## Translating FSP into LOTOS and Networks of Automata







Jeff Kramer

Imperial College London



Jeff Magee



Frédéric Lang



\* New affiliation: Universidad de Málaga

## Motivations

- Process algebras are abstract description languages to specify concurrent systems:
  - expressive and textual notations
  - compositional specifications
  - formal verification tools



• Fragmentation of the process algebra community

 $\Rightarrow$  languages incompatible in practice

- Our goal:
  - filling the gap between process algebras
  - making the joint use of existing tool-boxes possible

## Motivations

- FSP is a popular process algebra
  - + concise, expressive, and easy-to-use notation
  - basic verification means (LTSA)
    - $\Rightarrow$  animation and LTL property checking
- LOTOS is an ISO standard
  - + rich verification toolbox CADP
  - expressive notation, needs expertise
- Translating FSP into LOTOS:
  - FSP is a simple yet expressive notation
  - CADP is a rich toolbox to be used jointly with LTSA to analyse FSP specifications



a state of the state of	- Setter and a Carlos		8	
File Vent	Relation al blad_storeg larg	05-g "Edinburgh" (RC1)	100 500	
	Production unpation (17)     Production (17)     Production (17)     Production (17)     Production unique     Communique	a con		
ating in , key	Other splane     Other splane     Other splane     Process for motivative and published LSAs [55:4]     For the splane to an or a vision of the splane to an or a vision     Process for a published LSAs [55:4]     Process for a published lSAs	Cro-m-	<b>\$</b> 0	
	C Reduce on the low service Reductor	1		
bitalt.ps	Telector intern # Storg nanotenis C. Taccothenis C. Tacco equivalence C. Tacothegence C. Tal/caspulation C. Visal hair equivalence C. Taccorgences C. Salety equivalence	ef Consistence of Servers' 'Theory's regulation of Servers' 'Theory's research (2003) Reservables balants of the Reservables (2003) Reservables (1) Performance Reservables (1) Performan		
	Divergitors C Patid-induction # Total-induction C Real-line monitoring damp (ECS graph construction	analis malysis of "silait" - processes bising	C (E)	
test!.beg	Dies giben C Patrierekolon III Tellindeckon C Residen motioning derig ECS gesch contraction	spectrum analysis of "bitain" protectors biology 	C.C.	
test) ber	C Parkel velocition of Tate indexion C Parkel velocition of Tate indexion C Parkel velocition data (TES) grads periodiculture Parkel velocitication Reduction onlytics Reduction only	marticle analysis of "bitch" processes biology (ISCC provides) Gard Committee Training Committee Committee Training to Committee Planning Tales processes Tools and Planning Tales processes Tools and Planning Tales processes Tools and Planning Tales processes	des bey Story (Merc) 201 (Merc) Mercy Mercy 20000	
toott birg	C Padetactan # Tartactan C Padetactan # Tartactan C Padeta warbang targ US gada janaksia C Roberts ang Mattaia # Channels ang Mattaia # Channels and Mattaia # Salaman color of USA # Channels and USA	Andres malipies of Bildst. Gent Connector Court Andre If Space Training to Court Andre If Space Training to Court Andre If Space Training to Court Andre If Space Training to Part of Court Andre If Space Training Training Court Training Training Training Court Training	an bey http://dei 20 (dei 20 (	
	Predication # Tartadow     Preference     Pref	Construction     C	de bey BED / Fel SED / Fel SED / Fel SED SED SED SED SED SED SED SED SED SED	

## Comparison

Criteria	FSP	LOTOS
Binary communication	Yes	Yes
N-ary communication	Yes	Yes
M among N comm.	No	Yes (E-LOTOS)
Name matching	Yes	Yes
Tools	Yes (-)	Yes (++)
Graphical notations	Yes (++)	Yes (-)
Data	Simple	Complex
Expressiveness	Yes (+)	Yes (+)
Compositionality	No	Yes
User-friendliness	Yes (+)	Yes (-)
Conciseness/readability	Yes (+)	Yes (-)

### LOTOS + EXP.OPEN

- High-level translation between process calculi are preferred as often as possible:
  - Translation of behavioural operators easier
  - Mandatory to use some verification tools of CADP
  - Benefit from the Caesar.adt and Caesar compilers
- However, FSP composite processes are difficult to encode into LOTOS:
  - Synchronisations between complex labels
  - Priorities

 $\Rightarrow$  encoding into EXP.OPEN (EXP for short) which allows the description of networks of automata

- FSP, LOTOS, and EXP
- Translating FSP basic processes into LOTOS
- Translating FSP composite processes into EXP
- Prototype and validation
- Conclusion and future work

### Finite State Processes (FSP)

• Constants, ranges, sets

const C=3 range R=1..C set S={ash,eat}

• Expressive notation to specify labels

comm[k:R]	$\Rightarrow$	comm.1, comm.2, comm.3
order.m[S]	$\Rightarrow$	order.mash, order.meat

• Prefix, choice, if, sequence, hiding, renaming

SERVER = ( request[id:0..1] -> LOC[id] ),

LOC[id:0..1] = ( when id==0 over -> END |

when id!=0 comm.send[id] -> END ).

TWO = SERVER; SERVER; END /{comm/request} \{over}.

### Finite State Processes (FSP)

- Parallel composition C<sub>1</sub> | |C<sub>2</sub> of processes
- Label priority: >> { $l_1$ , ...,  $l_n$ }, << { $l_1$ , ...,  $l_n$ }
- Renaming  $\{l_1/l_1', ..., l_n/l_n'\}$ , hiding  $\{l_1, ..., l_n\}$
- Process labelling  $\{l_1, ..., l_n\}$ : C and sharing  $\{l_1, ..., l_n\}$ : C



#### Language of Temporal Ordering Specification (LOTOS)

• Abstract datatypes:

 $\Rightarrow$  sorts, operations, generators, axioms

• Basic LOTOS (only behaviours)

aa; exit [] ( bb; comm; exit |[comm]| cc; comm; exit )

• Full LOTOS (behaviours + data terms)

### Networks of Automata (EXP.OPEN)

- Parallel composition of automata (bcg format):
  - CCS, CSP, (E)LOTOS, MuCRL compositions, for instance

label par  $l_1$ , ...,  $l_m$  in  $B_1 \mid \mid ... \mid \mid B_n$  end par

 $B_1 \parallel \parallel ... \parallel \parallel B_n$  (interleaving)

- Synchronisation vectors

label par  $v_1$ , ...,  $v_m$  in  $B_1 \mid \mid ... \mid \mid B_n$  end par

• Renaming, hiding, cutting, priority operators

total rename  $l_1 \rightarrow l_1$ , ...,  $l_n \rightarrow l_n$  in B end rename total hide/cut  $l_1$ , ...,  $l_n$  in B end hide/cut total prio  $l_1$ , ...,  $l_n >$  all but  $l_1$ , ...,  $l_n$  in B end prio total prio all but  $l_1$ , ...,  $l_n > l_1$ , ...,  $l_n$  in B end prio

- FSP, LOTOS, and EXP
- Translating FSP basic processes into LOTOS
- Translating FSP composite processes into EXP
- Prototype and validation
- Conclusion and future work

## **Action Labels**

- One FSP label may describe several LOTOS ones
  - $\Rightarrow$  expansion of labels to make renaming and hiding possible
- Full expansion when renaming/hiding needed

lab[x:1..2] $\Rightarrow$ EVENT!CONS(LAB,CONS(1,NIL)) $\Rightarrow$ EVENT!CONS(LAB,CONS(2,NIL))

• Compact notation keeping variable otherwise

```
lab[x:1..2] ⇒
choice X:Int[] ... EVENT!CONS(LAB,CONS(X,NIL)) [X≥1 and X≤2]
```

### Sequential Processes

- Terminations:
  - END  $\Rightarrow$  exit
  - **STOP**  $\Rightarrow$  stop



- ERROR  $\Rightarrow$  P\_ERROR [EVENT\_ERROR]
- Action prefix  $l \rightarrow B \Rightarrow (l_1; exit [] ... [] l_n; exit) >> B$  $\rightarrow I_i$  obtained by expansion, renaming, hiding
- Choice: when G<sub>1</sub> B<sub>1</sub> | when G<sub>2</sub> B<sub>2</sub>

 $\Rightarrow [G_1] \rightarrow B_1[] [G_2] \rightarrow B_2$ 

- Sequential composition:  $B_1; B_2 \Rightarrow B_1 >> B_2$
- if G then  $B_1$  else  $B_2 \Rightarrow [G] \rightarrow B_1 [] [\neg G] \rightarrow B_2$

# Example (1)

```
SERVER = ( request[id:0..1] -> LOC[id] ),
```

```
LOC[id:0..1] = ( when id==0 over -> END |
```

```
when id!=0 comm.send[id] -> END ).
```

TWO = SERVER; SERVER; END /{comm/request} \{over}.



( EVENT !CONS (COMM, CONS (POS(0), NIL)) ; LOC\_1 [EVENT] (0 of Int)
[] EVENT !CONS (COMM, CONS (POS(1), NIL)) ; LOC\_1 [EVENT] (1 of Int) )



endproc

# Example (2)

```
SERVER = ( request[id:0..1] -> LOC[id] ),
```

```
LOC[id:0..1] = ( when id==0 over -> END |
```

```
when id!=0 comm.send[id] -> END ).
```

TWO = SERVER; SERVER; END /{comm/request} \{over}.

```
process SERVER_1 [EVENT] : exit :=
  ( EVENT !CONS (COMM, CONS (POS(0), NIL)) ; LOC_1 [EVENT] (0 of Int)
  [] EVENT !CONS (COMM, CONS (POS(1), NIL)) ; LOC_1 [EVENT] (1 of Int) )
where
  process LOC_1 [EVENT] (ID : Int) : exit :=
  [ID==POS(0)] -> i; exit
  []
  [ID==POS(1)] -> EVENT !CONS (COMM, CONS (SEND, CONS (POS(1), NIL))); exit
  endproc
  endproc
```

- FSP, LOTOS, and EXP
- Translating FSP basic processes into LOTOS
- Translating FSP composite processes into EXP
- Prototype and validation
- Conclusion and future work

### **Composite Processes**

- Process P is translated as "P.bcg" if sequential
- Parallel composition  $C_1 | | C_2 \Rightarrow \text{label par } l_1, ..., l_m$  in  $C_1 | | C_2$  end par with  $l_i = \text{alph}(C_1) \cap \text{alph}(C_2)$
- Label priority, hiding: total prio, total hide
- Renaming using vectors (1-to-many renaming)  $(\{l_1/l_1', ..., l_n/l_n'\} \Rightarrow \text{label par } v_1, ..., v_m \text{ in ... end par } v_1$
- Process labelling and sharing:
  - $\{l_1,...,l_n\}$ : C  $\Rightarrow$  prefixing with vectors + interleaving
  - $\{l_1, ..., l_n\}:: C \Rightarrow$  prefixing with vectors
- if G then  $C_1$  else  $C_2 \Rightarrow [G] \rightarrow C_1 [] [\neg G] \rightarrow C_2$

## Example

```
CLIENT = ( [1] -> send[1] -> CLIENT ).
```

||SYS = (TWO || comm:CLIENT ).

```
label par "EVENT !CONS (COMM, CONS (POS(1), NIL))",
          "EVENT !CONS (COMM, CONS (SEND, CONS (POS(1), NIL)))" in
  total cut exit in "TWO.bcg" end cut
   label par
               "EVENT !CONS (POS(1), NIL)"
                      -> "EVENT !CONS (COMM, CONS (POS(1), NIL))", ... in
          total cut exit in "CLIENT.bcg" end cut
       end par
end par
```

- FSP, LOTOS, and EXP
- Translating FSP basic processes into LOTOS
- Translating FSP composite processes into EXP
- Prototype and validation
- Conclusion and future work

### Prototype

- A prototype translator fsp2lotos:
  - total of 25,500 lines of SYNTAX, LOTOS NT, and C
  - validated on 10,500 lines of FSP specifications

72,000 I. LOTOS, 8,000 I. EXP, 2,000 I. SVL

- Translation in two steps:
  - parsing and building an abstract syntax tree
  - translating the tree into semantically equivalent LOTOS code
- In the paper, application to a semaphore example for which CADP is used to analyse FSP specifications

### **Semantics Preservation**

• Essential to ensure that verification on the LOTOS specification is valid on the FSP one



- Conjecture: our translation preserves a branching equivalence relation
- Checked automatically on all the examples with Bisimulator (tool part of CADP)

- FSP, LOTOS, and EXP
- Translating FSP basic processes into LOTOS
- Translating FSP composite processes into EXP
- Prototype and validation
- Conclusion and future work

## Conclusion

• Translation from FSP to LOTOS and EXP

 $\Rightarrow$  makes the joint use of LTSA and CADP possible

### Future Work

- $LTS_{FSP} \cong LTS_{LOTOS}$ : equivalence to be proven
- Application to a complex system, *e.g.*, in web services, where CADP tools would be necessary
- Encoding FSP safety and progress properties into mu-calculus formulas, input format of Evaluator