Chapter 1

Nondeterminism

1.1 Non-Determinism

So far all of the finite-state automata we have considered have been deterministic. Informally, a computation is *deterministic* provided that as the computation unfolds for any input, at each step in the computation, there is at most one "next step." On the other hand, for *non-deterministic* computations, there may be multiple "next steps" in the course of the computation. Multiple output forms would seem to imply that at some point in the computation there need be multiple "next steps."

In the case of the automata we have looked at so far, there was at most one initial state. Also, the delta function had at most one output. Consequently, there was at most one path through any given automata for a particular input string or tree.

In this section, we study what happens if we add non-determinism into the picture. In other words, we will effectively change the delta *function* to a *relation* and allow more than one initial state.

As it turns out, in terms of the class of recognizable stringsets and treesets, the addition of non-determinism has little consequence. However, for the classes of string and tree transductions, the effects of non-determinism are significant.

1.1.1 Non-deterministic string acceptors

Definition 1. *A* non-deterministic finite-state acceptor (NFA) *is a tuple* $(Q, \Sigma, I, F, \delta)$ *where*

- Q *is a finite set of states;*
- Σ *is a finite set of symbols (*the alphabet*);*
- $I \subseteq Q$ *is a set of* initial *states;*
- $F \subseteq Q$ *is a set of* accepting *(*final*) states; and*
- $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times Q$*. It is the* transition relation*.*

The symbol ϵ denotes a "free" change of state; that is the state of the system can change without any input being consumed.

The *epsilon closure* of a set of states $S \subseteq Q$ is the smallest subset S' of Q satisfying these properties:

1. $S \subseteq S'$
2. For all $q, r \in Q$: if $q \in S'$ and $r \in \delta(q, \epsilon)$ then $r \in S'$.

The epsilon closure of Q is denoted $C_\epsilon(Q)$.

δ is a relation and in order to define the "process" function, we use δ to define related functions. First, we define $\delta' : Q \times \Sigma \to \wp(Q)$. Recall that for any set A, $\wp(A)$ denotes the powerset of A, which is the set of all subsets of A.

$$\delta'(q, a) = \{q' \mid (q, a, q') \in \delta\}$$

Next we define $\delta'' : \wp(Q) \times \Sigma \to \wp(Q)$ as follows.

$$\delta''(Q, a) = \bigcup_{q \in Q} \delta'(q, a)$$

Finally, we can explain how a NFA processes strings using the recursive definition below.

$$\begin{aligned} \pi(Q, \lambda) &= Q \\ \pi(Q, aw) &= \pi(C_\epsilon(\delta''(Q, a)), w) \end{aligned} \tag{1.1}$$

Consider some DFA $A = (Q, \Sigma, I, F, \delta)$ and string $w \in \Sigma^*$. If $\pi(C_\epsilon(I), w) \cap F \neq \emptyset$ then we say A *accepts/recognizes/generates* w. Otherwise A *rejects* w.

Definition 2. *The stringset recognized by A is $L(A) = \{w \in \Sigma^* \mid \pi(C_\epsilon(I), w) \cap F \neq \emptyset\}$.*

Exercise 1.

1. Non-determinism makes expressing some stringsets easier because it can be done with fewer states.
   (a) Write an acceptor for the set of strings where the last letter must be a consonant.
   (b) Write an acceptor for the set of strings where the second-to-last letter must be a consonant.
   (c) Write an acceptor for the set of strings where the third-to-last letter must be a consonant.
2. Use the plurality of initial states to show that if R and S are stringsets each recognized by some NFA that the union $R \cup S$ is also recognized by a NFA.
3. Use epsilon transitions to show that if R and S are stringsets each recognized by some NFA that the concatenation RS is also recognized by a NFA.

Theorem 1. *The class of stringsets recognizable by the NFA are exactly the regular stringsets.*

This theorem states that there is no gain in expressivity by introducing non-determinism into the models. However, as we have seen there can be gains practically in terms of how we write the models.

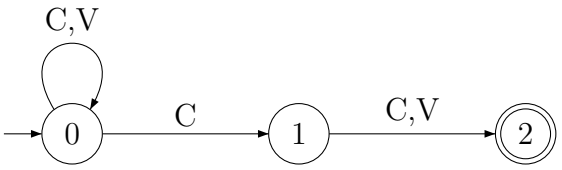
The proof of the above theorem shows how, for any NFA with state set Q, to construct an equivalent DFA. The construction is often called *the subset construction* because the states in the DFA will correspond to subsets of Q.

Here is the basic idea (ignoring epsilon transitions.) Consider any NFA $N = (Q_N, \Sigma, I_N, F_N, \delta_N)$. Then we can construct an equivalent DFA $D = (Q_D, \Sigma, I_D, F_D, \delta_D)$ as follows.

1. $Q_D = \wp(Q_N)$
2. $I_D = I_N$
3. $F_D = \{S \mid S \cap F_N \neq \emptyset\}$
4. $\delta_D(S, a) = \bigcup_{q \in S} \delta_N(q, a)$

Essentially, sets of states in the NFA correspond to distinct states in the equivalent DFA. (If the NFA includes epsilon transitions, it will be important to take into account the epsilon closure of S and the states reachable from S on a in the calculation $\delta_D(S, a)$.)

Here is an example. Consider the NFA below.

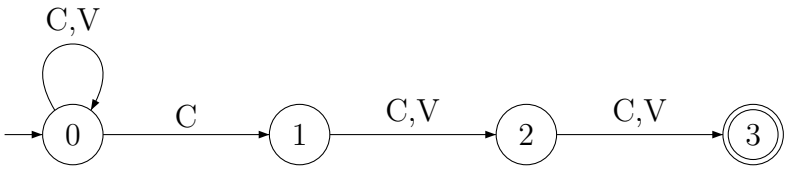


Here is the full transition table according to the subset construction. Final states are marked with the * and the initial state with →.

Q_D	C	V
$\emptyset$	$\emptyset$	$\emptyset$
→{0}	{0,1}	{0}
{1}	{2}	$\emptyset$
*{2}	$\emptyset$	$\emptyset$
{0,1}	{0,1,2}	{0,2}
*{0,2}	{0,1}	{0}
*{1,2}	{2}	{2}
*{0,1,2}	{0,1,2}	{0,2}

Exercise 2. Let's draw what this looks like. Which states are useless? If the useless states are removed, is the resulting machine complete?

Exercise 3. Write the DFA equivalent to the NFA shown below.



1.1.2 Tree acceptors

Theorem 2. *The class of treesets recognized by non-determinstic bottom-up acceptors is exactly the recognizable treesets (so the class of treesets recognized by determinstic bottom-up acceptors).*

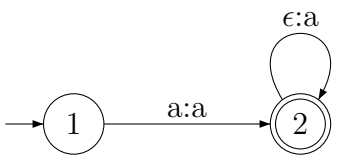
Theorem 3. *The class of treesets recognized by non-determinstic top-down acceptors is also exactly the recognizable treesets.*

1.1.3 Non-deterministic string transducers

Theorem 4. *The left projection (domain) of a regular relation is a regular stringset. The right projection (co-domain) of a regular relation is a regular stringset.*

Theorem 5. *Left and right sequential functions are proper subclasses of the relations recognized by non-deterministic string transducers.*

One reason why this is true is that epsilon transitions permit non-deterministic transducers to recognize *infinite relations* not functions. For example, the non-deterministic acceptor below relates input a to every element in $\{a, aa, aaa, \ldots\}$.



This is not possible for sequential transducers with the Σ^* monoid or the finite language monoid. However, it should be possible with the "Regular language" monoid where the outputs of the transitions are actually NFA that get "concatenated."

1.1.4 Tree transducers

Recall that a tree transducer is *linear* when the variable trees in the omega function include at most one of each variable. (In other words, trees are not copied.)

Theorem 6. *The domain of a non-deterministic tree transducer is a recognizable tree language. The image of a recognizable tree language by a linear tree transducer is recognizable.*

Theorem 7. *Generally, the class of tree relations recognized by non-deterministic top-down tree transducers and non-deterministic bottom-up tree transducers are incomparable.*

Theorem 8. *However, any linear top-down tree transducer is equivalent to a linear bottom-up tree transducer.*

1.1.5 Summary

For acceptors recognizing sets of strings or trees, non-determinism does NOT entail an increase in expressivity as the subset construction shows. However, non-determinstic computations can require exponentially fewer states.

For transducers, the current understanding is that non-determinism DOES lead to more expressive constructions. The ability of transducers to recognize inifinite relations is usually taken to be the key example. However, there are examples that do not require infinite relations as discussed in Heinz and Lai (2013).

There is much more to cover here more generally, see Hopcroft *et al.* (2001) and Comon *et al.* (2007)

Bibliography

Comon, H., M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: `http://tata.gforge.inria.fr/`. Release October, 12th 2007.

Heinz, Jeffrey, and Regine Lai. 2013. Vowel harmony and subsequentiality. In *Proceedings of the 13th Meeting on the Mathematics of Language (MoL 13)*, edited by Andras Kornai and Marco Kuhlmann, 52–63. Sofia, Bulgaria.

Hopcroft, John, Rajeev Motwani, and Jeffrey Ullman. 2001. *Introduction to Automata Theory, Languages, and Computation*. Boston, MA: Addison-Wesley.