Relations

- Relations allow us to express more complicated interconnections and relationships formally
- A relation R between sets S and T can be represented as a set of pairs (s, t) representing those elements of S and T that are related
- Therefore, a relation between sets S and T is a member of POW($S \times T$)
- Shorthand notation: $S \leftrightarrow T \ \equiv$ POW($S \times T$)
- Relations are often called many-to-many mappings

Relation domain and range

- The *domain* of a relation $R \in S \leftrightarrow T$ is the set of elements of S that are related to something in T
- Relation domain (denoted as $dom(R)$) is defined by $\{x \mid x \in S \wedge \exists y \bullet (y \in T \wedge (x, y) \in R)\}$
- The *range* of a relation $R \in S \leftrightarrow T$ is the set of elements of T that are related to something in S
- Relation range (denoted as $ran(R)$) is defined by $\{y \mid y \in T \wedge \exists x \bullet (x \in S \wedge (x, y) \in R)\}$

Operations on domain and range

$S <\| R$	Restriction of R by S (domain restriction), keep only those pairs whose first element is in S
$S <<\| R$	Anti-restriction of R by S (domain substraction), keep only those pairs whose first element is NOT in S
$R \|> T$	Co-restriction of R by T (range restriction), keep only those pairs whose second element is in T
$R \|>> T$	Anti-co-restriction of R by T (range substraction), keep only those pairs whose second element is NOT in T

Other operations on relations

$\sim R$	Inverse of R. The set $\{(t,s) \| (s,t) \in R\}$.
$R[U]$	Relational image of U by relation $R \in S \leftrightarrow T$. The set consisting of all elements of T related to some element of U by R.
$R_1;\ R_2$	Composition of relations $R_1 \in S \leftrightarrow T$ and $R_2 \in T \leftrightarrow U$. The set $\{(s,u) \| \exists t \bullet (s,t) \in R_1 \wedge (t,u) \in R_2\}$.
$R_1 <+ R_2$	Relational overriding. Relation R_1 is "updated" according to R_2.

Relations on a single set

$id(S)$	Identity relation The set $\{(s,s) \mid s \in S\}$.
$iterate(R,n)$	The nth iteration of R (n $\in NAT$), i.e. R composed with itself n times. Defined only for $R \in S \leftrightarrow S$. $iterate(R,0) \;=\; id(S)$ $iterate(R,n+1) \;=\; (R;\; iterate(R,n))$
$closure(R)$	The reflexive transitive closure of $R \in S \leftrightarrow S$. $closure(R) \;=\; id(S) \cup R \cup R;R \cup R;R;R\ldots$

Relations in abstract machines

- Abstract machine notation allows us to declare variables and constants of a relation type
- Relation variables can be assigned in assignments, though the value assigned must be a relation
- The weakest preconditions for such assignments are calculated in exactly the same way as it is done for simpler types

```
MACHINE Access
SETS
  USER; PRINTER; OPTION;
  PERMISSION = {ok, noaccess}
CONSTANTS options
PROPERTIES
  options ∈ PRINTER ↔ OPTION ∧
  dom(options) = PRINTER ∧
  ran(options) = OPTION
VARIABLES access
INVARIANT
  access ∈ USER ↔ PRINTER
INITIALISATION access := {}
OPERATIONS
  add(uu,pp) =
    PRE uu ∈ USER ∧ pp ∈ PRINTER
    THEN access := access ∪ {(uu,pp)}
    END;
  ...
```



```
  ...
  block(uu,pp) =
    PRE uu ∈ USER ∧ pp ∈ PRINTER
    THEN access := access − {(uu,pp)}
    END;
  ban(uu) =
    PRE uu ∈ USER
    THEN access := {uu} <<| access
    END;
  ans ← optionquery(uu,oo) =
    PRE uu ∈ USER ∧ oo ∈ OPTION
    THEN
      IF (uu,oo) ∈ (access; options)
      THEN ans := ok
      ELSE ans := noaccess
      END
    END;
  nn ← printquery(pp) =
    PRE pp ∈ PRINTER
    THEN nn := card (access |> {pp})
    END
END
```

Deferred sets

- Deferred sets are sets the definitions of which are not yet given
- Deferred sets can be used as machine parameters or as local sets (types) of a machine
- The behaviour of a machine with deferred sets can be described even though the precise form (implementation) of such sets will be decided later

Deferred sets as parameters

```
MACHINE Store(ITEM)
VARIABLES elements
INVARIANT elements ⊆ ITEM
INITIALISATION elements := {}
OPERATIONS
  input(ii)=
    PRE ii ∈ ITEM
    THEN elements := elements ∪ {ii}
    END;
    ...
END
```

The exact definition (implementation) of ITEM is not yet known

Deferred sets as local types

```
MACHINE Bank
SETS
  NAMES; ACCOUNTS
VARIABLES
  accounts, customers
INVARIANT
  accounts ⊆ ACCOUNTS ∧
  customers ⊆ NAMES ∧ ...
INITIALISATION
  accounts,customers := {}, {}
OPERATIONS
  ...
END
```

The exact definitions (implementations) of NAMES and ACCOUNTS will be decided later