The Specification Language Z

Bernhard Westfechtel

Lehrstuhl für Informatik III, RWTH Aachen

Characterization

- Formal specification of abstract data types
- Model-oriented specification
- Data types are defined with the help of sets, relations, and functions
- Operations are specified with pre- and postconditions
- Proofs based on logic and set theory
- Specifications may be refined in an evolutionary way

Introduction into the Z Notation

Survey

- Z is based on set theory and predicate logic
- Sets may be defined in the following ways:
  - Extensional: Enumeration of elements
  - Intensional: Specification of a predicate
- Operations on sets: union, intersection, ...
- Base types for sets:
  - Pre-defined type $\mathbb{Z}$ (for integers)
  - User-defined types (abstract data types)
- Type constructors:
  - Power set: $\mathbb{P}\,X$ denotes the set of all subsets of X
  - Cartesian product: $X \times Y$ is the set of all pairs (x, y), where $x \in X$ and $y \in Y$
- Strong typing:
  - All elements of a set must have the same type
  - Operations require operands of the same type

Running example: library

- A library lends books to readers
- For each book, there may be one or more copies
- Only registered users may borrow books from the library
- There is a maximal number of copies which may be issued to one user
- Operations:
  - Stock administration (addition and removal of copies)
  - User administration (registration and deregistration of users)
  - Issue (lending and returning of book copies)

Examples of base types and constructed types

$[Book, Copy, Reader]$	User-defined base types for books, copies, and readers
$\mathbb{Z}$	Set of integers
$\mathbb{P}\ \mathbb{Z}$	Set of all subsets of integers
$\mathbb{F}\ \mathbb{Z}$	Set of all finite subsets of $\mathbb{Z}$
$Book \times Copy$	Set of all pairs of books and copies

Examples for the definition of sets

$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$	Extensional definition of the set of integers from 1 to 10
$1 .. 10$	Interval notation
$\{n : \mathbb{Z} \mid 1 \leqslant n \land n \leqslant 10\}$	Intensional definition of the set of integers from 1 to 10
$\{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$	Extensional definition of the square numbers 1^2 .. 10^2
$\{n : \mathbb{Z} \mid 1 \leqslant n \land n \leqslant 10 \bullet n^2\}$	Intensional definition of the square numbers 1^2 .. 10^2

Elements of Z specifications

$readers : \mathbb{F}\ Reader$ $shelved : \mathbb{F}\ Copy$ $stock : \mathbb{F}\ (Copy \times Book)$ $issued : \mathbb{F}\ (Copy \times Reader)$ $max : \mathbb{Z}$	Variable declarations
$max \geqslant 0$ $\#stock \leqslant max$	Predicates
$Stock ==$ $\{s : \mathbb{F}\ (Copy \times Book) \mid$ $\forall\ c : Copy; b_1, b_2 : Book \bullet$ $(c, b_1) \in s \wedge (c, b_2) \in s \Rightarrow b_1 = b_2\}$ $\mathbb{N} == \{n : \mathbb{Z} \mid n \geqslant 0\}$ $\mathbb{N}_1 == \{n : \mathbb{Z} \mid n > 0\}$	Constant definitions

Type concept

- Types are maximal sets (e.g., $\mathbb{Z}$)
- Predicates for restricting these sets do not modify the type (e.g., $\mathbb{N}$ does not define a new type)
- Sets constrained by predicates may be used in variable declarations
  - » Example:
    $max : \mathbb{N}$ stands for $max : \mathbb{Z}; max \geqslant 0$
- Strong typing: The operands of an operator (e.g., $\cup$) must have the same type

Notations for quantification and sets

$\forall\ Decs \bullet Pred$	$Pred$ holds for all objects in $Decs$
$\forall\ Decs \mid Constr \bullet Pred =$ $\forall\ Decs \bullet Constr \Rightarrow Pred$	$Pred$ holds for all objects in $Decs$ meeting the constraint $Constr$
$\exists\ Decs \bullet Pred$	There is an object in $Decs$ which meets the predicate $Pred$
$\exists\ Decs \mid Constr \bullet Pred =$ $\exists\ Decs \bullet Constr \wedge Pred$	There is an object in $Decs$ which meets both the constraint $Constr$ and the predicate $Pred$
$\{Decs \mid Pred\}$	Set of all objects in $Decs$ which meet the predicate $Pred$
$\{Decs \mid Pred \bullet Expr\}$, e.g. $\{n : \mathbb{Z} \mid 1 \leqslant n \wedge n \leqslant 10 \bullet n^2\}$	Set of all values of all expressions $Expr$, where variables range over objects from $Decs$ satisfying the predicate $Pred$

Definition of enumeration types

$BookKind ::= hardcover \mid paperback$

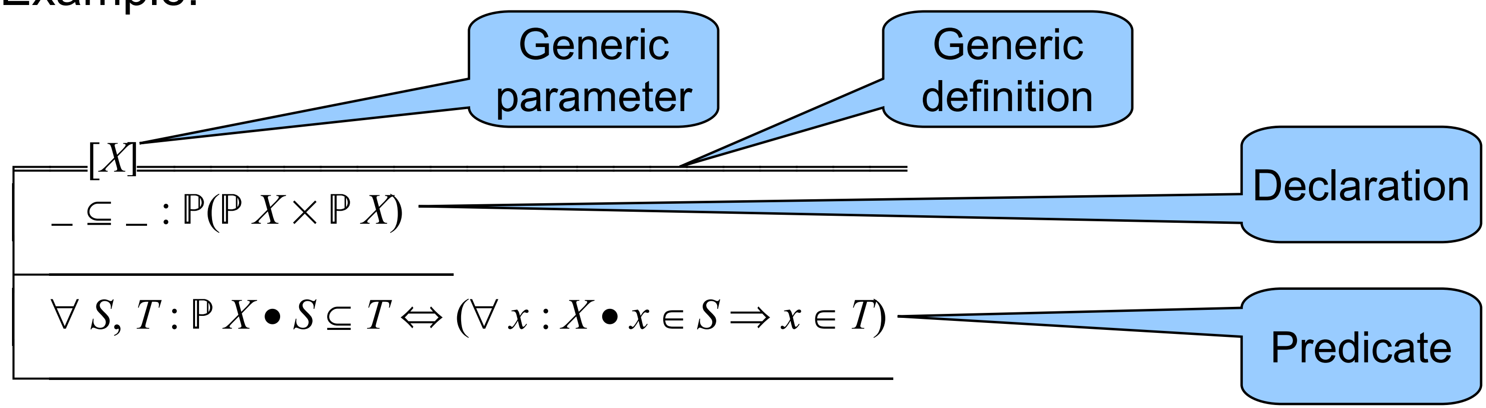
stands for

$[BookKind]$

$hardcover, paperback : BookKind$
$hardcover \neq paperback$
$\forall\, bk : BookKind \bullet bk = hardcover \lor bk = paperback$

Generic definitions

- There are polymorphic operators, which may be applied to operands of different types
- Such operators may be defined as generic
- Unconstrained genericity: any type may replace a generic parameter
- Example:



$$[X]$$

$$\_ \subseteq \_ : \mathbb{P}(\mathbb{P}\, X \times \mathbb{P}\, X)$$

$$\forall\, S, T : \mathbb{P}\, X \bullet S \subseteq T \Leftrightarrow (\forall\, x : X \bullet x \in S \Rightarrow x \in T)$$

(Binary) relations

$X \leftrightarrow Y = \mathbb{P}(X \times Y)$	Relation between X and Y
$x \mapsto y = (x, y)$	Pairs
$\text{dom } R = \{x : X \mid \exists y : Y \bullet x \mapsto y \in R\}$	Domain
$\text{ran } R = \{y : Y \mid \exists x : X \bullet x \mapsto y \in R\}$	Range
$S \lhd R = \{x : X; y : Y \mid x \in S \wedge x \mapsto y \in R \bullet x \mapsto y\}$	Domain restriction
$R \rhd T = \{x : X; y : Y \mid y \in T \wedge x \mapsto y \in R \bullet x \mapsto y\}$	Range restriction
$S \unlhd R = \{x : X; y : Y \mid x \notin S \wedge x \mapsto y \in R \bullet x \mapsto y\}$	Domain subtraction
$R \unrhd T = \{x : X; y : Y \mid y \notin T \wedge x \mapsto y \in R \bullet x \mapsto y\}$	Range subtraction
$R^{-1} = \{x : X; y : Y \mid x \mapsto y \in R \bullet y \mapsto x\}$	Inverse relation
$R \mathbin{;} S =$ $\{x : X; y : Y; z : Z \mid x \mapsto y \in R \wedge y \mapsto z \in S \bullet x \mapsto z\}$	Composition

Functions (1)

Function		Restrictions		
Type	**Symbol**	**dom f**	**injective**	**ran f**
Partial	$\nrightarrow$	$\subseteq X$		$\subseteq Y$
Total	$\rightarrow$	$= X$		$\subseteq Y$
Partial and injective	$\rightarrowtail\!\!\!\!\!\!+$	$\subseteq X$	+	$\subseteq Y$
Total and injective	$\rightarrowtail$	$= X$	+	$\subseteq Y$
Partial and surjective	$\twoheadrightarrow\!\!\!\!\!\!+$	$\subseteq X$		$= Y$
Total and surjective	$\twoheadrightarrow$	$= X$		$= Y$
Bijective	$\rightarrowtail\!\!\!\!\!\rightarrow$	$= X$	+	$= Y$
Partial and finite	$\mathrel{+\!\!\!+\!\!\!\rightarrow}$	$\subseteq X$		$\subseteq Y$
Partial, finite, injective	$\mathrel{\rightarrowtail\!\!\!\!\!\!\!+\!\!\!+}$	$\subseteq X$	+	$\subseteq Y$

Functions (2)

$f : X \leftrightarrow Y$ is a function $\Leftrightarrow$ $\forall x : X; y : Y; z : Z \mid x \mapsto y \in f \land x \mapsto z \in f \bullet y = z$	Functions are unique relations
$f\,x$	Application of a function f to an argument x
dom, $\lhd$, $\rhd$, $⩤$, $⩥$, f^{-1}, $f \fatsemi g$	Operations which are “inherited” from relations
$\lambda x : X \mid Pred \bullet Term =$ $\{x : X \mid Pred \bullet x \mapsto Term\}$	Lambda notation for the definition of functions
$f \oplus g = ((\mathrm{dom}\ g) ⩤ f) \cup g$	Combination of functions (g wins in case of a conflict)

Sequences

$\langle Reagan, Bush, Clinton, Bush\rangle$	Notation for sequences
$\text{seq}\ X == \{ f : \mathbb{N} \twoheadrightarrow X \mid \text{dom}\ f = 1 \mathrel{..} \# f \}$	Formal definition of sequences
$\langle Reagan, Bush, Clinton, Bush\rangle =$ $\{1 \mapsto Reagan, 2 \mapsto Bush, 3 \mapsto Clinton, 4 \mapsto Bush\}$	Example
$\text{seq}_1\ X == \text{seq}\ X \setminus \{ \langle\ \rangle \}$	Non-empty sequences
$\forall\ s : \text{seq}_1\ X \bullet$ $head\ s = s\ 1 \wedge tail\ s = \lambda\ n : 1 \mathrel{..} \#s - 1 \bullet s\ (n + 1)$	Head and tail of a non-empty sequence
$head\ \langle Reagan, Bush, Clinton, Bush\rangle = Reagan$ $tail\ \langle Reagan, Bush, Clinton, Bush\rangle =$ $\langle Bush, Clinton, Bush\rangle$	Example
$\forall\ s, t : \text{seq}\ X \bullet$ $s \frown t = s \cup \{ n : 1 \mathrel{..} \#t \bullet (n + \#s) \mapsto t\ n\}$	Concatenation
$\langle Reagan, Bush\rangle \frown \langle Clinton, Bush\rangle =$ $\langle Reagan, Bush, Clinton, Bush\rangle$	Example

Schemata

On schemata

- Schemata are specification units
- A schema consists of a set of declarations and a set of (conjunctive) predicates
- Schemata may be combined with the help of several operations, including e.g. schema inclusion, schema conjunction and schema disjunction)
- Data types are specified in a model-oriented way as follows:
  - » There is one schema for defining the representation of the data type (state) and the respective state invariants
  - » For each operation, there is one corresponding schema which defines its input and output behavior as well as the state changes affected by the operation

Schema for the state and its invariants

Name

Library

$stock : Copy \psurj Book$

$issued : Copy \psurj Reader$

$shelved : \mathbb{F}\ Copy$

$readers : \mathbb{F}\ Reader$

Declarations

$shelved \cup \mathrm{dom}\ issued = \mathrm{dom}\ stock$

$shelved \cap \mathrm{dom}\ issued = \varnothing$

$\mathrm{ran}\ issued \subseteq readers$

$\forall\ r : readers \bullet \#(issued \rhd \{ r \}) \leqslant maxloans$

Predicates

Schema for an operation

- An operation is defined by a schema which has to obey certain conventions (i.e., Z does not introduce special-purpose “operation schemata”)
- The operation is not declared explicitly!
- Operation name = Schema name
- Parameter:
  - $x?$: Input parameter
  - $y!$: Output parameter
- States:
  - s : “Before” state of an operation
  - s' : “After” state of an operation
- All declarations and predicates for s and s' must be repeated in the operation schema
- To be introduced: Short-hand notation

Example: Lending a book

Issue — Operation name

$$
\begin{array}{l}
\mathit{stock}, \mathit{stock}' : \mathit{Copy} \nrightarrow \mathit{Book} \\
\mathit{issued}, \mathit{issued}' : \mathit{Copy} \nrightarrow \mathit{Reader} \\
\mathit{shelved}, \mathit{shelved}' : \mathbb{F}\ \mathit{Copy} \\
\mathit{readers}, \mathit{readers}' : \mathbb{F}\ \mathit{Reader} \\
\boldsymbol{c? : \mathit{Copy};\ r? : \mathit{Reader}}
\end{array}
$$

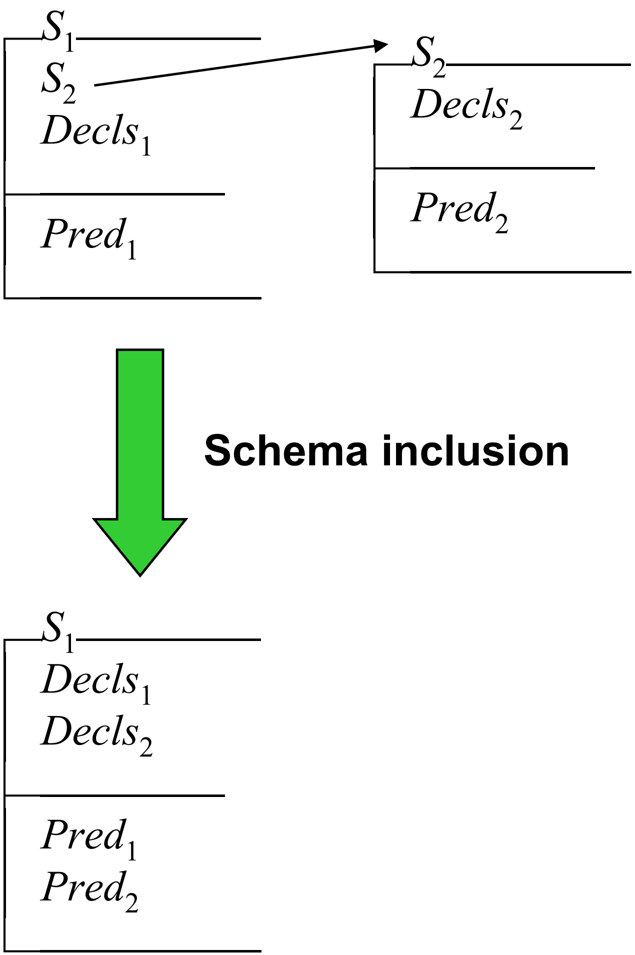
Parameter

$$
\begin{array}{l}
\mathit{shelved} \cup \mathrm{dom}\ \mathit{issued} = \mathrm{dom}\ \mathit{stock} \\
\mathit{shelved}' \cup \mathrm{dom}\ \mathit{issued}' = \mathrm{dom}\ \mathit{stock}' \\
\mathit{shelved} \cap \mathrm{dom}\ \mathit{issued} = \emptyset;\ \mathit{shelved}' \cap \mathrm{dom}\ \mathit{issued}' = \emptyset \\
\mathrm{ran}\ \mathit{issued} \subseteq \mathit{readers};\ \mathrm{ran}\ \mathit{issued}' \subseteq \mathit{readers}' \\
\forall\ r : \mathit{readers} \bullet \#(\mathit{issued} \rhd \{ r \}) \leqslant \mathit{maxloans} \\
\forall\ r : \mathit{readers}' \bullet \#(\mathit{issued}' \rhd \{ r \}) \leqslant \mathit{maxloans} \\
\boldsymbol{c? \in \mathit{shelved};\ r? \in \mathit{readers};\ \#(\mathit{issued} \rhd \{ r \}) < \mathit{maxloans}} \\
\boldsymbol{\mathit{issued}' = \mathit{issued} \oplus \{ c? \mapsto r? \};\ \mathit{stock}' = \mathit{stock};\ \mathit{readers}' = \mathit{readers}} \\
\boldsymbol{\mathit{shelved}' = \mathit{shelved} \setminus \{ c? \}}
\end{array}
$$

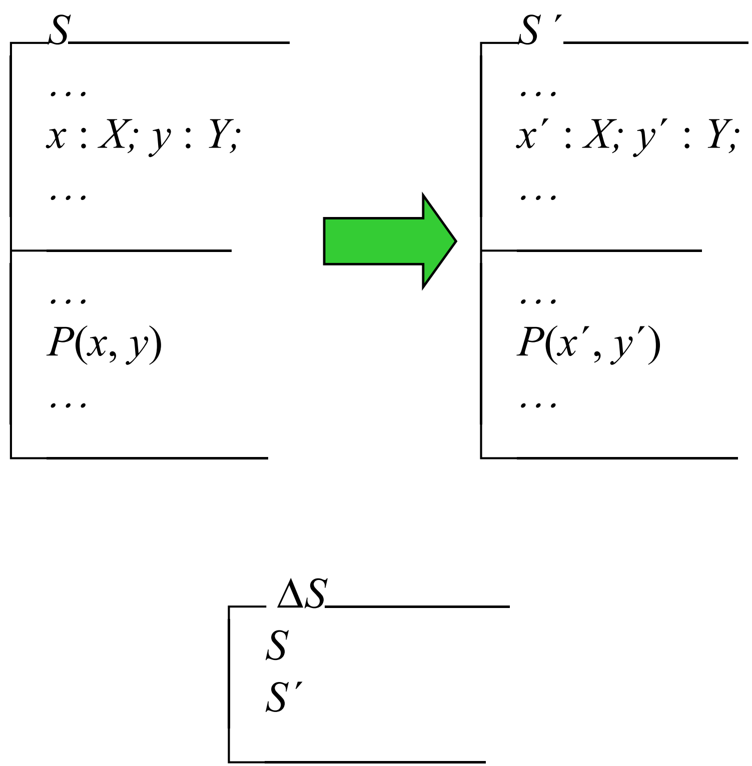
Pre-conditions

Post-conditions

Schema operators (1)



Schema inclusion



Schema decoration and Δ schema

Schema operators (2)

S_1

$Decls_1$

$Pred_1$

S_2

$Decls_2$

$Pred_2$

S_1

$Decls_1 \cup Decls_2$

$Pred_1 \vee Pred_2$

Schema disjunction

$S_1 \vee S_2$

S_1

$Decls_1$

$Pred_1$

S_2

$Decls_2$

$Pred_2$

S_1

$Decls_1 \cup Decls_2$

$Pred_1 \wedge Pred_2$

Schema conjunction

$S_1 \wedge S_2$

Schema operators (3)

Schema composition

- Given: Schemata for two operations Op_1 and Op_2 on the same state $State$
- Schema composition describes the sequential application of Op_1 and Op_2:
  - ⇨ $Op_1[{'}/{''}]$ denotes the schema which is derived from Op_1 by replacing variables v' with v''
  - ⇨ $Op_2[\ /{''}]$ denotes the schema which is derived from Op_2 by replacing variables v with v'
  - ⇨ $Op_1 \fatsemi Op_2 = \exists\, State'' \bullet Op_1[{'}/{''}] \wedge Op_2[\ /{''}]$

Precondition

- Let Op be a schema for an operation on state $State$ with output variables $Outs!$
- pre Op returns the precondition under which Op is applicable:
  - ⇨ $\text{pre } Op = \exists\, State'; Outs! \bullet Op$

Example of a schema inclusion

LibDB

$$stock : Copy \pfun Book$$
$$readers : \mathbb{F}\, Reader$$

LibLoans

$$issued : Copy \pfun Reader$$
$$shelved : \mathbb{F}\, Copy$$

$$shelved \cap \operatorname{dom} issued = \emptyset$$
$$\forall r : readers \bullet \#(issued \rhd \{ r \}) \leqslant maxloans$$

Library

LibDB
LibLoans

$$shelved \cup \operatorname{dom} issued = \operatorname{dom} stock$$
$$\operatorname{ran} issued \subseteq readers$$

Specification of a change operation with a Δ schema

Issue

$\boldsymbol{\Delta Library}$

$c? : Copy;\ r? : Reader$

$c? \in shelved;\ r? \in readers;\ \#(issued \rhd \{ r \}) < maxloans$

$issued' = issued \oplus \{ c? \mapsto r? \};\ stock' = stock;\ readers' = readers$

$shelved' = shelved \setminus \{ c? \}$

Specification of a read operation with a Ξ schema

$$
\begin{array}{|l}
\textbf{\textit{Ξ Library}} \\
\hline
\Delta \mathit{Library} \\
\hline
\mathit{NoChange} \equiv \\
\mathit{issued}' = \mathit{issued};\ \mathit{stock}' = \mathit{stock}; \\
\mathit{shelved}' = \mathit{shelved};\ \mathit{readers}' = \mathit{readers} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\mathit{WhoHasCopy} \\
\hline
\textbf{\textit{Ξ Library}} \\
c? : \mathit{Copy};\ r! : \mathit{Reader} \\
\hline
c? \in \operatorname{dom} \mathit{issued};\ r! = \mathit{issued}\ c? \\
\hline
\end{array}
$$

Example of a schema disjunction (1)

AddKnownTitle

$\Delta Library$
$b? : Book$
$rep! : Report$

$\mathbf{b? \in ran\ stock}$
$\exists\, c : Copy \mid c \notin \mathrm{dom}\ stock \bullet$
$\quad stock' = stock \oplus \{ c \mapsto b? \} \wedge$
$\quad shelved' = shelved \cup \{ c \}$
$issued' = issued;\ readers' = reader$
$\mathbf{rep! = FurtherCopyAdded}$

AddNewTitle

$\Delta Library$
$\mathrm{b}? : Book$
$rep! : Report$

$\mathbf{b? \notin ran\ stock}$
$\exists\, c : Copy \mid c \notin \mathrm{dom}\ stock \bullet$
$\quad stock' = stock \oplus \{ c \mapsto b? \} \wedge$
$\quad shelved' = shelved \cup \{ c \}$
$issued' = issued;\ readers' = reader$
$\mathbf{rep! = NewTitleAdded}$

$$AddCopy \mathrel{\widehat{=}} AddKnownTitle \vee AddNewTitle$$

Example of a schema disjunction (2)

$AddCopy$

$\Delta Library$

$b? : Book$

$rep! : Report$

$\exists\, c : Copy \mid c \notin \mathrm{dom}\ stock \bullet$

$\quad stock' = stock \oplus \{ c \mapsto b? \} \wedge$

$\quad shelved' = shelved \cup \{ c \}$

$issued' = issued;\ readers' = reader$

$\boldsymbol{b? \in \mathbf{ran}\ stock \Rightarrow rep! = FurtherCopyAdded}$

$\boldsymbol{b? \notin \mathbf{ran}\ stock \Rightarrow rep! = NewTitleAdded}$

Example of a schema conjunction

$$
\begin{array}{l}
\text{\_\_}\mathit{EnterNewCopy}\text{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}\\
\Delta \mathit{Library}\\
b? : \mathit{Book}\\
\hline
\exists\, c : \mathit{Copy} \mid c \notin \operatorname{dom} \mathit{stock} \bullet\\
\qquad \mathit{stock}' = \mathit{stock} \oplus \{\, c \mapsto b? \,\} \wedge\\
\qquad \mathit{shelved}' = \mathit{shelved} \cup \{\, c \,\}\\
\mathit{issued}' = \mathit{issued};\ \mathit{readers}' = \mathit{readers}
\end{array}
$$

$$
\begin{array}{l}
\text{\_\_}\mathit{AddCopyReport}\text{\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}\\
\mathit{Stock} : \mathit{Copy} \nrightarrow \mathit{Book}\\
b? : \mathit{Book}\\
\mathit{rep}! : \mathit{Report}\\
\hline
b? \notin \operatorname{ran} \mathit{stock}\\
\qquad \Rightarrow \mathit{rep}! = \mathit{NewTitleAdded}\\
b? \in \operatorname{ran} \mathit{stock}\\
\qquad \Rightarrow \mathit{rep}! = \mathit{FurtherCopyAdded}
\end{array}
$$

$$\mathit{AddCopy} \mathrel{\widehat{=}} \mathit{EnterNewCopy} \wedge \mathit{AddCopyReport}$$

Sample Specification

Example: Electronic dictionary

- Translation between two languages, called *Native* and *Foreign*
- Only orthographically correct words may be stored in the dictionary (*OrthoNative* and *OrthoForeign*, respectively)
- Each word of the native language is mapped onto a set of words of the foreign language (and vice versa)
- Operations to be provided:
  - » Insertion of a valid pair
  - » Output of all translations of a native word
  - » Output of all translations of a foreign word
  - » Testing the knowledge of a user:
    - ⇨ System selects a word randomly
    - ⇨ User supplies his translations
    - ⇨ System calculates the percentage of correct answers

Structure of the specification

- Base types and global definitions
- Abstract states
- Initialization
- Partial operations under normal conditions
- Calculation of preconditions
- Total operations (including error conditions)
- Summary and index

Syntax of Z

```
<specification> ::= (<paragraph>)* (* Specification consists of paragraphs *)

<paragraph> ::= "[" <ident> ("," <ident>)* "]" (* Base types *)
              | <axiomatic-box> (* Declarations plus optional predicates *)
              | <generic-box> (* ... plus generic parameters *)
              | <schema-box> (* "graphical" schema definition *)
              | <schema-name> [<gen-formals>] ≙ <schema-expr>
                          (* linear schema definition *)
              | <def-lhs> "==" <expr> (* Constant declaration *)
              | <ident> "::=" <branch> ("|" <branch>)+ (* Enumeration type *)
              | <predicate> (* Predicate for global variables *)
```

Base types and global definitions

[*Native*, *Foreign*]

(* All character strings in the respective alphabets *)

$OrthoNative : \mathbb{P}\ Native$
$OrthoForeign : \mathbb{P}\ Foreign$

(* Orthographically correct words *)

Message ::= *Ok* | *AlreadyKnownPair* | *NewPairEntered*
| *ErrorInForeignWord* | *ErrorInNativeWord* | *ErrorInBothWords*
| *UnknownNativeWord* | *UnknownForeignWord*
| *VocabIsEmpty* | *NoCorrectResponses*

(* Return codes for operations *)

Abstract states

WellFormedVocab (* Dictionary *)

$$\begin{array}{l} Vocab : OrthoNative \leftrightarrow OrthoForeign \\ NativeWordsKnown : \mathbb{F}\ OrthoNative \\ ForeignWordsKnown : \mathbb{F}\ OrthoForeign \\ \hline NativeWordsKnown = \mathrm{dom}\ Vocab \\ ForeignWordsKnown = \mathrm{ran}\ Vocab \end{array}$$

RecordOfProgress (* Testing of user *)

$$\begin{array}{l} CumuMaxMarks, CumuMarksScored, AveragePercent : \mathbb{N} \\ \hline 0 \leqslant AveragePercent \leqslant 100 \\ CumuMarksScored \leqslant CumuMaxMarks \\ AveragePercent = percent(CumuMarksScored, CumuMaxMarks) \end{array}$$

WordForWord (* Overall state *)

$$\begin{array}{l} WellFormedVocab \\ RecordOfProgress \end{array}$$

Initialization

$$
\begin{array}{l}
\textit{InitWord-For-Word} \\
\hline
\textit{WordForWord}' \\
\hline
\textit{Vocab}' = \emptyset \\
\textit{CumuMaxMarks}' = \textit{CumuMaxMarksScored}' = 0 \\
\hline
\end{array}
$$

Definition of partial operations (1)

$AddPair \mathrel{\widehat=} EnterPair \land ReportIfAlreadyKnown$

(* Insertion of a pair into the dictionary with return code *)

EnterPair

$\Delta WellFormedVocab$
$\Xi RecordOfProgress$
$n? : OrthoNative; f? : OrthoForeign$

$Vocab' = Vocab \cup \{ n? \mapsto f? \}$

(* Insertion of a pair *)

ReportIfAlreadyKnown

$Vocab : OrthoNative \leftrightarrow OrthoForeign$
$n? : OrthoNative; f? : OrthoForeign; rep! : Message$

$n? \mapsto f? \in Vocab \Rightarrow rep! = AlreadyKnownPair$
$n? \mapsto f? \notin Vocab \Rightarrow rep! = NewPairEntered$

(* Information if pair was already known *)

Definition of partial operations (2)

$ToForeign \mathrel{\widehat=} ForeignTranslations \land ReportIfKnownNative$

(* Translation of a word with return code *)

ForeignTranslations

$\Xi WordForWord$
$n? : OrthoNative; ftrans! : \mathbb{F}\ OrthoForeign$

$ftrans! = \operatorname{ran} (\{ n? \} \lhd Vocab)$

(* Retrieval of translations *)

ReportIfKnownNative

$\Xi WellFormedVocab$
$n? : OrthoNative; rep! : Message$

$n? \in NativeWordsKnown \Rightarrow rep! = Ok$
$n? \notin NativeWordsKnown \Rightarrow rep! = UnknownNativeWord$

(* Information whether there is a translation for the given word *)

Definition of partial operations (3)

$VocabTestNtoF \mathrel{\widehat=} SelectTestWordN \land CheckResponsesF \land UpdateScoreNtoF$

(* Test: Select word, check responses, update score *)

SelectTestWordN

$WellFormedVocab$
$TestWord! : OrthoNative$
$Translations : \mathbb{F}\ OrthoForeign$
$TransCount! : \mathbb{N}$

$TestWord! \in NativeWordsKnown$
$Translations = \operatorname{ran}(\{ TestWord! \} \lhd Vocab)$
$TransCount! = \#Translations$

(* Selection of a word *)

Definition of partial operations (4)

CheckResponsesF

$$Translations, CorrectResponses! : \mathbb{F}\ OrthoForeign$$
$$Responses? : \text{seq}\ Foreign$$
$$rep! : Message$$

$$CorrectResponses! = Translations \cap \text{ran}\ Responses?$$
$$CorrectResponses! = \varnothing \Rightarrow rep! = NoCorrectResponses$$
$$CorrectResponses! \neq \varnothing \Rightarrow rep! = Ok$$

(* Correct responses included? *)

Definition of partial operations (5)

$$
\begin{array}{l}
\textit{UpdateScoreNtoF} \\
\hline
\Xi \textit{WellFormedVocab} \\
\Delta \textit{RecordOfProgress} \\
\textit{Translations}, \textit{CorrectResponses}! : \mathbb{F}\ \textit{OrthoNative} \\
\textit{TransCount}!, \textit{NewAverage}! : \mathbb{N} \\
\hline
\textit{CumuMaxMarks}' = \textit{CumuMaxMarks} + \textit{TransCount}! \\
\textit{CumuMarksScored}' = \textit{CumuMarksScored} + \#\textit{CorrectResponses}! \\
\textit{NewAverage}! = \textit{AveragePercent}' \\
\hline
\end{array}
$$

(* Output of the number of correct responses and new average percentage *)

Calculation of preconditions

Operation	Inputs and outputs	Preconditions
AddPair	$n? : Native; f? : Foreign$ $rep! : Message$	$n? \in OrthoNative$ $f? \in OrthoForeign$
ToForeign	$n? : Native$ $ftrans! : \mathbb{F}\ OrthoForeign$ $rep! : Message$	$n? \in OrthoNative$
VocabTestNtoF	$Responses? : \mathrm{seq}\ Foreign$ $TestWord! : OrthoNative$ $CorrectResponses! : \mathbb{F}\ OrthoForeign$ $TransCount! : \mathbb{N}$ $NewAverage! : \mathbb{N}$ $rep! : Message$	$Vocab \neq \emptyset$

Total operations (error handling)

$TotalAidPair \mathrel{\widehat=} AddPair \lor AddPairError$

(* Total operation = normal operation + error handling *)

AddPairError

$\Xi WordForWord$

$n? : Native; f? : Foreign; rep! : Message$

$n? \in OrthoNative \land f? \notin OrthoForeign$
$\quad \Rightarrow rep! = ErrorInForeignWord$

$n? \notin OrthoNative \land f? \in OrthoForeign$
$\quad \Rightarrow rep! = ErrorInNativeWord$

$n? \notin OrthoNative \land f? \notin OrthoForeign$
$\quad \Rightarrow rep! = ErrorInBothWords$

Summary and index

$AddPair \mathrel{\widehat=} EnterPair \wedge ReportIfAlreadyKnown$

$ToForeign \mathrel{\widehat=} ForeignTranslations \wedge ReportIfKnownNative$

$VocabTestNtoF \mathrel{\widehat=} SelectTestWordN \wedge CheckResponsesF \wedge UpdateScoreNtoF$

$TotalAddPair \mathrel{\widehat=} AddPair \vee AddPairError$

$\ldots$

Proving of Specification Properties

Survey

- Foundations for proving specification properties:
  - Proposition logic
  - Predicate logic
  - Set theory
- Example-based demonstration of
  - Correctness of the initial state of a data type
  - Simplification of a precondition of an operation
  - Proving a property of an operation composition
- Not all used axioms will be introduced explicitly
- Example: Administration of soccer fans
  - Each fan is registered under a unique identification number
  - A subset of fans may be banned (hooligans)
  - Operations for inserting, deleting, banning fans, etc.

Z specification of soccer fan administration (1)

$[PERSON, ID]$

(* Given sets *)

Fid

$members : ID \rightarrowtail\!\!\!\!\!\!\!\!\;\;+\; PERSON$

$banned : \mathbb{P}\ ID$

$banned \subseteq \mathrm{dom}\ members$

(* State *)

$InitFid$

Fid'

$members' = \emptyset$

$banned' = \emptyset$

(* Initial state (without members) *)

Z specification of soccer fan administration (2)

AddMember

$$\Delta Fid$$
$$applicant? : PERSON$$
$$id! : ID$$

$$applicant? \notin \operatorname{ran} members$$
$$id! \notin \operatorname{dom} members$$
$$members' = members \cup \{ id! \mapsto applicant? \}$$
$$banned' = banned$$

DeleteMember

$$\Delta Fid$$
$$id? : ID$$

$$id? \in \operatorname{dom} members$$
$$members' = \{ id? \} \lhd members$$
$$banned' = banned \setminus \{ id? \}$$

BanMember

$$\Delta Fid$$
$$ban? : ID$$

$$ban? \in \operatorname{dom} members$$
$$members' = members$$
$$banned' = banned \cup \{ ban? \}$$

Correctness of the initial state

$\vdash \exists\, Fid' \bullet InitFid$

$\Leftrightarrow$ (Substitution of Fid' and $InitFid$)

$\vdash \exists\, members' : ID \rightarrowtail\!\!\!\!\!\!+\; PERSON;\ banned' : \mathbb{P}\, ID \mid$
$\quad banned' \subseteq \mathrm{dom}\ members' \bullet$
$\quad members' = \varnothing \wedge banned' = \varnothing$

$\Leftrightarrow$ ($\exists\, Decs \mid Constr \bullet Pred \equiv \exists\, Decs \bullet Constr \wedge Pred$)

$\vdash \exists\, members' : ID \rightarrowtail\!\!\!\!\!\!+\; PERSON;\ banned' : \mathbb{P}\, ID \bullet$
$\quad banned' \subseteq \mathrm{dom}\ members' \wedge members' = \varnothing \wedge banned' = \varnothing$

This proposition holds because:

- $\varnothing : ID \rightarrowtail\!\!\!\!\!\!+\; PERSON$
- $\varnothing : \mathbb{P}\, ID$
- $\varnothing \subseteq \varnothing$

Simplification of a precondition (1)

$$
\begin{array}{l}
\textit{PreAddMember} \\
\hline
\textit{Fid} \\
\textit{applicant?} : \textit{PERSON} \\
\hline
\exists\, \textit{Fid}' ; \textit{id!} : \textit{ID} \bullet \\
\quad \textit{applicant?} \notin \operatorname{ran} \textit{members} \wedge \\
\quad \textit{id!} \notin \operatorname{dom} \textit{members} \wedge \\
\quad \textit{members}' = \textit{members} \cup \{ \textit{id!} \mapsto \textit{applicant?} \} \wedge \\
\quad \textit{banned}' = \textit{banned} \\
\hline
\end{array}
$$

$\Leftrightarrow$ (Expansion of $\textit{Fid}'$)

Simplification of a precondition (2)

$$
\begin{array}{|l}
\hline
\textit{PreAddMember} \\
\textit{Fid} \\
\textit{applicant?} : \textit{PERSON} \\
\hline
\exists\ \textit{members}' : \textit{ID} \rightarrowtail\!\!\!\!\rightarrow \textit{PERSON};\ \textit{banned}' : \mathbb{P}\ \textit{ID};\ \textit{id!} : \textit{ID}\ \bullet \\
\qquad \textit{banned}' \subseteq \operatorname{dom} \textit{members}' \land \\
\qquad \textit{applicant?} \notin \operatorname{ran} \textit{members} \land \\
\qquad \textit{id!} \notin \operatorname{dom} \textit{members} \land \\
\qquad \textit{members}' = \textit{members} \cup \{\, \textit{id!} \mapsto \textit{applicant?} \,\} \land \\
\qquad \textit{banned}' = \textit{banned} \\
\hline
\end{array}
$$

$\Leftrightarrow$ (Elimination of existential quantifiers for *members′* and *banned′*)

Simplification of a precondition (3)

$$
\begin{array}{|l}
\textit{PreAddMember} \\
\hline
\textit{Fid} \\
\textit{applicant?} : \textit{PERSON} \\
\hline
\exists\, \textit{id!} : \textit{ID} \bullet \\
\qquad \textit{members} \cup \{ \textit{id!} \mapsto \textit{applicant?} \} \in \textit{ID} \rightarrowtail\!\!\!\!\twoheadrightarrow \textit{PERSON} \wedge \\
\qquad \textit{banned} \in \mathbb{P}\ \textit{ID} \wedge \\
\qquad \textit{banned} \subseteq \operatorname{dom} \textit{members} \cup \{ \textit{id!} \mapsto \textit{applicant?} \} \wedge \\
\qquad \textit{applicant?} \notin \operatorname{ran} \textit{members} \wedge \\
\qquad \textit{id!} \notin \operatorname{dom} \textit{members} \\
\hline
\end{array}
$$

$$\Leftrightarrow (\textit{Fid} \Rightarrow \textit{banned} \in \mathbb{P}\ \textit{ID})$$

Simplification of a precondition (4)

$$
\begin{array}{l}
\textit{PreAddMember} \\
\hline
\textit{Fid} \\
\textit{applicant}? : \textit{PERSON} \\
\hline
\exists\, \textit{id}! : \textit{ID} \bullet \\
\qquad \textit{members} \cup \{\, \textit{id}! \mapsto \textit{applicant}? \,\} \in \textit{ID} \rightarrowtail\!\!\!\!\!\twoheadrightarrow \textit{PERSON} \wedge \\
\qquad \textit{banned} \subseteq \operatorname{dom} \textit{members} \cup \{\, \textit{id}! \mapsto \textit{applicant}? \,\} \wedge \\
\qquad \textit{applicant}? \notin \operatorname{ran} \textit{members} \wedge \\
\qquad \textit{id}! \notin \operatorname{dom} \textit{members} \\
\hline
\end{array}
$$

$\Leftrightarrow$ ($\operatorname{dom} (R \cup S) = \operatorname{dom} R \cup \operatorname{dom} S$)

Simplification of a precondition (5)

$$
\begin{array}{l}
\textit{PreAddMember} \\
\hline
\textit{Fid} \\
\textit{applicant?} : \textit{PERSON} \\
\hline
\exists\, \textit{id!} : \textit{ID} \bullet \\
\qquad \textit{members} \cup \{\, \textit{id!} \mapsto \textit{applicant?} \,\} \in \textit{ID} \rightarrowtail\!\!\!\!\leftrightarrow \textit{PERSON} \land \\
\qquad \textit{banned} \subseteq \operatorname{dom} \textit{members} \cup \operatorname{dom} \{\, \textit{id!} \mapsto \textit{applicant?} \,\} \land \\
\qquad \textit{applicant?} \notin \operatorname{ran} \textit{members} \land \\
\qquad \textit{id!} \notin \operatorname{dom} \textit{members} \\
\hline
\end{array}
$$

$$\Leftrightarrow (\textit{Fid} \Rightarrow \textit{banned} \subseteq \operatorname{dom} \textit{members})$$

Simplification of a precondition (6)

PreAddMember

Fid

$applicant? : PERSON$

$\exists\, id! : ID \bullet$

$members \cup \{ id! \mapsto applicant? \} \in ID \rightarrowtail\!\!\!\!\!\!\!\rightarrow PERSON \wedge$

$applicant? \notin \operatorname{ran} members \wedge$

$id! \notin \operatorname{dom} members$

$$\Leftrightarrow \quad (members \in ID \rightarrowtail\!\!\!\!\!\!\!\rightarrow PERSON \wedge$$
$$applicant? \notin \operatorname{ran} members \wedge$$
$$id! \notin \operatorname{dom} members \Rightarrow$$
$$members \cup \{ id! \mapsto applicant? \} \in ID \rightarrowtail\!\!\!\!\!\!\!\rightarrow PERSON)$$

Simplification of a precondition (7)

PreAddMember

Fid

$applicant? : PERSON$

$\exists\, id! : ID \bullet$

$\quad applicant? \notin \text{ran}\ members \wedge$

$\quad id! \notin \text{dom}\ members$

$\Leftrightarrow$ (Remove first subexpression from the existential quantifier)

Simplification of a precondition (8)

PreAddMember

Fid
$applicant? : PERSON$

$applicant? \notin \operatorname{ran} members \land$
$\exists\, id! : ID \bullet id! \notin \operatorname{dom} members$

⇔ (Elimination of the existential quantifier)

PreAddMember

Fid
$applicant? : PERSON$

$applicant? \notin \operatorname{ran} members \land$
$\operatorname{dom} members \neq ID$

Proving a property of the specification (1)

Sequential execution of *AddMember* (with output $id!$) and *DeleteMember* (with input $id?$) does not change the state:

$$AddandDelete \mathrel{\widehat{=}} AddMember \mathbin{⨟} DeleteMember \mid id! = id? \vdash \Xi Fid$$

$$
\begin{array}{l}
\textbf{AddandDelete} \\
\hline
\Delta Fid \\
applicant? : PERSON \\
id? : ID; id! : ID \\
\hline
\exists Fid'' \bullet \\
\quad applicant? \notin \operatorname{ran} members \land \\
\quad id! \notin \operatorname{dom} members \land \\
\quad members'' = members \cup \{ id! \mapsto applicant? \} \land \\
\quad banned'' = banned \land \\
\quad id? \in \operatorname{dom} members'' \land \\
\quad members' = \{ id? \} \lhd members'' \land \\
\quad banned' = banned'' \setminus \{ id? \} \land \\
\quad id! = id? \\
\hline
\end{array}
$$

Proving a property of the specification (2)

$\Leftrightarrow$ (Expansion of Fid'')

AddandDelete

ΔFid
$applicant? : PERSON$
$id? : ID; id! : ID$

$\exists\, members'' : ID \rightarrowtail\!\!\!\!\!\rightarrow PERSON; banned'' : \mathbb{P}\, ID \bullet$
$banned'' \subseteq \operatorname{dom} members'' \wedge$
$applicant? \notin \operatorname{ran} members \wedge$
$id! \notin \operatorname{dom} members \wedge$
$members'' = members \cup \{ id! \mapsto applicant? \} \wedge$
$banned'' = banned \wedge$
$id? \in \operatorname{dom} members'' \wedge$
$members' = \{ id? \} \lhd members'' \wedge$
$banned' = banned'' \setminus \{ id? \} \wedge$
$id! = id?$

Proving a property of the specification (3)

⇔ (Elimination of existential quantifiers for *banned´´* and *members´´*)

AddandDelete

ΔFid
$applicant? : PERSON$
$id? : ID; id! : ID$

$members \cup \{ id! \mapsto applicant? \} \in ID \rightarrowtail PERSON \wedge$
$banned \in \mathbb{P}\ ID \wedge$
$banned \subseteq \mathrm{dom}\ members \cup \{ id! \mapsto applicant? \} \wedge$
$applicant? \notin \mathrm{ran}\ members \wedge$
$id! \notin \mathrm{dom}\ members \wedge$
$id? \in \mathrm{dom}\ members \cup \{ id! \mapsto applicant? \} \wedge$
$members' = \{ id? \} \lhd (members \cup \{ id! \mapsto applicant? \}) \wedge$
$banned' = banned \setminus \{ id? \} \wedge$
$id! = id?$

Proving a property of the specification (4)

Calculation of *members´*:

$members' =$

$\{ id? \} \lhd (members \cup \{ id! \mapsto applicant? \}) =$ $(id! = id?)$

$\{ id! \} \lhd (members \cup \{ id! \mapsto applicant? \}) =$ $(R \lhd (S \cup T) = (R \lhd S) \cup (R \lhd T))$

$(\{ id! \} \lhd members) \cup \{ id! \} \lhd \{ id! \mapsto applicant? \} =$ (Definition von $\lhd$)

$(\{ id! \} \lhd members) \cup \varnothing =$

$\{ id! \} \lhd members =$ $(id! \notin \mathrm{dom}\ members)$

$members$

Proving a property of the specification (5)

Calculation of $banned'$:

$banned' =$	
$banned \setminus \{ id? \} =$	$(id! = id?)$
$banned \setminus \{ id! \} =$	$(id! \notin \mathrm{dom}\ members \wedge$
	$banned \subseteq \mathrm{dom}\ members)$
$banned$	

$$members' = members \wedge banned' = banned \Rightarrow$$

$$Fid' = Fid \Rightarrow$$

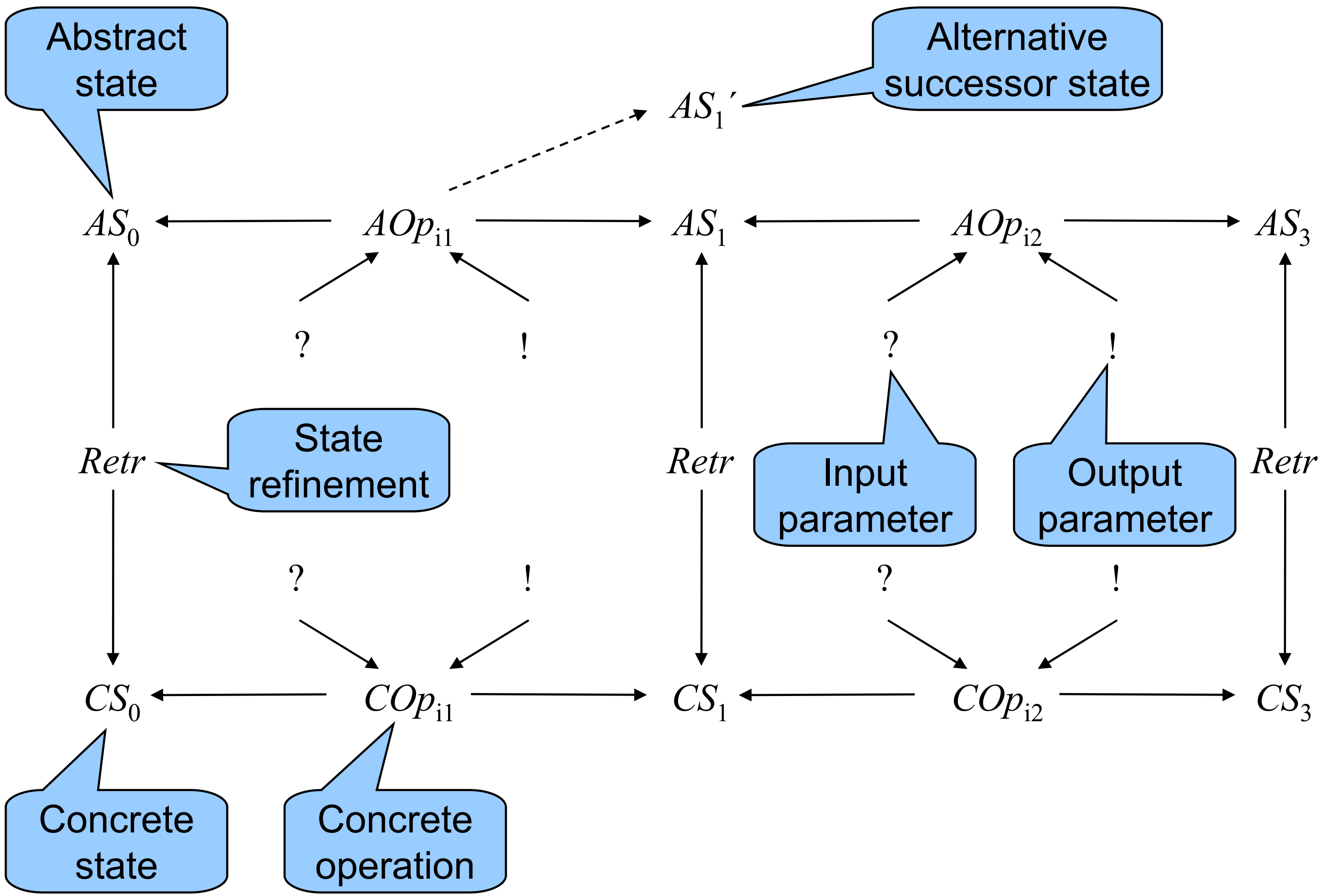
$$\Xi\ Fid$$

Refinement of Specifications

Goal and approach

- Starting point: abstract specification with an abstract state and abstract operations
- Goal: transformation into a concrete specification which is nearer to the final implementation
- Refinement may be performed multiple times (i.e., multiple levels)
- Definition of a refinement:
  - Relation between abstract and concrete states, where each concrete state is mapped onto at most one abstract state
  - Each concrete initial state must be mapped onto a correct abstract initial state
  - Each concrete operation is mapped onto a corresponding abstract operation
  - The behavior of the concrete operation must be consistent with the behavior of the abstract operation

Illustration



Formal definition of a refinement

- AS, CS — Schemata for abstract and concrete states
  $InitAS$, $InitCS$ — Schemata for initial states
- $Retr(ieve)$ — Schema for the correlation of abstract and concrete states

```
┌─ Retr ──────────────────────────────
│ AS
│ CS
├──────────────
│ RelASCS
└─────────────────────────────────────
```

- Each schema AO for an abstract operation is mapped onto a schema CO for the corresponding concrete operation

Theorems to be proved

- **Initialization theorem**
  Each concrete initial state represents an abstract initial state:
  $InitCS \wedge Retr' \vdash InitAS$

- **Applicability theorems**
  If an abstract operation is applicable in an abstract state, the corresponding concrete operation is applicable in the corresponding concrete state:
  $\text{pre } AOp \wedge Retr \vdash \text{pre } COp$

- **Correctness theorems**
  If an abstract operation is applicable and the corresponding concrete operation is applied, the behavior is latter is consistent with the behavior of the former:
  $\text{pre } AOp \wedge Retr \wedge COp \wedge Retr' \vdash AOp$

Abstract state of the soccer fan administration

FidScheme

$members : ID \rightarrowtail\!\!\!\!\!\!\rightarrow PERSON$
$banned : \mathbb{P}\ ID$

$banned \subseteq \mathrm{dom}\ members$
$\#members \leqslant maxmems$

(* Abstract state, now with maximal number of members *maxmems* *)

InitFidScheme

Fid'

$members' = \emptyset$
$banned' = \emptyset$

(* Initial state (without members) *)

Concrete state: array-based realization

banarr	
index	ℕ
1	*4*
2	*5*
3	*2*
...	...

membarr		
index	*ID*	*PERSON*
1	*WW8901*	*Tom Cobbley*
2	*WW8903*	*Bill Vandal*
3	*WW9001*	*Daisy Widden*
4	*WW9002*	*Joe Hooly*
5	*WW9004*	*Sandra Skintight*
6	*WW9007*	*Joan Brewer*
...	...	...

Z specification for the concrete state

$\mathrm{iseq}[X] == \mathrm{seq}\ X \cap (\mathbb{N} \rightarrowtail\!\!\!\!\!\!\leftrightarrow X)$

(* Representation of arrays by injective sequences *)

CFidScheme

$membarr : \mathrm{iseq}[ID \times PERSON]$
$banarr : \mathrm{iseq}[\mathbb{N}]$

$\mathrm{ran}\ membarr \in ID \rightarrowtail\!\!\!\!\!\!\leftrightarrow PERSON$
$\mathrm{ran}\ banarr \subseteq 1..\#membarr$
$\#membarr \leqslant maxmems$

(* Concrete state, with maximal number of members *maxmems* *)

InitCFidScheme

$CFidScheme'$

$membarr' = \langle\,\rangle$
$banarr' = \langle\,\rangle$

(* Initial state (without members) *)

Relation between abstract and concrete states

Retr

FidScheme
CFidScheme

$members = \mathrm{ran}\ membarr$

$banned = \mathrm{dom}\ (\ \mathrm{ran}\ (\ \mathrm{ran}\ banarr \lhd membarr\ )\ )$

(* Members are pairs occurring as elements of *membarr*.
The identifiers of banned persons are obtained as the first components of pairs which are marked by indices in *banarr*. *)

Initialization theorem

To demonstrate:

$InitCFidScheme \wedge Retr' \vdash InitFidScheme$

$members' = \mathrm{ran}\ membarr' = \mathrm{ran}\ \langle\, \rangle = \varnothing$

$$\begin{aligned} banned' &= \mathrm{dom}\ (\ \mathrm{ran}\ (\ \mathrm{ran}\ banarr' \lhd membarr'\ )\ ) \\ &= \mathrm{dom}\ (\ \mathrm{ran}\ (\ \mathrm{ran}\ \langle\, \rangle \lhd \langle\, \rangle\ )\ ) \\ &= \varnothing \end{aligned}$$

Abstract and concrete operation

AddMember

$$\Delta FidScheme$$
$$applicant? : PERSON$$
$$id! : ID$$

$$applicant? \notin \operatorname{ran} members$$
$$id! \notin \operatorname{dom} members$$
$$members' = members \cup \{ id! \mapsto applicant? \}$$
$$banned' = banned$$

CAddMember

$$\Delta CFidScheme$$
$$applicant? : PERSON$$
$$id! : ID$$

$$applicant? \notin \operatorname{ran} (\operatorname{ran} membarr)$$
$$id! \notin \operatorname{dom} (\operatorname{ran} membarr)$$
$$membarr' = membarr \frown \langle (id!, applicant?) \rangle$$
$$banarr' = banarr$$

Preconditions

$PreAddMember$

$FidScheme$
$applicant? : PERSON$

$applicant? \notin \operatorname{ran} members$
$\operatorname{dom} members \neq ID$
$\#members < maxmems$

$PreCAddMember$

$CFidScheme$
$applicant? : PERSON$

$applicant? \notin \operatorname{ran} (\operatorname{ran} membarr)$
$\operatorname{dom} (\operatorname{ran} membarr) \neq ID$
$\#membarr < maxmems$

Applicability theorem

To demonstrate:

$PreAddMember \wedge Retr \vdash PreCAddMember \Leftrightarrow$

$FidScheme; applicant? : PERSON; CFidScheme \mid$

$applicant? \notin \text{ran } members$	(H1)
$\text{dom } members \neq ID$	(H2)
$\#members < maxmems$	(H3)
$members = \text{ran } membarr$	(H4)
$banned = \text{dom } (\text{ ran } (\text{ ran } banarr \lhd membarr))$	(H5)

$\vdash$

$applicant? \notin \text{ran } (\text{ran } membarr)$	(G1)
$\text{dom } (\text{ran } membarr) \neq ID$	(G2)
$\#membarr < maxmems$	(G3)

Proof of the applicability theorem

Proof of (G1):

$applicant? \notin \operatorname{ran} members$	(H1)
$\Rightarrow applicant? \notin \operatorname{ran} (\operatorname{ran} membarr)$	(H4)

Proof of (G2):

$\operatorname{dom} (\operatorname{ran} membarr)$	
$= \operatorname{dom} members$	(H4)
$\neq ID$	(H2)

Proof of (G3):

$\#membarr = \#(\operatorname{ran} membarr)$	(*membarr* is injective)
$= \#members$	(H4)
$< maxmems$	(H3)

Correctness theorem

To demonstrate:

$$PreAddMember \wedge Retr \wedge CAddMember \wedge Retr' \vdash AddMember \Leftrightarrow$$

$$PreAddMember \wedge Retr \wedge CAddMember \wedge Retr'$$

$$\vdash$$

$$applicant? \notin \mathrm{ran}\ members \qquad \text{(G1)}$$

$$\mathrm{id}! \notin \mathrm{dom}\ members \qquad \text{(G2)}$$

$$members' = members \cup \{ id! \mapsto applicant? \} \qquad \text{(G3)}$$

$$banned' = banned \qquad \text{(G4)}$$

Proof of the correctness theorem

Proof of (G1):

$applicant? \notin \text{ran } members$ $\quad$ ($PreAddMember$)

Proof of (G2):

$id! \notin \text{dom } (\text{ran } membarr)$ $\quad$ ($CAddMember$)

$\quad \Leftrightarrow id! \notin \text{dom } members$ $\quad$ ($Retr$)

Proof of (G3):

$members'$

$\quad = \text{ran } membarr'$ $\quad$ ($Retr'$)

$\quad = \text{ran } (membarr \frown \langle (id!, applicant?) \rangle)$ $\quad$ ($CAddMember$)

$\quad = \text{ran } membarr \cup \{ (id!, applicant?) \}$ $\quad$ (Properties of ran and $\frown$)

$\quad = members \cup \{ id! \mapsto applicant? \}$ $\quad$ ($Retr$)

Proof of (G4):

$banned'$

$\quad = \text{dom}(\text{ran } (\text{ran } banarr' \lhd membarr'))$ $\quad$ ($Retr'$)

$\quad = \text{dom}(\text{ran } (\text{ran } banarr \lhd (membarr \frown \langle (id!, applicant?) \rangle)))$ $\quad$ ($CAddMember$)

$\quad = \text{dom}(\text{ran } (\text{ran } banarr \lhd membarr))$ $\quad$ ($CFidScheme$)

$\quad = banned$ $\quad$ ($Retr$)

Summary

Advantages of Z

- Based on theoretical foundations (logic and set theory) which should be known at least to mathematically trained users
- Very general approach
- Compact specifications
- Model-oriented specification is easier to understand/construct than behavioral specification
- Proofs with the help of predicate logic and set theory
- Step-wise refinement of specifications is supported

Disadvantages of Z

- Complex notation with many, many operators
- Abstract data types are modeled only implicitly, relying on certain conventions
- Modeling of operations with Δ schemata is hard to understand at first glance
- Notations and methods for structuring large specifications are missing (schemata are too fine-grained for this purpose)
- Transition from the specification to the implementation is difficult
- Often, Z is used only as a documentation aid

Literature

- B. Potter, J. Sinclair, D. Till: **An Introduction to Formal Specification and Z**, International Series in Computer Science, Prentice Hall (1991)
  Introductory textbook, on which this chapter is based.
- J.B. Wordsworth: **Software Development with Z**, International Computer Science Series, Addison-Wesley (1992)
  Another textbook.
- J.M. Spivey: **The Z Notation: A Reference Manual**, Second Edition, International Series in Computer Science, Prentice Hall (1992)
  Reference Manual including the language definition. Not appropriate as a textbook.
- J. Bowen: **Formal Specification & Documentation Using Z**; International Thomson Computer Press (1996)
  Textbook with a brief introduction into Z, followed by many case studies.