Relations and Functions

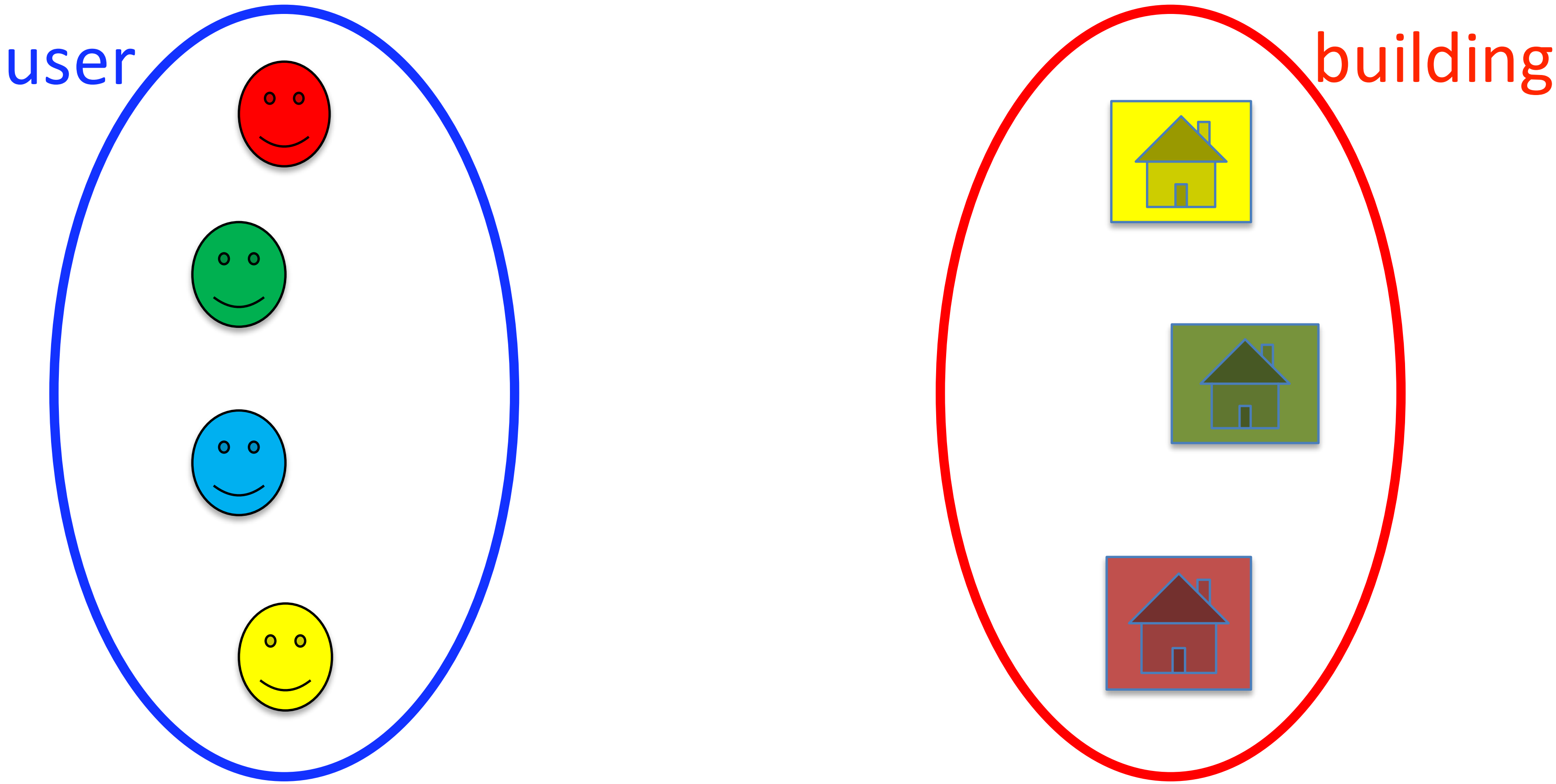
Michael Butler

SETSS 2016, Chongqing

Requirements for a Buildings Access System

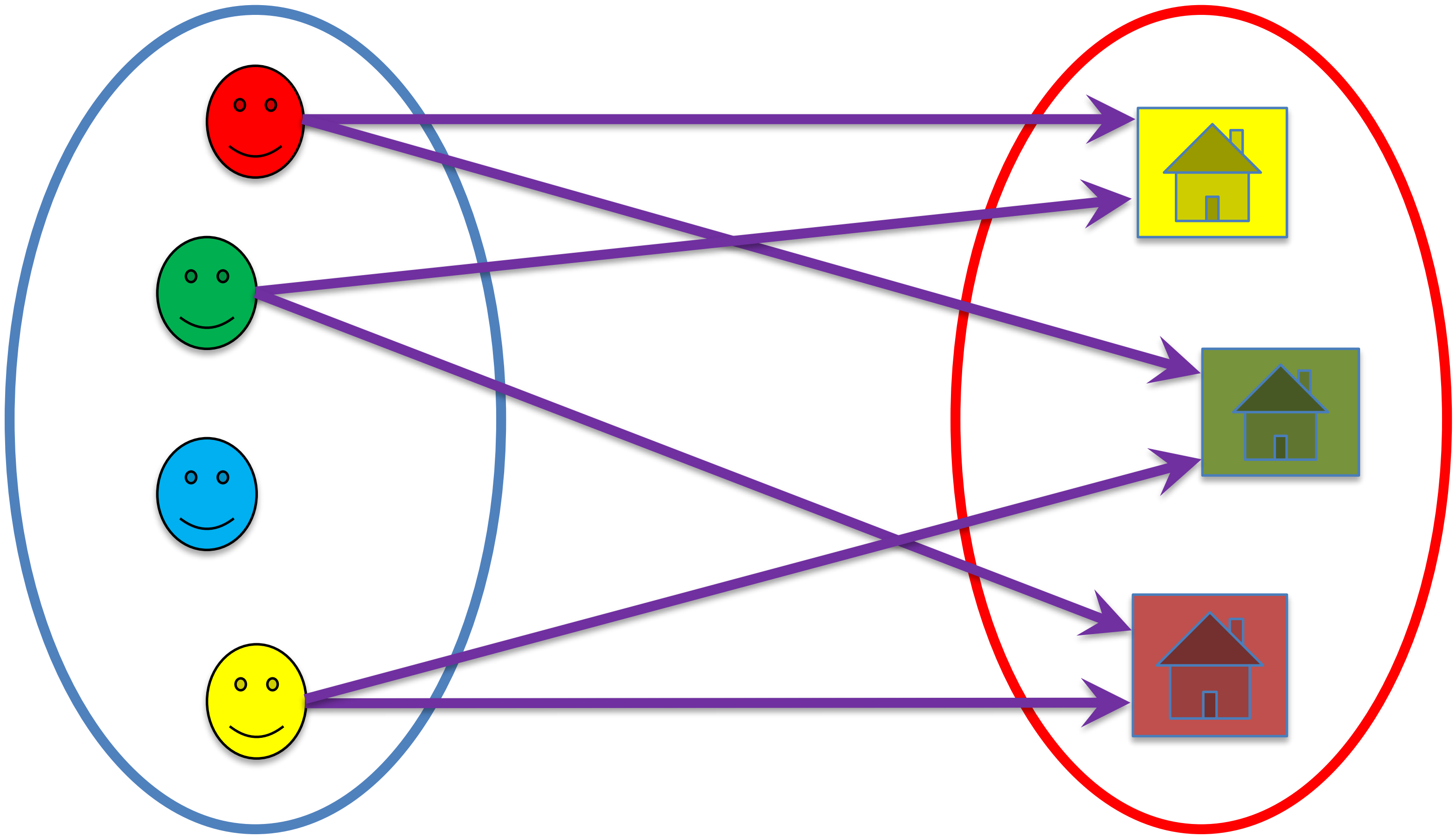
- Specify a system that controls access to a collection of buildings.
- Registered users will have access permission to enter certain buildings.
- A user can only enter buildings that they have access permission for.
- The system should keep track of the location of users.
- The system should manage registration and access permission for users.

Users and Buildings



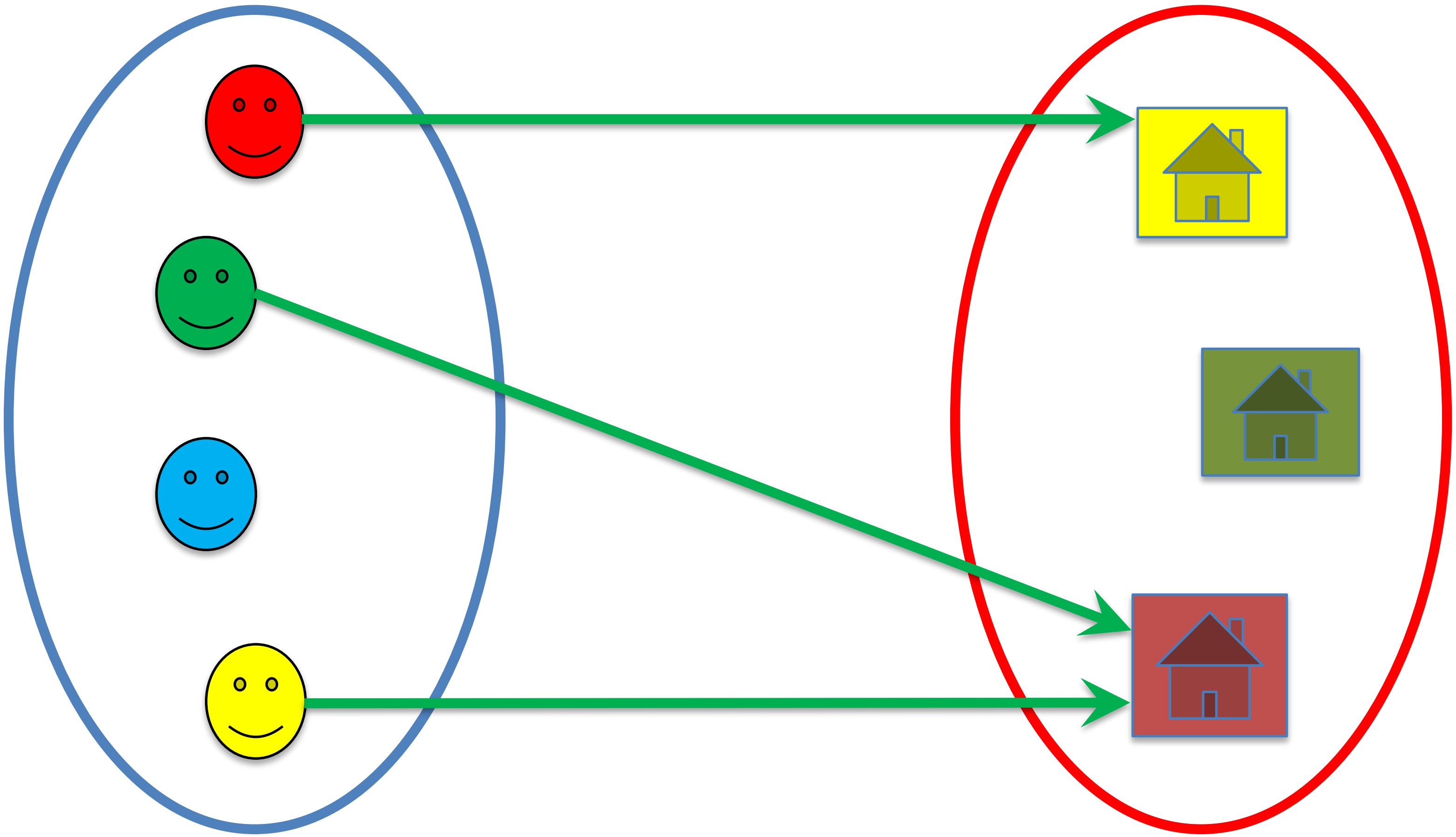
Carrier sets: USER BUILDING

Permission



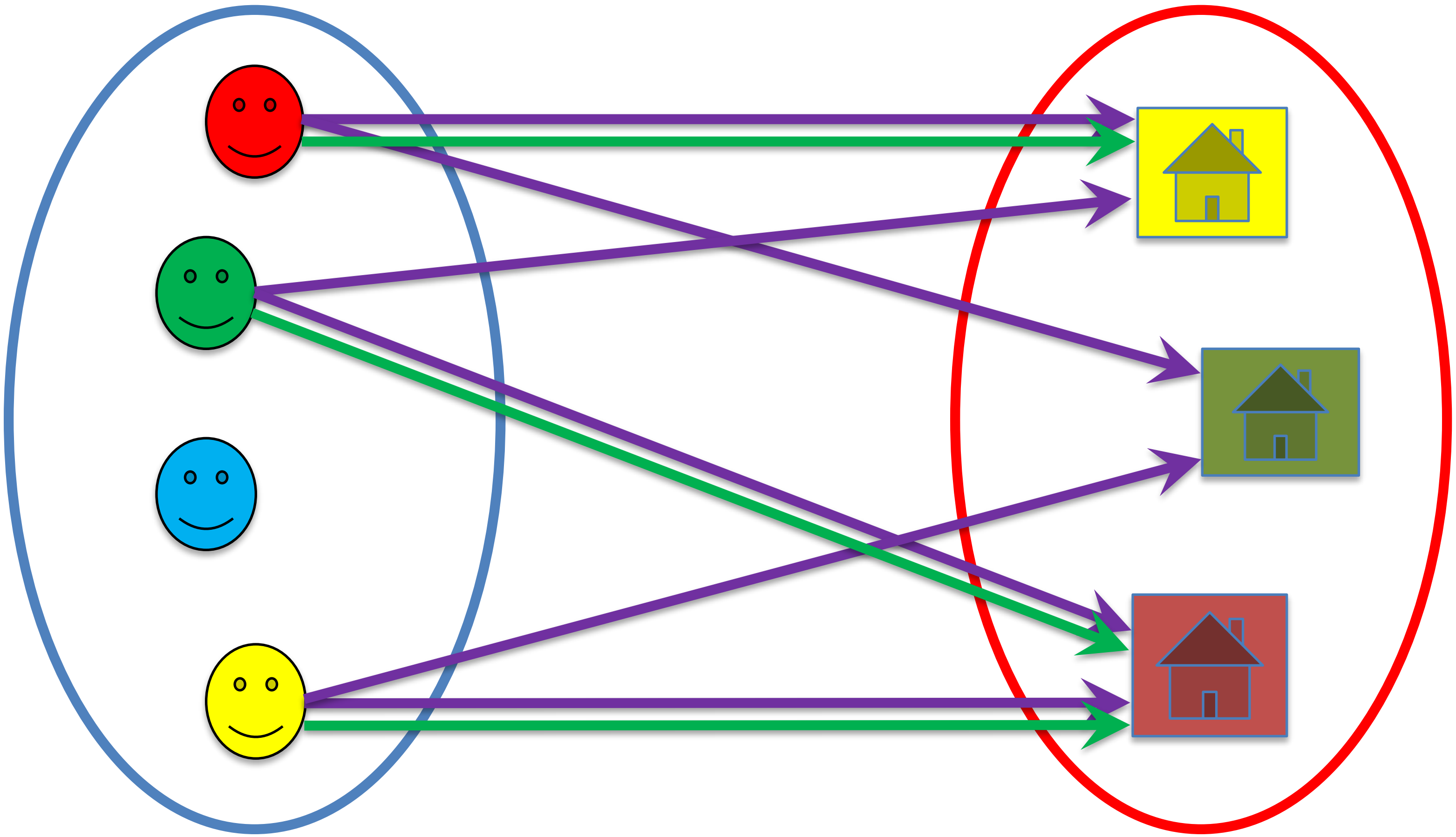
Many-to-many relation

Location



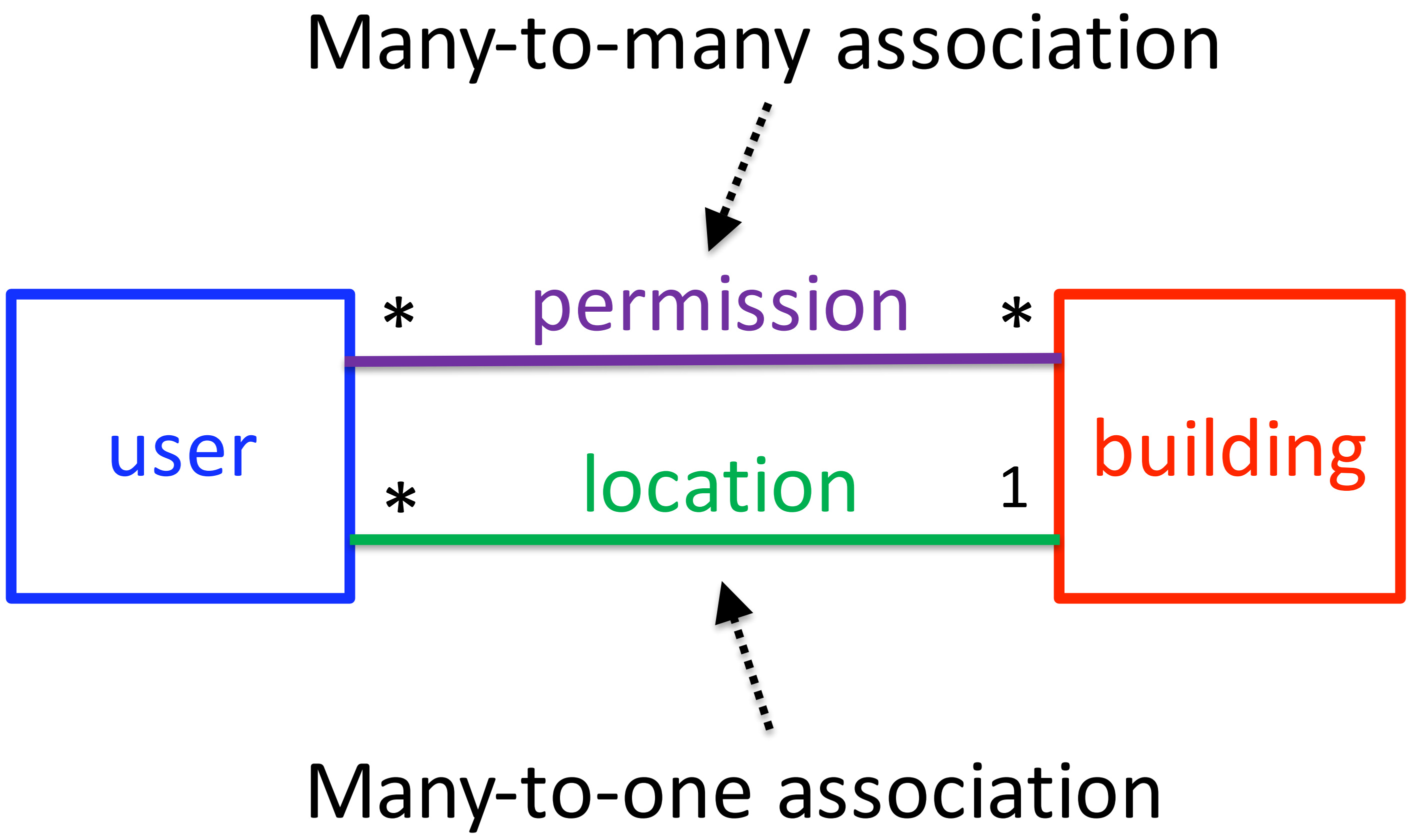
Many-to-one relation

Location conforms to Permission



Location $\subseteq$ Permission

Class diagram abstraction



Ordered Pairs and Cartesian Products

An ordered pair is an element consisting of two parts:

a first part and a second part.

An ordered pair with first part x and second part y is written: $\boxed{x \mapsto y}$

The Cartesian product of two sets is the set of pairs whose first part is in S and second part is in T.

The Cartesian product of S with T is written: $\boxed{S \times T}$

Cartesian Products: Definition and Examples

Defining Cartesian product:

Predicate	Definition
$x \mapsto y \quad \in \quad S \times T$	$x \in S \quad \wedge \quad y \in T$

Examples:

$$\{a, b, c\} \times \{1, 2\} \quad = \quad \{\ a \mapsto 1,\ a \mapsto 2,\ b \mapsto 1,\ b \mapsto 2,\ c \mapsto 1,\ c \mapsto 2\ \}$$

$$\{a, b, c\} \times \{\} \quad = \quad ?$$

$$\{\ \{a\},\ \{a, b\}\ \} \ \times\ \{1, 2\} \quad = \quad ?$$

Cartesian Products: Definition and Examples

Defining Cartesian product:

Predicate	Definition
$x \mapsto y \quad \in \quad S \times T$	$x \in S \quad \wedge \quad y \in T$

Examples:

$$\{a, b, c\} \times \{1, 2\} = \{ a \mapsto 1,\ a \mapsto 2,\ b \mapsto 1,\ b \mapsto 2,\ c \mapsto 1,\ c \mapsto 2 \}$$

$$\{a, b, c\} \times \{\} = \{\}$$

$$\{ \{a\},\ \{a, b\} \} \times \{1, 2\} = \{ \{a\} \mapsto 1,\ \{a\} \mapsto 2,\ \{a, b\} \mapsto 1,\ \{a, b\} \mapsto 2 \}$$

Cartesian Product is a Type Constructor

$S \times T$ is a new type constructed from types S and T.

Cartesian product is the type constructor for ordered pairs.

Given $x \in S$, $y \in T$, we have

$$\boxed{x \mapsto y \quad \in \quad S \times T}$$

$$4 \mapsto 7 \quad \in \quad ?$$

$$\{5, 6, 3\} \mapsto 4 \quad \in \quad ?$$

$$\{\ 4 \mapsto 8,\ 3 \mapsto 0,\ 2 \mapsto 9\ \} \quad \in \quad ?$$

Cartesian Product is a Type Constructor

$S \times T$ is a new type constructed from types S and T.

Cartesian product is the type constructor for ordered pairs.

Given $x \in S$, $y \in T$, we have

$$\boxed{x \mapsto y \quad \in \quad S \times T}$$

$$4 \mapsto 7 \quad \in \quad \mathbb{Z} \times \mathbb{Z}$$

$$\{5, 6, 3\} \mapsto 4 \quad \in \quad \mathbb{P}(\mathbb{Z}) \times \mathbb{Z}$$

$$\{\ 4 \mapsto 8,\ 3 \mapsto 0,\ 2 \mapsto 9\ \} \ \in \quad \mathbb{P}(\mathbb{Z} \times \mathbb{Z})$$

Sets of Order Pairs

A database can be modelled as a set of ordered pairs:

$$\begin{aligned} directory \quad = \quad \{\ & mary \mapsto 287573, \\ & mary \mapsto 398620, \\ & john \mapsto 829483, \\ & jim \mapsto 398620\ \} \end{aligned}$$

directory has type

$$directory \quad \in \quad \mathbb{P}(Person \times PhoneNum)$$

Relations

A relation is a set of ordered pairs.

A relation is a common modelling structure so Event-B has a special notation for it:

$$\boxed{T \leftrightarrow S} \quad = \quad \mathbb{P}(T \times S)$$

So we can write:

$$directory \quad \in \quad Person \leftrightarrow PhoneNum$$

Do not confuse the arrow symbols:

$\leftrightarrow$ combines two sets to form a set.

$\mapsto$ combines two elements to form an ordered pair.

Domain and Range

$$\begin{aligned} \mathit{directory} &= \{\ \mathit{mary} \mapsto 287573, \\ &\quad \mathit{mary} \mapsto 398620, \\ &\quad \mathit{john} \mapsto 829483, \\ &\quad \mathit{jim} \mapsto 398620\ \} \end{aligned}$$

$$\begin{aligned} \mathrm{dom}(\mathit{directory}) &= \{\mathit{mary}, \mathit{john}, \mathit{jim}\} \\ \mathrm{ran}(\mathit{directory}) &= \{287573, 398620, 829483\} \end{aligned}$$

Domain and Range Definition

- The domain of a relation R is the set of first parts of all the pairs in R, written $\boxed{dom(R)}$
- The range of a relation R is the set of second parts of all the pairs in R, written $\boxed{ran(R)}$

Predicate	Definition
$x \in dom(R)$	$\exists y \cdot \; x \mapsto y \;\in\; R$
$y \in ran(R)$	$\exists x \cdot \; x \mapsto y \;\in\; R$

Telephone Directory Model

- Phone directory relates people to their phone numbers.
- Each person can have zero or more numbers.
- People can share numbers.

context *PhoneContext*
sets *Person* *PhoneNum*
end

machine *PhoneBook*
variables *dir*
invariants $dir \in Person \leftrightarrow PhoneNum$

initialisation $dir := \{\}$

Extending the Directory

Add an entry to the directory:

$$
\begin{array}{lll}
AddEntry & \hat{=} & \textbf{any } p, n \textbf{ where} \\
 & & \quad p \in Person \\
 & & \quad n \in PhoneNum \\
 & & \textbf{then} \\
 & & \quad dir := dir \cup \{p \mapsto n\} \\
 & & \textbf{end}
\end{array}
$$

Relational Image

$$\begin{aligned} directory \;=\; \{\; & mary \mapsto 287573, \\ & mary \mapsto 398620, \\ & john \mapsto 829483, \\ & jim \mapsto 398620 \;\} \end{aligned}$$

Relational image examples:

$$directory[\;\{mary\}\;] \;=\; \{\;287573,\;398620\;\}$$

$$directory[\;\{\;john, jim\;\}\;] \;=\; \{\;829483,\;398620\;\}$$

Relational Image Definition

Assume $R \in S \leftrightarrow T$ and $A \subseteq S$

The relational image of set A under relation R is written $\boxed{R[A]}$

Predicate	Definition
$y \in R[A]$	$\exists x \cdot x \in A \ \wedge \ x \mapsto y \in R$

Modelling Queries using Relational Image

Determine all the numbers associated with a person in the directory:

$$\begin{array}{rcl} GetNumbers & \hat{=} & \textbf{any}\ p, result\ \textbf{where} \\ & & \quad p \in Person \\ & & \quad result = dir[\ \{p\}\ ] \\ & & \textbf{end} \end{array}$$

Determine all the numbers associated with a set of people:

$$\begin{array}{rcl} GetMultiNumbers & \hat{=} & \textbf{any}\ ps, result\ \textbf{where} \\ & & \quad ps \subseteq Person \\ & & \quad result = dir[\ ps\ ] \\ & & \textbf{end} \end{array}$$

Partial Functions

Special kind of relation: each domain element has at most one range element associated with it.

To declare f as a partial function:

$$\boxed{f \in X \nrightarrow Y}$$

This says that f is a many-to-one relation

Each domain element is mapped to one range element:

$$x \in dom(f) \implies card(\, f[\{x\}]\,) = 1$$

More usually formalised as a uniqueness constraint

$$x \mapsto y_1 \in f \;\wedge\; x \mapsto y_2 \in f \implies y_1 = y_2$$

Function Application

We can use function application for partial functions.

If $x \in \mathrm{dom}(f)$, then we write $\boxed{f(x)}$ for the unique range element associated with x in f.

If $x \notin \mathrm{dom}(f)$, then $f(x)$ is undefined.

If $card(\ f[\{x\}]\ ) > 1$, then $f(x)$ is undefined.

Examples

$$dir1 = \{ mary \mapsto 398620, jim \mapsto 493028, jane \mapsto 493028 \}$$

$$dir2 = \{ mary \mapsto 287573, mary \mapsto 398620, jane \mapsto 493028 \}$$

$$dir1 \in Person \nrightarrow Phone$$

$$dir1(jim) = 493028$$

$$dir1(sarah) \text{ is undefined}$$

$$dir2 \notin Person \nrightarrow Phone$$

$$dir2(mary) \text{ is undefined}$$

Well-definedness and application definitions

Expression	Well-definedness condition
$f(x)$	$x \in dom(f) \;\wedge\; f \in X \nrightarrow Y$

The following definition of function application assumes that $f(x)$ is well-defined:

Predicate	Definition
$y = f(x)$	$x \mapsto y \in f$

Birthday Book Example

Birthday book relates people to their birthday.

Each person can have at most one birthday.

People can share birthdays.

sets $PERSON \quad DATE$

variables $birthday$
invariants $birthday \in PERSON \nrightarrow DATE$

initialisation $birthday := \{\}$

Adding and checking birthdays

Add an entry to the directory:

$$
\begin{array}{lll}
AddEntry & \widehat{=} & \textbf{any } p, d \textbf{ where} \\
 & & \quad p \in Person \\
 & & \quad p \notin dom(birthday) \\
 & & \quad d \in Date \\
 & & \textbf{then} \\
 & & \quad birthday \;:=\; birthday \cup \{p \mapsto d\} \\
 & & \textbf{end}
\end{array}
$$

Check a person's birthday:

$$
\begin{array}{lll}
Check & \widehat{=} & \textbf{any } p, result \textbf{ where} \\
 & & \quad p \in dom(birthday) \\
 & & \quad result = birthday(p) \\
 & & \textbf{end}
\end{array}
$$

Domain Restriction

Given $R \in S \leftrightarrow T$ and $A \subseteq S$,
the domain restriction of R by A is writen $\boxed{A \lhd R}$

Restrict relation R so that it only contains pairs whose first part is in the set A.

Example:

$$\begin{aligned} \mathit{directory} \;=\; \{\; & \mathit{mary} \mapsto 287573, \;\; \mathit{mary} \mapsto 398620, \\ & \mathit{john} \mapsto 829483, \;\; \mathit{jim} \mapsto 398620 \;\} \end{aligned}$$

$$\begin{aligned} \{\mathit{john}, \mathit{jim}, \mathit{jane}\} \lhd \mathit{directory} \;=\; \{\; & \mathit{john} \mapsto 829483, \\ & \mathit{jim} \mapsto 398620 \;\} \end{aligned}$$

Domain Subtraction

Given $R \in S \leftrightarrow T$ and $A \subseteq S$,
the domain subtraction of R by A is written $\boxed{A \lhd R}$

Remove those pairs from R whose first part is in A.

Example:

$$\begin{aligned} directory \quad = \quad \{ \; & mary \mapsto 287573, \quad mary \mapsto 398620, \\ & john \mapsto 829483, \quad jim \mapsto 398620 \; \} \end{aligned}$$

$$\begin{aligned} \{john, jim, jane\} \lhd directory \quad = \quad \{ \; & mary \mapsto 287573, \\ & mary \mapsto 398620 \; \} \end{aligned}$$

Domain and Range, Restriction and Substraction

Assume $R \in S \leftrightarrow T$ and $A \subseteq S$ and $B \subseteq T$

Predicate	Definition	
$x \mapsto y \in A \lhd R$	$x \mapsto y \in R \land x \in A$	domain restriction
$x \mapsto y \in A \unlhd R$	$x \mapsto y \in R \land x \notin A$	domain subtraction
$x \mapsto y \in R \rhd B$	$x \mapsto y \in R \land y \in B$	range restriction
$x \mapsto y \in R \unrhd B$	$x \mapsto y \in R \land y \notin B$	range subtraction

Removing Entries from the Directory

Remove all the entries associated with a person in the directory:

$$RemovePerson \;\hat{=}\; \begin{array}{l} \textbf{any } p \textbf{ where} \\ \quad p \in Person \\ \textbf{then} \\ \quad dir := \{p\} \lhd\!\!\!- \; dir \\ \textbf{end} \end{array}$$

Remove all the entries associated with a number in the directory:

$$RemoveNumber \;\hat{=}\; \begin{array}{l} \textbf{any } n \textbf{ where} \\ \quad n \in PhoneNum \\ \textbf{then} \\ \quad dir := dir \; -\!\!\!\rhd \{n\} \\ \textbf{end} \end{array}$$

Function Overriding

Override f by g $\boxed{f \oplus g}$

f and g must be partial functions of the same type

Override: replace existing mappings with new ones

$$
\begin{aligned}
dir1 \quad = \quad \{ \; & mary \mapsto 398620, \; john \mapsto 829483, \\
& jim \mapsto 493028, \; jane \mapsto 493028 \; \}
\end{aligned}
$$

$$
\begin{aligned}
dir1 \oplus \{ \; & mary \mapsto 674321, \; jane \mapsto 829483\} \\
= \; \{ \; & mary \mapsto 674321, \; john \mapsto 829483, \\
& jim \mapsto 493028, \; jane \mapsto 829483 \; \}
\end{aligned}
$$

Function Overriding Definition

Definition in terms of function override and set union:

$$f \Leftarrow \{a \mapsto b\} = (\{a\} \lhd f) \cup \{a \mapsto b\}$$

$$f \Leftarrow g = (\mathrm{dom}(g) \lhd f) \cup g$$

Modifying a birthday

Modify an entry in the directory:

$$
\begin{array}{lll}
ModifyEntry & \hat{=} & \textbf{any } p, d \textbf{ where} \\
 & & \quad p \in dom(birthday) \\
 & & \quad d \in Date \\
 & & \textbf{then} \\
 & & \quad birthday \; := \; birthday \lhd\!\!- \{p \mapsto d\} \\
 & & \textbf{end}
\end{array}
$$

Syntactic shorthand:

$$
\begin{array}{lll}
ModifyEntry & \hat{=} & \textbf{any } p, d \textbf{ where} \\
 & & \quad p \in Person \\
 & & \quad d \in Date \\
 & & \textbf{then} \\
 & & \quad birthday(p) \; := \; d \\
 & & \textbf{end}
\end{array}
$$

Adding the domain as an explicit variable

variables $birthday, person$

invariants

$$birthday \in PERSON \pto DATE$$
$$person \subseteq PERSON$$
$$person = dom(birthday)$$

initialisation $birthday := \{\}$ $person := \{\}$

Total Functions

A total function is a special kind of partial function. To declare f as a total function:

$$\boxed{f \ \in \ X \rightarrow Y}$$

This means that f is well-defined for every element in X, i.e., $f \ \in \ X \rightarrow Y$ is shorthand for

$$f \ \in \ X \nrightarrow Y \quad \wedge \quad dom(f) = X$$

Modelling with Total functions

We can re-write the invariant for the birthday book to use total functions:

variables $birthday, person$
invariants

$$person \subseteq PERSON$$
$$birthday \in person \rightarrow DATE$$

Using the total function arrow means that we don't need to explicitly specify that $dom(birthday) = person$.

We can use *person* as a guard instead of $dom(birthday)$:

$$Check \;\hat{=}\; \textbf{any } p, result \textbf{ where } p \in person \quad result = birthday(p) \textbf{ end}$$

AddEntry needs to be modified

Add an entry to the directory:

$$
\begin{array}{lll}
AddEntry & \hat{=} & \textbf{any } p, d \textbf{ where} \\
 & & \quad p \in PERSON \\
 & & \quad p \notin person \\
 & & \quad d \in DATE \\
 & & \textbf{then} \\
 & & \quad birthday := birthday \cup \{p \mapsto d\} \\
 & & \quad person := person \cup \{p\} \\
 & & \textbf{end}
\end{array}
$$

Requirements for a Buildings Access System

- Specify a system that controls access to a collection of buildings.
- Registered users will have access permission to enter certain buildings.
- A user can only enter buildings that they have access permission for.
- The system should keep track of the location of users.
- The system should manage registration and access permission for users.

Types? Variables? Invariants? Events?

Buildings Access System

- Types: $USER$, $BUILDING$
- Variables: $register$, $permission$, $location$
- Invariants:
  - $register \subseteq USER$ //register is a set of users
  - $permission \in USER \leftrightarrow BUILDING$
    // relates users to the buildings they can access
  - $dom(permission) \subseteq register$
    // only register users may have permissions
  - $location \in USER \nrightarrow BUILDING$
    // user is located in at most one building
  - $location \subseteq permission$
    // user located in a building must have permission
    // for that building

Buildings Access System

Events:

- *RegisterUser, DeRegisterUser*
- *AddPermission, RevokePermission*
- *EnterBuilding, LeaveBuilding*