

Lecture

Specifications via Timed ASMs

Requirements

Text: Requirements Engineering Management Handbook

EECS4312 Software Requirements Engineering
Fall 2015

Jonathan Ostroff
York University

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Consider a device with a monitored variable pressure $p : \mathbb{R}$ and controlled variable *alarm* as the output as shown in the figure.



Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table

Validation
Strong invariant
Use Case

Consider a device with a monitored variable pressure $p : \mathbb{R}$ and controlled variable *alarm* as the output as shown in the figure.



Function Table

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Abstract State Machines

Consider a device with a monitored variable pressure $p : \mathbb{R}$ and controlled variable *alarm* as the output as shown in the figure.



Function Table

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table

Validation

Strong invariant

Use Case

Abstract State Machines

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

- The function table specifies the output *alarm* in terms of the input p (and real-valued constants *normal* and *hi*).

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Abstract State Machines

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

- The function table specifies the output *alarm* in terms of the input p (and real-valued constants *normal* and *hi*).
- The function table specifies when the output *alarm* will go high, i.e. generate an alarm signal. “No Change” means that *alarm* stays the same as in the previous state.

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Abstract State Machines

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

- The function table specifies the output *alarm* in terms of the input p (and real-valued constants *normal* and *hi*).
- The function table specifies when the output *alarm* will go high, i.e. generate an alarm signal. “No Change” means that *alarm* stays the same as in the previous state.
- In validating such specifications we have to deal with an infinite number of possible inputs (some slice of the real line representing pressures), as opposed to digital circuits such as the majority voting circuit or the Date Validity function table.

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does “old” mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of “old”

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Abstract State Machines

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

- The function table specifies the output *alarm* in terms of the input p (and real-valued constants *normal* and *hi*).
- The function table specifies when the output *alarm* will go high, i.e. generate an alarm signal. “No Change” means that *alarm* stays the same as in the previous state.
- In validating such specifications we have to deal with an infinite number of possible inputs (some slice of the real line representing pressures), as opposed to digital circuits such as the majority voting circuit or the Date Validity function table.
- Furthermore, how would we specify “No Change” from the previous time an input was received?

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table

Validation

Strong invariant

Use Case

Abstract State Machines

	<i>alarm</i>
$p \leq \text{normal}$	<i>False</i>
$\text{normal} < p < \text{hi}$	No Change
$p \geq \text{hi}$	<i>True</i>

- The function table specifies the output *alarm* in terms of the input p (and real-valued constants *normal* and *hi*).
- The function table specifies when the output *alarm* will go high, i.e. generate an alarm signal. “No Change” means that *alarm* stays the same as in the previous state.
- In validating such specifications we have to deal with an infinite number of possible inputs (some slice of the real line representing pressures), as opposed to digital circuits such as the majority voting circuit or the Date Validity function table.
- Furthermore, how would we specify “No Change” from the previous time an input was received?
- No Change (NC) means we need a state machine that has a present state and a previous state.**

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function

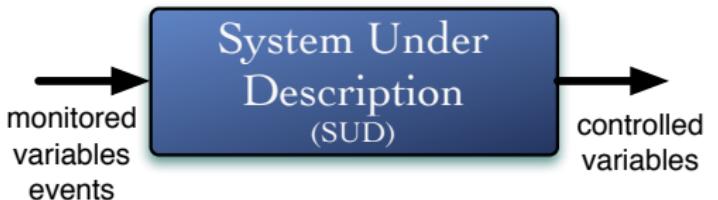
Table

Validation

Strong invariant

Use Case

Monitored and Controlled Variables



- What are the monitored events and variables?
- What are the controlled variables?
- What is the relationship between monitored variables and controlled variables

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Bank ATM Requirements

Requirements



Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case





- R1 A user shall be able to deposit and withdraw money at the ATM
- R2 The current balance, total deposits and total withdrawals are displayed at the ATM
- R3 The balance shall never be negative

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

Grammar: What are the input operations?

R1

A user shall be able to deposit and withdraw money at the ATM



Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Grammar: What are the input operations?

R1

A user shall be able to deposit and withdraw money at the ATM



bank-definitions.txt

```
system small_bank
nothing
deposit(v: REAL)
withdraw(v: REAL)
```

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Abstract user interface



```
system small_bank
nothing
deposit(v: REAL)
withdraw(v: REAL)
```

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Abstract user interface



```
system small_bank
nothing
deposit(v: REAL)
withdraw(v: REAL)
```

Abstract UI

The grammar defined by `bank-definitions.txt` describes an abstract user interface without the need to prematurely decide on an actual concrete UI.

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Output: Controlled variables

R2

The current balance, total deposits and total withdrawals are displayed at the ATM

Abstract state (from which the Output is derived)

Variable	Type	Description/Units
b	\mathbb{R}	balance in dollars
d	\mathbb{R}	total deposits in dollars
w	\mathbb{R}	total withdrawals

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

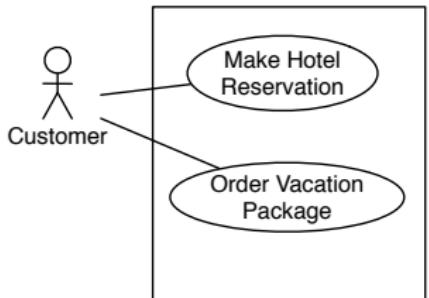
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

UML Use Cases

Hotel Reservation System



Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

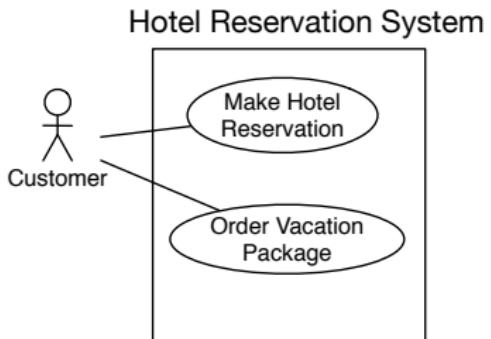
Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

UML Use Cases



Textual Use Case

- ① Enter (city, arrival, departure, room-type) and click Search
- ② **System:** Display list of hotels
- ③ Click on hotel's logo to read details
- ④ **System:** Display hot details
- ⑤ Click Book Now
- ⑥ **System:** Display payment form
- ⑦ Enter customer details, billing information and click Submit
- ⑧ **System:** Validate billing information and display result

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

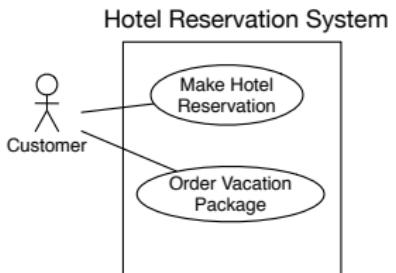
Table

Validation

Strong invariant

Use Case

UML Use Cases



What is a Use Case?

- A use case is a list of steps/events that describe the interactions between an external **Actor** and a **System** (software or hardware), to achieve a customer goal

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

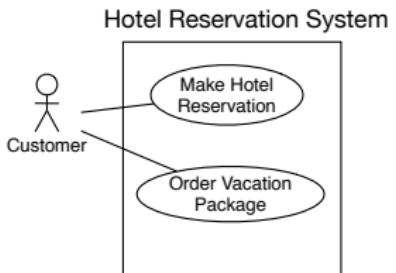
PVS Function Table

Validation

Strong invariant

Use Case

UML Use Cases



What is a Use Case?

- A use case is a list of steps/events that describe the interactions between an external **Actor** and a **System** (software or hardware), to achieve a customer goal
- An Actor can be a human, time or an external entity (such as a sensor, actuator, database, etc.)

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

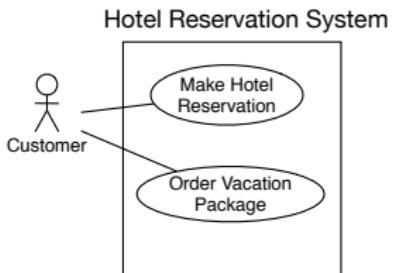
Table

Validation

Strong invariant

Use Case

UML Use Cases



What is a Use Case?

- A use case is a list of steps/events that describe the interactions between an external **Actor** and a **System** (software or hardware), to achieve a customer goal
- An Actor can be a human, time or an external entity (such as a sensor, actuator, database, etc.)
- Use case analysis is an important and valuable requirements analysis technique that has been widely used in modern software engineering

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

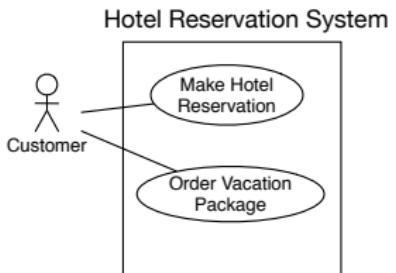
Table

Validation

Strong invariant

Use Case

UML Use Cases



What is a Use Case?

- A use case is a list of steps/events that describe the interactions between an external **Actor** and a **System** (software or hardware), to achieve a customer goal
- An Actor can be a human, time or an external entity (such as a sensor, actuator, database, etc.)
- Use case analysis is an important and valuable requirements analysis technique that has been widely used in modern software engineering
- It is part of the UML standard that you must master.
- See https://wiki.eecs.yorku.ca/project/eiffel/_media/bon:uml.pdf



Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

Use Case 1: ATM Deposit and withdraw

```
init
-> deposit(100)
  b: 100
  d: 100
  w: 0
-> withdraw(50)
  b: 50
  d: ???
  w: ???
-> withdraw(50)
  b: 0
  d: ???
  w: 100
-> withdraw(1)
  b: ???
  d: ???
  w: ???
```

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Use Case 1: Deposit and withdraw

```
init
-> deposit(100)
  b: 100
  d: 100
  w: 0
-> withdraw(50)
  b: 50
  d: 100
  w: 50
-> withdraw(50)
  b: 0
  d: 100
  w: 100
-> withdraw(1)
  b: ???
  d: 100
  w: 100
```

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Use Case 2: Nothing, deposit and withdraw

```
init
-> deposit(100)
  b: 100
  d: 100
  w: 0
-> withdraw(50)
  b: 50
  d: 100
  w: 50
-> nothing
  b: ???
  d: ???
  w: ???
-> deposit(125)
  b: ???
  d: ???
  w: ???
```

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Use Case 2: Nothing, deposit and withdraw

```
init
-> deposit(100)
  b: 100
  d: 100
  w: 0
-> withdraw(50)
  b: 50
  d: 100
  w: 50
-> nothing
  b: NC -- how to express no change?
  d: 100
  w: 50
-> deposit(125)
  b: ???
  d: ???
  w: ???
```

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Other Operations?

Other operations

How about transferring from one account to another?

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Other Operations?

Other operations

How about transferring from one account to another?

```
system bank
new(id: STRING)
deposit(id: STRING; amount: REAL)
withdraw(id: STRING; amount: REAL)
transfer(to: STRING; from: STRING; amount: REAL)
```

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Other Use Cases?

Other Use Cases

How many use cases can we come up with?

Complete Specifications?

What we need is a specification that is

- complete

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Other Use Cases?

Other Use Cases

How many use cases can we come up with?

Complete Specifications?

What we need is a specification that is

- complete
- Can be used to validate all use cases

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

Specification of ATM via a Function Table

	b	d	w	report
nothing	old b	old d	old w	old report
deposit(v)				
withdraw(v)				

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

Function Table: What does “old” mean?

	b	d	w	report
nothing	old b	old d	old w	old report
deposit(v)				
withdraw(v)				

What does old mean?

In an Eiffel routine it is clear.

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables

What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Function Table: What does “old” mean?

	b	d	w	report
nothing	old b	old d	old w	old report
deposit(v)				
withdraw(v)				

What does old mean?

In an Eiffel routine it is clear.

But we are dealing with console input/output modelling an abstract user interface!

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables

What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

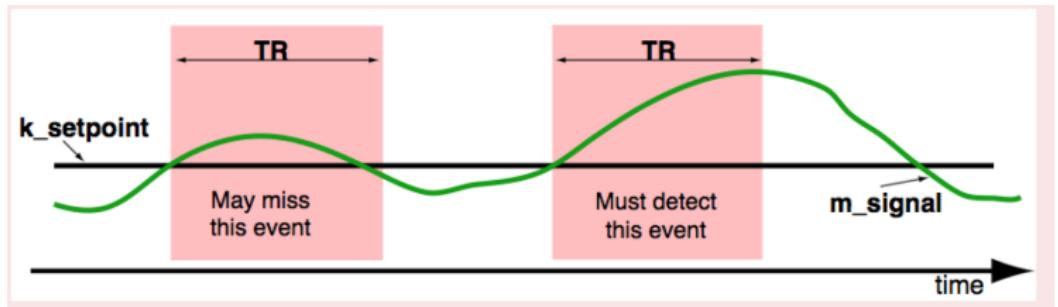
Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case



Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time i

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

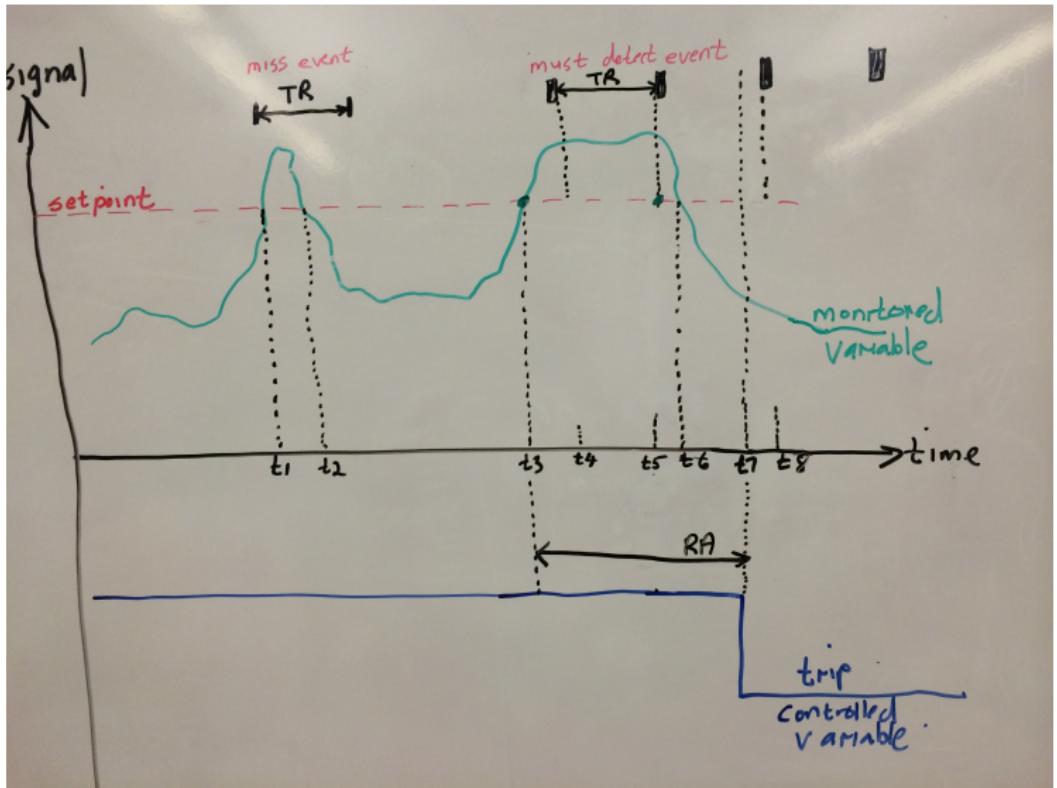
Table

Validation

Strong invariant

Use Case

Timing Resolution (TR) and Response Allowance (RA)



Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

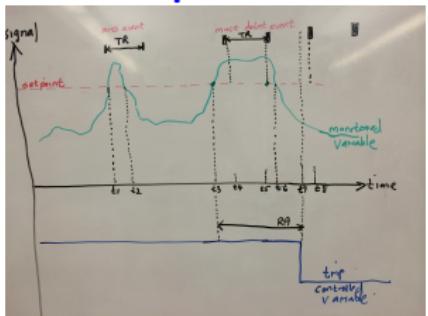
PVS Function
Table

Validation

Strong invariant

Use Case

Relationship between TR and RA



Relationship

- The timing resolution (TR) for a time continuous monitored variable is the minimum time duration of an initiating event dependent on that monitored variable for which the application must guarantee that it will detect that event. Thus, the TR is also an indication of the maximum time interval that the control computer can allow between successive sampling instances for that stimulus.
- The RA is measured from the time the measured event actually occurred in the physical domain, until the time the value of the controlled variable is generated and crosses the application boundary into the physical domain.

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

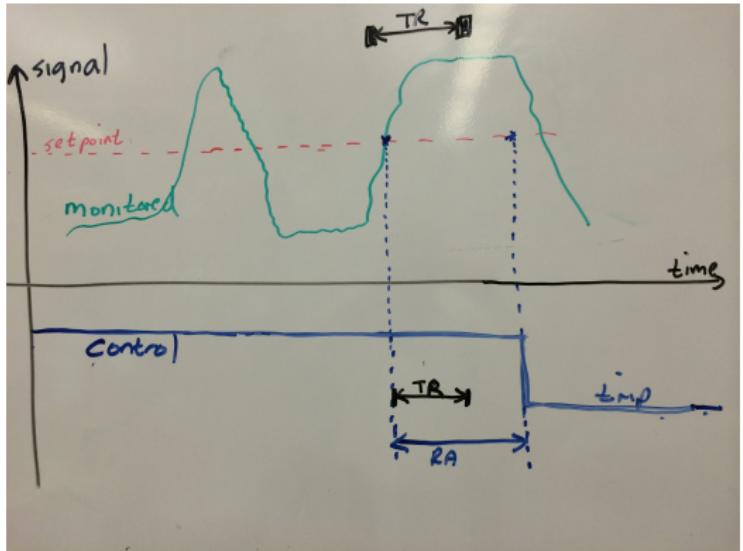
Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Relationship between TR and RA



What is the relationship between TR and RA?

- Timing Resolution (TR) and Response Allowance (RA)

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time i

Timed ASM
Theory
Meaning of "old"

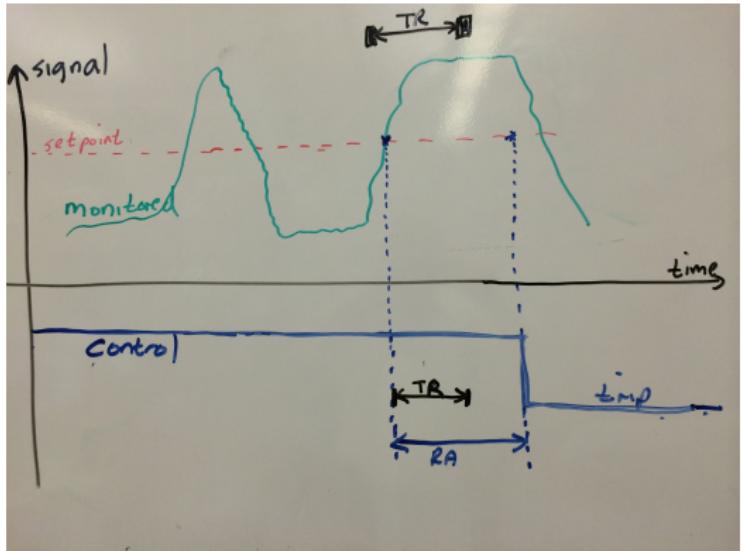
Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Relationship between TR and RA



What is the relationship between TR and RA?

- Timing Resolution (TR) and Response Allowance (RA)
- $TR < RA$

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

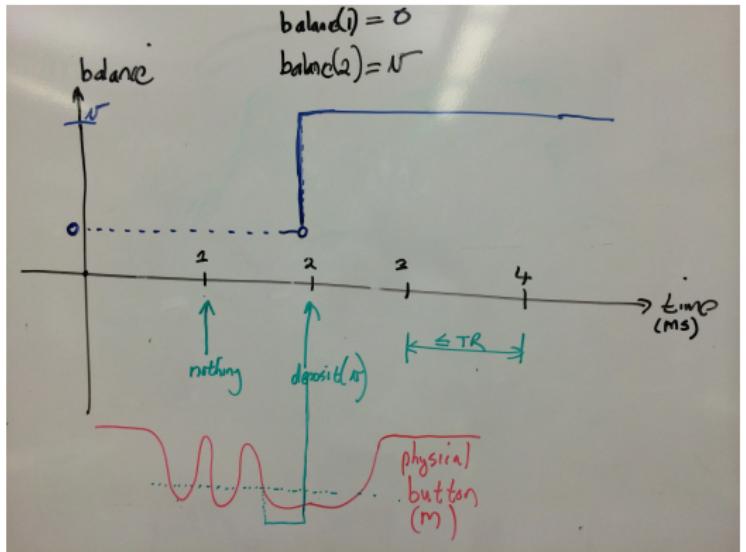
Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Discrete time i (relative to real-time)



Time i

- $i = 1$ is at the same time as real-time $t = .001s$
- $i = 2$ is at the same time as real-time $t = .002s$
- $\text{balance}(0) = 0$, $\text{balance}(1) = 0$, $\text{balance}(2) = v$
- $i = 2 \Rightarrow \text{balance}(i) = \text{balance}(i-1) + v$

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time i

Timed ASM
Theory
Meaning of "old"

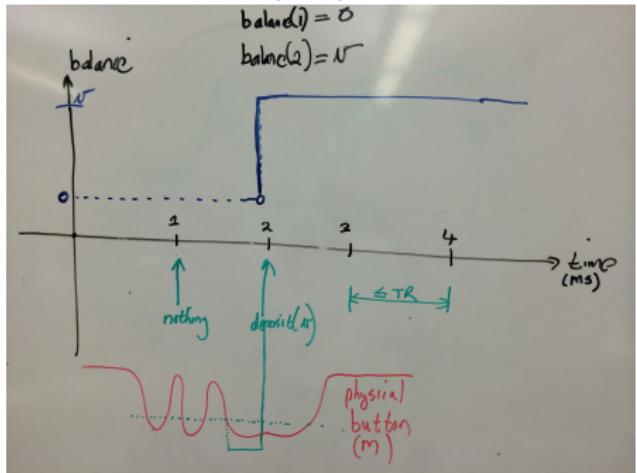
Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Physical button (red)



Separation of concerns

- At a hardware level, some circuits might take care of dealing with the electrical signal.
- Alternatively, the graphic runtime (e.g. Swing, WinForms, Eiffel Vision, Android View framework) and the OS might take care of button presses, swipes and other input actions.
- You might also have to take care of the (red) physical signal in software. If so, you can specify a hardware hiding module.

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution

Discrete time i

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Meaning of “old” – in Timed ASM Theory



Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Meaning of “old”– in Timed ASM Theory

```

DTIME : TYPE = nat
% psuedo, digitized real time
RTIME : TYPE =
{t : nnreal | (EXISTS (i : DTIME) : t = i * delta) }
% conversions
r2d(t: RTIME): DTIME = t / delta
d2r(i: DTIME): RTIME = i * delta
  
```

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does “old” mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of “old”

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Meaning of “old” – in Timed ASM Theory

```

DTIME : TYPE = nat
% psuedo, digitized real time
RTIME : TYPE =
{t : nnreal | (EXISTS (i : DTIME) : t = i * delta) }
% conversions
r2d(t: RTIME) : DTIME = t / delta
d2r(i: DTIME) : RTIME = i * delta
  
```

- Convert digital instance i to real-time:
 $d2r(i) = i * \text{delta} = i * .001s = .001 * i$ seconds,
 where delta is the sampling rate which must be less than the TR.
- i is the instance at which the press of the deposit button signal (generated by the user in the environment) crosses the application boundary and is detected by the computer.
- $\text{deposit}(i)$ is thus the instant at which the deposit event is detected by the computer at the monitored input.

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does “old” mean?

Timing Resolution
Discrete time i

Timed ASM Theory

Meaning of “old”

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Meaning of “old”

- i is the instance at which the press of the deposit button signal (generated by the user in the environment) crosses the application boundary and is detected by the computer.
- $i-1$ is the sampling instant before i
- In a requirements documents, the “idealized” computer can generate an instantaneous change in the controlled variables at instant i based on the monitored input at i , $i-1$ and previous sampling instants.
- However, when the computer system is implemented, we may allow RA seconds to elapse from the time the button was clicked in the environment.

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Meaning of “old”

- i is the instance at which the press of the deposit button signal (generated by the user in the environment) crosses the application boundary and is detected by the computer.
- $i-1$ is the sampling instant before i
- In a requirements documents, the “idealized” computer can generate an instantaneous change in the controlled variables at instant i based on the monitored input at i , $i-1$ and previous sampling instants.
- However, when the computer system is implemented, we may allow RA seconds to elapse from the time the button was clicked in the environment.

Meaning of old

old balance is thus: $balance(i-1)$, i.e. the value of the balance in the sampling instant before i

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of “old”

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

How do we specify Bank as a Function Table using T-ASM?



Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

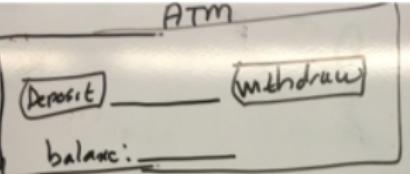
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

How do we specify Bank as a Function Table using T-ASM?

ATM



$b: \text{VAR real}$
 $d: \text{VAR real}$
 $w: \text{VAR real}$

-- balance.
 -- local deposits
 -- withdrawals
 -- " mean?

outputs

inputs	$\text{cmd}(i)$	$b(i)$	$d(i)$	$w(i)$
$i=0$				
$i > 0$	nothing			
	$\text{deposit}(\epsilon: \text{R})$?	?	?
	$\text{withdraw}(\epsilon: \text{R})$			

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Revised Function Table

Conditions

- What is c1?
- Why not just c1 and $\neg c1$.
- What is c2 and c3?
- What is NC?

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$	$b(i-1) + v$	$d(i-1) + v$	ok
		$v \leq 0$	NC	NC	er1
	with(v)	c1	$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$
		$\neg c1$	c2	NC	NC
			c3	NC	er2

Abbreviations help to fit the function table on one page

Revised Function Table

Conditions

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

NC means No Change

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$	$b(i-1) + v$	$d(i-1) + v$	ok
		$v \leq 0$	NC	NC	er1
	with(v)	c1	$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$
		$\neg c1$	NC	NC	er1
			c3	NC	er2

What we have specified is a finite state machine enriched with abstract state

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table

Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Function Table

Completeness

Is the function table complete and disjoint?

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Function Table

Completeness

Is the function table complete and disjoint?

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$	$b(i-1) + v$	$d(i-1) + v$	ok
		$v \leq 0$	NC	NC	er1
	with(v)	c1	$b(i-1) - v$	$w(i-1) + v$	ok
		$\neg c1$	c2	NC	er1
			c3	NC	er2

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table

Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Function Table Conciseness/Precision

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: $cmd(i)$			$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$			0	0	0	ok
$i > 0$	nothing		$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$		$b(i-1) + v$	$d(i-1) + v$	$w(i-1)$
		$v \leq 0$		NC	NC	er1
	with(v)	c1		$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$
		$\neg c1$	c2	NC	NC	er1
			c3	NC	NC	er2

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

Function Table Conciseness/Precision

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$	$b(i-1) + v$	$d(i-1) + v$	$w(i-1)$
		$v \leq 0$	NC	NC	er1
	with(v)	c1	$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$
		$\neg c1$	NC	NC	er1
		c2	NC	NC	er2

Conciseness

- How long would it take to write out the specification, informally, in English?
- Would it be vague? Or would it have the precision of the function table?
- Would the informal English be complete and disjoint?

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Function Table Conciseness/Precision

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$v > 0$	$b(i-1) + v$	$d(i-1) + v$	$w(i-1)$
		$v \leq 0$	NC	NC	er1
	with(v)	c1	$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$
		$\neg c1$	NC	NC	er1
		c2	NC	NC	er2

Conciseness

- How long would it take to write out the specification, informally, in English?
- Would it be vague? Or would it have the precision of the function table?
- Would the informal English be complete and disjoint?
- Recall something as simple as leap-year?

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Validation of invariants and use cases

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: <i>cmd(i)</i>		<i>b(i)</i>	<i>d(i)</i>	<i>w(i)</i>	<i>r(i)</i>
<i>i</i> = 0		0	0	0	ok
<i>i</i> > 0	nothing	<i>b(i-1)</i>	<i>d(i-1)</i>	<i>w(i-1)</i>	<i>r(i-1)</i>
	dep(<i>v</i>)	<i>b(i-1) + v</i>	<i>d(i-1) + v</i>	<i>w(i-1)</i>	ok
	<i>v ≤ 0</i>	NC	NC	NC	er1
	c1	<i>b(i-1) - v</i>	<i>d(i-1)</i>	<i>w(i-1) + v</i>	ok
	with(<i>v</i>)	NC	NC	NC	er1
	¬ c1	c3	NC	NC	er2

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness

Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Validation of invariants and use cases

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: <i>cmd</i> (<i>i</i>)		<i>b</i> (<i>i</i>)	<i>d</i> (<i>i</i>)	<i>w</i> (<i>i</i>)	<i>r</i> (<i>i</i>)
<i>i</i> = 0		0	0	0	ok
<i>i</i> > 0	nothing	<i>b</i> (<i>i</i> -1)	<i>d</i> (<i>i</i> -1)	<i>w</i> (<i>i</i> -1)	<i>r</i> (<i>i</i> -1)
	dep(<i>v</i>)	<i>b</i> (<i>i</i> -1) + <i>v</i>	<i>d</i> (<i>i</i> -1) + <i>v</i>	<i>w</i> (<i>i</i> -1)	ok
		NC	NC	NC	er1
	with(<i>v</i>)	<i>b</i> (<i>i</i> -1) - <i>v</i>	<i>d</i> (<i>i</i> -1)	<i>w</i> (<i>i</i> -1) + <i>v</i>	ok
		NC	NC	NC	er1
	¬ c1	c3	NC	NC	er2

Validation

- Can we use the function table to prove R3, viz, that $\forall i \in DTIME : balance(i) \geq 0$?

Abstract State Machines

System Boundary
Bank Requirements

Input/Output

Use Cases

Function Tables
What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory
Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Validation of invariants and use cases

$$c1 \hat{=} v > 0 \wedge b(i-1) - v \geq 0$$

$$c2 \hat{=} v \leq 0$$

$$c3 \hat{=} b(i-1) - v < 0$$

monitored inputs: $cmd(i)$		$b(i)$	$d(i)$	$w(i)$	$r(i)$
$i = 0$		0	0	0	ok
$i > 0$	nothing	$b(i-1)$	$d(i-1)$	$w(i-1)$	$r(i-1)$
	dep(v)	$b(i-1) + v$	$d(i-1) + v$	$w(i-1)$	ok
	$v \leq 0$	NC	NC	NC	er1
	c1	$b(i-1) - v$	$d(i-1)$	$w(i-1) + v$	ok
	with(v)	NC	NC	NC	er1
	$\neg c1$	c3	NC	NC	er2

Validation

- Can we use the function table to prove R3, viz, that $\forall i \in DTIME : balance(i) \geq 0$?
- Can we use the function table to show that a Use Case is satisfied?
- How?

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Time Theory

Time Theory has parameter *delta*, the sampling time.



Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Time Theory

Time Theory has parameter *delta*, the sampling time.

```

Time[delta: posreal] : THEORY
BEGIN
  % digital time
  DTIME : TYPE = nat

  % psuedo, digitized real time
  RTIME : TYPE =
    {t:nnreal | (EXISTS (i:DTIME) : t = i * delta) }

  % actual time
  TIME : TYPE = nnreal

  % Positive DTIME
  POS_DTIME: TYPE = posnat

  ...other functions

END Time
  
```

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Bank Theory

```

SmallBank[delta:posreal]: THEORY
BEGIN
  IMPORTING Time[delta]
  i: VAR DTIME

  STATUS      : TYPE = {ok, er1, er2}
  % Inputs
  v: real

  COMMAND: DATATYPE
  BEGIN
    nothing          : nothing?
    deposit (x:real) : deposit?
    withdraw (x:real) : withdraw?
  END COMMAND

  cmd: [DTIME -> COMMAND]

  % Outputs
  ...

```

[Abstract State Machines](#)

[System Boundary](#)

[Bank Requirements](#)

[Input/Output](#)

[Use Cases](#)

[Function Tables](#)

[What does "old" mean?](#)

[Timing Resolution](#)
[Discrete time /](#)

[Timed ASM Theory](#)

[Meaning of "old"](#)

[Tabular Specifications](#)

[Revised table](#)
[Completeness](#)
[Conciseness](#)
[Validation](#)

[DTIME theory](#)

[Bank I/O Vars](#)

[Monitored Events](#)
[DATATYPE](#)
[PVS Function Table](#)
[Validation](#)
[Strong invariant](#)
[Use Case](#)

Bank Theory

```

SmallBank[delta:posreal]: THEORY
BEGIN
  IMPORTING Time[delta]
  i: VAR DTIME

  STATUS      : TYPE = {ok, er1, er2}
  % Inputs
  v: real

  COMMAND: DATATYPE
  BEGIN
    nothing          : nothing?
    deposit (x:real) : deposit?
    withdraw (x:real) : withdraw?
  END COMMAND

  cmd: [DTIME -> COMMAND]

  % Outputs
  ...

```

Note that formatting is important!

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

Monitored Events as a DATATYPE

```

...
COMMAND: DATATYPE
BEGIN
    nothing          : nothing?
    deposit (x:real) : deposit?
    withdraw (x:real) : withdraw?
END COMMAND

i: VAR DTIME
cmd: VAR [DTIME -> COMMAND]
...

```

Commands

- The commands are constructors
- Can say: $\text{cmd}(i) = \text{deposit}(v)$ AND $b(i) = 425.00 \dots$

See PVS Language Reference

**Abstract State
Machines**

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE
PVS Function
Table

Validation

Strong invariant

Use Case

Monitored Events as a DATATYPE

An example of a datatype is stack:

```
stack[T: TYPE]: DATATYPE
BEGIN
empty: empty?
push(top:T, pop:stack): nonempty?
END stack
```

The stack datatype has two constructors, empty and push, that allow stack elements to be constructed. For example, the term $\text{push}(1, \text{empty})$ is an element of type $\text{stack}[\text{int}]$.

The recognizers `empty?` and `nonempty?` are predicates over the stack datatype that are true when their argument is constructed using the corresponding constructor. Given a stack element that is known to be `nonempty?`, the accessors `top` and `pop` may be used to extract the first and second arguments.

See PVS Language Reference p72

[Abstract State Machines](#)

[System Boundary](#)

[Bank Requirements](#)

[Input/Output](#)

[Use Cases](#)

[Function Tables](#)

[What does "old" mean?](#)

[Timing Resolution](#)
[Discrete time /](#)

[Timed ASM Theory](#)

[Meaning of "old"](#)

[Tabular Specifications](#)

[Revised table](#)
[Completeness](#)
[Conciseness](#)
[Validation](#)

[DTIME theory](#)

[Bank I/O Vars](#)
[Monitored Events](#)

DATATYPE

[PVS Function Table](#)
[Validation](#)

[Strong invariant](#)
[Use Case](#)

Outputs and Queries

% Outputs

```
b: [DTIME -> real]      % balance
d: [DTIME -> real]      % total deposits
w: [DTIME -> real]      % total withdraws
r: [DTIME -> STATUS]    % report
```

% Queries

```
deposit?(x: real) (i:POS_DTIME): bool =
  b(i) = b(i-1) + x AND d(i) = d(i-1) + x
  AND w(i) = w(i-1)
```

```
withdraw?(x: real) (i:POS_DTIME): bool =
  b(i) = b(i-1) - x AND d(i) = d(i-1)
  AND w(i) = w(i-1) + x
```

```
skip(i:POS_DTIME): bool =
  b(i) = b(i-1) AND d(i) = d(i-1)
  AND w(i) = w(i-1)
```

```
init: bool =
  b(0) = 0 AND d(0) = 0 AND w(0) = 0
```

**Abstract State
Machines**

**System Boundary
Bank
Requirements**

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**
Meaning of "old"

**Tabular
Specifications**
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

PVS Function Table 1

Requirements



```
% Response Function Table
bank_ft (i): bool =
  COND
    i = 0 -> init AND r(0) = ok,
    i > 0 ->
      CASES cmd(i) OF
        nothing : skip(i) AND r(i) = r(i-1),
        deposit(v):
          COND
            v > 0 -> deposit?(v)(i) AND r(i) = ok,
            ELSE -> skip(i)           AND r(i) = erl
          ENDCOND,
        withdraw(v): ...
      ENDCASES
  ENDCOND
```

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

PVS Function Table 1

Requirements



% Response Function Table

```
bank_ft (i) : bool =  
  COND  
    i = 0 -> init AND r(0) = ok,  
    i > 0 ->  
      CASES cmd(i) OF  
        nothing : skip(i) AND r(i) = r(i-1),  
        deposit(v) :  
          COND  
            v > 0 -> deposit?(v)(i) AND r(i) = ok,  
            ELSE -> skip(i) AND r(i) = erl  
          ENDCOND,  
          withdraw(v) : ...  
      END_CASES  
  ENDCOND
```

Do not confuse deposit(v) as a command and deposit?(x) as a query

Use CASES for the commands (datatype).

Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

PVS Function Table 2

Requirements



% Response Function Table

COND ...

i > 0 ->

CASES cmd(i) OF

nothing : ...
deposit(v) : ...
withdraw(v) :

COND

v > 0 AND b(i-1) - v >= 0
-> withdraw?(v)(i) AND r(i) = ok,

NOT (v > 0 AND b(i-1) - v >= 0)

-> COND

v <= 0

-> skip(i) AND r(i) = er1,

b(i-1) - v < 0

-> skip(i) AND r(i) = er2

ENDCOND

ENDCOND

ENDCASES

ENDCOND

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

PVS Function Table without Naming Abstraction

Requirements

% Response

```
bank_ft1 (cmd, b, d, w, r)(i): bool =      ...
withdraw(v):
  COND
    v > 0 AND b(i-1) - v >= 0
    ->      b(i) = b(i-1) - v
            AND d(i) = d(i-1)
            AND w(i) = w(i-1) + v
            AND r(i) = ok,
    NOT (v > 0 AND b(i-1) - v >= 0)
    ->  COND
        v <= 0 ->      b(i-1) = b(i)
                    AND d(i-1) = d(i)
                    AND w(i-1) = w(i)
                    AND r(i) = er1,
        b(i-1) - v < 0 -> b(i-1) = b(i)
                    AND d(i-1) = d(i)
                    AND w(i-1) = w(i)
                    AND r(i) = er2
  ENDCOND
ENDCOND ...
```



Abstract State
Machines

System Boundary

Bank
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function
Table

Validation

Strong invariant

Use Case

So what can we do with all this?

Validation

How?

Invariants such as R3

Use Cases

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table

Validation

Strong invariant
Use Case

% Invariants

```

invariant1(i:DTIME) : bool =
  0 <= b(i)

bank_invariant1 : CONJECTURE
  (FORALL i: bank_ft(i) )
=>
  FORALL i : invariant1(i)

```

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table

Validation

Strong invariant
Use Case

% Invariants

```

invariant1(i:DTIME) : bool =
  0 <= b(i)

bank_invariant1 : CONJECTURE
  (FORALL i: bank_ft(i) )
=>
  FORALL i : invariant1(i)

```

bank_invariant1 :

```

{-1}  (FORALL i: bank_ft(i) )
| -----
{1}   FORALL i: invariant1(i)

```

Rule?

What rule to apply?

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function Table

Validation

Strong invariant
Use Case

```

invariant1(b, d, w) (i:DTIME) : bool = 0 <= b(i)

bank_invariant1 : CONJECTURE
  (FORALL i: bank_ft1(cmd, b, d, w, r) (i))
=>
  FORALL i : invariant1(b, d, w) (i)
  
```

```

{-1}  FORALL i: bank_ft1(cmd, b, d, w, r) (i)
      |
      -----
{1}   FORALL i: invariant2(b, d, w) (i)
  
```

Rule?

What rule to apply?

Induction

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

Validate invariant R3: induct results in 2 subgoals

```
{-1}  (FORALL i: bank_ft(i))
|-----
{1}   FORALL i: invariant1(i)
```

Rule? (induct "i")

Inducting on i on formula 1, yields 2 subgoals:
 bank_invariant1.1 :

```
[-1]  (FORALL i: bank_ft(i))
|-----
{1}   invariant1(0)
```

Rule?

bank_invariant1.2 :

```
[-1]  (FORALL i: bank_ft(i))
|-----
{1}   FORALL j:
          invariant1(j) IMPLIES invariant1(j + 1)
```

Rule?

**Abstract State
Machines**

System Boundary

**Bank
Requirements**

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
**PVS Function
Table**

Validation

Strong invariant

Use Case

Validate invariant R3: first sub-goal base case $i = 0$

```
bank_invariant1.1 :
[-1]  FORALL i: bank_ft(i)
      -----
{1}    invariant1(0)
Rule? (inst -1 "0")
```

Instantiating the top quantifier.... simplifies to:

```
bank_invariant1.1 :
{-1}  bank_ft(0)
      -----
[1]    invariant(0)
Rule? (grind)
```

invariant1 rewrites invariant1(0)
to $0 \leq b(0)$

Trying repeated skolemization, ...

This completes the proof of bank_invariant1.1.

bank_invariant1.2 :

**Abstract State
Machines**

System Boundary

**Bank
Requirements**

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
**PVS Function
Table**

Validation

Strong invariant
Use Case

Validate invariant R3: second sub-goal inductive case

```
bank_invariant1.2 :
[-1]  FORALL i: bank_ft(i)
      |-----
{1}   FORALL j: invariant1(j)
          IMPLIES invariant1(j + 1)
```

Rule? (skeep)

Skolemizing...simplifies to:

```
bank_invariant1.2 :
```

```
{-1}  invariant1(j)
[-2]  FORALL i: bank_ft(i)
      |-----
{1}   invariant1(j + 1)
```

Rule?

What rule now?

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Validate invariant R3: second sub-goal inductive case

```

bank_invariant1.2 :
{-1} invariant1(j)
[-2] FORALL i: bank_ft(i)
|-----
{1} invariant1(j + 1)
Rule? (inst -2 "j+1")
Instantiating...simplifies to:
bank_invariant1.2 :
[-1] invariant1(j)
{-2} bank_ft(j + 1)
|-----
[1] invariant1(j + 1)
Rule? (grind)
invariant1 rewrites ....
Trying repeated skolemization, ...
This completes the proof of bank_invariant1.2.
Q.E.D
  
```

[Abstract State Machines](#)

[System Boundary](#)
[Bank Requirements](#)

[Input/Output](#)

[Use Cases](#)

[Function Tables](#)
 What does "old" mean?

[Timing Resolution](#)
[Discrete time /](#)

[Timed ASM Theory](#)
 Meaning of "old"

[Tabular Specifications](#)

[Revised table](#)
[Completeness](#)
[Conciseness](#)
[Validation](#)

[DTIME theory](#)

[Bank I/O Vars](#)
[Monitored Events](#)

[DATATYPE](#)
[PVS Function Table](#)

[Validation](#)

[Strong invariant](#)
[Use Case](#)

Validate invariant R3: M-x show-proof

Requirements



Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table

Validation

Strong invariant
Use Case

```
;; Proof bank_invariant1-1
;;;for formula SmallBank.bank_invariant1
;; developed with shostak decision procedures
(""
(skeep)
(induct "i")
(("1" (inst -1 "0") (grind))
 ("2" (skeep) (inst -2 "j+1") (grind))))
```

**Abstract State
Machines**

System Boundary

**Bank
Requirements**

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
**PVS Function
Table**
Validation

Strong invariant

Use Case

```
invariant2(i:DTIME) : bool =
    0 <= b(i)
AND 0 <= d(i)
AND 0 <= w(i)
AND d(i) - w(i) = b(i)

bank_invariant2 : CONJECTURE
  (FORALL i: bank_ft(i))
=>
  FORALL i : invariant2(i)
```

Proving an abstract Use Case 1

Requirements



```
use_case1_v1: CONJECTURE
    i > 0
    AND b(i) = 10.25
    AND bank_ft(i+1)
    AND bank_ft(i+2)
    AND cmd(i+1) = deposit(4.75)
    AND cmd(i+2) = withdraw(10.00)
=>
    b(i+2) = 5.00
```

- What proof rule?

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

```
use_case1_v1: CONJECTURE
    i > 0
    AND b(i) = 10.25
    AND bank_ft(i+1)
    AND bank_ft(i+2)
    AND cmd(i+1) = deposit(4.75)
    AND cmd(i+2) = withdraw(10.00)
=>
    b(i+2) = 5.00
```

- What proof rule?
- Try grind

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Proving an abstract Use Case 2

```
% Need to figure out what XXXX is?
% Use the method of weakest preconditions
x,y: posreal
use_case2: CONJECTURE
    i > 0
    AND XXXX
    AND bank_ft1(i+1)
    AND bank_ft1(i+2)
    AND cmd(i+1) = deposit (x)
    AND cmd(i+2) = withdraw(y)
=>
    b(i+2) = b(i) + x - y
    AND d(i+2) = d(i) + x
    AND w(i+2) = w(i) + y
```

- Work out what **XXX** is using weakest preconditions

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

Proving an abstract Use Case 2

```
% Need to figure out what XXXX is?
% Use the method of weakest preconditions
x,y: posreal
use_case2: CONJECTURE
  i > 0
  AND XXXX
  AND bank_ft1(i+1)
  AND bank_ft1(i+2)
  AND cmd(i+1) = deposit (x)
  AND cmd(i+2) = withdraw(y)
=>
  b(i+2) = b(i) + x - y
  AND d(i+2) = d(i) + x
  AND w(i+2) = w(i) + y
```

- Work out what **XXX** is using weakest preconditions
- What proof rule?

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE

PVS Function Table

Validation

Strong invariant

Use Case

Proving an abstract Use Case 2

```
% Need to figure out what XXXX is?
% Use the method of weakest preconditions
x,y: posreal
use_case2: CONJECTURE
  i > 0
  AND XXXX
  AND bank_ft1(i+1)
  AND bank_ft1(i+2)
  AND cmd(i+1) = deposit (x)
  AND cmd(i+2) = withdraw(y)
=>
  b(i+2) = b(i) + x - y
  AND d(i+2) = d(i) + x
  AND w(i+2) = w(i) + y
```

- Work out what **XXX** is using weakest preconditions
- What proof rule?
- Try grind

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

Using a RECORD for the state

State encoded as a record type and initial state

```
STATE: TYPE = [# b: real
               , d: real
               , w: real
               , r: STATUS
               #]
```

```
st: [DTIME -> STATE]
```

```
init_state: STATE =
  (# b := 0, d := 0, w:=0, r := ok #)
```

[Abstract State Machines](#)

[System Boundary](#)

[Bank Requirements](#)

[Input/Output](#)

[Use Cases](#)

[Function Tables](#)

[What does "old" mean?](#)

[Timing Resolution](#)
[Discrete time /](#)

[Timed ASM Theory](#)

[Meaning of "old"](#)

[Tabular Specifications](#)

[Revised table](#)
[Completeness](#)
[Conciseness](#)
[Validation](#)

[DTIME theory](#)

[Bank I/O Vars](#)
[Monitored Events](#)
[DATATYPE](#)
[PVS Function Table](#)
[Validation](#)
[Strong invariant](#)
[Use Case](#)

System I/O Specification

```

deposit_ft(x: real) (i:POS_DTIME): bool =
  COND
    x > 0 -> st(i) = st(i-1) WITH [ b_ := b_ + x
                                         , d_ := d_ + x
                                         , r := ok
                                         ],
    x <= 0 -> st(i) = st(i-1) WITH [r := er1]
  ENDCOND
  WHERE b_ = st(i-1)`b
        , d_ = st(i-1)`d
  
```

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

System I/O Specification

```

deposit_ft(x: real)(i:POS_DTIME): bool =
  COND
    x > 0 -> st(i) = st(i-1) WITH [ b_ := b_ + x
                                         , d_ := d_ + x
                                         , r := ok
                                         ],
    x <= 0 -> st(i) = st(i-1) WITH [r := er1]
  ENDCOND
  WHERE b_ = st(i-1)`b
        , d_ = st(i-1)`d

```

```

bank_ft(i): bool =
  COND
    i = 0 -> st(0) = init_state,
    i > 0 ->
      CASES cmd(i) OF
        nothing : st(i) = st(i-1),
        deposit(v) : deposit_ft(v)(i),
        withdraw(v): withdraw_ft(v)(i)
      ENDCASES
  ENDCOND

```

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

```

bank_ft (i) : bool =
  COND
    i = 0 -> st(0) = init_state,
    i > 0 ->
      CASES cmd(i) OF
        nothing      : st(i) = st(i-1),
        deposit(v)  : deposit_ft(v)(i),
        withdraw(v) : withdraw_ft(v)(i)
      ENDCASES
  ENDCOND

```

```

invariant1(i:DTIME) : bool =
  0 <= st(i)`b

```

[Abstract State Machines](#)

[System Boundary](#)

[Bank Requirements](#)

[Input/Output](#)

[Use Cases](#)

[Function Tables](#)

[What does "old" mean?](#)

[Timing Resolution](#)
[Discrete time /](#)

[Timed ASM Theory](#)

[Meaning of "old"](#)

[Tabular Specifications](#)

[Revised table](#)
[Completeness](#)

[Conciseness](#)
[Validation](#)

[DTIME theory](#)

[Bank I/O Vars](#)

[Monitored Events](#)

[DATATYPE](#)

[PVS Function Table](#)

[Validation](#)

[Strong invariant](#)

[Use Case](#)

```

bank_ft (i) : bool =
  COND
    i = 0 -> st(0) = init_state,
    i > 0 ->
      CASES cmd(i) OF
        nothing      : st(i) = st(i-1),
        deposit(v)   : deposit_ft(v)(i),
        withdraw(v) : withdraw_ft(v)(i)
      ENDCASES
  ENDCOND

```

```

invariant1(i:DTIME) : bool =
  0 <= st(i)`b

```

```

bank_invariant1 : CONJECTURE
  (FORALL i: bank_ft(i) )
  =>
  FORALL i : invariant1(i)

```

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table
Completeness

Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function Table

Validation
Strong invariant

Use Case

```

bank_ft (i) : bool =
  COND
    i = 0 -> st(0) = init_state,
    i > 0 ->
      CASES cmd(i) OF
        nothing     : st(i) = st(i-1),
        deposit(v) : deposit_ft(v)(i),
        withdraw(v): withdraw_ft(v)(i)
      ENDCASES
  ENDCOND

```

[Abstract State Machines](#)[System Boundary](#)[Bank Requirements](#)[Input/Output](#)[Use Cases](#)[Function Tables](#)[What does "old" mean?](#)[Timing Resolution](#)[Discrete time /](#)[Timed ASM Theory](#)[Meaning of "old"](#)[Tabular Specifications](#)[Revised table](#)[Completeness](#)[Conciseness](#)[Validation](#)[DTIME theory](#)[Bank I/O Vars](#)[Monitored Events](#)[DATATYPE](#)[PVS Function Table](#)[Validation](#)[Strong invariant](#)[Use Case](#)

**Abstract State
Machines**

System Boundary

**Bank
Requirements**

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
**PVS Function
Table**
Validation
Strong invariant
Use Case

Validation of Use Case Requirement

```

bank_ft (i) : bool =
COND
  i = 0 -> st(0) = init_state,
  i > 0 ->
    CASES cmd(i) OF
      nothing     : st(i) = st(i-1),
      deposit(v) : deposit_ft(v)(i),
      withdraw(v): withdraw_ft(v)(i)
    ENDCASES
  ENDCOND

```

% Proof this without grind

```

use_casel: CONJECTURE
  i > 0
  AND st(i)`b = 10.25
  AND bank_ft(i+1)
  AND bank_ft(i+2)
  AND cmd(i+1) = deposit(4.75)
  AND cmd(i+2) = withdraw(10.00)
  =>
  st(i+2)`b = 5.00

```

Informal Bank ATM Requirements

Deposit

\$42.50

Withdraw

Balance?

\$142.50

- R1 A user shall be able to deposit and withdraw money at the ATM
- R2 The current balance, total deposits and total withdrawals are displayed at the ATM
- R3 The balance shall never be negative

Compared with:

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Informal Bank ATM Requirements

Deposit

\$42.50

Withdraw

Balance?

\$142.50

- R1 A user shall be able to deposit and withdraw money at the ATM
- R2 The current balance, total deposits and total withdrawals are displayed at the ATM
- R3 The balance shall never be negative

Compared with:

The formal requirements document as a function table

With the formal document we can predict/calculate that it is safe and fit for use.

[Abstract State Machines](#)[System Boundary](#)[Bank Requirements](#)[Input/Output](#)[Use Cases](#)[Function Tables](#)[What does "old" mean?](#)[Timing Resolution](#)
[Discrete time /](#)[Timed ASM Theory](#)[Meaning of "old"](#)[Tabular Specifications](#)[Revised table](#)[Completeness](#)[Conciseness](#)[Validation](#)[DTIME theory](#)[Bank I/O Vars](#)[Monitored Events](#)[DATATYPE](#)[PVS Function Table](#)[Validation](#)[Strong invariant](#)[Use Case](#)

Informal vs. Precise Requirements Document

Toyota Crash

“Runaway
Toyota cases
ignored”

LA Times 2009



By RALPH VARTABEDIAN AND KEN BENSINGER

NOVEMBER 8, 2009

Los Angeles Times

More than 1,000 Toyota and Lexus owners have reported since 2001 that their vehicles suddenly accelerated on their own, in many cases slamming into trees, parked cars and brick walls, among other obstacles, a Times review of federal records has found.

The crashes resulted in at least 19 deaths and scores of injuries over the last decade, records show. Federal regulators say that is far more than any other automaker has experienced.

Owner complaints helped trigger at least eight investigations into sudden acceleration in Toyota and Lexus vehicles by the National Highway Traffic Safety Administration in the last seven years. Toyota Motor Corp. recalled fewer than 85,000 vehicles in response to two of those probes, and the federal agency closed six other cases without finding a defect.

**Abstract State
Machines**

**System Boundary
Bank,
Requirements**

Input/Output

Use Cases

Function Tables
What does “old”
mean?

**Timing Resolution
Discrete time /**

**Timed ASM
Theory**
Meaning of “old”

**Tabular
Specifications**

**Revised table
Completeness
Conciseness
Validation**

DTIME theory

**Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case**

Informal vs. Precise Requirements Document

Toyota Crash

“Runaway
Toyota cases
ignored”

LA Times 2009



By RALPH VARTABEDIAN AND KEN BENINGER

NOVEMBER 8, 2009

Los Angeles Times

More than 1,000 Toyota and Lexus owners have reported since 2001 that their vehicles suddenly accelerated on their own, in many cases slamming into trees, parked cars and brick walls, among other obstacles, a Times review of federal records has found.

The crashes resulted in at least 19 deaths and scores of injuries over the last decade, records show. Federal regulators say that is far more than any other automaker has experienced.

Owner complaints helped trigger at least eight investigations into sudden acceleration in Toyota and Lexus vehicles by the National Highway Traffic Safety Administration in the last seven years. Toyota Motor Corp. recalled fewer than 85,000 vehicles in response to two of those probes, and the federal agency closed six other cases without finding a defect.

Precise

With a precise document we can predict/calculate that it is safe and fit for use!

Abstract State
Machines

System Boundary
Bank,
Requirements

Input/Output

Use Cases

Function Tables
What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of “old”

Tabular
Specifications

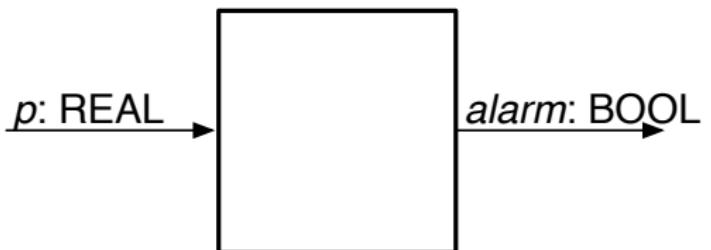
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Specification of the Pressure example

Consider a device with a monitored variable pressure $p : \mathbb{R}$ and controlled variable $alarm$ as the output as shown in the figure.



	$alarm$
$p \leq normal$	<i>False</i>
$normal < p < hi$	No Change
$p \geq hi$	<i>True</i>

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications

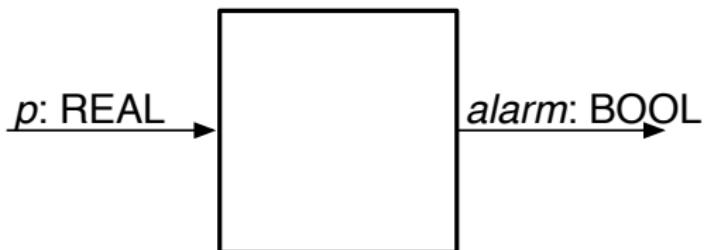
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Specification of the Pressure example

Consider a device with a monitored variable pressure $p : \mathbb{R}$ and controlled variable $alarm$ as the output as shown in the figure.



	$alarm$
$p \leq normal$	<i>False</i>
$normal < p < hi$	No Change
$p \geq hi$	<i>True</i>

PVS Time Theory?

How do we specify the pressure example in T-ASM theory?

Abstract State Machines

System Boundary

Bank Requirements

Input/Output

Use Cases

Function Tables

What does "old" mean?

Timing Resolution

Discrete time /

Timed ASM Theory

Meaning of "old"

Tabular Specifications

Revised table

Completeness

Conciseness

Validation

DTIME theory

Bank I/O Vars

Monitored Events

DATATYPE

PVS Function

Table

Validation

Strong invariant

Use Case

Specification of the Pressure example

```

alert: THEORY
BEGIN
    delta: posreal = 1 % TR = 1 seconds
    normal, hi: real
    IMPORTING Time[delta]
    p:      [DTIME -> real]   % Pressure
    alarm: [DTIME -> bool]

    spec(i:DTIME): bool =
        COND
            i = 0 -> NOT alarm(0)
            , i > 0 ->
                COND
                    p(i) <= normal -> alarm(i)=FALSE
                    , normal < p(i) AND p(i) < hi ->
                        alarm(i) = alarm(i-1)
                    , p(i) >= hi       -> alarm(i) = TRUE
                ENDCOND
            ENDCOND

```

**Abstract State
Machines**

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

**Timed ASM
Theory**

Meaning of "old"

**Tabular
Specifications**

Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Specification of the Pressure example

```

alert: THEORY
BEGIN
  delta: posreal = 1 % TR = 1 seconds
  normal, hi: real
  IMPORTING Time[delta]
  p:      [DTIME -> real]   % Pressure
  alarm: [DTIME -> bool]

  spec(i:DTIME): bool =
    COND
      i = 0 -> NOT alarm(0)
      , i > 0 ->
        COND
          p(i) <= normal -> alarm(i)=FALSE
          , normal < p(i) AND p(i) < hi ->
              alarm(i) = alarm(i-1)
          , p(i) >= hi       -> alarm(i) = TRUE
        ENDCOND
    ENDCOND

```

“spec” does not typecheck. Why?

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does “old”
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of “old”

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

Specification of the Pressure example

```

alert: THEORY
BEGIN
  delta: posreal = 1 % TR = 1 seconds
  normal, hi: real
  IMPORTING Time[delta]
  p: [DTIME -> real] % Pressure
  alarm: [DTIME -> bool]

  spec(i:DTIME): bool =
    COND
      i = 0 -> NOT alarm(0)
      , i > 0 -> COND
        p(i) <= normal -> alarm(i)=FALSE
        ,normal < p(i) AND p(i) < hi ->
          alarm(i) = alarm(i-1)
        ,p(i) >= hi -> alarm(i) = TRUE
    ENDCOND
ENDCOND

```

ax: AXIOM normal < hi

Abstract State
Machines

System Boundary

Bank,
Requirements

Input/Output

Use Cases

Function Tables

What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory

Meaning of "old"

Tabular
Specifications

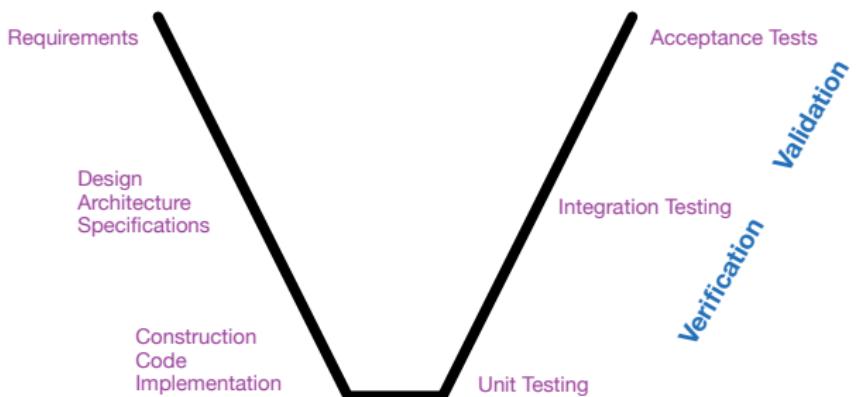
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events

DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case

V&V curve



Abstract State Machines

System Boundary
Bank Requirements

Input/Output

Use Cases

Function Tables
 What does "old" mean?

Timing Resolution
 Discrete time /

Timed ASM Theory
 Meaning of "old"

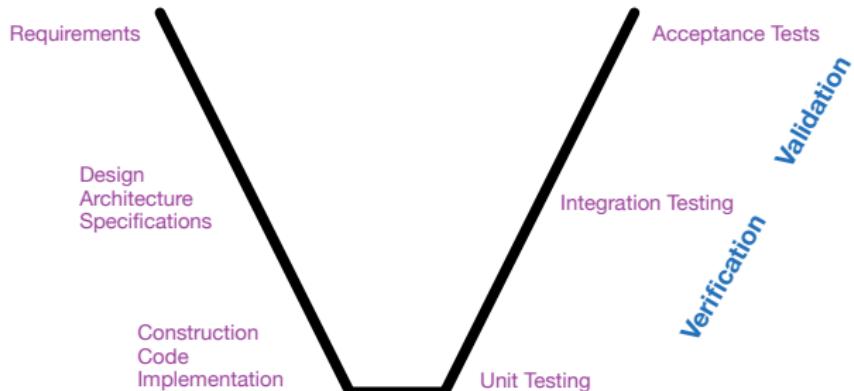
Tabular Specifications
 Revised table
 Completeness
 Conciseness
 Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

V&V curve

Requirements



Validation

- Timed ASM Theory can be used to specify continuous as well as discrete systems



Abstract State Machines

System Boundary
Bank Requirements

Input/Output

Use Cases

Function Tables
What does "old" mean?

Timing Resolution
Discrete time /

Timed ASM Theory
Meaning of "old"

Tabular Specifications

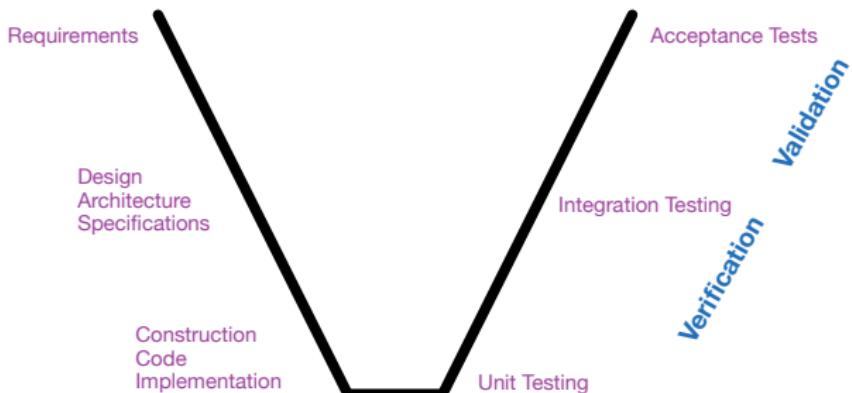
Revised table
Completeness
Conciseness
Validation

DTIME theory

Bank I/O Vars
Monitored Events
DATATYPE
PVS Function Table
Validation
Strong invariant
Use Case

V&V curve

Requirements



Validation

- Timed ASM Theory can be used to specify continuous as well as discrete systems
- Validation means checking that requirements are complete, disjoint and well-defined
- Validation means checking that requirements satisfy (a) Safety Invariants and (b) Use Cases (generalizations of Acceptance Tests)

Abstract State
Machines

System Boundary
Bank
Requirements

Input/Output

Use Cases

Function Tables
What does "old"
mean?

Timing Resolution
Discrete time /

Timed ASM
Theory
Meaning of "old"

Tabular
Specifications
Revised table
Completeness
Conciseness
Validation

DTIME theory
Bank I/O Vars
Monitored Events
DATATYPE
PVS Function
Table
Validation
Strong invariant
Use Case