Automata and Logic (CSC2428)
Lecture 10

Prof. Leonid Libkin

March 24, 2005

1 Alternating Automaton

In the previous lectures we have seen nondeterministic finite automata defined as $\mathcal{A} = (Q, q_0, F, \delta)$, where δ, the transition function, is defined as $\delta : Q \times \Sigma \to 2^Q$. Informally, if q is a state in which $\mathcal{A}$ reads a, then $\delta(q, a) = \{q_1, \ldots, q_k\}$ means that $\mathcal{A}$ accepts if one of the choices $q_1, \ldots, q_k$ leads in a final (accepting) state. We can view the value of $\delta(q, a)$ as a logical disjunction between the states $q_1, \ldots, q_k$: $\delta(q, a) = q_1 \vee \ldots \vee q_k$.

How about reversing this, and asking the automaton to accept only if all the states spanning at q lead to an accepting state, or in other words having something of the form $\delta(q, a) = q_1 \wedge \ldots \wedge q_k$. The alternating automata include both of these two views: we have two types of states, existential and universal, and a run in the automaton keeps alternating between these two types of states. Now the formal definitions. Currently there are two views of the alternating automata on strings; we will see both them.

Definition 1 *In the first defintion the set of states are partitioned in existential and universal states. The intuition here is that the automaton must accept at least a run starting from an existential state, and all the runs that are spanning from an universal state.*

An alternating automaton *is defined as* $\mathcal{A}(Q, Q_\exists, Q_\forall, q_0, F, \delta)$, *where*

- Q *is a set of states;*
- $Q_\exists$ *and* $Q_\forall$ *are partitions of* Q*;*
- $q_0 \in Q$ *is the initial state;*
- $F \subseteq Q$ *is the final state, and*
- $\delta : Q \times \Sigma \to 2^Q$ *is the transition function, for* Σ *the alphabet of* $\mathcal{A}$.

In the second defintion we have only one set of states, and we need an auxiliary definition.

Definition 2 *Let* X *be a set of variables. Then* $\texttt{Bool}^+(X)$ *is the set of* positive propositional boolean formulas *over* X*, defined recursively as:*

- $\top, \bot \in \texttt{Bool}^+(X)$ *;*
- *if* $\alpha, \beta \in \texttt{Bool}^+(X)$*, then* $\alpha \vee \beta, \alpha \wedge \beta \in \texttt{Bool}^+(X)$ *;*
- *if* $x \in X$*, then* $x \in \texttt{Bool}^+(X)$.

Definition 3 *The* alternating automaton *is defined as* $\mathcal{A}(Q, q_0, F, \delta)$*, where the transition function is* $\delta : Q \times \Sigma \to \texttt{Bool}^+(Q)$.

For example, if in some states q, $\mathcal{A}$ reads a and $\delta(q,a) = (q_1 \vee q_2) \wedge (q_1 \vee q_3)$, then there are two possibilities: I can either continue the run from q_1, and if it accepts then $\mathcal{A}$ accepts, or I can span two copies of the automaton, one in q_2 and the other one in q_3, and check that they both accepts. Then $\mathcal{A}$ accepts.

No matter the definition of the alternating automaton $\mathcal{A}$, we have the language accepted by $\mathcal{A}$ defined as

$$\mathcal{L}_{\texttt{alt}}(\mathcal{A}) = \{s | s \text{ is accepted by } \mathcal{A}\}$$

The two definitions for alternating automaton are equivalent.

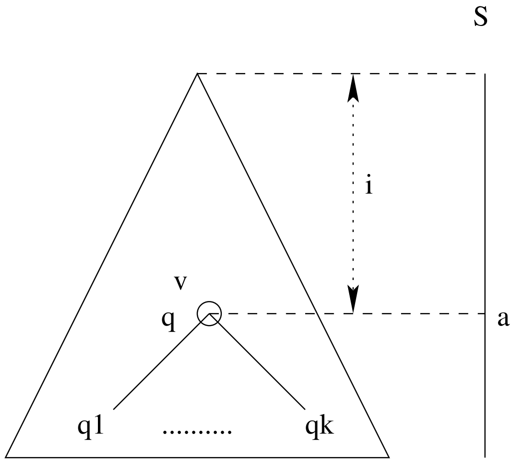

Figure 1: A run of $\mathcal{A}(Q, q_0, F, \delta)$

A run of $\mathcal{A}(Q, q_0, F, \delta)$ on a string s, is a tree R (see Figure 1) in which each node has at most $|Q|$ children, and there is a labelling $\lambda : R \to Q$, such that

- $\lambda(\texttt{root}) = q_0$,
- if $\lambda(v) = q$ and v is at level i in the tree, and the symbol of s in position i is a, then v has k children $v_1, v_2, \ldots, v_k$ labelled $q_1, q_2, \ldots, q_k$ such that $\{q_1, \ldots q_k\}$ makes $\delta(q,a)$ true, i.e., assigning 1's to $q_1, \ldots, q_k$ and 0's to all the other states will satisfy $\delta(q,a)$. When on a path, I touch $\top$, the brunch stops, and the tree will look like in Figure 2

Run R is accepting if every leaf at level $n = |s|$ is labelled by a state in F. Notice that not all the paths have to go all along the tree up to level n. Since this is an accepting run, if a path ends before, it will end up in a true state. However, if a state q is at level n, then $q \in F$.

Notice that this run is a special case of the previous automaton $\mathcal{A}(Q, Q_\exists, Q_\forall, q_0, F, \delta)$:

- if $q \in Q_\exists$, and $\delta(q,a) = \{q_1, \ldots, q_k\}$, then $\delta'(q,a) = q_1 \vee \ldots \vee q_k$,
- if $q \in Q_\forall$, and $\delta(q,a) = \{q_1, \ldots, q_k\}$, then $\delta'(q,a) = q_1 \wedge \ldots \wedge q_k$.

The transformation in the other sense is slightly more involved, and will not be treated here.

Obs. 1 *If $\mathcal{A}$ is an alternating automaton, then there exists a nondeterministic finite automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}_{\texttt{alt}}(\mathcal{A})$.*

Proof. (Ideea) If $\mathcal{A} = (Q, q_0, F, \delta)$, then $\mathcal{A}' = (2^Q, \{q_0\}, 2^F, \delta')$, where

$$\delta'(X, a) = \{Y \subseteq Q | Y \text{ makes true } \bigwedge \delta(q,a), q \in X\},$$

i.e., guess all states Y that occur on a same level of a tree. $\square$

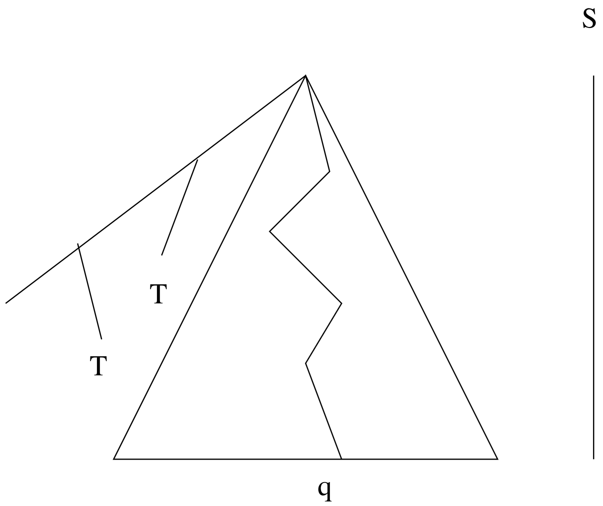

Figure 2: Accepting runs of $\mathcal{A}(Q, q_0, F, \delta)$

Corollary 1 *Languages of the form $\mathcal{L}_{\texttt{alt}}(\mathcal{A})$ are precisely regular languages.*

Remark 1 *Notice that the construction of the equivalent nondterministic finite automaton of an alternating automaton is exponential. However, the complement can be constructed in linear time (as opposed to exponential blow-up for the deterministic finite automata).*

1.1 Complementation of Alternating Automata

Definition 4 *Let $f \in \texttt{Bool}^+(X)$. Then the* dual *f^0 is defined by*

- $\top^0 = \bot$,
- $\bot^0 = \top, \bot^0 = \top$,
- $x^0 = x$ *for all* $x \in X$,
- $(\alpha \wedge \beta)^0 = \alpha^0 \vee \beta^0, (\alpha \vee \beta)^0 = \alpha^0 \wedge \beta^0$.

Definition 5 *If $\mathcal{A} = (Q, q_0, F, \delta)$ is an alternating automaton, then define the automaton $\overline{\mathcal{A}} = (Q, q_0, Q - F, \delta^0)$, where $\delta^0(q, a) = \delta(q, a)^0$.*

Lemma 2 *In the context of Definition 5,*

$$\mathcal{L}_{\texttt{alt}}(\overline{\mathcal{A}}) = \Sigma^* - \mathcal{L}_{\texttt{alt}}(\mathcal{A}).$$

Proof. Exercise! □

Obs. 2 *$\overline{\mathcal{A}}$ is constructed in linear time.*

Claim 3 *The exponential blow-up of the conversion from alternating to nondeterministic automaton is unavoidable, i.e., if $\mathcal{A}$ and $\mathcal{A}_{nd}$ are such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_{nd})$, then $||\mathcal{A}_{nd}|| = \Omega(2^{||\mathcal{A}||})$.*

Proof. Exercise! Ideea: take a regular language $\mathcal{L}$ such that

1. there is a small NFA $\mathcal{A}_{\mathcal{L}}$ that accepts $\mathcal{L}$;
2. there is no small NFA that accepts $\overline{\mathcal{L}}$

and use lower bounds for the powerset construction of deterministic automata, for which we know that complementation gives an exponential blow-up. □

1.2 Alternating Automata on infinite strings: Alternating Buchi Automata

Let $\mathcal{A} = (Q, q_0, F, \delta)$ be an alternating Buchi automaton. Then the run is just as before, for the finite strings case, but now the run does not end. Hence, it will be an infinite tree, see Figure 3.

As before, if $\delta(q, a)$ is satisfied by $q_1, \ldots, q_k$, then v has children $v_1, \ldots, v_k$, labelled by $q_1, \ldots, q_k$. A run is accepting if and only if every infinite branch is accepting with respect to the Buchi condition. An infinite branch is a string $s_\omega \in Q^\omega$, which is accepting when the states that occur infinitely oftern in s_ω has no empty intersection with F: $\mathtt{Inf}(s_\omega) \cap F \neq \emptyset$.

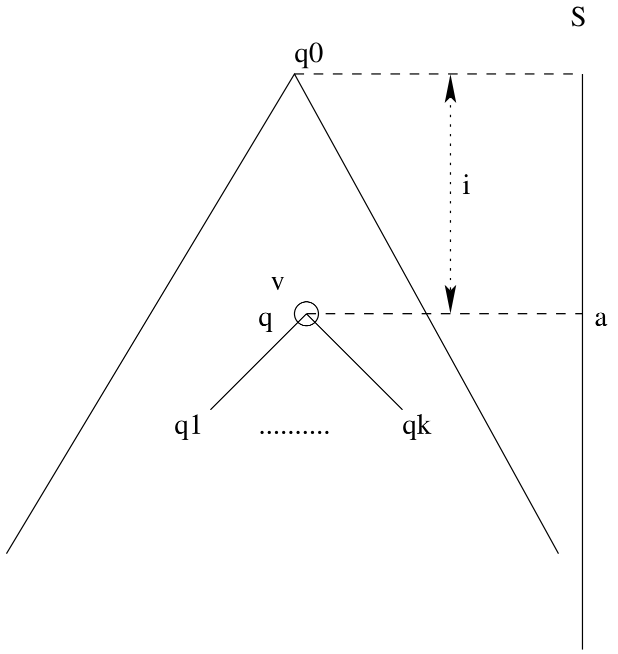

Figure 3: Run of Alternating Buchi Automaton

Proposition 4 *Alternating Buchi automata accept precisely the ω-regular languages.*

Proof. (Ideea.) Translate automata into MSO. □

Theorem 5 *(mid 80's.) If $\mathcal{A}$ is an alternating Buchi automaton, then one can construct in exponential time an equivalent Buchi nondeterministic automaton.*

Remark 2 *The size of the equivalent nondeterministic automaton from Theorem 5 is $2^{|Q|} \times 2^{|Q|}$. This is slightly worse than what we had before, for the finite case.*

1.3 Transform LTL into an alternating automaton (of linear size)

Recall that an LTL is defined over infinite strings, as the formulas

$$\phi, \phi' = a \mid \phi \wedge \phi' \mid \neg\phi \mid X\phi \mid \phi \cup \phi'$$

If the alphabet is 2^Σ, then the strings are $s \in (2^\Sigma)^*$.

Definition 6 *Let the* closure *of ϕ, $Cl(\phi)$ be the set of all subformulas of ϕ.*
Let the closure with respect to negation *of ϕ be $Cl^\neg(\phi) = Cl(\phi) \cup \{\neg\psi | \psi \in Cl(\phi)\}$.*

Definition 7 *Let α be a formula. We define the operator $\overline{\alpha}$ which represents the negation of the formula ϕ, after all the negations have been pushed down to the atoms (in this case, atoms are formulas from the closure). Formally, we define it by recursion, as follows:*

- $\overline{\top} = \bot, \overline{\bot} = \top$,
- $\overline{\psi} = \neg\psi$, *note that if* $\psi \in Cl^+(\phi)$, *then* $\neg\psi \in Cl^+(\phi)$,
- $\overline{\psi \vee \psi'} = \overline{\psi} \wedge \overline{\psi'}$,
- $\overline{\psi \wedge \psi'} = \overline{\psi} \wedge \overline{\psi'}$

Definition 8 *Let ϕ be a formula. We associate to ϕ the alternating automaton $\mathcal{A}^{\phi}_{\texttt{alt}} = (Q, q_0, F, \delta)$, such that*

- $Q = Cl^{\neg}(\phi)$;
- $q_0 = \phi$,
- $\delta : Q \times \text{alphabet } \rightarrow \texttt{Bool}^+(Q)$, (*i.e.*, $\delta : Cl^{\neg}(\phi) \times 2^{\Sigma} \rightarrow \texttt{Bool}^+(Cl^{\neg}(\phi))$), *such that*

  1. $\delta(a, S) = \begin{cases} \top & \text{if } a \in S \\ \bot & \text{if } a \notin S \end{cases}$;
  2. $\delta(\psi \wedge \psi', S) = \delta(\psi, S) \wedge \delta(\psi', S)$;
  3. $\delta(\neg\psi, S) = \overline{\delta(\psi, S)} \in \texttt{Bool}^+(Cl^{\neg}(\phi))$;
  4. $\delta(X\psi, S) = \psi$;
  5. $\delta(\psi \cup \psi', S) = \delta(\psi', S) \vee (\delta(\psi, S) \wedge \psi \cup \psi')$.

- $F = \{\neg(\psi \cup \psi') \in Cl^{\neg}(\phi)\}$;

Remark 3 *In the right hand side of the definition of δ, above, I always have a simpler formula. Only formulas $\psi \cup \psi', \neg(\psi \cup \psi')$ may occur infinitely often. So, unless I hit an "until" formula or a negation of an "until" formula, I will very quickly reduce the formula to atomic formulas.*

Theorem 6 *The automaton $\mathcal{A}^{\phi}_{\texttt{alt}}$ from Definition 8 accepts a sequence if and only if the formula ϕ is true.*
Proof. By induction on formulas. □

Remark 4 *The size of the automaton $\mathcal{A}^{\phi}_{\texttt{alt}}$ from Definition 8 is linear in the size of ϕ. Hence, by the Vardi-Walper theorem the size of the nondeterministic automaton equivalent to $\mathcal{A}^{\phi}_{\texttt{alt}}$ has size $2^{O(||\phi||)}$.*

2 μ-Calculus

The μ-Calculus L_μ is a powerful fix point logic. Assume we have $E_1, \ldots, E_n$ some binary relation symbols, and the formulas are defined by

$$\phi, \phi' = a \mid X \mid \phi \vee \phi' \mid \neg\phi \mid \Diamond(E_i)\phi \mid \mu X \phi \mid \nu X \phi,$$

if X occurs *positively*[1] in ϕ, and $\nu X \phi \leftrightarrow \neg\mu X \neg\phi(\neg X)$ is the greatest fixed point.

We consider the structure $\mathcal{M} = (\mathcal{U}, E_1, \ldots, E_n, (P_a)_{a \in \Sigma})$, where $E_i \subseteq \mathcal{U} \times \mathcal{U}$ are interpretation for the predicate symbols E_i's. Let $\texttt{Var}$ be the set of variables. Then a valuation is $v : \texttt{Var} \rightarrow 2^{\mathcal{U}}$.

[1] under an even number of negations

Definition 9 *(Semantics) $(\mathcal{M}, s, v) \models \phi$ is defined by recursion as follows:*

- $(\mathcal{M}, s, v) \models a \Leftrightarrow s \in P_a$
- $(\mathcal{M}, s, v) \models X \Leftrightarrow s \in v(X)$
- $(\mathcal{M}, s, v) \models \Diamond(E_i)\phi \Leftrightarrow (\mathcal{M}, s', v) \models \phi$ *for some* $(s, s') \in E_i$
- $(\mathcal{M}, s, v) \models \mu X \phi$ *if s is in the least fixed point of ϕ*

Definition 10 *The pair (ϕ, v) defines an operator $F_{\phi,v} : 2^{\mathcal{U}} \to 2^{\mathcal{U}}$ as follows:*

$$F_{\phi,v}(S) = \{s | (\mathcal{M}, s, v[S/X]) \models \phi\}.$$

Obs. 3 *$F_{\phi,v}$ is* monotone*: $S \subseteq S' \Rightarrow F_{\phi,v}(S) \subseteq F_{\phi,v}(S')$.*

Definition 11 *(Fixed point, Least fixed point, Greatest fixed point) Let $S \in 2^{\mathcal{U}}$. Then*

1. *S is a fixed point of $F_{\phi,v}$ if and only if $F_{\phi,v}(S) = S$.*
2. *S is a least fixed point of $F_{\phi,v}$ if S is a fixed point of $F_{\phi,v}$, and for every other fixed point S' of $F_{\phi,v}$, we have that $S \subseteq S'$.*
3. *S is a greatest fixed point of $F_{\phi,v}$ if S is a fixed point of $F_{\phi,v}$, and for every other fixed point S' of $F_{\phi,v}$, we have that $S' \subseteq S$.*

Theorem 7 *(Tarski/Knaster) Every monotone operator $F : 2^U \to 2^U$ has a least fixed point $lfp(F)$ such that*

$$lfp(F) = \cap\{S | F(S) = S\} = \cap\{S | F(S) \subseteq S\}.$$

Every monotone operator $F : 2^U \to 2^U$ has a greatest fixed point $gfp(F)$ such that

$$gfp(F) = \cup\{S | F(S) = S\} = \cup\{S | S \subseteq F(S)\}.$$

μ-Calculus is convenient because it can express many useful temporal logic properties:

- $\mu X(a \vee \Diamond X) = EFa$ – there is a path on which eventually a is true.
- $\nu X(a \wedge \Box X) = AGa$ – on every path a is always true.
- $\Box\phi \leftrightarrow \neg\Diamond\neg\phi$ – in every successor ϕ holds.
- $\Box\phi \leftrightarrow \neg\Diamond\phi$ – there is a successor in which ϕ holds.
- $\mu X(a \vee \Box X) = AFa$ – on every path eventually a is true.
- $\nu X(a \wedge \Diamond X) = EGa$ – there is a path such that a is true on all this path.

Let's consider that the structures are Σ-labelled infinite binary trees, so they are of the form

$$(\{0,1\}^*, \mathtt{succ}_0, \mathtt{succ}_1, (P_a))$$

$\Diamond_0\phi \hookrightarrow \Diamond(\mathtt{succ}_0)\phi$
$\Diamond_1\phi \hookrightarrow \Diamond(\mathtt{succ}_1)\phi$.

Assume ϕ is μ-calculus formula over infinite Σ-labelled binary trees, and ϕ has no free variables, So ϕ defines a class

$$L_\phi = \{T \in \Im^\omega_\Sigma | (T, \epsilon) \models \phi\}$$

Theorem 8 *(97-98) The following have equivalent expressiveness over infinite Σ-labelled binary trees:*

1. *MSO*
2. *Rabin tree automaton*
3. *μ-calculus*
4. *alternating tree automaton*

In the following we are interested what is the relation between the MSO and the μ-calculus over arbitrary structures and over non-binary trees. The conclusion is stated in Theorem 9 bellow. First let's give some helpful definitions.

Definition 12 *Assume we have two graphs (U, E, P_a, s_0) and (U', E', P'_a, s'_0).* Bisimulation *is an equivalence relation $\sim \subseteq U \times U'$, such that*

- $s_0 \sim s'_0$,
- *$\sim$ preserves P_a's,*
- *if $s \sim s'$ then*
  - *if $(s, s_1) \in E$ then $\exists (s', s'_1) \in E'$ such that $s_1 \sim s'_1$*
  - *if $(s', s'_1) \in E'$ then $\exists (s, s_1) \in E$ such that $s_1 \sim s'_1$.*

Definition 13 *MSO sentence ϕ is* bisimulation invariant *if $\mathcal{M} \models \phi(S) \Leftrightarrow \mathcal{M}' \models \phi(S')$ whenever $(\mathcal{M}, S)$ and $(\mathcal{M}', S')$ are bisimilar.*

Definition 14 Counting μ-Caluculus *has at least k different E_i successors in which ϕ is true:*

$$\phi, \phi' = a \mid X \mid \phi \vee \phi' \mid \neg\phi \mid \Diamond_k(E_i)\phi \mid \mu X \phi \mid \nu X \phi$$

Theorem 9 *(Janin/Walukiewicz)*

1. *Over arbitrary structures, bisimulation invariant MSO is equal to μ-calculus*
2. *Over infinite trees trees with only the successor relation, $(T, \texttt{succ})$, MSO is equal to counting μ-calculus.*