The Relevance of Recognizable Sets

Written Preliminary Exam II

Anoop Sarkar
Department of Computer Science
University of Pennsylvania, Philadelphia, PA 19104
anoop@linc.cis.upenn.edu

November 1, 1995

Abstract

This paper is a review of several results that connect the areas of theoretical computer science (particularly automata theory and context-free grammars) and logic (particularly decidability results of the second order theories). In particular, the properties of the set of trees accepted by finite state tree acceptors - also called recognizable sets are studied. The results presented are interesting because they give rise to an alternate characterization of the notion of a context-free grammar resulting in a descriptive complexity result for the theory of context-free languages and on the other hand, the results presented give a novel decidability result for the theory of n successor functions, the novelty lying in the use of tools from automata theory for the proof. And so, while this paper looks at applications of automata over trees and other objects to fields such as logic and context free grammars, the objective is to highlight how looking at these question through this way brings together notions from seemingly disparate fields. The focus will be to explicate these results to show how recognizable sets can be viewed as useful both to the theory of context free grammars and to decision problems of second order theories in logic.

Contents

1 Introduction

First of all, what are *recognizable sets*? They are essentially the subset of the domain of trees that are accepted by *tree acceptors*. Tree acceptors are like conventional finite automata, except for the fact that they accept trees rather than words as their input. This paper then explores the notion of a recognizable set and the relevance of tree acceptors to various areas of interest.

For this reason, this paper presents a series of results that intersect the areas of theoretical computer science (particularly automata theory and context-free grammars) and logic (particularly decidability results of the second order theories). These results are interesting because they give rise to an alternate characterization of the notion of a context-free grammar resulting in a descriptive complexity result for the theory of context-free languages and on the other hand, the results presented give a novel decidability result for the theory of n successor functions, the novelty lying in the use of tools from automata theory for the proof. And so, while this paper looks at applications of automata over trees and other objects to fields such as logic and context free grammars, the objective is to highlight how looking at these question through this way brings together notions from seemingly disparate fields. The focus will be on the pivotal insights that help us towards this goal and to explicate these results to show how recognizable sets can be viewed as useful both to the theory of context free grammars and to decision problems in second order theories in logic.

Figure 1 shows the relation of recognizable sets to its application to the areas of interest to this paper and how it creates a bridge between, for instance, the derivation tree sets of context-free grammars and definability in a monadic second order theory (SnS).

Some sections of this paper can also be viewed as examining the utility of exploring generalizations of automata theory and covers some applications of such extensions either as automata over trees or as automata over infinite sequences.

The paper is organized as follows: after setting up the basic notation and preliminary concepts to which the remainder of this section is devoted, we look at tree acceptors and their relation to context free languages in Section 2. This section also introduces the concept of a recognizable set.

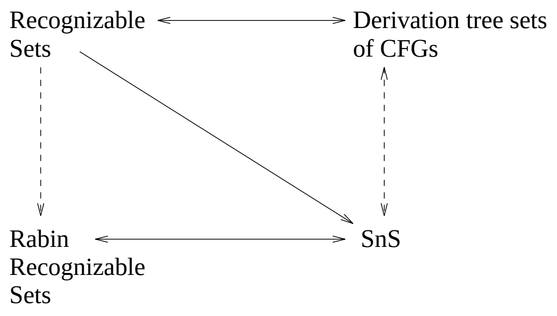

Figure 1: Relevance of Recognizable Sets

In Section 3 we look at the extension of automata theory to the domain of infinite objects and its usefulness to the decision problems of certain second order theories. In particular, we look at extensions of automata which accept infinite sequences of letters, otherwise known as w-words, in the theory of Büchi automata which is covered in Section 3.1. The decidability result for second order theories that have 1 successor function (also denoted by $S1S$) which uses these Büchi automata is seen in Section 3.2.

Section 4 extends the results from the previous section to a decidability result for second order theories with more than 1 successor function. The notion of a Rabin tree automaton and consequently the Rabin recognizable sets is explored in Section 4.1. These results give us the necessary tools with which we can answer the more general decision problems for second order theories with n successor functions for any n (also called SnS). This is covered in Section 4.2.

We complete the cycle of results in Section 4 by looking at the application of recognizable sets to the decision problems discussed in Section 3. We summarize and point out the relation between these different approaches in Section 5.

1.1 Preliminaries

Let us first set up the basic notions and notation that we will follow in this report. We shall employ the standard set theoretical notions: $\cup, \cap, \in$, etc. $A - B$ denotes the difference of sets A, B, *i.e.,* $A - B = \{x : x \in A \text{ and } x \notin B\}$. Let $\mathbb{N}$ be the set of natural numbers (the finite ordinals $0, 1, 2, \ldots,$) and set of all such natural numbers is the first infinite ordinal

ω. The cardinality of a set A is denoted by $|A|$.

If a function is defined for each element of a class K and $A \subseteq K$ then $f(A) = \{f(x) : x \in A\}$. The domain of a function f is denoted by $dom(f)$. Assertions like "C is the class defined by the conditions ..." or "C is the least class such that ..." are to be intertpreted to mean that C is the intersection of all classes satisfying the stated conditions.

Our notation for automata, words, languages, etc., is taken, for the most part from [8] and [20]. An alphabet A is a nonempty finite set of symbols (or *letters*). Unless otherwise stated, the letter Σ or the letters $A, B, C, \ldots$ denote alphabets.

A *word* over A, or simply a *word* when A is understood from context is a finite sequence of elements of A. A word with only one letter a is identified with a itself; ϵ denotes the empty word and concatenation of words is indicated by juxtaposition (so that $a\epsilon = a$). Usually, the initial letters $a, b, c, \ldots$ are used for single elements of the alphabet, and the letters u, v, w, x, y, z for words over an alphabet. The length of a word w is denoted by $|w|$. Sets of words are denoted by $X, Y, \ldots$.

If X, Y are sets of words then $X \cdot Y = \{xy : x \in X \text{ and } y \in Y\}$. For a finite alphabet A, $A^0 = \{\epsilon\}$ and for each finite n, $A^{n+1} = A^n \cdot A$; the union $\bigcup_{n<\omega} A^n$ is denoted by A^*. In particular, if A is an alphabet then A^* is the set of all finite words over A.

In a similar way, let A^ω denote the set of ω-sequences (or: ω-words) over A. An ω-word over A is written as $\alpha = \alpha(0)\alpha(1)\ldots$ with $\alpha(i) \in A$. Let $A^\infty = A^* \cup A^\omega$. Letters $\alpha, \beta, \ldots$ are used for ω-words and $L, L', \ldots$ are used for sets of ω-words (*i.e.,* ω-languages).

A set of words X is *regular* if for some alphabet Σ, X is a member of the least class C such that:

1. every finite subset of Σ^* belongs to C;
2. C is a Boolean algebra of sets (*i.e.,* if $Y, Z \in C$ then $Y \cap Z, Y \cup Z, Y - Z$ are also members of C);
3. if $Y, Z \in C$ then $Y \cdot Z \in C$; and
4. if $Y \in C$, then $Y^* \in C$.

Many characterizations of the regular sets are known in the literature. The earliest, due to Kleene [10], states that a set of words is regular iff it is the set of words accepted by some finite automaton. We discuss a related concept in Section 2. Among other characterizations, we have for example, that the regular sets coincide with the sets generated by a right linear grammar (Chomsky and Miller [3]) with the sets definable by a formal language (as in Büchi [2] and Elgot [6]) which we shall see in more detail in Section 3. Also, regular sets have been characterized as unions of equivalence classes of a congruence relation of finite index (the Myhill-Nerode Theorem, [12]).

The logical connectives are written $\neg, \vee, \wedge, \rightarrow, \exists, \forall$. As a shorthand for the quantifiers "there exist infinitely many n" and "there exist finitely many n" we use $\exists^{\omega} n$ and $\exists^{<\omega} n$.

For $W \subseteq A^*$ for some finite alphabet A, let

$$\begin{aligned} \text{prefix}(W) &\overset{\text{def}}{=} \{u \in A^* \mid \exists v(uv \in W)\} \\ W^{\omega} &\overset{\text{def}}{=} \{\alpha \in A^{\omega} \mid \alpha = w_0 w_1 \ldots \text{ with } w_i \in W, i \geq 0\} \\ \vec{W} &\overset{\text{def}}{=} \{\alpha \in A^{\omega} \mid \exists^{\omega} n \alpha(0) \ldots \alpha(n-1) \in W\} \end{aligned}$$

2 Recognizable Sets and Automata over Trees

In this section we explore the extension of conventional finite automata theory as defined in [8] for instance, to the domain of trees. This extension was explored by Thatcher [18] and also by Doner [5]. As we shall see, the set of trees from this tree domain that are recognizable by the automata over trees (that are the object of attention in this section) is exactly the projection of the set of derivation trees of a context-free grammar. This is an important result towards the set of results that are the focus of this paper and this is seen in Section 4.

Let these automata over trees be denoted by *tree-acceptors*. First, we describe the universe of discourse for these tree-acceptors which are finite trees of symbols rather than usual sequences of symbols, *i.e.,* the analog of A^* over a finite alphabet A for conventional finite automata theory which accept a sequence of words from A^*.

2.1 Trees

In graph theory, an *undirected tree* is a undirected graph which is connected and has no cycles (any of its nodes can be thought of as the root), while a *directed tree* is a directed graph with a special node called the root such that there is a unique path from the root to any other node (and thus resulting in no cycles). However, there is no special order among each of the nodes in each of these definitions. Thus we need a notion of *ordered tree.*

Definition 2.1 (Gorn [7]) *A* Σ-tree *is a function from a non-empty finite closed subset of* $\mathbb{N}*$ *(a set of strings of natural numbers) into a finite alphabet* Σ *where* $U \subseteq \mathbb{N}*$ *is closed if:*

1. *if* $uv \in U$ (i.e., *u is a prefix of v) then $u \in U$, and*
2. *if* $ui \in U$, *where* $i \in \mathbb{N}*$, *and* $1 \leq j \leq i$ *then* $uj \in U$.

The elements of U are called *tree addresses.* When $u \in U$ and $ui \in U$ for some $i \in \mathbb{N}*$ we say that ui is a child of the parent u. For the purposes of this section we are only interested in *finitely* branching tree domains, *i.e.,* those in which a parent can have only finitely many children and the depth of a tree is also finite. We shall consider more general tree domains in Section 3.

Example 2.2 *In Figure 2 we see an example of a Σ-tree T where $\Sigma = \{a, b, c, d\}$ and $dom(T) \overset{\text{def}}{=} \{\epsilon, 0, 1, 00, 01\}$ and if by convention we assume $T(\epsilon) \overset{\text{def}}{=} T_\epsilon$, then we have $T_\epsilon = a, T_0 = b, T_1 = e, T_{00} = c, T_{01} = d$.*

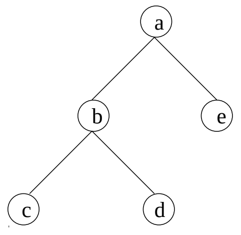

Figure 2: Graph representaton of a Σ-tree

The concept Σ-tree can be seen as generalizing the concept of Σ-word (*i.e.,* the usual notion of word in conventional finite automata theory), where if the usual meaning of Σ-word can be replaced with a function with range Σ and with domain a finite set consisting of all initial segments of some word in $\{a\}^*$, the generalization to Σ-trees is straightforward.

2.2 Recursion over Trees

This section develops some basic notions about trees which are useful in later sections. Also, in this section we define the notion of tree recursion to enable us to use induction over trees while constructing proofs. The definitions in this section closely follow the concepts in Doner [5].

In Definition 2.1 we define a Σ-tree τ as a function $\tau : U \rightarrow \Sigma$ where U was a finite non-empty closed subset of $\mathbb{N}*$. For simplicity, we redefine this as follows: U is a finite non-empty closed subset of $\{0, \ldots, p-1\}^*$ where the Σ-tree is now said to be of order p. For example, the Σ-tree in Figure 2 is a Σ-tree of order 2, and the tree is a function with domain which is a finite subset of $\{0,1\}^*$. This redefinition is done without loss of generality to simplify the exposition of the proofs. The class of all Σ-trees of a given order p is denoted by $\Sigma^{\#}$.

The empty tree, *i.e.,* the function with domain $\emptyset$ is denoted by ϵ. A *terminal* of a tree τ is a word $w \in dom(\tau)$ such that no extension of w is also in $dom(\tau)$. The set of all terminals of τ is $\mathrm{fr}(\tau)$ (frontier of τ). Assume that the trees have domains which are finite subsets of $\{0,1\}^*$. Then, the subtree of τ beginning at w is $\tau \setminus w$[1]. If τ, τ' are Σ-trees, then $\tau[w/\tau']$ is the result of replacing the subtree of τ beginning at w with the tree τ'[2]. Notice that $\tau[w/\tau']$ is a Σ-tree only in case $w \in \{u0, u1 : u \in dom(\tau)\} \cup \{\epsilon\}$. For $\sigma \in \Sigma$ and $\tau, \tau' \in \Sigma^{\#}$, we put $\sigma[\tau, \tau'] = (\sigma[0/\tau])[1/\tau']$. Thus $\sigma[\tau, \tau']$ is the unique tree π such that $\pi_\epsilon = \sigma, \pi \setminus 0 = \tau$, and $\pi \setminus 1 = \tau'$. Every tree except ϵ can be expressed in the form $\sigma[\tau, \tau']$ for some σ, τ, τ'.

The notation $\sigma[\tau, \tau']$ facilitates a form of induction over trees; namely, if for a given proposition $P(\tau)$, where τ ranges over Σ-trees, we can prove

[1]Formally, if τ is a Σ-tree and $w \in \{0,1\}^*$ then $\tau \setminus w$ is the Σ-tree π such that $\pi_u = \pi_{wu}$ for each $u \in \{0,1\}^*$.

[2]That is, $\tau[w/\tau']$ is the function π such that

$$\pi_{wv} = \tau'_v \text{ for all } v \in \{0,1\}^*$$

and

$$\pi_u = \tau_u \text{ for all } u \in \{0,1\}^* - (\{w\} \cdot \{0,1\}^*).$$

1. $P(\epsilon)$

2. if $P(\tau)$ and $P(\tau')$, then $P(\sigma[\tau, \tau'])$ for every $\sigma \in \Sigma$,

then we infer $P(\tau)$ for every $\tau \in \Sigma^{\#}$.

Example 2.3 *We can define* depth *of a tree τ by tree recursion:*

$$\begin{aligned} \text{depth}(\epsilon) &= 0, \\ \text{depth}(\sigma[\tau, \tau']) &= 1 + \max(\text{depth}(\tau), \text{depth}(\tau')). \end{aligned}$$

Another approach, equivalent to the one taken here, and which is taken by Thatcher [18], is to represent the trees as terms in a formal language: The elements of Σ are construed as two place function symbols (or p-place function symbols for trees of order p). The empty tree ϵ is represented by a new constant symbol λ and for any $\sigma \in \Sigma$ and if η, η' are the terms representing $\tau, \tau' \in \Sigma^{\#}$ respectively, then $\sigma(\eta, \eta')$ is the term representing $\sigma[\tau, \tau'])$. Thus, the tree in Figure 2 is represented by the term

$$a(b(c(\lambda, \lambda), d(\lambda, \lambda)), e(\lambda, \lambda)) \tag{1}$$

Notice that in (1) by replacing the $(,)$ by $[,]$ and the λ by ϵ, or vice versa, enables us to view the tree in Figure 2 in either representation.

2.3 Tree Acceptors

This section is devoted to the development of a generalized notion of finite automata, which admits trees rather than words as their inputs. It turns out, as shown in [18, 5] that a large part of conventional finite automata theory continues to hold in the generalized context. Further details can be found in [15, 6].

The definitions for the deterministic tree acceptors (see Definition 2.4) as well as the non-deterministic tree acceptors (see Definition 2.5) are closely related to their finite automata counterparts (see [8]).

Definition 2.4 *A* Σ-tree acceptor *is a 4-tuple $\mathcal{U} = \langle Q, t, q_0, F \rangle$, where*

1. Q is a non-empty finite set of states;

2. *t is a mapping of $Q \times Q \times \Sigma$ into Q, the* transition function*;*

3. *$q_0 \in Q$ is the* initial state*;*

4. *$F \subseteq Q$ (the set of* final states*).*

Associated with $\mathcal{U}$ is the function $\bar{t} : \Sigma^{\#} \rightarrow Q$ defined by

$$\begin{aligned} \bar{t}(\epsilon) &= q_0 \\ \bar{t}(\sigma[\tau, \tau']) &= t(\bar{t}(\tau), \bar{t}(\tau'), \sigma), \end{aligned}$$

for all $\sigma \in \Sigma$ and $\tau, \tau' \in \Sigma^{\#}$. $\mathcal{U}$ *accepts* a tree $\tau \in \Sigma^{\#}$ if $\bar{t}(\tau) \in F$.

$$T(\mathcal{U}) = \{\tau \mid \bar{t}(\tau) \in F\}$$

$T(\mathcal{U})$ is the set of Σ-trees accepted by $\mathcal{U}$. $T(\mathcal{U})$ is also known as the behaviour of $\mathcal{U}$ relative to F or $\mathrm{bh}_{\mathcal{U}}(F)$ [18].

Definition 2.5 *A* nondeterministic Σ-tree acceptor *is a 4-tuple $\mathcal{U} = \langle Q, t, I, F \rangle$, where*

1. *Q is a non-empty finite set of states;*

2. *t is a mapping of $Q \times Q \times \Sigma$ into the nonempty subsets of Q, the* transition function*;*

3. *$I \subseteq Q$ is the nonempty set of* initial states*;*

4. *$F \subseteq Q$ (the set of* final states*).*

For the nondeterministic tree acceptor

$$T(\mathcal{U}) = \{\tau \mid \exists q(q \in \bar{t}(\tau) \wedge q \in F)\}.$$

When necessary, the tree acceptors of Definition 2.4 will be referred to as *deterministic* tree acceptors.

Just as with finite automata, it turns out that the class of sets accepted by nondeterministic tree acceptors is the same as the class of sets accepted by the deterministic tree acceptors. This result is obtained in a way that is exactly parallel to the "subset" construction used with finite automata (see pp. 22–23 [8]).

To get an intuitive picture of a tree acceptor "recognizing" an input Σ-tree, one can describe a tree acceptor $\mathcal{U}$ "acting" on the corresponding Σ-tree by producing a *state tree* of the same shape (*i.e.,* with almost the same domain, depth greater by one) by assigning initial state to each terminal node ($\overline{t}(\epsilon) = q_0$) and at any time when states have been assigned to subtrees τ and τ' of node σ then that node is assigned a state by the transition function t (by applying it to the subtrees, $t(\overline{t}(\tau), \overline{t}(\tau'), \sigma)$). The output state or final state is the state assigned to the root of the tree and the tree is accepted if that state is in F.

We now formalize this intuitive notion of a *state tree* and acceptance via such a construction.

Definition 2.6 *Let* $\mathcal{U} = \langle Q, t, q_0, F\rangle$ *be a* Σ*-tree acceptor and let* $\tau \in \Sigma^{\#}$*. The* state tree *(or S-tree)* π compatible *with* τ *(or* $\mathcal{U}$*-compatible where* $\mathcal{U}$ *is not clear from the context) is defined by*

1. $dom(\pi) = \{\epsilon\} \cup (dom(\tau) \cdot \{0, 1\})$,
2. $\pi_w = \overline{t}(\tau \setminus w)$ *for each* $w \in dom(\pi)$.

In the state tree π for τ, $depth(\pi) = 1 + depth(\tau)$. This is analogous to the situation in finite automata, where a sequence of states compatible with an input word is always greater than the word by one. Now we can state (without proof) the intuitive notion of acceptance presented earlier.

Lemma 2.7 *If* $\mathcal{U} = \langle Q, t, q_0, F\rangle$ *is a* Σ*-tree acceptor,* $\tau \in \Sigma^{\#}$*, and* π *is compatible with* τ*, then* $\tau \in T(\mathcal{U})$ *iff* $\pi_\epsilon \in F$.

Now we are in a position to define the notion of a *recognizable set.* This term was first introduced in this context by Mezei and Wright [11].

Definition 2.8 *A set* $A \subseteq \Sigma^{\#}$ *is* recognizable *(over* Σ*) if* $A = T(\mathcal{U})$ *for some* Σ*-tree acceptor* $\mathcal{U}$.

To conclude we present some relevant results without proof. More detail on these and related areas are to be found in Thatcher [18], Thatcher and Wright [19] and Doner [5].

Theorem 2.9 *The class of recognizable sets is a Boolean algebra,* i.e., *it is closed under finite unions, finite intersections, and differences.*

Theorem 2.10 *If $\mathcal{U}$ is any tree acceptor, then it is effectively decidable whether*

1. $T(\mathcal{U}) = \emptyset$.

2. $T(\mathcal{U})$ *is finite.*

2.4 Context-Free Derivation Trees and Recognizability

In this section we make the first connection between recognizable sets (recognizability as applied to Σ-trees, see Definition 2.8) and context-free grammars. To obtain this connection, all we have to notice is that the set of derivation trees of any context-free grammar can be viewed as a Σ-tree as defined in Sections 2.1 and 2.2.

We define context-free grammars in the usual way (following the notational devices of [8]).

Definition 2.11 *A* context-free grammar *is denoted by $G = \langle N, T, P, S_0 \rangle$, where*

1. *N and T are finite sets of* non-terminal *and* terminal *symbols respectively;*

2. *P is a finite set of* productions *of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$;*

3. *S_0 is a special non-terminal called the* start symbol.

Given this definition we can easily define the set of derivation trees of a context-free grammar in the following way: Denote D_G^σ as the set of derivation trees of G from symbol $\sigma \in N \cup T$. The following recursive definition then defines D_G^σ as a set of Σ-trees:

$$s \in D_G^s \quad \text{for} \quad s \in T, \tag{2}$$

$$\text{if } \tau \in D_G^{s_1} \quad \text{and} \quad \tau' \in D_G^{s_2} \text{ and } \sigma \rightarrow s_1 s_2 \text{ then } \sigma[\tau, \tau'] \in D_G^\sigma. \tag{3}$$

$D_G^{S_0}$ is the set of derivation trees of G. Now, the relation of the recognizable sets to the set of derivation trees can be succintly presented in the following two propositions.

Proposition 2.12 *For any context-free grammar $G = \langle N, T, P, S_0 \rangle$, the set $D_G^{S_0}$ of derivation trees from initial state S_0 to terminal strings is a recognizable subset of the tree domain of $N \cup T$ (*i.e.*, Σ-trees where $\Sigma = N \cup T$).*

Proof. Given the grammar G, construct the nondeterministic tree acceptor $\mathcal{U} = \langle N \cup T, t, I, F \rangle$ on the alphabet $N \cup T$, where t is defined as:

1. $s' \in t(\bar{t}(\epsilon), \bar{t}(\epsilon), s)$ iff $s' \in (N \cup T) \setminus T$ and $s \in T$ and $s = s'$. Let $I = \{s' \mid s' \in \bar{t}(s)\}$.

2. If $\sigma[\tau, \tau'] \in D_G^{\sigma}$ corresponds to $\sigma \to s_1 s_2 \in P$ and $s' \in N \cup T$ then $s' \in t(\bar{t}(\tau), \bar{t}(\tau'), \sigma)$ iff $s = s'$.

And finally we define the set of final states as $F = \{q \mid q \in \bar{t}(\tau), \tau \in D_G^{S_0}\}$.

To show the result it suffices to show that (a) $\mathcal{U}$ is a tree acceptor, and (b) every tree σ, $\sigma \in D_G^{S_0}$, is accepted by $\mathcal{U}$. (a) is true by construction of the proof, and (b) is proved by applying tree induction on the definition of derivation trees (base case is (2) and induction is done in step (3)). From this it follows that $D_G^{S_0} = T(\mathcal{U})$. □

We can now state the converse to Proposition 2.12.

Proposition 2.13 *Every recognizable subset of $\Sigma^{\#}$ over a finite alphabet Σ is a projection of the set of derivations of a context-free grammar.*

Proof. Given a tree acceptor $\mathcal{U} = \langle Q, t, q_0, F \rangle$ that recognizes a subset $U \subseteq \Sigma^{\#}$, we define a context-free grammar $G = \langle N, T, P, S_0 \rangle$, where

$$\begin{aligned}
N &= \{x : x \in Q \times \Sigma\} \\
T &= \{\langle \bar{t}(f), f \rangle \mid f \in \sigma\}, \text{by definition } f \in \Sigma^{\#} \text{ as well.} \\
S_0 &= \{\langle q, f \rangle \mid q \in F \text{ and } f \in \Sigma\}, \textit{and} \\
P &= \{\langle q, f \rangle \to \langle q_1, f_1 \rangle \langle q_2, f_2 \rangle \mid f[q_1, q_2] = q, \text{ where } f, f_1, f_2 \in \Sigma\}
\end{aligned}$$

Tree induction easily shows that if σ is a derivation tree in this constructed context-free grammar then by constructing a tree with values as the second projection of each node (each node is a tuple that is in $Q \times \Sigma$ and hence the second projection will adorn the resulting tree with values from Σ). This tree can now be seen as a Σ-tree, which is input to $\mathcal{U}$, the tree acceptor, and the first component is effectively the *state tree* or S-tree (see Definition 2.6). Thus, the second projection over the set of derivation trees constructed is our original recognizable set U. $\Box$

In the above proof to Proposition 2.13, we have to keep in mind our earlier assumptions concerning our notation for trees as $\sigma[\tau, \tau']$ where τ and τ' are subtrees of σ. Due to this assumption the context free grammar constructed is always in Chomsky Normal Form (see [8]). To make the proof more general we simply consider n-tuples in the construction of P in the above proof. Also, we need to consider a variant of context-free grammar where in production rules $A \to \alpha$, α are allowed to be regular expressions, *i.e.,* for each $s \in N, P_s = \{w \mid s \to w\}$ is regular. This is not a significant extension and is used to make the proof simpler to construct. The languages obtained with such a modified context-free grammar are still the context-free languages. In fact, exactly such a construction is used by Thatcher [18].

Putting Propositions 2.12 and Proposition 2.13 together (and accounting for the fact that tree acceptors like conventional finite automata are closed under projections) we get the following characterization of the set of Σ-trees.

Corollary 2.14 *A set of Σ-trees is recognizable iff it is a projection of the set of derivations of some context-free grammar.*

3 Automata over Infinite Sequences

In this section, we consider some extensions of automata to the domain of infinite objects. First we consider Büchi automata (ref. Büchi [2, 1]) which accept ω-words (an infinite sequence over a finite alphabet) and then extend this notion in Section 4 look at Rabin tree automata which accept elements from an infinite tree domain. We follow the notational conventions of Thomas [20] through most of the results in this section.

Recall that A^ω denotes the set of ω-sequences or ω-words over a finite alphabet A. An ω-word is written as $\alpha = \alpha(0)\alpha(1)\ldots$ with $\alpha(i) \in A$. Let $A^\infty = A^* \cup A^\omega$. Notations for segments of ω-words are

$$\alpha(m, n) \quad \overset{\text{def}}{=} \quad \alpha(m)\ldots\alpha(n-1) \text{ for } m \leq n \text{ and}$$

$$\alpha(m,\omega) \quad \overset{\text{def}}{=} \quad \alpha(m)\alpha(m+1)\dots$$

For an ω-word $\sigma = \sigma(0)\sigma(1)\dots$ from A^ω, the "infinity set" of σ is

$$\text{In}(\sigma) \overset{\text{def}}{=} \{a \in A \mid \exists^\omega n(\sigma(n) = a)\}. \tag{4}$$

3.1 Büchi Automata

Büchi automata are nondeterministic finite automata equipped with an acceptance condition that is appropriate for ω-words: and ω-word is accepted if the automaton can read it from left to right while assuming a sequence of states in which some final state occurs infinitely often. This condition is termed as *Büchi acceptance.*

Definition 3.1 *A Büchi automaton over the alphabet A is of the form $\mathcal{A} = \langle Q, q_0, \delta, F\rangle$ with finite* state set *Q and* initial state *$q_0 \in Q$,* transition relation *$\delta \subseteq Q \times A \times Q$, and a set $F \subseteq Q$ of final states.*

A *run* of a Büchi automaton $\mathcal{A}$ on an ω-word $\alpha = \alpha(0)\alpha(1)\dots$ from A^ω is a sequence $\sigma = \sigma(0)\sigma(1)\dots$ such that $\sigma(0) = q_0$ and $(\sigma(i), \alpha(i), \sigma(i+1)) \in \delta$ for $i \geq 0$. The run is called *successful* if $\text{In}(\sigma) \cap F \neq \emptyset$, *i.e.,* some state of F occurs infinitely often in it. $\mathcal{A}$ *accepts* α if there is a successful run of $\mathcal{A}$ on α. Let

$$L(\mathcal{A}) = \{\alpha \in A^\omega \mid \mathcal{A} \text{ accepts } \alpha\}$$

be the ω-language *recognized* by $\mathcal{A}$. The language L accepted by some Büchi automaton $\mathcal{A}$ is called *Büchi recognizable.*

Consider some examples of Büchi recognizable languages.

Example 3.2 *Let $\Sigma = \{a,b,c\}$ be a finite alphabet. Define $L_1 \subseteq \Sigma^\omega$ such that $\alpha \in L_1$ iff b occurs after any occurrence of a in α.*

In Figure 3, A is a Büchi automaton (in the usual state-transition graph notation) recognizing L_1 and $\overline{A}$ recognizes the complement of L_1 i.e., *$A^\omega - L_1$.*

Example 3.3 *B in Figure 4 is the Büchi automaton for the ω-language $L_2 \subseteq \Sigma^\omega$ defined as follows: $\alpha \in L_2$ iff between any two occurrences of letter a in α there is an even number of letters b, c.*

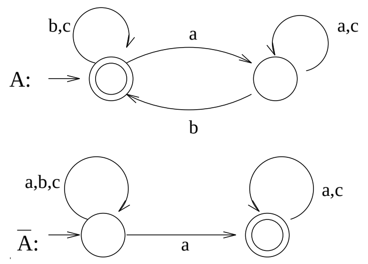

Figure 3: Büchi automata for L_1 and its complement.

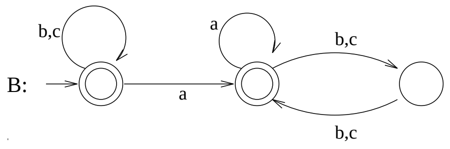

Figure 4: Büchi automata for L_2.

From examining the above examples, it becomes clear that the nature of Büchi recognizable ω-languages has to be done by looking at the sequence of states of the automaton visited in the recognition of a ω-word. Let $w = a_0 \ldots a_{n-1}$ be a finite word over alphabet A and let $\mathcal{A} = \langle Q, q_0, \delta, F \rangle$ be a Büchi automaton, then define $s \xrightarrow{w} s'$ if there is a state sequence $s_0, \ldots, s_n$ such that $s_0 = s, (s_i, a_i, s_{i+1}) \in \delta$ for $i < n$, and $s_n = s'$. Let

$$W_{ss'} = \{w \in A^* \mid s \xrightarrow{w} s'\} \tag{5}$$

3.1.1 Closure Properties

The following lemma verifies the closure properties of sets recognizable by Büchi automata.

Lemma 3.4 *If $V \subseteq A^*$ is regular, then*

1. *V^ω is Büchi recognizable.*
2. *If $L_1, L_2 \subseteq A^\omega$ are Büchi recognizable, then $V \cdot L_1$, $L_1 \cup L_2$ and $L_1 \cap L_2$ are Büchi recognizable.*

Proof.

1. Since $V^\omega = (V - \{\epsilon\})^\omega$, assume that V does not contain the empty word; and assume that no transition goes into the finite automata recognizing V. A Büchi automaton $\mathcal{A}'$ recognizing V^ω is obtained by adding transition (s, a, q_0) for any transition (s, a, s') with $s' \in F$, to the original Büchi automaton $\mathcal{A}$ with q_0 as the single final state of $\mathcal{A}'$.

2. The claims of concatenation and union can be proved in a way exactly parallel to the case for regular sets of finite words, by simple manipulation of the original automata. For the case of intersection: consider $\mathcal{A}_1 = \langle Q_1, q_1, \delta_1, F_1\rangle$ recognizing L_2 and $\mathcal{A}_2 = \langle Q_2, q_2, \delta_2, F_2\rangle$ recognizing L_2. Then $\mathcal{A} = \langle Q_1 \times Q_2 \times \{0, 1, 2\}, (q_1, q_2, 0), \delta, F\rangle$ recognizes $L_1 \cap L_2$, where δ follows δ_1 and δ_2 for the first two components of the states, and changes the third from 0 to 1 when an F_1 state occurs in the first component, and from 1 to 2 when subsequently a F_2 state occurs in the second component and back to 0 immediately afterwards. Then 2 occurs infinitely often as the third component in the run iff some F_1 state and some F_2 state occur infinitely often in the first two components. Hence with the set of final states $F \overset{\text{def}}{=} Q_1 \times Q_2 \times \{2\}$ we obtain the desired Büchi automaton.

$\square$

3.1.2 Büchi Acceptance

We are now in a position to better analyze the workings of Büchi automata. The following theorem informs us of the way a Büchi automaton accepts any ω-sequence.

Theorem 3.5 (Büchi [2]) *An ω-language $L \subseteq A^\omega$ is Büchi recognizable iff L is a finite union of the sets U and V^ω where $U, V \subseteq A^*$ are regular sets of finite words (assuming $V \cdot V \subseteq V$).*

Proof. Each of the finitely many $W_{ss'}$ is regular. By the definition of Büchi acceptance the ω-language recognized by $\mathcal{A}$ is

$$L(\mathcal{A}) = \bigcup_{s\in F} W_{q_0 S} \cdot (W_{ss})^\omega. \tag{6}$$

The direction from left to right is clear from equation (6). Notice that $W_{ss} \cdot W_{ss} \subseteq W_{ss}$. The converse result follows from Lemma 3.4. $\square$

A representation of an ω-language in the form $L = \bigcup_{i=1}^{n} U_i \cdot V_i$, where U_i, V_i are given by regular expressions, is called a *ω-regular expression.* Since the construction in Lemma 3.4 is defined as an algorithm, *i.e.,* there is an effective procedure for each of the constructions, the conversion of ω-regular expressions into Büchi automata and vice versa can be carried out effectively.

It is clear from equation (6) that a Büchi automaton $\mathcal{A}$ accepts some ω-word iff $\mathcal{A}$ reaches some final state (via some arbitrary word u) which can then be revisited by a loop of state sequences (via some other arbitrary word v). Hence, the existence of a reachable final state which is located in a loop of $\mathcal{A}$ can be checked by an effective procedure. So we obtain the following theorem:

Theorem 3.6

1. *Any nonempty regular ω-language contains an ultimately periodic ω-word,* i.e., *a ω-word of the form $uvvvv\ldots$*

2. *The emptiness problem for Büchi automata is decidable.*

3.2 Decidability of S1S

The original motivation of Büchi in his work on ω-languages was the analysis of a system of monadic second order logic for the formalization of the properties of sequences. To this end, Büchi introduced the "sequential calculus" which we shall go into in this section. Büchi [2] showed the remarkable fact that any condition on sequences that is written in this calculus can be reformulated as a statement about acceptance of sequences by an automaton.

If we consider Kleene's [10] original notion of *synthesis* (effectively obtaining a finite automata from a regular expression) and *analysis* (the reverse result), then the result by Büchi simply replaces the regular expression in the above definition by his system of monadic second order logic with 1 successor function (or $S1S$). And thus, we can view these decidability result as well known analysis and synthesis theorems.

3.2.1 The Sequential Calculus

The "sequential calculus" or by its more succint name $S1S$ is a system of monadic second-order logic, due to quantification over sets, which are unary relations and hence monadic second-order objects.

An ω-word $v \in A^\omega$ is represented by a model-theoretic structure of the form

$$\Upsilon_v = (\mathbb{N}, 0, +1, <, (Q_a)_{a \in A}), \tag{7}$$

where $(\mathbb{N}, 0, +1, <)$ is the structure of the natural numbers with zero, the successor function, and the usual ordering relation, and furthermore where

$$Q_a \overset{\text{def}}{=} \{i \in \mathbb{N} \mid \alpha(i) = a, a \in A\} \tag{8}$$

In Theorem 3.10 we shall see that $<$ is second-order definable in terms of successor. Hence its use does not add any power to the formalism.

The corresponding first order language for this structure contains variables $x, y, \ldots$ over the natural numbers which are used to denote positions in ω-words. For example, atomic formulae are of the form "$x + 1 < y$" which says that the position following x comes before y or "$x \in Q_a$" indicates that position x carries the letter a.

Example 3.7 *The set $L_1 \subseteq \{a, b, c\}^\omega$ in Example 3.1 can be defined by the sentence:*

$$\forall x(x \in Q_a \rightarrow \exists z(z \in Q_b \wedge x < z))$$

Recall that acceptance in L_1 was defined to be an occurrence of b after any occurrence of a.

We also allow variables $X, Y, \ldots$ for *sets* of natural numbers and quantifiers ranging over them.

Example 3.8 *The use of quantification over sets of natural numbers occurs in the definition of the ω-language L_2 in Example 3.1 which can be represented as the sentence φ_2:*

$$\begin{aligned}\varphi_2 : \forall x \forall y (x \in Q_a \wedge y \in Q_a \wedge x < y \wedge \neg \exists z (x < z \wedge z < y \wedge z \in Q_a) \rightarrow \\ \exists X (x \in X \wedge \forall z (z \in X \leftrightarrow \neg (z + 1 \in X)) \wedge \neg (y \in X))).\end{aligned}$$

Note that in Example 3.8 the set quantifier postulates a set containing every second position starting with position x. This ensures that the number of letters between positions x and y is even. Recall that we want acceptance in L_2 iff there are even numbers of letters b, c between any two occurrences of a.

The interpreted system $S1S_A$ (the *second order theory of one successor over* A) is defined as follows: *Terms* are constructed from the constant 0 and the variables $x, y, \dots$ by applications of the $+1$ successor function. *Atomic formulas* are of the form $t = t'$, $t < t'$, $t \in Q_a$ (for $a \in A$), $t \in X$, where t, t' are terms and X is a set variable.

$S1S_A$ *formulas* are constructed from the atomic formulas using the connectives $\neg, \vee, \rightarrow, \leftrightarrow$ and the quantifiers $\exists, \forall$ which range over either kind of variable. If $X_1, \dots, X_n$ occur free (not scoped by a quantifier) in a formula φ then we write $\varphi(X_1, \dots, X_n)$. Formulas with no free variables are called *sentences*.

Given $v \in A^\omega$ and a $S1S_A$ sentence φ, we write $\Upsilon_v \models \varphi$ (with Υ_v defined as in (7)) if φ is satisfied in Υ_v under the usual interpretation. Then the ω-language defined by a $S1S_A$ sentence φ is $L(\varphi) = \{v \in A^\omega \mid \Upsilon_v \models \varphi\}$.

For instance, if $v = abcaabcaaabc\dots$ and φ_1 is defined as in Example 3.7, then we have $\Upsilon_v \models \varphi_1$ and $L(\varphi_1)$ is the ω-language L_1 of Example 3.1.

In general, we can replace Q_a with the set variables X_k and replace the alphabet A in $S1S_A$ with an implicit alphabet $\{0,1\}^n$. This formalism is called $S1S$.

Under such an implicit alphabet an ω-word $v \in (\{0,1\}^n)^\omega$, the formula "$x \in X_k$" says that the x^{th} letter of v has 1 in its k^{th} component. Formally, v is represented by the structure $\Upsilon_v = (\mathbb{N}, 0, +1, <, P_1, \dots, P_n)$ where $P_k = \{i \mid k^{th} \text{ component of } v(i) = 1\}$ and $\Upsilon_v \models \varphi(X_1, \dots, X_n)$ iff X_k has an interpretation in P_k.

Embedding of $S1S_A$ formulas in a given alphabet A into the alphabet $\{0,1\}^n$, for suitable n is accomplished by simply replacing formulas "$x \in Q_a$" with the corresponding conjunction of formulas "$x \in X_k$" and "$\neg x \in X_k$".

Example 3.9 *Consider the ω-sequence $v \in (\{0,1\}^2)^\omega$, where if the letters from the two components of $\{0,1\}^2$ are written as columns:*

$$\begin{array}{ccl} v & : & 1\,0\,1\,0\,1\,0\,1\,0\,\dots \\ & & 1\,1\,1\,0\,0\,0\,0\,0\,\dots \end{array}$$

Since in v there are infintely many letters with first component 1 *and second component* 0*, we have*

$$\Upsilon_v \models \forall x \exists y (x < y \wedge y \in X_1 \wedge \neg(y \in X_2))$$

3.2.2 Büchi's Theorem

In this section we identify definability in $S1S$ with the notion of Büchi acceptance as defined in Section 3.1.2.

Theorem 3.10 (Büchi's Theorem) *An ω-language is definable in $S1S$ iff it is regular.*

Proof. If an ω-language is regular then let the Büchi automaton that recognized it be $\mathcal{A} = \langle Q, q_0, \delta, F \rangle$ with $Q = \{0, \ldots, m\}$ and $q_0 = 0$. For any ω-word $v \in A^\omega$ a successful run on v can be expressed by defining a set for positions for each state visited in the run. This is given by a $(m+1)$-tuple of sets $Y_0, \ldots, Y_m$. Then $L(\mathcal{A})$ is defined by the sentence:

$$\begin{aligned} \exists Y_0 \ldots \exists Y_m (\bigwedge_{i \neq j} \neg \exists y (y \in Y_i \wedge y \in Y_j) \wedge 0 \in Y_0 \\ \wedge \forall x \bigvee_{(i,a,j) \in \delta} (x \in Y_i \wedge x \in Q_a \wedge x + 1 \in Y_j) \\ \wedge \bigvee_{i \in F} \forall x \exists y (x < y \wedge y \in Y_i)) \end{aligned} \tag{9}$$

For the other direction we simplify the problem by showing the reduction of $S1S$ to a simpler formalism $S1S_0$ and then showing Büchi recognizability for $S1S_0$ definable sets.

For the first step each $S1S$-formula $\varphi(X_1, \ldots, X_n)$ interpreted in ω-words over $\{0,1\}^n$ (as explained previously) we reduce it to a $S1S_0$-formula where *only* second-order variables X_i occur and the atomic formulas are all of the form $X_i \subseteq X_j$, and $\mathrm{Succ}(X_i, X_j)$ which means that if X_i, X_j are singleton sets $\{x\}, \{y\}$ where $x + 1 = y$.

In the second step, induction over $S1S_0$ formulas shows that $L(\varphi)$ is Büchi recognizable.

1. *Reduction of $S1S$ to $S1S_0$.* Carry out the following syntactic rewritings:

   - Eliminate superpositions of "+1":

$$(x+1)+1 \in X \text{ becomes } \exists y \exists z (x + 1 = y \wedge y + 1 = z \wedge z \in X)$$

- Eliminate the symbols 0 and $<$:

$$\begin{array}{rll} 0 \in X & \text{becomes} & \exists x(x \in X \wedge \neg \exists y(y < x)), \\ x < y & \text{becomes} & \forall X(x+1 \in X \wedge \forall z(z \in X \rightarrow z+1 \in X) \rightarrow y \in X). \end{array}$$

  We have now a formula that has atomic formulas of type $x = y$, $x+1 = y$, and $x \in X$ only.

- Eliminate first order variables:

  $\forall x \exists y(x+1 = y \wedge y \in Z)$ becomes $\forall X(\mathrm{Sing}(X) \rightarrow \exists Y(\mathrm{Sing}(Y) \wedge \mathrm{Succ}(X,Y) \wedge Y \subseteq Z))$.

  where,

$$\begin{array}{rll} X = Y & \text{is} & X \subseteq Y \wedge Y \subseteq X, \\ X \neq Y & \text{is} & \neg X = Y, \\ \mathrm{Sing}(X) & \text{is} & \exists Y(Y \subseteq X \wedge Y \neq X \wedge \neg \exists Z(Z \subseteq X \wedge Z \neq X \wedge Z \neq Y)), \end{array}$$

  The formula for $\mathrm{Sing}(X)$, *i.e.,* X is a singleton set, spells out the fact that there is exactly one proper subset of X.

By this process we arrive at a $S1S_0$-formula equivalent to a given $S1S$-formula.

2. *Büchi recognizability of $S1S_0$-definable sets.* By induction over $S1S_0$ formulas we show there is a Büchi automaton $\mathcal{A}$ over $\{0,1\}^n$ with $L(\mathcal{A}) = L(\varphi)$ for any $S1S_0$ formula $\varphi(X_1, \ldots, X_n)$. The corresponding Büchi automaton over $\{0,1\}^2$ for the atomic formula $X_1 \subseteq X_2$ is given in Figure 5(a) and the one for $\mathrm{Succ}(X_1, X_2)$ is in Figure 5(b).

   For the induction step it suffices to consider $\neg, \vee$ and $\exists$. Cases $\neg$ and $\vee$ are clear by the closure of regular ω-languages under complement and union. For $\exists$, we have to show closure of the regular ω-languages under *projection.* Considering a simpler case, assume that given a $S1S_0$ formula $\varphi(X_1, X_2)$ we can write $\varphi'(X_1) = \exists X_2(\varphi(X_1, X_2))$. Then for Büchi automaton $\mathcal{A}$ with $L(\mathcal{A}) = L(\varphi)$ over $\{0,1\}^2$ we construct an automaton $\mathcal{A}'$ over $\{0,1\}$ by changing letters of the transitions of $\mathcal{A}$ from $(1,0), (1,1)$ to

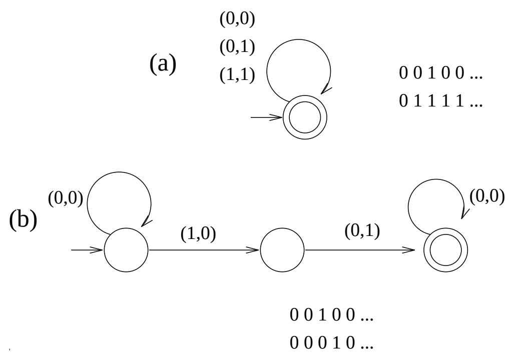

Figure 5: Büchi automata for $S1S_0$ atomic formulas

1 and $(0,0),(0,1)$ to 0, for example. A successful run of $\mathcal{A}'$ *guesses* a second component for the given input sequence in $\{0,1\}^\omega$ and the resulting sequence in $(\{0,1\}^2)^\omega$ it is a successful run of $\mathcal{A}$. Thus, $\mathcal{A}'$ recognizes $L(\varphi')$. □

In the above theorem, if we force interpretation over finite models then the results also hold for sets of *finite* words. This means that the notion *regular* is seen to be equivalent to *monadic second-order definable* for languages as well as for ω-languages. In a finite word w of length k, the variables $x, y, \dots$ refer to positions in w (from 1 to k) and the variables $X, Y, \dots$ to subsets of $\{1, \dots, k\}$. Moreover, the successor function is redefined to have a maximal element k, for instance, $k + 1 = k$.

Theorem 3.11 (Elgot [6], Büchi [2]) *A set $W \subseteq A^*$ is regular iff it is definable in $S1S$ (interpreted over finite word models).*

By Theorem 3.10 we can construct a Büchi automaton for *sentences* of $S1S$, *i.e.,* formulas without free variables. And by our earlier result in Theorem 3.5 which tells us that an existence of a successful run has an effective procedure, it can be decided whether an $S1S$-sentence holds in $(\mathbb{N}, 0, +1, <)$. This proves Büchi's primary motivation stated in Theorem 3.12.

Theorem 3.12 *Truth of sentences of S1S is decidable.*

Büchi's paper [2] ends with an open question as to whether the result can be extended to monadic second order theories with 2 successor functions or more than 2 successor functions. This is the question that is considered in the next section.

4 Recognizable Sets and Decidability of SnS

This section extends the results from the previous section to a decidability result for second order theories with more than 1 successor function.

4.1 Büchi and Rabin Recognizable Sets

In this section we make a return to tree acceptors. Here however, they are motivated as a generalization of the Büchi automata of the last section. To this end, we look at tree automata over infinite trees. The objective is to extend the results of the previous sections to monadic second order theories with more than 1 successor function.

4.1.1 Paths and Concatenation over Trees

A *path* through a tree t is a maximal subset of $dom(t)$ linearly ordered by $<$. If p is a path through t, then $t \mid \pi$ denotes the restriction of the function t to the set π. As before we restrict our attention to the binary branching trees. Let T_A be the set of finite binary trees over alphabet A and let T_A^ω be the set of infinite trees with domain $\{0,1\}^*$ over A. Let $T_A^\infty = T_A \cap T_A^\omega$. Subsets of T_A or T_A^ω are called *tree languages*.

Whereas, $\mathrm{fr}(\tau)$ is the set of terminals of the tree, the set $\mathrm{fr}^+(\tau)$ known as the *outer frontier* is the set of points just extending the domain of the tree and is defined as follows for a tree τ:

$$\mathrm{fr}^+(\tau) \overset{\mathrm{def}}{=} \{wi \notin dom(\tau) \mid w \in dom(\tau), i < k\}.$$

We define *tree concatenation* in terms of tree substitution where if $T \subseteq T_A, T' \subseteq T_A^\omega$ over alphabet A and if $c \in A$ then $T \cdot_c T'$ contains all trees which result from $\tau[c/\tau']$ for each $\tau \in T$ and for each occurrence of c on $\mathrm{fr}(\tau)$ by $\tau' \in T'$, and where different trees are admitted for different occurrences of c.

A Kleene closure for the operation $\cdot_c$ for some $c \in A$ for some $T \subseteq T_A$ denoted by T^{*c} and is defined as:

$$T^{0c} \quad = \quad \{c\}$$

$$\begin{aligned} T^{(n+1)c} &= T^{nc} \cup (T \cdot_c T^{nc}), \text{ and} \\ T^{*c} &= \bigcup_{n \geq 0} T^{nc}. \end{aligned}$$

Tree concatenation is also defined over tuples such as $\rho = \langle c_1, \ldots, c_m \rangle$ of concatenation symbols instead of single symbols c as before. This works as follows: for $T, T_1, \ldots, T_m \subseteq T_A$ and for trees $\tau \in T$ we obtain the concatenated set of trees over a tuple ρ by substituting each occurrence of c_i in $\text{fr}(\tau)$ with the tree T_i. This is denoted by $T \cdot_\rho (T_1, \ldots, T_m)$.

The set $(T_1, \ldots, T_m)^{\omega_\rho}$ is the ω-fold iteration of the $\cdot_\rho$ operation. This set contains all trees $\tau \in T_A^\omega$ for which there are trees $\tau_0, \tau_1, \ldots$ recursively built as follows, where $\rho = \text{fr}(\tau) = \{c_1, \ldots, c_m\}$:

$$\begin{aligned} \tau_0 &\in \{c_1, \ldots, c_m\}, \\ \tau_{m+1} &\in \{t_m\} \cdot_c (T_1, \ldots, T_m). \end{aligned}$$

τ is the common extension of the trees τ'_m which is derived from τ_m by deleting the symbols c_i from the frontier.

4.1.2 Automata over Infinite Trees

First we extend the Büchi automaton which works on ω-sequences to an infinite tree domain, *i.e.,* to define acceptance of infinite Σ-trees. We define these modified tree acceptors or tree automata in this section.

Definition 4.1 *A* Büchi tree automaton *over finite alphabet A is of the form $\mathcal{A} = \langle Q, q_0, \delta, F \rangle$ with finite* state set Q, initial state $q_0 \in Q$, transition relation $\delta \subseteq Q \times A \times Q \times Q$, *and a set $F \subseteq Q$ of final states.*

A *run* of Büchi tree automaton $\mathcal{A}$ on a tree $t \in T_A^\omega$ is a map $r : \{0,1\}^* \to Q$ (which corresponds to a state tree for $\mathcal{A}$) with $r(\epsilon) = q_0$ and $(r(w), t(w), r(t(w0), r(w1)) \in \delta$ for $w \in \{0,1\}^*$. The run r is *successful* if on each path some final state occurs infinitely often, *i.e.,* for all paths π,

$$\text{In}(r \mid \pi) \cap F \neq \emptyset.$$

A set $T \subseteq T_A^\omega$ is *Büchi recognizable* if it consists of trees accepted by some Büchi tree automaton.

Definition 4.2 *A* Rabin tree automaton *over finite alphabet A has the form $\mathcal{A} = \langle Q, q_0, \delta, \Omega\rangle$ with Q, q_0, δ defined as in Definition 4.1, and $\Omega = \{(L_1, U_1), \ldots, (L_n, U_n)\}$ is a collection of* accepting pairs *of state sets $L_i, U_i \subseteq Q$.*

A run r of the Rabin tree automaton $\mathcal{A}$ is *successful* if for all paths π there exists an $i \in \{1, \ldots, n\}$ with

$$\begin{aligned} \text{In}(r \mid \pi) \cap L_i &= \emptyset, \text{ and} \\ \text{In}(r \mid \pi) \cap U_i &\neq \emptyset. \end{aligned}$$

A set $T \subseteq T_A^\omega$ is *Rabin recognizable* if it consists of trees accepted by some Rabin tree automaton. Since any Büchi tree automaton can be regarded as a Rabin tree automaton (taking set $\Omega = \{(\emptyset, F)\}$), any Büchi recognizable set of infinite trees is Rabin recognizable.

To illustrate the function of Büchi and Rabin tree automata:

Example 4.3 *Consider a tree language T_0 over alphabet $A = \{a, b\}$:*

$$T_0 = \{t \in T_A^\omega \mid \textit{ some path through } t \textit{ carries infinitely many } a\}.$$

A Büchi tree automaton $\mathcal{A}$ which recognizes T_0 can be seen to work as follows: by non-deterministic choice, $\mathcal{A}$ guesses a path down the tree and on this path assumes a final state iff letter a is met, while on other parts of the tree only a fixed finite state is computed. Then the existence of a successful run amounts to the existence of a path in t with infinitely many values of a. Thus T_0 in Example 4.3 is Büchi recognizable (and hence Rabin recognizable.

Example 4.4 *But consider T_1, the complement language of T_0 defined as:*

$$T_1 = \{t \in T_A^\omega \mid \textit{ all paths through } t \textit{ carry only finitely many } a\}.$$

T_1 is recognized by a Rabin automaton with one accepting pair of states $(\{q_a\}, Q)$ where Q is the state set and q_a is computed iff letter a is encountered. Büchi tree automata, however, do not offer in their acceptance condition such a "finiteness test" along paths. The existence of T_1 which cannot be accepted by any Büchi automaton (proof omitted) gives us the following theorem.

Theorem 4.5 (Rabin [14]) *There is a tree language which is Rabin recognizable but not Büchi recognizable.*

We conclude discussion of these automata by listing some closure properties which are useful (without proof).

Theorem 4.6

1. *Languages that are recognizable by Büchi and Rabin tree automata are closed under union and projection.*
2. *Büchi tree automata are not closed under complementation.*
3. *Rabin tree automata are closed under complementation.*

Case 2 of the above theorem can easily be seen from Theorem 4.5. The proof of Case 3 however is a very involved construction in Rabin [13] and has been the topic of much subsequent attention (see Thomas [20] for details).

4.1.3 Relation to Recognizable Sets

In this section the structure of successful runs of Büchi and Rabin tree automata is analyzed. The notion of a Büchi recognizable set is explored and some results about Rabin recognizable sets of trees are given.

First consider Büchi tree automata and show a representation of Büchi recognizable sets in terms of recognizable sets of finite trees.

Let $\mathcal{A} = \langle Q, q_0, \delta, F\rangle$ be a Büchi tree automaton, where $F = \{q_1, \ldots, q_m\}$, and let $r : \{0,1\}^* \to Q$ be a successful run of $\mathcal{A}$ on the tree $t \in T^\omega_A$. The claim is that r can be built up from a finite run trees of the automaton $\mathcal{A}$ which are delimited by final states of $\mathcal{A}$. Starting from any node u of r the next occurences of final states enclose a finite tree:

$$D_u \overset{\text{def}}{=} \{w \in u \cdot \{0,1\}^* \mid r(v) \notin F \text{ for all } v \text{ with } u < v \leq w\}. \tag{10}$$

D_u defines a finitely branching tree which does not contain an infinite path (since r has a successful run). So by König's Lemma[3] D_u is finite, and the nodes of its outer frontier have r-values which are final states. This argument allows us to decompose r into layers of finite trees. The first layer $F_0 = D_\epsilon$ and then we define $F_{n+1} = \bigcup_{u \in \mathrm{fr}(F_n)} D_u$.

As a consequence we can represent $T(\mathcal{A})$ using recognizable sets of finite trees. Consider trees $\tau \in T_{A \cup F}$ which has values from F only in $\mathrm{fr}(\tau)$ and otherwise valued in A. Let $\overline{\tau}$ denote the tree obtained by deleting the F-valued frontier or $\mathrm{fr}^{-1}(F)$.

Now for $q \in Q$, let T_q consist of all such trees τ where there is a run of $\mathcal{A}$ on $\overline{\tau}$ which starts in state q and reaches on inputs $\mathrm{fr}^{+}(\overline{\tau})$ exactly the states of $\mathrm{fr}(\tau)$. Each set T_q is recognizable.

This means that an infinite tree is accepted by a Büchi tree automaton $\mathcal{A}$ iff it belongs to the set:

$$T_{q_0} \cdot_\theta (T_{q_1}, \ldots, T_{q_m})^{\omega_\theta}, \tag{11}$$

where $\theta = \langle q_1, \ldots, q_m \rangle$ which is the sequence of final states in F. Conversely, if we are given a $(m+1)$-tuple $\langle T_0, T_1, \ldots, T_m \rangle$ of recognizable sets of finite trees, the expression corresponding to equation (11) defines a Büchi recognizable set of infinite trees. This conclusion is stated in the following theorem:

Theorem 4.7 *A set $T \subseteq T_A^\omega$ is Büchi recognizable iff there are recognizable sets T_0, $T_1, \ldots, T_m \subseteq T_{A \cup \Phi}$, where $\Phi = \{c_1, \ldots, c_m\}$ such that $T = T_0 \cdot_\Phi (T_1, \ldots, T_m)^{\omega_\Phi}$.*

The above construction is used to show that the emptiness problem for Büchi tree automata is decidable (see Rabin [14]). For this, we set up an algorithm which eliminates step by step those states of a given Büchi tree automaton $\mathcal{A} = \langle Q, q_0, \delta, F \rangle$ that are useless for successful runs. Certainly, a state q cannot appear in a successful run if the set T_q is empty. So eliminate successively those states q from $\mathcal{A}$ where T_q is empty and update the transition relation of $\mathcal{A}$ accordingly. Each of the T_q sets is recognizable and so its emptiness can be checked effectively by using Theorem 2.10. The elimination procedure stops after at most $|Q|$ steps, delivering a state set Q_0. Then all that remains is to show

[3] A finitely branching tree with no infinite branch is finite. Unless the branches are labelled, this requires the Axiom of Choice.

that $\mathcal{A}$ accepts some infinite tree iff Q_0 still contains the initial state q_0. To prove this, assume that q_0 is not eliminated and let $q_1, \ldots, q_m$ be the final states remaining in Q_0. Since $m \geq 1$ by non-emptiness of T_{q_0} and that also $T_{q_1}, \ldots, T_{q_m}$ are nonempty we know that the set given by the equation (11) is nonempty, and that it is a subset of $T(\mathcal{A})$.

To show the converse result, since $T(\mathcal{A}) \neq \emptyset$ the state q_0 will not be eliminated and the set can be recognized by the procedure specified in the results of Theorem 4.7. Thus, we have the following result:

Theorem 4.8 *The emptiness problem for Büchi tree automata is decidable.*

An example tree τ in a non-empty Büchi recognizable set T can be obtained by choosing finite trees $\tau_0, \ldots, \tau_m$ from the sets $T_{q_0}, \ldots, T_{q_m}$ which the above construction produces, and setting $\tau = \tau_0 \cdot (\tau_1, \ldots, \tau_m)^\omega$. In this infinite tree τ the number of distinct subtrees τ_u is bounded by $\sum_i |\,dom(\tau_i)|$ and hence it is finite. This finiteness condition on trees is termed as *regularity* and trees that satisfy it are called *regular* trees.

Definition 4.9 *An infinite tree $\tau \in T_A^\omega$ is said to be* regular *if there are only finitely distinct subtrees τ_u in τ, where $u \in \{0,1\}^*$.*

An equivalent definition would be that $\tau : \{0,1\}^* \to A$ is regular iff there is a finite automaton $\mathcal{A}$ over finite words which can "generate" τ in the sense that the states of $\mathcal{A}$ are partitioned into sets $Q_a (a \in A)$ such that $\mathcal{A}$ reaches a state in Q_a via input u iff $\tau(u) = a$. In terms of regular expressions this means that for each letter $a \in A$, there is a regular expression r_a which defines the language $\{u \in \{0,1\}^* \mid \tau(u) = a\}$.

We can extend the above consideration of Büchi recognizable sets to show that any nonempty Rabin recognizable set contains a regular tree, and thereby see that the emptiness problem for Rabin tree automata is decidable. We state the result here without proof (see Rabin [16] or Thomas [20]).

Theorem 4.10

1. *Any nonempty Rabin recognizable set of trees contains a regular tree.*

2. *The emptiness problem for Rabin tree automata is decidable.*

Since unary regular trees are ultimately periodic ω-words, Theorem 4.10 can be seen as generalizing the decidability result for Büchi automata given in Theorem 3.6.

4.2 Decidability of SnS

For a transfer of the preceding results from automata theory to logic, trees are represented as model-theoretic structures. If A is the alphabet $\{0,1\}^n$, a tree $\gamma \in T_A^\omega$ is coded by a model of the form

$$\Gamma_\gamma = (\{0,1\}^*, \epsilon, S_0, S_1, <, P_1, \ldots, P_n),$$

where S_0, S_1 are the two successor functions over $\{0,1\}^*$ with $S_0(w) = w0$ and $S_1(w) = w1$, $<$ is the proper prefix relation over $\{0,1\}^*$ and $P_1, \ldots, P_n$ are subsets of $\{0,1\}^*$ with $w \in P_i$ iff the i^{th} component of $t(w)$ is 1.

The interpreted formalism $S2S$, or the *second order theory of two successors* is made up from variables $x, y, \ldots$ ranging over elements and variables $X, Y, \ldots$ ranging over subsets of $\{0,1\}^*$ (See Section 3.2.1). *Terms* are obtained from the individual variables $x, y, \ldots$ and the constant ϵ by applications of S_0 and S_1. *Atomic formulas* are of the form $t = t', t < t', t \in X$, where t, t' are terms and X is a set variable. Arbitrary *formulas* are generated from atomic formulas by Boolean connectives and the quantifiers $\forall, \exists$ which range over elements and subsets of $\{0,1\}^*$. If $\varphi(X_1, \ldots, X_n)$ is an $S2S$-formula and Γ_γ a tree model as denoted above, $\Gamma_\gamma \models \varphi(X_1, \ldots, X_n)$ if φ is satisfied in Γ_γ with P_i as interpretation for X_i. Let

$$T(\varphi) = \{\gamma \in T_A^\omega \mid \Gamma_\gamma \models \varphi(X_1, \ldots, X_n)\}$$

If $T = T(\varphi)$ for some $S2S$-formula φ, T is *definable* in $S2S$.

The "weak" system called $WS2S$ is obtained when the set quantifiers range over finite subsets of $\{0,1\}^*$ only. If $T = T(\varphi)$ for some $WS2S$-formula φ (*i.e.,* some $S2S$-formula under this weak interpretation), T is definable in $WS2S$ or simply *weakly definable.*

These definitions carry through for finite tree models Γ_γ where $\gamma \in T_A$. In this case there is no difference between the strong and the weak interpretation.

Note that $S1S$ as defined in Section 3.2.1 results from taking $S2S$ and deleting the successor function S_1 and by restriction of the underlying models to the domain 0^*. In a way similar to $S1S$, the primitive ϵ and $<$ are definable in terms of S_0, S_1 and hence do not add power but are rather used for easier formalization.

The infinite binary tree in these model theoretic terms stands for the structure $(\{0,1\}^*, S_0, S_1)$.

Example 4.11 *Using abbreviations such as $\leq$ and $X \subseteq Y$, the following are examples of $S2S$ formulas:*

$$\begin{aligned}
\mathrm{Chain}(X) &: \forall x \forall y(x \in X \wedge y \in X \rightarrow x < y \vee x = y \vee y < x), \\
\mathrm{Path}(X) &: \mathrm{Chain}(X) \wedge \neg \exists Y(X \subseteq Y \wedge X \neq Y \wedge \mathrm{Chain}(Y)), \\
x \preceq y &: x \leq y \vee \exists z(S_0(z) \leq x \wedge S_1(z) \leq y), \\
\mathrm{Min}(x, X) &: \forall z(z \in X \rightarrow x \preceq z), \\
\mathrm{Fin}(X) &: \forall Y(Y \subseteq X \wedge Y \neq \emptyset \rightarrow (\exists y \mathrm{Min}(y, Y) \wedge \exists y \mathrm{Min}(y, Y))).
\end{aligned}$$

$x \preceq y$ gives the total lexicographical ordering of $\{0,1\}^*$. $\mathrm{Min}(x, X)$ means that x is $\preceq$-minimal in X. And $\mathrm{Fin}(X)$ shows definability of finiteness in $S2S$ and hence showing that $WS2S$ can be interpreted in $S2S$.

Consider as an example of a tree language the set T_0 from Example 4.3 using alphabet $\{0,1\}$ instead of the original alphabet of $\{a, b\}$. The $S2S$-formula defining this language is:

$$\varphi(X_1) = \exists Y(\mathrm{Path}(Y) \wedge \forall x(x \in Y \rightarrow \exists y(y \in Y \wedge x < y \wedge y \in X_1))).$$

The analog of Büchi's theorem (see Theorem 3.10) for sets of trees can now be stated.

Theorem 4.12

1. *A set $T \subseteq T_A$ of finite trees is definable in $(W)S2S$ iff T is recognizable. (Doner [5], Thatcher and Wright [19])*

2. *A set $T \subseteq T_A^\omega$ is definable in $S2S$ iff T is Rabin recognizable. (Rabin [13])*

Proof. Assume $A = \{0,1\}^n$. For the implications from right to left, formalize the acceptance condition for the given tree automaton $\mathcal{A}$. Assume for case 2 that the Rabin tree automaton has states $0, \ldots, m$ and the accepting pairs $(L_1, U_1), \ldots, (L_r, U_r)$. $T(\mathcal{A})$

is defined by a $S2S$-formula which is constructed in a similar way as in the proof of Theorem 3.10. The $S2S$-formula is:

$$\begin{aligned}\exists Y_0 \ldots \exists Y_m (\text{“}Y_0, \ldots, Y_m \text{ represent a run of } \mathcal{A} \text{ on } X_1, \ldots, X_n\text{”} \\ \wedge \forall Z(\text{Path}(Z) \rightarrow \bigvee_{1 \leq i \leq r} (\bigwedge_{j \in L_i} \exists^{<\omega} x(x \in Z \wedge x \in Y_j) \\ \wedge \bigvee_{j \in U_i} \exists^{\omega} x(x \in Z \wedge x \in Y_j)))) \end{aligned} \qquad (12)$$

The converse is also shown in a way exactly paralleling the method used in Theorem 3.10: First $S2S$ is reduced to a pure second order formalism $S2S_0$ with atomic formulas of the form $S_0(X_i, X_j)$, $S_1(X_i, X_j)$, $X_i \subseteq X_j$ only. Induction over $S2S_0$-formulas $\varphi(X_1, \ldots, X_n)$ shows (Rabin) recognizability of $T(\varphi)$. The steps for $\vee$ and $\exists$ are easy as the nondeterministic (Rabin) automata are closed under union and projection (see Theorem 4.6). To show case 1 apply Theorem 2.9; the proof for Rabin automata follows similarly. Concerning negation use again Theorem 2.9 and for case 2 use the complementation result in Theorem 4.6. □

Theorem 4.12 completes the cycle of results that provides the link between context free languages, recognizable sets and definability in $S2S$.

5 Conclusion

Figure 6 shows the results that have been covered in this paper and their relation. Dotted lines indicate progression between the topics. Essentially, the paper shows how the notion of a recognizable set can be used to characterize context free languages, and definability in $S2S$ and hence giving us a descriptive complexity result for context free languages in terms of definability in $S2S$.

Some instances of other applications of the theories presented in this paper are apparent in the notion of regular trees (see Definition 4.9), where regular trees can be seen as the simplest infinte terms and are useful in the semantics of program schemes where they appear as unravellings of program flow. Courcelle [4] is a survey paper of results in this area.

Other extensions of the results in this paper are to the tree sets of the *indexed languages* (or, IL) where it is well known that *CFL* $\subset$ *IL* $\subset$ *CSL*. Schimpf and Gallier [17] introduce

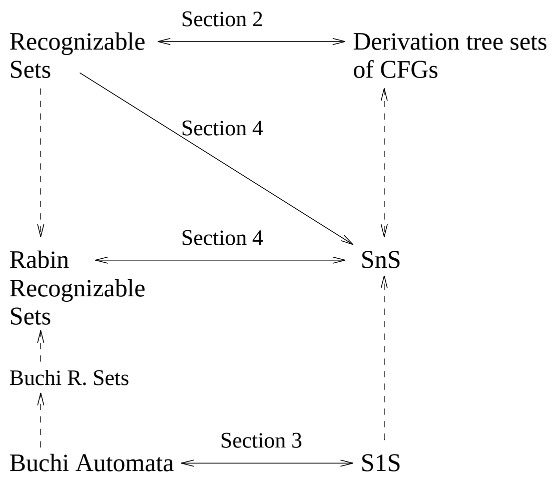

Figure 6: Results covered in the paper.

an extension to tree automata termed as *tree pushdown automata* which addresses this issue. Along the same lines, it is well known that the class of *mildly context senstive* languages which include *Tree Adjoining Languages* and *Linear Indexed Languages*, where

$$CFL \subset MCSL \subset IL \subset CSL$$

Linear versions of the Schimpf-Gallier tree automaton are equivalent to the tree sets of a mildly context-sensitive grammar formalism (from [9]).

There are a variety of results about tree acceptors and automata over infinite sequences and trees which were not considered in this paper. Thomas [20] is a survey paper which covers many important results in this area.

References

[1] J. R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel, editor, *Proceedings of the International Congress on Logic, Methodology and Philosophy of Science*, pages 1–11. Stanford Univ. Press, Stanford, CA, 1960.

[2] J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.

[3] N. Chomsky and G. Miller. Finite state languages. *Information and Control*, 1:91–112, 1958.

[4] B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[5] John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.

[6] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.

[7] Saul Gorn. Explicit definitions and linguistic dominoes. In Hart and Takasu, editors, *Systems and Computer Science*, pages 77–115. University of Toronto Press, 1965.

[8] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, Reading, MA, 1979.

[9] A. K. Joshi and Y. Schabes. Tree adjoining grammars and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree Automata*, volume 10 of *Studies in Computer Science and Artificial Intelligence*, pages 409–431. North-Holland, Amsterdam, 1992.

[10] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, number 34 in Annals of Mathematics Studies, pages 3–41. Princeton University Press, Princeton, 1956.

[11] J. Mezei and J. B. Wright. Generalized ALGOL-like languages. Technical Report RC-1528, IBM Research Paper, December 20 1965.

[12] A. Nerode. Linear automaton transformations. *Proc. AMS*, 9:541–544, 1958.

[13] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[14] M. O. Rabin. Weakly definable relations and special automata. In Y. Bar-Hillel, editor, *Mathematical Logic and Foundations of Set Theory*, pages 1–23. North-Holland, Amsterdam, 1970.

[15] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3:114–125, 1959.

[16] Michael O. Rabin. *Automata on Infinite Objects and Church's Problem*, volume 13 of *Regional Conference Series in Mathematics*. American Mathematical Society, Providence, RI, 1972.

[17] K. M. Schimpf and J. H. Gallier. Tree pushdown automata. *Journal of Computer and System Sciences*, 30:25–39, 1985.

[18] J. W. Thatcher. Characterizing derivation trees of context free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[19] J. W. Thatcher and J. B. Wright. Generalized finite automata with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.

[20] Wolfgang Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 135–191. Elsevier, 1990.