Automata and Logic (CSC2428)
Lecture 1

Prof. Leonid Libkin
notes by Anya Tafliovich

January 13, 2005

1 Introduction

Connections between logic and automata, in particular between monadic second order logic and various finite state machines, date back to the late 1950s. They helped prove major results about automata and also led to some of the most powerful techniques for proving decidability results. More recently, they found applications in a variety of areas, such as formal methods and databases.

1.1 Course prerequisites

Some knowledge of propositional logic, predicate calculus, and string automata.

1.2 Course topics

The course covers the basics of the logic / automata connections, and looks into several CS applications.

We start with the classic results from 1950s — automata on finite strings and trees (both ranked and unranked) and relationship to monadic second-order logic and first-order logic. We then move to infinite strings and deal with Büchi automata and logics, such as monadic second-order logic, first-order logic, and linear time logic for expressing temporal properties over infinite strings. Then we consider infinite binary trees, monadic second-order logic and Rabin automata, and prove Rabin's decidability theorem. We derive some powerful results from it, in particular decidability for pushdown systems.

The rest depends on how much time we have, but ideally we should also cover branching time logic, CTL* over infinite binary trees, alternating automata and the μ-calculus, and automatic structures over strings and trees.

The topics not covered in the lectures will be left for students' presentations at the end of the course.

2 Automata on finite strings

A *non-deterministic finite state automaton* (NFA) $\mathcal{A}$ is a tuple

$$\mathcal{A} = (\Sigma, Q, q_0, F, \delta), \text{ where}$$

1. Σ is an alphabet
   Note: Σ^* is a set of (finite) strings over Σ
2. Q is a set of states
3. q_0 is an initial state
4. F is a set of final states
5. $\delta : Q \times \Sigma \to 2^Q$ is a transition function

If s is a string of length n, then λ_s is a *labeling function* from positions to members of the alphabet, $\lambda_s : \{0, \ldots, n-1\} \to \Sigma$.

For example, if s is the string *abaaba*, then the labeling function labels $\lambda_s(0) = a$, $\lambda_s(1) = b$, etc.

Then the string can be represented by a tuple $S = (\{0, \ldots, n-1\}, \lambda_s)$.

A *run* $\rho_{\mathcal{A},S} : \{0, \dots, n-1\} \to Q$ of automaton $\mathcal{A}$ on a string S is:

$$\rho_{\mathcal{A},S}(0) \in \delta(q_0, \lambda_s(0))$$
$$\rho_{\mathcal{A},S}(i+1) \in \delta(\rho_{\mathcal{A},S}(i), \lambda_s(i+1))$$

A run $\rho_{\mathcal{A},S}$ is *accepting* iff $\rho_{\mathcal{A},S}(n-1) \in F$.

An automaton $\mathcal{A}$ *accepts* a string S iff there exists an accepting run of $\mathcal{A}$ on S .

The *language* accepted by automaton $\mathcal{A}$ is $L(\mathcal{A}) = \{s \in \Sigma^* \mid \mathcal{A} \text{ accepts } s\}$.

An automaton $\mathcal{A}$ is *deterministic* iff

$$\forall q \in Q \ \forall a \in \Sigma \ \ |\delta(q, a)| \leq 1$$

i. e. for each $s \in \Sigma^*$, there is exactly one run $\rho_{\mathcal{A},S}$. Note that a transition function of a DFA is $\delta : Q \times \Sigma \to Q$.

Result: NFA = DFA (i.e. they accept exactly the same languages).

Theorem: If $\mathcal{A}$ in an NFA then there is a DFA $\mathcal{A}'$, such that $L(\mathcal{A}) = L(\mathcal{A}')$.

Proof (Power-set construction):
Suppose $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, then we construct $\mathcal{A}'$ as follows:

$$\mathcal{A}' = (\Sigma, 2^Q, \{q_0\}, F', \delta'), \text{ where } \qquad F' = \{X \subseteq Q \mid X \cap F \neq \emptyset\}$$
$$\delta' : 2^Q \times \Sigma \to 2^Q \ , \ \delta'(X, a) = \bigcup_{q \in X} \delta(q, a)$$

3 Logic background

3.1 First-order predicate calculus (FO)

3.1.1 Syntax

A *vocabulary* σ is:

1. function symbols: $f_1, f_2, \dots$ of non-negative arities $m_1, m_2, \dots$
2. constant symbols (zero-ary function symbols): $c_1, c_2, \dots$
3. predicate symbols: $P_1, P_2, \dots$ of positive arities $n_1, n_2, \dots$

FO *terms* and FO *formulae*:
(assume an infinite supply of variables)

1. each variable x is a term
   Free variables (FV) : x
2. if $t_1, \dots, t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \dots, t_k)$ is a term
   FV : $\bigcup_i \text{FV}(t_i)$

3. each constant symbol c_i is a term
   FV : none

4. if $t_1, \ldots, t_k$ are terms and P is a k-ary predicate symbol, then $P(t_1, \ldots, t_k)$ is an *atomic* formula
   FV : $\bigcup_i \mathrm{FV}(t_i)$

5. if t and t' are terms, then $t = t'$ is a formula
   FV : $\mathrm{FV}(t) \cup \mathrm{FV}(t')$

6. if ϕ and ψ are formulae and x is a variable, then $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\exists x\phi$, and $\forall x\phi$ are formulae
   FV : $\mathrm{FV}(\phi) \cup \mathrm{FV}(\psi)$, $\mathrm{FV}(\phi) \cup \mathrm{FV}(\psi)$, $\mathrm{FV}(\phi)$, $\mathrm{FV}(\phi)$ - x, and $\mathrm{FV}(\phi)$ - x, respectively

3.1.2 Semantics (Tarski semantics)

An *interpretation* (or a *structure*) $\mathcal{M}$ is:

$$\mathcal{M} = (U, f_1^{\mathcal{M}}, f_2^{\mathcal{M}}, \ldots, c_1^{\mathcal{M}}, c_2^{\mathcal{M}}, \ldots, P_1^{\mathcal{M}}, P_2^{\mathcal{M}}, \ldots)$$

where

1. U is the *universe*: a non-empty set
2. $f_i^{\mathcal{M}} : U^{n_i} \to U$, where n_i is the arity of the function symbol f_i
3. $c_i^{\mathcal{M}} \in U$
4. $P_i^{\mathcal{M}}$ of arity n_i is a subset of U^{n_i}

Each term $t(\overline{x})$, where $|\overline{x}| = m$, is assigned an element $t^{\mathcal{M}}(\overline{a}) \in U$, where $\overline{a} \in U^m$ is defined by a corresponding object assignment.

For a formula ψ, $\mathcal{M} \vDash \psi$, meaning $\mathcal{M}$ *satisfies* ψ, (under some object assignment) is defined by:

1. $\mathcal{M} \vDash t_1(\overline{a}) = t_2(\overline{a})$ iff $t_1^{\mathcal{M}}(\overline{a}) = t_2^{\mathcal{M}}(\overline{a})$
2. $\mathcal{M} \vDash P(t_1(\overline{a}), \ldots, t_k(\overline{a}))$ iff $(t_1^{\mathcal{M}}(\overline{a}), \ldots, t_k^{\mathcal{M}}(\overline{a})) \in P^{\mathcal{M}}$
3. $\mathcal{M} \vDash \neg\psi$ iff $\mathcal{M} \nvDash \psi$
4. $\mathcal{M} \vDash \psi \wedge \phi$ iff $\mathcal{M} \vDash \psi$ and $\mathcal{M} \vDash \phi$
5. $\mathcal{M} \vDash \psi \vee \phi$ iff $\mathcal{M} \vDash \psi$ or $\mathcal{M} \vDash \phi$
6. $\mathcal{M} \vDash \exists x\ \psi(\overline{a}, x)$ iff $\mathcal{M} \vDash \psi(\overline{a}, b)$ for some $b \in U$
7. $\mathcal{M} \vDash \forall x\ \psi(\overline{a}, x)$ iff $\mathcal{M} \vDash \psi(\overline{a}, b)$ for every $b \in U$

3.2 Representing strings as structures

To represent a string by a structure, we define a vocabulary:

1. a binary predicate symbol $<$ and
2. predicate symbols $P_a, P_b, \ldots$ for every $a, b, \ldots \in \Sigma$

A string $s \in \Sigma^*$, $|s| = n$, is then represented by the following structure $\mathcal{M}_s$:

$$\mathcal{M}_s = (U, <^{\mathcal{M}_s}, P_a^{\mathcal{M}_s}, P_b^{\mathcal{M}_s}, \ldots), \text{ where } U = \{0, \ldots, n-1\},$$
$$<^{\mathcal{M}_s} \text{ is a binary } < \text{ on numbers, and}$$
$$P_a^{\mathcal{M}_s} = \{i \leq n-1 | a \text{ occurs in position } i \text{ in } s\}$$

For example, consider a string *abaab*. The structure $\mathcal{M}$ corresponding to this string is:

$$(\{0,1,2,3,4\}, <, P_a, P_b), \text{ where } P_a = \{0,2,3\} \text{ and}$$
$$P_b = \{1,4\}$$

Consider another example. A property "a occurs before b in a string" can be represented by the following FO formula:

$$\exists i\ \exists j\ i < j \wedge P_a(i) \wedge P_b(j)$$

A property "every a must be preceded by a b in a string" can be represented by the following FO formula:

$$\forall i\ P_a(i) \longrightarrow \exists j < i\ P_b(j)$$

where we use the symbol $\longrightarrow$ for material implication (i.e. $(A \longrightarrow B) = (\neg A \vee B)$).

3.3 Monadic second-order logic (MSO)

Monadic second order logic adds quantification over sets to first-order calculus.

3.3.1 Syntax

We add *second-order variables*, i.e. variables that range over subsets of the universe, to the vocabulary of FO. We denote second-order variables by capital letters.

MSO terms and MSO formulae:

1. all FO terms and all FO formulae
2. if X is a second-order variable and t is a term, then $X(t)$ is a formula
3. if ψ is a formula, then $\exists X \psi$ and $\forall X \psi$ are also formulae

3.3.2 Semantics

We have an interpretation (or a structure) $\mathcal{M}$, as before.

Each term $t(\overline{x})$, where $|\overline{x}| = m$ is assigned an element $t^{\mathcal{M}}(\overline{a}) \in U$, where $\overline{a} \in U^m$ is defined by a corresponding object assignment.

Each formula $\psi(\overline{x}, \overline{X})$, where $|\overline{X}| = m$, is assigned an element $\psi(\overline{a}, \overline{S})$, where $\overline{S} = (S_1, \ldots, S_m)$, $S_i \subseteq U$ is defined by a corresponding object assignment.

For a formula ψ, $\mathcal{M} \vDash \psi$, meaning $\mathcal{M}$ *satisfies* ψ, (under some object assignment) is defined by:

1. all rules from FO
2. for $\psi(\overline{x}, X) = X(t(\overline{x}))$, $\mathcal{M} \vDash \psi(\overline{a}, S)$ iff $t^{\mathcal{M}}(\overline{a}) \in S$

3. $\mathcal{M} \vDash \exists X\ \psi(X, \ldots)$ iff there exists some $X \subseteq U$, such that $\mathcal{M} \vDash \psi(S, \ldots)$

4. $\mathcal{M} \vDash \forall X\ \psi(X, \ldots)$ iff for every $X \subseteq U$, $\mathcal{M} \vDash \psi(S, \ldots)$

Consider the following example:

Given a regular expression $(aa)^*$, find a sentence $\psi \in MSO$, such that $s \in L((aa)^*)$ iff $\mathcal{M}_s \vDash \psi$.

$$\begin{aligned}
\psi == \forall i\ P_a(i) \wedge \exists X\ & \forall j\ (\neg \exists k\ (k > j) \longrightarrow \neg X(j)) \wedge \\
& \text{(does not contain the last element)} \\
& \forall j\ (\neg \exists k\ (k < j) \longrightarrow X(j)) \wedge \\
& \text{(contains the first element)} \\
& \forall i, j\ (j > i \wedge \forall k (j > k \longrightarrow i \geq k) \longrightarrow (X(j) \longleftrightarrow \neg X(i))) \\
& \text{(contains every other element)}
\end{aligned}$$

4 Büchi 1959 results

A language $L \subseteq \Sigma^*$ is *definable* in a logic $\mathcal{L}$ if there is a sentence ψ of $\mathcal{L}$, such that

$$L = \{s \in \Sigma^* \mid \mathcal{M}_s \vDash \psi\} = L(\psi)$$

For example, if ψ is the formula from the above example, then $L(\psi) = L((aa)^*)$.

Theorem (Büchi 1959): A language is regular iff it is definable in MSO.

Proof 1 (easy direction):

Here we prove that a regular language is definable in MSO: given a regular language $L \subseteq \Sigma^*$, we need to construct a sentence $\psi \in MSO$, such that $L = L(\psi)$.

Recall that a language is regular iff it is accepted by some DFA.
Let $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ be a DFA, such that $L(\mathcal{A}) = L$.
We now produce $\psi_{\mathcal{A}}$, such that $\mathcal{M}_s \vDash \psi_{\mathcal{A}}$ iff a (unique) run of $\mathcal{A}$ on s ends in an accepting state, i.e. $\mathcal{A}$ accepts s. Assume that $Q = \{q_0, \ldots, q_{k-1}\}$. Then

$$
\begin{aligned}
\psi_{\mathcal{A}} ==& \exists X_0, \ldots, X_{k-1} \\
& \forall x \left(\bigvee_{i=0}^{k-1} X_i(x) \wedge \neg \bigvee_{i \neq j} \exists (X_i(x) \wedge X_j(x)) \right) \\
& (X_i\text{'s partition the universe}) \\
\wedge \ & \forall x \left((\forall y \ x \leq y) \longrightarrow \bigwedge_{a \in \Sigma} (P_a(x) \longrightarrow X_i(x)) \right) \ \text{, where } q_i = \delta(q_0, a) \\
& (\text{position 0 is in } X_i\text{, where } q_i = \delta(q_0, a)\text{, where } a \text{ is the first symbol}) \\
\wedge \ & \forall x \forall y \left((x < y \wedge \neg \exists z (x < z \wedge z < y)) \longrightarrow \bigwedge_{\substack{a \in \Sigma \\ i \leq k}} (X_i(x) \wedge P_a(y) \longrightarrow X_j(y)) \right) \\
& \quad \text{, where } q_j = \sigma(q_i, a) \\
& (\text{if position } m \text{ is in } X_i\text{, the symbol in position } m+1 \\
& \text{is } a\text{, then position } m+1 \text{ is in } X_j\text{, where } q_j = \delta(q_i, a)) \\
\wedge \ & \forall x \left(\forall y (y \leq x) \longrightarrow \bigvee_{q_i \in F} X_i(x) \right) \\
& (\text{if } m \text{ is the last position, then } m \in X_i\text{, where } q_i \in F)
\end{aligned}
$$

Proof 2 (hard direction):

Here we prove that a language definable in MSO is regular. There are two complications: (1) we have two kinds of variables (first- and second-order) and (2) we cannot do induction on sentences (not formulae!)
We cope with the first complication by introducing MSO_0, in which:

1. all variables are second-order
2. $Sng(X)$ iff $|X| = 1$
3. $Less(X, Y)$ iff $X = x$, $Y = y$, and $x < y$
4. $X \subseteq Y$

Fact: $\text{MSO} = \text{MSO}_0$

We will deal with the second complication in the next lecture