Non Deterministic Tree Automata

By: Or Kamara

From: Nondeterministic Tree Automata in Automata, Logic and infinite games, edited by Gradel, Thomas and Wilke (chapter 8)
So far

Word automata -

Consume infinite sequences of alphabet symbols (ω-words)
Today

Tree automata -

Finite-state automata

which process infinite trees
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
- Buchi tree automata Vs. Muller tree automata
- The Complementation Problem for Automata on Infinite Trees
  - Game theoretical approach
  - Complementation proof
First of all - WHY?

- Tree automata are similar to logical theories →
  Reduce problems in logic to problems for automata.

- Tree automata are more suitable than words when non-determinism needs to be modeled.
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
- Buchi tree automata Vs. Muller tree automata
- The Complementation Problem for Automata on Infinite Trees
  - Game theoretical approach
  - Complementation proof
Definitions - infinite binary tree

- \( T^\omega = \{0, 1\}^* \) of all finite words on \{0,1\}
- Elements \( u \in T^\omega \) are the nodes of \( T^\omega \):
  - \( \varepsilon \) - root
  - \( u_0, u_1 \) - immediate successors of node \( u \)

- Again :)  
  - Left - \( U_0 \)
  - Right - \( U_1 \)
Definitions - infinite binary tree

- **Path** - $\omega$-word $\pi \in \{0,1\}^\omega$
- Set $\text{Pre}<(\pi) \subseteq \{0,1\}^*$ of all prefixes of path $\pi$
  - Describes the set of nodes which occur in $\pi$

- Example:
  - The rightmost path in the tree is $\pi = 1^\omega$
  - All the prefixes of this path are:
  - $\text{Pre}<(\pi) = \{\epsilon, 1, 11, 111, 1111, \ldots\} = \{1\}^*$

![Diagram of an infinite binary tree](image)
Definitions - infinite binary tree

- Let $u, v \in T_\omega$, then $v$ is a **successor** of $u$, if there exists $w \in T_\omega$ such that $v = uw$
- Denoted by $u < v$

- Example:
  - $01$ is successor of $0$
  - $101$ is successor of $1 \backslash 10$
Definitions - infinite binary tree

- Our tree can be labeled
- $\Sigma$ is alphabet
- A mapping $t : T^\omega \rightarrow \Sigma$
  - Maps each node of $T^\omega$ to a symbol in $\Sigma$

*Example 8.1.* Let $\Sigma = \{a, b\}$, $t(\varepsilon) = a$, $t(w0) = a$ and $t(w1) = b$, $w \in \{0, 1\}^*$. 

![Diagram](image)
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
- Buchi tree automata Vs. Muller tree automata
- The Complementation Problem for Automata on Infinite Trees
  - Game theoretical approach
  - Complementation proof
Until now, automata consume one input symbol at a time
  ○ Enter a successor state determined by a transition relation
  ○ $\delta : Q \times \Sigma \rightarrow Q$

Now, we want to run automata on infinite trees

The **transition function**:
  ○ $\delta : Q \times \Sigma \rightarrow Q \times Q$
Definitions - Finite-State Tree Automata

Tree automaton is of the form $A = (Q, \Sigma, \delta, q_0, F)$, where:

- $Q$ is finite set of states
- $\Sigma$ is a finite alphabet
- $\delta \subseteq (Q \times \Sigma) \times (Q \times Q)$ is the transition function
- $q_0$ is the initial state
- $F$ is the acceptance component
Definitions - **Finite-State Tree Automata**

- Computations start at the root of an input tree and work through the input on each path in parallel.

- A transition \((q,a,q_1,q_2)\) allows to pass from state \(q\) at node \(u\) with label \(a\) i.e. \(t(u) = a\), to the states \(q_1,q_2\) at the successor nodes \(u_0,u_1\).
Assignments of states to the tree nodes

- $\rho: \{0,1\}^* \rightarrow Q$ with:
  - $\rho(\varepsilon) = q_0$
  - $(\rho(u), t(u), \rho(u_0), \rho(u_1)) \in \delta$ for all $u \in \{0,1\}^*$

- Example:
  - $(q_0, a, q_0, q_0)$
  - $(q_0, b, q_1, q_0)$
  - $(q_1, a, q_1, q_1)$
  - $(q_1, b, q_1, q_1)$
Definitions - run

- **Successful run**
  - Each path of the ρ is successful with respect to acceptance condition

- **Acceptance conditions:**
  - Buchi
  - Muller
  - Rabin
  - Parity

- **Language** of an automaton A with alphabet Σ, is the set of Σ-trees which are accepted by A
  - Denoted L(A)
Buchi Tree Automaton

- Tree automaton $A = (Q, \Sigma, \delta, q_0, F)$ accepts tree $t$ if there exists a run $\rho$ of $A$ on $t$, such that on EACH PATH of $\rho$, a state from $F$ occurs infinitely times.

What about Muller?

- For each path $\pi \in \{0,1\}^\omega$ the Muller acceptance condition is satisfied
  - $\text{Inf}(\rho|\pi) \in F$
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
- Buchi tree automata Vs. Muller tree automata
- The Complementation Problem for Automata on Infinite Trees
  - Game theoretical approach
  - Complementation proof
Starting Windows
Examples (1)

$L(A)$ is the set of all $\Sigma$-trees having at least one $b$ on every branch

Let’s look on every path separately

- $F = \{q_1\}$
- Transition function
  - $(q_0, a, q_0, q_0)$
  - $(q_0, b, q_1, q_1)$
  - $(q_1, a, q_1, q_1)$
  - $(q_1, b, q_1, q_1)$
Examples (1)

$L(A)$ is the set of all $\Sigma$-trees having at least one $b$ on every branch

Let's look on every path separately

- $F = \{q_1\}$
- Transition function
  - $(q_0, a, q_0, q_0)$
  - $(q_0, b, q_1, q_1)$
  - $(q_1, a, q_1, q_1)$
  - $(q_1, b, q_1, q_1)$
Examples (2)

$L(A_2)$ is the set of all $\Sigma$-trees which have at least one branch with infinitely many $b$’s

- $Q = \{q_a, q_b, q_+\}$
- $F = \{q_b, q_+\}$
- Transition relation:

```
\begin{align*}
q_a & \rightarrow q_a a a a a \ldots \\
q_a & \rightarrow q_a a a \ldots \\
q_a & \rightarrow q_a q_+ a/b q_+ q_+ q_+ \ldots \\
q_+ & \rightarrow +q +q +q +q \ldots \\
\end{align*}
```
Examples (2)

$L(A2)$ is the set of all $\Sigma$-trees which have at least one branch with infinitely many $b$’s.

- Non deterministic
- $Q = \{q_a, q_b, q_+\}$
- $F = \{q_b, q_+\}$
- Transition relation:
Examples (3)

$L(A3)$ is the set of all $\Sigma$-trees having infinitely many $a$’s on every branch

Buchi tree automata $A3$:

- $Q = \{q_a, q_b\}$
- Initial state - $q_a$
- $F = \{q_a\}$
- Transition function
  - $(q_a, a, q_a, q_a)$
  - $(q_b, a, q_a, q_a)$
  - $(q_a, b, q_b, q_b)$
  - $(q_b, b, q_b, q_b)$
Examples (4)

$L(A4)$ is the set of $\Sigma$-trees in which every branch contains only finitely many $b$'s.

Muller tree automata $A4$:

- $Q = \{q_a, q_b\}$
- Initial state - $q_a$
- $F = \{\{q_a\}\}$
- Transition function (the same as Example 3)
  - $(q_a, a, q_a, q_a)$
  - $(q_b, a, q_a, q_a)$
  - $(q_a, b, q_b, q_b)$
  - $(q_b, b, q_b, q_b)$
- Is it the same as Example 3?
Examples (5)

$L(A5)$ is the set of all $\Sigma$-trees having at least one path $\pi$ through $t$ such that $t|\pi \in (a + b)^*(ab)^\omega$

- We will use muller
- $A5$ memorizes in its state the last read input symbol
- $A5$ switches back to the initial state $q_I$ if he get unexpected symbol
- Infinite alternation between a state $q_a$ memorizing input symbol $a$ and a state $q_b$ memorizing $b$
- A will guess a path through $t$ and checks, if the label of this path belongs to $(a+b)^*(ab)^\omega$
Examples (5)

- **Guess** - decide whether the left or the right successor node of the input tree belongs to the path
- \( q_d \) signals that we are outside the guessed path
- \( \mathcal{A} = (\{q_I, q_a, q_b, q_d\}, \{a, b\}, \delta, q_I, \{\{q_a, q_b\}, \{q_d\}\}) \)

\[ \delta: \]
- Initial - \( (q_I, a, q_a, q_d) \setminus (q_I, a, q_d, q_a) \setminus (q_I, b, q_b, q_d) \setminus (q_I, b, q_d, q_b) \)
- For \( q_d \) - \( (q_d, a, q_d, q_d) \setminus (q_d, b, q_d, q_d) \)
- Change letter - \( (q_a, b, q_b, q_d) \setminus (q_a, b, q_d, q_b) \setminus (q_b, a, q_a, q_d) \setminus (q_b, a, q_d, q_a) \)
- Same letter - \( (q_a, a, q_I, q_d) \setminus (q_a, a, q_d, q_I) \setminus (q_b, b, q_I, q_d) \setminus (q_b, b, q_d, q_I) \)
Examples (5)

There is no situation where $\text{Inf} (\rho|\pi) =$

- $\{q_d\}$
- $\{q_a\} \setminus \{q_a, q_d\}$
- $\{q_b\} \setminus \{q_b, q_d\}$
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
  - Buchi tree automata Vs. Muller tree automata
  - The Complementation Problem for Automata on Infinite Trees
    - Game theoretical approach
    - Complementation proof
Theorem 1

Buchi tree automata are strictly weaker than Muller tree automata

- **In Hebrew:** There exists a Muller tree automaton recognizable language which is not Buchi tree automaton recognizable

Proof

- The language
  - $T = \{ t \in T[a,b] \mid \text{any path through } t \text{ carries only finitely many } b \}$

  can obviously be recognized by a Muller tree automaton (example 4)
Theorem 1 (proof)

- Assume for contradiction that $T$ is recognized by a Buchi tree automaton $B = (Q, \Sigma, \delta, q_i, F)$
- Let $n = |F| + 1$
- Consider the following tree:

\[
t(w) = \begin{cases} 
  b & w \in (1^+0)^i \text{ for } i \in \{1, \ldots, n\} \\
  a & \text{else}
\end{cases}
\]
Theorem 1 (proof)

- Because the automata accepts $t$:

- Path $1^\omega$ - a final state is visited say at $1m^0$
- Path $1m^001^\omega$ - a final state is visited say at $1m^001^m$
- ...
- Proceeding in this way we obtain $n + 1$ positions -

- $V_0 = 1m^0, V_2 = 1m^001^m, \ldots, V_n = 1m^001^m0^m \ldots 1^{mn}$ that get to final state
- For certain $i < j$, the same state appears at $V_i$ and $V_j$
- Between $V_i$ and $V_j$ - at least one label $b$ (by our construction)
Theorem 1 (proof)

We now construct another input tree $t'$ by infinite repetition of the path from $V_i$ to $V_j$ ($\pi$)

- This tree contains an infinite path which carries infinitely many b’s, thus $t' \not\in T$
- We can easily construct a successful run on $t'$ by copying the actions of $\pi$ infinitely often $\Rightarrow t' \in T$
- Contradiction
Theorem 2

Muller, parity, Rabin and Streett tree automata all recognize the same tree languages
Parity Tree Automaton

- Tree automaton $A = (Q, \Sigma, \delta, q_0, C)$
- Coloring $C: Q \rightarrow \{0, \ldots, k\}$
- Accepts tree $t$ if there exists a run $\rho$ of $A$ on $t$ such that on EACH PATH of $\rho$, the maximal color assumed infinitely often is even
- Example:
  - Automata that recognize trees that each path of them has only finitely many $b$
  - Use $q_a$ and $q_b$ to signal the letters $a, b$
  - $C(q_a) = 0$, $C(q_b) = 1$
  - The maximal color is even $\iff$ the letter $b$ occurs finitely often on the path
Time for a break!
Outline

- Motivation
- Infinite binary tree
- Finite-State Tree Automata
- Examples
- Buchi tree automata Vs. Muller tree automata
- The Complementation Problem for Automata on Infinite Trees
  - Game theoretical approach
  - Complementation proof
Parity Tree Automaton

- Tree automaton $A = (Q, \Sigma, \delta, q_0, C)$
- Coloring $C: Q \rightarrow \{0, \ldots, k\}$
- Accepts tree $t$ if there exists a run $\rho$ of $A$ on $t$ such that on EACH PATH of $\rho$, the maximal color assumed infinitely often is even

- Example:
  - Automata that recognize trees that each path of them has only finitely many $b$
  - Use $q_a$ and $q_b$ to signal the letters $a, b$
  - $C(q_a) = 0, C(q_b) = 1$
  - The maximal color is even $\iff$ the letter $b$ occurs finitely often on the path
Closure under complementation

- We will now show closure under complementation for tree languages acceptable by parity tree automata
  - and hence acceptable by Muller tree automata
- For every automata $A = (S, T, T_0, \mathcal{F})$, there is an automata $A' = (S', T', T_0', \mathcal{F}')$ such that:
  \[ v \in L(A') \iff v \notin L(A) \]
- We identify a parity tree automaton $A = (Q, \Sigma, \delta, q_I, c)$ and an input tree $t$ with an infinite two-person game $G_{A,t}$
Rules

- The players move alternately
- Player 0 (*Automaton*): picking transition from $\Delta$ such that the alphabet symbol of this transition equals that at the current node
- Player 1 (*Pathfinder*): determines whether to proceed with the left or the right successor
- Example
Run Example

$\Delta$: 

- $(q_I, b, q_b, q_{b_I})$
- $(q_I, a, q_I, q_{I})$
- $(q_b, b, q_b', q_{b})$
- $(q_b', a, q_{a_I}, q_{a})$
- $(q_a, a, q_a, q_{a})$
- $(q_a', b, q_{l_b}, q_{l_b})$
Winning

- **Play** - single sequence of actions
  - $\pi = s_0, d_1, s_1, d_2, ...$
- $In(\pi) = \{ s \in S | s=s_n \text{ for infinitely many } n \}$
- Player 0 **wins the play** if this infinite state sequence satisfies the acceptance condition of $A$
  - $In(\pi) \in F$
- Otherwise Player 1 wins
- Game = set of plays
Winning strategy

- **Automaton** - all paths of the corresponding run meet the acceptance condition -> A accepts the tree
- **Pathfinder** - if there exists a path which violates the acceptance condition for every state sequence chosen by player 0 -> A rejects the tree

In other words:

- Automaton has winning strategy in $G_{A,t} \iff t \in L(A)$
- Pathfinder has winning strategy in $G_{A,t} \iff t \notin L(A)$
Game Positions

- A play is an infinite sequence of game positions which alternately belong to player 0 or player 1
- A game can be considered as a **graph** which consists of all game positions as vertices
- Edges between different positions indicate that the succeeding position is reachable from the preceding one by a valid action
Definitions - Game Positions

- Player 0 (should decide on transition):
  \[ V_0 = \{(w,q) | w \in \{0,1\}^*, q \in Q\} \]

- Player 1 (should decide on state):
  \[ V_1 = \{(w,\delta) | w \in \{0,1\}^*, \delta \in \Delta_{t(w)}\} \]
min-parity game

Game $G_{A,\alpha}$ for a parity automata $A = (Q,\Sigma,\delta,q_I,c)$ and $\alpha \in \Sigma^\omega$ is a graph $(V, E, C)$ that serves as an arena for the two players 0 and 1.

The graph $(V, E, C)$ is defined as follows:

- The set of vertices $V$ can be partitioned into the two sets $V_0$ and $V_1$
  - $V_0 = Q \times \omega$
  - $V_1 = Q \times \mathcal{P}(Q) \times \omega$
- The edge relation $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ is defined by
  - $((q, i), (q, M, j)) \in E \iff j = i + 1$ and $M \in \text{Mod}(\delta(q, \alpha(i)))$
  - $((p, M, i), (q, j)) \in E \iff j = i$, $q \in M$, and $c(q) \in C$
Game position $u = (w,q)$ of player 0
Player 0 chooses a transition $\tau = (q,t(w),q_0,q_1)$
Game position $v = (w,\tau)$ of player 1
Edge $(u,v)$ then represents a valid move of player 0
Player 1 chooses a direction $i \in \{0,1\}$
Game position $u' = (w_i,q'_i)$ of player 0
Edge $(v,u')$ represents a valid move of player 1
Definition

We need to color the vertices (since it is a parity game):

The **coloring function** $C: V_0 \cup V_1 \rightarrow \{0, 1, ..., k\}$

- $\forall (w,q) \in V_0, C((w,q)) = c(q)$
- $\forall (w,(q,t(w),q_0,q_1)) \in V_1, C((w,(q,t(w),q_0,q_1))) = c(q)$

Why are we doing all of this?

- The game $G_{A,t}$ meet exactly the definition of min-parity game
- The notions of a **strategy**, a **memoryless strategy** and a **winning strategy**, as defined last lecture apply to the games $G_{A,t}$ as well
Lemma

A tree automaton $A$ accepts an input tree $t \iff$
There is a winning strategy for Player 0 from position $(\varepsilon, q_I)$ in the game $G_{A,t}$
Proof - 1'st Direction

- A accepts the input tree $t \Rightarrow$ there exists an accepted run $\rho$
- The run $\rho$ keeps track of all transitions that have to be chosen in order to accept the input tree $t$
- For any of the nodes $(w,q) \in V_0$, where $(w_0,q_0)$ and $(w_1,q_1)$ are the immediate successors, we can derive the corresponding transition
  - $\delta = \langle q, t(w), q_0, q_1 \rangle \in \Delta$
- Since $\rho$ determines for each node and each path the correct transition, Player 0 can always choose the right transition, independently of Player 1’s decisions
- He will always win
- (Player 1’s decide on the direction, but A accepts all the paths of $t$)
Proof - 2'nd Direction

- We can use the winning strategy $f_0$ for player 0 in $G_{A,t}$ to construct a successful run of $A$ on $t$
- For each game position $(w,q) \in V_0$,
  $f_0$ determines the correct transition $\delta = (q,t(w),q_0,q_1) \in \Delta$
- Player 0 must be prepared to proceed from both $(w_0,q_0)$ and $(w_1,q_1)$ since he cannot predict player 1's decision
- However, for both positions the winning strategy can determine correct transitions
- Hence we label $w$ by $q$, $w_0$ by $q_0$ and $w_1$ by $q_1$ => obtain the entire run $\rho$
- $\rho$ is successful since it is determined by a winning strategy $f_0$
Winning Strategy

Known facts about parity games:

- Parity games are determined
  - One of the players has a memoryless winning strategies
- Memoryless winning strategies are enough to win a game

In other words -

From any game position in $G_{A,t}$, either Player 0 or Player 1 has a memoryless winning strategy
The Complementation of Finite Tree Automata Languages

- Outline:
- Given a parity tree automaton $A$, we have to specify a tree automaton $B$ that accepts all input trees rejected by $A$
- **Rejecting** means that there is no winning strategy for player 0 from position $(\epsilon, q_i)$ in the game $G_{A,t}$
- This guarantees the existence of a memoryless winning strategy starting at $(\epsilon, q_i)$ for player 1
- We will construct an automaton that checks exactly this
Memoryless Strategy of Player 1

- Function $f_1 : \{0,1\}^* \times \Delta \rightarrow \{0,1\}$ determining a direction 0 (left successor) or 1 (right successor)
- There is a natural isomorphism between such functions and functions of the type $f_1 : \{0,1\}^* \rightarrow (\Delta \rightarrow \{0,1\})$
- $f_1$ is a tree (with functions as labels)
- We call such trees strategy trees
- If the corresponding strategy is winning for player 1 in the game $G_{A,t}$, we say it is a winning tree for $t$
Fact

From the previous definitions:

Let $A$ be a parity tree automaton and $t$ be an input tree.

There exists a winning tree $s$ for player 1 $\iff$ if $A$ does not accept $t$.
First step -

- $\omega$-automaton $M$ will decide if each path of $t$, using the strategy for player 1 defined by $s$ (tree), will be accepted by $A$. If yes it accepts.
- $M$ will need to check all the possible strategies for player 0
- As we saw -
  - At least once $A$’s acceptance condition is met $\iff$ $s$ cannot be a winning tree for $t$
- $M$ needs to handle all $\omega$-words of the form
  $$u = (s(\varepsilon), t(\varepsilon), \pi_1)(s(\pi_1), t(\pi_1), \pi_2)....$$
Example

- Path $\pi = 0110...$ on $t$

<table>
<thead>
<tr>
<th>$u$:</th>
<th>$f_\varepsilon$</th>
<th>$f_0$</th>
<th>$f_{01}$</th>
<th>$f_{011}$</th>
<th>$f_{0110}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t(\varepsilon)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t(0)$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t(01)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t(011)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t(0110)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $f: \Delta \rightarrow \{0, 1\}$ (from $s$)
- $a \in \Sigma$
- $i \in \{0, 1\}$ ($f(\tau) = i$)
M - definitions

- $M = (Q, \Sigma', \Lambda, q_I, c)$
- $\Sigma' = \{(f, a, i) \mid f : \Delta \to \{0, 1\}, a \in \Sigma, i \in \{0, 1\}\}$
- A and M have the same acceptance conditions
- $\Lambda$ (transitions):
  - For $(f, a, i) \in \Sigma'$
    - $\text{map}_a$ denotes the set of all mappings from $\Delta_a$ to $\{0, 1\}$
    - $f \in \text{map}_a$, and $\tau = (q_a, q'_0, q'_1) \in \Delta_a$ such that $f(\tau) = i$
    - M has a transition $(q, (f, a, i), q'_i)$
M - informally

- The automaton M has to check for each possible move of Player 0 if the outcome is winning for Player 0.
- M uses the same acceptance condition as A:
  - It will accept if the run on a path is consistent with s and will be accepted by A.
- If M won't accept an $\omega$-word $u$ it means that player 1 win:
  - Since M checked all the options for player 0 and none of them worked.
Lemma

The tree $s$ is a winning tree for $t \iff L(s,t) \cap L(M) = \emptyset$

- **Language of $L(s,t)$** - all the possible paths that player 1 can choose, while using strategy $s$
- **Language of $L(M)$** - all the paths which are good for player 0 and consistent with player 1's strategy

- If $L(s,t) \cap L(M) = \emptyset$ then all the paths in $t$ which are consistent with $s$ will make player 1 win
  - i.e., $s$ is a winning strategy for player 1
Why is $M$ useful?

- The word automaton $M$ accepts all sequences over $\Sigma$ which satisfy A's acceptance condition.
- In order to construct $B$, we first of all generate a word automaton $S$ such that $L(S) = \Sigma' \setminus L(M)$.
- We’ll use Safra’s construction (chapter 3).
Building $S$

- We can transform $M$ to a Buchi-automaton
- By Safra, we can build deterministic Rabin automaton that accepts $L(M)$
- The Streett condition is the negation of the Rabin condition
- Finally
  - Word automaton $S = (Q', \Sigma', \delta, q'_1, \Omega)$
  - $L(S) = \Sigma' \setminus L(M)$
Building B from S

- B will run S in parallel along each path of an input tree
- The transition relation of B is defined by
- \((q,a,q_0,q_1) \in \Delta' \iff\) there exist transitions in S
  - \(\delta(q,(f,a,0)) = q_0\)
  - \(\delta(q,(f,a,1)) = q_1\)

Final Theorem:

The class of languages recognized by finite-state tree automata is closed under complementation

It remains to be shown that indeed \(T(B) = T^\omega_\Sigma \setminus T(A)\)
Proof - 1'st Direction

- We assume $t \in T(B)$
- There exists an accepting run $\rho$ of $B$ on $t$
- For each path $\pi = \pi_1 \pi_2 ... \in \{0,1\}^{\omega}$ the corresponding state sequence satisfies $B$'s acceptance condition
- There are transitions of $S$
  - $\delta(q, (s(w), t(w), 0)) = q_1$
  - $\delta(q, (s(w), t(w), 1)) = q_2$
- And the corresponding transition of $B$ $(q, t(w), q_1, q_2)$
- Player 1 is the winner $\Rightarrow$ all words $u \in L(s, t)$ are accepted by $S$
- Since $L(S) = \Sigma' \setminus L(M) \Rightarrow L(s, t) \cap L(M) = \emptyset$
- By the previous lemma - $s$ is a winning tree for Player 1
- A does not accept $t$
Proof - 2'nd Direction

- We assume $t \notin T(A)$
- There exists a winning tree $s$ for tree $t$ of player 1
- $L(s, t) \cap L(M) = \emptyset$
- $L(s, t) \subseteq L(S)$ (since $L(S) = \Sigma' \setminus L(M)$)
- For each path $\pi = \pi_1\pi_2 \cdots \in \{0, 1\}^\omega$ there exists a run on the $\omega$-word $u = (s(\epsilon), t(\epsilon), \pi_1)(s(\pi_1), t(\pi_1), \pi_2) \cdots \in L(s, t)$ that satisfies $\Omega$ (of automata $S$)
- By construction of $B$ there exists an accepting run of $B$ on $t$
- $t \in T(B)$
So?

1. From A (tree automaton) we built M (words automaton)
2. From M we built S such that $L(S) = \Sigma' \setminus L(M)$
3. From S we built B
Outline

● Motivation
● Infinite binary tree
● Finite-State Tree Automata
● Examples
● Buchi tree automata Vs. Muller tree automata
● The Complementation Problem for Automata on Infinite Trees
  ○ Game theoretical approach
  ○ Complementation proof