

# Developing Black Box Specifications Through Sequence Enumeration

Stacy Prowell

`sprowell@cs.utk.edu`

*University of Tennessee*

107 Ayres Hall, Knoxville, TN 37996-1301, USA

*Q-Labs, Inc.*

5516 Lonas Road, Knoxville, TN 37909, USA

1 of 18

# The Box Structure Development Method

- Developed by Mills (1985)
- Describes three views of a software system:
  - **Black Box** is a history-based specification
  - **State Box** is a state machine
  - **Clear Box** is a procedure
- Separates *behavioral* concerns from *implementation* concerns

2 of 18

# Black Box Specification

- Black box is a function
  - Domain is stimulus sequences
  - Range is software responses
- Focus on **behavior**, not implementation
- Is **complete, consistent, and correct**

Stimulus  
Sequences



$$f: S^* \rightarrow R$$



Responses

# Black Box Issues

- How can a black box specification be developed?
- How can **completeness**, **consistency**, and **correctness** be assured?
- How does one manage the level of detail when developing a black box?

These issues are addressed by **sequence enumeration** and **sequence abstraction**.

# Sequence Enumeration

- List stimulus sequences by length
- Assign a response to each stimulus sequence [consistency]
- Two special cases:
  - If there is no external response for a sequence, assign the value “null,” denoted 0
  - If the sequence is self-contradictory (impossible), assign the value “illegal,” denoted  $\omega$
- Trace each entry to requirements [correctness]

5 of 18

# Example: First Try

1. Safe uses three-digit combination.
2. Clear key must be pressed after each incorrect entry.
3. After entering combination, safe unlocks.
4. Safe locks when door closed.

Seq.	Resp.	Trace
$\lambda$	0	Method
c	0	(5*) (6*)
D	Lock safe	(4) (5)
d	0	(5) (6)
d <sub>1</sub>	0	(5) (6)
d <sub>2</sub>	0	(5) (6)
d <sub>3</sub>	0	(5) (6)
cc	0	(5) (6)
cD	Lock safe	(4) (5)
...	...	...
Dc	0	(7*)
DD	$\omega$	(5) (8*)

# Sequence Equivalences

The sequences  $D$  and  $cD$  both leave the safe locked and ready for combination entry.

Two sequences are **equivalent** if and only if the black box function gives the same response for extensions of the sequences.

$$u \equiv v \Leftrightarrow \forall w \in S^+, \text{BB}(uw) = \text{BB}(vw)$$

Responses are always the same for extensions; no need to extend both.

# Example: Second Try

Seq	Resp	Equiv
$\lambda$	0	
c	0	$\lambda$
D	Lock safe	
d	0	$\lambda$
d <sub>1</sub>	0	$\lambda$
d <sub>2</sub>	0	$\lambda$
d <sub>3</sub>	0	$\lambda$
Dc	0	D
DD	$\omega$	
Dd	0	
Dd <sub>1</sub>	0	
Dd <sub>2</sub>	0	Dd
Dd <sub>3</sub>	0	Dd
Ddc	0	D

Seq	Resp	Equiv
DdD	$\omega$	
Ddd	0	Dd
Ddd <sub>1</sub>	0	Dd
Ddd <sub>2</sub>	0	Dd
Ddd <sub>3</sub>	0	Dd
...	...	...
Dd <sub>1</sub> d <sub>2</sub> c	0	D
Dd <sub>1</sub> d <sub>2</sub> D	$\omega$	
Dd <sub>1</sub> d <sub>2</sub> d	0	Dd
Dd <sub>1</sub> d <sub>2</sub> d <sub>1</sub>	0	Dd
Dd <sub>1</sub> d <sub>2</sub> d <sub>2</sub>	0	Dd
Dd <sub>1</sub> d <sub>2</sub> d <sub>3</sub>	Release	$\lambda$

8 of 18

# Complete Enumeration

- A complete enumeration has no sequences left to extend
  - It gives the unique response (and requirements trace) for every sequence
  - It is thus the:
    - complete
    - consistent
    - traceably correct
- black box function

9 of 18

# Black Box Issues

- How does one manage the level of detail when developing a black box?

# Sequence Abstraction

Sequence enumeration can be focused and controlled through explicit **sequence abstraction**.

Sequence abstractions:

- are an alternate view of the stimulus sequence
- are used to hide or amplify details
- allow developers to work at a more productive level

# Example

Let:

C denote  $d_1d_2d_3$

E denote  $d + d_2 + d_3$   
 $+ d_1(d + d_1 + d_3)$   
 $+ d_1d_2(d + d_1 + d_2)$

Then sequences can be viewed abstractly as:

$Dd_1d_2d_3$  is DC

$Dd$  is DE

$Dd_1d_2d$  is DE

12 of 18

# Sequence Abstraction

A sequence abstraction is a many-to-one mapping from “atomic” sequences to “abstract” sequences, such that:

- Abstract sequences are no longer than atomic sequences [**non-increasing**]
- Abstractions preserve the historical order of events [**stimulus ordering**]
- All abstract stimuli used at once are **disjoint**

13 of 18

# Example: Third Try

Seq	Resp	Equiv
$\lambda$	0	
c	0	$\lambda$
D	Lock safe	
C	0	$\lambda$
E	0	$\lambda$
Dc	0	D
DD	$\omega$	
DC	Release	$\lambda$
DE	0	
DEc	0	D
DED	$\omega$	
DEC	0	DE
DEE	0	DE

The abstract enumeration:

- Hides digit details
- Reduces the work
- Makes a natural abstraction explicit
- Is still complete, consistent, and traceably correct

# Removing Abstractions

Abstractions are functions of the form:

$$X^* \rightarrow Y^*$$

They can be composed with the black box function to obtain a black box at a lower level of abstraction.

# Results

- Sequence enumeration is a way to develop a black box specification which is
  - complete
  - consistent
  - traceably correct
- Abstractions can be used to focus and control the enumeration process
- Abstractions can be removed, if desired

16 of 18

# Results

In practice:

- interesting behavior is revealed for short sequences
- substantial missed behavior is discovered
- sequences do not get very long

17 of 18

# Future

- Improved automation support
- Applications to embedded systems
- Support for automated testing