Structuring with SEES and USES

- The SEES and USES structuring mechanisms permit read-only access between machines
- A machine can be accessed by a number of other machines (shared access).
- This allows some parts of the state or operations to be expressed as a separate machine if many other machines (components) require knowledge of that part of the state
- In contrast, INCLUDES and EXTENDS mechanisms allow only exclusive access to an inluded machine

The SEES relationship

- MACHINE M2
  SEES M1
- M2 is provided read access to another machine M1
- Sets, constants, and variables of M1 are visible in M2 (read access)
- Invariant of M2 can refer to M1 sets and constants but not M1 variables. Since M1 is not under control of M2, M1 variables can be changed independently of M2
- Only query operations of M1 can be called from M2

The SEES relationship (cont.)

- When M2 sees M1, they are both considered as distinct machines (M1 is not part of M2 description as in the INCLUDES case)
- SEES relation is not transitive, i.e. M2 does not automatically see machines that are seen by M1
- On the other hand, if M2 sees M1, it also sees any machines that M1 includes

Using SEES

SEES is especially useful when

- definition of some deferred or enumerated set (type) should be widely available
- some part of state is required by a number of other machines (components)

The USES relationship

- MACHINE M2
  USES M1
- M2 is provided read access to another machine M1 in the same way as SEES does
- In addition, USES allows M2 invariant to refer to the variables of M1. Thus, the relationship between M2 state and M1 state can be expressed
- Since M2 does not control M1, there should be another machine M3 (which includes both M2 and M1) that guarantees that the M2 invariant is preserved

Machine Composition Mechanisms

	CONSTANTS and SETS	VARIABLES	OPERATIONS	Access	Transitivity
A SEES B	visible	visible in operations not in Inv.	only enquiry operations are visible	shared	no
A USES B	visible	visible	only enquiry operations are visible	shared	no
A INCLUDES B	visible	visible	can be called, cannot be exported	exclusive	yes
A EXTENDS B	visible	visible	can be called and exported	exclusive	yes

```
MACHINE Goods
SETS GOODS
END

MACHINE Price
SEES Goods
VARIABLES price
INVARIANT price ∈ GOODS → NAT1
INITIALISATION
   price :∈ GOODS → NAT1
OPERATIONS
   setprice(gg, pp) =
     PRE
        gg ∈ GOODS ∧ pp ∈ NAT1
     THEN price(gg) := pp
     END;
   pp ← pricequery(gg) =
     PRE gg ∈ GOODS
     THEN pp := price(gg)
     END
END
```

```
MACHINE Shop
SEES Price, Goods
VARIABLES takings
INVARIANT takings ∈ NAT
INITIALISATION
   takings := 0
OPERATIONS
   pp ← sale(gg) =
     PRE
        gg ∈ GOODS
     THEN
        takings := takings + price(gg)  ‖
        pp ← pricequery(gg)
     END;
   tt ← total =
     BEGIN
        tt := takings
     END
END
```

```
MACHINE Life
SETS
   PERSON; SEX = {boy, girl}
VARIABLES male, female
INVARIANT
   male ⊆ PERSON ∧ male ⊆ PERSON  ∧
   male ∩ female = {}
INITIALISATION
   male,female := {}, {}
OPERATIONS
   born(nn, ss) =
     PRE
        nn ∈ PERSON  ∧  ss ∈ SEX  ∧
        nn ∉ male∪female
     THEN
        IF ss = boy
        THEN male := male ∪ {nn}
        ELSE female := female ∪ {nn}
        END
     END;
     ...
```

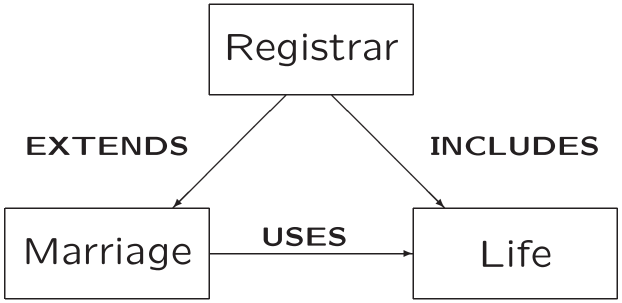
```
     ...
   die(nn) =
     PRE
        nn ∈ PERSON  ∧
        nn ∈ male∪female
     THEN
        IF nn = male
        THEN male := male − {nn}
        ELSE female := female − {nn}
        END
     END
END
```

```
MACHINE Marriage
USES Life
VARIABLES marriage
INVARIANT
   marriage ∈ male ⤔ female
INITIALISATION
   marriage := {}
OPERATIONS
   wed(mm, ff) =
     PRE
        mm ∈ male ∧ mm ∉ dom(marriage) ∧
        nn ∈ female ∧ nn ∉ ran(marriage)
     THEN marriage(mm) := ff
     END;
   part(mm, ff) =
     PRE
        mm ∈ male ∧ ff ∈ female ∧
        (mm,ff) ∈ marriage
     THEN marriage := marriage − (mm,ff)
     END;
     ...
```

```
     ...
   pp ← partner(nn) =
     PRE
        nn ∈ dom(marrige)∪ran(marriage)
     THEN
        IF nn ∈ dom(marriage)
        THEN pp := marriage(nn)
        ELSE pp := ∼marriage(nn)
        END
     END
END
```

```
MACHINE Registrar
EXTENDS Marriage
INCLUDES Life
PROMOTES born
OPERATIONS
   dies(nn) =
     PRE
        nn ∈ male ∪ female
     THEN
        die(nn) ‖
        IF nn ∈ dom(marriage)
        THEN part(nn, marriage(nn))
        ELSE part(∼marriage(nn), nn)
        END
     END
END
```

Structure of Registrar specification


Registrar
EXTENDS
INCLUDES
Marriage
USES
Life