— Lecture Notes —

Automata

1 Finite State Automata

1 Definition A *finite state automaton* $\mathcal{A}$ is presented as a 5-tuple $\langle \Sigma, Q, q_0, F, E \rangle$. It is often useful to think of FSAs as types of directed graphs, with labeled arcs.

> Having labeled arcs means that it is possible to have more than one arc between a given pair of nodes.

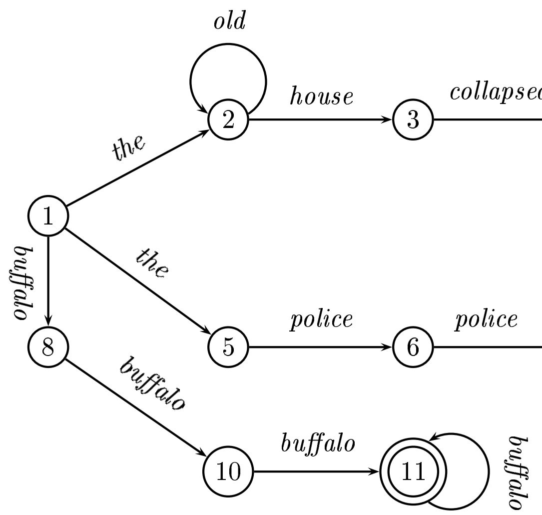
The parts are as follows.

> Σ is a finite, unranked vocabulary.
>
> Q is a finite set of *states*.
>
> $q_0 \in Q$ is the designated *start state*.
>
> $F \subseteq Q$ is the set of *final states*.
>
> $E \subseteq Q \times \Sigma^* \times Q$ is the set of *arcs* or *edges*.

Note that we allow arcs labeled with *strings* over Σ. Mainly this is to allow us to use the empty string ϵ as an arc label.

> **Fact:** Any automaton with arcs labeled by strings can be converted to an automaton with arcs labeled only by either singleton strings or ϵ. So, apart from ϵ-arcs, we usually only have edges that look like they are labeled by elements of Σ.

2 Pictures of FSAs



Conventionally we represent the initial state with an incoming arrow with no source. Here I have just stipulated that state 1 is the initial state. Final states are indicated by enclosing them in a double ring. The sets Q and Σ are usually implicit, and can be recovered by inspecting the picture.

3 Transition Functions It is common to present FSAs with a transition function δ rather than a set of arcs (i.e., a transition relation).

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \mapsto 2^Q$$

Note that we are assuming a *normal* automaton, with arcs labeled by strings of at most length 1.

This "nondeterministic" transition function can be extended to a function on longer strings δ^* by the following homomorphism.

1. $\forall q \in Q, \delta^*(q, \epsilon) = \{q\}$
2. $\forall w \in \Sigma^*, \forall a \in \Sigma \cup \{\epsilon\}$,

$$\delta^*(q, wa) = \bigcup_{q' \in \delta^*(q,w)} \delta(q', a)$$

4 The Language Recognized by a FSA Given a FSA $\mathcal{A} = \langle \Sigma, Q, q_0, F, \delta \rangle$, we define the language it accepts, $L(\mathcal{A})$ as follows.

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset\}$$

5 Running a (nondeterministic) FSA A run of a FSA is a nondeterministic computation in which, given that we are in state q looking at input symbol a, we select a single member of $\delta(q, a)$ and make that our next state. The definition of the language accepted by $\mathcal{A}$ given above can then be re-read as stating that there exists some run of $\mathcal{A}$ on w that ends in a final state.

6 Sink States We have swept a detail under the rug: seen as an edge set, E is just a relation, and it may be that for pairs $(q, a) \in Q \times \Sigma$ there is no transition defined in E. But if we think in terms of transition *functions* then we either have to say that δ is a *partial function* or we have to say what happens in δ to correspond to the absence of an arc from E.

We usually do this by adding to the automaton an extra state called the *sink state*, and sometimes denoted $\perp$. All transitions that we have "left out" are assumed to be to the sink state. Note that that entails that all transitions out of the sink state go directly back to the sink state, hence its name.

Some constructions on automata (notably those based on the general *cross-product* construction) are much simpler to express if we can assume that the transition function is total, i.e., that for every state $q \in Q$ the out-degree of q is (at least) $|\Sigma|$.

7 Reachable and Useful States It is possible to have automata such that for some state q there is no string w such that $q \in \delta^*(q_0, w)$. We call q unreachable, and observe that it can never take part in any run of the automaton. If, for a given q, there exists a string w such that $q \in \delta^*(q_0, w)$, we call q *reachable*.

It is also possible to have states which are reachable from the start state, but from which the final state is unreachable. All of these states are equivalent as far as the language definition goes; in particular, they are all sink states. Any state q for which there exists a string w such that $\delta^*(q, w) \in F$ is called *useful*.

Call an automaton such that all its states are both reachable and useful (with the possible exception of a single sink state) *reduced* and the process of constructing a reduced automaton from a given automaton *reduction*.

8 Closure and Constructions The set of FSAs is closed under the following operations. In all cases the proof is based on a construction which builds a witness to the truth of the proposition. Therefore we say that the set of FSAs is *effectively* closed under these constructions. We can break these constructions into the following groups.

9 "Utility" Constructions These constructions take a single automaton as input and return an equivalent but "better" automaton as output.

Reduction This is really two separate constructions: first remove all the unreachable states, then all the useless states, and then restrict the transition relation to the remaining states.

ϵ-Removal This and the previous construction are based on depth first traversals of a graph. As with any depth first traversal, the key is not to visit the same state twice. In this construction we want to replace all *paths* of the form $\epsilon \circ \epsilon \circ \ldots \epsilon a$ with a single arc labeled a.

Determinization This is based on the *subset construction*. The states of the output automaton are *sets* of states of the input automaton. In this way the nondeterministic function $\delta_{\mathcal{A}} : Q \times (\Sigma \cup \{\epsilon\}) \mapsto 2^Q$ becomes a deterministic function $\delta_{\mathcal{B}} : 2^{Q_{\mathcal{A}}} \times (\Sigma \cup \{\epsilon\}) \mapsto 2^{Q_{\mathcal{A}}}$

Minimization This is based on constructing *equivalence classes* of states in the input automaton. This and the subset construction both make states out of sets of old states, but note that in the subset construction we may have as many as $2^{|Q|}$ new states, but in this construction, where our new states define a *partition* of the old states, we cannot have more than $|Q|$ new states, and we will probably have fewer. *The minimization construction is only defined for deterministic automatata, however.* I think a general minimization algorithm for nondeterministic automata remains an open problem to this day.

10 Regular Constructions These are the constructions under which regular sets are closed.

Concatenation

Kleene Closure

Union

These are most simply done by gluing automata together with ϵ-transitions. Then, if you are feeling ambitious, you remove the ϵ-transitions, determinize and minimize the result.

11 Boolean Constructions The FSAs form a boolean algebra, that is, they are closed under union, intersection and complementation. (Union can be seen as belonging ambiguously to the regular and the boolean constructions; interestingly, the actual construction that witnesses its membership in these two groups is different in each case.)

Intersection

Complementation

Intersection is done by the *cross-product construction*, which makes a new automaton whose states are pairs of states from the input automata.

Complementation is done by simply flipping the final states: the final states of the new automaton are all and only the nonfinal states of the old. *This construction is another one that only works on deterministic automata. Also it requires total automata: the sink state of the input automaton will become a final state of the output automaton.*

References

[1] F. Gécseg and M. Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[2] Emmanuel Roche and Yves Schabes, editors. *Finite-State Language Processing*. MIT Press, Cambridge, Massachusetts, 1997.

[3] Seppo Sippu and Eljas Soisalon-Soininen. *Parsing Theory: Volume I Languages and Parsing*. Springer, Berlin, 1988.