

# Section 3.4: Isomorphism of Finite Automata

---

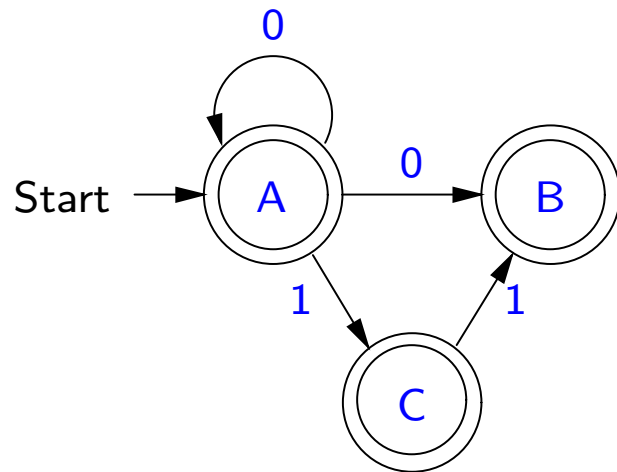
Copyright © 2003–2004 Alley Stoughton

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  $\text{\LaTeX}$  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/>.

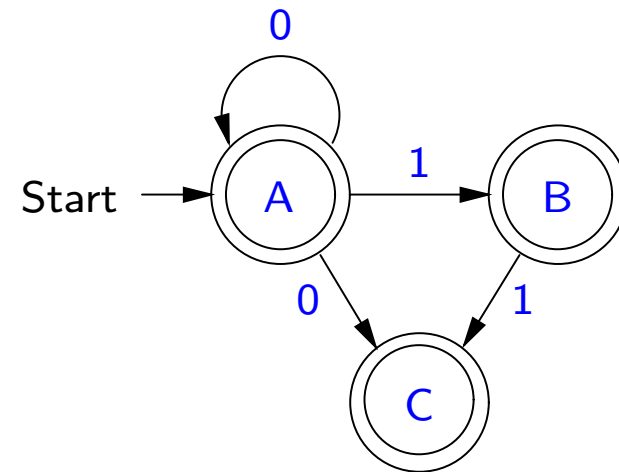
## (3.4) Introduction (Cont.)

Let  $M$  and  $N$  be the finite automata



( $M$ )

and

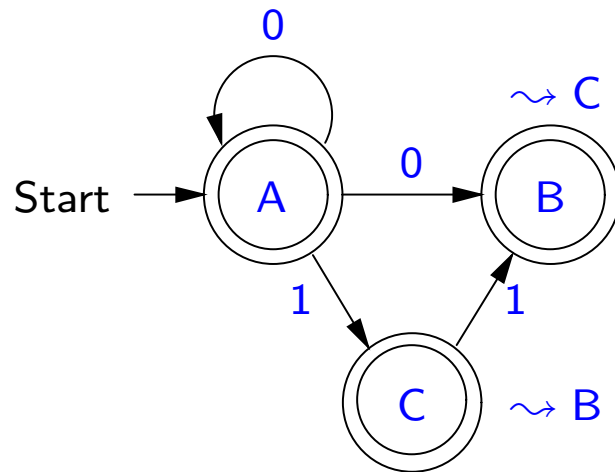


( $N$ )

How are  $M$  and  $N$  related?

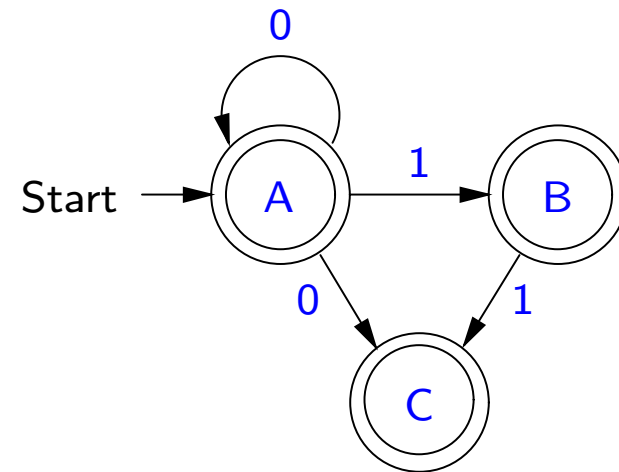
## (3.4) Introduction (Cont.)

Let  $M$  and  $N$  be the finite automata



(M)

and



(N)

How are  $M$  and  $N$  related? Although they are not equal, they do have the same “structure”, in that  $M$  can be turned into  $N$  by replacing  $A$ ,  $B$  and  $C$  by  $A$ ,  $C$  and  $B$ , respectively. When FAs have the same structure, we will say they are “isomorphic”.

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) =$
- $\{ h(q) \mid q \in A_M \} =$
- $\{ (h(q), x, h(r)) \mid (q, x, r) \in T_M \} =$

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) = s_N$ ;
- $\{ h(q) \mid q \in A_M \} =$
- $\{ (h(q), x, h(r)) \mid (q, x, r) \in T_M \} =$

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) = s_N$ ;
- $\{ h(q) \mid q \in A_M \} = A_N$ ;
- $\{ (h(q), x, h(r)) \mid (q, x, r) \in T_M \} =$

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) = s_N$ ;
- $\{ h(q) \mid q \in A_M \} = A_N$ ;
- $\{ (h(q), x, h(r)) \mid (q, x, r) \in T_M \} = T_N$ .

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) = s_N$ ;
- $\{ h(q) \mid q \in A_M \} = A_N$ ;
- $\{ (h(q), x, h(r)) \mid (q, x, r) \in T_M \} = T_N$ .

We define a relation **iso** on **FA** by:  $M \text{ iso } N$  iff there is an isomorphism from  $M$  to  $N$ . We say that  $M$  and  $N$  are *isomorphic* iff  $M \text{ iso } N$ .

## (3.4) Definition of Isomorphism

An *isomorphism*  $h$  from an FA  $M$  to an FA  $N$  is a bijection from  $Q_M$  to  $Q_N$  such that

- $h(s_M) = s_N$ ;
- $\{h(q) \mid q \in A_M\} = A_N$ ;
- $\{(h(q), x, h(r)) \mid (q, x, r) \in T_M\} = T_N$ .

We define a relation **iso** on **FA** by:  $M \text{ iso } N$  iff there is an isomorphism from  $M$  to  $N$ . We say that  $M$  and  $N$  are *isomorphic* iff  $M \text{ iso } N$ .

Consider our example FAs  $M$  and  $N$ , and let  $h$  be the function

$$\{(A, A), (B, C), (C, B)\}.$$

Then  $h$  is an isomorphism from  $M$  to  $N$ . Hence  $M \text{ iso } N$ .

## (3.4) Properties of Isomorphism

### Proposition 3.4.1

The relation **iso** is reflexive on **FA**, symmetric and transitive.

**Proof.** If  $M$  is an FA, then the identity function on  $Q_M$  is an isomorphism from  $M$  to  $M$ .

If  $M, N$  are FAs, and  $h$  is a isomorphism from  $M$  to  $N$ , then the inverse of  $h$  is an isomorphism from  $N$  to  $M$ .

If  $M_1, M_2, M_3$  are FAs,  $f$  is an isomorphism from  $M_1$  to  $M_2$ , and  $g$  is an isomorphism from  $M_2$  to  $M_3$ , then the composition of  $g$  and  $f$  is an isomorphism from  $M_1$  to  $M_3$ .  $\square$

## (3.4) Properties of Isomorphism (Cont.)

### Proposition 3.4.2

Suppose  $M$  and  $N$  are isomorphic FAs. Then  $L(M) \subseteq L(N)$ .

**Proof.** Let  $h$  be an isomorphism from  $M$  to  $N$ . Suppose  $w \in L(M)$ . Then, there is a labeled path

$$lp = q_1 \xRightarrow{x_1} q_2 \xRightarrow{x_2} \cdots q_{n-1} \xRightarrow{x_{n-1}} q_n,$$

such that  $w = x_1x_2 \cdots x_{n-1}$ ,  $lp$  is valid for  $M$ ,  $q_1 = s_M$  and  $q_n \in A_M$ . Let

$$lp' = h(q_1) \xRightarrow{x_1} h(q_2) \xRightarrow{x_2} \cdots h(q_{n-1}) \xRightarrow{x_{n-1}} h(q_n).$$

Then the label of  $lp'$  is  $w$ ,  $lp'$  is valid for  $N$ ,  $h(q_1) = h(s_M) = s_N$  and  $h(q_n) \in A_N$ , showing that  $w \in L(N)$ .  $\square$

## (3.4) Properties of Isomorphism (Cont.)

### **Proposition 3.4.3**

*Suppose  $M$  and  $N$  are isomorphic FAs. Then  $M \approx N$ .*

**Proof.**

□

## (3.4) Properties of Isomorphism (Cont.)

### Proposition 3.4.3

Suppose  $M$  and  $N$  are isomorphic FAs. Then  $M \approx N$ .

**Proof.** Since  $M \text{ iso } N$ , we have that  $N \text{ iso } M$ , by Proposition 3.4.1.

□

## (3.4) Properties of Isomorphism (Cont.)

### Proposition 3.4.3

Suppose  $M$  and  $N$  are isomorphic FAs. Then  $M \approx N$ .

**Proof.** Since  $M \text{ iso } N$ , we have that  $N \text{ iso } M$ , by Proposition 3.4.1. Thus, by Proposition 3.4.2, we have that  $L(M) \subseteq L(N) \subseteq L(M)$ . Hence  $L(M) = L(N)$ , i.e.,  $M \approx N$ .  $\square$

## (3.4) Renaming States

Let  $X = \{ (M, f) \mid M \in \mathbf{FA} \text{ and } f \text{ is a bijection from } Q_M \text{ to some set of symbols} \}$ . The function  $\mathbf{renameStates} \in X \rightarrow \mathbf{FA}$  takes in a pair  $(M, f)$  and returns the  $\mathbf{FA}$  produced from  $M$  by renaming  $M$ 's states using the bijection  $f$ .

### Proposition 3.4.4

*Suppose  $M$  is an FA and  $f$  is a bijection from  $Q_M$  to some set of symbols. Then  $\mathbf{renameStates}(M, f)$  iso  $M$ .*

The following function is a special case of  $\mathbf{renameStates}$ . The function  $\mathbf{renameStatesCanonically} \in \mathbf{FA} \rightarrow \mathbf{FA}$  renames the states of an FA  $M$  to:

- $A, B$ , etc., when the automaton has no more than 26 states (the smallest state of  $M$  will be renamed to  $A$ , the next smallest one to  $B$ , etc.); or
- $\langle 1 \rangle, \langle 2 \rangle$ , etc., otherwise.

## (3.4) An Algorithm for Finding Isomorphisms

The book presents and proves the correctness of a relatively simple algorithm for finding an isomorphism from one FA to another, if one exists, and for indicating that there are no such isomorphisms, otherwise. The algorithm is based on backtracking.

## (3.4) Isomorphism Finding/Checking in Forlan

The Forlan module `FA` also defines the functions

```
val isomorphism           : fa * fa * sym_rel -> bool
val findIsomorphism      : fa * fa -> sym_rel
val isomorphic           : fa * fa -> bool
val renameStates         : fa * sym_rel -> fa
val renameStatesCanonically : fa -> fa
```

The function `isomorphism` checks whether a relation on symbols is an isomorphism from one FA to another. The function `findIsomorphism` tries to find an isomorphism from one FA to another; it issues an error message if it fails to find one. The function `isomorphic` checks whether two FAs are isomorphic. The function `renameStates` issues an error message if the supplied relation isn't a bijection from the set of states of the supplied FA to some set; otherwise, it returns the result of `renameStates`. And the function `renameStatesCanonically` acts like `renameStatesCanonically`.

## (3.4) Forlan Examples

Suppose that `fa1` and `fa2` have been bound to our example finite automata  $M$  and  $N$ , respectively. Then, here are some example uses of the above functions:

```
- val rel = FA.findIsomorphism(fa1, fa2);  
val rel = - : sym_rel  
- SymRel.output("", rel);  
(A, A), (B, C), (C, B)  
val it = () : unit  
- FA.isomorphism(fa1, fa2, rel);  
val it = true : bool  
- FA.isomorphic(fa1, fa2);  
val it = true : bool
```

## (3.4) Forlan Examples (Cont.)

```
- val rel' = FA.findIsomorphism(fa1, fa1);  
val rel' = - : sym_rel  
- SymRel.output("", rel');  
(A, A), (B, B), (C, C)  
val it = () : unit  
- FA.isomorphism(fa1, fa1, rel');  
val it = true : bool  
- FA.isomorphism(fa1, fa2, rel');  
val it = false : bool
```

## (3.4) Forlan Examples (Cont.)

```
- val rel'' = SymRel.input "";
@ (A, 2), (B, 1), (C, 0)
@ .
val rel'' = - : sym_rel
- val fa3 = FA.renameStates(fa1, rel'');
val fa3 = - : fa
- FA.output("", fa3);
{states}
0, 1, 2
{start state}
2
{accepting states}
0, 1, 2
{transitions}
0, 1 -> 1; 2, 0 -> 1 | 2; 2, 1 -> 0
val it = () : unit
```

## (3.4) Forlan Examples (Cont.)

```
- val fa4 = FA.renameStatesCanonically fa3;
val fa4 = - : fa
- FA.output("", fa4);
{states}
A, B, C
{start state}
C
{accepting states}
A, B, C
{transitions}
A, 1 -> B; C, 0 -> B | C; C, 1 -> A
val it = () : unit
- FA.equal(fa4, fa1);
val it = false : bool
- FA.isomorphic(fa4, fa1);
val it = true : bool
```