



Vienna Development Method for Specification Language (VDM-SL)

Formal Methods in Software Engineering



VDM-SL

- Formal Method
- IMB Laboratory Vienna
- System description (functionality performed on data)
- Data \Rightarrow State
- Various Components
 - VDM++ for object-oriented and concurrent systems
 - Specification Language

Data Types

Data Type	DT	Description	Example Values
bool	B	Boolean datatype	false, true
nat	N	natural numbers (including zero)	0, 1, 2, 3, 4, 5 ...
nat1	N1	natural numbers (excluding zero)	1, 2, 3, 4, 5, ...
int	Z	integers	..., -3, -2, -1, 0, 1, 2, 3, ...
rat		rational numbers	a/b, where a and b are integers, b is not 0
real	R	real numbers	...
char	char	characters	A, B, C, ...
token		structureless tokens	...
<A>		the quote type containing the value <A>	...

Examples

□ Types

- `UserId = nat`
- `CGPA = real`

□ Invariants

- `UserId = nat`
- `inv uid == uid <= 9999`
- `CGPA = real`
- `CGPA <= 4`

Basic Type Constructors

Construct	Description
$T_1 \mid T_2 \mid \dots \mid T_n$	Union of types T_1, \dots, T_n
$T_1 * T_2 * \dots * T_n$	Cartesian product of types T_1, \dots, T_n
$T :: f_1:T_1 \dots f_n:T_n$	Composite (Record) type

Examples

- `SignalColour = <Red> | <Yellow> | <Green>`
- `Subjects = <Formal Methods> | <Algorithms> | <Subject A>`
- `Composit (Field:Type)`
 - `T :: f1:A1
 f2:A2
 ...
 fn:An`
 - `Date :: day:nat1
 month:nat1
 year:nat
 inv mk_Date(d,m,y) == d <=31 and m<=12`

Composit (Field:Type)

- `T :: f1:A1
 f2:A2
 ⋮
 fn:An`
- `Date :: day:nat1
 month:nat1
 year:nat
inv mk_Date(d,m,y) == d <=31 and m<=12`
- `mk_Date(22,2,2023)`
 - `day=22`
 - `month=2`
 - `year=2023`
- `mk_Date(22,2,2023).month =>2`



Case Study: Incubator

- Temperature (-10,+10)
- Increase()
- Decrease()
- View/display()



System State



- state <SystemName> of
- <StateName> : <Type> end

- Example

- **state** IncubatorSystem **of**
 - temp : int/Z
 - **end**



Operations



- the operation header
- ext wr: the external clause
- pre: the precondition
- post: the postcondition
- Example
 - increment()
 - ext wr temp: Z
 - pre temp < 10
 - post temp = temp + 1



Example

- ❑ decrement()
- ❑ ext wr temp: Z
- ❑ pre temp > -10
- ❑ post temp = temp - 1



Example 2: Quiz Evaluation



□ As a home task



Constants

- Constant_Name Type : Value
- MIN Z:-10
- MAX Z:10
- Pre Condition
- Temp<MIN

Functions: Explicit

▣ Signature

▣ $\text{Function_name Inputs_types} \rightarrow \text{Output_type}$

▣ Definition

▣ $\text{Function_name(inputs)} \Delta \text{output}$

▣ $\text{Add}: R \times R \rightarrow R$

▣ $\text{Add}(x,y) \Delta x + y$



Functions: implicit

- ▣ Pre conditions
- ▣ Post conditions
- ▣ $\text{Add}(x:\mathbb{R}, y:\mathbb{R})\ z:\mathbb{R}$
- ▣ Pre True
- ▣ Post $z = x + y$

Example

- ▣ $\text{Abs}(x:\mathbb{Z})\ r:\mathbb{N}$
- ▣ Pre True
- ▣ Post $x < 0 \wedge r = -x \vee x \geq 0 \wedge r = x$

- ▣ $x < 0 \wedge r = -x$

- ▣ *OR*

- ▣ $x \geq 0 \wedge r = x$

Example: Explicit

▣ $\text{Abs}: \mathbb{Z} \rightarrow \mathbb{N}$

▣ $\text{Abs}(x) \Delta$

▣ if $x < 0$

▣ Then $-x$

▣ Else x

Example: Explicit

- ▣ $\text{factorial}: \mathbb{N} \rightarrow \mathbb{N}$
- ▣ $\text{factorial}(n) \Delta$
 - ▣ if $n=0$
 - ▣ then 1
 - ▣ else $n * \text{factorial}(n-1)$

State Invariant

- ▣ Global Constraint
- ▣ $\text{Inv}: \text{State} \rightarrow \text{Boolean}$
- ▣ $\text{inv mk-IncubatorMonitor}(t) \Delta \text{MIN} \leq t \leq \text{MAX}$




Incubator Controller

- ❑ requestedTemp : Integer
- ❑ actualTemp : Integer
- ❑ setInitialTemp(Integer)
- ❑ requestChange(Integer) : Signal
- ❑ increment() : Signal
- ❑ decrement() : Signal
- ❑ getRequestedTemp() : Integer
- ❑ getActualTemp() : Integer



Signal

- <<enumeration>>
 - Signal
 - INCREASE
 - DECREASE
 - DO_NOTHING
- 

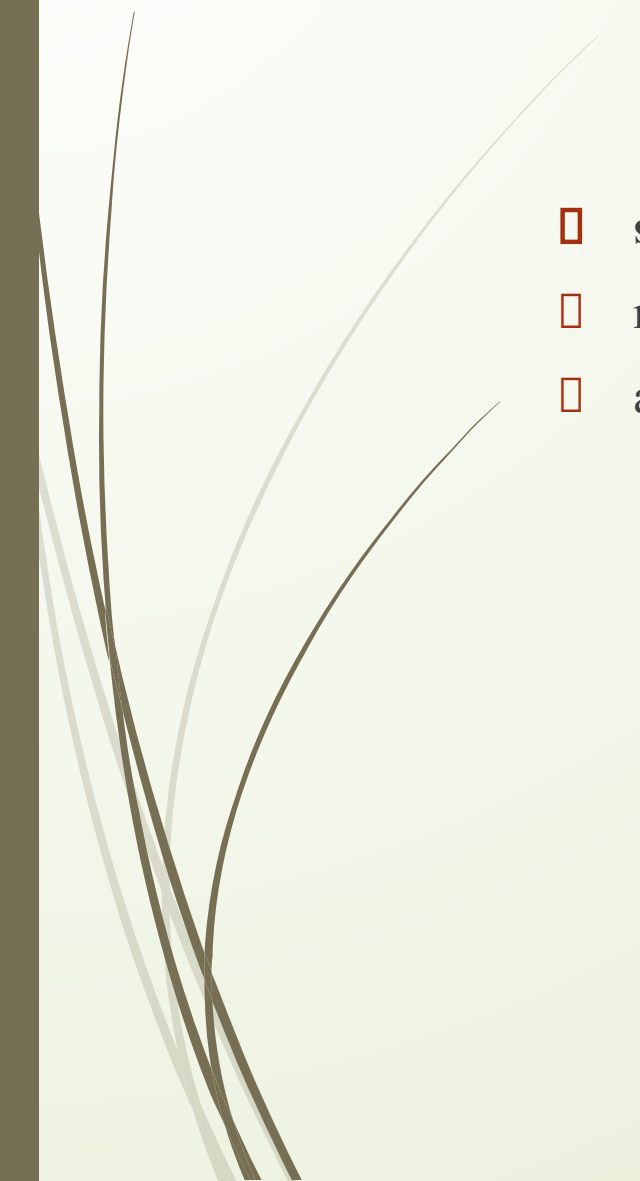


□ types

□ Signal = <INCREASE>|<DECREASE>|<DO_NOTHING>



State

- **state** IncubatorController of
 - requestedTemp : [Z]
 - actualTemp : [Z]
- 


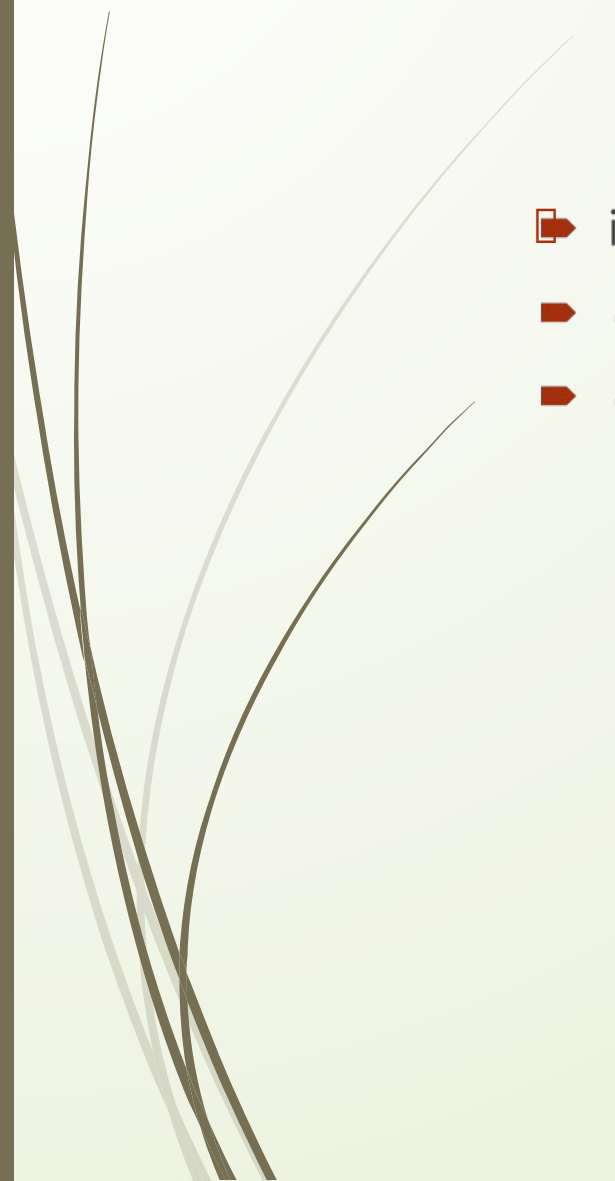
Constructor


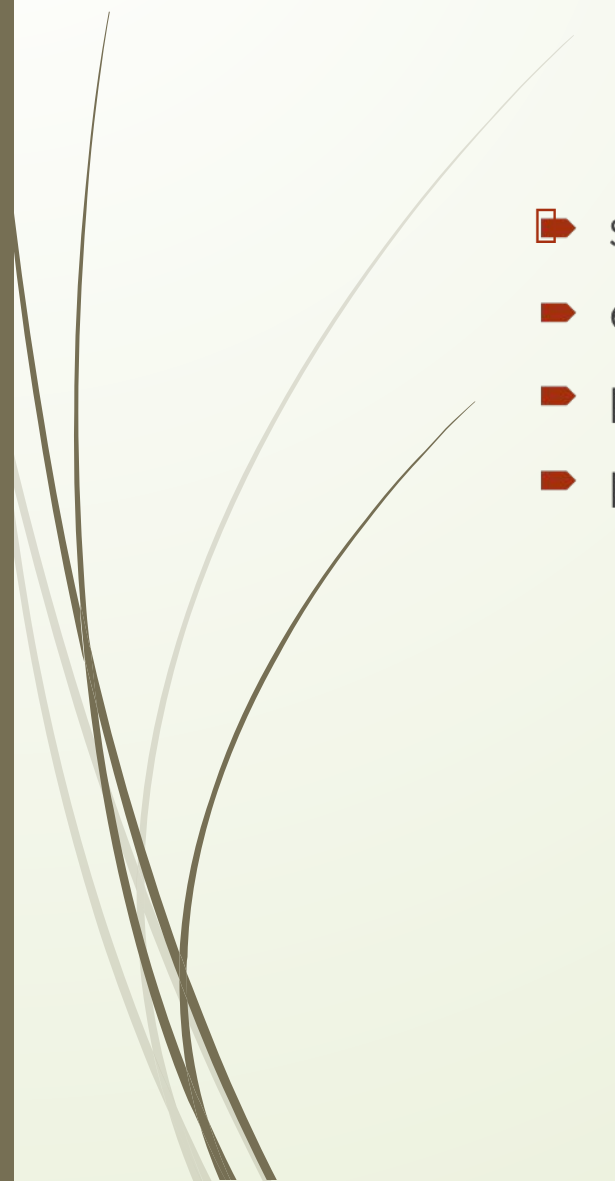
- ▣ $\text{inv mk-IncubatorController } (r, a) \Delta$
 - ▣ $MIN \leq r \leq MAX \vee r = \text{nil} \wedge$
 - ▣ $MIN \leq a \leq MAX \vee a = \text{nil}$
- ▣ $\text{init mk-IncubatorController } (r, a) \Delta r = \text{nil} \wedge a = \text{nil}$



Is Range

- ▣ `inRange(val: Z) result: B`
- ▣ pre TRUE
- ▣ post *result* $\Leftrightarrow MIN \leq val \leq MAX$

- 
- 
- ▣ $\text{inv mk-IncubatorController}(r, a) \Delta$
 - ▣ $(\text{inRange}(r) \vee r = \text{nil}) \wedge$
 - ▣ $(\text{inRange}(a) \vee a = \text{nil})$

- 
- 
- ▣ `setInitialTemp(templn : Z)`
 - ▣ `ext wr actualTemp : [Z]`
 - ▣ `pre inRange(templn) v actualTemp = nil`
 - ▣ `post actualTemp = templn`

Change

- requestChange(tempIn : Z) signalOut : Signal
- Ext
 - wr requestedTemp : [Z]
 - rd actualTemp : [Z]
- pre inRange(tempIn) \wedge actualTemp \neq nil
- Post
 - requestedTemp tempIn \wedge
 - (tempIn > actualTemp \wedge signalOut = <INCREASE>
 - \vee
 - tempIn < actualTemp \wedge signalOut = <DECREASE>
 - \vee
 - tempIn = actualTemp \wedge signalOut = <DO_NOTHING>)

increment

- increment () signalOut : Signal
- Ext
 - rd requestedTemp : [Z]
 - wr actualTemp : [Z]
- Pre
 - $\text{actualTemp} < \text{requestedTemp} \wedge$
 - $\text{actualTemp} \neq \text{nil} \wedge \text{requestedTemp} \neq \text{nil}$
- Post
 - $\text{actualTemp} = \text{actualTemp} + 1 \wedge$
 - $(\text{actualTemp} < \text{requestedTemp} \wedge$
 - $\text{signalOut} = \text{<INCREASE>} \vee$
 - $\text{actualTemp} = \text{requestedTemp} \wedge$
 - $\text{signalOut} = \text{<DO_NOTHING>})$



Comments



□ -- comments



Complete System

- Book FORMAL SOFTWARE DEVELOPMENT Chapter 3
 - Section 3.17
 - Page 41 to 43