Logic and Computation in Finitely-Presentable Infinite Structures

Lecture 2: Formal languages and automata

Valentin Goranko and Sasha Rubin

ESSLLI 2006, Malaga, August 2006
Formal Languages

Words
Trees
Automata accepting sets
Automata accepting relations
Words

- alphabet $\Sigma$: finite set of symbols, usually $\{0, 1, \cdots, k - 1\}$, $k \geq 2$; write $2$ for $\{0, 1\}$
- word: sequence over $\Sigma$
- finite words are denoted $u, v, w \cdots$, and infinite words (also called $\omega$-words) are denoted $\alpha, \beta, \gamma, \cdots$
- empty word $\lambda$, length $|u|$, concatenation $uv$ or $u\alpha$. 
Languages of words

- A set of words is called language,
- a set of infinite words is called an $\omega$-language,
- concatenation $UV$, power $U^* := \cup_n U^n$,
- $\omega$-power $U^\omega := \{ \alpha \mid \alpha = u_1 u_2 \cdots , u_i \in U \}$,
- all finite words: $\Sigma^*$, all infinite words: $\Sigma^\omega$. 
Operations on finite words

On finite words:

1. for each $a \in \Sigma$, the $a$-successor function $s_a : u \mapsto au$,
2. the prefix-relation $u \preceq_{\text{prefix}} v$ :if $v = uz$ for some $z \in \Sigma^*$,
3. the equal-length relation $=_{\text{len}} (u, v) :\text{if } |u| = |v|$.

$\mathcal{W}(\Sigma) := (\Sigma^*; (s_a)_{a \in \Sigma}, \preceq_{\text{prefix}}, =_{\text{len}})$.

We can define the following using FO in $\mathcal{W}(2)$:

- the set of words of length a multiple of $k$ (for fixed $k \in \mathbb{N}$).
- the set of words with an even number of 0s.
- the binary relation $u[|w|]$ iff the $|w|$th symbol of $u$ is 1.

**Think about.** What is FO-definable in $\mathcal{W}(2)$?
Operations on infinite words

On infinite words, identify $u$ with $\overrightarrow{u} := u10^\omega$.

1. $s_a : \Sigma^* \rightarrow \Sigma^*$ defined by $s_a(\overrightarrow{u}) := \overrightarrow{ua}$,
2. $\overrightarrow{u} \preceq_{\text{prefix}} \beta$ :if $u$ is a prefix of $\beta$,
3. $\equiv_{\text{len}} (\overrightarrow{u}, \overrightarrow{v})$ :if $|u| = |v|$.

$$\mathcal{W}^\omega(\Sigma) := (\Sigma^\omega; (s_a)_{a \in \Sigma}, \preceq_{\text{prefix}}, \equiv_{\text{len}}).$$

We can define the following using FO in $\mathcal{W}^\omega(2)$:

- the set of $\alpha$ with infinitely many 1s.
- binary relation $\beta[\overrightarrow{w}] = 1$ iff the $|w|$th symbol of $\beta$ is 1.

In lecture 3. What is FO-definable in $\mathcal{W}^\omega(2)$?
Trees

We consider vertex-labelled binary-branching trees.
Fix a labelling alphabet $\Sigma$.

- $\omega$-tree $T : \{0, 1\}^* \rightarrow \Sigma$,
- the set of $\omega$-trees $\text{trees}^\omega(\Sigma)$,

In general,
- tree partial function $T : \{0, 1\}^* \rightarrow \Sigma$ with prefix-closed domain $\text{dom}(T)$,
- a leaf is a maximal element of $\text{dom}(T)$,
- a tree is called finite if $\text{dom}(T)$ is finite,
- the set of finite trees $\text{fintrees}(\Sigma)$,
- the outer boundary of a finite tree $T$ are those $w \notin \text{dom}(T)$ whose immediate predecessor is in $\text{dom}(T)$. 
Operations on trees

On finite trees:

1. \( s^d_a(T) \) extends every leaf of \( T \) in direction \( d \in \{0, 1\} \) and labels each new vertex with the symbol \( a \),

2. \( T \preceq_{\text{ext}} T' \) if \( T' \) extends \( T \) (as functions), and

3. \( T \equiv_{\text{dom}} T' \) if they have the same domain.

With a finite tree \( T \) associate the \( \omega \)-tree \( \overrightarrow{T} \) extending \( T \) with value 1 on the outer boundary, and value 0 everywhere else.

Extend the tree operations to infinite trees as for infinite words. For instance \( s^d_a : \overrightarrow{T} \mapsto \overrightarrow{s^d_a(T)} \).

- \( \mathcal{T}(\Sigma) := (\text{fintrees}(\Sigma), (s^d_a)_{d \in \{0, 1\}, a \in \Sigma}, \preceq_{\text{ext}}, \equiv_{\text{dom}}) \).
- \( \mathcal{T}^\omega(\Sigma) := (\text{trees}^\omega(\Sigma), (s^d_a)_{d \in \{0, 1\}, a \in \Sigma}, \preceq_{\text{ext}}, \equiv_{\text{dom}}) \).
Summary of structures

- \( \mathcal{W}(\Sigma) := (\Sigma^*; (s_a)_{a \in \Sigma}, \preceq_{\text{prefix}}, =_{\text{len}}) \).
- \( \mathcal{W}_\omega(\Sigma) := (\Sigma_\omega; (s_a)_{a \in \Sigma}, \preceq_{\text{prefix}}, =_{\text{len}}) \).
- \( \mathcal{T}(\Sigma) := (\text{fintrees}(\Sigma), (s^d_a)_{d \in \{0,1\}, a \in \Sigma}, \preceq_{\text{ext}}, \equiv_{\text{dom}}) \).
- \( \mathcal{T}_\omega(\Sigma) := (\text{trees}_\omega(\Sigma), (s^d_a)_{d \in \{0,1\}, a \in \Sigma}, \preceq_{\text{ext}}, \equiv_{\text{dom}}) \).
Models of computation

**Find.** Class of languages that are (effectively) closed under Boolean operations, projection, etc.

- A language $L \subseteq \Sigma^*$ is *recursive or computable* if there is some Turing machine $M$ such that $M$ accepts $x \in \Sigma^*$ if and only if $x \in L$.
- Closed under Boolean operations, but not projection.

Resource bounded Turing Machines.

- Restrict to TM’s run in certain time/space in the size of the input: Polynomial time, linear space, etc.
- Do we have closure properties? Related to open questions (eg P vs. NP).

Finite automata have effective closure properties.

- Classically, finite automata operate on finite words.
- Finite automata operating on infinite words and trees.
Automata operating on $\omega$-words

A Büchi automaton is a tuple $\langle \Sigma, S, \iota, F, \Delta \rangle$, where:

- $\Sigma$ is the alphabet
- $S$ is a finite set of states
- $\iota \in S$ is the initial state
- $F \subseteq S$ is a set of accepting states
- $\Delta \subseteq S \times \Sigma \times S$ is a transition function.

Input: infinite word $\alpha \in \Sigma^\omega$.

Run: infinite word $\rho \in S^\omega$ so that $\rho[0] = \iota$ and for every $n \in \mathbb{N}$, $\Delta(\rho[n], \alpha[n], \rho[n+1])$. 
Acceptance condition for Büchi automaton

\( \inf(\rho) \): the set of states that occur infinitely often in \( \rho \).

A word \( \alpha \) is accepted by the automaton if some run \( \rho \) on \( \alpha \) satisfies \( \inf(\rho) \cap F \neq 0 \) (Büchi condition).

**Examples** Fix \( \Sigma = \{0, 1\} \).

- The set of \( \omega \)-words with infinitely many 1s.
- The set of \( \omega \)-words with finitely many 1s.
- The set of \( \omega \)-words with an even number of 1s.

An \( \omega \)-language is called *regular* if it is exactly the set of \( \omega \)-words accepted by some Büchi automaton.
An $\omega$-language is called \textit{regular} if it is exactly the set of $\omega$-words accepted by some Büchi automaton.

\textbf{Aim.} Define regular relations $R \subset (\Sigma^\omega)^r$ that have closure properties.

Turn $R$ into $\otimes R \subset (\Sigma^r)^\omega$ and then run an automaton with larger alphabet $(\Sigma^r)$.

- $\otimes(0^\omega, 1^\omega) := (0^1)^\omega$.
- $\otimes(01^\omega, 1^\omega) := (0^1)1^1)^\omega$.

Convolution $\otimes : (\Sigma^\omega)^r \rightarrow (\Sigma^r)^\omega$ where $\otimes(\alpha)[i] = (\alpha_1[i], \cdots, \alpha_r[i])$ for every $i \in \mathbb{N}$. Define $\otimes(R)$ as $\{\otimes(\alpha) \mid \alpha \in R\}$.

A relation $R$ of $\omega$-words is \textit{regular} (or \textit{\omega-regular}) if $\otimes(R)$ is regular.
Examples of regular relations

For a set $X \subseteq \mathbb{N}$, can think of $\alpha_X \in 2^\omega$ as its characteristic string.

- $\{\alpha_X \mid X \text{ is finite}\}$.
- $\{(\alpha_X, \alpha_Y) \mid X \subseteq Y\}$

Can think of $\alpha \in 2^\omega$ as the binary representation of a real number $r_\alpha$ in $[0, 1]$.

- $\{(\alpha, \beta) \mid r_\alpha \leq r_\beta\}$.
- $\{\alpha \mid r_\alpha \leq q\}$, for a fixed rational $q \in [0, 1]$.
- $\{(\alpha, \beta, \gamma) \mid r_\alpha + r_\beta = r_\gamma \mod 1\}$. 
Büchi automata and closure properties

**Proposition.** Regular relations are effectively closed under Boolean operations and FO logical operations.

That is, if $\otimes(R), \otimes(R')$ are regular, then so is $\otimes(T)$ where $T$ is one of:

- $R \cap R'$, $R \cup R'$, $R \setminus R'$;
- the projection of $R$; cylindrification of $R$; permutation of co-ordinates of $R$;
- instantiation of $R$ by an ultimately periodic sequence.

Moreover, an automaton for $\otimes(T)$ can be constructed from automata for $\otimes(R)$ and $\otimes(R')$.

(Intersection requires a little reflection, and complementation requires a few ideas Büchi (1962).)
Büchi automata and testing emptiness

**Proposition.** There is an algorithm that given a Büchi automaton, decides whether or not it accepts the empty language.

(Sketch) Test for a loop containing an accepting state, reachable from the initial state.