



## ***C340 Concurrency: Modelling Concurrency in FSP***

***Wolfgang Emmerich***



## ***What do we have to model?***

- ***Relative or absolute speed?***
  - ***Neither!***
- ***Concurrency or parallelism?***
  - ***Interleaved model of concurrency!***
- ***Relative order of actions?***
  - ***Arbitrary interleaving!***
- ***We use an asynchronous model of execution!***



## FSP: Parallel Composition

- If  $P$  and  $Q$  are processes then  $(P \parallel Q)$  denotes the parallel execution of  $P$  and  $Q$
- $\parallel$  is used to model parallel composition of processes
- Names of concurrent processes are preceded by  $\parallel$
- **Example:**

CONVERSE = (think->talk->STOP).

ITCH = (scratch->STOP).

$\parallel$  CONVERSE\_ITCH = (ITCH  $\parallel$  CONVERSE).

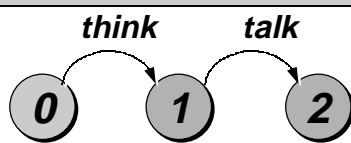
© Wolfgang Emmerich, 1999

3



## Equivalent LTSs

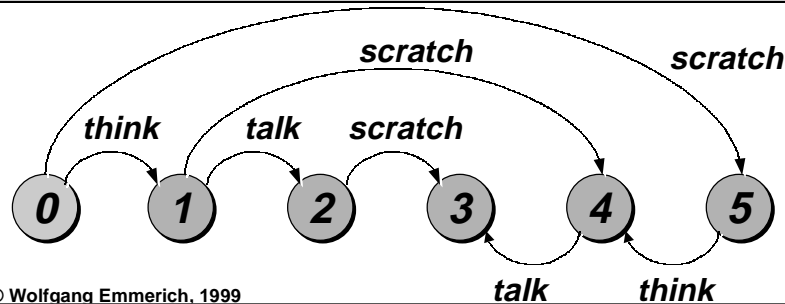
CONVERSE = (think->talk->STOP).



ITCH = (scratch->STOP).



$\parallel$  CONVERSE\_ITCH = (ITCH  $\parallel$  CONVERSE).



© Wolfgang Emmerich, 1999

4



## *Properties of Parallel Composition*

- *Parallel composition operator has two important algebraic properties*
- *Commutativeness*
  - $(P \mid \mid Q) = (Q \mid \mid P)$
  - *ordering is not important!*
- *Associativeness*
  - $((P \mid \mid Q) \mid \mid R) = (P \mid \mid (Q \mid \mid R)) = (P \mid \mid Q \mid \mid R)$
  - *brackets can be omitted!*



## *FSP: Process Interactions*

- *Concurrent processes that share actions interact with each other*
- *Used to model synchronisation*
- *Example:*  
MAKER = (make->ready->MAKER).  
USER = (ready->use->USER).  
||MAKER\_USER = (MAKER || USER).
- *Product has to be ready before it can be used.*

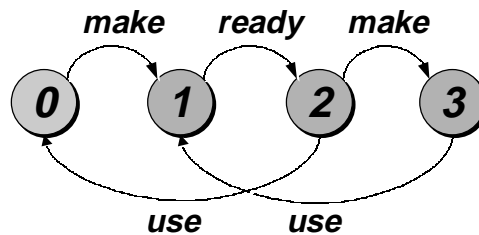


## Equivalent LTS

```

MAKER = (make->ready->MAKER) .
USER  = (ready->use->USER) .
|| MAKER_USER = (MAKER || USER) .

```



Demo

© Wolfgang Emmerich, 1999

7



## Handshake

- An action that is acknowledged by another action is referred to as handshake
- Widely used to structure process interactions
- Example:

```

MAKERv2=(make->ready->used->MAKERv2) .
USERv2 =(ready->use->used ->USERv2) .
|| MAKER_USERv2 = (MAKERv2 || USERv2) .

```

- LTS:



© Wolfgang Emmerich, 1999

8



## FSP: Process Labelling

- **The process label**  $a:P$  **prefixes each label in the alphabet of  $P$  with  $a$**
- **Avoids name clashes in different instantiations of processes and enables reuse.**
- **Example:**  
SWITCH = (on->off->SWITCH).  
|| TWOSWITCH=(a:SWITCH || b:SWITCH).  
■ **Alphabet of || TWOSWITCH:**  
{a.on, a.off, b.on, b.off}

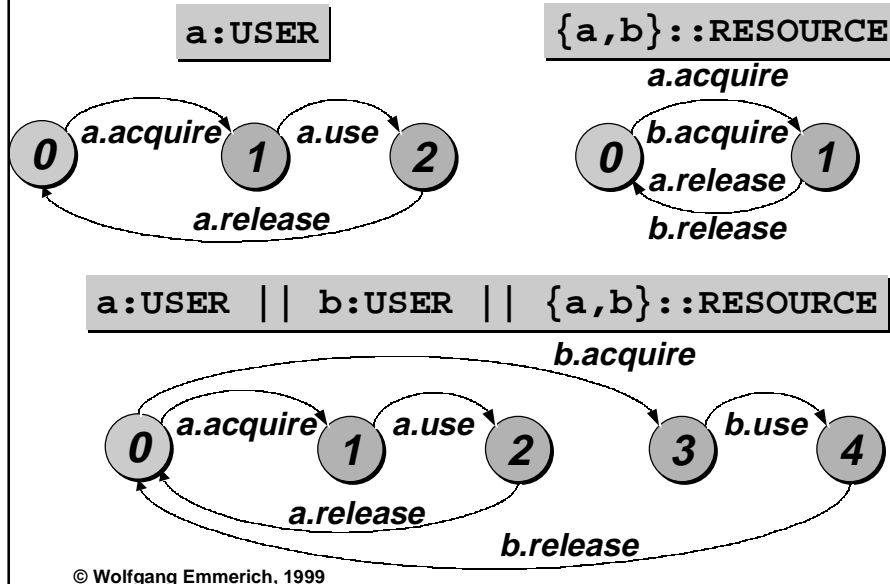


## FSP: Process Labelling (cont'd).

- **The process label**  $\{a_1, \dots, a_x\}::P$  **replaces every label  $n$  in the alphabet of  $P$  with label  $a_1.n, \dots, a_x.n$ .**
- **Example:**  
RESOURCE=(acquire->release->RESOURCE).  
USER = (acquire->use->release->USER).  
|| RESOURCE\_SHARE =  
(a:USER || b:USER || {a,b}::RESOURCE).



## Equivalent LTSs



11



## FSP: Relabelling

- **Relabelling functions** change names of action labels. The relabelling function is:  $/\{\text{new1}/\text{old1}, \dots, \text{newn}/\text{oldn}\}$ .
- Used to synchronise actions with different names in composite processes.
- **Example:**  
`CLIENT=(call->wait->continue->CLIENT).`  
`SERVER=(request->serve->reply->SERVER).`  
`||CLIENT_SERVER = (CLIENT || SERVER)`  
`/{call/request, reply/wait}.`

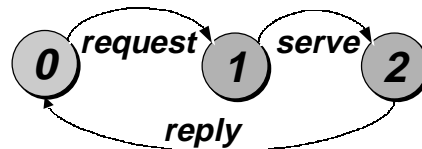
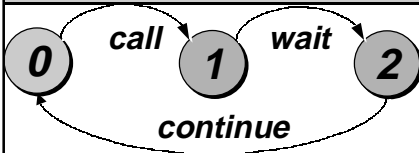
© Wolfgang Emmerich, 1999

12



## Equivalent LTSs

```
CLIENT=(call->wait->
continue->CLIENT).
```



```
SERVER=(request->serve
->reply->SERVER).
```

```
|| CLIENT_SERVER = (CLIENT || SERVER)
/{call/request, reply/wait}.
```



© Wolfgang Emmerich, 1999

13



## FSP: Hiding

- The hiding operator  $\setminus \{a_1..a_x\}$  removes action labels  $a_1..a_x$  from alphabet of  $P$  and hides them
- Hidden actions are labelled  $\tau$
- Hidden actions in different processes are not shared
- **Example:**

```
USER=(acquire->use->release->
USER) \ {use}.
```

© Wolfgang Emmerich, 1999

14



## FSP: Interfaces

- The **interface operator**  $@\{a_1 \dots a_n\}$  **hides all actions in the alphabet of  $P$  that do not occur in the set  $a_1 \dots a_n$ .**
- **Complementary to hiding**
- **Like hiding used to reduce complexity of resulting LTS.**
- **Example:**  

$$\text{USER} = (\text{acquire} \rightarrow \text{use} \rightarrow \text{release} \rightarrow \text{USER}) @ \{\text{acquire}, \text{release}\}.$$

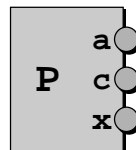
© Wolfgang Emmerich, 1999

15

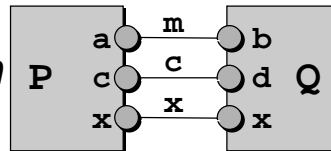


## FSP: Structure Diagrams

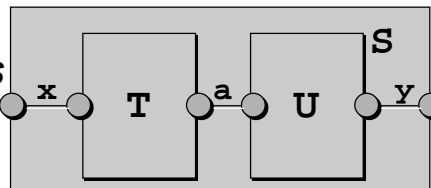
**Process  $P$  with  
alphabet  $\{a, c, x\}$**



**Parallel Composition  
 $(P \parallel Q) / \{m/a, m/b, c/d\}$**



**Composite Process  
 $||S = (T \parallel U) @ \{x, y\}$**



© Wolfgang Emmerich, 1999

16





## Summary

- *Parallel Composition*
- *Process Interactions*
- *Process Labelling*
- *Process Relabelling*
- *Hiding / Interfaces*
- *Structure Diagrams*
- *Next session: Tutorial on FSP modelling*
- *Solve Exercises 3 and 4 of tutorial sheet*