Black Box Testing Techniques

Dr. John H Robb, PMP, IEEE SEMC
UTA Computer Science and Engineering

Black Box Testing Techniques

- Typical “black-box” test analysis and design techniques include:
  - Equivalence partitioning
  - Boundary value analysis
  - Decision table testing
  - State transition analysis
  - Use Case testing
  - Logical expressions and test coverage (incl. Karnaugh maps)
  - Orthogonal Array Testing
  - Sequence Enumeration

- Black box is a bit of a misnomer – we don’t and can’t test strictly black box

- So these are correctly called Specification based test analysis and design techniques, but I use the term “black box” because it is so commonly used

Specifying a System

If the systems planners and customer do not specify what is expected in all types of interactions with the system, i.e. the behavior of the system, someone else will.

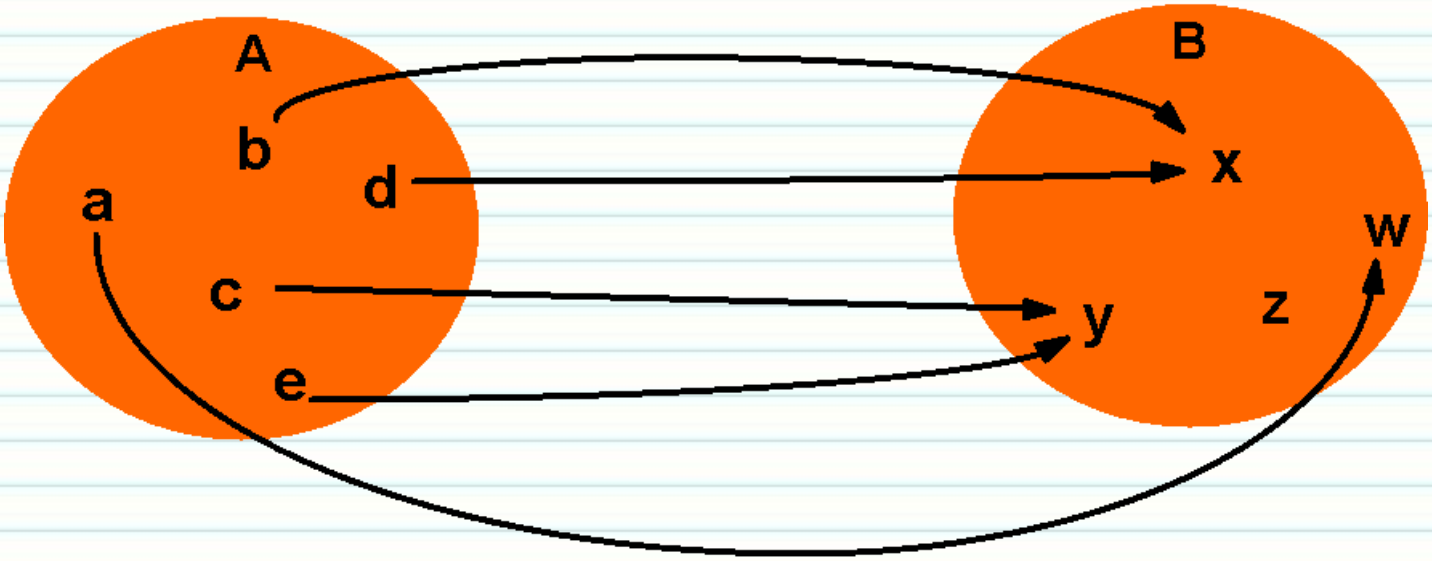
That someone else is most likely the programmer when he or she is coding the ELSE option of some IF statement. There is a very low probability that the programmer's guesses as to the expected behavior will be what the customer expects.

-- James Kowal, "Behavior Models: Specifying User's Expectations"

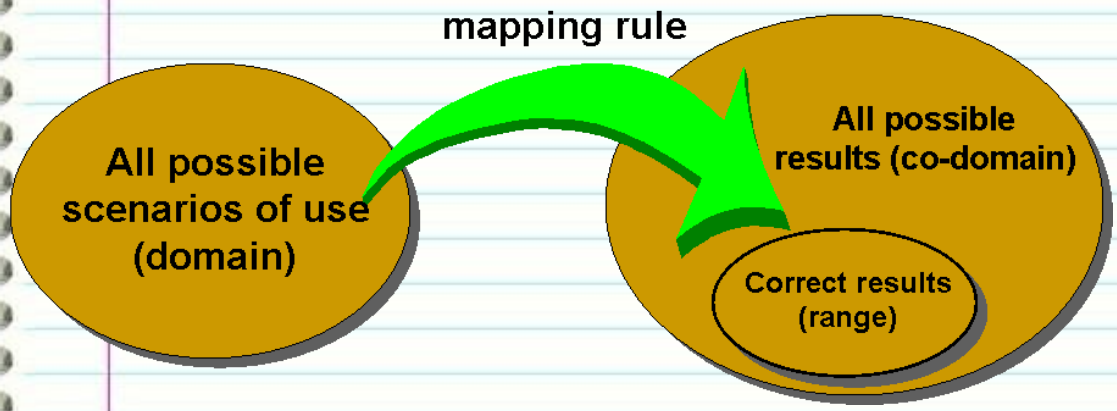
How do we ensure correct operation of software?

- Testing?
  - A sampling process
  - As complexity increases, a sufficient sample becomes more impractical
- Peer reviews?
  - Often subjective
  - Labor intensive
  - No objective criteria for sufficiency
- A better approach
  - Derive mathematically rigorous specification and design from requirements
  - Prove critical design properties using basic mathematics of software

A complete function



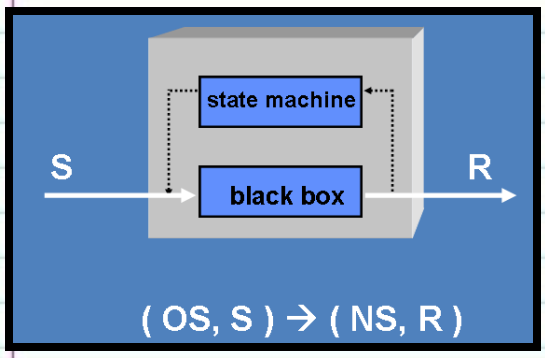
Software is a rule for a function

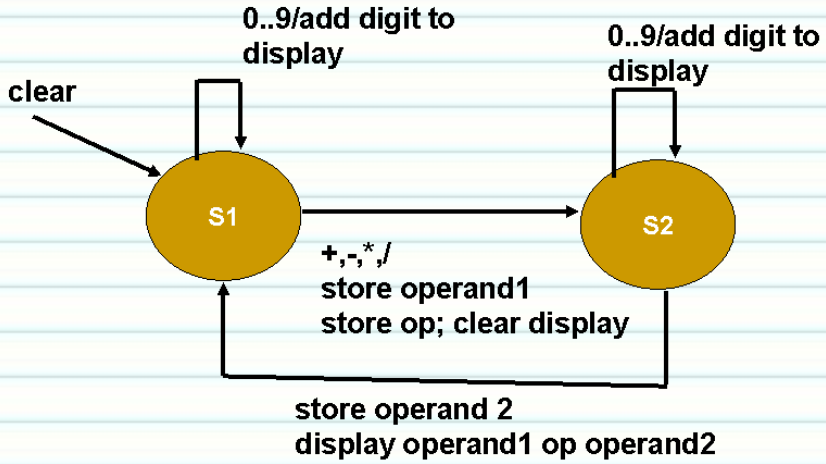


A software program is a rule for computing a mathematical function that map *all* possible stimulus histories to *all* possible responses

State view

- Defines the state space
- Encapsulate stimulus history as state data (procedure free)
- (old state, stimulus) → (new state, response)
- Preserves black box specification
- Several possible state implementations of the black box



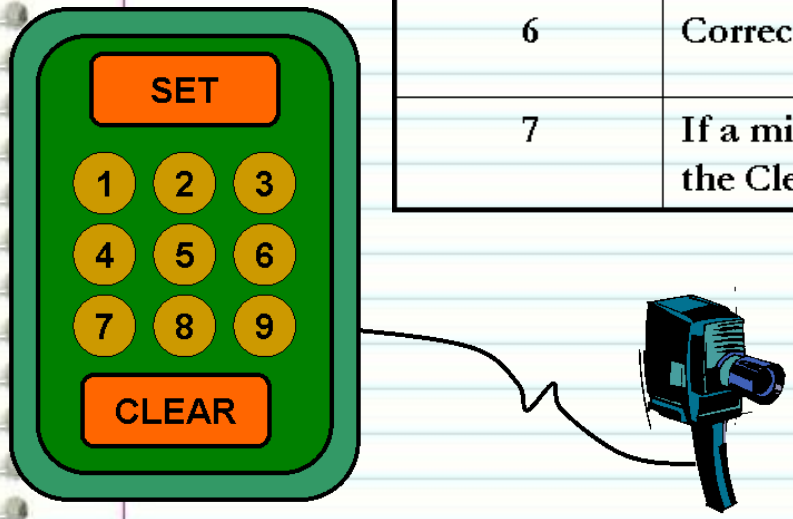


A rigorous specification process

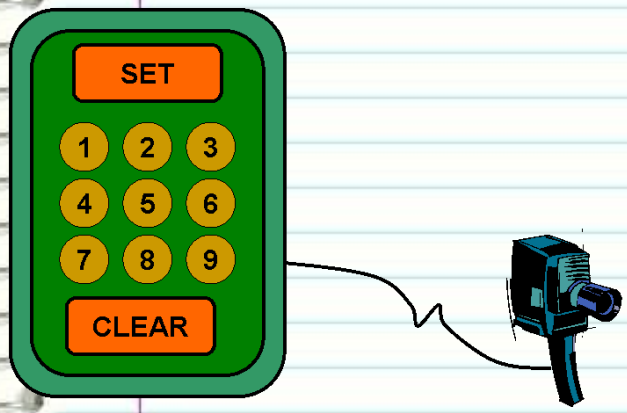
1. Establish the system boundary
2. Define the human/software/hardware interfaces
3. Itemize the stimuli and the responses
4. Perform the enumeration
5. Perform the canonical sequence analysis
6. Generate the state machine specification
7. Transform the state machine to code or another formal notation

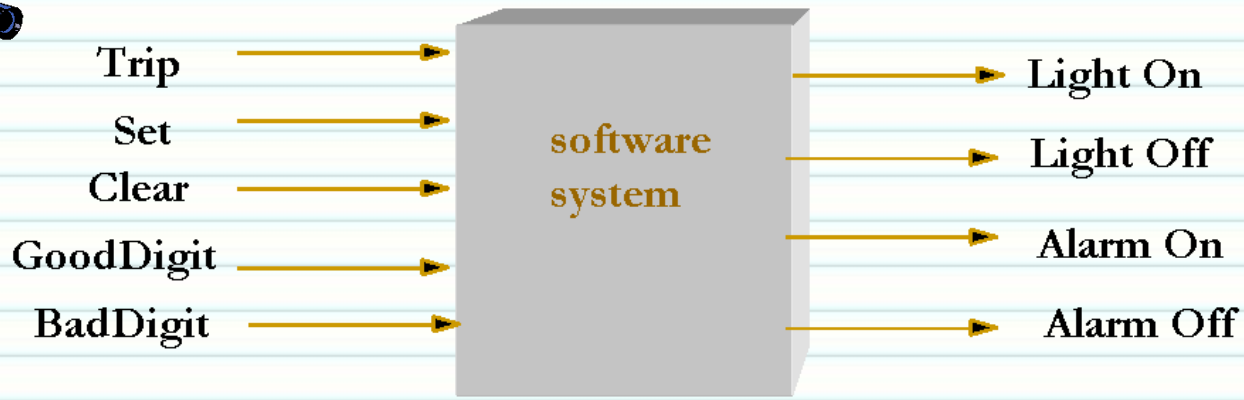
Example – Specification of a Security Alarm

Tag #	Requirements for the Security System
1	The security alarm has a detector that sends a trip signal when motion is detected.
2	The security alarm is activated by pressing the Set button.
3	The Set button is illuminated when the security alarm is set.
4	If the Trip signal occurs while the security alarm is set, a high-pitched tone (alarm) is emitted.
5	A three-digit code must be entered to turn off the alarm tone.
6	Correct entry of the code deactivates the security alarm.
7	If a mistake is made when entering the code, the user must press the Clear button before the code can be reentered.



Inputs and Outputs





- Stimulus; an event resulting in information flow from the outside to the inside of the system boundary
- Output; externally observable item of information flow from inside to outside the system boundary
- Response; occurrence of one or more outputs caused by a stimulus

Itemize Security Alarm Stimuli

Stimuli	Description	Requirements trace
Set	Device activator	2
Trip	Signal from detector	1
Clear	Clear entry	7
GoodDigit	A digit that is part of the correct entry of the three digit code that deactivates the alarm and device	5,6
BadDigit	Incorrect entry of a digit in the code	7

- Trip, Set, Clear are atomic stimuli
- GoodDigit and BadDigit are abstractions (single stimulus respresenting multiple events)
  - Represent correct and incorrect behavior of digits in a three-digit code
- Abstraction serves the purpose of hiding well understood atomic level details (whether a digit is good or bad)
  - Must be defined by disjoint predicates representing all possible stimuli

Itemize Security Alarm Responses (cont.)

Response	Description	Requirements trace
Light on	Set button illuminated	3
Light off	Set button not illuminated	6
Alarm on	High pitched sound activated	4
Alarm off	High pitched sound deactivated	5

Sequence-based Enumeration Process

- Define response to all possible stimulus sequences by considering them in order of length
- Continue until all sequences of a given length are either illegal or equivalent to previous sequences
- Identify canonical sequences
- A sequence is
  - *Equivalent* to another sequence if their responses to the same future stimuli are identical
  - *Canonical* if it carries forward to a new state

Enumeration Mechanics

- For each stimulus sequence of length n;
  - Document correct response based on requirements
  - If no requirement found, create derived requirement
  - Record requirements trace
  - Check for equivalence with previous sequences
- Extend only those sequences that are not equivalent

Sequence Enumeration – length 0 & 1

Sequence	Response	Equivalence	Requirements trace number
Length 0			
Idle	Null		D1; the security alarm is initially deactivated
Length 1			
S	Light on		2,3
T		Idle	D1
B		Idle	D1
C		Idle	D1
G		Idle	D1

Carry to next level

S	Set
T	Trip
B	Bad Digit
C	Clear
G	Good Digit

Sequence Enumeration – length 2

Sequence	Response	Equivalence	Requirements trace number
Length 2			
SS	Null	S	
ST	Alarm on		4
SB	Null	S	D2; Digit entries ignored until alarm on
SC	Null	S	
SG	Null	S	D2; Digit entries ignored until alarm on

The discovery and documentation of derived requirements is a natural and desirable part of the sequence enumeration process

Derived Requirements

- Requirements are updated based on our sequence discoveries of length 2

Tag	Requirements
D1	The security alarm is initially deactivated.
D2	After the device has been set, the Set button has no further effect until the device has been deactivated.
D3	The device produces no external response to an erroneous entry.
D4	The device produces no external response to a Clear entry.
D5	The device produces no external response to correct entry of a GoodDigit until all three digits of the code have been entered.

Sequence Enumeration – length 3

Sequence	Response	Equivalence	Requirements trace number
Length 3			
STS	Null	ST	D4; ignore set after first occurrence
STT	Null	ST	D5; ignore Trip after first occurrence
STB	Null		7
STC	Null	ST	
STG	Null		5

Derived Requirements

- Requirements are again updated based on our discoveries of length 3

Tag	Requirements
D1	The security alarm is initially deactivated.
D2	After the device has been set, the Set button has no further effect until the device has been deactivated.
D3	The device produces no external response to an erroneous entry.
D4	The device produces no external response to a Clear entry.
D5	The device produces no external response to correct entry of a GoodDigit until all three digits of the code have been entered.
D6	After the trip signal has set off the alarm, the trip signal has no further effect until the device has been deactivated.
D7	Incomplete entry of the code prior to a trip signal will be regarded as an erroneous entry that requires a Clear and a reentry of the correct code to deactivate the alarm.

Sequence Enumeration – length 4

Sequence	Response	Equivalence	Requirements trace number
Length 4			
STBS	Null	STB	D3; after bad digit entry, wait for clear
STBT	Null	STB	D3; after bad digit entry, wait for clear
STBB	Null	STB	7
STBC	Null	ST	7 - Hitting the clear button resets
STBG	Null	STB	7 - One bad digit means must reset

Sequence Enumeration – length 4

Sequence	Response	Equivalence	Requirements trace number
Length 4			
STGS	Null	STG	D4; ignore set after first occurrence
STGT	Null	STG	D5; ignore Trip after first occurrence
STGB	Null	STB	Any B digit means reset
STGC	Null	ST	Hitting the clear button resets - 7
STGG	Null		Two good digits entered

Sequence Length Five

Sequence	Response	Equivalence	Requirements trace
Length 5			
STGGS	Null	STGG	D2
STGGT	Null	STGG	D6
STGGB	Null	STB	D3
STGGC	Null	ST	D4
STGGG	Alarm off, Light off	Empty	3, 5, 6

Each entry has an equivalence –
Therefore we can stop enumerating!!!

Enumeration complete

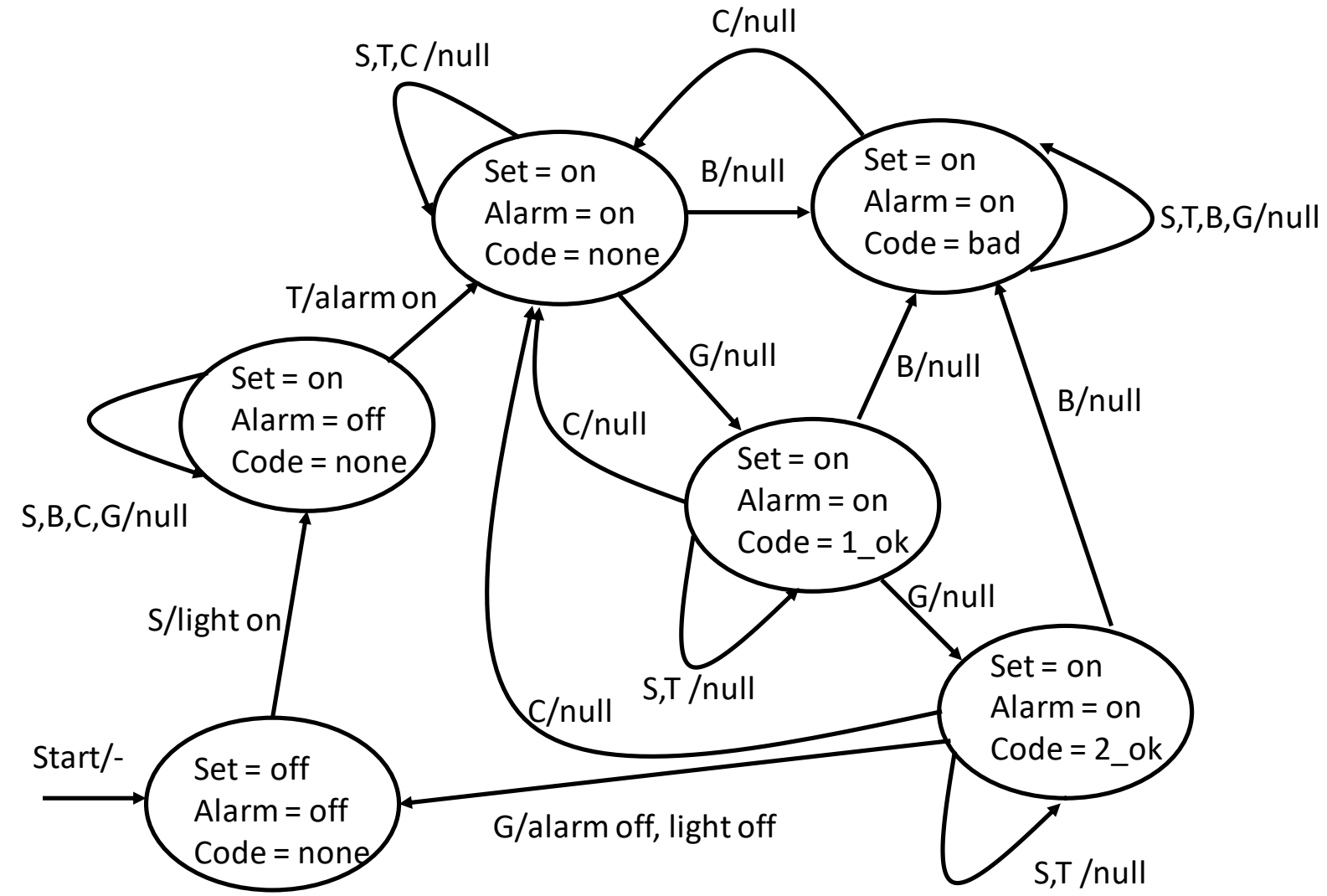
- Every scenario has been mapped to a response (complete)
- Every scenario has been mapped to only one response (set), (consistent)
- Requirements can now confirm that assumptions and derived requirements are correct
- A sequence represents a scenario of use
- Sequence enumeration reveals;
  - Possible scenarios
  - Impossible scenarios
  - Intended uses
  - Erroneous use
  - Reducible sequences
  - Irreducible sequences

Final Derived Requirements

- The Requirements have been update with the final set based on all enumerated sequences

Tag	Requirements
D1	The security alarm is initially deactivated.
D2	After the device has been set, the Set button has no further effect until the device has been deactivated.
D3	The device produces no external response to an erroneous entry.
D4	The device produces no external response to a Clear entry.
D5	The device produces no external response to correct entry of a GoodDigit until all three digits of the code have been entered.
D6	After the trip signal has set off the alarm, the trip signal has no further effect until the device has been deactivated.
D7	Incomplete entry of the code prior to a trip signal will be regarded as an erroneous entry that requires a Clear and a reentry of the correct code to deactivate the alarm.

State Transition Diagram



Specification Work Product Summary

- Requirements
- Stimulus List
- Response List
- Initial Stimulus/Condition/ Response Table
- Enumeration
- Canonical Sequence Analysis
- Black Box Specification Tables
- State Box Specification Tables

Another Example

There is a soda machine. It has a quarter slot (Q), a select soda button (S) - only one kind of soda, a return change button (R). It has the following outputs/actions: Dispense soda (D), Return change (C)), and a LED display (M). Soda's are 75 cents.

Pressing the S button causes the soda to be dispensed (D) when 75 cents credit or more is received, otherwise ignored. Pressing the (R) returns one deposited quarter (Q). D dispenses a soda, and indicates the credit is less 75 cents.

Initial state - (I) - the machine has no credit, has unlimited sodas, displays "Welcome", and is on.

Inputs

quarter slot (Q)

select soda button (S)

return change button (R)

Outputs/actions

Dispense soda (D)

Return change (C)

Message (M)

Condition	Message
No quarters	Welcome
1 quarter deposited	25 cents credit
2 quarter deposited	50 cents credit
3 quarter deposited	75 cents credit

Another Example (cont)

Rules for state diagram and sequence enumeration (here and **homework**):

- Each state MUST address all inputs.
- Inputs
  1. show inputs as a single letter
  2. only show positive (true) values for inputs (unless and interlock must be added)
  3. Inputs cannot occur simultaneously - unless stated otherwise
- Outputs
  1. Show all outputs with each sequence e.g., D=F, C=F, M="Welcome"

Rules for this soda machine

- Assume that to dispense change the output C must be set true - depict this as C=T. No dispense is C=F
- Assume to dispense soda the output D must be set true, you can depict this as D=T. No dispense is D=F
- See problem description for other behavior rules

Another Example (cont)

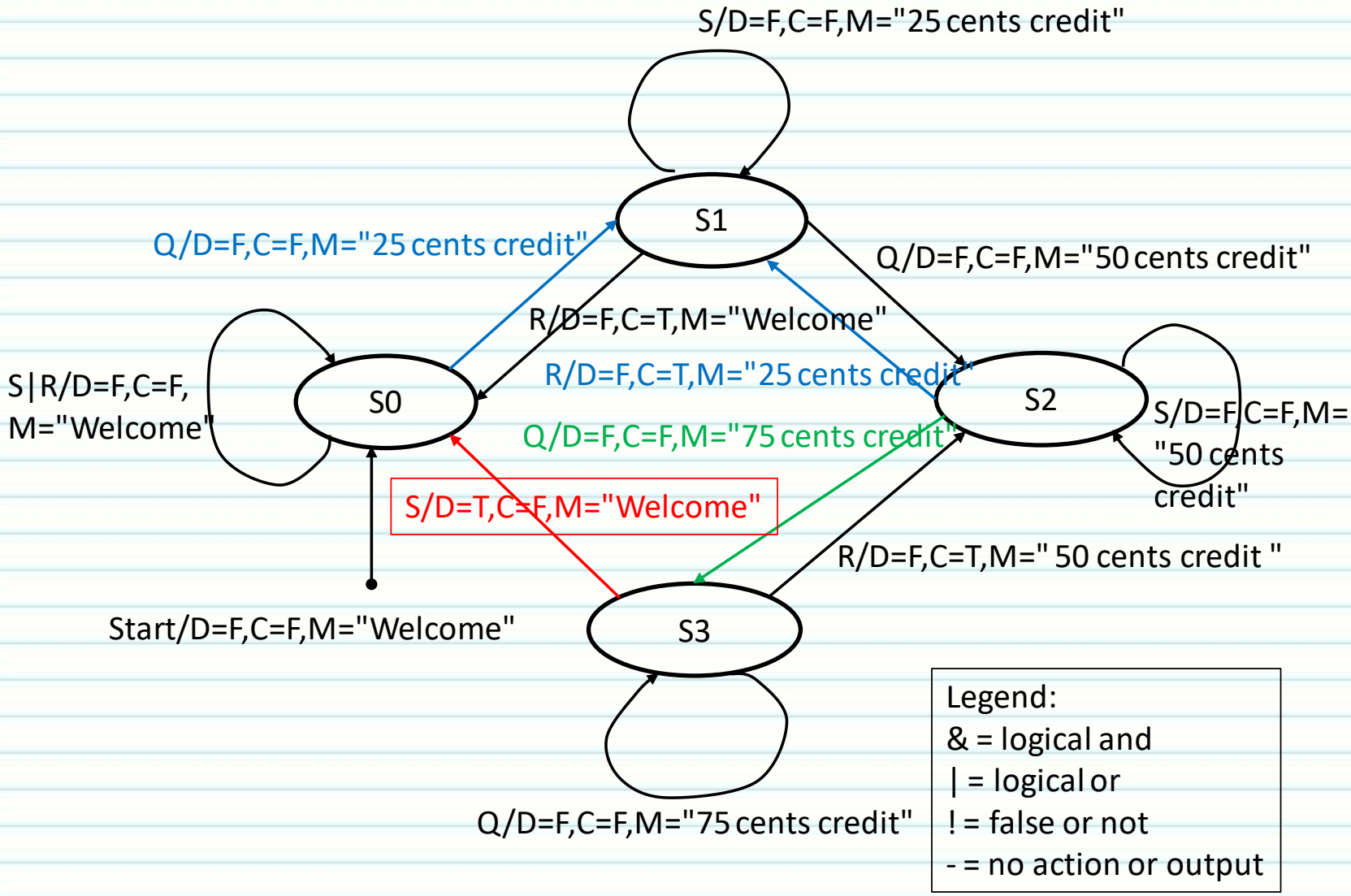
Sequence Enumeration:

Length	Sequence	Response	Equivalence	Carry to next level
0	Idle	D=F,C=F,M="Welcome"	-	No
1	Q	D=F,C=F,M="25 cents credit"	-	Yes
	S	D=F,C=F,M="Welcome"	Idle	No
	R	D=F,C=F,M="Welcome"	Idle	No
2	QQ	D=F,C=F,M="50 cents credit"	-	Yes
	QS	D=F,C=F,M="25 cents credit"	Q	No
	QR	D=F,**C=T**,M="Welcome"	Idle	No
3	QQQ	D=F,C=F,M="75 cents credit"	-	Yes
	QQS	D=F,C=F,M="50 cents credit"	QQ	No
	QQR	D=F,**C=T**,M="25 cents credit"	Q	No
4	QQQQ	D=F,C=F,M="75 cents credit"	QQQ	No
	QQQS	**D=T**,C=F,M="Welcome"	Idle	No
	QQQR	D=F,**C=T**,M="50 cents credit"	QQ	No

Canonical sequences are: Q, QQ, QQQ.
Note: QQQS is NOT a canonical sequence. Neither is Idle

From the canonical sequences we can determine the state diagram (following slide)

Another Example (cont)



Another Example (cont)

- Notice that on the previous diagram instead of showing !D we showed D=F
- This is an alternative way of showing outputs - I show you both ways for familiarity

Corresponding State Table

• This is from the state diagram but can be easily derived from the Seq Enum table

Current State	Inputs			Expected Outputs			Next State
	Q	S	R	C	D	M	
Start	-	-	-	F	F	Welcome	S0
S0	**T**	F	F	F	F	25 cents credit	S1
S0	F	**T**	F	F	F	Welcome	S0
S0	F	F	**T**	F	F	Welcome	S0
S1	**T**	F	F	F	F	50 cents credit	S2
S1	F	**T**	F	F	F	25 cents credit	S1
S1	F	F	**T**	**T**	F	Welcome	S0
S2	**T**	F	F	F	F	75 cents credit	S3
S2	F	**T**	F	F	F	50 cents credit	S2
S2	F	F	**T**	**T**	F	25 cents credit	S1
S3	**T**	F	F	F	F	75 cents credit	S3
S3	F	**T**	F	F	**T**	Welcome	S0
S3	F	F	**T**	**T**	F	50 cents credit	S2

Corresponding Test Case Table

• We are making an assumption that we can set the current state (even though it is not specified as a data input)

Test Case Number	Current State	Inputs			Expected Outputs			Next State
		Q	S	R	C	D	M	
1	Start	-	-	-	F	F	Welcome	S0
2	S0	**T**	F	F	F	F	25 cents credit	S1
3	S0	F	**T**	F	F	F	Welcome	S0
4	S0	F	F	**T**	F	F	Welcome	S0
5	S1	**T**	F	F	F	F	50 cents credit	S2
6	S1	F	**T**	F	F	F	25 cents credit	S1
7	S1	F	F	**T**	**T**	F	Welcome	S0
8	S2	**T**	F	F	F	F	75 cents credit	S3
9	S2	F	**T**	F	F	F	50 cents credit	S2
10	S2	F	F	**T**	**T**	F	25 cents credit	S1
11	S3	**T**	F	F	F	F	75 cents credit	S3
12	S3	F	**T**	F	F	**T**	Welcome	S0
13	S3	F	F	**T**	**T**	F	50 cents credit	S2