ERA diagram semantics in VDM-SL

2000/01

J.N. Oliveira

jno@di.uminho.pt

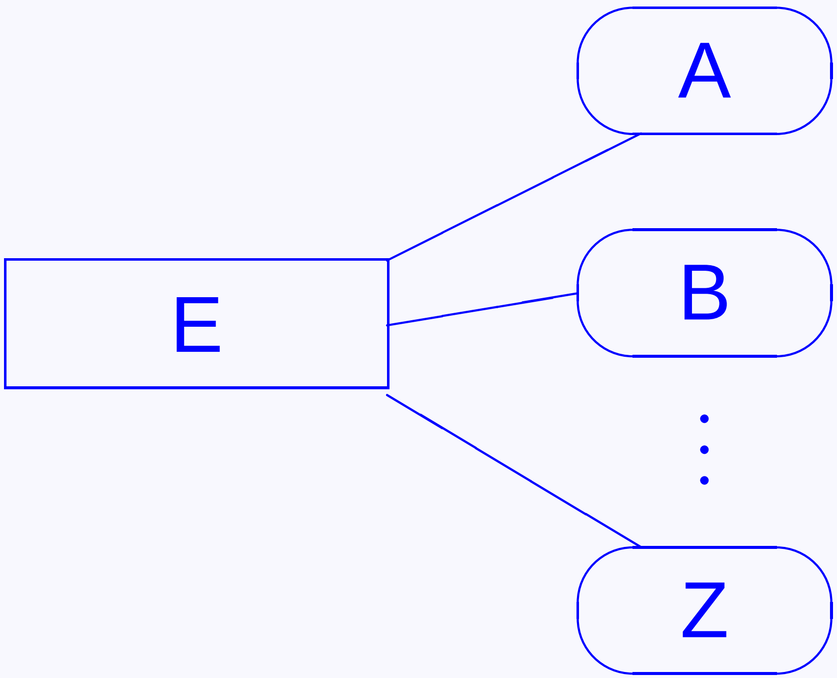
Dept. Informática, Universidade do Minho

ER-diagram Cookbook

- A quick reference list of rules for converting **ER-diagrams** into **VDM-SL** notation is provided.
- Symbols E, F denote entities
- Symbols $\#E$, $\#F$ denote the relevant key attributes
- Symbols A, B, C, D, G, $\cdots$, Z denote attribute domains.

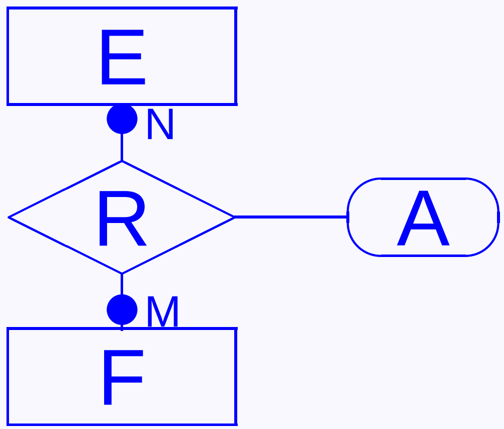
Entities

ER	**VDM-SL**

	Define
	`map #E to E;`
	`E :: a: A`
	`     b: B`
	`     .`
	`     .`
	`     .`
	`     z: Z;`

M:M Relationships

ER



VDM-SL

Define

```
#EF :: ek: #E
       fk: #F;
```

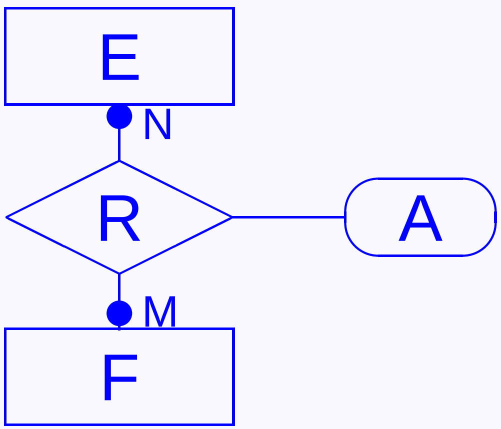
in

```
ER :: e: map #E to E;
      f: map #F to F;
      r: map #EF to A
inv mk_ER(e,f,r) ==
    { k.ek | k in set dom r }
               subset dom e and
    { k.fk | k in set dom r }
               subset dom f;
```

M:M Relationship improved

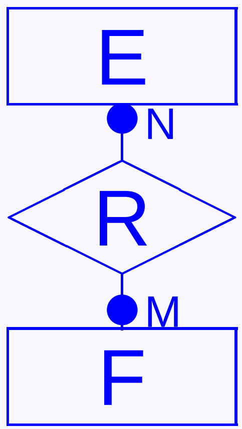
Can be made simpler by merging `r` with either e or `f`, *e.g.*:

ER	**VDM-SL**

	Define
	`Ex :: e: E`
	`      r: map #F to A;`
	in
	`ER :: e: map #E to Ex;`
	`      f: map #F to F;`
	`inv mk_ER(e,f) ==`
	`    forall x in set rng e &`
	`      dom(x.r) subset dom f;`

M:M without attribute

ER	VDM-SL

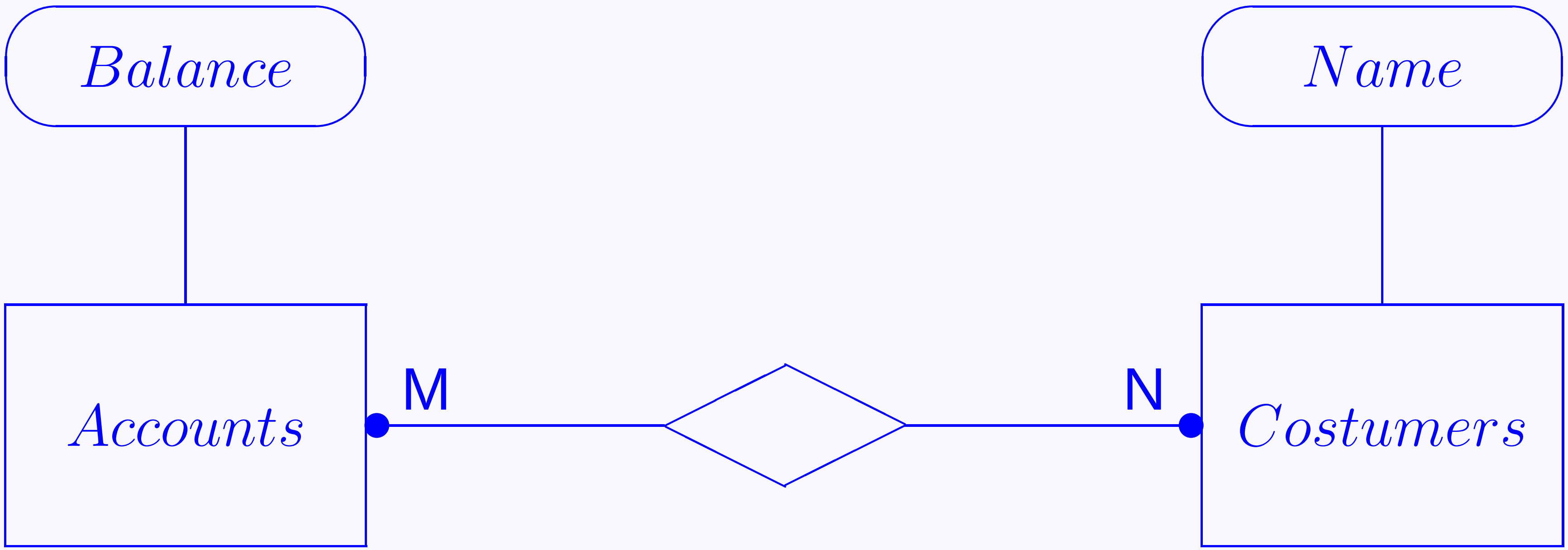
	Define
	`Ex :: e: E`
	`      r: set of #F;`
	in
	`ER :: e: map #E to Ex;`
	`      f: map #F to F`
	`   inv mk_ER(e,f) ==`
	`      forall x in set rng e &`
	`             x.r subset dom f;`

Reasoning

$$
\begin{aligned}
& Ex = E \times (\#F \rightharpoonup A) \\
\cong \quad & \{\, A = 1\} \\
& Ex = E \times (\#F \rightharpoonup 1) \\
\cong \quad & \{\, A = 1\} \\
& Ex = E \times \mathcal{P}(\#F)
\end{aligned}
$$

M:M Relationship example

A naïve “bank account management system”:



M:M example (VDM-SL)

VDM-SL-equivalent model:

```
ER :: e: map AccountId to AccountInf
      f: map CostumerId to Name
   inv mk_ER(e,f) == forall x in set rng e &
       x.r subset dom f;

AccountInf :: a: Balance
              r: set of CostumerId ;
```

Informal meaning of `inv mk_ER`:

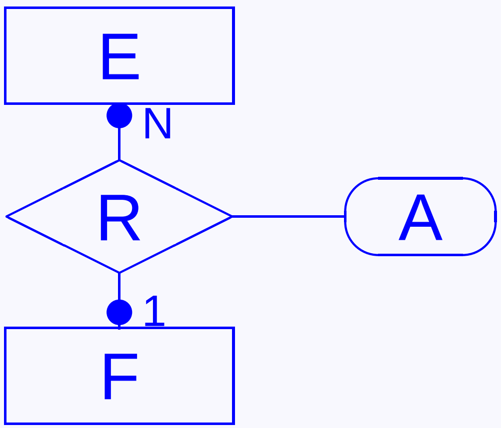
The information of each account can only refer to known bank costumers.

M:1 Relationships

One F at most is related to a given E, so *`map #F to A`* “shrinks” to an optional pair:

ER	VDM-SL

ER



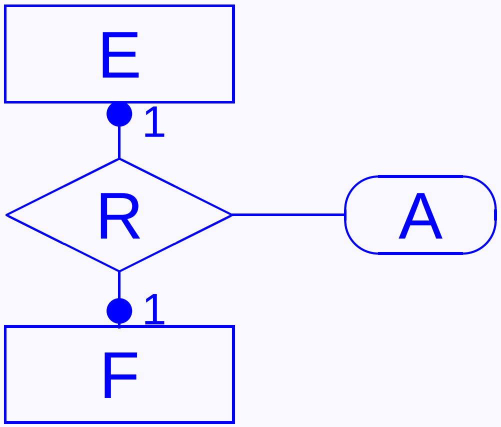
VDM-SL

```
Ex :: e: E
      r: [ R ];

R  :: f: #F
      a: A;

ER :: e: map #E to Ex;
      f: map #F to F;
inv mk_ER(e,f) ==
    let g = { k |-> e(k).r.f |
                k in set dom e & is_R(e(k).r) }
    in rng g subset dom f;
```

1:1 Relationships

Further to M:1, relationship has to be injective:

ER	**VDM-SL**

	`inv mk_ER(e,f) ==`
	`    let g = { k |-> e(k).r.f |`
	`                k in set dom e & is_R(e(k).r) }`
	`    in rng g subset dom f and injective(g);`

Auxiliary predicate “injective”

Recall

- R is injective iff *ker* $R \subseteq id$
- when R is simple (partial function), then

$$a(\textit{ker}\ R)b \;\equiv\; a, b \in \textit{dom}\ R \wedge R\,a = R\,b$$

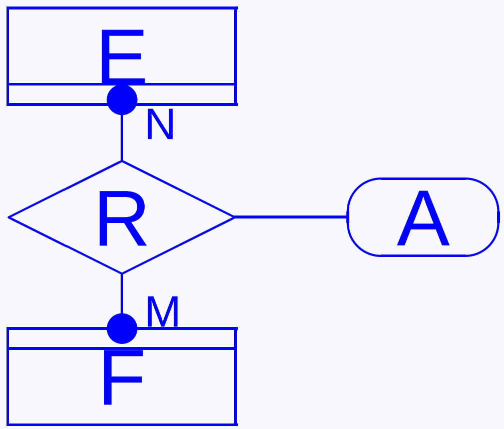
- So *ker* $R \subseteq id$ means $\forall a, b \in \textit{dom}\ R.(R\,a = R\,b \Rightarrow a = b)$

Thus:

```
injective[@A,@B] : map @A to @B -> bool
injective(f)==
        forall a,b in set dom f &
               f(a)=f(b) => a=b
```

Compulsory relationships

ER	VDM-SL

	Define
	`#EF :: ek: #E`
	`       fk: #F;`
	in
	`ER :: e: map #E to F;`
	`      f: map #F to F;`
	`      r: map #EF to A`
	`inv mk_ER(e,f,r) ==`
	`    { k.ek | k in set dom r }`
	`               = dom e and`
	`    { k.fk | k in set dom r }`
	`               = dom f;`

Compulsory rel. semantics

At invariant level, `subset` gives place to set-theoretic equality “=”, that is:

```
s = r == s subset r and r subset s
```