

## Data Refinement and Schemas

A partial operation

$$\text{Recip} \triangleq [r, r' : \mathbb{R} \mid r \neq 0 \wedge r' = 1/r]$$

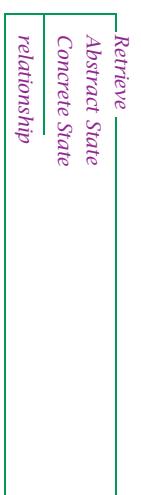
### Meaning

$$\overline{\dot{\{ \text{Recip} \bullet \theta S \mapsto \theta S' \}}}$$

$$\{r, r' : \mathbb{R}^\perp \mid$$

$$(r \neq 0 \wedge r \neq \perp \wedge r' = 1/r) \vee r = 0 \vee r = \perp \bullet \\ \theta S \mapsto \theta S'\}$$

### Retrieve relation



### Forwards simulation

$$r = \{R \bullet \theta A \mapsto \theta C\}$$

$$ao = \{AO \bullet (\theta A, i?) \mapsto (\theta A', o!)\}$$

$$co = \{CO \bullet (\theta C, i?) \mapsto (\theta C', o!)\}$$

$$ai = \{AI \bullet \theta A'\}$$

$$ci = \{CI \bullet \theta C'\}$$

$$ci \subseteq ai \circ r$$

$$\Leftrightarrow \forall c : C \bullet c \in ci \Rightarrow c \in ai \circ r \quad [\text{by property of } \subseteq]$$

$$\Leftrightarrow \forall C \bullet \theta C \in ci \Rightarrow \theta C \in ai \circ r \quad [\text{by schema calculus}]$$

$$\Leftrightarrow \forall C \bullet \theta C \in ci \Rightarrow \exists A \bullet \theta A \in ai \wedge \theta A \mapsto \theta C \in r \quad [\text{by property of } \circ]$$

$$\Leftrightarrow \forall C \bullet \theta C \in \{CI \bullet \theta C'\} \Rightarrow \exists A \bullet \theta A \in \{AI \bullet \theta A'\} \wedge \theta A \mapsto \theta C \in \{R \bullet \theta A \mapsto \theta C\} \quad [\text{by definition}]$$

$$\Leftrightarrow \forall C \bullet CI \Rightarrow \theta A \mapsto \theta C \in \{R \bullet \theta A \mapsto \theta C\} \quad [\text{by comprehension}]$$

$$\exists A \bullet AI \wedge R$$

### Rules for forwards simulation

$$\begin{aligned}
 \text{F-init} & \quad \forall C \bullet CI \Rightarrow \exists A \bullet AI \wedge R' \\
 \text{F-corr} & \quad \forall A_i \bullet C; C' \bullet \\
 & \quad \text{pre } AO \wedge R \wedge CO \Rightarrow \\
 & \quad \exists A' \bullet AO \wedge R \\
 \forall A_i \bullet C \bullet & \\
 & \quad \text{pre } AO \wedge R \Rightarrow \text{pre } CO
 \end{aligned}$$

### Installation

$$\forall C' \mid CI \bullet (\exists A' \mid AI \bullet R')$$

### Preconditions

$$\forall A; C \mid \text{pre } AO \wedge R' \bullet \text{pre } CO$$

### Correctness

$$\forall A_i \bullet C; C' \mid \text{pre } AO \wedge R \wedge CO \bullet (\exists A' \bullet R' \wedge AO)$$

### Why refine?

- Implementation: the design is nearer to the level of the programming language;
- Efficiency: the space/time trade-off.

### A building entry system

[Staff]

| maxentry :  $\mathbb{N}$

### Abstract system

$\text{ASystem} \triangleq [ s : \mathbb{P} \text{Staff} \mid \#s \leq \text{maxentry} ]$

$\text{ASystemInit} \triangleq [ \text{ASystem}' \mid s' = \emptyset ]$

$\begin{array}{l} \text{AEnterBuilding} \\ \Delta \text{ASystem} \\ p? : \text{Staff} \\ \#s < \text{maxentry} \\ p? \notin s \\ s' = s \cup \{p?\} \end{array}$

### Concrete system

$\text{CSystem} \triangleq [ l : \text{iseq Staff} \mid \#l \leq \text{maxentry} ]$

$\text{CSystemInit} \triangleq [ \text{CSystem}' \mid l' = \langle \rangle ]$

$\begin{array}{l} \text{CEnterBuilding} \\ \Delta \text{CSystem} \\ p? : \text{Staff} \\ \#l < \text{maxentry} \\ p? \notin \text{ran } l \\ l' = l \cup \langle p? \rangle \end{array}$

### Refinement

#### Initialisation

$\begin{array}{l} \text{ListRetrieveSet} \\ \Delta \text{ASystem} \\ \text{CSystem} \\ s = \text{ran } l \end{array}$

## Operations

$$\begin{aligned} & \forall ASystem; CSystem \mid \text{pre } AEnterBuilding \wedge \text{ListRetrieveSet}' \bullet \\ & \quad \text{pre } CEnterBuilding \\ & \forall ASystem; CSystem; CSystem' \mid \\ & \quad \text{pre } ALeaveBuilding \wedge \text{ListRetrieveSet} \wedge \text{CleaveBuilding} \bullet \\ & \quad (\exists ASystem' \bullet \text{ListRetrieveSet}' \wedge ALeaveBuilding) \\ & \quad (\exists ASystem' \bullet \text{ListRetrieveSet}' \wedge AEnterBuilding) \end{aligned}$$

## A mean machine

$$AMemory' \hat{=} [ s : \text{seq } \mathbb{N} ]$$

$$AMemoryInit \hat{=} [ AMemory' \mid s' = \langle \rangle ]$$

$$\begin{array}{c} \text{AMean} \\ \Xi \text{AMemory} \\ m! : \mathbb{R} \\ \hline s \neq \langle \rangle \\ m! = \frac{\sum_{i=1}^{\#s} (s_i)}{\#s} \end{array}$$

## Specification

Operation	Precondition
AMemoryInit	true
AEnter	true
AMean	$s \neq \langle \rangle$

*CENTER* $\Delta CMemory$  $n? : \mathbb{N}$  $sum' = sum + n?$  $size' = size + 1$ **Design***CMean* $\exists CMemory$  $m! : \mathbb{R}$  $size \neq 0$  $m! = \frac{sum}{size}$ 

Operation

true

*CENTER*

true

*CMean* $size \neq 0$ **Retrieve relation***SumSizeRetrieve* $AMemory$  $CMemory$ 

$$sum = \sum_{i=1}^{\#s} (s_i)$$

 $size = \#s$ **Initialisation**

$$\forall CMemory' \mid CMemoryInit \bullet$$

$$(\exists AMemory' \mid AMemoryInit \bullet SumSizeRetrieve')$$
**Operations** $\forall AMemory; CMemory \mid$  $\text{pre } AEnter \wedge \text{SumSizeRetrieve}' \bullet \text{pre } CENTER$  $\forall AMemory; CMemory'; CMemory' \mid$  $\text{pre } AEnter \wedge \text{SumSizeRetrieve}' \wedge CENTER \bullet$  $(\exists AMemory' \bullet \text{SumSizeRetrieve}' \wedge AEnter)$  $\forall AMemory; CMemory \mid$  $\text{pre } AMemory \wedge \text{SumSizeRetrieve}' \bullet \text{pre } CMean$  $\forall AMemory; CMemory; CMemory' \mid$  $\text{pre } AMemory \wedge \text{SumSizeRetrieve}' \wedge CMean \bullet$  $(\exists AMemory' \bullet \text{SumSizeRetrieve}' \wedge AMemory)$ **Abstract program**var  $sum, size : \mathbb{N} \bullet$ 

..

proc  $enter$  (val  $n? : \mathbb{N}$ );  
 $sum, size : [ true, sum' = sum + n? \wedge size' = size + 1 ]$ ;proc  $mean$  (res  $m! : \mathbb{R}$ );  
 $sum, size : [ true, sum' = sum + n? \wedge size' = size + 1 ]$ ; $m! : [ size \neq 0, m! = sum / size ]$

**Code**

```

PROGRAM MeanMachine(input,output);
VAR
  n,sum,size: 0..maxint;
  m: real;
PROC Enter(n: 0..maxint);
BEGIN
  sum := sum + n;
  size := size + 1
END;
PROC Mean(VAR m: real);
BEGIN
  m := sum / size
END;

```

```

BEGIN
  sum := 0;
  size := 0;
  WHILE NOT eof DO
    BEGIN
      read(n);
      Enter(n)
    END;
    Mean(m);
    write(m)
  END.

```

**A better way?**

*MeanMachine*  
 $\alpha, \omega : \text{seq } \mathbb{N}$

$$\alpha \neq \langle \rangle$$

$$\omega = \left\langle \frac{\sum_{i=1}^{\#\alpha} (\alpha^i)}{\#\alpha} \right\rangle$$

**Dictionary**

*ADict*  $\cong [ad : \wp Word]$

*CDict<sub>1</sub>*  
 $cd_1 : \text{iseq Word}$

$\forall i, j : \text{dom } cd_1 \mid i \leq j \bullet (cd_1.i) \leq w (cd_1.j)$

*CDict<sub>2</sub>*  
 $cd_2 : \text{iseq } (\wp Word)$

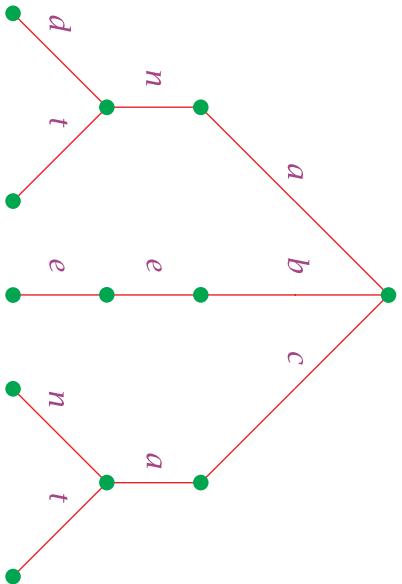
$\forall i : \text{dom } cd_2 \bullet \forall w : (cd_2.i) \bullet \#w = i$

## Word trees

```
WordTree ::= tree<(Letter →1 WordTree)> |  
          treeNode<(Letter → WordTree)>  
  
CDict3 ≡ [ cd3 : WordTree ]
```

## Example

```
tree{a → tree{n → tree{d → treeNode∅, t → treeNode∅}}},  
b → tree{e → tree{e → treeNode∅}},  
c → tree{a → tree{n → treeNode∅, t → treeNode∅}}
```



## Example

```
tree{t → tree{i → tree{n → treeNode∅, y → treeNode∅}}}
```

## Initialisation

### Preconditions

$\forall C'; A' \mid CI \wedge R' \bullet AI$

### Rules for backwards simulation

#### Correctness

$$\begin{aligned} \forall C \mid & (\forall A \mid R \bullet \text{pre } AO) \bullet \\ & (\forall A'; C' \mid CO \wedge R' \bullet \\ & (\exists A \bullet R \wedge AO)) \\ \forall C \bullet & (\forall A \mid R \bullet \text{pre } AO) \Rightarrow \\ & \text{pre } CO \end{aligned}$$

### Phoenix

[T]

Booked ::= yes | no

Phoenix  
 $p_{pool} : \wp T$   
 $bkd : \text{Booked}$

### Phoenix operations

$$\begin{aligned} p_{book} & \Delta_{\text{Phoenix}} \\ bkd & = \text{no} \\ p_{pool} & \neq \emptyset \\ bkd' & = \text{yes} \\ p_{pool}' & = p_{pool} \end{aligned}$$

### Apollo

 $TT ::= \text{null} \mid \text{ticket}(T)$ 

$$\begin{aligned} P_{Arrive} & \Delta_{\text{Phoenix}} \\ t! : T & \\ bkd & = \text{yes} \\ p_{pool} & \neq \emptyset \\ bkd' & = \text{no} \\ t! \in p_{pool} & \\ p_{pool}' & = p_{pool} \setminus \{t!\} \end{aligned}$$

## Apollo operations

$\Delta\text{Book}$	$\Delta\text{Apollo}$
$tkt = \text{null}$	$tkt \neq \text{null}$
$apool \neq \emptyset$	$tkt' = \text{null}$
$tkt' \neq \text{null}$	$t! = \text{ticket}^\sim tkt$
$\text{ticket}^\sim tkt' \in apool$	$apool' = apool \setminus \{\text{ticket}^\sim tkt'\}$

## Retrieve relation

$\Delta\text{Phoenix}$	$\Delta\text{Apollo}$
$bkd = \text{no} \Rightarrow tkt = \text{null} \wedge ppool = apool$	$tkt \neq \text{null}$

$bkd = \text{yes} \Rightarrow$	$tkt' = \text{null}$
$tkt \neq \text{null}$	$t! = \text{ticket}^\sim tkt$
$\wedge$	$apool = apool \cup \{\text{ticket}^\sim tkt\}$

```
pre .ARRIVE ∧ ApolloPhoenixRetr ∧ PARRIVE ⊢
∃ Apollo' • ApolloPhoenixRetr' ∧ ARRIVE
```

## Mastermind

One player chooses a code of six coloured pegs, the other tries to guess what it is, but when is the choice made?

```
 $t! \in apool \cup \{\text{ticket}^\sim tkt\}$ 
 $\nRightarrow$ 
 $t! = \text{ticket}^\sim tkt$ 
```

## Vending machine

### Choosing a drink

```

YesNo ::= yes | no
Digits == 0 .. 9
seq3[X] == { s : seq X | #s = 3 }

VMSpec ≜ [ busy, vend : YesNo ]

```

$\text{VMSpecInit} \triangleq [ \text{VMSpec}' \mid \text{busy}' = \text{vend}' = \text{no} ]$

## Completing a transaction

### Design

```

VendSpec
ΔVMSpec
ol : YesNo
busy' = no
ol = vend

```

```

FirstPunch
ΔVMDesign
d? : Digit
digits = 0
digits' = 1

NextPunch
ΔVMDesign
d? : Digit
(0 < digits < 3  $\wedge$  digits' = digits + 1)  $\vee$ 
(digits = 0  $\wedge$  digits' = digits)

```

## Vending machine

```

Choose
ΔVMSpec
i? : seq3 Digit
busy = no
busy' = yes

```

## Proof opportunities

$\text{VMSpecInit}$  is refined by  $\text{VMDesignInit}$

$\text{Choose}$  is refined by  $\text{FirstPunch}$

$\exists \text{VMSpec}$  is refined by  $\text{NextPunch}$

$\text{VendSpec}$  is refined by  $\text{VendDesign}$

## Different inputs and outputs

$\text{RetrieveVM}$

$\text{VMSpec}$

$\text{VMDesign}$

$\text{busy} = \text{no} \Leftrightarrow \text{digits} = 0$

## Forwards simulation

$$\begin{aligned} & \forall \text{VMSpec}; \text{VMDesign}; \text{VMDesign}' \mid \\ & \quad \text{pre } \text{Choose} \wedge \text{RetrieveVM} \wedge \text{FirstPunch} \bullet \\ & \quad \exists \text{VMSpec}' \bullet \text{RetrieveVM}' \wedge \text{Choose} \\ \\ & \text{busy} = \text{no} \wedge \\ & \text{busy} = \text{no} \Leftrightarrow \text{digits} = 0 \wedge \\ & \text{digits} = 0 \wedge \text{digits}' = 1 \bullet \\ & \exists \text{busy}', \text{vend}' : \text{YesNo} \bullet \\ & \text{busy}' = \text{no} \Leftrightarrow \text{digits}' = 0 \wedge \text{busy}' = \text{no} \wedge \text{o}' = \text{vend} \end{aligned}$$

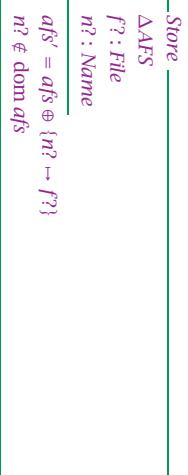
## Not a forwards simulation

$$\begin{aligned} & \forall \text{VMSpec}; \text{VMDesign}; \text{VMDesign}' \mid \\ & \quad \text{pre } \text{VendSpec} \wedge \text{RetrieveVM} \wedge \text{VendDesign} \bullet \\ & \quad \exists \text{VMSpec}' \bullet \text{RetrieveVM}' \wedge \text{VendSpec} \\ \\ & \text{busy} = \text{no} \Leftrightarrow \text{digits} = 0 \wedge \\ & \text{digits}' = 0 \bullet \\ & \exists \text{busy}', \text{vend}' : \text{YesNo} \bullet \\ & \text{busy}' = \text{no} \Leftrightarrow \text{digits}' = 0 \wedge \text{busy}' = \text{no} \wedge \text{o}' = \text{vend} \end{aligned}$$

## Abstract file system

$\text{AFS} \cong [\text{afs} : \text{Name} \leftrightarrow \text{File}]$

$\text{AFSInit} \cong [\text{AFS}' \mid \text{afs}' = \emptyset]$



### Concrete file system

CFS
$cfs : Name \leftrightarrow seq\ Byte$
$tfs : Name \leftrightarrow seq\ Byte$
$\text{dom } cfs \cap \text{dom } tfs = \emptyset$

$$CFSInit \triangleq [ CFS' \mid cfs' = tfs' = \emptyset ]$$

Start
$\Delta CFS$
$n? : Name$
$n? \notin \text{dom } cfs \cup \text{dom } tfs$
$tfs' = tfs \oplus \{n? \mapsto \langle \rangle\}$
$cfs' = cfs$

Next
$\Delta CFS$
$n? : Name$
$b? : Byte$
$n? \in \text{dom } tfs$
$tfs' = tfs \oplus \{n? \mapsto (tfs\ n?) \sim \langle b? \rangle\}$
$cfs' = cfs$

### Retrieve

Stop
$\Delta CFS$
$n? : Name$
$n? \in \text{dom } tfs$
$tfs' = \{n?\} \lhd tfs$
$cfs' = cfs \oplus \{n? \mapsto tfs\ n?\}$

retr_file : seq Byte $\rightarrow$ File
$\text{RetriveACFS}$
$AFS$
$CFS$

## Simulations

(AFS, AFSInit,  $\exists$ AFS,  $\exists$ AFS, Store, Read)  
(CFS, CFSInit, Start, Next, Stop, Read)  
(AFS, AFSInit, Store,  $\exists$ AFS,  $\exists$ AFS, Read)  
(CFS, CFSInit, Start, Next, Stop, Read)

## Summary

- operations as relations
- retrieve relations
- forwards simulation
- backwards simulation