

3.6: *Checking Acceptance and Finding Accepting Paths*

In this section we study algorithms for:

- checking whether a string is accepted by a finite automaton;
and
- finding a labeled path that explains why a string is accepted by a finite automaton.

Processing a String from a Set of States

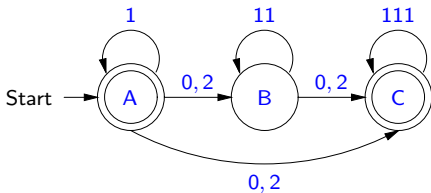
Suppose M is a finite automaton. We define a function $\Delta_M \in \mathcal{P} Q_M \times \mathbf{Str} \rightarrow \mathcal{P} Q_M$ by: $\Delta_M(P, w)$ is the set of all $r \in Q_M$ such that there is an $lp \in \mathbf{LP}$ such that

- w is the label of lp ;
- lp is valid for M ;
- the start state of lp is in P ; and
- r is the end state of lp .

When the FA M is clear from the context, we sometimes abbreviate Δ_M to Δ .

Δ Function Examples

Suppose M is the finite automaton



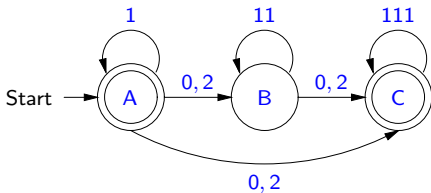
Then, $\Delta_M(\{A\}, 12111111) = \{B, C\}$, since

$$A \xRightarrow{1} A \xRightarrow{2} B \xRightarrow{11} B \xRightarrow{11} B \xRightarrow{11} B \quad \text{and} \quad A \xRightarrow{1} A \xRightarrow{2} C \xRightarrow{111} C \xRightarrow{111} C$$

are all of the labeled paths that are labeled by 12111111 , valid in M and whose start states are A .

Δ Function Examples

Suppose M is the finite automaton



Then, $\Delta_M(\{A, B, C\}, 11) = \{A, B\}$, since

$$A \xrightarrow{1} A \xrightarrow{1} A \quad \text{and} \quad B \xrightarrow{11} B$$

are all of the labeled paths that are labeled by 11 and valid in M .

An Algorithm for Calculating $\Delta(P, w)$

Suppose M is a finite automaton, $P \subseteq Q_M$ and $w \in \mathbf{Str}$. We can calculate $\Delta_M(P, w)$ as follows.

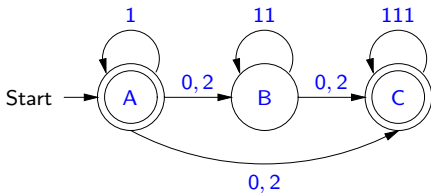
Let S be the set of all suffixes of w . Given $y \in S$, we write $\mathbf{pre} y$ for the unique x such that $w = xy$.

First, we generate the least subset X of $Q_M \times S$ such that:

- (1) for all $p \in P$, $(p, w) \in X$;
- (2) for all $q, r \in Q_M$ and $x, y \in \mathbf{Str}$, if $(q, xy) \in X$ and $q, x \rightarrow r \in T_M$, then $(r, y) \in X$.

Calculating $\Delta(P, w)$

Suppose M is the finite automaton



Here are the elements of X , when $P = \{A\}$ and $w = 2111$:

- $(A, 2111)$;
- $(B, 111)$, because of the transition $A, 2 \rightarrow B$;
- $(C, 111)$, because of the transition $A, 2 \rightarrow C$;
- $(B, 1)$, because of the transition $B, 11 \rightarrow B$;
- $(C, \%)$, because of the transition $C, 111 \rightarrow C$.

Calculating $\Delta(P, w)$

Lemma 3.6.1

For all $q \in Q_M$ and $y \in S$,

$$(q, y) \in X \quad \text{iff} \quad q \in \Delta_M(P, \text{pre } y).$$

Lemma 3.6.2

For all $q \in Q_M$, $(q, \%) \in X$ iff $q \in \Delta_M(P, w)$.

Proof. Suppose $(q, \%) \in X$. Lemma 3.6.1 tells us that $q \in \Delta_M(P, \text{pre } \%)$. But $\text{pre } \% = w$, and thus $q \in \Delta_M(P, w)$. Suppose $q \in \Delta_M(P, w)$. Since $w = \text{pre } \%$, we have that $q \in \Delta_M(P, \text{pre } \%)$. Lemma 3.6.1 tells us that $(q, \%) \in X$. \square

By Lemma 3.6.2, we have that

$$\Delta_M(P, w) = \{ q \in Q_M \mid (q, \%) \in X \}.$$

Thus, we return the set of all states q that are paired with $\%$ in X .

Checking String Acceptance and Finding Accepting Paths

Proposition 3.6.3

Suppose M is a finite automaton. Then

$$L(M) = \{ w \in \mathbf{Str} \mid \Delta_M(\{s_M\}, w) \cap A_M \neq \emptyset \}.$$

Finding Accepting Paths

Given a finite automaton M , subsets P, R of Q_M and a string w , how do we search for a labeled path that is labeled by w , valid in M , starts from an element of P , and ends with an element of R ? What we need to do is associate with each pair

$$(q, y)$$

of the set X that we generate when computing $\Delta_M(P, w)$ a labeled path lp such that lp is labeled by **pre** y , lp is valid in M , the start state of lp is an element of P , and the end state of lp is q .

With a bit of care, we can ensure that these labeled paths are as short as possible.

As we generate the elements of X , we look for a pair of the form $(q, \%)$, where $q \in R$. Our answer will then be the labeled path associated with this pair.

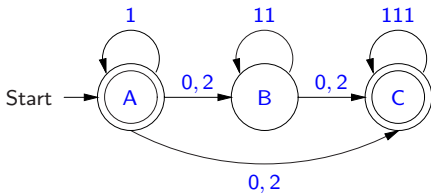
Checking Acceptance in Forlan

The Forlan module **FA** also contains the following functions for processing strings, checking string acceptance, and finding labeled paths:

```
val processStr      : fa -> sym set * str -> sym set
val accepted       : fa -> str -> bool
val findLP         : fa -> sym set * str * sym set -> lp
val findAcceptingLP : fa -> str -> lp
```

Forlan Examples

Suppose `fa` is the finite automaton



We begin by applying our four functions to `fa`, and giving names to the resulting functions:

```
- val processStr = FA.processStr fa;
val processStr = fn : sym set * str -> sym set
- val accepted = FA.accepted fa;
val accepted = fn : str -> bool
```

Forlan Examples

Continuing:

```
- val findLP = FA.findLP fa;
val findLP = fn : sym set * str * sym set -> lp
- val findAcceptingLP = FA.findAcceptingLP fa;
val findAcceptingLP = fn : str -> lp
```

Next, we'll define a set of states and a string to use later:

```
- val bs = SymSet.input "";
@ A, B, C
@ .
val bs = - : sym set
- val x = Str.input "";
@ 11
@ .
val x = [-,-] : str
```

Forlan Examples

Here are some example uses of our functions:

```
- SymSet.output("", processStr(bs, x));  
A, B  
val it = () : unit  
- accepted(Str.input "");  
@ 12111111  
@ .  
val it = true : bool  
- accepted(Str.input "");  
@ 1211  
@ .  
val it = false : bool
```

Forlan Examples

More examples:

```
- LP.output("", findLP(bs, x, bs));  
B, 11 => B  
val it = () : unit  
- LP.output("", findAcceptingLP(Str.input ""));  
@ 12111111  
@ .  
A, 1 => A, 2 => C, 111 => C, 111 => C  
val it = () : unit  
- LP.output("", findAcceptingLP(Str.input ""));  
@ 222  
@ .  
no such labeled path exists  
  
uncaught exception Error
```