The impact of EFSM Composition on Functional ATPG

Davide Bresolin, Giuseppe Di Guglielmo,
Franco Fummi, Graziano Pravadelli, Tiziano Villa

University of Verona
Department of Computer Science

Outline

Introduction

Laerte++: a functional ATPG

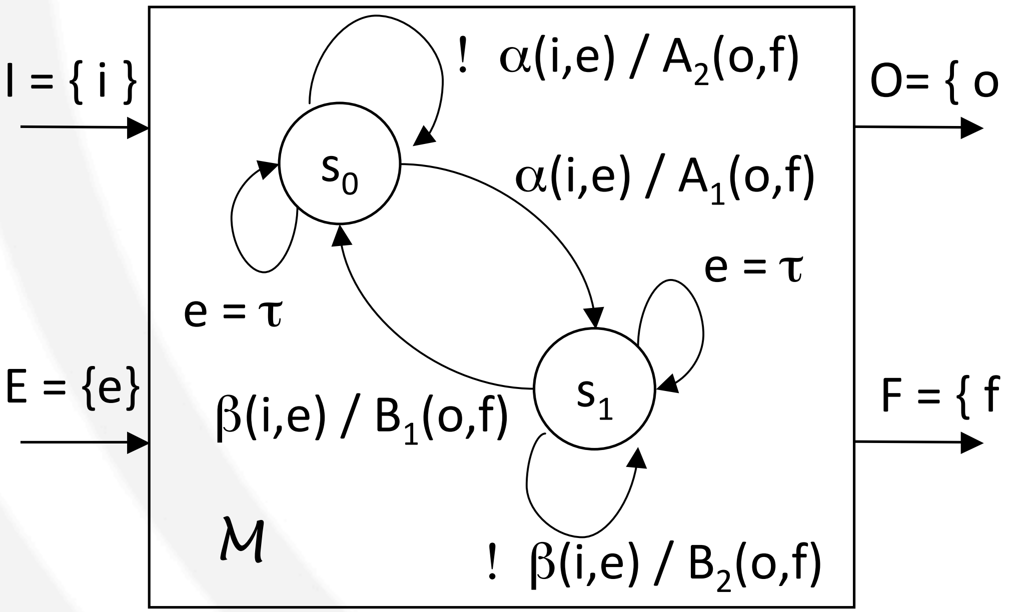
- Extended Event FSM (EEFSM) to model the DUT
- Constraint Logic Programming (CLP) engine to generate test vectors
- Two step algorithm: random walk + backjumping
- Transition and fault coverage to evaluate the quality of the test vectors

The EEFSM model

- EEFSM are I/O FSM augmented by a set of *registers* and of I/O *events*
- I/O events are used to model the clock and the *sensitivity list* construct of HDL
- Activation of a transition depends on the *state*, on *inputs*, on *events*, and on the value of the *registers.*

An EEFSM example

M : process(i)
begin
  case state is
  when s_0 =>
    if α (i) then
      A_1(o); state := s_1;
    else
      A_2(o); state := s_0;
    end if;
  when s_1 =>
    if β (i) then
      B_1(o); state := s_0;
    else
      B_2(o); state := s_1;
    end if;
  end case;
end process;



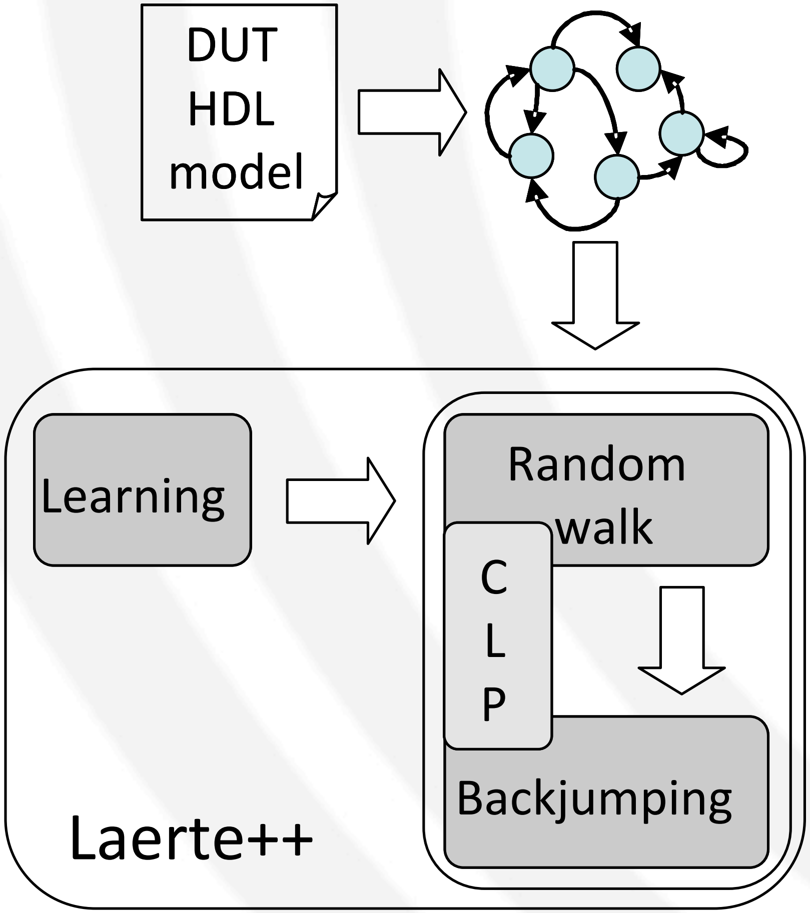
Advantages of EEFSM

- They allow for more compact representations (no space state explosion)
- The event-based semantics make the asynchronous composition cleaner
- Events are used to activate only the components that should trigger a transition

Hard and Easy transitions

- Transitions that depends only on *primary inputs* are *Easy-to-traverse (ETT)* transitions.
- Transition that depends on the *value of the registers* are *Hard-to-traverse (HTT)* and should be treated with special care.

Laerte++ engine



EFSM extraction

To fire easy-to-traverse (*ETT*) transitions

To fire hard-to-traverse (*HTT*) transitions

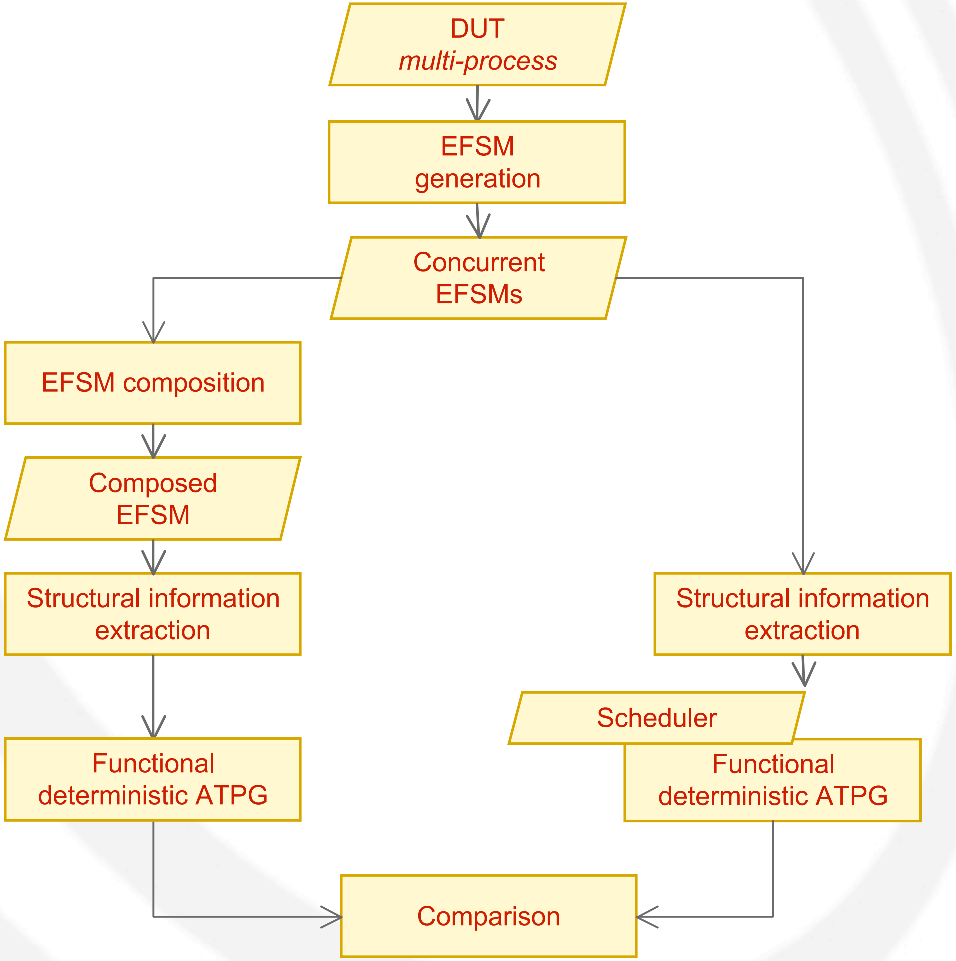
Bottlenecks

- With multi-process design, scheduling can be problematic
- DUTs with a large number of HTT have a low transition coverage
- Invocation of the CLP engine is time consuming

A possible solution

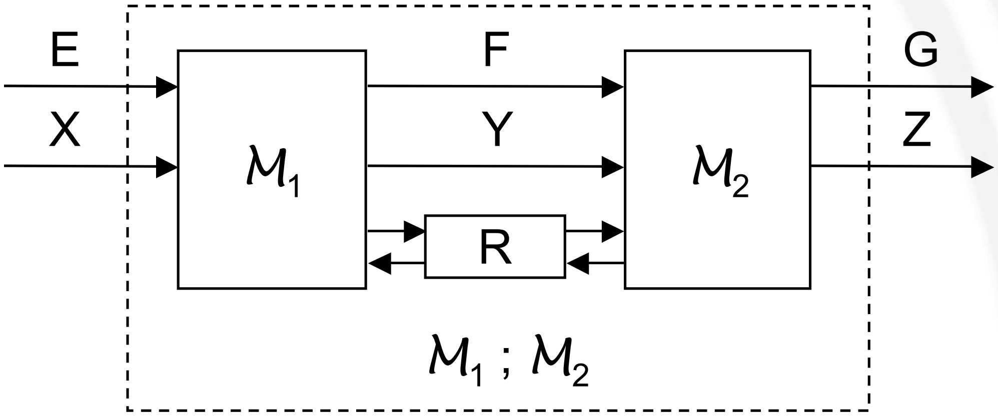
- **Compose the processes into a single EEFSM**

- Scheduling is simplified
- Some HTT become Easy
- Less CLP invocations
- Higher transition coverage



Serial composition

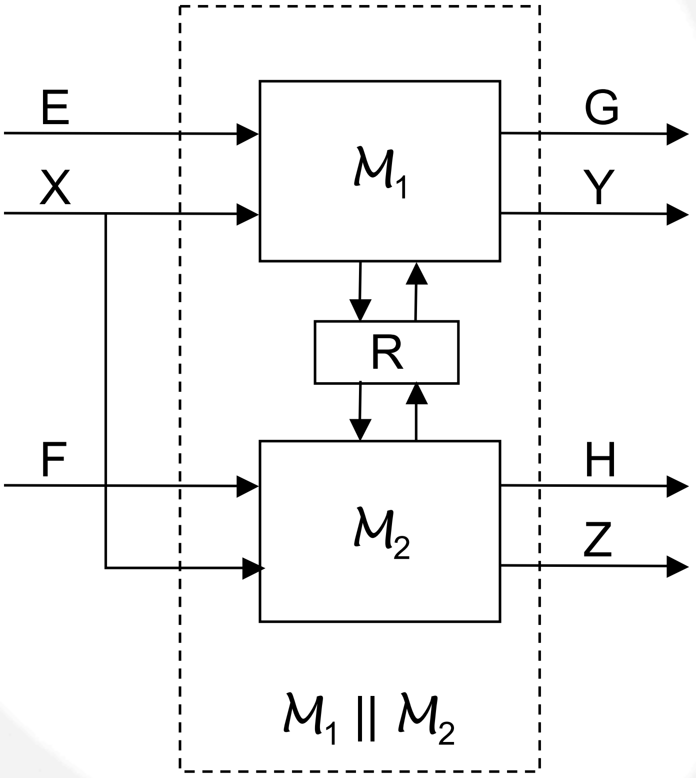
- Outputs of $\mathcal{M}_1$ are inputs of $\mathcal{M}_2$



- We do not allow the two EEFSM to update R simultaneously
- $\mathcal{M}_2$ fires a transition only if F is in its sensitivity list

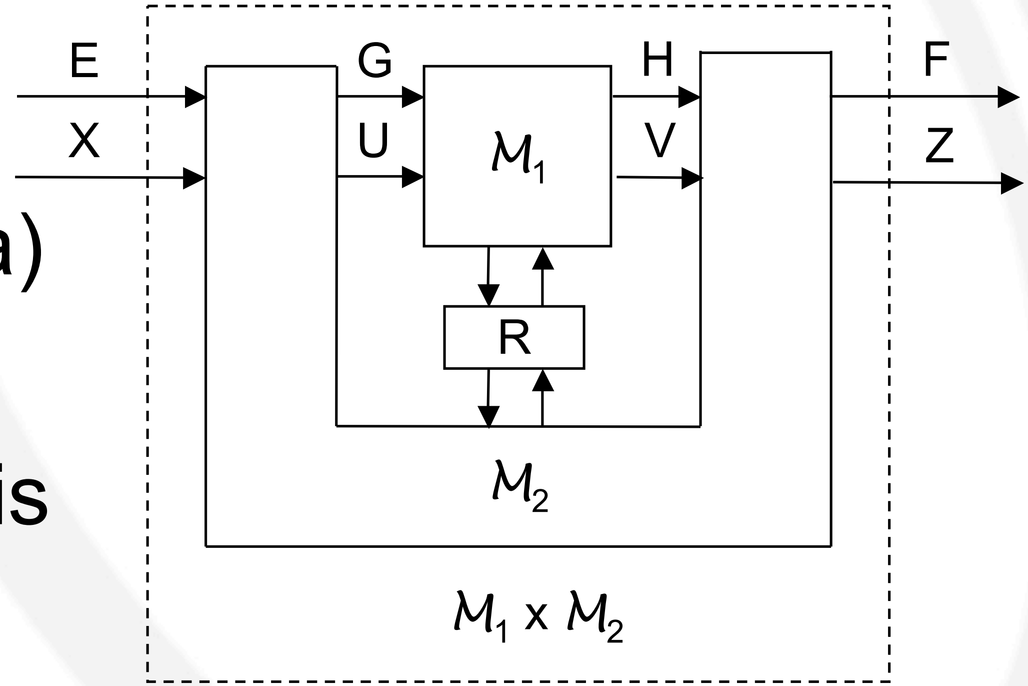
Parallel composition

- $\mathcal{M}_1$ and $\mathcal{M}_2$ share the same inputs
- R cannot be updated simultaneously
- Outputs are computed in parallel
- Transitions are not necessarily synchronized



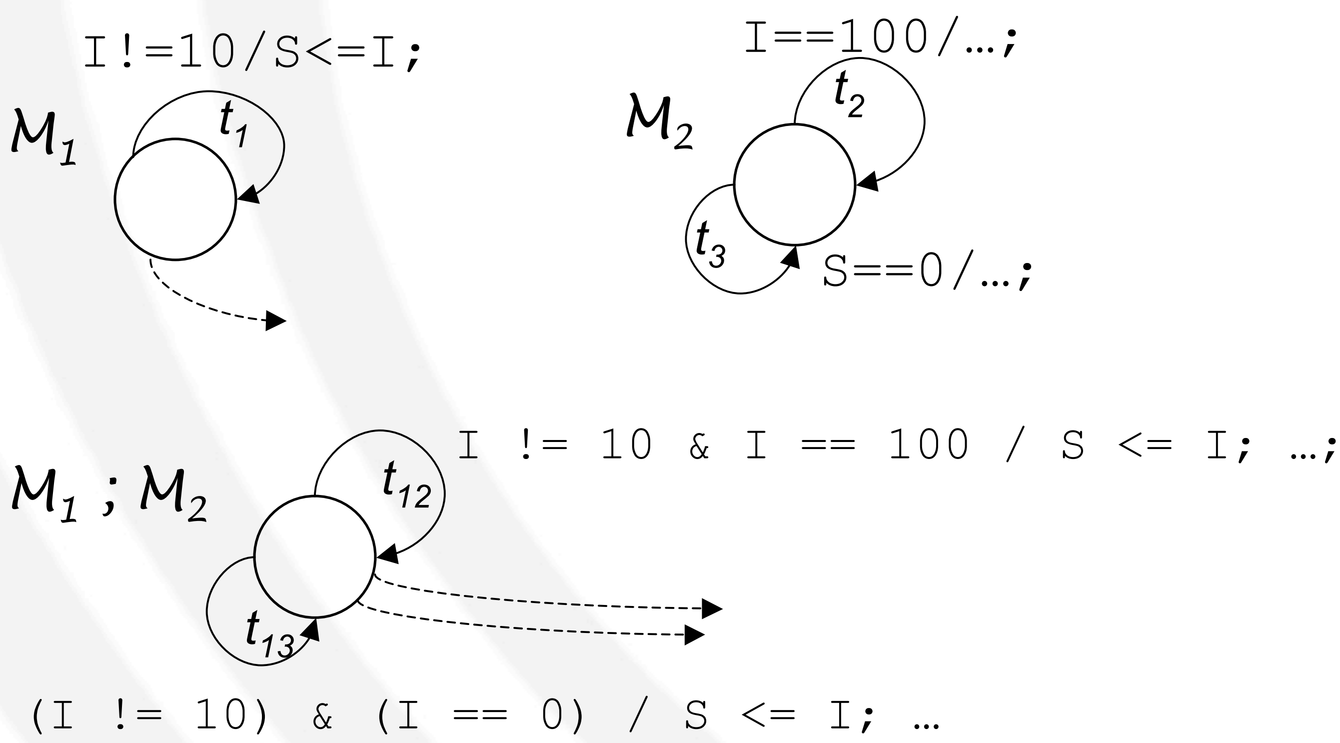
Feedback composition

- Some outputs of $\mathcal{M}_1$ are inputs of $\mathcal{M}_2$ (and viceversa)

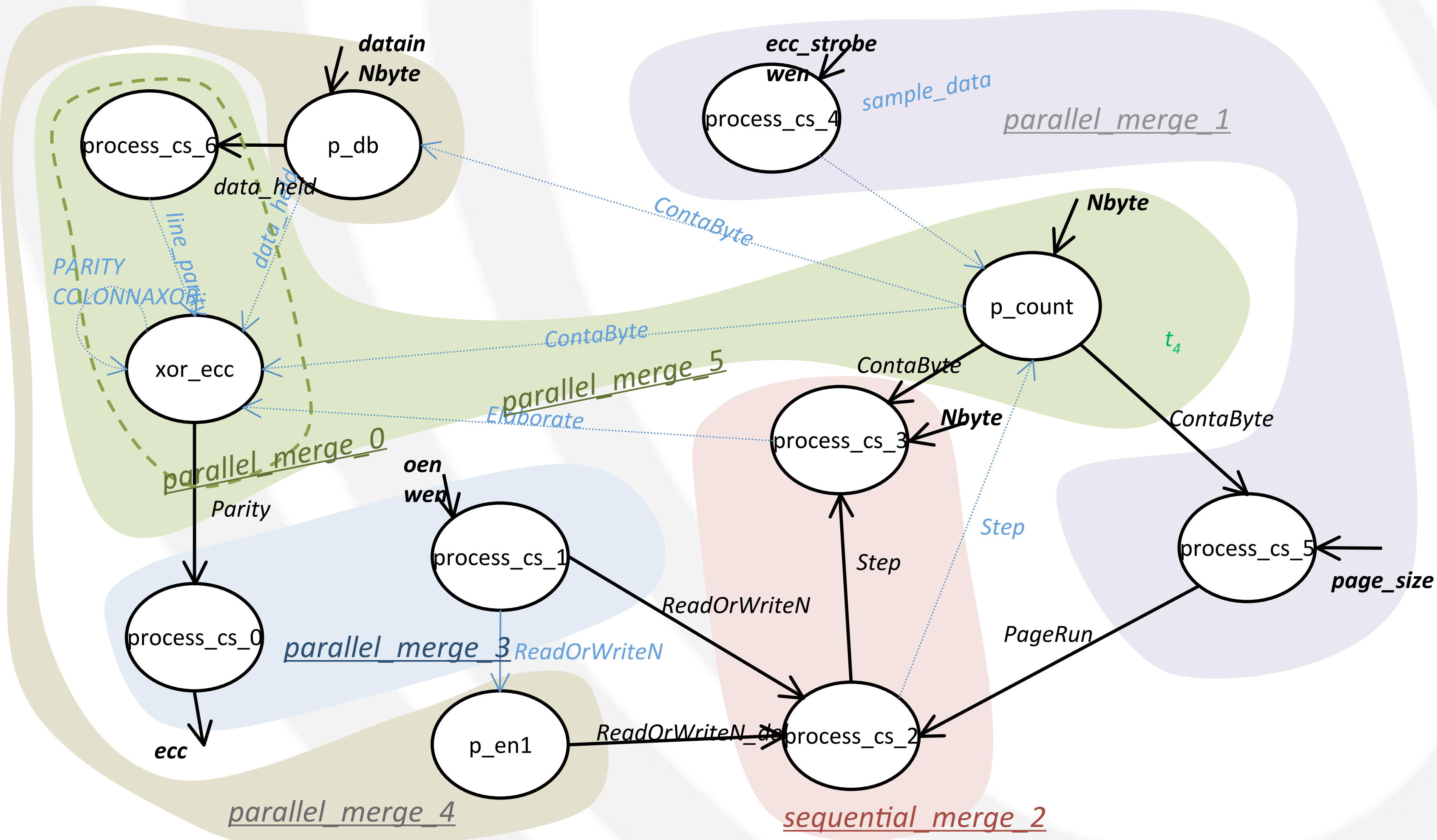


- This composition is well-defined only if there are no algebraic loops in the dependencies

Hard transitions became Easy!



Composing Processes



Experimental results

- Three industrial benchmarks:
  - *Vr01*: module of a face recognition system
  - *Ecc1, Ecc2*: ECC code of a 16bit page of data

DUT	PI	PO	P	FF	Gate	Trn	t(s)	BC
Vr01	65	16	8	73	615	69	7.1	2168
Ecc1	25	32	9	79	703	17	1.7	1022
Ecc2	55	32	7	88	832	24	2.4	1032

Experimental Results: *vr01*

D	T%	F%	t(s.)	TV	CSI	CST(s.)
0	80	54.10	228.38	15000	584935	182.7
1	80	54.10	201.35	15000	542523	161.1
2	80	54.10	187.34	15000	489031	159.3
3	93	71.80	134.92	15000	439413	103.4
4	93	71.80	132.33	15000	419321	100.0
5	93	71.80	103.43	15000	371391	73.1
6	100	98.90	13.93	256	16129	10.4
7	100	98.90	13.59	263	16129	10.1

Experimental Results: *ecc1, ecc2*

D	T%	F%	t(s.)	TV	CSI	CST(s.)
0	100	87.7	4.852	884	11680	3.785
8	100	87.7	3.612	973	5162	2.998

Ecc01

D	T%	F%	t(s.)	TV	CSI	CST(s.)
0	71	32.1	312.32	15000	612042	291.30
8	100	97.4	17.13	315	48984	16.98

Ecc02

Conclusions and Future Work

- EFSM composition has proved to be a valuable approach
- We are testing this approach on more complex case studies