Safe Controller – An Application of Sequence-Based Specification

Lan Lin, Jesse H. Poore, Stacy J. Prowell

Department of Computer Science
University of Tennessee
Knoxville, TN 37996

ut-cs-05-555

June 2005

Contents

List of Tables

List of Figures

1 Requirements

The safe controller as presented in [1] is a simple application of sequence-based specification techniques. Requirements for the simple safe controller are given in Table 1.

Tag	Requirement
1	The combination consists of three digits (0-9) which must be entered in the correct order to unlock the safe. The combination is fixed in the safe firmware.
2	Following an incorrect combination entry, a "clear" key must be pressed before the safe will accept further entry. The clear key also resets any combination entry.
3	Once the three digits of the combination are entered in the correct order, the safe unlocks and the door may be opened.
4	When the door is closed, the safe automatically locks.
5	The safe has a sensor which reports the status of the lock.
6	The safe ignores keypad entry when the door is open.
7	There is no external confirmation for combination entry other than unlocking the door.
8	It is assumed (with risk) that the safe cannot be opened by means other than combination entry while the software is running.
D1	Sequences with stimuli prior to system initialization are illegal by system definition.
D2	Re-initialization (power-on) makes previous history irrelevant.

Table 1: Safe Controller Requirements

2 Enumeration

2.1 System Boundary, Atomic Stimuli, and Responses

The system boundary cuts the interfaces between the system and the external power, keypad, door sensor, and lock actuator. Atomic stimuli and responses can be identified as in Table 2 and Table 3.

Stimulus	Description	Interface
0-9	Digit press	keypad
C	Clear key press	keypad
D	Door closed	door sensor
L	Power on with door locked	power, door sensor
U	Power on with door unlocked	power, door sensor

Table 2: Safe Controller Stimuli

Response	Description	Interface
lock	Locking the door	lock actuator
unlock	Unlocking the door	lock actuator

Table 3: Safe Controller Responses

2.2 Abstraction 1

Atomic stimulus set: $X = \{0, 1, 2, \ldots, 9, C, D, L, U\}$
Abstract stimulus set: $Y_1 = \{d, C, D, L, U\}$
Abstraction in ANF: $\phi : X^* \to Y_1^*$

$$\phi(\lambda) = \lambda$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$\phi(ux) = \begin{cases} \phi(u)d & \text{if } p_d(ux) \\ \phi(u)C & \text{if } p_C(ux) \\ \phi(u)D & \text{if } p_D(ux) \\ \phi(u)L & \text{if } p_L(ux) \\ \phi(u)U & \text{if } p_U(ux) \\ \phi(u) & \text{otherwise} \end{cases}$$

Characteristic predicates $p_i : X^* \to \{\text{true, false}\},\ \forall i \in Y_1$

$$p_d(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_d(ux) = \begin{cases} \text{true} & \text{if } x \in \{0, 1, \ldots, 9\} \\ \text{false} & \text{otherwise} \end{cases}$$

$$p_C(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_C(ux) = \begin{cases} \text{true} & \text{if } x = C \\ \text{false} & \text{otherwise} \end{cases}$$

$$p_D(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_D(ux) = \begin{cases} \text{true} & \text{if } x = D \\ \text{false} & \text{otherwise} \end{cases}$$

$$p_L(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_L(ux) = \begin{cases} \text{true} & \text{if } x = L \\ \text{false} & \text{otherwise} \end{cases}$$

$$p_U(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_U(ux) = \begin{cases} \text{true} & \text{if } x = U \\ \text{false} & \text{otherwise} \end{cases}$$

Predicate refinement is needed to enumerate under Abstraction 1, when letting d denote any digit press results in a slight overabstraction. Predicate p is introduced for this purpose. Let $c_1c_2c_3$ be the correct combination entry.

$$p : X^* \rightarrow \{\text{ true, false}\}$$
$$\forall u \in X^*,\ p(u) = \text{true} \Leftrightarrow \exists v \in X^*,\ u = vc_1c_2c_3.$$

The complete enumeration is produced as in Table 4.

Sequence	Response	Equivalence	Trace
λ	0		Method
d	ω		D1
C	ω		D1
D	ω		D1
L	0		5
U	0		5
Ld	0		7
LC	0	L	2,7
LD	ω		8
LL	0	L	5,D2
LU	0	U	5,D2
Ud	0	U	6
UC	0	U	6
UD	lock	L	4
UL	0	L	5,D2
UU	0	U	5,D2
Ldd	0		7
LdC	0	L	2,7
LdD	ω		8
LdL	0	L	5,D2
LdU	0	U	5,D2
$Ldd[d, p]$	unlock	U	1,3,7
$Ldd[d, \neg p]$	0		1,2,7
$LddC$	0	L	2,7
$LddD$	ω		8
$LddL$	0	L	5,D2
$LddU$	0	U	5,D2
$Ldd[d, \neg p]d$	0	$Ldd[d, \neg p]$	2,7
$Ldd[d, \neg p]C$	0	L	2,7
$Ldd[d, \neg p]D$	ω		8
$Ldd[d, \neg p]L$	0	L	5,D2
$Ldd[d, \neg p]U$	0	U	5,D2

Table 4: Sequence Enumeration under Abstraction 1

2.3 Abstraction 2

Atomic stimulus set: $X = \{0, 1, 2, \ldots, 9, C, D, L, U\}$
Abstract stimulus set: $Y_2 = \{G, B, C, D, L, U\}$
Abstraction in ANF: $\phi : X^* \to Y_2^*$

$$\phi(\lambda) = \lambda$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$\phi(ux) = \begin{cases} \phi(u)G & \text{if } p_G(ux) \\ \phi(u)B & \text{if } p_B(ux) \\ \phi(u)C & \text{if } p_C(ux) \\ \phi(u)D & \text{if } p_D(ux) \\ \phi(u)L & \text{if } p_L(ux) \\ \phi(u)U & \text{if } p_U(ux) \\ \phi(u) & \text{otherwise} \end{cases}$$

Characteristic predicates $p_i : X^* \to \{\text{true}, \text{false}\},\ \forall i \in \{C, D, L, U\}$

$$p_i(\lambda) = \text{false}$$
$$\forall u \in X^*,\ \forall x \in X,$$
$$p_i(ux) = \begin{cases} \text{true} & \text{if } x = i \\ \text{false} & \text{otherwise} \end{cases}$$

Informal definitions for p_G and p_B can be used to obtain a complete enumeration. Let $p_G = true$ denote entering the correct combination entry in order, and let $p_B = true$ denote entering the combination incorrectly. Their formal definitions can be derived easily with a complete enumeration in hand. The complete enumeration under Abstraction 2 is constructed in Table 5.

To get formal definitions for characteristic predicates p_G and p_B, we need complete enumerations for them in X^*. To avoid inefficient work, we introduce the abstraction of d to represent any digit press. Three predicates are defined as follows, which will be used to define p_G and p_B:

Let $c_1c_2c_3$ be the correct combination entry.
p_{c_1}: "The current digit is the correct 1st digit c_1."
p_{c_2}: "The current digit is the correct 2nd digit c_2."
p_{c_3}: "The current digit is the correct 3rd digit c_3."
$p_{c_i} : X^* \to \{\text{true}, \text{false}\},\ \forall i \in \{1, 2, 3\}$

Sequence	Response	Equivalence	Trace
λ	0		Method
G	ω		D1
B	ω		D1
C	ω		D1
D	ω		D1
L	0		5
U	0		5
LG	unlock	U	1,3,7
LB	0		1,2,7
LC	0	L	2,7
LD	ω		8
LL	0	L	5,D2
LU	0	U	5,D2
UG	0	U	6
UB	0	U	6
UC	0	U	6
UD	lock	L	4
UL	0	L	5,D2
UU	0	U	5,D2
LBG	0	LB	2,7
LBB	0	LB	2,7
LBC	0	L	2,7
LBD	ω		8
LBL	0	L	5,D2
LBU	0	U	5,D2

Table 5: Sequence Enumeration under Abstraction 2

$$
\begin{array}{l}
p_{c_i}(\lambda) = \text{false} \\
\forall u \in X^*,\ \forall x \in X, \\
p_{c_i}(ux) = \begin{cases} \text{true} & \text{if } x = c_i \\ \text{false} & \text{otherwise} \end{cases}
\end{array}
$$

Enumerations for p_G and p_B are shown in Table 6. It happens that the sequence column and the equivalence column are the same for two enumerations; thus we combine two tables into one.

Sequence	Resp. for p_G	Resp. for p_B	Equiv.
λ	false	false	
$[d, p_{c_1}]$	false	false	
$[d, \neg p_{c_1}]$	false	true	
C	false	false	λ
D	false	false	λ
L	false	false	λ
U	false	false	λ
$[d, p_{c_1}][d, p_{c_2}]$	false	false	
$[d, p_{c_1}][d, \neg p_{c_2}]$	false	true	$[d, \neg p_{c_1}]$
$[d, p_{c_1}]C$	false	false	λ
$[d, p_{c_1}]D$	false	false	λ
$[d, p_{c_1}]L$	false	false	λ
$[d, p_{c_1}]U$	false	false	λ
$[d, \neg p_{c_1}]d$	false	true	$[d, \neg p_{c_1}]$
$[d, \neg p_{c_1}]C$	false	false	λ
$[d, \neg p_{c_1}]D$	false	false	λ
$[d, \neg p_{c_1}]L$	false	false	λ
$[d, \neg p_{c_1}]U$	false	false	λ
$[d, p_{c_1}][d, p_{c_2}][d, p_{c_3}]$	true	false	λ
$[d, p_{c_1}][d, p_{c_2}][d, \neg p_{c_3}]$	false	true	$[d, \neg p_{c_1}]$
$[d, p_{c_1}][d, p_{c_2}]C$	false	false	λ
$[d, p_{c_1}][d, p_{c_2}]D$	false	false	λ
$[d, p_{c_1}][d, p_{c_2}]L$	false	false	λ
$[d, p_{c_1}][d, p_{c_2}]U$	false	false	λ

Table 6: Enumerations for p_G and p_B

Canonical sequence analysis for enumerations in Table 6 is performed to derive formal definitions for p_G and p_B. The analysis is given in Table 7.

Canonical Sequence	Combo
λ	0
$[d, p_{c_1}]$	1
$[d, \neg p_{c_1}]$	x
$[d, p_{c_1}][d, p_{c_2}]$	2

Table 7: Canonical Sequence Analysis for Enumerations for p_G and p_B

Specification function $Combo : X^* \rightarrow \{0,\ 1,\ x,\ 2\}$

$$
\begin{aligned}
&Combo(\lambda) = 0\\
&\forall u \in X^*,\ \forall x \in X,\\
&Combo(ux) = \begin{cases}
1 & \text{if } Combo(u) = 0 \wedge x = c_1\\
x & \text{if } Combo(u) = 0 \wedge x \neq c_1 \wedge x \in \{0,\ 1,\ \ldots,\ 9\}\\
2 & \text{if } Combo(u) = 1 \wedge x = c_2\\
x & \text{if } Combo(u) = 1 \wedge x \neq c_2 \wedge x \in \{0,\ 1,\ \ldots,\ 9\}\\
0 & \text{if } Combo(u) = 1 \wedge x \in \{C,\ D,\ L,\ U\}\\
0 & \text{if } Combo(u) = x \wedge x \in \{C,\ D,\ L,\ U\}\\
0 & \text{if } Combo(u) = 2 \wedge x = c_3\\
x & \text{if } Combo(u) = 2 \wedge x \neq c_3 \wedge x \in \{0,\ 1,\ \ldots,\ 9\}\\
0 & \text{if } Combo(u) = 2 \wedge x \in \{C,\ D,\ L,\ U\}\\
Combo(u) & \text{otherwise}
\end{cases}
\end{aligned}
$$

Characteristic predicate $p_G : X^* \rightarrow \{\text{true}\ ,\ \text{false}\}$

$$
\begin{aligned}
&p_G(\lambda) = \text{false}\\
&\forall u \in X^*,\ \forall x \in X,\\
&p_G(ux) = \begin{cases}
\text{false} & \text{if } Combo(u) = 0\\
\text{false} & \text{if } Combo(u) = 1\\
\text{false} & \text{if } Combo(u) = x\\
\text{true} & \text{if } Combo(u) = 2 \wedge x = c_3\\
\text{false} & \text{if } Combo(u) = 2 \wedge x \neq c_3 \wedge x \in \{0,\ 1,\ \ldots,\ 9\}
\end{cases}
\end{aligned}
$$

Characteristic predicate $p_B : X^* \rightarrow \{\text{true}\ ,\ \text{false}\}$

$p_B(\lambda) = \text{false}$
$\forall u \in X^*,\ \forall x \in X,$

$$p_B(ux) = \begin{cases} \text{false} & \text{if } Combo(u) = 0 \wedge x \in \{C,\ D,\ L,\ U,\ c_1\} \\ \text{true} & \text{if } Combo(u) = 0 \wedge x \neq c_1 \wedge x \in \{0,\ 1,\ \ldots,\ 9\} \\ \text{false} & \text{if } Combo(u) = 1 \wedge x \in \{C,\ D,\ L,\ U,\ c_2\} \\ \text{true} & \text{if } Combo(u) = 1 \wedge x \neq c_2 \wedge x \in \{0,\ 1,\ \ldots,\ 9\} \\ \text{false} & \text{if } Combo(u) = x \wedge x \in \{C,\ D,\ L,\ U\} \\ \text{true} & \text{if } Combo(u) = x \wedge x \in \{0,\ 1,\ \ldots,\ 9\} \\ \text{false} & \text{if } Combo(u) = 2 \wedge x \in \{C,\ D,\ L,\ U,\ c_3\} \\ \text{true} & \text{if } Combo(u) = 2 \wedge x \neq c_3 \wedge x \in \{0,\ 1,\ \ldots,\ 9\} \end{cases}$$

3 State Machine

For the following sections, we only consider representations of the safe controller module equivalent to the enumeration under Abstraction 2.

> $M = \langle Q, X, \delta, q_\lambda, R, \nu, \phi \rangle$ is a Mealy machine for the module, where
>
> $Q = \{q_\lambda, q_L, q_U, q_{LB}, q_\omega\}$,
>
> $X = \{G, B, C, D, L, U\}$,
>
> $R = \{0, \omega, lock, unlock\}$,
>
> $\delta : Q \times X \to Q$ and $\nu : Q \times X \to R$ are total functions defined as follows:

$$\begin{array}{ll} \delta(q_\lambda, G) = q_\omega & \nu(q_\lambda, G) = \omega \\ \delta(q_\lambda, B) = q_\omega & \nu(q_\lambda, B) = \omega \\ \delta(q_\lambda, C) = q_\omega & \nu(q_\lambda, C) = \omega \\ \delta(q_\lambda, D) = q_\omega & \nu(q_\lambda, D) = \omega \\ \delta(q_\lambda, L) = q_L & \nu(q_\lambda, L) = 0 \\ \delta(q_\lambda, U) = q_U & \nu(q_\lambda, U) = 0 \\ \delta(q_L, G) = q_U & \nu(q_L, G) = unlock \\ \delta(q_L, B) = q_{LB} & \nu(q_L, B) = 0 \\ \delta(q_L, C) = q_L & \nu(q_L, C) = 0 \\ \delta(q_L, D) = q_\omega & \nu(q_L, D) = \omega \\ \delta(q_L, L) = q_L & \nu(q_L, L) = 0 \\ \delta(q_L, U) = q_U & \nu(q_L, U) = 0 \end{array}$$

$$\begin{array}{ll}
\delta(q_U, G) = q_U & \nu(q_U, G) = 0 \\
\delta(q_U, B) = q_U & \nu(q_U, B) = 0 \\
\delta(q_U, C) = q_U & \nu(q_U, C) = 0 \\
\delta(q_U, D) = q_L & \nu(q_U, D) = lock \\
\delta(q_U, L) = q_L & \nu(q_U, L) = 0 \\
\delta(q_U, U) = q_U & \nu(q_U, U) = 0 \\
\delta(q_{LB}, G) = q_{LB} & \nu(q_{LB}, G) = 0 \\
\delta(q_{LB}, B) = q_{LB} & \nu(q_{LB}, B) = 0 \\
\delta(q_{LB}, C) = q_L & \nu(q_{LB}, C) = 0 \\
\delta(q_{LB}, D) = q_\omega & \nu(q_{LB}, D) = \omega \\
\delta(q_{LB}, L) = q_L & \nu(q_{LB}, L) = 0 \\
\delta(q_{LB}, U) = q_U & \nu(q_{LB}, U) = 0
\end{array}$$

The state machine diagram is drawn in Figure 3.1.

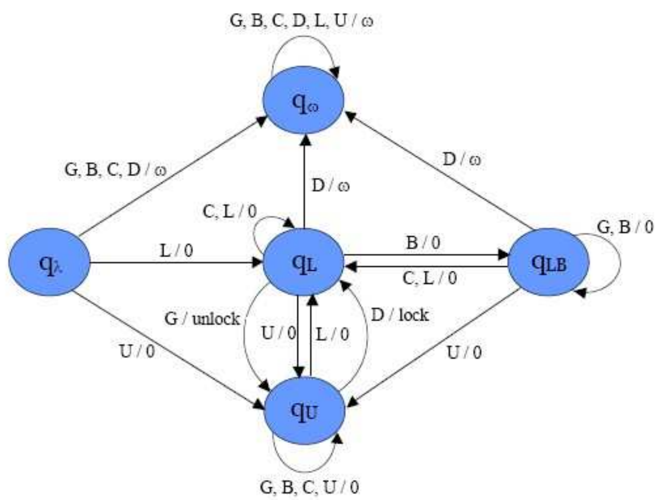

Figure 3.1: A State Machine for the Safe Controller under Abstraction 2

4 Regular Expression

Regular expressions for each legal equivalence class:

$$\begin{cases} r_\lambda & = & \lambda \\ r_L & = & (L+U(G+B+C+U)^*(D+L)) \\ & & (C+L+B(G+B)^*(C+L)+ \\ & & (G+U)(G+B+C+U)^*(D+L)+ \\ & & B(G+B)^*U(G+B+C+U)^*(D+L))^* \\ r_U & = & (U+ \\ & & (L+U(G+B+C+U)^*(D+L)) \\ & & (C+L+B(G+B)^*(C+L)+ \\ & & (G+U)(G+B+C+U)^*(D+L)+ \\ & & B(G+B)^*U(G+B+C+U)^*(D+L))^*(G+U)+ \\ & & (L+U(G+B+C+U)^*(D+L)) \\ & & (C+L+B(G+B)^*(C+L)+ \\ & & (G+U)(G+B+C+U)^*(D+L)+ \\ & & B(G+B)^*U(G+B+C+U)^*(D+L))^*B(G+B)^*U) \\ & & (G+B+C+U)^* \\ r_{LB} & = & (L+U(G+B+C+U)^*(D+L)) \\ & & (C+L+B(G+B)^*(C+L)+ \\ & & (G+U)(G+B+C+U)^*(D+L)+ \\ & & B(G+B)^*U(G+B+C+U)^*(D+L))^*B(G+B)^* \end{cases}$$

Regular expressions for each response:

$$\begin{cases} r_0 & = & \lambda + r_\lambda(L+U) + r_L(B+C+L+U)+ \\ & & r_U(G+B+C+L+U) + r_{LB}(G+B+C+L+U) \\ r_\omega & = & (r_\lambda(G+B+C+D) + r_L D + r_{LB} D) \\ & & (G+B+C+D+L+U)^* \\ r_{lock} & = & r_U D \\ r_{unlock} & = & r_L G \end{cases}$$

The second set is expressed in terms of the first for brevity.

5 Prefix-recursive Function

5.1 Canonical Sequence Analysis

Specification functions with one possible set of values for each are given in Table 8 for the enumeration in Table 5.

Canonical Sequence	Door	Error
λ	unknown	
L	lock	false
U	unlock	
LB	lock	true

Table 8: Sequence Analysis for Enumeration under Abstraction 2

5.2 Specification Function

Specification function $Door : Y_2^* \to \{unknown, lock, unlock\}$

$$Door(\lambda) = unknown$$
$$\forall u \in Y_2^*,\ \forall x \in Y_2,$$
$$Door(ux) = \begin{cases} lock & \text{if } Door(u) = unknown \wedge x = L \\ unlock & \text{if } Door(u) = unknown \wedge x = U \\ unlock & \text{if } Door(u) = lock \wedge x = G \\ unlock & \text{if } Door(u) = lock \wedge x = U \\ lock & \text{if } Door(u) = unlock \wedge x = D \\ lock & \text{if } Door(u) = unlock \wedge x = L \\ Door(u) & \text{otherwise} \end{cases}$$

Specification function $Error : Y_2^* \to \{true, false\}$

$$Error(\lambda) = any$$
$$\forall u \in Y_2^*,\ \forall x \in Y_2,$$
$$Error(ux) = \begin{cases} false & \text{if } Error(u) = any \wedge x = L \\ any & \text{if } Error(u) = false \wedge x = G \\ true & \text{if } Error(u) = false \wedge x = B \\ any & \text{if } Error(u) = false \wedge x = U \\ false & \text{if } Error(u) = any \wedge x = D \\ false & \text{if } Error(u) = true \wedge x = C \\ false & \text{if } Error(u) = true \wedge x = L \\ any & \text{if } Error(u) = true \wedge x = U \\ Error(u) & \text{otherwise} \end{cases}$$

Note that when $Error(u)$ takes the value “any”, it can be either true or false.

5.3 Abstract Black Box

Abstract black box $BB_{Y_2} : Y_2^* \to R$

$$
\begin{array}{l}
BB_{Y_2}(\lambda) = 0 \\
\forall u \in Y_2^*,\ \forall x \in Y_2, \\
BB_{Y_2}(ux) = \begin{cases}
\omega & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = unknown \wedge \\
 & \qquad Error(u) = any \wedge x \in \{G, B, C, D\} \\
0 & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = unknown \wedge \\
 & \qquad Error(u) = any \wedge x \in \{L, U\} \\
unlock & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = lock \wedge \\
 & \qquad Error(u) = false \wedge x = G \\
0 & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = lock \wedge \\
 & \qquad Error(u) = false \wedge x \in \{B, C, L, U\} \\
\omega & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = lock \wedge \\
 & \qquad Error(u) = false \wedge x = D \\
0 & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = unlock \wedge \\
 & \qquad Error(u) = any \wedge x \in \{G, B, C, L, U\} \\
lock & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = unlock \wedge \\
 & \qquad Error(u) = any \wedge x = D \\
0 & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = lock \wedge \\
 & \qquad Error(u) = true \wedge x \in \{G, B, C, L, U\} \\
\omega & \text{if} \quad BB_{Y_2}(u) \neq \omega \wedge Door(u) = lock \wedge \\
 & \qquad Error(u) = true \wedge x = D \\
\omega & \text{if} \quad BB_{Y_2}(u) = \omega
\end{cases}
\end{array}
$$

5.4 Atomic Black Box

Given formal definitions of the abstract black box (in prefix-recursive form) and the abstraction function (in ANF), we can obtain the atomic black box by function composition. Here we still take Abstraction 2 as an example.

Abstract black box: $BB_{Y_2} : Y_2^* \to R$
Abstraction: $\phi : X^* \to Y_2^*$
Atomic black box: $BB_X : X^* \to R$
Relationship: $BB_X = BB_{Y_2} \circ \phi$

The abstract black box is formally defined from the complete enumeration, as shown in Section 5.3. For brevity we are omitting some of the details here and simply use different predicates to denote the conditions under which different response values are implied. Note that these predicates partition the whole abstract space.

Abstract stimulus set: $Y_2 = \{G,\, B,\, C,\, D,\, L,\, U\}$
Response set: $R = \{lock,\, unlock,\, 0,\, \omega\}$
Abstract black box: $BB_{Y_2} : Y_2^* \rightarrow R$

$$
\begin{aligned}
& BB_{Y_2}(\lambda) = 0 \\
& \forall v \in Y_2^*,\ \forall y \in Y_2, \\
& BB_{Y_2}(vy) = \begin{cases} lock & \text{if } p_{lock}(vy) \\ unlock & \text{if } p_{unlock}(vy) \\ 0 & \text{if } p_0(vy) \\ \omega & \text{if } p_\omega(vy) \end{cases}
\end{aligned}
$$

The abstraction function is defined as in Section 2.3. Note that the characteristic predicates for all abstract stimuli do not partition the whole atomic space, thus we have the "otherwise" case.

Atomic stimulus set: $X = \{0,\, 1,\, 2,\, \ldots,\, 9,\, C,\, D,\, L,\, U\}$
Abstract stimulus set: $Y_2 = \{G,\, B,\, C,\, D,\, L,\, U\}$
Abstraction in ANF: $\phi : X^* \rightarrow Y_2^*$

$$
\begin{aligned}
& \phi(\lambda) = \lambda \\
& \forall u \in X^*,\ \forall x \in X, \\
& \phi(ux) = \begin{cases} \phi(u)G & \text{if } p_G(ux) \\ \phi(u)B & \text{if } p_B(ux) \\ \phi(u)C & \text{if } p_C(ux) \\ \phi(u)D & \text{if } p_D(ux) \\ \phi(u)L & \text{if } p_L(ux) \\ \phi(u)U & \text{if } p_U(ux) \\ \phi(u) & \text{otherwise} \end{cases}
\end{aligned}
$$

By function composition, we have the atomic black box defined as follows. Note that when none of the characteristic predicate is satisfied on an atomic sequence, we are losing some information in the abstraction, thus are losing some information in the atomic black box function.

Atomic black box: $BB_X : X^* \to R$
$BB_X = BB_{Y_2} \circ \phi$

$$
\begin{aligned}
&BB_X(\lambda) = 0 \\
&\forall u \in X^*,\ \forall x \in X, \\
&BB_X(ux) = \begin{cases}
lock & \text{if} \quad \begin{aligned}[t] & p_G(ux) \wedge p_{lock}(\phi(u)G) \\ & \vee p_B(ux) \wedge p_{lock}(\phi(u)B) \\ & \vee p_C(ux) \wedge p_{lock}(\phi(u)C) \\ & \vee p_D(ux) \wedge p_{lock}(\phi(u)D) \\ & \vee p_L(ux) \wedge p_{lock}(\phi(u)L) \\ & \vee p_U(ux) \wedge p_{lock}(\phi(u)U) \end{aligned} \\
unlock & \text{if} \quad \begin{aligned}[t] & p_G(ux) \wedge p_{unlock}(\phi(u)G) \\ & \vee p_B(ux) \wedge p_{unlock}(\phi(u)B) \\ & \vee p_C(ux) \wedge p_{unlock}(\phi(u)C) \\ & \vee p_D(ux) \wedge p_{unlock}(\phi(u)D) \\ & \vee p_L(ux) \wedge p_{unlock}(\phi(u)L) \\ & \vee p_U(ux) \wedge p_{unlock}(\phi(u)U) \end{aligned} \\
0 & \text{if} \quad \begin{aligned}[t] & p_G(ux) \wedge p_0(\phi(u)G) \\ & \vee p_B(ux) \wedge p_0(\phi(u)B) \\ & \vee p_C(ux) \wedge p_0(\phi(u)C) \\ & \vee p_D(ux) \wedge p_0(\phi(u)D) \\ & \vee p_L(ux) \wedge p_0(\phi(u)L) \\ & \vee p_U(ux) \wedge p_0(\phi(u)U) \end{aligned} \\
\omega & \text{if} \quad \begin{aligned}[t] & p_G(ux) \wedge p_\omega(\phi(u)G) \\ & \vee p_B(ux) \wedge p_\omega(\phi(u)B) \\ & \vee p_C(ux) \wedge p_\omega(\phi(u)C) \\ & \vee p_D(ux) \wedge p_\omega(\phi(u)D) \\ & \vee p_L(ux) \wedge p_\omega(\phi(u)L) \\ & \vee p_U(ux) \wedge p_\omega(\phi(u)U) \end{aligned} \\
unknown & \text{if} \quad \begin{aligned}[t] & \neg(p_G(ux) \vee p_B(ux) \vee p_C(ux) \\ & \vee p_D(ux) \vee p_L(ux) \vee p_U(ux)) \end{aligned}
\end{cases}
\end{aligned}
$$

6 Compatibility

It is obvious that BB_X is a refinement of both BB_{Y_1} and BB_{Y_2}. Consider the two black box functions with the same value set R:

$BB_{Y_1} = \langle Y_1, R, \Sigma_{Y_1}, \Gamma_{Y_1} \rangle$

$BB_{Y_2} = \langle Y_2, R, \Sigma_{Y_2}, \Gamma_{Y_2} \rangle$

Consider the following abstraction $\phi_{12} : Y_1^* \to Y_2^*$

$$
\begin{array}{l}
\phi_{12}(\lambda) = \lambda \\
\forall u \in Y_1^*,\ \forall x \in Y_1, \\
\phi_{12}(ux) = \begin{cases}
\phi_{12}(u)G & \text{if } p_G(ux) \\
\phi_{12}(u)B & \text{if } p_B(ux) \\
\phi_{12}(u)C & \text{if } p_C(ux) \\
\phi_{12}(u)D & \text{if } p_D(ux) \\
\phi_{12}(u)L & \text{if } p_L(ux) \\
\phi_{12}(u)U & \text{if } p_U(ux) \\
\phi_{12}(u) & \text{otherwise}
\end{cases}
\end{array}
$$

The formal definitions for the characteristic predicates are the same as the definitions in Section 2.3, except that $x \in \{0, 1, \ldots, 9\}$ is replaced with $x = d$ whenever it appears.

Consider the equivalence classes represented by canonical sequences under Moore equivalence for Y_1^* and Y_2^*.

Canonical Sequence in Y_1^*	Canonical Sequence in Y_2^*
λ	λ
L	L
U	U
UD	UD
$Ldd[d, p]$	LG
$Ldd[d, \neg p]$	LB
Ld	
Ldd	

$\forall u \in Y_1^*,$

$$\begin{array}{ll}
\text{if } u \in [\lambda] & \phi(u) \in [\lambda] \\
\text{if } u \in [L] & \phi(u) \in [L] \\
\text{if } u \in [U] & \phi(u) \in [U] \\
\text{if } u \in [UD] & \phi(u) \in [UD] \\
\text{if } u \in [Ldd[d,p]] & \phi(u) \in [LG] \\
\text{if } u \in [Ldd[d,\neg p]] & \phi(u) \in [LB] \\
\text{if } u \in [Ld] & \phi(u) \in [L] \cup [LB] \\
\text{if } u \in [Ldd] & \phi(u) \in [L] \cup [LB]
\end{array}$$

thus $\Gamma_{Y_1}([u]_{\Sigma_{Y_1}}) \subseteq \Gamma_{Y_2}([\phi(u)]_{\Sigma_{Y_2}})$

From above we can claim BB_X is a refinement of BB_{Y_1}, which is a refinement of BB_{Y_2}.

References

[1] Stacy J. Prowell and Jesse H. Poore, "Foundations of Sequence-Based Software Specification", *IEEE Transactions on Software Engineering*, Vol. 29, No. 5, May 2003.