STRONGLY DIRECTED HYPERGRAPHS AS TREE ACCEPTORS

DAKOTAH LAMBERT

Abstract. This article presents a visualization method for bottom-up tree acceptors through the use of strongly directed hypergraphs, themselves a newly introduced type of graph. A labeled, ordered variant of the parallel aggregated ordered hypergraph framework is used for the display of these graphs, but any mechanism will suffice.

1. Introduction

Finite-state acceptors over strings are commonly represented as directed graphs, in order to provide a visualization for the mathematical objects. This ability to consider the computation as a labeled directed graph has proven useful in other ways, as well: for example, membership in given complexity classes can be decided by graph-theoretic means (cf. Caron, 2000). These string acceptors are useful in describing regular stringsets (Rabin and Scott, 1959), but Nerode (1958) demonstrates that properly context-free stringsets are beyond this capacity.

Although there exist context-free stringsets that are not finite-state in terms of their string yield, every such stringset is necessarily finite-state over trees (Rogers, 1997). Therefore a finite-state tree acceptor might be desirable in some contexts. Unlike string acceptors, tree acceptors do not have a commonly accepted visual presentation form. This article proposes two possible solutions based on a newly proposed type of graph: the strongly directed hypergraph. One form more closely resembles the directed graph representation of a string acceptor, while the other avoids the problems that plague large graphs with many edges, where structure quickly becomes unreadable through the tangle of crossing edges.

2. Background: String Acceptors

A finite-state string acceptor is a five-tuple $\langle Q, \Sigma, \delta, q_0, F\rangle$, where Q is a finite set of states, Σ a finite alphabet, $\delta\colon \Sigma \times Q \to Q$ a transition function, $q_0 \in Q$ an initial state, and $F \subseteq Q$ a set of final states. Because Q and Σ are finite, δ consists of a finite number of edges. A good visualization should account for each component of this tuple.

Standard edge-labeled graphs suffice here. For instance, the graph shown in Fig. 1 represents strings over unstressed (U), stressed (S), and primary-stressed (P) syllable types, where words begin with an unstressed syllable, stressed and unstressed syllables alternate, and the last stressed syllable is primary-stressed.

The set of vertices, $\{1, 2, 3, 4\}$ is Q. The set of labels of edges, $\{\mathrm{U}, \mathrm{S}, \mathrm{P}\}$, is Σ. Then q_0 is the state 1, denoted by its unlabeled in-edge from nowhere, and F are

2010 *Mathematics Subject Classification.* Primary 68Q45.

Key words and phrases. Finite-state methods, tree languages, visualization.

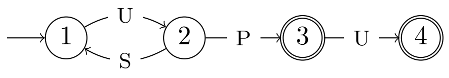


Figure 1. A representation of a finite-state string acceptor as a directed graph.

the states $\{3, 4\}$ denoted by doubled outlines. Finally the labeled edges represent δ, where an edge from a vertex x to a vertex y labeled a means that $\delta(a, x) = y$.

3. Tree-Acceptors

A bottom-up finite-state tree acceptor is similar to a string-acceptor, however there is no initial state. We have a four-tuple $\langle Q, \Sigma, \delta, F\rangle$, where Q and Σ are again a finite set of states and an alphabet, respectively, $F \subseteq Q$ is a set of final states, and $\delta\colon \Sigma \times Q^* \to Q$ is a transition function (Q^* is the set of finite sequences over Q). In this case, the element of Σ is the label of the tree node, and the sequence of states is that assigned to the node's children, in order. Initial states are not mentioned specifically, as a leaf nodes labeled x is assigned its state by $\delta(x, \varepsilon)$, where ε represents the unique sequence of length zero. Because tree grammars are assumed to have a finite maximum number of children associated with a given node, δ is again a finite map. The transitions can therefore be listed. Consider a simple example:

- $Q = \{q_a, q_b, q_S\}$
- $\Sigma = \{a, b, S\}$
- $F = \{q_S\}$

- $\delta(a, \varepsilon) = q_a$
- $\delta(b, \varepsilon) = q_b$
- $\delta(S, \langle q_a, q_b\rangle) = q_S$
- $\delta(S, \langle q_a, q_S, q_b\rangle) = q_S$

A standard directed graph cannot handle this, because each transition relates more than two states. A directed hypergraph relates a set of states to a set of states, which is useful in quantity but not in kind. Here we introduce the notion of a strongly directed hypergraph, in which edges relate not sets to sets but sequences to sequences. For a bottom-up tree-transducer, the output sequence is degenerate, always consisting of a single element. Figure 2 demonstrates one representation of such a structure. The larger white nodes are states, while the smaller black nodes are labels. The edges connecting states to labels are themselves labeled by the set of indices in the input sequence at which that state occurs. The edges connecting labels to states are unlabeled (or alternatively, always labeled 0) and indicate the target of the transition. The final states again have a doubled outline. A valid transition consists of zero or more labeled edges from state-nodes to a single label-node whose labels combine to form an initial segment of the natural numbers $\{0, \ldots, n\}$ (which can be empty for empty input-sequences), and a single unlabeled edge from this label-node to a state-node.

Another representation, perhaps simpler and more readable, appears in Fig. 3. This layout is based on the hypergraph representations of Valdivia et al. (2021) Each row is associated with a state, and each column a transition. The accepting states are framed. A node appears in a cell if that state is present in that transition; it has a doubled outline if the state is the output side of the edge, and it contains a label representing the set of indices of the input sequence at which it occurs. Leaves are represented by single-state hyperedges that have no input component,

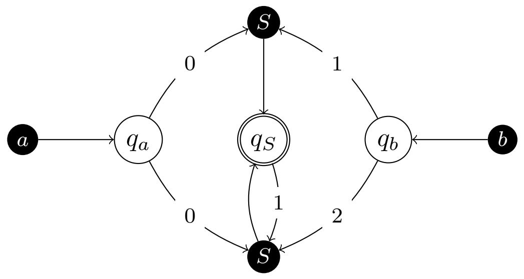


FIGURE 2. A representation of the strongly directed hypergraph associated with a tree acceptor that recognizes all and only those trees defined by the rewrite rules $S \to aSb$ and $S \to ab$.

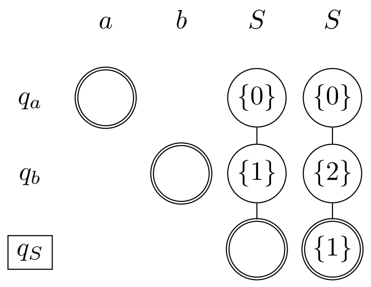


FIGURE 3. Another representation of the structure in Fig. 2.

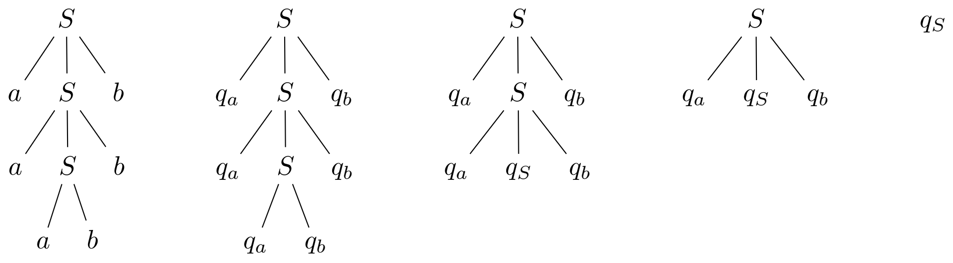


FIGURE 4. A sample accepting computation. Here subtrees are replaced by their states upon assignment.

such as the columns labeled a and b in Fig. 3. The two columns labeled S represent the transitions $\delta(S, \langle q_a, q_b \rangle) = q_S$ and $\delta(S, \langle q_a, q_S, q_b \rangle) = q_S$, respectively. Figure 4 demonstrates computation over a sample tree.

As another example, consider a grammar that generates Boolean combinations over the names "Kim", "John", and "Mary":

- $Q = \{q_{\mathrm{DP}}, q_{\mathrm{BO}}\}$
- $\Sigma = \{\mathrm{and}, \mathrm{or}, \mathrm{Kim}, \mathrm{John}, \mathrm{Mary}, \mathrm{DP}, \mathrm{BO}\}$
- $F = \{q_{\mathrm{DP}}\}$
- $\delta(\mathrm{DP}, \langle q_{\mathrm{DP}} \rangle) = q_{\mathrm{DP}}$
- $\delta(\mathrm{BO}, \langle q_{\mathrm{BO}} \rangle) = q_{\mathrm{BO}}$
- $\delta(\mathrm{DP}, \langle q_{\mathrm{DP}}, q_{\mathrm{BO}}, q_{\mathrm{DP}} \rangle) = q_{\mathrm{DP}}$
- $\delta(\mathrm{Kim}, \varepsilon) = q_{\mathrm{DP}}$
- $\delta(\mathrm{John}, \varepsilon) = q_{\mathrm{DP}}$
- $\delta(\mathrm{Mary}, \varepsilon) = q_{\mathrm{DP}}$
- $\delta(\mathrm{and}, \varepsilon) = q_{\mathrm{BO}}$
- $\delta(\mathrm{or}, \varepsilon) = q_{\mathrm{BO}}$

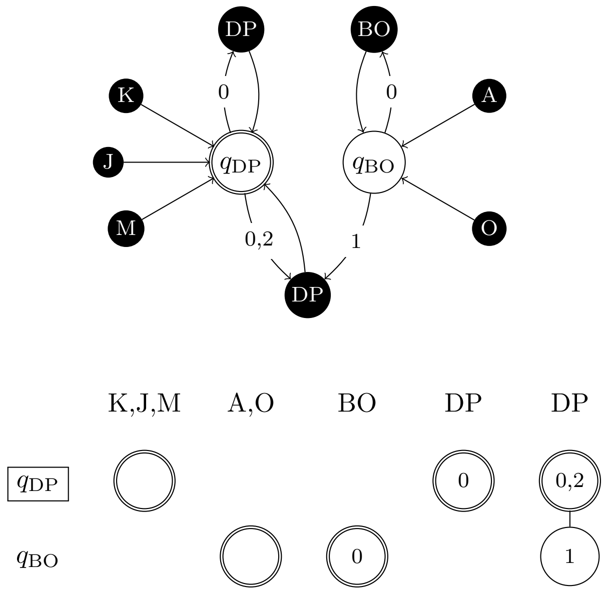
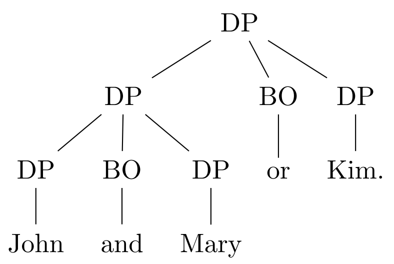


Figure 5. Representing the example on Boolean combinations. Some alphabet symbols have been simplified: **K**im, **J**ohn, **M**ary, **A**nd, and **O**r.

This is represented in Fig. 5 and would accept trees like the following:



4. Conclusion

This article has introduced a new variant of hypergraphs, strongly directed hypergraphs, that relate not sets of nodes to sets of nodes, but sequences to sequences. Using these structures, one can visually represent finite-state acceptors over trees. Two possible representations are provided: one that closely resembles the directed graph structure of string acceptors, and one that more directly encodes the transition function. The latter will never result in large sets of crossing lines, and therefore might be more readable in some situations.

In much the same way that graph-theoretic methods have paved the way for understanding some properties of string-acceptors, it is hoped that these structures may prove useful in understanding the properties of tree-acceptors.

References

Pascal Caron. Families of locally testable languages. *Theoretical Computer Science*, 242(1–2):361–376, 2000. doi: 10.1016/S0304-3975(98)00332-6.

Anil Nerode. Linear automaton transformations. In *Proceedings of the American Mathematical Society*, volume 9, pages 541–544. American Mathematical Society, August 1958. doi: 10.1090/s0002-9939-1958-0135681-9.

Michael Oser Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, April 1959. doi: 10.1147/rd.32.0114.

James Rogers. Strict LT_2 : regular :: local : recognizable. In Christian Retoré, editor, *Logical Aspects of Computational Linguistics: First International Conference, LACL '96 (Selected Papers)*, volume 1328 of *Lecture Notes in Computer Science*, pages 366–385, Berlin, 1997. Springer-Verlag. doi: 10.1007/BFb0052167.

Paola Valdivia, Paolo Buono, Catherine Plaisant, Nicole Dufournaud, and Jean-Daniel Fekete. Analyzing dynamic hypergraphs with parallel aggregated ordered hypergraph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(1):1–13, January 2021. doi: 10.1109/TVCG.2019.2933196.