

Schema language

The schema language is used to structure and compose mathematical descriptions: collating pieces of information, encapsulating them, and naming them for re-use.

It is the second component of the Z notation, the first being the mathematical language of logic and set theory.

Schemas

Declaration and constraint

Any mathematical description will involve declarations of identifiers, and constraints upon them.

In a formal specification, groups of identifiers may be used to describe compound objects or related properties. These identifiers will often be declared—and constrained—together.

We may simplify our formal specifications by factoring out patterns of declaration and constraint.

Examples

Both of the following mathematical expressions involve a declaration and a constraint:

$$\{ a, b : \mathbb{N} \mid a = 2 * b \bullet a \mapsto b \}$$

$$\exists c : \mathbb{N} \mid c \neq 1 \bullet c < 2$$

Example

A formal specification might involve the given types

sold : *Seat* ↔ *Customer*

seating : \mathbb{P} *Seat*

and the constraint

$\text{dom } \textit{sold} \sqsubseteq \textit{seating}$

Schemas

A schema is a pattern of declaration and constraint.

We will define schemas, and use them as declarations, as predicates, and as sets.

Notation

We may write schemas in horizontal form:

$[declaration \mid constraint]$

or in vertical form:

$declaration$
$constraint$

In the vertical form, we may elide the semicolons between declarations and the conjunctions between predicates.

Examples

$[a : Z; c : \mathbb{P}Z \mid c \neq \emptyset \wedge a \in c]$

$a : Z$
$c : \mathbb{P}Z$
$c \neq \emptyset$
$a \in c$

Naming schemas

We may give names to schemas, and use these names to refer to the corresponding pattern of declaration and constraint. The definition

$Name \hat{=} Exp$

introduces a schema name *Name* which is equivalent to *Exp*, which must be a schema expression.

Example

$Schema \hat{=} [a : Z; c : \mathbb{P}Z \mid c \neq \emptyset \wedge a \in c]$

$Schema \hat{=}$

$a : Z$
$c : \mathbb{P}Z$
$c \neq \emptyset$
$a \in c$

Notation

We may write the second definition above as:

$Schema$

$a : Z$
$c : \mathbb{P}Z$
$c \neq \emptyset$
$a \in c$

Example

$BoxOffice$

$sold : Seat \leftrightarrow Customer$
$seating : \mathbb{P}Seat$
$dom\ sold \subseteq seating$

Equivalence

Two schemas are said to be equivalent if they declare the same set of identifiers—with the same types—and impose a logically equivalent constraint upon them.

Example

$seating : \mathbb{P} Seat$
 $sold : Seat \leftrightarrow Customer$
 $dom\ sold \subseteq seating$
 $sold \in Seat \leftrightarrow Customer$

Using schemas

We may use a schema name:

- in place of a declaration: after quantifiers, in set comprehensions, and in lambda and mu expressions
- in place of a predicate; in this case, only the constraint part is important
- in place of a set

Examples

$\forall Schema \bullet a \in \mathbb{N}$
 $\exists a : \mathbb{Z}; c : \mathbb{P} \mathbb{Z} \bullet Schema$

Question

What is the type of the following function?

$\lambda BoxOffice \bullet seating \setminus dom\ sold$

Bindings

A binding is a mapping from names to values. The names are names of identifiers; the values are values of expressions of the appropriate type.

Notation

The binding that associates the name of identifier x with the value of expression e , and the name of y with the value of f can be written

$$\langle x \rightsquigarrow e, y \rightsquigarrow f \rangle$$

Example

$$s ::= \langle a \rightsquigarrow 2, c \rightsquigarrow \{1, 2, 3\} \rangle$$

Schemas as sets

A schema corresponds to a set of bindings.

Each binding in this set associates the identifiers declared in the schema with values in such a way that the constraint part of the schema is satisfied.

If this set is maximal within the specification, then it is said to be a schema type.

Example

The following bindings are associated with *Schema*:

$$\langle a \rightsquigarrow 1, c \rightsquigarrow \{0, 1, 2\} \rangle$$

$$\langle a \rightsquigarrow 0, c \rightsquigarrow \mathbb{Z} \rangle$$

$$\langle a \rightsquigarrow -3, c \rightsquigarrow \{-3\} \rangle$$

Example

The following definition introduces a schema type:

$$\text{SchemaType } \begin{array}{l} a : \mathbb{Z} \\ c : \mathbb{P} \mathbb{Z} \end{array}$$

Question 

What is the type of the following set?

$$\{ \text{Schema} \mid c \subseteq \{0, 1\} \}$$

Characteristic binding

If *Schema* is the name of a schema, then we write θSchema to denote the characteristic binding of *Schema*.

In this binding, each of the named identifiers is associated with its value in the current scope.

Example

$$\theta\text{Schema} = \langle a \rightsquigarrow a, c \rightsquigarrow c \rangle$$

Question

Is there any difference between θSchema and $\theta\text{SchemaType}$?

Characteristic tuples

Where a schema is used as part of a declaration, the characteristic binding for that schema is used as the corresponding part of the characteristic tuple.

Example

$$\{\text{Schema}, b : \mathbb{Z}\} = \{\text{Schema}, b : \mathbb{Z} \bullet (\theta\text{Schema}, b)\}$$

Schemas as sets (again)

When we use a schema name *Schema* as a set, we are referring to the following set of bindings:

$$\{\text{Schema} \bullet \theta\text{Schema}\}$$

Example

$\{ \text{Schema} \bullet \{ \text{Schema} \} \}$
 $\{ a : \mathbb{Z}; c : \mathbb{P} \mathbb{Z} \mid c \neq \emptyset \wedge a \in c \bullet \{ (a \rightsquigarrow a, c \rightsquigarrow c) \} \}$

Example

Date
month : Month
day : 1..31
month $\in \{ \text{sep}, \text{apr}, \text{jun}, \text{nov} \} \Rightarrow \text{day} \leq 30$
month = feb $\Rightarrow \text{day} \leq 29$

Schemas as declarations

When a schema name is used in place of a declaration, the named identifiers are introduced under the given constraint.

Component selection 

If s is an object of schema type and x is one of the named components in s , then $s.x$ is the value associated with x in s . For example, if s is the binding $\langle X \rightsquigarrow v \rangle$, then $s.x = v$.

$\text{marina} \& \text{birthday} : \text{Date}$ 

$\text{marina} \& \text{birthday} . \text{month} = \text{jan}$

$\text{marina} \& \text{birthday} . \text{day} = 28$

Example

$\exists (\text{Date}) \bullet \text{day} = 31$

$\exists \text{day} : 1..31; \text{month} : \text{Month} \mid$
 $(\text{month} \in \{ \text{sep}, \text{apr}, \text{jun}, \text{nov} \} \Rightarrow \text{day} \leq 30) \wedge$
 $\text{month} = \text{feb} \Rightarrow \text{day} \leq 29 \bullet$
 $\text{day} = 31$

Question 

What are the elements of the following set?

$$\{ \textit{Date} \mid \textit{day} = 31 \bullet \textit{month} \}$$

Question 

What are the elements of the following set?

$$\{ d : \textit{Date} \mid d.\textit{day} = 31 \bullet d.\textit{month} \}$$

Schemas as predicates

When a schema name is used in place of a predicate, a statement is made about the identifiers declared in the schema.

This statement is logically equivalent to the constraint information in the whole of the schema.

Example

$$\forall \textit{day} : \mathbb{Z}, \textit{month} : \textit{Month} \bullet \textit{Date} \Rightarrow \textit{day} \leq 31$$

$$\forall \textit{day} : \mathbb{Z}, \textit{month} : \textit{Month} \bullet$$

$$(\textit{day} \in 1..31 \wedge$$

$$(\textit{month} \in \{\textit{sep}, \textit{apr}, \textit{jun}, \textit{nov}\} \Rightarrow \textit{day} \leq 30) \wedge$$

$$\textit{month} = \textit{feb} \Rightarrow \textit{day} \leq 29) \Rightarrow$$

$$\textit{day} \leq 31$$

Normalisation

A schema in which there is no constraint information in the declaration part is said to be in normal form.

It is sometimes useful to replace a schema with an equivalent schema in normal form.

Example

NormalDate
<i>month</i> : <i>Month</i>
<i>day</i> : \mathbb{Z}
$\textit{day} \in 1..31$
$\textit{month} \in \{\textit{sep}, \textit{apr}, \textit{jun}, \textit{nov}\} \Rightarrow \textit{day} \leq 30$
$\textit{month} = \textit{feb} \Rightarrow \textit{day} \leq 29$

Note

The declaration part of a normalised schema is that of a schema type.

DateType $\underline{\hspace{2cm}}$
 month : Month
 day : \mathbb{Z}

Otherwise we have a subrange type

Question

If we define

NotDate $\underline{\hspace{2cm}}$
 month : Month
 day : 1..31
 $\neg ((\text{month} \in \{\text{sep, apr, jun, nov}\} \Rightarrow \text{day} \leq 30) \wedge$
 $(\text{month} = \text{feb} \Rightarrow \text{day} \leq 29))$

is the following statement true or false?

$\forall \text{DateType} \mid \neg \text{Date} \bullet \text{NotDate}$

Renaming

If *old* is one of the identifiers declared in *Schema*, then the schema *Schemal[new/old]* is the schema that

- declares *new* instead of *old*
- imposes the same constraint upon *new* as it did upon *old*

Example

The schema *Date[dd/day, mm/month]* is equivalent to

$mm : \text{Month}$
 $dd : 1..31$
 $mm \in \{\text{sep, apr, jun, nov}\} \Rightarrow dd \leq 30$
 $mm = \text{feb} \Rightarrow dd \leq 29$

Decoration

If *Schema* is a schema, then *Schemal'* is the schema that declares the same identifiers as *Schema*, and imposes the same constraint, except that every identifier is decorated with a prime symbol (').

Example

Schemal' is equivalent to

$d' : \mathbb{Z}$
 $c' : \mathbb{P}\mathbb{Z}$
 $c' \neq \emptyset$
 $d' \in c'$

Example

BoxOffice' is equivalent to

seating' : \mathbb{P} *Seat*
sold' : *Seat* \leftrightarrow *Customer*
 $\text{dom } \textit{sold}' \subseteq \textit{seating}'$

Equality of bindings

Two bindings are equal if they map the same identifiers to equal values.

$$\langle X \rightsquigarrow V \rangle = \langle X \rightsquigarrow W \rangle \Leftrightarrow V = W$$

Decoration of characteristic bindings 

If b is a characteristic binding, then b' is a binding in which the same names are associated with primed values.

$$\langle X \rightsquigarrow X \rangle' = \langle X \rightsquigarrow X' \rangle$$

Binding decoration binds more loosely than θ :

$$\theta \textit{Schema}' = (\theta \textit{Schema})'$$

Example

The decorated binding $\theta \textit{Schema}'$ is

$$\langle a \rightsquigarrow a', c \rightsquigarrow c' \rangle$$

The characteristic binding $\theta(\textit{Schema})$ is

$$\langle a' \rightsquigarrow a', c' \rightsquigarrow c' \rangle$$

Decoration and schema types

Decoration does not change the type of a binding: $\theta \textit{Schema}'$ is an object of schema type *Schema*.

The declaration *Schema'* may be used to introduce $\theta \textit{Schema}'$: it declares every variable in the decorated binding.

Example

The declaration *Schema*: *Schema'* may be used to introduce two objects of schema type *Schema*:

$$\langle a \rightsquigarrow a, c \rightsquigarrow c \rangle \quad \text{and} \quad \langle a \rightsquigarrow a', c \rightsquigarrow c' \rangle$$

Question 

Is the following proposition true or false?

$$\forall a' : \mathbb{Z}; c' : \mathbb{P}\mathbb{Z} \bullet$$

$$\theta\text{Schema} \in \text{Schema}$$

Quantification over schema types

We will sometimes encounter quantified expressions in which the components of the declaration schema do not appear; a characteristic binding is used instead.

Where this is the case, we may avoid unnecessary expansion by reasoning at the level of schemas and bindings.

Substitution of bindings

A pair of bindings from the same schema type may be used to specify a renaming; the effect is that of a partial decoration.

The substituted schema

$$\text{SchemaA}[\theta\text{SchemaB}' / \theta\text{SchemaB}]$$

may be obtained from *SchemaA* by decorating the identifiers that appear in *θSchemaB*.

Example

Suppose that *NewSchema* is defined by

$$\text{NewSchema} \frac{\begin{array}{l} a : \mathbb{Z} \\ b : \mathbb{Z} \\ c : \mathbb{P}\mathbb{Z} \\ b \notin c \end{array}}{\quad}$$

NewSchema[*θSchema* / *θSchema*] is equivalent to

$$\frac{\begin{array}{l} a' : \mathbb{Z} \\ b : \mathbb{Z} \\ c' : \mathbb{P}\mathbb{Z} \\ b \notin c' \end{array}}{\quad}$$

Universal introduction

$$\frac{\begin{array}{l} [\theta S \in S^{\uparrow\{t\}} \\ \vdots \\ P \end{array}}{\forall S \bullet P} \text{ [V-intro}^{\{t\}}]$$

Provided that *θS* is not free in the other assumptions.

Universal elimination

$$\frac{\theta S' \in S \quad \forall S \bullet P}{P[\theta S' / \theta S]} \text{ [V-elim]}$$

Existential introduction

$$\frac{\theta S' \in S \quad P[\theta S' / \theta S]}{\exists S \bullet P} \text{ [E-intro]}$$

Existential elimination

$$\frac{\begin{array}{c} [\theta S \in S]^{[i]} \\ [P]^{[i]} \\ \vdots \\ \exists S \bullet P \\ Q \end{array}}{Q} \text{ [E-elim]^{[i]}}$$

Provided that θS is not free in the other assumptions and that θS is not free in Q .

Summary

- schema language
- schemas as sets
- schemas as declarations
- schemas as predicates
- decoration
- schema types