

Data Refinement and Schemas

Using Z

Woodcock & Davies

A partial operation

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

Meaning

$$\overbrace{\{ \text{Recip} \bullet \theta S \mapsto \theta S' \}}^{\bullet}$$

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

$$(r \neq 0 \wedge r \neq \perp \wedge r' = 1/r) \vee r = 0 \vee r = \perp \bullet$$

$$\theta S \mapsto \theta S' \}$$

Ising Z

17-4

Retrieve relation

Retrieve
Abstract State
Concrete State
relationship

Forwards simulation

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

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

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

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

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

17-6

Ising Z

$$ci \subseteq ai \circ r$$

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

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

$$\theta C \in ai \circ r$$

$$\Leftrightarrow \forall C \bullet \theta C \in ci \Rightarrow \quad [\text{by property of } \circ]$$

$$\exists A \bullet \theta A \in ai \wedge \theta A \mapsto \theta C \in r$$

$$\Leftrightarrow \forall C \bullet \theta C \in \{ CI \bullet \theta C' \} \Rightarrow \quad [\text{by definition}]$$

$$\exists A \bullet \theta A \in \{ AI \bullet \theta A' \} \wedge$$

$$\theta A \mapsto \theta C \in \{ R \bullet \theta A \mapsto \theta C \}$$

$$\Leftrightarrow \forall C \bullet CI \Rightarrow \quad [\text{by comprehension}]$$

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

Rules for forwards simulation

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

Jsing Z

17-8

Installation

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

Preconditions

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

Ising Z

17-10

Correctness

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

Why refine?

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

Ising Z

17-12

A building entry system

[*Staff*]

| *maxentry* : \mathbb{N}

Abstract system

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

$ASystemInit \triangleq [ASysm' \mid s' = \emptyset]$

Jsing Z

17-14

$AEnterBuilding$

$\Delta ASysm$

$p? : Staff$

$\#s < maxentry$

$p? \notin s$

$s' = s \cup \{p?\}$

$ALeaveBuilding$

$\Delta ASysm$

$p? : Staff$

$p? \in s$

$s' = s \setminus \{p?\}$

Concrete system

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

$CSystemInit \triangleq [CSystem' \mid I' = \langle \rangle]$

Ising Z

17-16

$CEnterBuilding$

$\Delta CSystem$

$p? : Staff$

$\#I < maxentry$

$p? \notin \text{ran } I$

$I' = I \cup \langle p? \rangle$

$CLeaveBuilding$

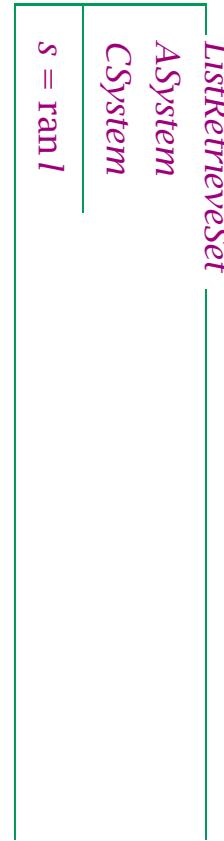
$\Delta CSystem$

$p? : Staff$

$p? \in \text{ran } I$

$I' = I \setminus (Staff \setminus \{p?\})$

Refinement



Ising Z

17-18

Initialisation

$\forall CSysytem' \mid CSysytemInit \bullet$
 $(\exists ASystem' \mid ASystemInit \bullet ListRetrieveSet')$

Operations

- $\forall ASystem; CSysm \mid \text{pre } AEnterBuilding \wedge ListRetrieveSet' \bullet$
- $\text{pre } CEnterBuilding$
- $\forall ASystem; CSysm; CSysm' \mid$
- $\text{pre } AEnterBuilding \wedge ListRetrieveSet \wedge CEnterBuilding \bullet$
- $(\exists ASystem' \bullet ListRetrieveSet' \wedge AEnterBuilding)$

Ising Z

17-20

- $\forall ASystem; CSysm \mid \text{pre } ALeeaveBuilding \wedge ListRetrieveSet' \bullet$
- $\text{pre } CLeaveBuilding$
- $\forall ASystem; CSysm; CSysm' \mid$
- $\text{pre } ALeeaveBuilding \wedge ListRetrieveSet \wedge CLeaveBuilding \bullet$
- $(\exists ASystem' \bullet ListRetrieveSet' \wedge ALeeaveBuilding)$

A mean machine

$AMemory \triangleq [s : \text{seq } \mathbb{N}]$

$AMemoryInit \triangleq [AMemory' \mid s' = \langle \rangle]$

Ising Z

17-22

$AMenter$
 $\Delta AMemory$
 $n? : \mathbb{N}$
 $s' = s \setminus \langle n? \rangle$

$AMean$
 $\Xi AMemory$
 $m! : \mathbb{R}$
 $s \neq \langle \rangle$
 $m! = \frac{\sum_{i=1}^{\#s} (s i)}{\#s}$

Specification

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

Ising Z

17-24

$$CMemory \triangleq [\ sum : \mathbb{N}; \ size : \mathbb{N}]$$

$$InitCMemory \triangleq [\ CMemory' \mid sum' = 0 \wedge size' = 0]$$

CEnter

$\Delta CMemory$

$n? : \mathbb{N}$

$sum' = sum + n?$

$size' = size + 1$

CMean

$\exists CMemory$

$m! : \mathbb{R}$

$size \neq 0$

$m! = \frac{sum}{size}$

Ising Z

17-26

Design

Operation	Precondition
<i>InitCMemory</i>	<i>true</i>
<i>CEnter</i>	<i>true</i>
<i>CMean</i>	$size \neq 0$

Retrieve relation

SumSizeRetrieve —
AMemory
CMemory

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

Ising Z

17-28

Initialisation

$\forall \text{CMemory}' \mid \text{CMemoryInit} \bullet$
 $(\exists \text{AMemory}' \mid \text{AMemoryInit} \bullet \text{SumSizeRetrieve}')$

Operations

$$\begin{aligned}
 & \forall AMemory; CMemory \mid \\
 & \quad \text{pre } AEnter \wedge \text{SumSizeRetrieve}' \bullet \text{pre } CEnter \\
 & \forall AMemory; CMemory; Memory' \mid \\
 & \quad \text{pre } AEnter \wedge \text{SumSizeRetrieve} \wedge CEnter \bullet \\
 & \quad (\exists AMemory' \bullet \text{SumSizeRetrieve}' \wedge AEnter) \\
 & \forall AMemory; CMemory \mid \\
 & \quad \text{pre } AMean \wedge \text{SumSizeRetrieve}' \bullet \text{pre } CMean \\
 & \forall AMemory; CMemory; Memory' \mid \\
 & \quad \text{pre } AMean \wedge \text{SumSizeRetrieve} \wedge CMean \bullet \\
 & \quad (\exists AMemory' \bullet \text{SumSizeRetrieve}' \wedge AMean)
 \end{aligned}$$

Ising Z

17-30

Abstract program

```

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

Ising Z

17-32

A better way?

$$\begin{array}{l} \text{MeanMachine} \\ \alpha, \omega : \text{seq } \mathbb{N} \\ \alpha \neq \langle \rangle \\ \omega = \left\langle \frac{\sum_{i=1}^{\# \alpha} (\alpha i)}{\# \alpha} \right\rangle \end{array}$$

Ising Z

17-34

Dictionary

$$ADict \cong [ad : \mathbb{P} Word]$$

$CDict_1$
$cd_1 : \text{iseq Word}$
$\forall i, j : \text{dom } cd_1 \mid i \leq j \bullet (cd_1 i) \leq_w (cd_1 j)$

Ising Z

17-36

$CDict_2$
$cd_2 : \text{seq}(\mathbb{P} \text{Word})$
$\forall i : \text{dom } cd_2 \bullet \forall w : (cd_2 i) \bullet \#w = i$

Word trees

$WordTree ::= \text{tree}\langle\langle Letter \rightarrow_1 WordTree \rangle\rangle \mid \text{treeNode}\langle\langle Letter \rightarrow WordTree \rangle\rangle$

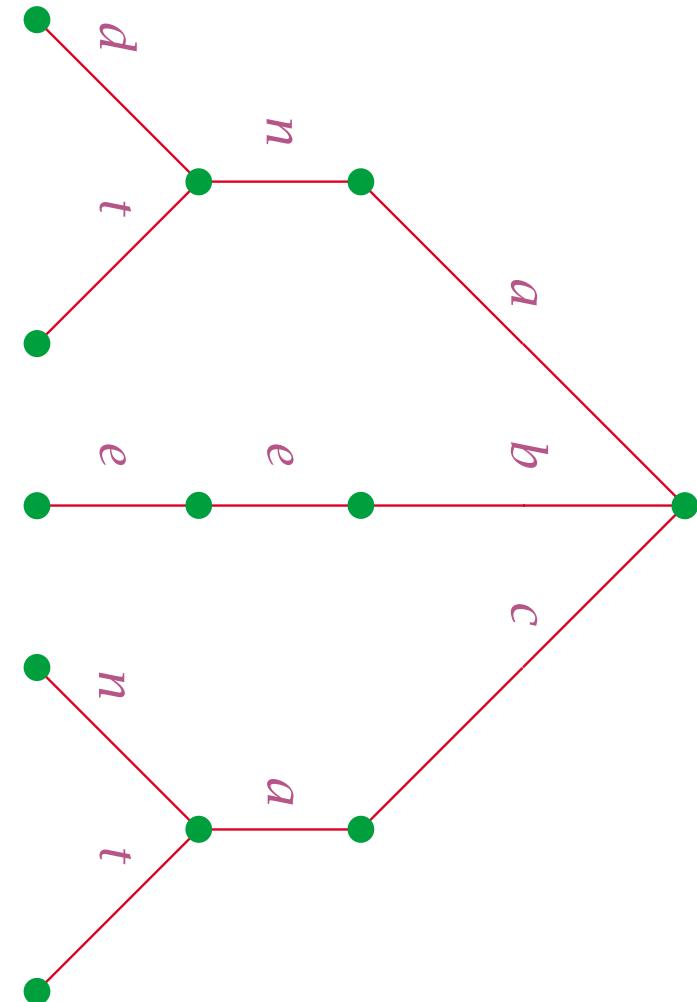
$CDict_3 \triangleq [cd_3 : WordTree]$

Ising Z

17-38

Example

```
tree{a ↦ tree{n ↦ tree{d ↦ treeNode Ø, t ↦ treeNode Ø}}},  
b ↦ tree{e ↦ tree{e ↦ treeNode Ø}},  
c ↦ tree{a ↦ tree{n ↦ treeNode Ø, t ↦ treeNode Ø}}}  
}
```



Ising Z

17-40

Example

```
tree{t => tree{i => tree{n => treeNode{y => treeNode{}}}}}
```

Initialisation

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

Ising Z

17-42

Preconditions

$$\forall C \mid (\forall A \mid R \bullet \text{pre } AO) \bullet \text{pre } CO$$

Correctness

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

Ising Z

17-44

Rules for backwards simulation

B-init	$\forall A; C \bullet CI \wedge R \Rightarrow AI$
B-corr	$\forall C \bullet$
	$(\forall A \mid R \bullet \text{pre } AO) \Rightarrow$
	$\forall A'; C' \bullet CO \wedge R' \Rightarrow$
	$\exists A \bullet R \wedge AO$
$\forall C \bullet$	
$(\forall A \mid R \bullet \text{pre } AO) \Rightarrow$	
pre CO	

Phoenix

[T]

Booked ::= yes | no

Phoenix _____
ppool : $\mathbb{P} T$
bkd : Booked

Ising Z

17-46

Phoenix operations

PBook _____
ΔPhoenix _____
bkd = no
ppool ≠ ∅
bkd' = yes
ppool' = ppool

*P*Arrive
 Δ Phoenix
 $t! : T$

$bkd = yes$
 $ppool \neq \emptyset$
 $bkd' = no$
 $t! \in ppool$
 $ppool' = ppool \setminus \{t!\}$

Ising Z

17-48

Apollo

$TT ::= null \mid ticket\langle\langle T \rangle\rangle$

Apollo
 $apool : \mathbb{P} T$
 $tkt : TT$

Apollo operations

$\Delta Book$
 $\Delta Apollo$
 $tkt = null$
 $apool \neq \emptyset$
 $tkt' \neq null$
 $\text{ticket}^\sim tkt' \in apool$
 $apool' = apool \setminus \{\text{ticket}^\sim tkt'\}$

Ising Z

17-50

$\Delta Arrive$
 $\Delta Apollo$
 $t! : T$
 $tkt \neq null$
 $tkt' = null$
 $t! = \text{ticket}^\sim tkt$
 $apool' = apool$

Retrieve relation

```
ApolloPhoenixRetr
Phoenix
Apollo
[ bkd = no ⇒ tkt = null ∧ ppool = apool
  bkd = yes ⇒
    tkt ≠ null
    ∧
    ppool = apool ∪ {ticket~ tkt} ]
```

Ising Z

17-52

Conjectures

- The Phoenix system is data refined by the Apollo system.
- The Apollo system is data refined by the Phoenix system.

pre $AArrive \wedge ApolloPhoenixRetr \wedge PArrive \vdash$
 $\exists Apollo' \bullet ApolloPhoenixRetr' \wedge AArrive$

$t! \in apool \cup \{ticket^{\sim} tkt\}$
↗
 $t! = ticket^{\sim} tkt$

Ising Z

17-54

Mastermind

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

Vending machine

YesNo ::= *yes* | *no*

Digits ::= 0 .. 9

$\text{seq}_3[X] ::= \{ s : \text{seq } X \mid \#s = 3 \}$

$\text{VMSpec} \triangleq [\text{busy}, \text{vend} : \text{YesNo}]$

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

Ising Z

17-56

Choosing a drink

<i>Choose</i>	
ΔVMSpec	
$i? : \text{seq}_3 \text{ Digit}$	
$\text{busy} = \text{no}$	
$\text{busy}' = \text{yes}$	

Completing a transaction

$VendSpec$
$\Delta VM Spec$
$o! : YesNo$
$busy' = no$
$o! = vend$

Ising Z

17-58

Design

 $VMDesign \triangleq [digits : 0 \dots 3]$ $VMDesignInit \triangleq [VMDesign' \mid digits' = 0]$

FirstPunch

$\Delta \text{VMDesign}$

$d? : \text{Digit}$

$\text{digits} = 0$

$\text{digits}' = 1$

NextPunch

$\Delta \text{VMDesign}$

$d? : \text{Digit}$

$(0 < \text{digits} < 3 \wedge \text{digits}' = \text{digits} + 1) \vee$
 $(\text{digits} = 0 \wedge \text{digits}' = \text{digits})$

Ising Z

17-60

VendDesign

$\Delta \text{VMDesign}$

$o! : \text{YesNo}$

$\text{digits}' = 0$

Proof opportunities

VMSpecInit is refined by *VMDesignInit*

Choose is refined by *FirstPunch*

\exists *VMSpec* is refined by *NextPunch*

VendSpec is refined by *VendDesign*

Ising Z

17-62

Different inputs and outputs

RetrieveVM

VMSpec

VMDesign

busy = no \Leftrightarrow *digits = 0*

Forwards simulation

$$\begin{array}{l} \forall \text{VM}\text{Spec}; \text{VM}\text{Design}; \text{VM}\text{Design}' \mid \\ \quad \text{pre } \text{Choose} \wedge \text{RetrieveVM} \wedge \text{FirstPunch} \bullet \\ \quad \exists \text{VM}\text{Spec}' \bullet \text{RetrieveVM}' \wedge \text{Choose} \end{array}$$

$$\begin{array}{l} \text{busy} = \text{no} \wedge \\ \text{busy}' = \text{no} \Leftrightarrow \text{digits} = 0 \wedge \\ \text{digits} = 0 \wedge \text{digits}' = 1 \bullet \\ \exists \text{busy}', \text{vend}' : \text{YesNo} \bullet \\ \quad \text{busy}' = \text{no} \Leftrightarrow \text{digits}' = 0 \wedge \text{busy}' = \text{yes} \end{array}$$

Ising Z

17-64

Not a forwards simulation

$$\begin{array}{l} \forall \text{VM}\text{Spec}; \text{VM}\text{Design}; \text{VM}\text{Design}' \mid \\ \quad \text{pre } \text{VendSpec} \wedge \text{RetrieveVM} \wedge \text{VendDesign} \bullet \\ \quad \exists \text{VM}\text{Spec}' \bullet \text{RetrieveVM}' \wedge \text{VendSpec} \end{array}$$

$$\begin{array}{l} \text{busy} = \text{no} \Leftrightarrow \text{digits} = 0 \wedge \\ \text{digits}' = 0 \bullet \\ \exists \text{busy}', \text{vend}' : \text{YesNo} \bullet \\ \quad \text{busy}' = \text{no} \Leftrightarrow \text{digits}' = 0 \wedge \text{busy}' = \text{no} \wedge o! = \text{vend} \end{array}$$

Abstract file system

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

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

Ising Z

17-66

<i>Read</i>
$\exists \text{AFS}$
$n? : \text{Name}$
$f! : \text{File}$
$n? \in \text{dom } \text{afs}$
$f! = \text{afs } n?$

<i>Store</i>
ΔAFS
$f? : \text{File}$
$n? : \text{Name}$
$\text{afs}' = \text{afs} \oplus \{n? \mapsto f?\}$
$n? \notin \text{dom } \text{afs}$

Ising Z

17-68

Concrete file system

<i>CFS</i>
<i>cfs</i> : <i>Name</i> $\rightarrow\!\!\! \rightarrow$ seq <i>Byte</i>
<i>dfs</i> : <i>Name</i> $\rightarrow\!\!\! \rightarrow$ seq <i>Byte</i>
dom <i>cfs</i> \cap dom <i>dfs</i> = \emptyset

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

<i>Start</i>	ΔCFS
$n? : Name$	
	$n? \notin \text{dom } cfs \cup \text{dom } tfs$
	$tfs' = tfs \oplus \{n? \mapsto \langle \rangle\}$
	$cfs' = cfs$

Ising Z

17-70

<i>Next</i>	ΔCFS
$n? : Name$	
$b? : Byte$	
	$n? \in \text{dom } tfs$
	$tfs' = tfs \oplus \{n? \mapsto (tfs\ n?) \wedge \langle b?\rangle\}$
	$cfs' = cfs$

Stop
 ΔCFS
 $n? : \text{Name}$
 $n? \in \text{dom } tfs$
 $tfs' = \{n?\} \triangleleft tfs$
 $cfs' = cfs \oplus \{n? \rightarrow tfs\ n?\}$

Ising Z

17-72

Retrieve

$|$
 $\text{retr_file} : \text{seq Byte} \rightarrow \text{File}$

RetrieveACFS
 AFS
 CFS
 $|$
 $afts = cfs \circ \text{retr_file}$

Simulations

(AFS, AFSInit, Ξ AFS, Ξ AFS, Store, Read)

(CFS, CFSInit, Start, Next, Stop, Read)

(AFS, AFSInit, Store, Ξ AFS, Ξ AFS, Read)

(CFS, CFSInit, Start, Next, Stop, Read)

Ising Z

17-74

Summary

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