

CoE/EE460 Switching Theory

Lecture 17

Washington University
Spring 2001

<http://www.arl.wustl.edu/~lockwood/class/coe460/>

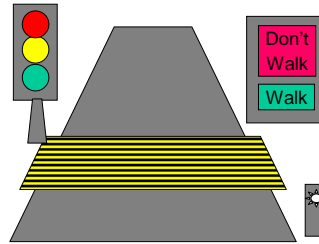
Copyright 2001, John W Lockwood

Announcements

- Announcements
 - Exam 2
 - Exam 2 : Wednesday - April 18 (in class)
 - See Notes from Lecture 16
- Reading
 - Hachtel & Somenzi : Section III / Chapter 7

FSM Synthesis

- Inputs (X)
 - B: Button
 - "Push to Walk"
- Outputs (Z)
 - R: Red
 - Y: Yellow
 - G: Green
 - W: Walk
 - D: Don't walk
- States (S)
 - Drive
 - Slow
 - Walk
- State Mapping
 - $S_1 S_0$: (Traditional)
 - $S_2 S_1 S_0$: (One-Hot)



- State Transitions
 - $S(t+1) = \delta(X, S(t))$
 - $S_0 = \delta_0(S_0, S_1, X)$
 - $S_1 = \delta_1(S_0, S_1, X)$
- Output Function
 - $Z = \lambda(S(t))$
 - $R = \lambda_R(S_0, S_1)$
 - $G = \lambda_G(S_0, S_1)$
 - ...
 - $D = \lambda_D(S_0, S_1)$

Morse Code Decoder

A	• —	N	• •	1	— • • • •	period	• • • • •
B	• • • •	O	— • • •	2	• • • • •	comma	— • • • •
C	• • • • •	P	— • • •	3	• • • • •	colon	— • • • •
D	• • • •	Q	— • • • •	4	• • • • •	query	• • • • •
E	• (1 unit)	R	• • • •	5	• • • • •	apostrophe	• • • • •
F	• • • •	S	• • • •	6	• • • • •	hyphen	• • • • •
G	• • • •	T	— (3 units)	7	• • • • •	fraction bar	— • • • •
H	• • • •	U	• • • •	8	• • • • •	parentheses	— • • • •
I	• •	V	• • • •	9	• • • • •	quotation marks	• • • • •
J	• • • • •	W	• • • •	0	— • • • •		
K	• • • •	X	• • • •				
L	• • • •	Y	• • • •				
M	• • • •	Z	• • • •				

©1994 Encyclopaedia Britannica, Inc.

- Input (X)
 - • Dot
 - — Dash
 - o Pause
- State Mapping
 - S * _
- State Transitions
 - $S(t+1) = \delta(X, S(t))$
- Output Function
 - $Z = \lambda(S(t), X)$

FSM Distinguishing Sequence

- Consider
 - Two states, s and t , of a given FSM
 - Input Sequence $\mathbf{x} = (x_0, x_1, \dots, x_{k-1})$
- Given that Input \mathbf{x} produces:
 - State sequence $(s_0, s_1, \dots, s_{k-1})$ with $s_0 = s$
 - With outputs $z^s = (z^s_0, z^s_1, \dots, z^s_{k-1})$
 - State sequence $(t_0, t_1, \dots, t_{k-1})$ with $t_0 = t$
 - With outputs $z^t = (z^t_0, z^t_1, \dots, z^t_{k-1})$
- Then
 - \mathbf{x} is a length- k distinguishing sequence for s and t if and only if $z^s_{k-1} \neq z^t_{k-1}$

K-Equivalent States

- Consider
 - Two states, s and t
 - All possible input sequences, \mathbf{x} , of length k
- Given that
 - There does not exist
 - a distinguishing sequence
 - of length k or less
 - for s and t
- Then
 - States s and t are k -equivalent, $s \equiv_k t$

Equivalent States

- Equivalent states
 - Two states of a given FSM are equivalent if and only if they are (n-1)-equivalent
- Equivalent Machine
 - Machines with n states that are each (n-1) equivalent are equivalent

Partition / Refinement : Partition P_1

- Determines set of equivalent state pairs in n-state FSM
- Start with Initial set of all states
 - $P_0 = \{ A, B, C, D, E, F \}$
- Iterate to form P_{i+1} from P_i based on differences in output for a given input
 - $P_1 = \{ A, C, E \}, \{ B, D, F \}$

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	D / 1
B	D / 0	F / 0
C	E / 0	B / 1
D	B / 0	F / 0
E	C / 0	F / 1
F	B / 0	C / 0

Partition / Refinement : Partition P_2

- Iterate to form P_{i+1} from P_i
 - $P_1 = \{ B_1 = \{ A, C, E \}, B_2 = \{ B, D, F \} \}$

- Consider Successors

- S_{i+1} from $S_i = \{ A, C, E \}$

- Given $x=0$: $S_{i+1} = \{ E:1, E:1, C:1 \}$
 - Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{ D:2, B:2, F:2 \}$
 - Cannot differentiate!

- S_{i+1} from $S_i = \{ B, D, F \}$

- Given $x=0$: $S_{i+1} = \{ D:2, B:2, B:2 \}$
 - Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{ F:2, F:2, C:1 \}$
 - Separate $\{ B, D \}, \{ F \}$

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	D / 1
B	D / 0	F / 0
C	E / 0	B / 1
D	B / 0	F / 0
E	C / 0	F / 1
F	B / 0	C / 0

Partition / Refinement : Partition P_3

- Iterate to form P_{i+1} from P_i
 - $P_2 = \{ B_1 = \{ B, D \}, B_2 = \{ F \}, B_3 = \{ A, C, E \} \}$

- Consider Successors

- S_{i+1} from $S_i = \{ B, D \}$

- Given $x=0$: $S_{i+1} = \{ D:1, B:1 \}$
 - Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{ F:2, F:2 \}$
 - Cannot differentiate!

- S_{i+1} from $S_i = \{ F \}$

- Already singleton

- S_{i+1} from $S_i = \{ A, C, E \}$

- Given $x=0$: $S_{i+1} = \{ E:3, E:3, C:3 \}$
 - Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{ D:1, B:1, F:2 \}$
 - Separate $\{ A, C \}, \{ E \}$

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	D / 1
B	D / 0	F / 0
C	E / 0	B / 1
D	B / 0	F / 0
E	C / 0	F / 1
F	B / 0	C / 0

Partition / Refinement : Partition P_4

- $P_3 = \{ B_1=\{B,D\}, B_2=\{F\}, B_3=\{A,C\}, B_4=\{E\} \}$

– S_{i+1} from $S_i=\{B,D\}$

- Given $x=0$: $S_{i+1} = \{D:1,B:1\}$
– Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{F:2,F:2\}$
– Cannot differentiate!

– S_{i+1} from $S_i=\{F\}$

- Already singleton

– S_{i+1} from $S_i=\{A,C\}$

- Given $x=0$: $S_{i+1} = \{E:4,E:4\}$
– Cannot differentiate!
- Given $x=1$: $S_{i+1} = \{B:1,D:1\}$
– Cannot differentiate!

– S_{i+1} from $S_i=\{E\}$

- Already singleton

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	D / 1
B	D / 0	F / 0
C	E / 0	B / 1
D	B / 0	F / 0
E	C / 0	F / 1
F	B / 0	C / 0

Partition / Refinement : Example 2 (1/2)

- Determines set of equivalent state pairs in n-state FSM

- Start with Initial set of all states

– $P_0 = \{ A,B,C,D,E,F,G \}$

- Iterate to form P_{i+1} from P_i based on differences in output for a given input

- $P_1^0 = \{ \{A,B,C,D,F,G\}, \{E\} \}$
- $P_1^1 = \{ \{B,C\}, \{A,D,E,F,G\} \}$

– $P_1 = \{ \{A,D,F,G\}, \{B,C\}, \{E\} \}$

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	C / 0
B	C / 0	A / 1
C	B / 0	G / 1
D	G / 0	A / 0
E	F / 1	B / 0
F	E / 0	D / 0
G	D / 0	G / 0

Partition / Refinement : Example 2 (2/2)

- Given
 - $P_1 = \{ \{A,D,F,G\}, \{B,C\}, \{E\} \}$

- Compute
 - P_2
 - P_3

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	E / 0	C / 0
B	C / 0	A / 1
C	B / 0	G / 1
D	G / 0	A / 0
E	F / 1	B / 0
F	E / 0	D / 0
G	D / 0	G / 0

Partition / Refinement : Example 3

- Given
 - State machine, as shown
 - $P_0 = \{ A, B, C \}$

- Compute
 - P_1
 - P_2

- How many states are needed?

- Why is this a trick question?

Present State	Next state/ Output with X=0	Next state/ Output with X=1
A	B / 0	C / 1
B	A / 0	C / 1
C	A / 0	C / 1