Strict LT_2 : Regular :: Local : Recognizable

James Rogers

University of Central Florida

1 Introduction

In the early sixties, in the context of exploring decidability of properties of sequences, Büchi [3] and Elgot [6] established a *descriptive* characterization of the regular sets in terms of definability in the monadic second-order theory of the natural numbers with successor—the theory that is now known as S1S. The idea is that if we interpret an alphabet Σ as a set of monadic predicate variables, then we can interpret non-empty sequences drawn from Σ, i.e., possibly infinite non-empty strings in Σ, as assignments associating each variable $\sigma \in \Sigma$ with a subset of $\mathbb{N}$: namely, the set of positions in the string at which σ occurs. The regular sets are, under this interpretation, the sets of strings that are satisfying assignments for some formula $\varphi(\Sigma)$ in the language of S1S.

To be precise, let $\mathsf{N}_1 = \langle \mathbb{N}, 0, \leq, s \rangle$ the structure of the natural numbers with zero, the usual ordering, and successor.[1] The language of S1S includes the signature of N_1, a set of variables ranging over $\mathbb{N}$ and another ranging over arbitrary subsets of $\mathbb{N}$, quantification over both sorts of variables, and the usual logical connectives. S1S is the set of all sentences in this language that are true in N_1. The set of sequences defined by a formula $\varphi(\Sigma)$ with free-variables among those in Σ are just those generated by the assignments that make $\varphi(\Sigma)$ true in N_1.

Büchi and Elgot were concerned with infinite sequences for which Büchi introduced a class of finite automata over infinite strings that are now referred to as *Büchi automata*. Büchi's central result is that S1S is decidable. He establishes this by showing that, for any S1S formula, there is a Büchi automaton that accepts exactly the set of its satisfying assignments, and that it is decidable if the set accepted by such an automaton is empty. Then $(\exists \mathbf{X})[\varphi(\mathbf{X})] \in \mathrm{S1S}$ iff the set accepted by the automaton for $\varphi(\mathbf{X})$ is non-empty.[2] As finiteness is definable in S1S, one can restrict to finite strings by restricting the sets assigned to the variables in Σ to be finite. There is an additional complication in that one must deal with the empty string, but this is inconsequential. Under these circumstances Büchi's automata become simple finite state automata. Thus any subset of Σ^* that is definable in S1S is regular and, since the converse is easy to show, definability in S1S characterizes the regular languages.

[1] The ordering is actually inessential since it is monadic second-order definable from successor.

[2] Or, more directly but less intuitively clear, a sentence φ is in S1S iff the automaton for φ accepts; since Σ is empty, acceptance is independent of the input.

This result provides a powerful tool for establishing language-theoretic complexity results that is particularly attractive in the realm of constraint-based approaches to syntax: constraints or principles that are definable in S1S can define only regular languages. The utility of the result, of course, is limited by the weakness of the class it characterizes; the regular languages are not particularly interesting from a linguistic point of view. The obvious step, then, is to extend the result to larger language-theoretic complexity classes.

Intuitively, one can approach capturing the context-free languages in a manner similar to Büchi's characterization of the regular languages by noting that the characteristic operation in generating regular languages is concatenation of strings while the characteristic operation in generating CFLs is substitution in strings, and that substitution in strings is analogous to concatenation of trees in the following sense: substitution of γ for X in a sentential form $\alpha X \beta$ corresponds to concatenating the derivation tree for $\alpha X \beta$ with a local tree (a tree of depth one) at X expanding X to γ. Now, one can interpret trees as a generalization of the natural numbers with successor in which there are multiple successor functions, i.e., if we take $\mathsf{N}_1 = \langle \mathbb{N}, 0, \leq, s \rangle$ to be the complete unary branching tree, the complete n-branching tree is $\mathsf{N}_n = \langle T_n, \epsilon, \triangleleft^*, \trianglelefteq, r_i \rangle_{i<n}$ where:

- T_n is the complete n-branching tree domain (i.e., $\{0, \ldots, n-1\}^*$).
- ϵ is the root of T_n.
- r_i is the i^{th} successor function: $r_i(w) = w \cdot i$.
- $\triangleleft^*$ is reflexive domination: $w \triangleleft^* w \cdot v$ for all $w, v \in T_n$.
- $\trianglelefteq$ is the lexicographic order on T_n: $w \trianglelefteq v$ iff $w \triangleleft^* v$ or there is some $u \in T_n$ and $i < j$ such that $u \cdot i \triangleleft^* w$ and $u \cdot j \triangleleft^* v$.[3]

Rabin [13] has explored SnS in a manner similar to Büchi's exploration of S1S, including defining a type of finite-state automaton over infinite trees, and a similar, but much more difficult, proof of decidability of SnS for all $n \leq \omega$. Again in this context, if we restrict our attention to the finite structures we find ourselves in more familiar territory. In the finite case, Rabin automata reduce to ordinary finite-state tree automata [8]. One gets, then, that sets of finite trees are definable in SnS, $n < \omega$ (or, equivalently, sets of finite trees with bounded branching are definable in SωS) iff they are *recognizable*. The class of string languages that are the yields of recognizable sets are exactly the CFLs. Thus the move from strings (S1S) to trees (SnS) lifts the result from regular languages to CFLs: principles and constraints on trees that are definable in SnS for $n < \omega$ license only weakly context-free languages. (See [12] and [5].)

While SnS gives us a weak characterization of the CFLs—in terms of string sets—in most linguistic applications we are more interested in strong characterizations of classes of languages—in the way grammars analyze those strings, i.e., the set of structures they generate. From this point of view the result is not completely satisfactory; the recognizable sets are a strict superset of the *local sets*—the sets of derivation trees generated by CF grammars. A useful example of a set of trees separating the two is the set of binary branching trees in which

[3] Again, $\triangleleft^*$ is definable except in N_ω—the ω-branching tree—as is $\trianglelefteq$. In N_1 $\triangleleft^* = \trianglelefteq = \leq$.

at most one node is labeled 'B' while all others are labeled 'A'. The set is not local: if a CFG can rewrite any 'A' to 'AB', for instance, then it can rewrite every 'A' in this way. While the label of a node in a tree generated by a CFG depends only on the labels of its parent and siblings, tree automata assign each node a *state* (which depends only on the state and label of the parent and the states of the siblings) as well as a label. The example set is recognizable because a tree automaton can distinguish nodes that dominate 'B's from those that don't by assigning them distinct states—by storing a bounded amount of context in the state it can distinguish 'A's that may rewrite to 'B' from those that may not.

Note that if we refine the categories labeling the trees to distinguish 'A's that dominate 'B's (call them 'A/B') from those that don't (call them just 'A'), we get a set of binary branching trees in which at most one node is labeled 'B' while all others are labeled either 'A', if they do not dominate a 'B', or 'A/B', if they do, which *is* a local set. The original set, then, is a *projection* (a relabeling via a typically many-to-one mapping of categories to categories) of a local set. This approach works in general. A set of trees is a projection of a local set whenever it is possible to refine its categories into finitely many sub-categories in a way that distinguishes nodes that must expand in distinct ways. In the case of the recognizable sets this is always possible. Since the automaton can only distinguish nodes on the basis of their label and its state, if we divide each category into distinct sub-categories for each state we will be able to capture the behavior of the automaton with a CFG. Since there are only finitely many states, the refinement is finite. Thus the recognizable sets are projections of the local sets (the converse is easy to see as well) [14].

Sets of finite trees (with bounded branching), then, are definable in SnS, $n < \omega$ iff they are a projection of a local set. As the notation employed in the example is intended to imply, there is a close connection between this mechanism of refining categories on the basis of limited context and the slashed categories of GPSG and HPSG. Thus, at least in some domains, the recognizable sets are essentially the strongly context-free sets. One can argue, on the other hand, a position requiring every category distinguished by a theory of syntax to have empirical justification. Under this assumption arbitrary refinement of categories is highly suspect. If the projection is necessary, then the ontology of the theory is defective. If the original set of categories is correct, then the projection should be impermissible. In any case it would be desirable to have a notion of exactly what the formal power of the projection is. Thus the question arises, how must one restrict SnS in order to characterize the local sets themselves? There is, of course, a trivial way of doing this. We could restrict ourselves to a class of formulae that are essentially direct translations of context-free rewrite rules. But we would like a more natural class of formulae, perhaps even a class that is maximal in the sense that all non-trivial extensions of the class permit definition of non-local sets. The larger class will not have increased descriptive power—we could take the trivial translations as a normal form for the formulae in the class—but we would hope that the larger class would offer better insight into the distinction between local and recognizable sets, as well as allowing more natural expression of constraints defining local sets.

In this paper we define a sub-language of SnS that characterizes the local sets. The key insight behind this result is the fact that the local sets are characterized by "subtree substitution closure": A recognizable set is local iff it is closed under substitution of sub-trees at similarly labeled nodes. As a corollary, we get that the local sets are closed under intersection but not under union. This, of course, is in contrast to the class of context-free string languages which is closed under union but not intersection, and is in contrast to the recognizable sets as well which are closed under all Boolean operations (since the language of SnS is closed wrt the Boolean connectives). The corollary implies that any language characterizing the local sets will not be closed wrt disjunction. On the other hand, the language must support some disjunction—the labels on nodes must be contingent on the labels of the nodes that occur in the same local trees. Similarly, the language will need some form of universal quantification since the productions of a CFG specify conditions that hold for every local tree occurring in the set. But unrestricted quantification, even just first-order universal quantification, allows definition of non-local sets. For instance $(\forall x, y)[(A(x) \vee B(x)) \wedge (B(x) \wedge B(y) \rightarrow x \approx y)]$ picks out our example non-local set. We need, then, a restricted form of quantification.

We are left with a language, Loc, employing restricted quantification in the form of what are, in essence, modal operators for parent, child and left- and right-sibling. Nesting of these operators and of disjunction is restricted in such a way that labels on nodes can only depend on the labels of its parent, its children, or its siblings. We show that a set of finite trees with bounded branching is definable in Loc iff it is local. While Loc is not all that far removed from the trivial translation of CF rewriting rules, it is not hard to show that it is close to being maximal—each syntactic form we rule out can be used to define a non-local set.

Perhaps the primary significance of this result is the way in which it demonstrates how profoundly weak CFGs are as generators of trees. This weakness is highlighted if we reverse the generalization that lifted us from regular to recognizable sets and consider the class of strings we can define as unary branching trees. In this case we are left with the language Loc_1 with operators only for immediate predecessor and immediate successor. We show that the class of sets of strings definable in Loc_1 is the class of *strict 2-locally testable* languages (strict LT_2), an extremely weak subclass of the star-free languages [11].

This accounts for the title of this paper. From the point of view of descriptive complexity, the natural generalization of the class of regular languages, in moving from strings to trees, is the class of recognizable sets. The class of languages corresponding analogously to the local sets is the class of strict LT_2 languages. This is confirmed by a number of other parallels in the relationships between these classes. It is not hard to show that the strict LT_2 languages are characterized by a suffix substitution closure condition that is the string equivalent of the subtree substitution closure condition of the local sets. Similarly, if we examine, in the realm of unary branching trees, the proof of the fact that the recognizable sets are projections of local sets we get that every regular set is the projection (more typically in the context of strings, the homomorphic image) of a strict LT_2 language—in essence, a result that was arrived at via a different route by Chomsky and Schützenberger in the early 60's [4].

In the following section we formalize the notions of local and recognizable sets, and establish the distinction between them. Next, we sketch existing descriptive characterizations of the regular and context-free string languages and the recognizable sets. Then, in section 4, we develop our descriptive characterization of the local sets. Finally, we close by exploring what the characterization tells us about the relationship between these classes of sets of trees and about the formal significance of allowing the syntactic properties of categories to depend on a bounded amount of their context.

2 Local and Recognizable Sets

We begin with a formal development of the classes of trees that yield context-free languages.

2.1 Trees

The foundation of our notion of trees is Gorn's [9] view of trees as collections of node addresses.

Definition 1 (Tree-domains). A *tree-domain*, T, is a subset of $\mathbb{N}^*$ that is both "downward" and "left" closed:

$$wv \in T \Rightarrow w \in T, \text{ for all } w, v \in T$$
$$wi \in T \Rightarrow wj \in T, \text{ for all } w \in T, j < i \in \mathbb{N}$$

The *complete, n-branching tree-domain*, T_n, is the tree-domain in which every node has exactly n successors, i.e., $\{i \mid i < n\}^*$.

Most relationships in trees that have arise in linguistic theories can be resolved into relationships of (hereditary) constituency and of the left-to-right ordering of strings. These have natural definitions on tree-domains. They can be combined to provide a total order on the nodes of the tree-domain which co-incides with the lexicographic ordering of their addresses.

Definition 2. If T is a tree-domain for all $w, v \in T$, then:

- (**Domination**) $w \triangleleft^* v \overset{\text{def}}{\iff} (\exists u \in T)[wu = v]$.
- (**Precedence**) $w \prec v \overset{\text{def}}{\iff} (\exists u \in T, i < j \in \mathbb{N})[ui \triangleleft^* w \text{ and } uj \triangleleft^* v]$.
- (**Lexicographic Order**) $w \trianglelefteq v \overset{\text{def}}{\iff} w \triangleleft^* v \text{ or } w \prec v$.

Note that $\trianglelefteq$ is the ordinary lexicographic order on the strings in T.

Commonly, (but not universally) when trees are used to express syntactic analyses each node is labeled with the category of the phrase it spans. We will take the set of category labels to be an arbitrary finite alphabet Σ. Our primary objects of interest, then, are tree-domains with labels drawn from Σ.

Definition 3 (Σ-labeled trees). For Σ any finite set (an alphabet), a *Σ-labeled tree*, $\langle T, \tau \rangle$, is a tree-domain, T, along with a map $\tau : T \to \Sigma$, i.e., an assignment of Σ to the nodes of the tree-domain.

As usual, we extend τ to sequences of nodes by taking $\tau(\langle w_0, \ldots, w_n \rangle)$ to be $\langle \tau(w_0), \cdots, \tau(w_n) \rangle$.

The defining characteristic of the "context-freeness" of a syntactic theory is that it is (or can be) stated in terms of *local* relationships—relationships that hold between a category and its immediate constituents (i.e., between categories labeling a node and its children) or between phrase mates (categories labeling siblings). Such relationships can be expressed as local trees.

Definition 4 (Local trees). The *local tree* occurring at a node w in a Σ-labeled tree $\mathcal{T} = \langle T, \tau \rangle$ is the pair

$$L(\mathcal{T}, w) \stackrel{\text{def}}{=} \langle \tau(w), \tau(\langle wi \mid i \in \mathbb{N}, wi \in T \rangle) \rangle .$$

(That is, the pair consisting of the label of that node and the sequence of labels of its children.)

The set of local trees occurring in $\mathcal{T}$ is

$$L(\mathcal{T}) \stackrel{\text{def}}{=} \{ L(\mathcal{T}, w) \mid w \in T \} .$$

Finally, there are specific sequences of the nodes or labels in a tree and, in particular, sets of such sequences (i.e., languages) that have distinguished roles in the theory.

Definition 5. Suppose $\mathcal{T} = \langle T, \tau \rangle$ is a Σ-labeled tree. Then:

- The set of **leaves** of $\mathcal{T}$ is the set of its atomic categories—those labeling nodes with no children:

$$l(\mathcal{T}) \stackrel{\text{def}}{=} \{ \sigma \in \Sigma \mid \langle \sigma, \epsilon \rangle \in L(\mathcal{T}) \}$$

- The **yield** of $\mathcal{T}$ is the sequence of labels of the leaves of $\mathcal{T}$ in left-to-right order:

$$\mathrm{Yd}(\mathcal{T}) \stackrel{\text{def}}{=} \tau(\langle w_i \mid w_i \in l(\mathcal{T}), i < j \Rightarrow w_i \prec w_j \rangle).$$

- A **path** in $\mathcal{T}$ is a set $p \subseteq T$ where:
  1. $\varepsilon \in p$
  2. $w \in p$ and $w0 \in T \Rightarrow (\exists i)[wi \in p]$
     (i.e., if w is in p and has children in T then some child of w is in p.)
  3. $w, v \in p \Rightarrow w \triangleleft^* v$ or $v \triangleleft^* w$
     (i.e., p is linearly ordered by $\triangleleft^*$.)

– The **path language** of $\mathcal{T}$ is the set of words in Σ^* that label paths in $\mathcal{T}$ (in order of domination):

$$\mathrm{PL}(\mathcal{T}) \stackrel{\text{def}}{=} \{\tau(\langle w_0, \ldots, w_n \rangle) \mid \{w_i \mid i \leq n\} \text{ is a path in } \mathcal{T}, i < j \Rightarrow w_i \triangleleft^* w_j\}\,.$$

Again as usual, we extend these to a set of trees $\mathbb{T}$ by taking the union over each tree in the set, e.g.,

$$L(\mathbb{T}) \stackrel{\text{def}}{=} \bigcup_{\mathcal{T} \in \mathbb{T}} [L(\mathcal{T})].$$

In particular, the yield of a set of trees $\mathbb{T}$ is the string language associated with that set

$$\mathrm{Yd}(\mathbb{T}) \stackrel{\text{def}}{=} \{\mathrm{Yd}(\mathcal{T}) \mid \mathcal{T} \in \mathbb{T}\}\,.$$

2.2 Local Sets

While Context-free grammars are familiar objects, we present them in a somewhat unusual manner, one that is chosen to emphasize the relationships between CFGs and the local sets and between CFGs and tree automata.

Definition 6 Context-free grammars. A (positive) *context-free grammar* (CFG) is a finite set of local trees over some alphabet Σ:

$$G \subseteq \Sigma \times \Sigma^*, \text{ finite}\,.$$

For any $\Sigma_0 \subseteq \Sigma$ the set of trees *admitted* by G wrt Σ_0 is the set of finite trees constructed from the local trees in G in which the root is labeled with some symbol in Σ_0:

$$G(\Sigma_0) \stackrel{\text{def}}{=} \{\mathcal{T} = \langle T, \tau \rangle \mid T \text{ finite}\,,\ \tau(\epsilon) \in \Sigma_0, \text{ and } L(\mathcal{T}) \subseteq G\}$$

The set of terminal symbols of G is:

$$\Lambda_G \stackrel{\text{def}}{=} \{\sigma \in \Sigma \mid \langle \sigma, \epsilon \rangle \in G\}\,.$$

This definition generalizes the standard notion of CFG in two ways: we allow a finite set of start symbols (rather than just one), and we allow productions for terminal symbols. Thus terminal symbols may label interior nodes of the tree, although, since leaves are local trees with empty sequences of children and must be licensed by G, only terminal symbols may label the leaves (i.e., $l(\mathcal{T}) \subseteq \Lambda(G)$). These generalizations are inconsequential for string languages, of course, but are significant when considering the set of trees admitted by a grammar, where the standard notion is unnaturally restrictive. Note that all grammars in this form are *positive*—they do not admit languages including the empty string. Since local trees with empty sequences of children are taken to be terminals, there is no way of erasing a non-terminal.

It should be clear that a tree is admitted by a CFG iff it is a derivation tree generated by that CFG in its more typical presentation.

Definition 7 (Local sets). A set of trees is *local* iff it is the set of trees admitted by some CFG.

The *context-free (string) languages* are exactly the yields of the local sets.

2.3 Properties of the Local Sets

In this section we establish a closure property that characterizes the local sets—they contain every tree that can be derived from the other trees in the set by substituting any subtree for another rooted at a similarly labeled node. This implies non-closure of local sets under union and complement, a property that will play a prominent role in developing their descriptive characterization.

Definition 8 (Subtrees). Suppose $\mathcal{T} = \langle T, \tau \rangle$ is a Σ-labeled tree for some Σ and $w \in T$. Then:

- $T{\downarrow}_w \stackrel{\text{def}}{=} \{v \mid wv \in T\}$, the *sub-domain* of T *rooted at* w.
- $T{\uparrow}_w \stackrel{\text{def}}{=} T \setminus \{wv \mid v \in \mathbb{N}^*\}$, the *sub-domain* of T *properly above* w.
- $\mathcal{T}{\downarrow}_w \stackrel{\text{def}}{=} \left\langle T{\downarrow}_w, \tau|_{T{\downarrow}_w} \right\rangle$, the *sub-tree* of $\mathcal{T}$ *rooted at* w.
- $\mathcal{T}{\uparrow}_w \stackrel{\text{def}}{=} \left\langle T{\uparrow}_w, \tau|_{T{\uparrow}_w} \right\rangle$, the *sub-tree* of $\mathcal{T}$ *properly above* w.

Definition 9 (Tree substitution). Suppose $A = \langle T^A, \tau^A \rangle$ and $B = \langle T^B, \tau^B \rangle$ are labeled trees (not necessarily over the same alphabet) and $w \in T^A$. Let

- $T^A \stackrel{w}{\leftarrow} T^B \stackrel{\text{def}}{=} T{\uparrow}_w \cup \{wv \mid v \in T^B\}$ and
- $(\tau^A \stackrel{w}{\leftarrow} \tau^B)(v) \stackrel{\text{def}}{=} \begin{cases} \tau^B(u), & \text{if } v = wu \\ \tau^A(v), & \text{otherwise} . \end{cases}$

Then $A \stackrel{w}{\leftarrow} B \stackrel{\text{def}}{=} \left\langle T^A \stackrel{w}{\leftarrow} T^B, \tau^A \stackrel{w}{\leftarrow} \tau^B \right\rangle$.

Thus $A \stackrel{w}{\leftarrow} B$ is the result of substituting B for the sub-tree rooted at w in A.

Lemma 10 Subtree substitution closure. *A set, $\mathbb{T}$, of finite trees with bounded branching is local iff for all $A = \langle T^A, \tau^A \rangle, B = \langle T^B, \tau^B \rangle \in \mathbb{T}$, all $w \in T^A$, and all $v \in T^B$ if $\tau^A w = \tau^B v$ then $A \stackrel{w}{\leftarrow} B{\downarrow}_v \in \mathbb{T}$.*

That is, $\mathbb{T}$ is a local set iff it is closed under substitution of subtrees rooted at similarly labeled nodes.[4]

Proof. Let

$$G^{\mathbb{T}} = L(\mathbb{T}), \quad \Sigma_0^{\mathbb{T}} = \{\sigma \mid (\exists \langle T, \tau \rangle \in \mathbb{T})[\tau(\varepsilon) = \sigma]\} .$$

$G^{\mathbb{T}}$ is the set of all local trees occurring in $\mathbb{T}$, $\Sigma_0^{\mathbb{T}}$ the set of all symbols labeling the root of some tree in $\mathbb{T}$. $L(\mathbb{T})$ is finite since Σ is finite and the branching is bounded.

It is easy to see that every $\mathcal{T}$ in $\mathbb{T}$ is also in $G^{\mathbb{T}}(\Sigma_0^{\mathbb{T}})$.

For the converse, suppose that $\mathcal{T} = \langle T, \tau \rangle \in G^{\mathbb{T}}(\Sigma_0^{\mathbb{T}})$. Choose $\mathcal{T}_\varepsilon = \langle T_\varepsilon, \tau_\varepsilon \rangle \in \mathbb{T}$ such that $\tau_\varepsilon(\varepsilon) = \tau(\varepsilon)$. Since $\tau(\varepsilon) \in \Sigma_0^{\mathbb{T}}$, such $\mathcal{T}_\varepsilon$ exists.

For each $w \in T$, in lexicographic order:

[4] This is a simplified version of a similar result, in a somewhat different setting, due to Kit Fine [7].

- Choose $\mathcal{T}' = \langle T', \tau' \rangle \in \mathbb{T}$ and $v \in T'$ such that $L(T', v) = L(T, w)$. Such $\mathcal{T}'$ and v exist because $L(T, w) \in L(\mathbb{T})$.
- Let $\mathcal{T}_{w'} = \mathcal{T}_w \overset{w}{\leftarrow} \mathcal{T}'{\downarrow_v}$ where w' is the successor of w (in the lexicographic order).

Then:

- $\mathcal{T}_{w'} \in \mathbb{T}$ for each $w \in T$, since $\mathcal{T}_\varepsilon$ is and $\mathbb{T}$ is closed under subtree substitution.
- For each $w \in T$, the substructure of $\mathcal{T}_{w'}$ restricted to nodes in local trees rooted at ε through w in lexicographic order is equal to the substructure of $\mathcal{T}$ similarly restricted.

It follows that, if w is the maximal node in T, then $\mathcal{T} = \mathcal{T}_{w'} \in \mathbb{T}$. $\dashv$

Corollary 11. *The class of local sets is closed under intersection, but not under complement or union.*

Note, in contrast, that the class of context-free string languages is closed under union, but not under intersection or complement.

2.4 Recognizable Sets

Tree automata can be thought of as CFGs with a set of auxiliary labels (states) that are employed in controlling the derivation but do not show up in the derived tree.

Definition 12 (Tree Automata). A *tree automaton* over an alphabet Σ and a finite set of states Q is a finite set of triples:

$$\mathcal{A} \subseteq \Sigma \times Q \times Q^*, \text{ finite }.$$

For any $Q_0 \subseteq Q$ the set of trees *recognized* by $\mathcal{A}$ (wrt Q_0) is

$$\begin{aligned}\mathcal{A}(Q_0) \overset{\text{def}}{=}\\ &\{\mathcal{T} = \langle T, \tau \rangle \mid T \text{ finite and}\\ &\quad \exists r : T \to Q \text{ such that}\\ &\qquad r(\epsilon) \in Q_0 \text{ and}\\ &\qquad (\forall w \in T)[\langle \tau(w), r(w), r(\langle wi \in T \rangle)\rangle \in \mathcal{A}]\}.\end{aligned}$$

Which is to say, $\langle T, \tau \rangle$ is accepted by $\mathcal{A}$ wrt Q_0 iff it is possible to assign states in Q to the nodes in T, with the root being assigned a state in Q_0, that is consistent with $\mathcal{A}$ in the sense that if a node labeled σ is assigned state q and its children are assigned the sequence of states $\langle q_0, \ldots, q_n \rangle$ (ε if there are no children) then $\langle \sigma, q, \langle q_0, \ldots, q_n \rangle \rangle$ is in $\mathcal{A}$. The assignment r of states to nodes in the $\mathcal{T}$ is referred to a **run** of $\mathcal{A}$ (for Q_0) on $\mathcal{T}$.

A set of trees is *recognizable* iff it is $\mathcal{A}(Q_0)$ for some $\mathcal{A}$ and Q_0.

$$
\begin{aligned}
\mathcal{A}_{1\mathrm{B}} = \{\ & \langle A, S_0, \langle S_0, S_1\rangle\rangle\,, \\
& \langle A, S_0, \langle S_1, S_0\rangle\rangle\,, \\
& \langle A, S_1, \langle S_1, S_1\rangle\rangle\,, \\
& \langle A, S_0, \epsilon\rangle\,, \\
& \langle A, S_1, \epsilon\rangle\,, \\
& \langle B, S_0, \langle S_1, S_1\rangle\rangle\,, \\
& \langle B, S_0, \epsilon\rangle \qquad\qquad \}
\end{aligned}
$$

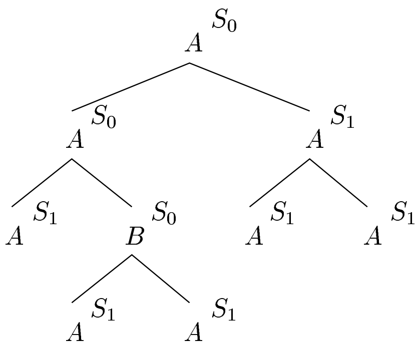

Fig. 1. Separating the recognizable and local sets.

Clearly every local set is recognizable—take Q to be Σ. The converse is not true, however, the inclusion is proper. For example, let $\mathcal{A}_{1B}$ be the tree automaton of Fig. 1. The set $\mathcal{A}_{1B}(\{S_0\})$ is the set of binary-branching trees in which at most one node is labeled B, all others are labeled A (a set we will refer to as "OneB"). To see that this is not a local set, note that if the local tree $\langle A, \langle A, B\rangle\rangle$, for instance, occurs in a tree accepted by a CFG then it must occur in that CFG. But then that CFG will license trees in which $\langle A, \langle A, B\rangle\rangle$ occurs any number of times.

The automaton of the example recognizes OneB by distinguishing nodes that may dominate a B from those that may not by assigning them distinct states (S_0 and S_1, respectively). If we make that state explicit by labeling the trees with pairs in $\Sigma \times Q$ then the set becomes a local set.

$$
\begin{aligned}
& G_{1B} = \\
& \{\ \langle A, S_0\rangle \to \langle\langle A, S_0\rangle\,, \langle A, S_1\rangle\rangle\,, \\
& \quad \langle A, S_0\rangle \to \langle\langle A, S_1\rangle\,, \langle A, S_0\rangle\rangle\,, \\
& \quad \langle A, S_0\rangle \to \langle\langle B, S_0\rangle\,, \langle A, S_1\rangle\rangle\,, \\
& \quad \langle A, S_0\rangle \to \langle\langle A, S_1\rangle\,, \langle B, S_0\rangle\rangle\,, \\
& \quad \langle A, S_0\rangle \to \quad \varepsilon, \\
& \quad \langle A, S_1\rangle \to \langle\langle A, S_1\rangle\,, \langle A, S_1\rangle\rangle\,, \\
& \quad \langle A, S_1\rangle \to \quad \varepsilon, \\
& \quad \langle B, S_0\rangle \to \langle\langle A, S_1\rangle\,, \langle A, S_1\rangle\rangle\,, \\
& \quad \langle B, S_0\rangle \to \quad \varepsilon \qquad\qquad \}
\end{aligned}
$$

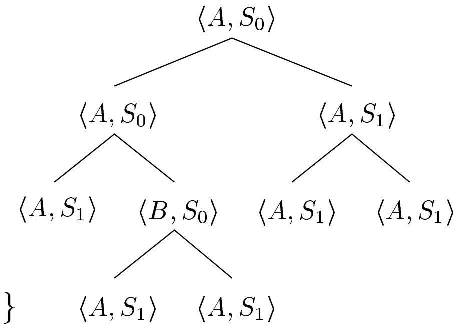

Fig. 2. Making the state explicit.

The CFG G_{1B} of Fig. 2 generates trees in which no more than one node is labeled $\langle B, S_0\rangle$, all nodes that properly dominate such a node are labeled $\langle A, S_0\rangle$, and those that don't are labeled $\langle A, S_1\rangle$. In effect we are making OneB local by refining the original categories ($\{A, B\}$) into finitely many sub-categories ($\{\langle A, S_0\rangle, \langle A, S_1\rangle, \langle B, S_0\rangle, \langle B, S_1\rangle\}$) which suffice to distinguish those nodes that may dominate Bs from those that may not. Note that the original trees are recoverable from the "refined" trees by taking the first projection of their labels. This approach works in general—whenever it is possible to refine the categories labeling a set of trees into finitely many sub-categories in such a way that the ways in which a given category may be expanded is determined entirely by its sub-category then the equivalent set of trees labeled with the sub-categories will be a local. This is always the case for recognizable sets.

Definition 13. Suppose $\mathcal{T} = \langle T, \tau\rangle$ is a tree and the range of τ is a set of n-tuples. The i^{th}-*projection* of $\mathcal{T}$, for $0 < i \leq n$ is $\langle T, \pi_i \circ \tau\rangle$—$T$ with nodes labeled with the i^{th} component of their labels in $\mathcal{T}$.

Lemma 14 Thatcher. *Every recognizable set is the projection of a local set.*

Proof. The proof simply carries out the idea of the example. Suppose $\mathbb{T} = \mathcal{A}(Q_0)$ for some tree automaton $\mathcal{A}$ over Σ and Q and set of initial states Q_0. Let $\hat{\mathcal{A}}(Q_0)$ be the set of $\Sigma \times Q$-labeled trees:

$$\hat{\mathcal{A}}(Q_0) = \{\langle T, \tau\rangle \mid \mathcal{T} = \langle T, \pi_1 \circ \tau\rangle \in \mathcal{A}(Q_0) \text{ and } \pi_2 \circ \tau \text{ is a run of } \mathcal{A} \text{ on } \mathcal{T}\}\,.$$

Which is to say that $\langle T, \tau\rangle \in \hat{\mathcal{A}}(Q_0)$ iff the first projection of $\mathcal{T}$ is a tree in $\mathcal{A}(Q_0)$ and the second projection of τ encodes a run of $\mathcal{A}(Q_0)$ on that tree.

Thus $\mathcal{A}(Q_0)$ is a projection of $\hat{\mathcal{A}}(Q_0)$. That $\hat{\mathcal{A}}(Q_0)$ is a local set follows from the fact that it is closed under subtree substitution, which follows easily from the definition of recognition. $\dashv$

Corollary 15. *A language is a CFL iff it is the yield of a recognizable set.*

This follows from the fact that we might as well the label the leaves in $\hat{\mathcal{A}}(Q_0)$ with Σ.

3 Descriptive Characterizations

We now turn to the characterization of these language-theoretic classes as classes of structures definable in certain restricted logical languages.

Definition 16 (SnS). Let $\mathsf{N}_n = \langle T_n, \epsilon, \triangleleft^*, \trianglelefteq, r_i\rangle_{i<n}$, where T_n is the complete n-branching tree-domain, ε is the root of T_n, $\triangleleft^*$ and $\trianglelefteq$ are domination and lexicographic order on T_n, respectively, and r_i takes elements of T_n to their i^{th} child.

SnS is the monadic second-order theory of N_n; *wSnS* is the *weak* monadic second-order theory of N_n, the theory in which second-order quantification is restricted to finite sets.

Finiteness is a definable property of subsets in SnS, therefore wSnS is a definable fragment of SnS.

Theorem 17 (Büchi). *S1S is decidable.*

Theorem 18 (Rabin). *SnS is decidable for all $n \leq \omega$.*

Büchi's result is a special case of Rabin's. Rabin first establishes this result for the $n = 2$ case. The proof is based on the idea that assignments making some formula $\varphi(\mathbf{X})$ over the signature of S2S true in N_2 are, in effect, $\mathcal{P}(\mathbf{X})$-labeled trees with domain T_2 (where $\mathcal{P}(\mathbf{X})$ is the powerset of the set of elements of $\mathbf{X}$). He gives an effective procedure for constructing, for any such $\varphi(\mathbf{X})$, a *Rabin tree-automaton*—an automaton over infinite trees which is, in essence, a tree automaton with an acceptance condition based on the sets of states that occur infinitely often along the paths of the tree—which accepts a $\mathcal{P}(\mathbf{X})$-labeled tree iff it encodes a satisfying assignment for $\varphi(\mathbf{X})$. It follows that $(\exists \mathbf{X})[\varphi(\mathbf{X})] \in S2S$ iff the set of trees accepted by this automaton is non-empty. He then shows that emptiness is decidable for sets of trees accepted by this class of automata and that the class of sets of labeled trees they accept (the class of *Rabin recognizable sets*) is closed under Boolean operations. Thus, the result extends to arbitrary formulae over the signature of S2S.

Rabin then generalizes this result to SnS for all $n \leq \omega$ by showing that N_n for any such n can be embedded in N_2 via the map $h : T_n \to (1^*0)^*$ where:

$$h(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ h(u) \cdot 1^i \cdot 0 & \text{if } w = ui. \end{cases}$$

This map locates the i^{th} child of w at the left child of the i^{th} node on the right-most branch of the subtree rooted at w. This is pretty much the standard way of interpreting an arbitrarily branching tree as a binary branching tree. The contribution of the proof is the observation that the range of h and the relations and functions of N_n in terms of the range of h are all definable on N_2.

Definition 19. A set $\mathbb{T}$ of Σ-labeled trees, where $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$, is **definable in** S$n$S iff there is a formula $\varphi(X_U, X_1, \ldots, X_m)$ such that

$$\mathsf{N}_n \models \varphi[X_U \mapsto T, X_i \mapsto \tau^{-1}(\sigma_i)] \Leftrightarrow \langle T, \tau \rangle \in \mathbb{T}.$$

In words, the trees in the set encode all and only the satisfying assignments for φ.

Lemma 20. *A set of finite trees is definable in* SnS *iff it is recognizable.*

This is a consequence of the fact that, if we restrict ourselves to sets of finite trees, Rabin's tree automata reduce to ordinary tree automata. Note that, since the language of SnS is closed wrt all Boolean connectives, this implies that recognizable sets are closed under all Boolean operations.

Theorem 21 (Büchi). *A language is regular iff it is the path language of a set of finite (unary branching) trees definable in S1S.*

This follows from the fact that the path languages of recognizable sets are regular.

Theorem 22 (Doner). *A language is context-free iff it is the yield of a set of finite trees definable in wSnS for some $n < \omega$.*

Since we are restricting ourselves to sets of finite trees and finiteness is definable in SnS, we need not restrict Doner's result to weak SnS. Similarly, when we are working with sets of trees with bounded branching we can assume that the bound is given explicitly and work in SωS.

Corollary 23. *A language is context-free iff it is the yield of a set of finite trees definable in SnS for some $n < \omega$; equivalently, if it is the yield of a set of finite trees with bounded branching that is definable in SωS.*

Thus definability in S1S and SωS characterize the regular and context-free languages in the *weak* sense—they characterize the string languages. But, in linguistic applications, we are generally more interested in the way in which a theory of syntax analyzes those strings, with the structures it assigns to them. While this makes no difference for Büchi's result, at the context free level it is significant—definability in SnS does not characterize the local sets but, rather, the recognizable sets. Whether the distinction is significant is open to debate. The mechanism of refining the categories of a recognizable set to resolve it into a local set is, in essence, the same mechanism as the slashed categories of GPSG and HPSG. In the OneB example we are distinguishing 'A's that dominate 'B's in very much the same way we might distinguish VPs that dominate an NP gap (i.e., a VP/NP) from those that do not (an ordinary VP). From one point of view, then, the distinction between local and recognizable sets is not linguistically significant.

On the other hand, to the extent that theories of syntax are intended to be accounts of naturally occurring phenomena, one would expect that the categories they distinguish should all be empirically justified—should be identifiable from observable regularities of language. Arbitrary refinement of those categories would then be highly suspect; the need for such refinement should suggest that the original ontology was incomplete, but its extension ought not be undertaken without similar justification. From this point of view the distinction between local and recognizable sets is quite significant. In any case it would be desirable to have a notion of exactly what the formal power of the projection mapping the recognizable sets into the local sets is.

4 Descriptive Characterization of the Local Sets

We would like to define a class of formulae that characterize the local sets in the same way that language of SωS characterizes the recognizable sets, i.e., sets of trees will be definable by formulae in the class iff they are local. We could do this minimally by restricting to formulae that are more or less direct translations of context-free productions, but, while this class suffices, it is overly restrictive. We would like to find a more natural class of formulae, certainly one that supports more natural expression of linguistically interesting constraints, perhaps even a class that is maximal in the sense that all non-trivial extensions of the class permit definition of non-local sets.

In this section we define such a class. As it turns out, the class is not that far removed from the minimal "direct translation of CFG rewriting rules", but it is, in fact, close to maximal—each syntactic form that we exclude can be used to license non-local sets.

The corollary to the subtree substitution closure lemma assures that there is no such class of formulae that is robust in the sense of being closed under Boolean connectives. The fact that the local sets are not closed under union implies that it cannot be characterized, for instance, by any class of formulae defined by quantifier prefix or by any other class of formulae closed under disjunction. On the other hand, the class of formulae must include some disjunction—the labels on nodes will be contingent on the labels of the nodes that occur in the same local trees. What we need then, is a class of formulae with limited disjunction.

Similarly, we will need some form of universal quantification since the productions of a CFG specify conditions that hold for every local tree occurring in the set. Unrestricted universal quantification, on the other hand, even only first-order, allows definition of non-local sets. For instance

$$(\forall x, y)[(A(x) \vee B(x)) \wedge (B(x) \wedge B(y) \rightarrow x \approx y)]$$

picks out the set *OneB*. We need, then, a restricted form of quantification as well.

4.1 Restricting Quantification

The motivating idea is that we can ensure that subtree substitution closure will hold by limiting formulae in such a way that their truth at a node depends only on one or the other of the local trees that node is a member of. We start by restricting existential quantification to nodes that are immediately related by parent or sibling—to the form:

$$(\exists y)[x \mathrm{P} y \wedge \phi(y)],$$

where y is new and $\mathrm{P} \in \{\triangleleft, \triangleright, \prec, \succ\}$ which are interpreted as follows:

$$\begin{array}{lll}
\mathsf{N}_n \models x \triangleleft y[x \mapsto w, y \mapsto v] \stackrel{\text{def}}{\iff} v = wi, i < n & & \text{(Parent)} \\
\mathsf{N}_n \models x \triangleright y[x \mapsto w, y \mapsto v] \stackrel{\text{def}}{\iff} w = vi, i < n & & \text{(Child)} \\
\mathsf{N}_n \models x \prec y[x \mapsto w, y \mapsto v] \stackrel{\text{def}}{\iff} w = ui, v = u \cdot (i+1) & & \text{(Left Sibling)} \\
\mathsf{N}_n \models x \succ y[x \mapsto w, y \mapsto v] \stackrel{\text{def}}{\iff} w = u \cdot (i+1), v = ui & & \text{(Right Sibling)}
\end{array}$$

Satisfaction of a formula employing such restricted quantification is relative to the interpretation of x. We will say that a formula of this form is made true in N_n (in general) by an assignment, s, iff it is made true by every assignment that extends s with $x \mapsto w$ for any $w \in \mathsf{N}_n$, i.e., if it is made true by s at every point in N_n. In effect, then, we limit ourselves to a single universal quantifier that takes scope over the entire formula.

This is the standard interpretation of diamond modalities from correspondence theory, where the relations underlying the modalities are parent, left sibling, and their inverses. Following this connection we will use modality-flavored abbreviations for the restricted quantifiers:

$$
\begin{aligned}
\triangleleft\, \varphi(x) &\equiv (\exists y)[x \triangleleft y \wedge \varphi(y)]\\
\triangleright\, \varphi(x) &\equiv (\exists y)[x \triangleright y \wedge \varphi(y)]\\
\prec\, \varphi(x) &\equiv (\exists y)[x \prec y \wedge \varphi(y)]\\
\succ\, \varphi(x) &\equiv (\exists y)[x \succ y \wedge \varphi(y)]
\end{aligned}
$$

Furthermore we will use:

$$\bot \equiv (\exists x)[x \triangleleft x] \qquad\qquad \top \equiv \neg\bot.$$

Restricting quantification in this way leaves us with a perfectly normal modal language for trees (see [1, 2, 10]), albeit a very weak one. It is, in fact, too weak to axiomatize the theory of trees, but it is perfectly adequate to define local sets among the class of sets of of finite trees. The problem, of course, is that it is still too strong—it is disjunctively closed.

4.2 Restricting Disjunction

Consider the set of trees sketched schematically in Fig. 3. This includes all trees of the form on the left in which the portion between the 'A's is iterated zero or more times, but does not include the tree on the right. The set is non-local since substitution of the subtree rooted at the lower C in the tree on the left for that rooted at the upper C produces the excluded tree. This set of trees is not difficult to define in our restricted language:

$$A \vee B \vee C \vee a \vee b \vee c \tag{1}$$

$$\neg\triangleright\top \rightarrow A \tag{2}$$

$$\neg\triangleleft\top \leftrightarrow a \vee b \vee c \tag{3}$$

$$A \rightarrow \triangleleft(\neg\succ\top \wedge a \wedge \prec(C \wedge \neg\prec\top) \vee \neg\succ\top \wedge c \wedge \neg\prec\top) \tag{4}$$

$$B \rightarrow \triangleleft(\neg\succ\top \wedge C \wedge \prec(b \wedge \neg\prec\top)) \tag{5}$$

$$C \rightarrow \triangleleft(\neg\succ\top \wedge (A \vee B) \wedge \neg\prec\top) \tag{6}$$

$$C \rightarrow \prec b \vee \triangleleft B. \tag{7}$$

The first portion of this is harmless. The problem is (7) which requires 'C's to either have a b on their right or a B as a child. This rules out the excluded tree.

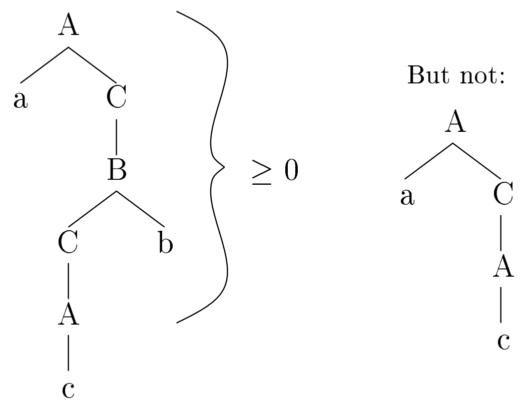

Fig. 3. A non-local set

The problematic axiom is a constraint that relates the form of the dominating local tree to that of the dominated local tree. Thus it will not, in general, be preserved if we insert that dominated tree into a new context. The intuition here is that if we rule out such constraints by limiting disjunction to conditions that hold in the same local tree then we will get invariance with respect to subtree substitution.

To this end we define three sub-languages with the modal prefixes restricted in such a way that they can express only conditions that hold immediately at a node (Loc_P), in the dominated local tree (Loc_+), or in the dominating local tree (Loc_-).

- Loc_P — Modal-free formulae
- Loc_+ — formulae over $\{\wedge, \vee, \neg, \triangleleft, \triangleright, \prec, \succ\}$ and $X(x)$ such that
  - all $\{\prec, \succ, \triangleright\}$ are in the scope of exactly one more $\triangleleft$ than $\triangleright$, and
  - all $\triangleleft$ are in the scope of exactly as many $\triangleright$ as $\triangleleft$.
- Loc_- — formulae over $\{\wedge, \vee, \neg, \triangleleft, \triangleright, \prec, \succ\}$ and $X(x)$ such that
  - all $\{\prec, \succ, \triangleright\}$ are in the scope of exactly as many $\triangleleft$ as $\triangleright$, and
  - all $\triangleleft$ are in the scope of exactly one more $\triangleright$ than $\triangleleft$.

We then limit disjunction to formulae within Loc_+ or within Loc_-.

- Loc — conjunctions of
  - disjunctions of formulae in Loc_+, and
  - disjunctions of formulae in Loc_-.

Note that $\mathsf{Loc}_\mathsf{P} = \mathsf{Loc}_+ \cap \mathsf{Loc}_-$ and thus purely propositional formulae are unrestricted.

It is not hard to find examples of non-local sets that can be defined using disjunctions of formulae in any of the forms of Loc_+ with any of the forms of

Loc_-. The form of (7) is one example, where $\prec b$ is in Loc_- and $\triangleleft B$ is in Loc_+. Another can be found by replacing it with

$$C \rightarrow \triangleleft B \vee \triangleright B$$

which also excludes the desired tree. The languages Loc_+ and Loc_- are not exhaustive. There are formulae in the full language that are in neither. These are all, in effect, of forms similar to:

$$\triangleleft\triangleleft\phi \quad \text{or} \quad \triangleright\{\prec, \succ, \triangleright\}\phi.$$

It is similarly easy to find examples of disjunctions of formulae in these forms and formulae in Loc_P that define non-local trees, for example,

$$A \rightarrow \neg\triangleright\triangleright A$$

serves just as well for (7). In this sense the language Loc is very nearly minimally restricted.

4.3 Definability in Loc Characterizes the Local Sets

Lemma 24. *Every local set is definable in* Loc.

Proof. Suppose G is a CFG over Σ and the local trees in G with root A are:

$$\langle A, \langle A_1\ A_2\ A_3\rangle\rangle\,,\ \langle A, \langle A_4\ A_5\rangle\rangle \text{ and } \langle A, \varepsilon\rangle\,.$$

We can capture the requirement that every local tree with root labeled A is one of these three with:

$$\begin{aligned} A(x) \rightarrow \triangleleft\, (&\succ \bot \wedge A_1(x) \wedge \prec (A_2(x) \wedge \prec (A_3(x) \wedge \prec \bot)) \ \vee \\ &\succ \bot \wedge A_4(x) \wedge \prec (A_5(x) \wedge \prec \bot) \ \vee \\ &\bot \quad). \end{aligned}$$

This is a disjunction in Loc_+. The conjunction of all such formulae for the local trees in G, then, is a Loc formula that enforces the condition that every local tree occurs in G. We can conjoin this with

$$\bigvee_{A \in \Sigma} [A(x)]$$

and, for some $\Sigma_0 \subseteq \Sigma$,

$$\neg \triangleright \top \rightarrow \bigvee_{A \in \Sigma_0} [A(x)]$$

which require all nodes to be labeled in Σ and the root to be labeled in Σ_0. These are disjunctions in Loc_P and Loc_-, respectively, and so the entire conjunction is in Loc. It is easy to verify that a set of finite trees satisfies this conjunction iff it is in $G(\Sigma_0)$. $\dashv$

For the converse, we begin by establishing normal forms for Loc_+ and Loc_-.

Lemma 25 Loc_+**-Normal Form.** *Every* Loc_+ *formula is expressible as a disjunction of conjunctions of* Loc_+ *formulae in which:*

- *the* $\triangleright$*-depth is 0.*
- *the* $\triangleleft$*-depth is no more than 1.*
- *no* $\{\prec, \succ\}$ *occurs outside the scope of* $\triangleleft$*.*
- *no* $\triangleleft$ *occurs inside the scope of* $\{\prec, \succ\}$*.*

That is, disjunctions of conjunctions of formulae of the form

$$(\triangleleft \{\prec, \succ\}^*)^{\leq 1} \mathsf{Loc}_\mathsf{P}.$$

Proof. The proof follows from some simple equivalences in Loc_+ (where the underlined material is the same on both sides of the equivalence):

$$\begin{aligned}
\triangleleft \underline{\{\prec, \succ\}^*}_i \triangleright \varphi(x) &\equiv \triangleleft \underline{\{\prec, \succ\}^*}_i \top \wedge \varphi(x). \\
\triangleleft \underline{\{\prec, \succ\}^*}_i \triangleright \neg\varphi(x) &\equiv \triangleleft \underline{\{\prec, \succ\}^*}_i \top \wedge \neg\varphi(x). \\
&\quad (\text{since } \triangleleft \underline{\{\prec, \succ\}^*}_i \top \rightarrow \triangleleft\!\triangleright \top.) \\
\triangleleft \neg\underline{\{\prec, \succ\}^*}_i \triangleright \varphi(x) &\equiv \triangleleft \neg\underline{\{\prec, \succ\}^*}_i \top \vee (\neg\varphi(x) \wedge \triangleleft \top). \\
\triangleleft \neg\underline{\{\prec, \succ\}^*}_i \neg \triangleright \varphi(x) &\equiv \triangleleft \neg\underline{\{\prec, \succ\}^*}_i \top \vee (\varphi(x) \wedge \triangleleft \top). \\
&\quad (\text{since } \triangleleft \top \rightarrow \triangleleft\!\triangleright \top.)
\end{aligned}$$

⊣

Lemma 26 Loc_-**-Normal Form.** *Every* Loc_- *formula, when interpreted over trees with bounded branching, is expressible as a disjunction of conjunctions of* Loc_- *formulae in which:*

- *the* $\triangleleft$*-depth is 0.*
- *the* $\triangleright$*-depth is no more than 1.*
- *no* $\{\prec, \succ\}$ *occurs inside the scope of* $\triangleright$*.*

That is, disjunctions of conjunctions of formulae of the form

$$(\{\prec, \succ\}^* \triangleright)^{\leq 1} \mathsf{Loc}_\mathsf{P}.$$

Proof. Again the proof follows from some simple equivalences in Loc_- (assuming that branching is bounded to be no more than n):

$$\begin{aligned}
\triangleright\triangleleft \varphi(x) &\equiv \triangleright \top \wedge \varphi(x) \vee \bigvee_{i<n} [\prec^i \varphi(x)] \vee \bigvee_{i<n} [\succ^i \varphi(x)]. \\
\triangleright \neg \triangleleft \varphi(x) &\equiv \triangleright \top \wedge \neg\varphi(x) \wedge \bigwedge_{i<n} [\neg \prec^i \varphi(x)] \wedge \bigwedge_{i<n} [\neg \succ^i \varphi(x)].
\end{aligned}$$

⊣

Lemma 27. *Every set of finite trees with bounded branching definable in* Loc *is local.*

Proof. This follows from the fact that every Loc formula is logically equivalent to a conjunction of formulae in either Loc_+-normal form or Loc_--normal form and is thus true at a point iff each of those conjuncts is true at that point. But the truth of a formula in Loc_+-normal form at some point w depends only on the local tree rooted at w while the truth of a formula in Loc_--normal form depends only on the local tree rooted at w's parent. It follows that this will be preserved by subtree substitution. $\dashv$

Theorem 28. *A set of finite trees with bounded branching is local iff it is definable in* Loc.

5 Consequences

Perhaps the main thing to be gotten from this characterization of the local sets is an appreciation of just how weak CFGs are as generators of trees, or, equivalently a realization that the refinement of categories on the basis of even bounded context has considerably stronger formal consequences than it might first appear. In terms of descriptive complexity, the projection that strips the contextual information lifts us from an extremely weak language, Loc, all the way to the monadic second-order language. In linguistic terms this suggests that the question of whether to admit slashed categories, i.e., whether to allow even limited refinement of the fundamental categories of the theory, is not a minor point but, in fact, has significant formal consequences.

One way to use this characterization to get an intuitive notion of the distinction between local and recognizable sets is to consider what happens when we reverse the process that lifted us from regular to context-free string languages. If we restrict our models to unary branching trees we are left with the a language in which one can talk only about immediate successor and immediate predecessor:

$$\mathsf{Loc}_1 \stackrel{\text{def}}{=} (\triangleleft\triangleright)^{\mathsf{i}}\triangleleft\mathsf{Loc}_{\mathsf{P}} \text{ or } (\triangleright\triangleleft)^{\mathsf{i}}\triangleright\mathsf{Loc}_{\mathsf{P}}.$$

Membership on sets of strings definable in this language depends only on relationships between adjacent positions in the string and can be tested by considering each such pair of positions independently. This places them within the class of *strict 2-locally testable* languages (strict LT_2) an extremely week subclass of the star-free languages [11] and, as every strict LT_2 language is definable in Loc_1, definability in Loc_1 characterizes strict LT_2. Thus our intuition that generalizing strings to trees should lift us from regular sets to local sets is, in fact, misleading. The tree analog of the class of regular sets is the class of recognizable sets, the string analog of the class of local sets is strict LT_2.

We can verify this analogy in a couple of ways. It is easy to see that strict LT_2 languages are closed under suffix substitution, the string analog of subtree substitution. Also, if we consider the string analog of Thatcher's proof that the recognizable sets are projections of local sets we get a construction in which we form sequences of pairs in which the first projections form a string in the language and the second form the sequence of states visited by a finite-state automaton in accepting that string. Such sequences form a strict LT_2 language. Thus the regular sets are projections of the strict 2-locally testable sets, a fact that turns out to be a minor variation of a result that appears in Chomsky and Schützenberger's 1963 paper on CFLs [4].

References

1. Patrick Blackburn, Claire Gardent, and Wilfried Meyer-Viol. Talking about trees. In *EACL 93*, pages 21–29. EACL, 1993.
2. Patrick Blackburn and Wilfried Meyer-Viol. Modal logic and model-theoretic syntax. In Martin de Rijke, editor, *Advances in Intensioinal Logic*, pages 27–58. Kluwer, 1996.
3. J. R. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 6:66–92, 1960.
4. N. Chomsky and M. P. Schüzenberger. The algebraic theory of context-free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, Studies in Logic and the Foundations of Mathematics, pages 118–161. North-Holland, Amsterdam, 2nd (1967) edition, 1963.
5. John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4:406–451, 1970.
6. Calvin C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–51, 1961.
7. Kit Fine. A new characterization of context-free languages. Manuscript, Sept. 1991.
8. Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.
9. Saul Gorn. Explicit definitions and linguistic dominoes. In John F. Hart and Satoru Takasu, editors, *Systems and Computer Science, Proceedings of the Conference held at Univ. of Western Ontario, 1965*. Univ. of Toronto Press, 1967.
10. Marcus Kracht. Syntactic codes and grammar refinement. 4:41–60, 1995.
11. Robert McNaughton and Seymour Papert. *Counter-Free Automata*. Number 65 in Research Monograph. MIT Press, Cambridge, MA, 1971.
12. J. Mezei and J. B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.
13. Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, July 1969.
14. 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.

This article was processed using the LaTeX macro package with LLNCS style