ERA diagram semantics in VDM-SL

2000/01

J.N. Oliveira

jno@di.uminho.pt

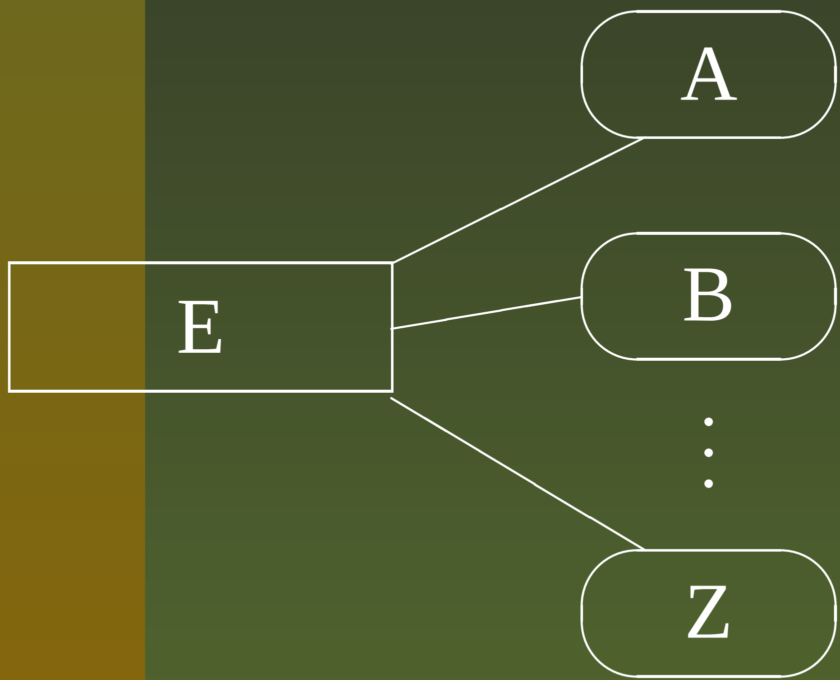
Dept. Informática, Universidade do Minho

ER-diagram Semantics 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

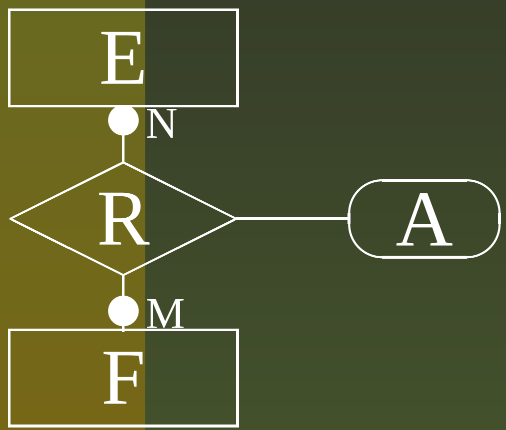
ER	VDM-SL



Define

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

M:M Relationships

ER	VDM-SL
	Define

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

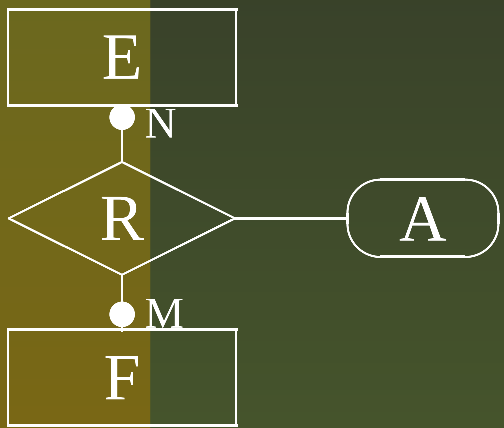
in

```
ER :: e: map #E to E;
      f: map #F to F;
      r: map #EF to A
inv 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

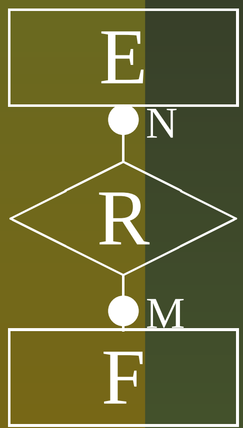
Can be made simpler by merging `r` with either `e` or `f`, *e.g.*:

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 ER(e,f) ==`
	`    forall x in set rng e &`
	`       dom(x.r) subset dom f;`

M:M Relationship 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 ER(e,f) ==
       forall x in set rng e &
              x.r subset dom f;
```

Reasoning

$$Ex = E \times (\#F \rightharpoonup A)$$

$$\cong \quad \{\, A = 1\}$$

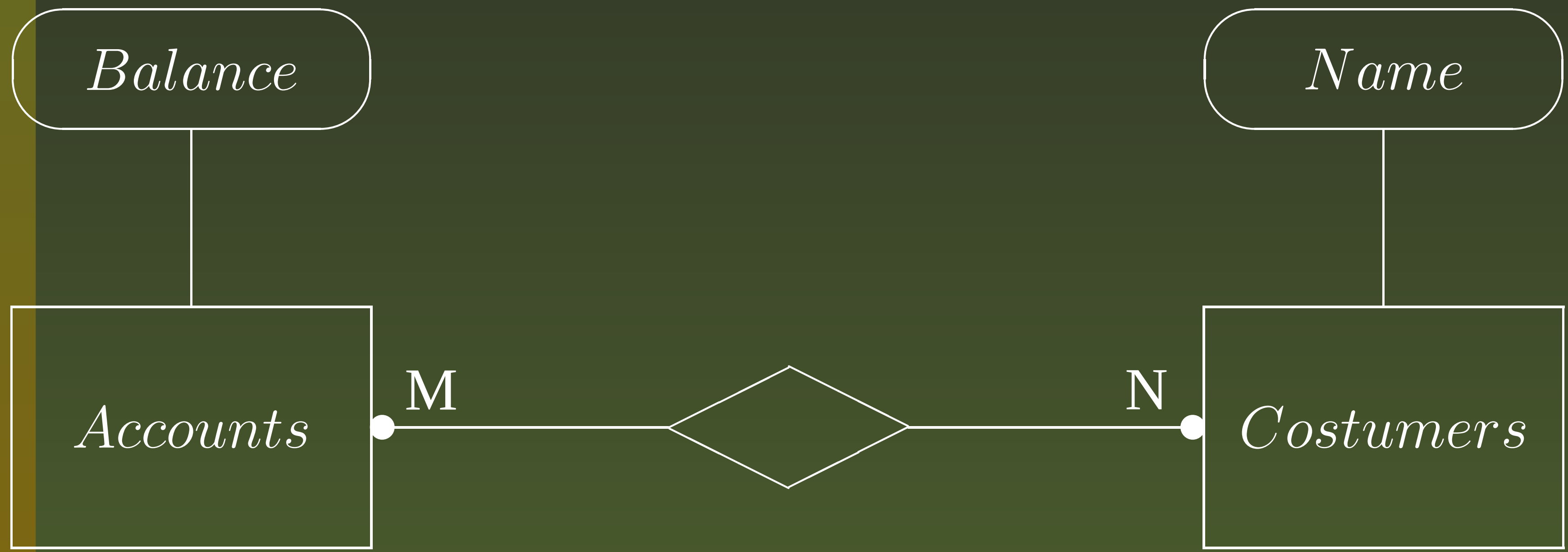
$$Ex = E \times (\#F \rightharpoonup 1)$$

$$\cong \quad \{\, A = 1\}$$

$$Ex = E \times \mathcal{P}(\#F)$$

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 ER(e,f) == forall x in set rng e &
         x.r subset dom f;

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

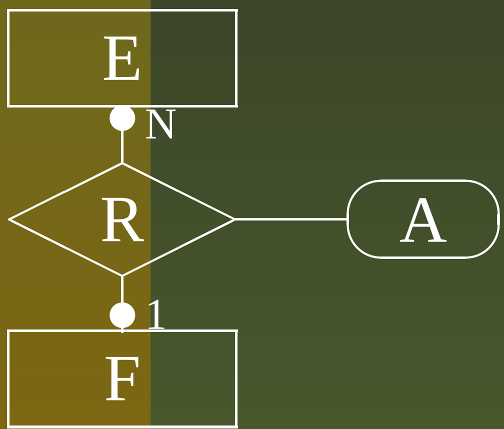
Informal meaning of `inv ER`:

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

M:1 Relationships

Only one F can (at most) be related to a given E, so *map #F to A* "shrinks" to an optional pair:

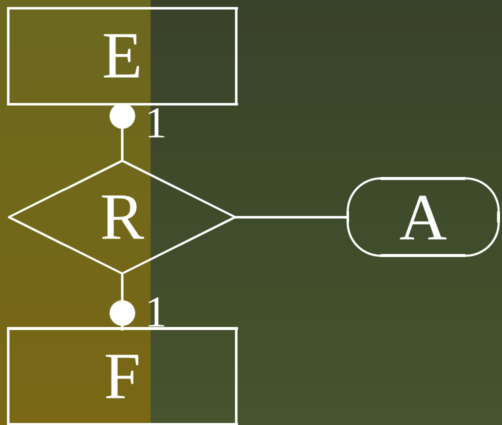
ER	VDM-SL

	`Ex :: e: E`
	`      r: [ R ];`
	
	`R  :: f: #F`
	`      a: A;`
	in
	`ER :: e: map #E to Ex;`
	`      f: map #F to F;`
	`inv ER(e,f) ==`
	`    { x.r.f | x in set rng e & is_R(x.r) } sub-`
	`set dom f;`

1:1 Relationships

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

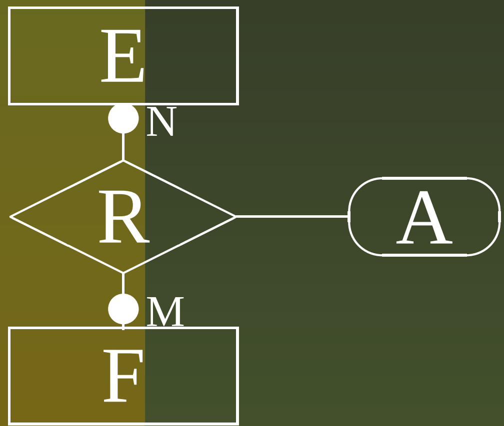
ER	VDM-SL

	`Ex :: e: E`
	`      f: #F`
	`      a: A;`
	in
	`ER :: e: map #E to Ex;`
	`      f: map #F to F;`
	`inv ER(e,f) ==`
	`    { x.f | x in set rng e} subset dom f and`
	`    injective({ k |-> e(k).f | k in dom e });`

Auxiliary predicate “injective”

```
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

<table>
<tr><th>ER</th><th>VDM-SL</th></tr>
<tr><td>
</td><td>

Define

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

in

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

</td></tr>
</table>

Compulsory relationship semantics

At invariant level, `subset` gives place to set-theorectic equality "=", that is:

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