### Refinement

- Refinement is the process of moving <u>from abstract</u> specifications <u>to less abstract</u> specifications <u>while</u> <u>preserving the behaviour</u> of the former
- Refinement is based on data or operation transformation which allows the behaviour of the abstract system to be <u>simulated</u> by the more refined (concrete, detailed) system
- Two kinds of refinement:
  - data refinement
  - refinement of nondeterminism

A Specification Methods 13.02.2006

## Refinement (cont.)

- Intermediate specifications are presented in REFINEMENT components
- A REFINES clause identifies the component that this refinement is refining
- The abstract component M and the concrete component N are two alternative descriptions of <u>the same</u> system, with <u>the same external interface</u>, but with different internal implementations

### Refinement (cont.)

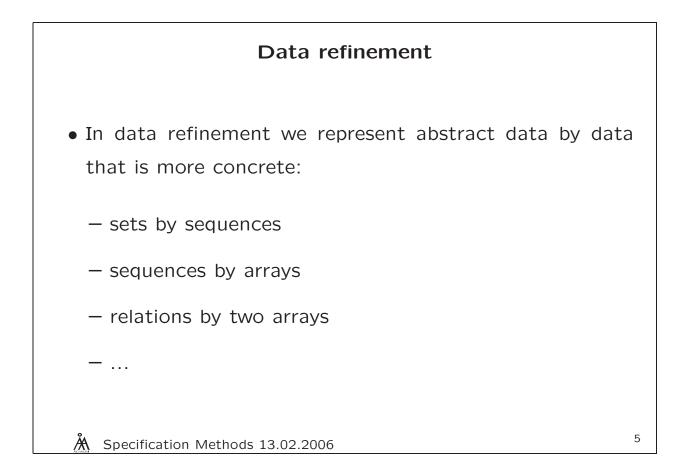
- The states of the abstract machine M and the concrete refinement N must be composed of distinct sets of variables
- These states are linked by a <u>refinement relation</u> which expresses the correspondence between the abstract and concrete states
- This correspondence must be
  - established as a result of corresponding initialisations
  - preserved by each of the operations

A Specification Methods 13.02.2006

#### Sequential composition

- In addition to the AMN statements introduced so far, in refinements we are allowed to use sequential composition of several statements
- The syntax: S; T
- The standard meaning the statement T is executed after the statement S (if the latter terminates)
- The weakest precondition rule:

[S; T] P = [S] ([T] P)



# • In refinement of nondeterminism • In refinement of nondeterminism we resolve nondeterminism presented in a specification by describing how the choice is to be made. Examples: $-e :\in S \subseteq (\text{is refined by}) \quad e := \min(S)$ $- \text{ CHOICE } S1 \text{ OR } S2 \text{ END } \subseteq S1$ $- \text{ PRE } P \text{ THEN } S \text{ END } \subseteq S1$ $- \text{ PRE } P \text{ THEN } S \text{ END } \subseteq$ IF P THEN S ELSE skip END $- \text{ ANY } zz \text{ WHERE } P \text{ THEN } S \text{ END } \subseteq$ ANY $zz \text{ WHERE } P \wedge Q \text{ THEN } S \text{ END}$

```
MACHINE Team
SETS ANSWER = \{in, out\}
VARIABLES team
INVARIANT team \subset 1..22 \land card(team) = 11
INITIALISATION team := 1..11
OPERATIONS
  substitute(pp, rr) =
    PRE
     pp \in team \land rr \in 1..22 \land rr \notin team
   THEN
     team := (team \cup {rr}) - {pp}
    END;
  aa \leftarrow query(pp) =
    PRE pp ∈ 1..22
   THEN
     IF pp \in team THEN aa := in
     ELSE aa := out END
    END;
END
```

7

A Specification Methods 13.02.2006

**REFINEMENT** TeamR **REFINES** Team VARIABLES teamr INVARIANT teamr  $\in$  1..11  $\rightarrow$  1..22  $\wedge$ ran(teamr) = teamINITIALISATION teamr := id(1..11)**OPERATIONS** substitute(pp, rr) =BEGIN teamr (~teamr(pp)) := rr END;  $aa \leftarrow query =$ IF pp  $\in$  ran(teamr) THEN aa := in ELSE aa := out END END

8

Specification Methods 13.02.2006

```
REFINEMENT TeamR
REFINES Team
VARIABLES teama
INVARIANT
  teama \in 1..22 \rightarrow ANSWER \land
  team = \simteama [{in}]
INITIALISATION
  teama := (1..11) \times \{in\} \cup (12..22) \times \{out\}
OPERATIONS
  substitute(pp, rr) =
    BEGIN
     teama(pp) := out;
     teama(rr) := in
    END;
  aa \leftarrow query =
    BEGIN
     aa := teama(pp)
    END
END
A Specification Methods 13.02.2006
```

```
REFINEMENT JukeboxR
REFINES Jukebox
CONSTANTS freefreq
PROPERTIES freefreq \in NAT1
VARIABLES creditr, playlist, free
INVARIANT
  creditr \in NAT \land creditr = credit \land
  playlist \in iseq(TRACK) \land
  ran(playlist) = playset \land
  free \in 0..freefreq
INITIALISATION
  creditr, playlist := 0, \{\}
OPERATIONS
  pay(cc) =
    BEGIN
     creditr := creditr + cc
    END;
  ...
                                               10
Specification Methods 13.02.2006
```

```
select(tt) =
    BEGIN
     IF tt ∉ ran(playlist)
     THEN playlist := playlist \leftarrow tt
     END;
     IF free = freefreq
     THEN
       free := 0
     ELSE
       free := free + 1;
       creditr := creditr -1
     END
    END;
  tt \leftarrow play =
    BEGIN
     tt := first(playlist);
     playlist := tail(playlist)
    END
END
```

```
Specification Methods 13.02.2006
```