$

Infinite Occurrences

- $\exists^\omega i \cdot Y(i)$ – there exists infinitely many $i$th such that $Y(i)$

- For $\rho \in Q^\omega$
  - $In(\rho)$ is the set of states that occur infinitely often
  - $In(\rho) = \{q \in Q \mid \exists^\omega i \cdot \rho(i) = q\}$

  Büchi condition
  - $F$ is $F \subseteq Q$
  - $In(w) \cap F \neq \emptyset$
  - weak fairness – something occurs infinitely often

  Muller condition
  - $F$ is $\{F_1, \ldots, F_n\} \subseteq 2^Q$
  - $\exists i \cdot In(w) = F_i$
Acceptance Conditions

- Rabin condition ("pairs")
  - \( \mathcal{F} \) is \( \{( R_1, G_1), \ldots (R_n, G_n)\} \) with \( R_i, G_i \subseteq Q \)
  - \( \exists i \cdot \text{In}(w) \cap R_i = \emptyset \land \text{In}(w) \cap G_i \neq \emptyset \)
  - Rabin \((\emptyset, F)\) is equivalent to Büchi \( F \)

- Street condition ("complemented pairs")
  - \( \mathcal{F} \) is \( \{( F_1, E_1), \ldots , (F_n, E_n)\} \) with \( E_i, F_i \subseteq Q \)
  - \( \forall i \cdot \text{In}(w) \cap F_i \neq \emptyset \Rightarrow \text{In}(w) \cap E_i \neq \emptyset \)
  - strong fairness
    - if infinitely often enabled, then infinitely often executed
  - Street \((Q, F)\) is equivalent to Büchi \( F \)

Acceptance Conditions

- Parity condition
  - \( \mathcal{F} \) is \( F_1 \subseteq \cdots \subseteq F_n \) with \( F_i \subseteq Q \)
  - smallest \( i \) for which \( \text{In}(w) \cap F_i \neq \emptyset \) is even

- co-Büchi condition
  - \( \mathcal{F} \) is \( F \subseteq Q \)
  - accepts \( w \) if \( \text{In}(w) \cap F = \emptyset \)

- Nondeterministic Büchi-, Muller-, Rabin-, and Street-automata all recognize the same \( \omega \)-languages
Example: Acceptance

- Language over \( \{a, b, c\}^\omega \)
  - if \( a \) occurs infinitely often, then so does \( b \)
- Automaton with states \( q_a, q_b, \) and \( q_c, \) and \( \delta \)

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
<th>( \delta(q, b) )</th>
<th>( \delta(q, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_a )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
<tr>
<td>( q_b )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
<tr>
<td>( q_c )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
</tbody>
</table>

- Acceptance conditions
  - Street – single pair \( \{q_a\}, \{q_b\} \)
  - Muller – all states \( F \) where \( q_a \in F \Rightarrow q_b \in F \)
    \( \{q_b\}, \{q_c\}, \{q_b, q_c\}, \{q_a, q_b\}, \{q_a, q_b, q_c\} \)

Example: Acceptance

- Automaton with states \( q_a, q_b, \) and \( q_c, \) and \( \delta \)

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
<th>( \delta(q, b) )</th>
<th>( \delta(q, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_a )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
<tr>
<td>( q_b )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
<tr>
<td>( q_c )</td>
<td>( q_a )</td>
<td>( q_b )</td>
<td>( q_c )</td>
</tr>
</tbody>
</table>

- Acceptance conditions
  - Rabin
    - either \( b \) occurs infinitely often, or both \( a \) and \( b \) have finite occurrences
    - two pairs \( (\emptyset, \{q_b\}), (\{q_a, q_b\}, \{q_c\}) \)
  - Parity
    - \( \emptyset, \{q_b\}, \{q_a, q_b\}, \{q_a, q_b, q_c\} \)
### Example: Acceptance

- For Büchi acceptance condition simulate Rabin pairs by nondeterminism

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, a)$</th>
<th>$\delta(q, b)$</th>
<th>$\delta(q, c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_a$</td>
<td>$q_a$</td>
<td>$q_b$</td>
<td>${q_c, q'}$</td>
</tr>
<tr>
<td>$q_b$</td>
<td>$q_a$</td>
<td>$q_b$</td>
<td>${q_c, q'}$</td>
</tr>
<tr>
<td>$q_c$</td>
<td>$q_a$</td>
<td>$q_b$</td>
<td>${q_c, q'}$</td>
</tr>
<tr>
<td>$q'$</td>
<td>$q_a$</td>
<td>$q_b$</td>
<td>$q'$</td>
</tr>
</tbody>
</table>

- Every time $c$ occurs, guess that a suffix containing only $c$ is reached
- Büchi acceptance condition
  - $F = \{q_b, q'\}$

### Computational Tree of an Automaton

- A set of all runs of an automaton $A$ over a fixed word $w$ is called a computational tree
- Each node in the computational tree is labeled by a history $h \in Q^*$
- A history is a list of all states visited by the automaton so far
- For a deterministic automaton, the computational tree is linear
  - there is only one possible run!
Computational Tree: Example

- A deterministic automaton over \{a, b\}

<table>
<thead>
<tr>
<th>state</th>
<th>(\delta(q, a))</th>
<th>(\delta(q, b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_0)</td>
<td>(q_0)</td>
<td>(q_1)</td>
</tr>
<tr>
<td>(q_1)</td>
<td>(q_0)</td>
<td>(q_1)</td>
</tr>
</tbody>
</table>

- Computational tree over \((aab)^\omega\)

\[(q_0), (q_0, q_0); (q_0, q_0, q_0), (q_0, q_0, q_0, q_1), \ldots\]

Comp. Tree: Nondeterministic Case

- For a nondeterministic automaton, the computational tree contains all possible choices.

- Formally, the computational tree \(T\) of \(A\) over \(w\) is recursively defined as:
  - the root is labeled by \(q_0\),
  - for a node \(k \in T\) labeled with the history \(x \cdot y\), where \(x \in Q^*,\) and \(y \in Q\)
  - if \(\delta(y, w(|x \cdot y|)) = \{t_1, \ldots, t_n\}\), then
  - \(k\) has \(n\) successors, and
  - \(i\)th successor is labeled with \(x \cdot y \cdot t_i\)
Example: nondeterministic automaton

- Nondeterministic automaton over \( \{a, b\} \)

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
<th>( \delta(q, b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>{( q_0, q_1 )}</td>
<td>( q_0 )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_1 )</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_2 )</td>
<td>( q_2 )</td>
</tr>
</tbody>
</table>

- acceptance condition is Büchi \( F = \{q_1\} \)
- corresponds to \( \Diamond \Box a \)
- Computational tree over \( aba^\omega \)

Computational Tree: Acceptance

- An infinite history \( h \in Q^\omega \) corresponds to an infinite branch \( \beta \) of the computational tree iff for any prefix of \( h \) there exists a node in \( \beta \) labeled with it
- An automaton \( A \) accepts a word \( w \) iff
  - there exists an infinite branch \( \beta \) in the computational tree of \( A \) over \( w \), such that
  - an infinite history corresponding to \( \beta \) is an accepting run
Alternation

For a non-deterministic automaton $A$, a transition $\delta(q, a) = \{t_1, \ldots, t_n\}$ can be interpreted as

- when $A$ is in state $q$ and has read letter $a$
  - create $n$ copies of $A$
  - switch $i$th copy to state $t_i$
  - run each copy on the rest of the word
- a word is accepted iff it is accepted by at least one copy

We can dualize the acceptance condition to be

- a word is accepted iff it is accepted by all copies

In this case, the computational tree is linear

- but, each node is labeled with multiple histories

Example of a Dual Automaton

Automaton over $\{a, b\}$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, a)$</th>
<th>$\delta(q, b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>${q_0, q_1}$</td>
<td>$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q_1$</td>
<td>$q_2$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_2$</td>
<td>$q_2$</td>
</tr>
</tbody>
</table>

- just as before but $\{q_0, q_1\}$ means pick both, not pick one!
- acceptance condition is Büchi $F = \{q_1\}$
- Computational tree over $aba^\omega$ is linear
Another Example of a Dual Automaton

- Automaton over \( \{a, b, c\} \)

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
<th>( \delta(q, b) )</th>
<th>( \delta(q, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( {q_1, q_2} )</td>
<td>( {q_1, q_2} )</td>
<td>( {q_1, q_2} )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( q_3 )</td>
<td>( q_1 )</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_2 )</td>
<td>( q_3 )</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_3 )</td>
<td>( q_3 )</td>
<td>( q_3 )</td>
</tr>
</tbody>
</table>

- acceptance condition is Büchi \( F = \{q_3\} \)
- accepts \( \Diamond((\Diamond a) \land (\Diamond b)) \)
- Computational tree over \( ccabc^\omega \) is linear

Alternating Automata

- Alternating automata combine the two interpretations
- the transition relation becomes \( Q \times \Sigma \rightarrow 2^{2^Q} \)
- \( \delta(q, a) = \{T_1, \ldots, T_n\} \) is interpreted as
  - when \( a \) is read at state \( q \), pick one of \( T_i \subseteq Q \)
  - create as many copies of \( A \) as \( |T_i| \), and send them along the word
  - a word is accepted iff it is accepted by all the copies
- A computational tree of an alternating automaton is
  - branching
  - each node can be labeled with multiple histories
Alternating Automata: Example

Example alternating automaton over \{a, b, c\}

<table>
<thead>
<tr>
<th>state</th>
<th>(\delta(q, a))</th>
<th>(\delta(q, b))</th>
<th>(\delta(q, c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_0)</td>
<td>({q_0}, {q_1})</td>
<td>(q_2)</td>
<td>({q_1, q_2})</td>
</tr>
<tr>
<td>(q_1)</td>
<td>(q_1)</td>
<td>(q_3)</td>
<td>(q_1)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(q_3)</td>
<td>(q_2)</td>
<td>(q_2)</td>
</tr>
<tr>
<td>(q_3)</td>
<td>(q_3)</td>
<td>(q_3)</td>
<td>(q_3)</td>
</tr>
</tbody>
</table>

- Büchi acceptance \(\{q_3\}\)
- computational tree over \(acbaa^\omega\)
- a run over \(acbaa^\omega\)
- corresponds to \(a U (\Diamond a \land \Diamond b)\)

Alternating Automata: Acceptance

A word is accepted iff there exists an infinite branch such that all of its infinite histories satisfy the acceptance condition

Alternatively,

- a run of an alternating word automaton is a tree
  - each branch in the computational tree is a run
  - the set of infinite histories associated with a branch forms a tree
- a run is accepting iff all of its branches are accepting
- a word is accepted iff there exists an accepting run
Symbolic Representation

A transition relation \( Q \times A \rightarrow 2^Q \) can be represented symbolically as a boolean formula over \( Q \):
- \( q_1 \lor q_2 \) is equivalent to \( \{q_1\}, \{q_2\} \)
- \( q_1 \land q_2 \lor q_3 \) is equivalent to \( \{q_1, q_2\}, \{q_3\} \)

Intuition
- \( q_1 \lor q_2 \) means
  - split into two copies
  - one switches to \( q_1 \), the other to \( q_2 \)
  - accept iff at least one copy accepts
- \( (q_1 \lor q_2) \land q_3 \)
  - split into 3 copies
  - 1st switches to \( q_1 \), 2nd to \( q_2 \), and 3rd to \( q_3 \)
  - accept if both the 3rd copy and either one of 1st or 2nd accept

Why Do We Need This?

Complementation is easy
- let \( \varphi \) be a boolean formula over \( X \)
- a dual \( \varphi_c \) of \( \varphi \) is obtained by switching \( \land \) with \( \lor \)
- a dual of \( (a \land b) \lor c \) is \( (a \lor b) \land c \)
- a complement of \( A = (\Sigma, Q, \delta, q_0, \mathcal{F}) \) is \( A_c = (\Sigma, Q, \delta_c, q_0, \mathcal{F}_c) \), where
  - \( \delta_c \) is the dual of \( \delta \), \( \mathcal{F}_c = Q^\omega \setminus \mathcal{F} \)

There is an easy translation from temporal logic (LTL) to alternating Büchi word automaton

But, “there is no free lunch”
- an alternating automaton has finite number of states
- but, can split into infinitely many copies
- conversion to non-alternating automaton is not always possible, or cheap!
From LTL to Automata

- For an LTL formula $\varphi$
  - closure of $\varphi$, $cl(\varphi)$, is the set of all subformulas of $\varphi$

- An alternating automaton $A_\varphi$ that accepts all $2^{AP}$ labeled words that satisfy $\varphi$ is built as follows
  - $A_\varphi = (2^{AP}, cl(\varphi), \delta, \varphi, F)$
  - $\delta(q, \sigma)$ is defined as follows

    $\begin{align*}
    \delta(a, \sigma) &= a \in \sigma & \delta(\neg a, \sigma) &= a \notin \sigma \\
    \delta(\circ \varphi, \sigma) &= \varphi & \delta(\Box \varphi, \sigma) &= \delta(\varphi, \sigma) \land \Box \varphi \\
    \delta(\diamond \varphi, \sigma) &= \delta(\varphi, \sigma) \lor \diamond \varphi & \delta(\varphi \cup \psi, \sigma) &= \delta(\psi, \sigma) \lor \\
    & & \delta(\varphi, \sigma) \land \varphi \cup
    \end{align*}$

- $F = \{ \Box \psi \mid \Box \psi \in cl(\varphi) \}$ is a Büchi acceptance condition

Examples

- $a \cup b$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, {a, b})$</th>
<th>$\delta(q, {a})$</th>
<th>$\delta(q, {b})$</th>
<th>$\delta(q, \emptyset)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$b$</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>$a \cup b$</td>
<td>true</td>
<td>$a \cup b$</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

- no accepting states
Examples

- $a U b$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, {a, b})$</th>
<th>$\delta(q, {a})$</th>
<th>$\delta(q, {b})$</th>
<th>$\delta(q, \emptyset)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$b$</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>$\Diamond b$</td>
<td>true</td>
<td>$\Diamond b$</td>
<td>true</td>
<td>$\Diamond b$</td>
</tr>
<tr>
<td>$a U \Diamond b$</td>
<td>true</td>
<td>($\Diamond b) \lor (a U \Diamond b)$</td>
<td>true</td>
<td>$\Diamond b$</td>
</tr>
</tbody>
</table>

- no accepting states

Examples

- $\square a$

<table>
<thead>
<tr>
<th>state</th>
<th>${a}$</th>
<th>$\emptyset$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>$\square a$</td>
<td>$\square a$</td>
<td>$\square a$</td>
</tr>
</tbody>
</table>

- acceptance condition $\{\square a\}$

- $(\square a) \land (\square b)$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, {a, b})$</th>
<th>$\delta(q, {a})$</th>
<th>$\delta(q, {b})$</th>
<th>$\delta(q, \emptyset)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>$b$</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>$\square a$</td>
<td>$\square a$</td>
<td>$\square a$</td>
<td>false</td>
<td>$\square a$</td>
</tr>
<tr>
<td>$\square b$</td>
<td>$\square b$</td>
<td>$\square b$</td>
<td>false</td>
<td>$\square b$</td>
</tr>
<tr>
<td>$(\square a) \land (\square b)$</td>
<td>$(\square a) \land (\square b)$</td>
<td>$(\square a) \land (\square b)$</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
Automata over Infinite Trees

Outline

- Automata on infinite trees
  - deterministic automata
  - nondeterministic automata
  - alternating automata
- Constructing an Alternating Tree Automaton for CTL
Trees

A tree is a tuple \((V_t, V_l, E, r)\), where
- \(V_t\) and \(V_l\) are the set of tree and leaf nodes, respectively
- \((V_t \cup V_l, E)\) is a directed acyclic graph
- \(E \subseteq V_t \times (V_t \cup V_l)\) is the set of edges
- \(r \in V_t\) is the root node, \(\forall x \in V_t \cdot (x, r) \notin E\)

A tree is the set of paths from the root to the leaves
- assume nodes at each level are enumerated
- each path is an element of \(\mathbb{N}^*\)
  - \(\epsilon\) is the root node
  - \(0 \cdot 1 \cdot 0\) means: go to child 0, then 1, then 0

Trees

A tree \(\tau\) is a subset of \(\mathbb{N}^*\) such that
- \(\tau\) is prefix closed
  - \(\epsilon \in \tau\)
  - \(\forall x \in \mathbb{N}^* \cdot \forall y \in \mathbb{N} \cdot (x \cdot y) \in \tau \Rightarrow x \in \tau\)
- \(\tau\) is child closed
  - \(\forall x \in \mathbb{N}^* \cdot \forall y \in \mathbb{N} \cdot (x \cdot y) \in \tau \Rightarrow \forall z \leq y \cdot (x \cdot z) \in \tau\)
- each node \(x \in \tau\) is described by the unique path from the root to \(x\)

A degree \(d(x)\) of a node \(x\) is the number of successors of \(x\)
- \(\forall y < d(x) \cdot (x \cdot y) \in \tau \wedge (x \cdot d(x)) \notin \tau\)
Trees

- A tree $\tau$ is $n$-ary iff
  - every non-leaf node has degree $d(x)$
  - $\forall x \in \tau \cdot d(x) = n \lor d(x) = 0$

- A tree is leafless if degree of every node $> 0$

- A $D$ labeled tree is a tuple $(\tau, L)$, where
  - $\tau$ is a tree
  - $L : \mathbb{N}^* \rightarrow D$ is a labeling function

- A string is 1-ary tree

- An infinite string is a leafless 1-ary tree

- A finite word is a $\Sigma$-labeled 1-ary tree

- An infinite word is a $\Sigma$-labeled 1-ary leafless tree

Tree Automata

- A tree automaton is a tuple $A = (\Sigma, Q, q_0, \delta, \mathcal{F})$, where
  - $\Sigma$ is a finite alphabet
  - $Q$ is a finite set of states
  - $q_0 \in Q$ is the initial state
  - $\delta$ is the transition relation
    - different depending on the type of the automaton
  - $\mathcal{F} \subseteq Q^\omega$ is the acceptance condition
    - can be Büchi, Rabin, Street, Parity, etc.

- For a deterministic $n$-ary tree automaton $\delta : Q \times \Sigma \rightarrow Q^n$, where $\delta(q, a) = (w_0, \ldots, w_{n-1})$ means
  - if $A$ is in state $q$, and reads node labeled with $a$, then
    - $A$ splits into $n$ copies
    - copy $i$ is switched to state $w_i$, and
    - is sent to the $i$th successor of the tree node
Example

deterministic automaton accepting all binary \(\{a, b\}\)-labeled trees that have a \(b\) along every branch

corresponds to \(AFb\)

<table>
<thead>
<tr>
<th>state</th>
<th>(\delta(q, a))</th>
<th>(\delta(q, b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_0)</td>
<td>((q_0, q_0))</td>
<td>((q_1, q_1))</td>
</tr>
<tr>
<td>(q_1)</td>
<td>((q_1, q_1))</td>
<td>((q_1, q_1))</td>
</tr>
</tbody>
</table>

acceptance is Büchi \(\{q_1\}\)

Run and Acceptance

A run of a deterministic tree automaton on a \(\Sigma\)-labeled \(n\)-ary tree \((T, V)\) is a \(\mathbb{N}^* \times Q\)-labeled tree \((T, V_r)\), where

- \(V_r(x) = (x, q)\) indicates that the automaton read letter \(V(x)\) while in state \(q\)
- \(V_r(\epsilon) = (\epsilon, q_0)\)
- if \(V_r(x) = (x, q)\) and \(\delta(q, V(x)) = (w_0, \ldots, w_{n-1})\), then
  \(\forall y < n \cdot (x \cdot y) \in T\), and
  \(V_r(x \cdot y) = (x \cdot y, w_y)\)

A run is accepting iff all of its branches satisfy the acceptance condition
Computational Tree of a Tree Automaton

- A computational tree of $A$ on a tree $(T, V)$ is a tree of all runs of $A$ on $(T, V)$
- Computational tree of a deterministic tree automaton is linear
- Each node of the computational tree is labeled by a set of histories
- A history is a string $(\mathbb{N} \times Q)^*$ describing a run of an automaton over a single branch of the input tree
- A branch $\beta$ of a computational tree is accepting iff all infinite histories associated with it are accepting
- A tree is accepted iff exists an accepting branch of the computational tree

Non-Deterministic Tree Automata

- For a non-deterministic tree automaton $\delta : Q \times \Sigma \rightarrow 2^Q^n$, where $\delta(q, a) = \{W_0, \ldots, W_k\}$ means
  - if $A$ is in state $q$, and reads a node labeled with $a$
  - pick $W_i \in \delta(q, a)$ and proceed as a deterministic automaton
- A run of a non-deterministic automaton is defined as for the deterministic case
- A computational tree of a non-deterministic tree automaton is branching
  - a tree is accepted iff there exists an accepting branch of the computational tree
  - or equivalently, iff there exists an accepting run
Example

- non-deterministic binary tree automaton that accepts an \( \{a, b\} \)-labeled tree if at least one branch contains an \( a \)
- corresponds to \( EFa \)

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
<th>( \delta(q, b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>( (q_1, q_1) )</td>
<td>( {(q_0, q_1), (q_1, q_0)} )</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>( (q_1, q_1) )</td>
<td>( (q_1, q_1) )</td>
</tr>
</tbody>
</table>

- acceptance is Büchi \( \{q_1\} \)

Symbolic Transition Relation

- For deterministic and non-deterministic tree automata transition relation can be described by a boolean formula over \( \mathbb{N} \times Q \)
- For deterministic binary tree automaton
  - \( \delta(q, a) = (w_0, w_1) \) becomes
  - \( \delta(q, a) = (0, w_0) \land (1, w_1) \)
- For a non-deterministic binary tree automaton a choice is encoded by a disjunction
  - \( \delta(q, a) = \{(w_0, w_1), (w_2, w_3)\} \) becomes
  - \( \delta(q, a) = ((0, w_0) \land (1, w_1)) \lor ((0, w_2) \land (1, w_3)) \)
- note that both conjunction and disjunction are used
Alternating Tree Automata

- For a set $X$, let $B(X)$ denote the set of all positive boolean formulas over $X$
- A set $Y \subseteq X$ satisfies a formula $\theta \in B(X)$ if treating atoms in $Y$ as true, and in $X \setminus Y$ as false, makes $\theta$ true
- $X = \{a, b, c\}$
- $\{a, b\}$ satisfies $a \land b \lor c$, and
- does not satisfy $a \land b \land c$
- An alternating n-ary tree automaton is a tree automaton with transition relation $\delta(q, a) \in B(\{0, \ldots, n - 1\} \times Q)$
- $(0, q_1) \lor (0, q_2) \land (1, q_1)$

Example

- Alternating automaton that accepts all binary $\{a, b\}$-labeled trees where $b$ occurs as a child of every node at the second level
- Corresponds to $AXEXb$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, a)$</th>
<th>$\delta(q, b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$(0, q_1) \land (1, q_1)$</td>
<td>$(0, q_1) \land (1, q_1)$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$(0, q_2) \lor (1, q_2)$</td>
<td>$(0, q_2) \lor (1, q_2)$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$(0, q_4) \land (1, q_3)$</td>
<td>$(0, q_3) \land (1, q_3)$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$(0, q_3) \land (1, q_3)$</td>
<td>$(0, q_3) \land (1, q_3)$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>$(0, q_4) \land (1, q_4)$</td>
<td>$(0, q_4) \land (1, q_4)$</td>
</tr>
</tbody>
</table>

- acceptance is Büchi $\{q_3\}$
Alternating Automata

- A run of an alternating n-ary tree automaton $A$ over a $\Sigma$-labeled tree $(T, V)$ is a $\mathbb{N}^* \times Q$ labeled tree $(T_r, V_r)$
- $V_r(\epsilon) = (\epsilon, q_0)$
- if $V_r(x) = (y, q)$ and $\delta(q, a) = \theta$, then there exists a possibly empty set $Y = \{(c_0, w_0), \ldots, (c_k, w_k)\}$ such that
  - $Y$ satisfies $\theta$, and
  - for all $0 \leq i \leq k$, $x \cdot i \in T_r$, and $V_r(x \cdot i) = (y \cdot c_i, w_i)$

- A tree $(T, V)$ is accepted by $A$ iff there exists an accepting run of $A$ over $(T, V)$

ATA Computational Tree

- As before, we can build a computational tree of $A$ over a $\Sigma$-labeled tree $(T, V)$
- Nodes in the computational tree are labeled with histories
  - but, nodes at the same level can have different number of histories
  - this happens since an alternating automaton is allowed to send multiple copies to the same direction, and even skip some directions

- A tree is accepted by the automaton iff there exists an accepting infinite branch in the computational tree
**Example**

- Automaton over binary \( \{a\} \)-labeled tree

<table>
<thead>
<tr>
<th>state</th>
<th>( \delta(q, a) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_0 )</td>
<td>((0, q_0) \wedge (1, q_2) \vee (0, q_1))</td>
</tr>
<tr>
<td>( q_1 )</td>
<td>((0, q_1) \wedge (0, q_2) \wedge (1, q_2))</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>((0, q_2))</td>
</tr>
</tbody>
</table>

- Computational tree is branching

**Extending to Arbitrary Trees**

- We only considered trees with a fixed branching degree
- Let \( \mathcal{D} \subseteq \mathbb{N} \)
  - A \( \mathcal{D} \)-tree is a tree such that a branching degree of every node is in \( \mathcal{D} \)
  - \( \forall x \cdot d(x) \in \mathcal{D} \)
- A \( \mathcal{D} \)-tree automaton has transition relation \( \delta : Q \times \Sigma \times \mathcal{D} \rightarrow B(\mathbb{N} \times Q) \)
  - \( \delta \) is defined separately for each branching degree
  - \( \delta(q, a, k) \) can only contain terms from \( \{0, k - 1\} \times Q \)
- A size of a \( \mathcal{D} \)-tree automaton \( A_{\mathcal{D}} \) is
  \( \|A_{\mathcal{D}}\| = |\mathcal{D}| + |Q| + |F| + ||\delta|| \)
  \( ||\delta|| = \sum_{q, a, k} |\delta(q, a, k)| \) where \( \delta(q, a, k) \neq \text{false} \)
**Model: Kripke Structure**

- As usual, our models are Kripke structures \( K = (AP, S, s_0, R, L) \)
  - \( AP \) is the set of atomic propositions
  - \( S \) is a finite set of states
  - \( s_0 \in S \) an initial state
  - \( R \subseteq S \times S \) the transition relation
  - \( L : S \rightarrow 2^{AP} \) is the labeling function

- A Kripke structure induces a \( S \)-labeled tree \((T_K, V_K)\)
  - \( V_K : \mathbb{N}^* \rightarrow S \) labels each node with a state
    - \( V_K(\epsilon) = s_0 \)
  - \( T_K \subseteq \mathbb{N}^* \) is a tree such that
    - for \( y \in T_K \) with \( R(V_K(y)) = (w_0, \ldots, w_m) \) we have
      \[ \forall 0 \leq i \leq m \cdot (y \cdot i) \in T_K \quad \text{and} \quad V_K(y \cdot i) = w_i \]

---

**Computation Tree**

- A Kripke structure can be seen as a computation tree over its atomic propositions

- For a Kripke structure \( K \)
  - \( (T_K, V_K) \) is its tree unrolling
  - \( (T_K, L \circ V_K) \) is its computation tree
Temporal Logic: CTL

Computation Tree Logic is interpreted over a computation tree of a Kripke structure.

Definition

\[
\begin{align*}
||p||_s &= p \in L(s, p) \\
||\neg \varphi||_s &= \neg ||\varphi||_s \\
||\varphi \land \psi||_s &= ||\varphi||_s \land ||\psi||_s \\
||\varphi \lor \psi||_s &= ||\varphi||_s \lor ||\psi||_s \\
||EX \varphi||_s &= \exists t \in R(s) \cdot ||\varphi||_t \\
||AX \varphi||_s &= \forall t \in R(s) \cdot ||\varphi||_t \\
||E[\varphi U \psi]||_s &= ||\mu Z \cdot \psi \lor \varphi \land EX Z||_s \\
||A[\varphi U \psi]||_s &= ||\mu Z \cdot \psi \lor \varphi \land AX Z||_s \\
||E[\varphi R \psi]||_s &= ||\nu Z \cdot \psi \lor (\varphi \lor EX Z)||_s \\
||A[\varphi R \psi]||_s &= ||\nu Z \cdot \psi \lor (\varphi \lor AX Z)||_s
\end{align*}
\]

From CTL to ATA

For a CTL formula \( \varphi \) we construct an alternating \( \mathcal{D} \)-tree automaton \( A_{\mathcal{D}, \varphi} \) that accepts all \( \mathcal{D} \)-trees that are models of \( \varphi \).

\[
A_{\mathcal{D}, \varphi} = (2^{AP}, cl(\varphi), \varphi, \delta, F)
\]

- the alphabet is all subsets of \( AP \)
- states correspond to sub-formulas of \( \varphi \)
- initial state is \( \varphi \)
- acceptance condition is Büchi and consists of all \( AR \) and \( ER \) sub-formulas
- \( \delta \) is the transition relation

Intuitively, \( A_{\mathcal{D}, \varphi} \) accepts a tree from a state \( q \) iff the tree is the model of the formula associated with \( q \).
From CTL to ATA

\[
\begin{align*}
\delta(p, \sigma, k) &= \text{true if } p \in \sigma \\
\delta(p, \sigma, k) &= \text{false if } p \not\in \sigma \\
\delta(\neg p, \sigma, k) &= \text{false if } p \in \sigma \\
\delta(\neg p, \sigma, k) &= \text{true if } p \not\in \sigma \\
\delta(\varphi \land \psi, \sigma, k) &= \delta(\varphi, \sigma, k) \land \delta(\psi, \sigma, k) \\
\delta(\varphi \lor \psi, \sigma, k) &= \delta(\varphi, \sigma, k) \lor \delta(\psi, \sigma, k) \\
\delta(AX\varphi, \sigma, k) &= \bigwedge_{c=0}^{k-1} (c, \varphi) \\
\delta(EX\varphi, \sigma, k) &= \bigvee_{c=0}^{k-1} (c, \varphi) \\
\delta(A[\varphi U \psi], \sigma, k) &= \delta(\psi, \sigma, k) \lor \delta(\varphi, \sigma, k) \land \bigwedge_{c=0}^{k-1} (c, A[\varphi U \psi]) \\
\delta(A[\varphi R \psi], \sigma, k) &= \delta(\psi, \sigma, k) \land (\delta(\varphi, \sigma, k) \lor \bigwedge_{c=0}^{k-1} (c, A[\varphi R \psi])) \\
\delta(E[\varphi U \psi], \sigma, k) &= \delta(\psi, \sigma, k) \lor \delta(\varphi, \sigma, k) \land \bigvee_{c=0}^{k-1} (c, E[\varphi U \psi]) \\
\delta(E[\varphi R \psi], \sigma, k) &= \delta(\psi, \sigma, k) \land (\delta(\varphi, \sigma, k) \lor \bigvee_{c=0}^{k-1} (c, E[\varphi R \psi]))
\end{align*}
\]

Examples

\[\psi = AFAGp\]
- in negation normal form: \(A[\text{true } U (A[\text{false } R p])]\)
- alphabet \(2^\{p\}\)

\[
\begin{array}{c|c|c}
\text{state} & \delta(q, \{p\}, k) & \delta(q, \emptyset, k) \\
\hline
\psi & \bigwedge_{c=0}^{k-1} (c, A[\text{false } R p]) \lor \bigwedge_{c=0}^{k-1} (c, \psi) & \bigwedge_{c=0}^{k-1} (c, \psi) \\
A[\text{false } R p] & \bigwedge_{c=0}^{k-1} (c, A[\text{false } R p]) & \text{false}
\end{array}
\]

- acceptance condition is Büchi \(\{A[\text{false } R p]\}\)
Examples

- $\psi = A[(-AXp) \ U \ b]$
  - in negation normal form: $A[(EX-p) \ U \ q]$
  - alphabet $\{p, b\}$

<table>
<thead>
<tr>
<th>state</th>
<th>$\delta(q, {p, b}, k)$</th>
<th>$\delta(q, {p}, k)$</th>
<th>$\delta(q, {b}, k)$</th>
<th>$\delta(q, \emptyset, k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\psi$</td>
<td>true</td>
<td>$\bigvee_{c=0}^{k-1} (c, -p) \land \bigwedge_{c=0}^{k-1} (c, \psi)$</td>
<td>true</td>
<td>$\bigvee_{c=0}^{k-1} (c, -p) \land \bigwedge_{c=0}^{k-1} (c, \psi)$</td>
</tr>
<tr>
<td>$\neg p$</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

- acceptance condition is empty