Notes on specifying systems in EST

Robert Meolic, Tatjana Kapus

Faculty of EE & CS

University of Maribor

Proceedings of the Fifteenth International Electrotechnical and Computer Science Conference (ERK 2006), Portorož, Slovenia

volume B
pages 23-26

1 Outline

- ☞ **formal verification**: prove the correctness of system behaviour,
- ☞ **usage:** software, hardware, communication protocols, etc.,
- ☞ **requires:** formal specification of system behaviour,
- ☞ **requires:** formal specification of correct behaviour,
- ☞ **requires:** methods and algorithms (e.g. model checking).

In our paper we discuss the formalism for specification of system behaviour used in verification tool **EST**.

The formalism is based on well-known calculus CCS.

2 Introduction

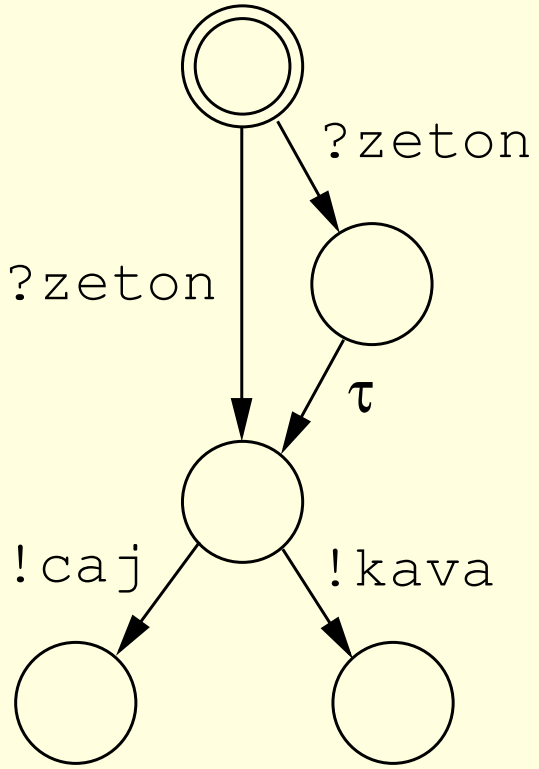
EST is a tool for formal verification of systems. A system to be verified should be specified in a CCS-like syntax. Before verification, specifications are transformed into LTSs.

The EST specifications use operators which can be classified into two groups:

- standard CCS operators and
- additional operators which are introduced to shorten specifications and to facilitate traslations from other formalisms.

3 Labelled transition system

An **LTS** $\mathcal{M}$ is a quadruple $(\mathcal{S}, \mathcal{A}_\tau, \delta, s_0)$:



- $\mathcal{S}$ is a non-empty set of states;
- $\mathcal{A}_\tau$ is a set of actions containing unobservable action τ;
- $\delta \subseteq \mathcal{S} \times \mathcal{A}_\tau \times \mathcal{S}$ is the transition relation;
- s_0 is the initial state.

4 Example 1: Peterson's algorithm

```
PROCESS P1
while (true) {
  <noncriticial section>
  b1=true;
  k=2;
  while (b2==true && k==2)
  {
    wait;
  }
  <criticial section>
  b1=false;
}
```

```
PROCESS P2
while (true) {
  <noncriticial section>
  b2=true;
  k=1;
  while (b1==true && k==1)
  {
    wait;
  }
  <criticial section>
  b2=false;
}
```

5 Example 1: Model with LTSs

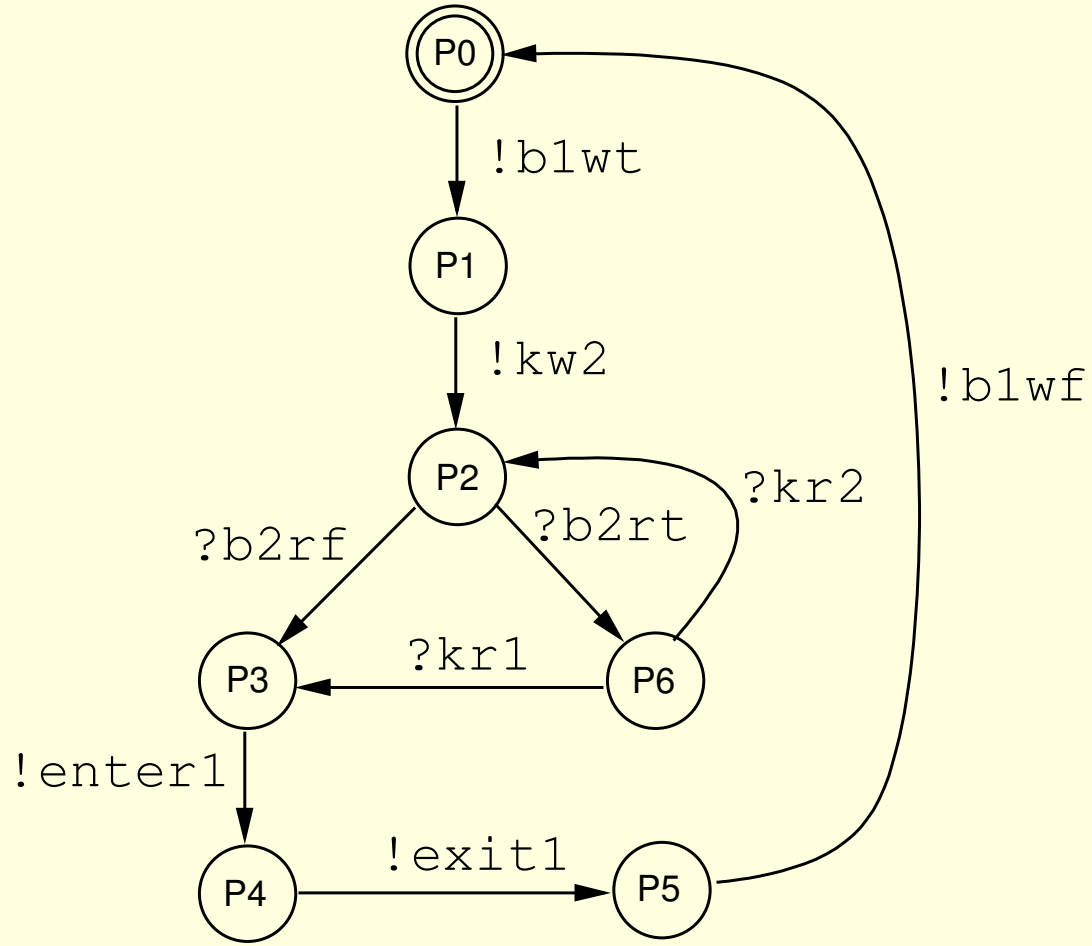


Figure 1: An LTS representing process P1 in Peterson's algorithm

6 Calculus of Communicating Systems

CCS is a process calculus. Due to the nice laws valid for its operators it is also classified as process algebra.

Each CCS expression defines a process (also called an agent).

CCS operators supported by EST are:

- Prefix (.),
- Summation ($+$),
- Composition ($|$),
- Restriction ($\backslash$), and
- Relabelling ($[\ ]$).

7 Operator Prefix

Operator Prefix is defined with the following rule:

$$\textbf{Prefix}\ \frac{}{a.\mathcal{M} \xrightarrow{a} \mathcal{M}}$$

Process $a.\mathcal{M}$ can execute action a and afterwards behave as process $\mathcal{M}$.

This means that the initial state of the LTS representing process $a.\mathcal{M}$ should have an a-transition to the initial state of the LTS representing process $\mathcal{M}$.

8 Operator Summation

Operator Summation is defined with the following two rules:

$$\textbf{Sum1}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1'}{(\mathcal{M}_1 + \mathcal{M}_2) \xrightarrow{a} \mathcal{M}_1'}$$

$$\textbf{Sum2}\ \frac{\mathcal{M}_2 \xrightarrow{a} \mathcal{M}_2'}{(\mathcal{M}_1 + \mathcal{M}_2) \xrightarrow{a} \mathcal{M}_2'}$$

If any process participating in the summation can execute action a, then the sum can also execute action a.

This means that the initial state of the LTS representing process $\mathcal{M}_1 + \mathcal{M}_2$ should have exactly those transitions which are present in the initial states of the LTSs representing processes $\mathcal{M}_1$ and $\mathcal{M}_2$.

9 Operator Composition

Operator Composition is defined as follows:

$$\textbf{Com1}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1'}{(\mathcal{M}_1|\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1'|\mathcal{M}_2)}$$

$$\textbf{Com2}\ \frac{\mathcal{M}_2 \xrightarrow{a} \mathcal{M}_2'}{(\mathcal{M}_1|\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1|\mathcal{M}_2')}$$

$$\textbf{Com3}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1' \quad \mathcal{M}_2 \xrightarrow{\bar{a}} \mathcal{M}_2'}{(\mathcal{M}_1|\mathcal{M}_2) \xrightarrow{\tau} (\mathcal{M}_1'|\mathcal{M}_2')}\ (a \neq \tau)$$

Composition is used to model synchronous communication between two processes.

10 Operator Restriction

In EST, operator Restriction is defined as follows:

$$\textbf{Res}\ \frac{\mathcal{M} \xrightarrow{a} \mathcal{M}'}{(\mathcal{M}\backslash\hat{b}) \xrightarrow{a} (\mathcal{M}'\backslash\hat{b})}\ (b \neq \tau, a = \tau \vee \hat{a} \neq \hat{b})$$

Process $\mathcal{M}\backslash\hat{b}$ behaves like process $\mathcal{M}$ but it cannot execute action with name $\hat{b}$.

This means that states in the LTS representing process $\mathcal{M}\backslash\hat{b}$ have exactly those transitions which are present in the LTS reperesenting process $\mathcal{M}$ and are labelled with an action whose action name is not equal to $\hat{b}$.

11 Operator Relabelling

Operator Relabelling is defined as follows:

$$\textbf{Rel}\ \frac{\mathcal{M} \xrightarrow{a} \mathcal{M}'}{\mathcal{M}[f] \xrightarrow{f(a)} \mathcal{M}'[f]}$$

Here, f is a relabelling function such that $f(\bar{a}) = \overline{f(a)}$ and $f(\tau) = \tau$. In EST, a relabelling function is given as a pair of action names.

LTS representing process $\mathcal{M}[f]$ is obtained from LTS representing process $\mathcal{M}$ by changing all transition labels according to the relabelling function f.

12 Algebraic laws for CCS operators

Two specifications are equivalent only if the processes they define are strongly equivalent [Mil89].

The following basic laws for defined operators are consistent with this presumption:

- $\mathcal{M}_1 + \mathcal{M}_2 = \mathcal{M}_2 + \mathcal{M}_1$
- $(\mathcal{M}_1 + \mathcal{M}_2) + \mathcal{M}_3 = \mathcal{M}_1 + (\mathcal{M}_2 + \mathcal{M}_3)$
- $\mathcal{M}_1 | \mathcal{M}_2 = \mathcal{M}_2 | \mathcal{M}_1$
- $(\mathcal{M}_1 | \mathcal{M}_2) | \mathcal{M}_3 = \mathcal{M}_1 | (\mathcal{M}_2 | \mathcal{M}_3)$
- $\mathcal{M} \backslash \hat{a} \backslash \hat{b} = \mathcal{M} \backslash \hat{b} \backslash \hat{a}$

13 More on algebraic laws

Composition is distributive over Summation:

$$\mathcal{M}_1 \mid (\mathcal{M}_2 + \mathcal{M}_3) = (\mathcal{M}_1|\mathcal{M}_2) + (\mathcal{M}_1|\mathcal{M}_3)$$

The opposite is not true. Also, Prefix is distributive neither over Summation nor over Composition:

- $\mathcal{M}_1 + (\mathcal{M}_2|\mathcal{M}_3) \neq (\mathcal{M}_1 + \mathcal{M}_2) \mid (\mathcal{M}_1 + \mathcal{M}_3)$
- $a.(\mathcal{M}_1 + \mathcal{M}_2) \neq a.\mathcal{M}_1 + a.\mathcal{M}_2$
- $a.(\mathcal{M}_1|\mathcal{M}_2) \neq a.\mathcal{M}_1 \mid a.\mathcal{M}_2$

14 Example 2: Dining philosophers

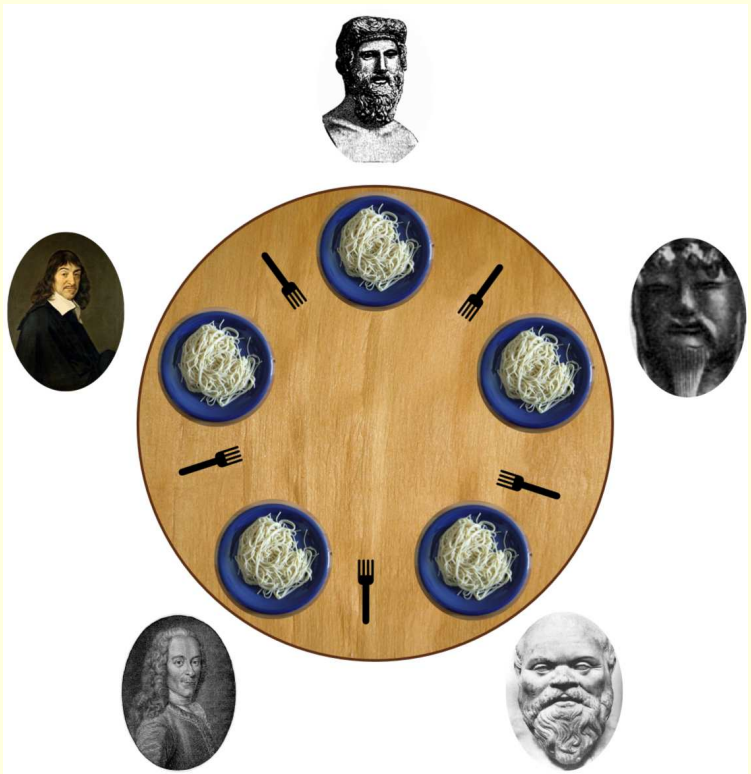
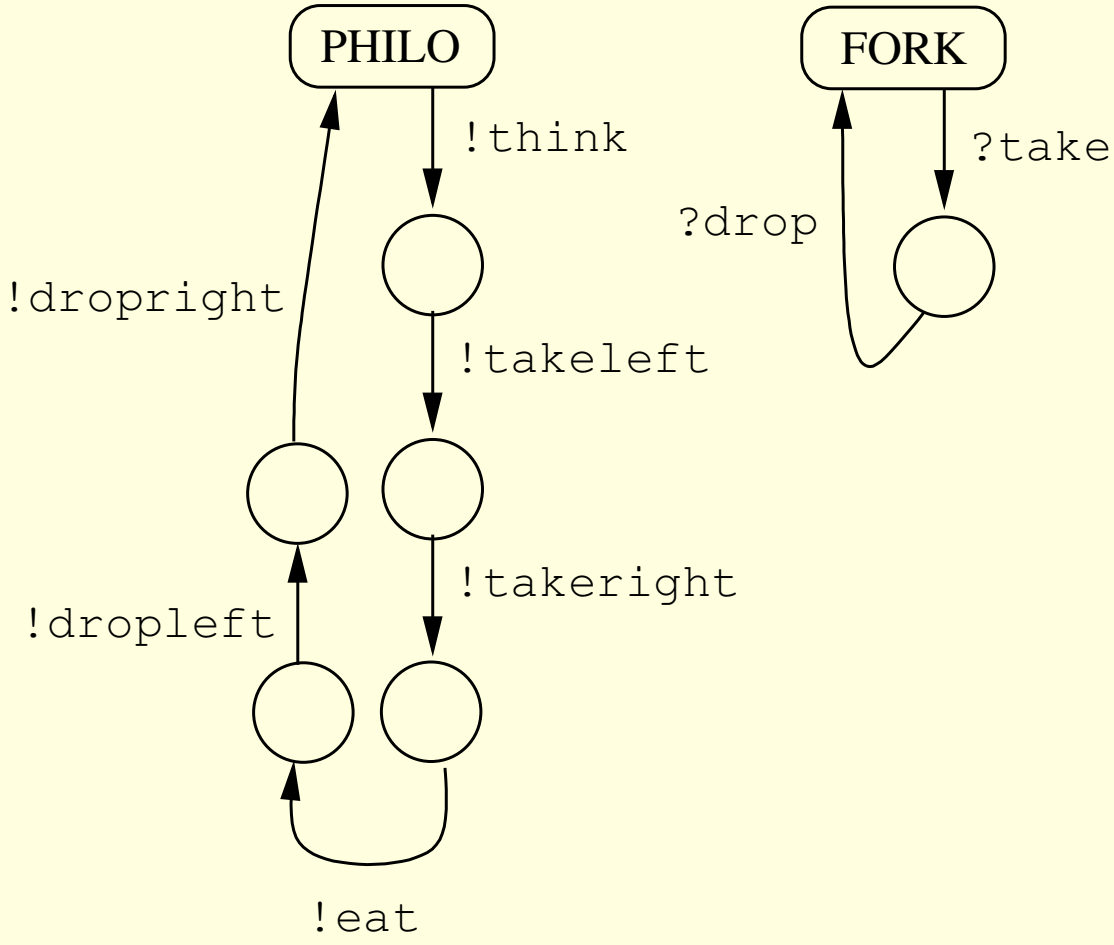


Figure 2: Dining philosophers (from Wikipedia)

15 Example 2: EST specification (1)

```
PHILO =
  !think.
  !takeleft.
  !takeright.
  !eat.
  !dropleft.
  !dropright.
  PHILO

FORK =
  ?take.
  ?drop.
  FORK
```



16 Example 2: EST specification (2)

```
DINNER =(
  PHILO
  [think1/think]
  [take2/takeleft] [take1/takeright]
  [eat1/eat]
  [drop2/dropleft] [drop1/dropright]
| FORK
  [take1/take] [drop1/drop]
| PHILO
  [think2/think]
  [take1/takeleft] [take2/takeright]
  [eat2/eat]
  [drop1/dropleft] [drop2/dropright]
| FORK
  [take2/take] [drop2/drop]
)\take1\drop1\take2\drop2
```

17 Example 2: The composition

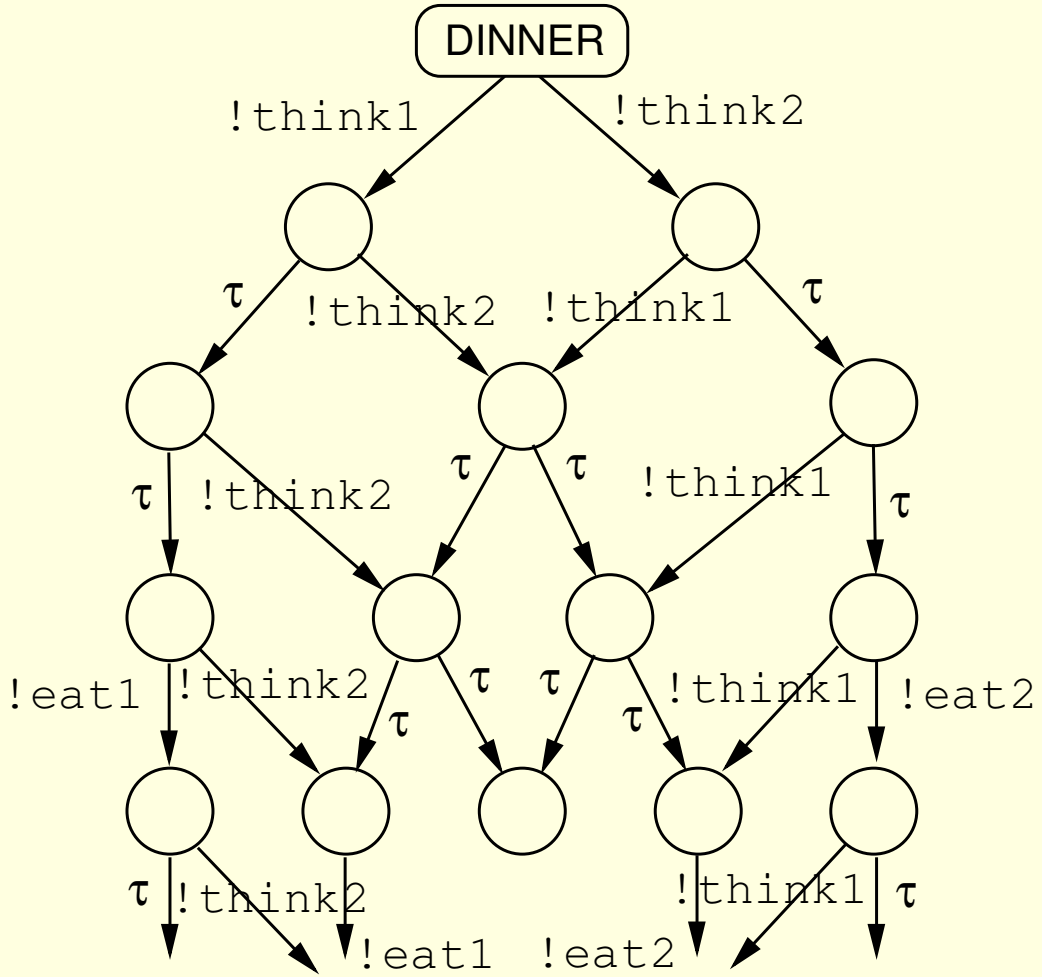


Figure 3: A part of the obtained LTS for two philosophers

18 Additional operators

CCS is quite a simple formalism and it is not very practical for specification of real systems.

To enhance it a little, the parser in EST is extended with some operators from other formalisms (e.g. CSP and LOTOS).

- Synchronisation ($||$),
- Partial synchronisation ($|[\ ]|$),
- Interleaving ($|||$), and
- PrefixN ($(+).$).

19 Operator Synchronisation

Operator Synchronisation is defined as follows:

$$\textbf{Syn1}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1'}{(\mathcal{M}_1||\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1'||\mathcal{M}_2)}\ (a=\tau \vee \bar{a} \notin \mathcal{A}_\tau')$$

$$\textbf{Syn2}\ \frac{\mathcal{M}_2 \xrightarrow{a} \mathcal{M}_2'}{(\mathcal{M}_1||\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1||\mathcal{M}_2')}\ (a=\tau \vee \bar{a} \notin \mathcal{A}_\tau)$$

$$\textbf{Syn3}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1' \quad \mathcal{M}_2 \xrightarrow{\bar{a}} \mathcal{M}_2'}{(\mathcal{M}_1||\mathcal{M}_2) \xrightarrow{\tau} (\mathcal{M}_1'||\mathcal{M}_2')}\ (a \neq \tau)$$

Opposite to operator Composition, operator Synchronisation **requires** strictly synchronous execution.

20 Partial synchronisation

Operator Partial synchronisation is defined as follows:

$$\textbf{Part1}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1'}{(\mathcal{M}_1 |[A]| \mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1' |[A]| \mathcal{M}_2)}\ (a \notin A)$$

$$\textbf{Part2}\ \frac{\mathcal{M}_2 \xrightarrow{a} \mathcal{M}_2'}{(\mathcal{M}_1 |[A]| \mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1 |[A]| \mathcal{M}_2')}\ (a \notin A)$$

$$\textbf{Part3}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1' \quad \mathcal{M}_2 \xrightarrow{\bar{a}} \mathcal{M}_2'}{(\mathcal{M}_1 |[A]| \mathcal{M}_2) \xrightarrow{\tau} (\mathcal{M}_1' |[A]| \mathcal{M}_2')}\ (a \in A)$$

Partial synchronisation is a generalisation of Synchronisation such that it requires synchronous execution of some actions, only.

21 Operator Interleaving

Operator Interleaving is defined as follows:

$$\textbf{Int1}\ \frac{\mathcal{M}_1 \xrightarrow{a} \mathcal{M}_1'}{(\mathcal{M}_1|||\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1'|||\mathcal{M}_2)}$$

$$\textbf{Int2}\ \frac{\mathcal{M}_2 \xrightarrow{a} \mathcal{M}_2'}{(\mathcal{M}_1|||\mathcal{M}_2) \xrightarrow{a} (\mathcal{M}_1|||\mathcal{M}_2')}$$

Operator Interleaving is used to model asynchronous parallel execution of processes. It can be easily derived from operator Partial synchronisation:

$$\mathcal{M}_1|||\mathcal{M}_2 = \mathcal{M}_1|[\emptyset]|\mathcal{M}_2$$

22 Operator PrefixN

Operator PrefixN is a useful abbreviation supported by the EST parser. Let $\underline{A_i}$ stand for $a_{i,1}.a_{i,2}.\ldots.a_{i,n_i}$. Then:

$$(\underline{A_1}+\underline{A_2}+\ldots+\underline{A_N}).\mathcal{M} \triangleq \underline{A_1}.\mathcal{M}+\underline{A_2}.\mathcal{M}+\ldots+\underline{A_N}.\mathcal{M}$$

Let $\underline{A'_i}$ stand for $a_{i,2}.\ldots.a_{i,n_i}$. Then operator PrefixN can be also given as follows:

PrefixN
$$\frac{}{(\underline{A_1}+\ldots+\underline{A_i}+\ldots+\underline{A_N}).\mathcal{M} \stackrel{a_{i,1}}{\rightarrow} \underline{A'_i}.\mathcal{M}}$$

23 Example 3: Gas Station problem

The system consists of an operator, a pump, and customers. The operator initially accepts money prepaid by customers and then activates the pump. On receiving the charge information from the pump, the operator gives the change to the customer.

Figure 4: Gas station in Hiroshima (from Wikipedia)

24 Example 3: CCS specification

```
OPERATOR =
  (?prepay1+?prepay2).OP_PREPAID +
  (?charge1+?charge2).OP_CHARGED
OP_PREPAID =
  (?avlbl.!act+!occupied).OPERATOR
OP_CHARGED =
  (!change1+!change2).
  (?wait.!act+!none).OPERATOR
QUEUE = !avlbl.QUEUE_ACTIVE
QUEUE_ACTIVE =
  ?none.QUEUE +
  ?occupied.!wait.QUEUE_ACTIVE
PUMP =
  ?act.
  (?start1+?start2).
  (?finish1+?finish2).
  (!charge1+!charge2).PUMP
```

```
CUST =
  !prepay.!start.!stop.?change.CUST
STATION =
  OPERATOR
  ||[avlbl,occupied,none,wait]||
  QUEUE
  ||[activate,charge1,charge2]||
  PUMP
CUSTOMERS =
  CUST [prepay1/prepay][start1/start]
       [stop1/stop][change1/change]
  |||
  CUST [prepay2/prepay][start2/start]
       [stop2/stop][change2/change]
SYSTEM =
  STATION
  ||[start1,start2,finish1,finish2]||
  CUSTOMERS
```

25 Example 3: Obtained LTS

```
OPERATOR = (?prepay1+?prepay2).OP_PREPAID + (?charge1+?charge2).OP_CHARGED
OP_PREPAID = (?avlbl.!act+!occupied).OPERATOR
OP_CHARGED = (!change1+!change2).(?wait.!act+!none).OPERATOR
```

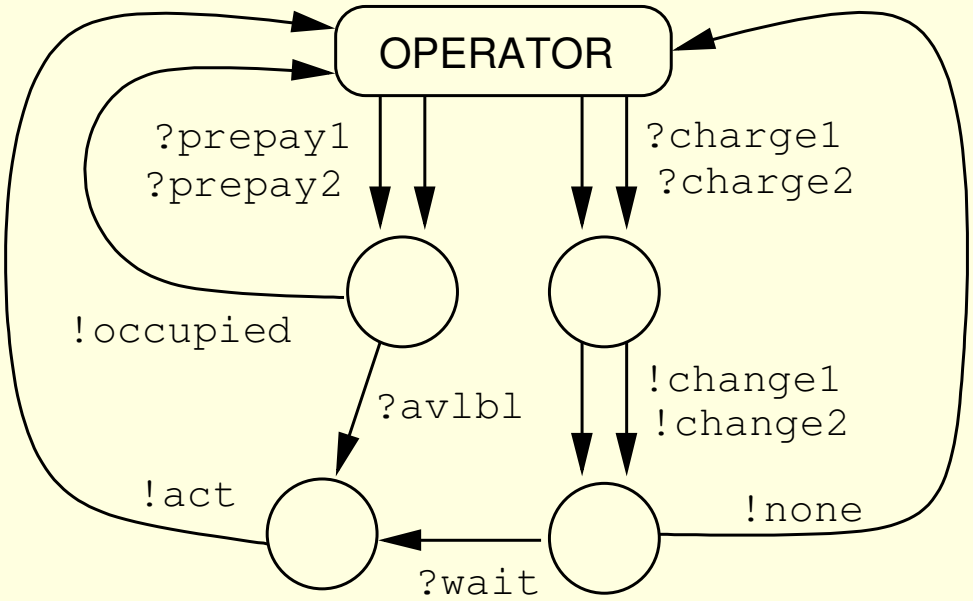


Figure 5: A detail from the Gas Station problem

26 Conclusion

- **EST** is a tool for formal verification of systems.
- A system to be verified should be specified in a CCS-like syntax.
- In the paper we make an overview of operators supported by EST.
- Besides standard CCS operators, the parser includes additional operators similar to those from CSP and LOTOS.
- EST is free software. Homepage:
  `http://lms.uni-mb.si/EST/`