Automata and Logic (CSC2428)
Lecture 9

Prof. Leonid Libkin
notes by Eric Hsu

March 17, 2005

1 Overview

In this lecture we will review the games-based proof of Rabin's complementation theorem, which is used to prove his theorem that regular languages over infinite binary trees are MSO-definable.

From there we will consider the associated MSO theory of S2S, exhibiting that sentences from S2S can be translated into "input-free" automata. We will show that it is decidable whether a given tree automaton accepts any inputs, both for such input-free automata, and also in general. The argument is once again based on games, with the end result being that S2S is decidable.

We will then show decidability for several other theories by specifying isomorphisms to S2S, namely the theory of rational numbers under ordering, the theory of SnS (trees with fixed but arbitrary branching), and the theory of SωS (trees with infinite branching). Another such theory will be that of push-down automata and their associated reachability graphs.

Finally, we will compare S2S to weak S2S, which is restricted to bounded quantification. (Weak S2S is a proper subset of S2S, unlike to the previous result where weak S1S was proved equivalent to S1S.)

2 Games and Decidability

2.1 Games

A *game* $\mathcal{G}$ is a tuple

$$\mathcal{G} = (V_0, V_1, E, C, c, W), \text{ where}$$

- V_0 and V_1 are sets of vertices.
- $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$ represents transitions between the two sets of vertices.
- C is a set of colors,
- $c : V_0 \cup V_1 \to C$ is a coloring funtion mapping vertices to colors,
- W is a winning condition of form specified below.

To play the game, Player 0 starts at some initial $v_0 \in V_0$, and follows an edge to one of the elements of V_1. From there, Player 1 will make a move, and so on. An instance $s \in C^\omega$ of playing such a game is an infinite string of colors, derived from the string of vertices selected by the alternating players. That is, a record of the moves would read:

$$v_0 v_1 v_2 v_3 \ldots \text{ where } \langle v_i, v_{i+1} \rangle \in E, v_{2i} \in V_0, \text{ and } v_{2i+1} \in V_1.$$

And in turn, s would equal $c(v_0)c(v_1)c(v_2)c(v_3)\ldots$ by application of the coloring function to the sequence of vertices.

The winning condition W consists of a sequence of expanding sets of colors, as depicted below:

$$E_0 \subset F_0 \subset E_1 \subset F_1 \subset \ldots \subset E_l \subset F_l \subset C$$

A instance $s \in C^\omega$ of playing the game satisfies the winning condition if the first color that occurs infinitely often falls into an F set instead of an E set:

$$s \in C^\omega \text{ satisfies } W \iff \exists k. Inf(s) \cap E_k = \emptyset \text{ and } Inf(s) \cap F_k \neq \emptyset$$

Player i has a winning strategy from position v if i can play in such a way that the resulting $s \in C^\omega$ always satisfies W. Further, Player i has a *"memoryless"* winning strategy if the above holds such that Player i's move only depends on the node v on which the player moves (as opposed to depending on the history of moves.)

2.2 Connection to Determinacy, Decidability

Theorem. *Determinacy Theorem*: From every position $v \in V_0 \cup V_1$, either Player 0 or Player 1 has a memoryless winning strategy.

For us, this result applies in the context of the "Automaton-Pathfinder Game." The basic idea is that one player is the automaton, which gets to choose the next (pair of) states given the transition function and the current state/symbol. The other player, called the pathfinder, gets to choose which of the (two) children will constitute the next state, and the process repeats. The format of such an automaton is specified below; the corresponding game played over a given tree T would be denoted $\mathcal{G}_{\mathcal{A},T}$.

$$\text{Automaton } \mathcal{A} = (Q, q_0, \Delta, Acc)$$
$$\Delta \subseteq Q \times \Sigma \times Q \times Q$$
(given a state and symbol, Δ offers choices for pairs of successor states.)

The automaton has a winning strategy if it can gurarantee the repeated future appearance of a winning symbol, no matter how the pathfinder chooses; this is like accepting the tree. By the Determinacy Theorem, either this condition holds, or else the pathfinder can always win, signifying the rejection of the tree.

During the last lecture we saw Rabin's Complementation Theorem in terms of such a game. The automaton rejects the tree iff it has no winning strategy starting at the root, as it would have to meet the acceptance condition over all possible paths. By the Determinacy Theorem, this is equivalent to the pathfinder having a winning strategy. Thus, the crux of Rabin's proof was to provide a construction for an automaton representing the play of the pathfinder. That is, the pathfinder itself represents an automaton, namely the complementary automaton that accepts the complement of the language.

2.3 Regular Languages over Infinite Binary Trees are MSO-Definable

Complementation was the only real roadblock in transferring all the usual constructions for MSO representations of infinite strings to the case of infinite binary trees. Taking care of this complication yielded the following result:

Theorem. A set of infinite binary trees is regular $\iff$ it is MSO-definable.

So, if $\Phi(X_1...X_m)$ is an MSO formula (with signature ranging over $succ_0$ and $succ_1$, the succesor relations for left and right branching), then we can construct some $\mathcal{A}_\Phi$ over alphabet $\{0,1\}^m$ such that $\Phi(X_1...X_m)$ is true iff $\mathcal{A}_\Phi$ accepts the tree $T[X_1...X_m]$.

$T[X_1...X_m]$ denotes an infinite binary tree where nodes are labeled by strings of 0's and 1's, indicating whether that node belongs to the various X_i. More formally:

$$label(s) = (\sigma_1...\sigma_m) \text{ where } \sigma_i = 1 \iff s \in X_i$$

3 Decidability of S2S

The above efforts move us closer to proving the decidability of S2S, the MSO theory of infinite binary trees. Decidability will be based on the games framework again, and the demonstration of a winning strategy for the automaton player corresponding to a given S2S sentence.

3.1 S2S

Recall that S1S was presented as the MSO theory of $(\mathbb{N}, succ)$. Now we are reaching further, to handle infinite binary trees: S2S is the MSO theory of $(\{0,1\}^*, succ_0, succ_1)$. That is, the universe of discourse is infinite strings of 0's and 1's encoding trees, and the predicates specify a node's left and right descendants.

By the methodology shown in the preceeding sections, we already know that each S2S formula $\Phi(\vec{X})$ is effectively translated into an automaton $\mathcal{A}$. So now the goal is to show that S2S is a decidable theory.

3.2 Input-Free Automata

The first step will be to note that every sentence Φ in S2S is effectively translated into an "input-free" automaton. Essentially, this is an automaton that does not bother to read labels:

$$\text{Input-Free Automaton } \mathcal{A} = (Q, q_0, \Delta \subseteq Q \times Q \times Q, Acc).$$

To check whether a sentence is true, then, we just need to check whether its input-free automaton accepts the unlabeled tree. Thus decidability of a theory is reduced to checking whether $\mathcal{L}(\mathcal{A}) = \emptyset$ for some automaton $\mathcal{A}$.

3.3 Decidability of Language-Emptiness

Well, is the above question decidable? We will show that it is, in fact we will go further and show it is decidable for all tree automata, not just input-free tree automata.

Theorem. Given a tree automaton $\mathcal{A}$, it is decidable whether $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

Corollary (Rabin 1970). S2S is decidable. (We would just check whether or not the unlabeled tree is accepted by the appropriate automaton.)

3.3.1 Input-Free Decidability

We will begin the proof of the limited form of the theorem by specifying the input-free automaton in question and placing it in the context of a game versus a pathfinder.

Let Input-Free Automaton $\mathcal{A}_0 = (Q, q_0, \Delta \subseteq Q \times Q \times Q, Acc)$.
In game $\mathcal{A}_0$, the automaton begins by choosing some $(q_0, q_1, q_2) \in \Delta$.
The pathfinder continues by choosing q_1 or q_2 as the next state, and so on...

Thus, the automaton wins iff there is always an accepting run of $\mathcal{A}_0$. As before, the Determinacy Theorem shows that there is a memoryless winning strategy for either the automaton or the pathfinder from every position–here we care only about the root. So we want to show that it is decidable to determine who is the winner starting from the root.

Lemma. For any game, if $|V_0 \cup V_1| < \omega$ then for every $v \in V_0 \cup V_1$ it is decidable whether Player 0 wins from v, with complexity $NP \cap coNP$. (The complexity is both NP and $coNP$ because if one player wins, then certainly the other player doesn't.)

Proof. We will show that the question is NP via the nondeterministic prover/polynomial-time verifier route. So the first step will be to simply guess a winning strategy for Player 0, the automaton. For each $v_0 \in V_0$, keep exactly one of the edges leading out from v_0 in the graph representing the game. That is, we take away all of Player 0's choices save one. On ther other hand, we retain all the edges from any $v_1 \in V_1$; we keep all of the choices for the pathfinder.

So now we need a polynomial method for checking whether Player 1 wins. This is easy now that Player 0 no longer has any options. The verifier just checks, in polynomial time, whether Player 1 can reach some v whose color is a member of an E set but not the corresponding F set. That is, it checks whether $c(v) \in E_i - F_{i-1}$. If he can do so repeatedly, then he wins. This concludes the proof of the lemma.

3.3.2 General Decidability

The above already gives us the decidability of S2S, as it can be formulated as an input-free automaton. Incidentally, though, the full theorem holds, even over labeled trees and their associated automata. The proof is by extending the above argument, essentially packing the alphabet into the state space.

Specifically, consider some automaton $\mathcal{A}$ operating on trees labeled by alphabet Σ: (Q, q_0, Δ, Acc). We construct an equivalent input-free automaton $\mathcal{A}' = (Q \times \Sigma, q_0 \times \Sigma, \Delta', Acc')$. Here the states contain the various possible labels that might have appeared at a given node, and hence there are several possible initial states (depending on the label of the root node). The new transition function is defined as follows:

$$\begin{gathered}\langle q_0, a, q_1, q_2\rangle \in \Delta \\ \implies \langle\langle q_0, a\rangle, \langle q_1, a'\rangle, \langle q_2, a''\rangle\rangle \in \Delta' \\ \text{for } \textit{any } a', a'' \in \Sigma.\end{gathered}$$

That is, the new input-free automaton essentially guesses the tree that A would have accepted. The acceptance condition Acc' is the same as before, just similarly modified to ignore the Σ portion of the new state space.

4 Aside: Applying Games to Automata

Consider how infinite strings all contain one aperiodic string. Now we will consider a similar phenomenon with infinite trees. (The motivation for such a consideration is the unfolding of a Kripke structure.)

An infinite Σ-labeled tree is *regular* if it is generated by some DFA $(Q, \{0,1\}^*, q_0, \delta, Acc)$ together with some function $f : Q \to \Sigma$ that labels the sequence of states. In other words, the label of a given s is $f(\delta(q_0, s))$. (For this particular setting, we should just forget about the acceptance condition Acc.)

By characterizing automata as players in games, as we have been doing above, the following result can be proved:

Theorem. If $\mathcal{L}(\mathcal{A}) \neq \emptyset$ then it contains a regular tree.

The proof of the theorem is left as an exercise.

5 Decidability of Other Theories, Based on Decidability of S2S

Decidability for many additional theories has been and will continue to be proved by reduction to S2S. We will define such a reduction and show several examples of its application.

5.1 Reduction to S2S

S2S is the MSO theory of $T^\omega = (\{0,1\}^*, succ_0, succ_1)$. Now consider some other structure (U, Ω), where U is the universe of discourse and Ω is the set of relation symbols. Examples of such structures could be $(\mathbb{N}, +)$ or $(\mathbb{Q}, <)$, or whatever. As further notation, let Ω consist of relation symbols $R_1 ... R_j$ and let each relation R_i be of arity n_i.

Then let $\Phi_0(X_0), ..., \Phi_j(X_j)$ be a series of S2S formulas, with arities $|X_i| = n_i$. Further, let $\Phi(\vec{x})$ be some *first order* formula, and denote its set of satisfying trees by $\Phi(T^\omega)$, as follows:

$$\Phi(T^\omega) = \{s \in T^\omega \mid T^\omega \vDash \Phi(s)\}$$

Then to *reduce* (U, Ω) to S2S, we just need to show the following isomorphism:

$$(\Phi(T^\omega), \{\Phi_i(T^\omega)\}_i) \cong (U, \Omega)$$

As we have already shown the decidability of S2S, exhibiting such a reduction proves the decidability of (U, Ω). To elaborate, you could just interpret the theory in S2S, mapping its universe of discourse to trees satisfying Φ and simulating its various relations with the various Φ_i's. In fact, you can use the terminology that (U, Ω) is *interpretable* in S2S.

5.2 Example: $(\mathbb{Q}, <)$

The rational numbers under ordering can be interpreted in S2S via the following isomorphism:

$$(\mathbb{Q}, <) \cong (\{0,1\}^* 1, <_{lex})$$

Thus, the theory is decidable; we could answer any query in its language by interpreting the query as the equivalent S2S formula.

5.3 Example: n-ary Trees

We can expand our theory to cover n-ary trees, where nodes have n children. Let T_n^ω denote such a theory, which can also be called SnS rather than S2S:

$$T_n^\omega = (\{0, ..., n-1\}^*, \{succ_0, ..., succ_{n-1}\})$$
$$\text{where } succ_i(s) = s \circ i.$$

It turns out that SnS is also decidable, as it can be interpreted in S2S by encoding its universe as follows:

$$\text{Universe } U_n = (01 + 001 + ... + 0000...1)^*$$

The first term in the regular expression corresponds to the 1st successor of a node, and the second corresponds to the second successor. The final term, which consists of n 0's followed by a 1, corresponds to the nth successor of a node.

Accordingly, the various $succ_i$ relations can be designed to recognize this encoding, by parsing out the 0's and 1's. In fact, they are even definable in *first-order* logic.

5.4 Example: Infinite-branching Trees

Why stop at a fixed n? The theory of infinite-length, infinitely branching trees, denoted T^ω_ω, is also decidable. Here we will not want to deal with denoting each of the infinite number of succesor functions, and will instead define SωS as follows:

$$T^\omega_\omega = (\mathbb{N}^*, <, <_{lex}) \text{ where}$$
$$s < s \circ i,\ i \in \mathbb{N} \text{ and}$$
$$\langle \textit{left sibiling} \rangle <_{lex} \langle \textit{right sibling} \rangle.$$

That is, $<$ relates a node to any of its successors, and $<_{lex}$ defines the ordering between the various successors of a given node.

This theory is interpretable in S2S by slightly extending the previous example. As before we encode the universe of discourse, U_ω, by $(0^+1)^*$. That is, the ith child of a node is denoted by the suffix consisting of i 0's followed by a 1.

The definitions for $<$ and $<_{lex}$ are left as an exercise. The basic idea is that the succesor relation allows you to define an MSO descendant formula, which leads to everything else.

5.5 Example: Push-Down Automata Reachability

The next example reduction will concern push-down automata, specifically their reachability graphs. We will define both before proceeding to show the isomorphism.

5.5.1 Push-Down Automata and Their Reachability Graphs

A pushdown automaton $\mathcal{P}$ is basically a regular automaton augmented with a stack:

$$\mathcal{P} = (Q, \Sigma, \Gamma, q_0, \gamma_0, \delta, A), \text{ where}$$

- Q is a set of states, Σ is the tape alphabet, and q_0 is the initial state, as before. The acceptance condition, A, will be of no concern to us here.
- Γ is the stack alphabet.
- γ_0 will be the lone, initial character on the stack.
- $\delta : Q \times \Sigma \times \Gamma \to 2^{Q \times \Gamma^*}$ is the transition function. Given the current state, tape symbol, and top symbol on the stack, it chooses a new state and pushes some string onto the stack.

A *configuration* of such a $\mathcal{P}$ represents a particular stage of the computation. It is represented by string of the form sq where s is the contents of a stack, and q is the current state state. (In writing out the contents of the stack, it is read from the bottom up; the first character of s will be γ_0.) Thus, for instance, the intial configuration is $\gamma_0 q_0$.

A configuration is *reachable* if it can arise from the initial configuration by some run of the PDA on some string in Σ^*. Thus we can define a *reachability graph* $G_{\mathcal{P}} = (R_{\mathcal{P}}, E_{\mathcal{P}})$, representing which configurations are reachable from which:

$$R_{\mathcal{P}} : \text{ the set of reachable configurations.}$$
$$E_{\mathcal{P}} \subseteq R_{\mathcal{P}} \times R_{\mathcal{P}} : \text{ edge relation on } R_{\mathcal{P}}.$$

$E_{\mathcal{P}}$ is easy to understand but a little convoluted to specify:

$$(s\gamma q, ss'q') \in E_{\mathcal{P}}$$
$$\Longleftrightarrow$$
$$(q', s') \in \delta(q, a, \gamma) \text{ for some } a.$$

The first configuration in the first pair shows the automaton in state q with the stack consisting of some prefix s followed by the single topmost character γ. If the transition function allows a move to some new configuration on some tape character a, there is a (directed) edge to that configuration. In short, the reachability graph shows all the reachable configurations, and how to get from one to the other. We will refer to this graph as the "pushdown graph" for $\mathcal{P}$.

5.5.2 Decidability of Pushdown Graphs

Theorem. (Müller-Schupp) The MSO theory of every pushdown graph is decidable.

Typically, the point of the push-down automaton is to model recursive procedure calls. The theorem shows that it is decidable to model the possible evolutions of the procedure stack by querying the pushdown graph.

To prove the theorem, we will interpret $G_{\mathcal{P}}$ in SnS, where n $= |Q| + |\Gamma|$ is the size of the state space plus the size of the stack alphabet. It is easy to define whether some pair $(s\gamma q, ss'q') \in E_{\mathcal{P}}$, because you only have to reason about the last few characters. We can complete the construction by defining $R_{\mathcal{P}}$ in terms of $E_{\mathcal{P}}$:

$$\Phi_0(X) \equiv X(\gamma_0 q_0) \wedge \forall_{xy}(X(x) \wedge E_{\mathcal{P}}(x,y) \supset X(y))$$
$$\text{and}$$
$$\Phi_{R_{\mathcal{P}}}(x) \equiv \exists X(\Phi_0(X) \wedge \forall Y(\Phi_0(Y) \supset X \subseteq Y) \wedge X(x)).$$

The formula indicates "the minimal set that contains the initial configuration $(\gamma_0 q_0)$ and is closed under $E_{\mathcal{P}}$." This completes the isomorphism to SnS, and hence proves completeness for the theory of pushdown graphs.

5.6 Example: Prefix-Recognizable Graph

A final example of a theory that can be interpreted in S2S is that of prefix-recognizable graphs. Let Σ be some alphabet, and $R \subseteq_{finite} \Sigma^* \times \Sigma^*$ be some finite set of pairs of strings over Σ. Further, let L be some regular language.

Then we can define the relation E_R^L as follows:

$$E_R^L = \bigcup_{\langle s1, s2\rangle \in R} \{(ss_1, ss_2) \mid s \in L\}$$

In short, E_R^L holds over pairs of strings that are made by attaching some prefix from L to each of two words that are paired by R. We state without proof a theorem by Caucal, that the MSO theory of (Σ^*, E_R^L) is decidable, by reduction to S2S.

6 S2S vs. Weak S2S

Recall from previous lectures that weak S1S, denoted wS1S, was the theory of S1S restricted to allow quantification over only finite sets. Further, recall that S1S was shown to be equivalent to wS1S. What about the relationship between S2S and wS2S?

6.1 S2S and wS2S are not equivalent

First of all, we can show that wS2S $\subseteq$ S2S; it is possible to define the finiteness of a set. The specific construction is supposed to be widely-known, and is left as an exercise.

Rabin proved a theorem that helps address the opposite direction, concerning the strictness of this subset relation:

Theorem. (Rabin) A set L of Σ-labeled trees is definable in weak MSO iff both L and its complement are accepted by Buchi tree automata.

Note that the theorem concern's Buchi's acceptance condition, which is weaker than Rabin's or Müller's. The following corollary follows directly, and settles the issue of equivalence:

Corollary. wS2S $\subsetneq$ S2S; that is, it is a proper subset.

6.2 Equivalence When Restricted to FO Free Variables

So how dow we define a class of trees in S2S? Let $|\Sigma| = k$. Then we would use some formula of the form $\Phi(X_1, ..., X_k)$. With respect to such formulas $\Phi(\vec{X})$ with free 2nd-order variables, we have seen that weak MSO is weaker than MSO over T_2^ω (the set of infinite binary trees).

What about formulas $\Phi(\vec{x})$ that only range over 1st-order free variables? In this case, are wMSO and MSO the same for T_2^ω? Rabin proved that they are, in the 70's, using a lengthy proof that involved mostly logic. It is left as an exercise to prove the result using E-F games, in the restricted case where there is *one* free variable.