

Section 3.5: Checking Acceptance and Finding Accepting Paths

In this section we study algorithms for:

- Checking whether a string is accepted by a finite automaton;
- Finding a labeled path that explains why a string is accepted by a finite automaton.

Copyright © 2003–7 Alley Stoughton

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

The L^AT_EX source of these slides, the associated book, and the distribution of the Forlan toolset are available on the WWW at <http://people.cis.ksu.edu/~stough/forlan/>.

1

(3.5) 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 ;
- r is the end state of lp .

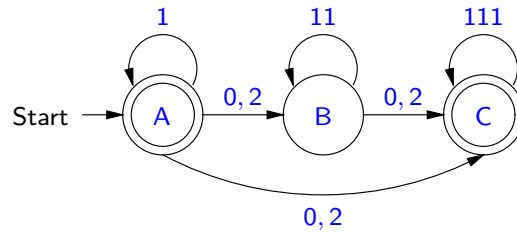
In other words, $\Delta_M(P, w)$ consists of all of the states that can be reached from elements of P by labeled paths that are labeled by w and valid for M .

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

2

(3.5) Δ Function Examples

Suppose M is the finite automaton



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

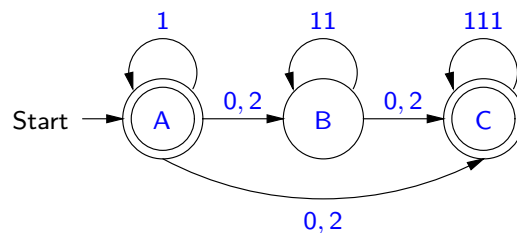
$$A \xrightarrow{1} A \xrightarrow{2} B \xrightarrow{11} B \xrightarrow{11} B \xrightarrow{11} B \quad \text{and} \quad A \xrightarrow{1} A \xrightarrow{2} C \xrightarrow{111} C \xrightarrow{111} C$$

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

3

(3.5) Δ Function Examples (Cont.)

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 .

4

(3.5) 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:

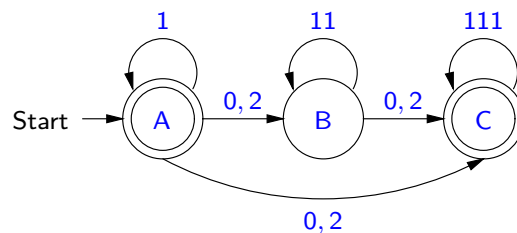
- (1) for all $p \in P$, $(p, w) \in X$;
- (2) for all $q, r \in Q_M$ and $x, y \in \mathbf{Str}$, if $(q, xy) \in X$ and $(q, x, r) \in T_M$, then $(r, y) \in X$.

We start by using rule (1), adding (p, w) to X , whenever $p \in P$. Then X (and any superset of X) will satisfy property (1). Then, rule (2) is used repeatedly to add more pairs to X . Since $Q_M \times S$ is a finite set, eventually X will satisfy property (2).

5

(3.5) Calculating $\Delta(P, w)$ (Cont.)

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

6

(3.5) Calculating $\Delta(P, w)$ (Cont.)

Lemma 3.5.1

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

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

Lemma 3.5.2

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

Proof. Suppose $(q, \%) \in X$. Lemma 3.5.1 tells us that $q \in \Delta_M(P, \mathbf{pre}(\%))$. But $\mathbf{pre}(\%) = w$, and thus $q \in \Delta_M(P, w)$.

Suppose $q \in \Delta_M(P, w)$. Since $w = \mathbf{pre}(\%)$, we have that $q \in \Delta_M(P, \mathbf{pre}(\%))$. Lemma 3.5.1 tells us that $(q, \%) \in X$. \square

By Lemma 3.5.2, 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 .

7

(3.5) Checking String Acceptance

Proposition 3.5.3

Suppose M is a finite automaton. Then

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

Proof. Suppose $w \in L(M)$. Then w is the label of a labeled path lp such that lp is valid in M , the start state of lp is s_M and the end state of lp is in A_M . Let q be the end state of lp . Thus $q \in \Delta_M(\{s_M\}, w)$ and $q \in A_M$, showing that $\Delta_M(\{s_M\}, w) \cap A_M \neq \emptyset$.

Suppose $\Delta_M(\{s_M\}, w) \cap A_M \neq \emptyset$, so that there is a q such that $q \in \Delta_M(\{s_M\}, w)$ and $q \in A_M$. Thus w is the label of a labeled path lp such that lp is valid in M , the start state of lp is s_M , and the end state of lp is $q \in A_M$. Thus $w \in L(M)$. \square

(3.5) Checking String Acceptance (Cont.)

According to Proposition 3.5.3, to check if a string w is accepted by a finite automaton M , we simply use our algorithm to generate $\Delta_M(\{s_M\}, w)$, and then check if this set contains at least one accepting state.

9

(3.5) 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 $\text{pre}(y)$, 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.

(3.5) Checking Acceptance in Forlan

The Forlan module `FA` also contains the following functions for processing strings and checking string acceptance:

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

The function `processStr` takes in a finite automaton M , and returns a function that takes in a pair (P, w) and returns $\Delta_M(P, w)$. The function `accepted` takes in a finite automaton M , and returns a function that checks whether a string x is accepted by M .

11

(3.5) Finding Labeled Paths in Forlan

The Forlan module `FA` also contains the following functions for finding labeled paths:

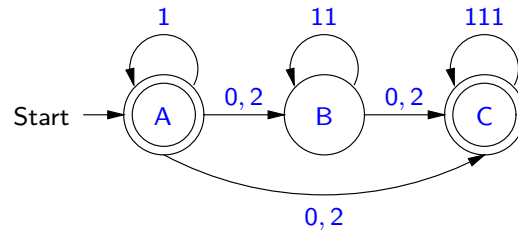
```
val findLP           : fa -> sym set * str * sym set -> lp
val findAcceptingLP : fa -> str -> lp
```

The function `findLP` takes in a finite automaton M , and returns a function that takes in a triple (P, w, R) and tries to find a labeled path lp that is labeled by w , valid for M , starts out with an element of P , and ends up at an element of R . It issues an error message when there is no such labeled path. The function `findAcceptingLP` takes in a finite automaton M , and returns a function that looks for a labeled path lp that explains why a string w is accepted by M . It issues an error message when there is no such labeled path. The labeled paths returned by these functions are always of minimal length.

12

(3.5) 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
```

13

(3.5) Forlan Examples (Cont.)

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
```

14

(3.5) Forlan Examples (Cont.)

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
```

15

(3.5) Forlan Examples (Cont.)

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
```

16