Switch to full screen

Exploring Aspects in the Context of Reactive Systems

Exploring Aspects in the Context of Reactive Systems

An attempt at understanding AOP in the semantical framework we know best !

Karine Altisen, Florence Maraninchi, David Stauch

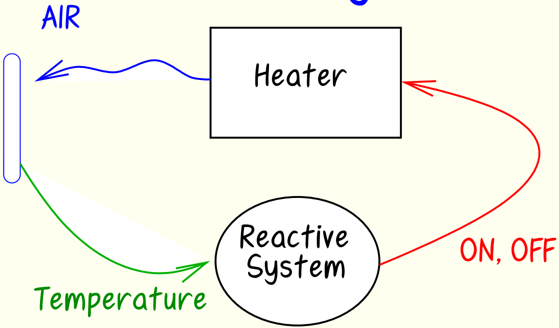
Verimag &

Institut National Polytechnique de Grenoble/Ensimag

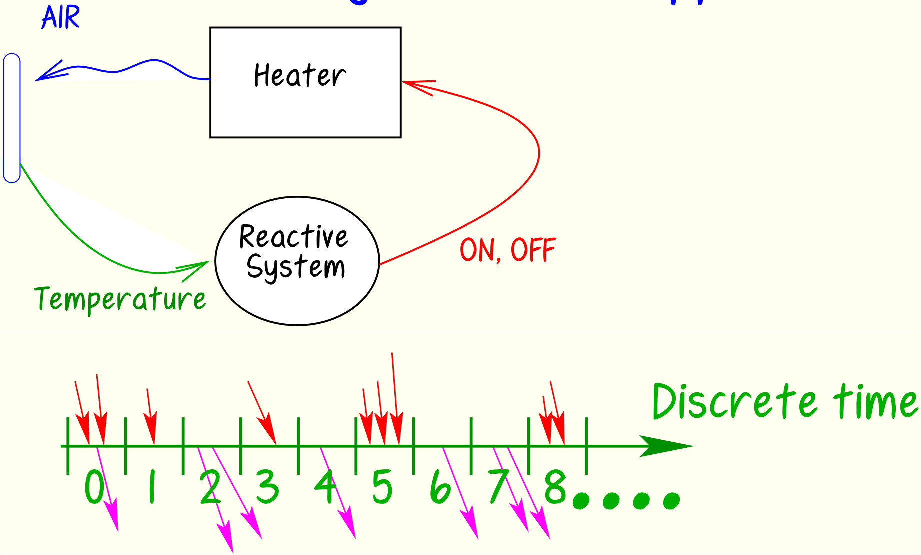
This paper...

- Reactive systems and the synchronous approach
  Programming with products of automata
- Candidate aspects in reactive programming
- A declarative (i.e., not constructive at all!) setting
- Candidate weaving mechanisms
- Conclusion

Reactive Systems and the Synchronous Approach



Reactive Systems and the Synchronous Approach



Reactive Systems and the Synchronous Approach

Languages :

- Lustre, Signal : dataflow
- Esterel : imperative with control structures
- Argos (inspired by Statecharts) : explicit automata

Reactive Systems and the Synchronous Approach

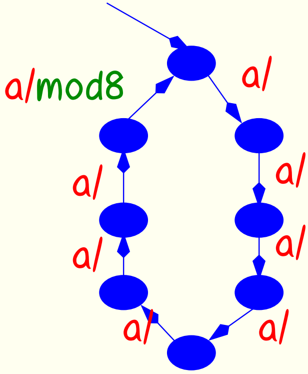
Languages :

- Lustre, Signal : dataflow
- Esterel : imperative with control structures
- Argos (inspired by Statecharts) : explicit automata

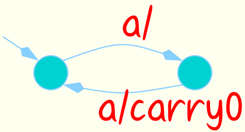
All these languages have a common semantical basis :

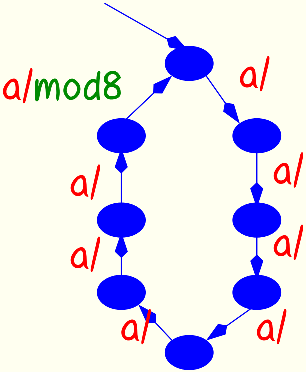
- deterministic and reactive Mealy machines +
- synchronous product +
- encapsulation

A modulo 8 counter - the program

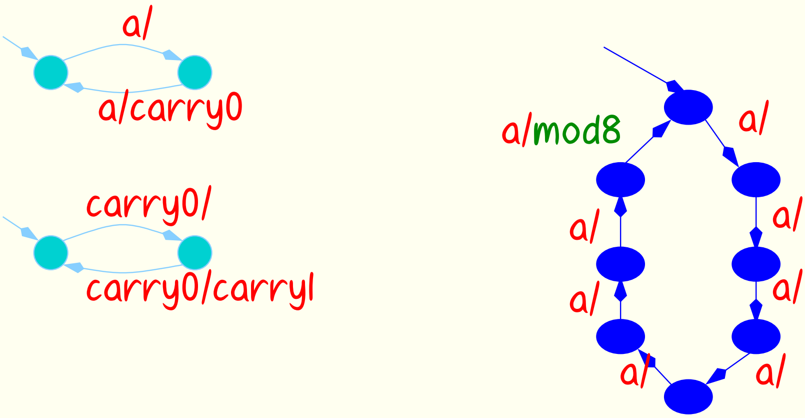


A modulo 8 counter – the program

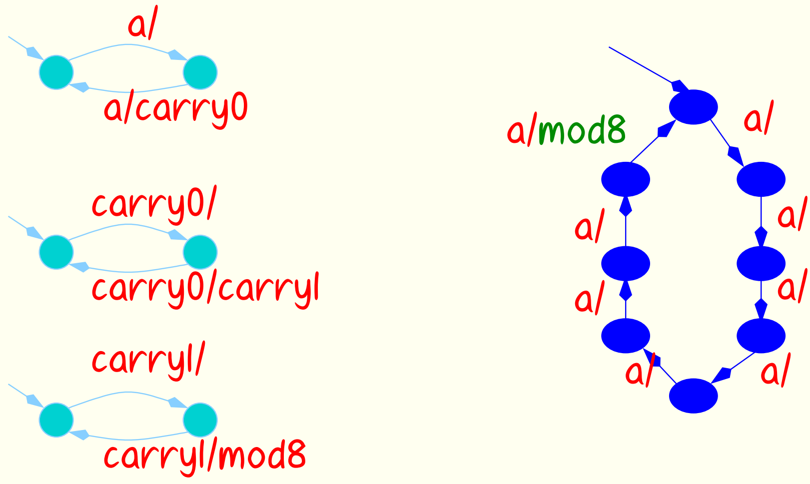




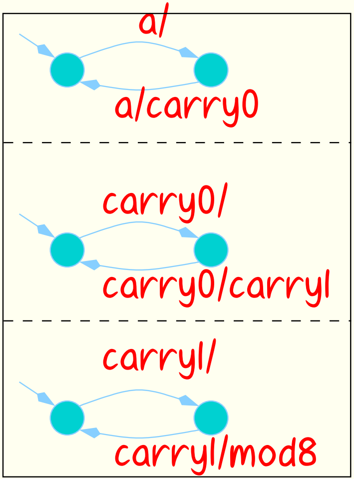
A modulo 8 counter – the program

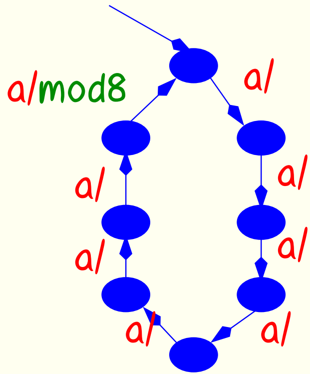


A modulo 8 counter – the program

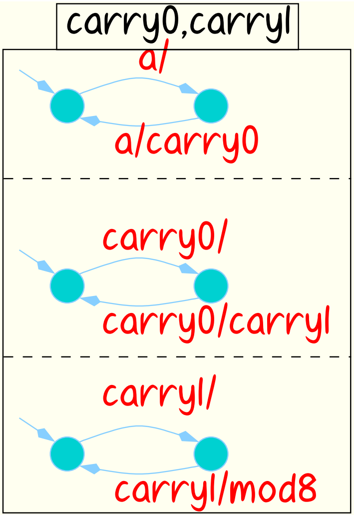


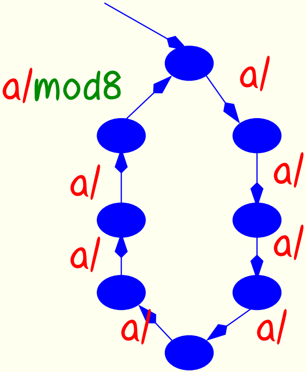
A modulo 8 counter – the program



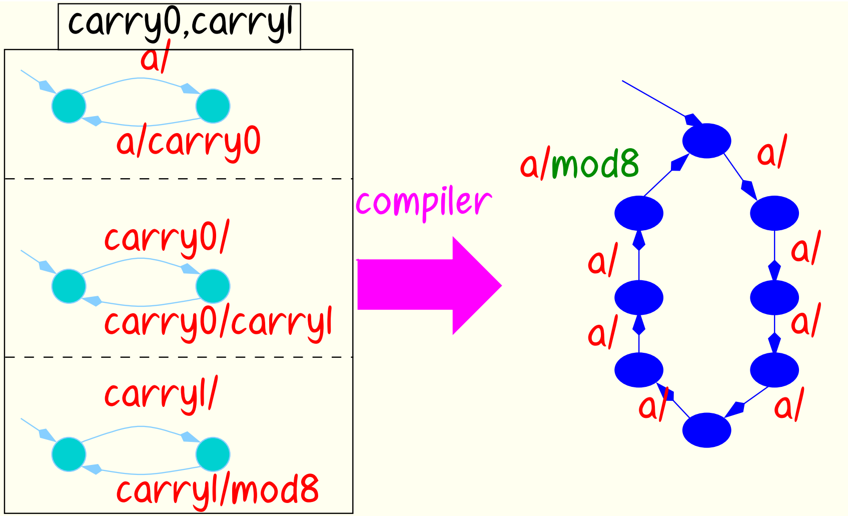


A modulo 8 counter – the program

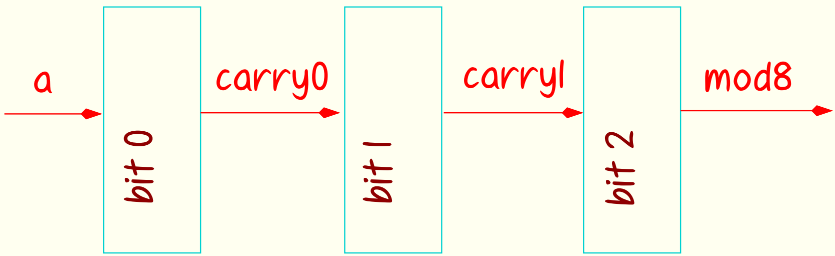




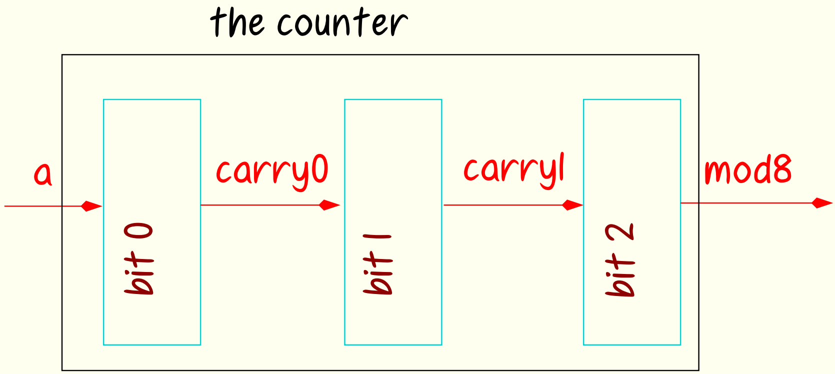
A modulo 8 counter - the program



A modulo 8 counter - the dataflow view



A modulo 8 counter – the dataflow view

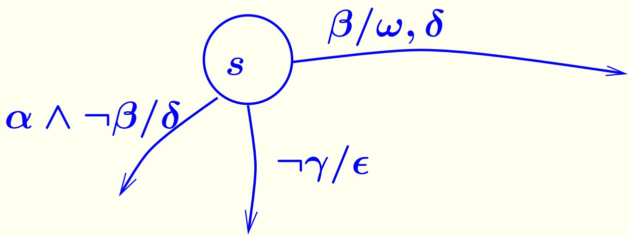


Basic Automata skip details

A $\boldsymbol{signal}$ alphabet: $\boldsymbol{\mathcal{A} = \{\alpha, \beta, \gamma, \ldots\}}$

Basic Automata skip details

A *signal* alphabet: $\mathcal{A} = \{\alpha, \beta, \gamma, \ldots\}$
A Boolean Mealy machine : $M = (S, s_0, I, O, T)$
Inputs/Outputs : $I, O \subseteq \mathcal{A}$,
Transitions : $T \subseteq S \times \mathcal{B}(I) \times 2^O \times S$
$\mathcal{B}(I)$: Boolean Expressions with variables in I



Basic Automata skip details

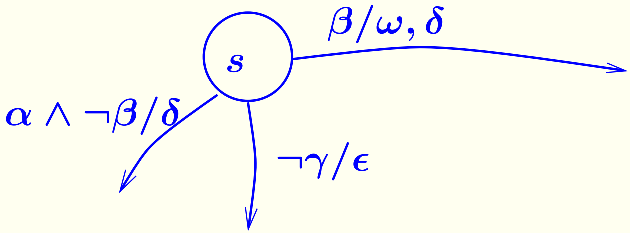
A *signal* alphabet: $\mathcal{A} = \{\alpha, \beta, \gamma, ...\}$

A Boolean Mealy machine : $M = (S, s_0, I, O, T)$

Inputs/Outputs : $I, O \subseteq \mathcal{A}$,

Transitions : $T \subseteq S \times \mathcal{B}(I) \times 2^O \times S$

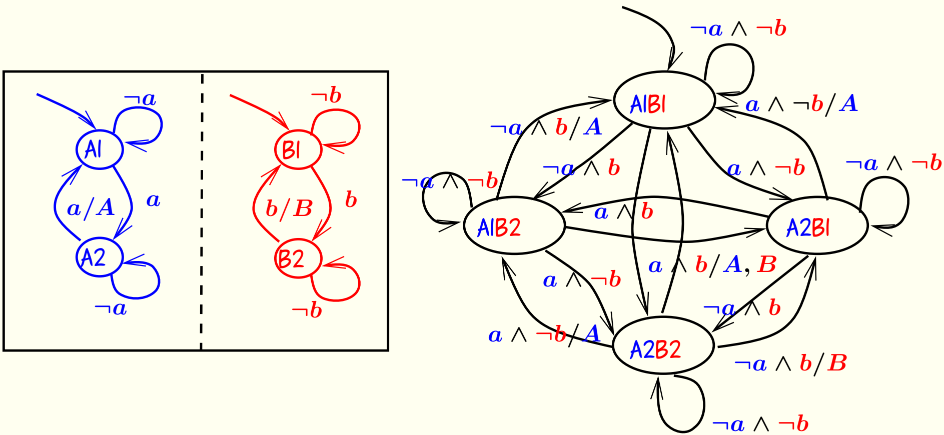
$\mathcal{B}(I)$: Boolean Expressions with variables in I



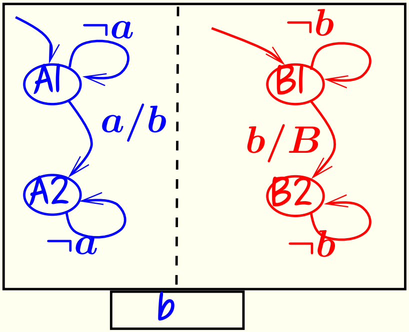
+ determinism and reactivity

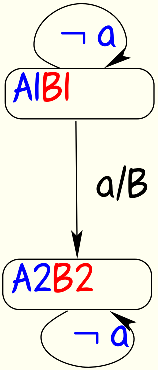
Parallel composition with no synchronization

Cartesian product with conjunction of guards, union of output sets.

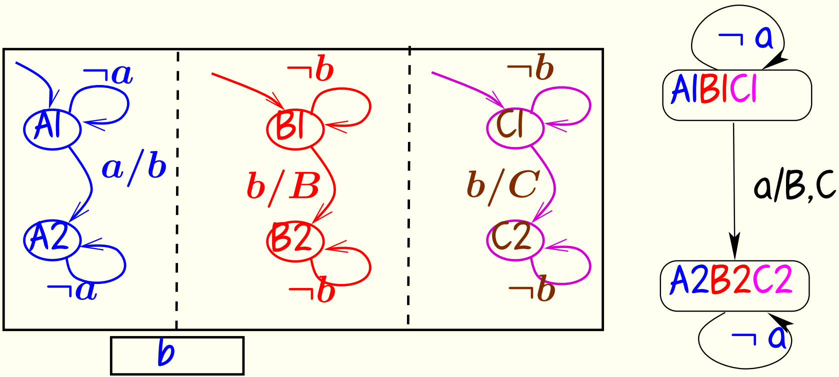


Synchronization: what we want to obtain (1)

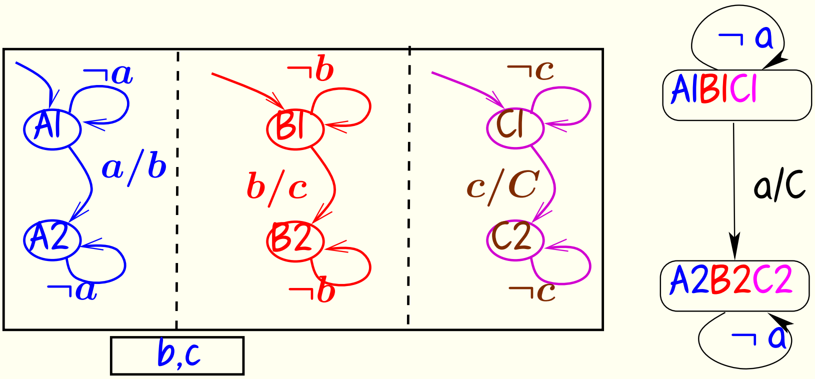




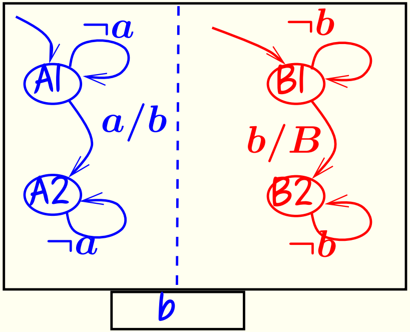
Synchronization: what we want to obtain (2)

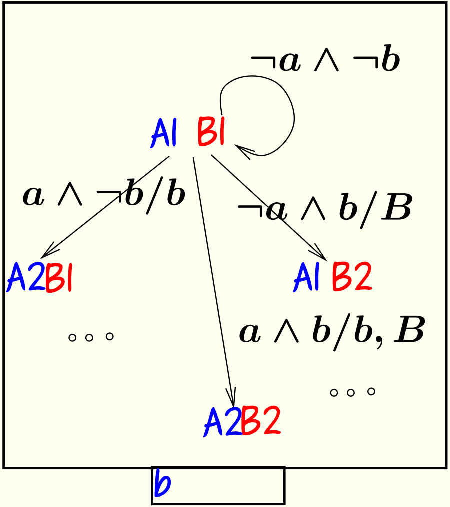


Synchronization: what we want to obtain (3)



Synchronization: the encapsulation



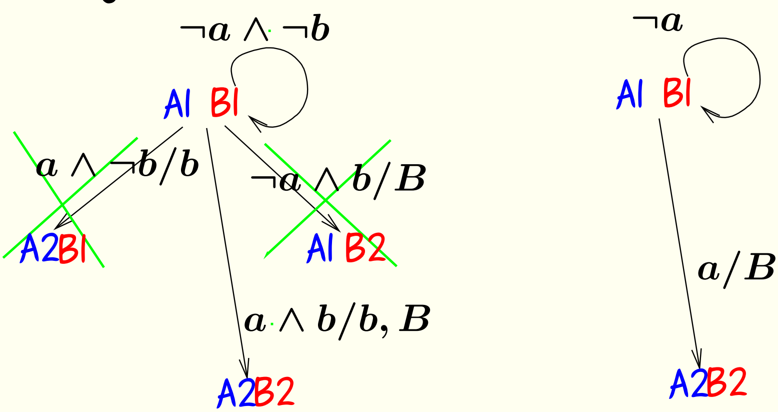


Synchronization: the encapsulation

Keep the transition c/e if and only if :

$(b \in e \Longrightarrow c \wedge b \neq \text{false}) \wedge (b \notin e \Longrightarrow c \wedge \neg b \neq \text{false})$

+ hiding of b.



Candidate Aspects

The synchronous broadcast is very powerful... yet, some transformations seem difficult to implement in a structural fashion.

Candidate Aspects

The synchronous broadcast is very powerful... yet, some transformations seem difficult to implement in a structural fashion.

- Reinitialize the system on the occurrence of an additional signal `r`

Candidate Aspects

The synchronous broadcast is very powerful... yet, some transformations seem difficult to implement in a structural fashion.

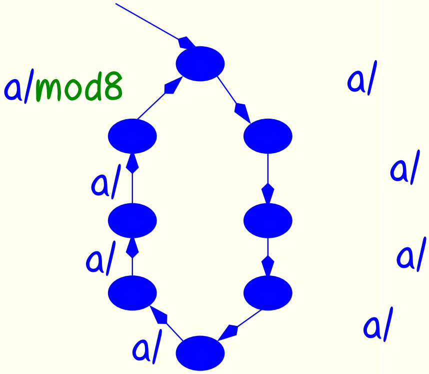
- Reinitialize the system on the occurrence of an additional signal `r`
- clock (or filter) a system so that it does not emit anything when an additional signal is present

Candidate Aspects

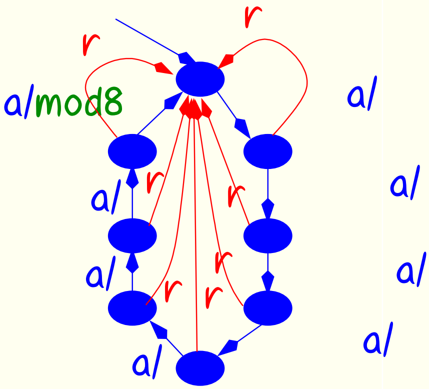
The synchronous broadcast is very powerful... yet, some transformations seem difficult to implement in a structural fashion.

- Reinitialize the system on the occurrence of an additional signal `r`
- clock (or filter) a system so that it does not emit anything when an additional signal is present
- Add a validity bit to each input, and output a default value instead of the value computed by the system, whenever the validity bit is false.

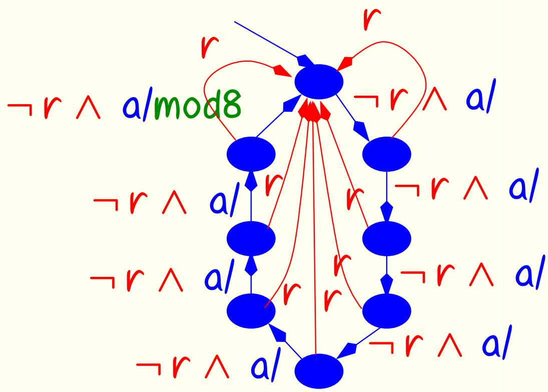
Reinitialize the counter with `r`



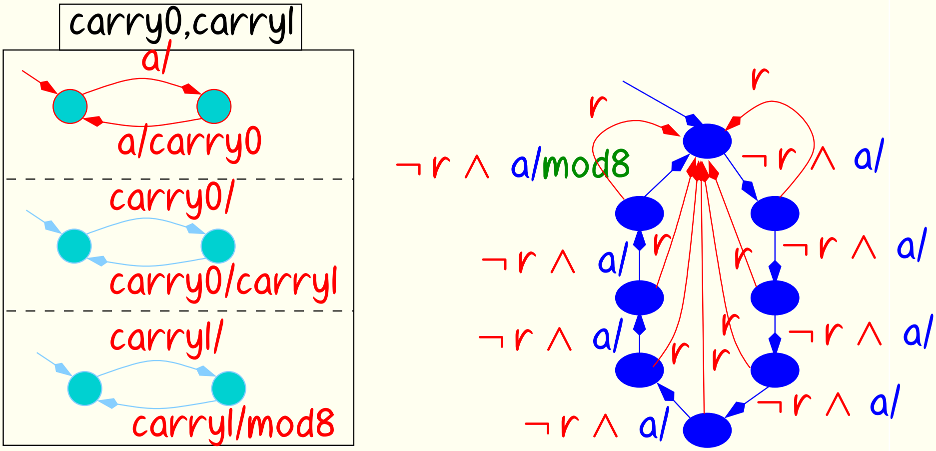
Reinitialize the counter with `r`



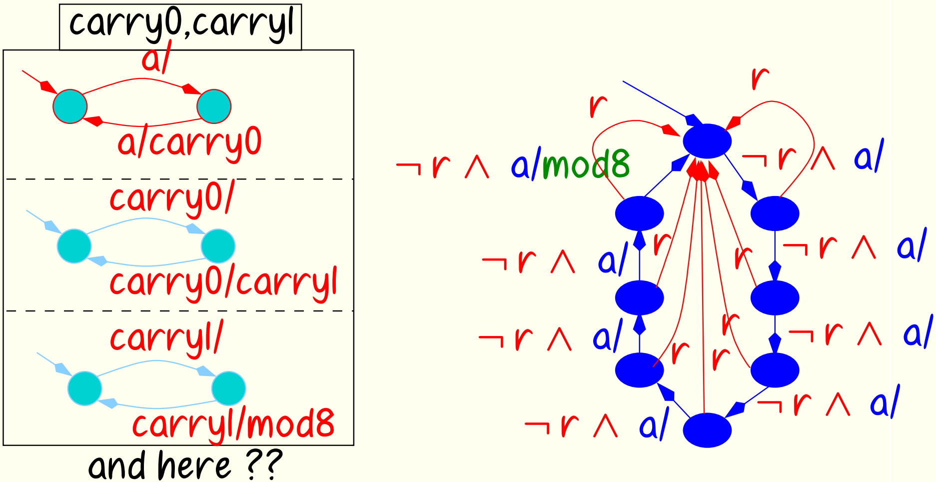
Reinitialize the counter with `r`



Reinitialize the counter with r



Reinitialize the counter with `r`



A Declarative Setting (I)

Start from a program $\boldsymbol{P}$
with inputs $\boldsymbol{I} \cup \boldsymbol{I'}$ and outputs $\boldsymbol{O} \cup \boldsymbol{O'}$.

A Declarative Setting (I)

Start from a program P
with inputs $I \cup I'$ and outputs $O \cup O'$.

Define an aspect A by:
• additional inputs and outputs I'' and O''
• a set of traces on $I' \cup I''$ and $O' \cup O''$ defining the semantics of $P \triangleleft$ A

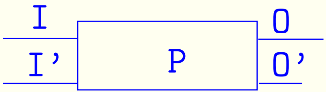
A Declarative Setting (I)

Start from a program P
with inputs $I \cup I'$ and outputs $O \cup O'$.

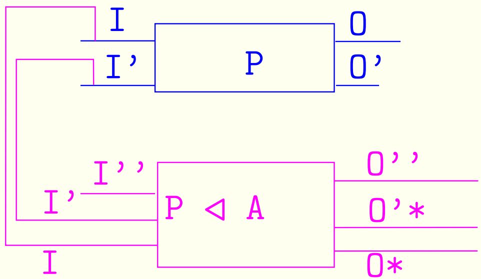
Define an aspect A by:
- additional inputs and outputs I'' and O''
- a set of traces on $I' \cup I''$ and $O' \cup O''$ defining the semantics of $P \triangleleft$ A

The set of traces may be specified by a temporal-logic formula, or an reactive synchronous observer, or ...

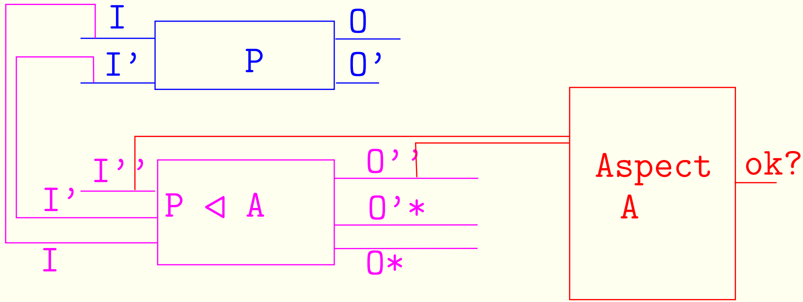
A Declarative Setting (2)



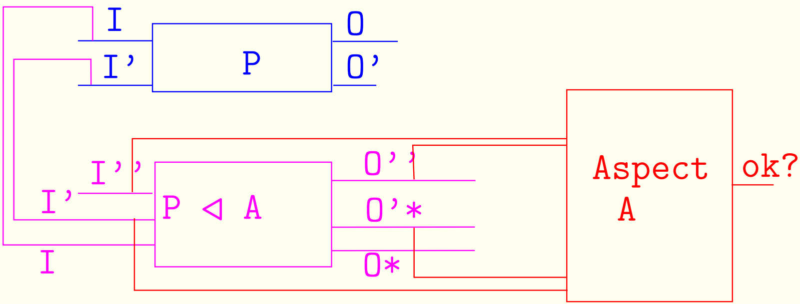
A Declarative Setting (2)



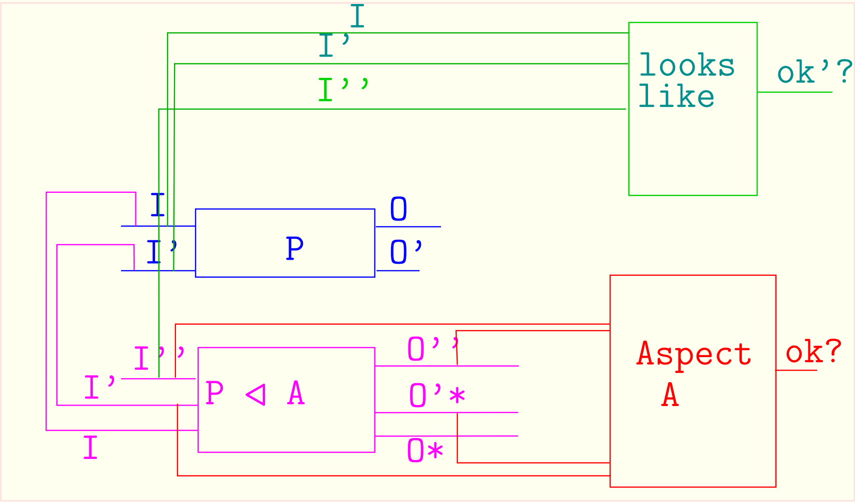
A Declarative Setting (2)



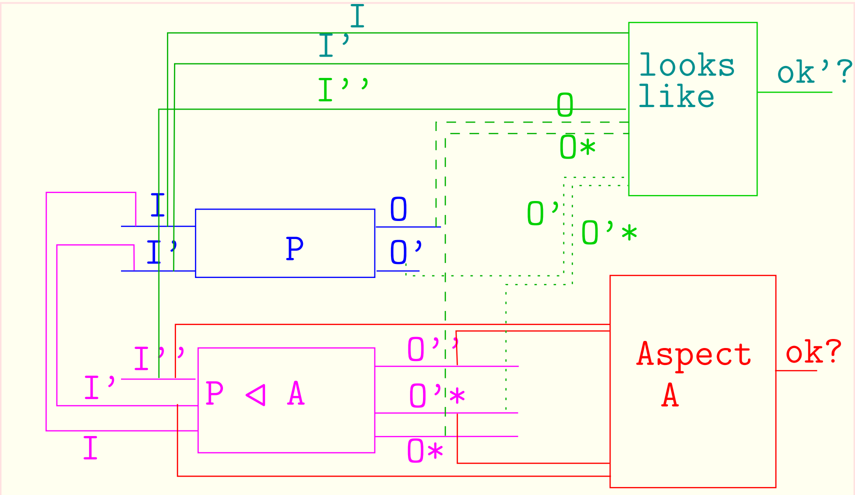
A Declarative Setting (2)



A Declarative Setting (2)



A Declarative Setting (2)



A Declarative Setting (3)

Comparing the behaviours of P and $P \triangleleft A$:

- Projecting on a set of variables

A Declarative Setting (3)

Comparing the behaviours of $\boldsymbol{P}$ and $\boldsymbol{P} \triangleleft A$:

- Projecting on a set of variables
- Projecting on a set of instants in time

A Declarative Setting (3)

Comparing the behaviours of P and $P \triangleleft A$:

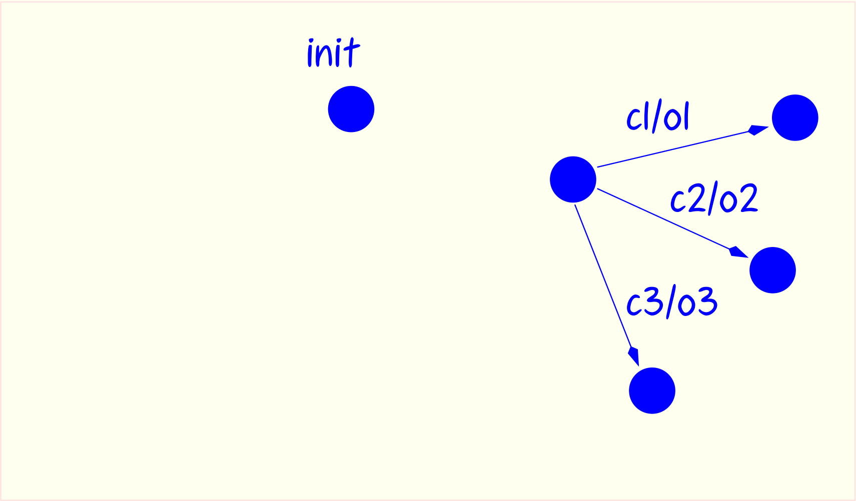
- Projecting on a set of variables
- Projecting on a set of instants in time
- Accepting time shifts ($P \triangleleft A$ responds later than P)

A Declarative Setting (3)

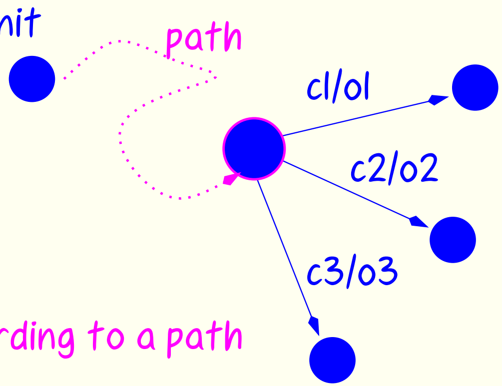
Comparing the behaviours of P and $P \lhd A$:

- Projecting on a set of variables
- Projecting on a set of instants in time
- Accepting time shifts ($P \lhd A$ responds later than P)
- A combination of these three criteria
- ...

Weaving Mechanism (I)

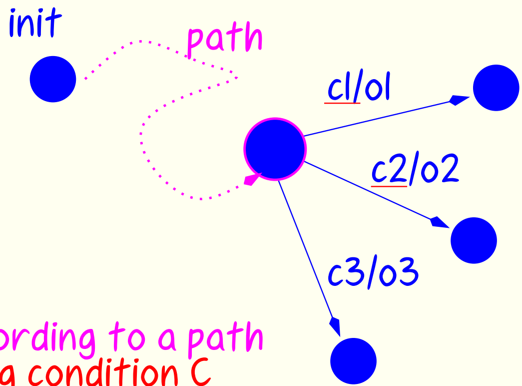


Weaving Mechanism (I)



select states according to a path

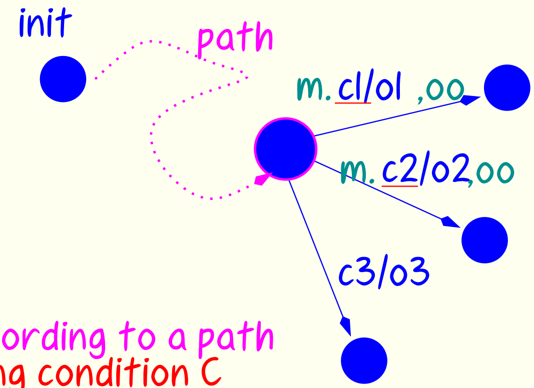
Weaving Mechanism (I)



select states according to a path

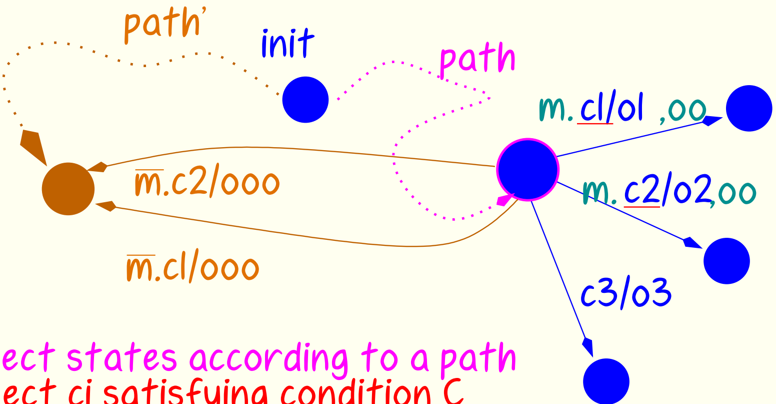
select ci satisfying condition C

Weaving Mechanism (I)



select states according to a path
select ci satisfying condition C
Reinforce the ci's by m, add outputs

Weaving Mechanism (I)

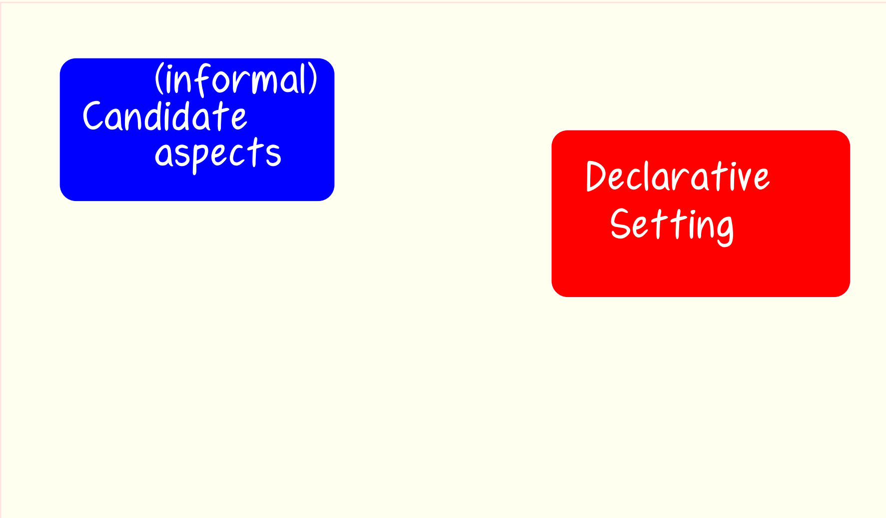


select states according to a path
select ci satisfying condition C
Reinforce the ci's by m, add outputs
Complete the automaton (need ooo, path')

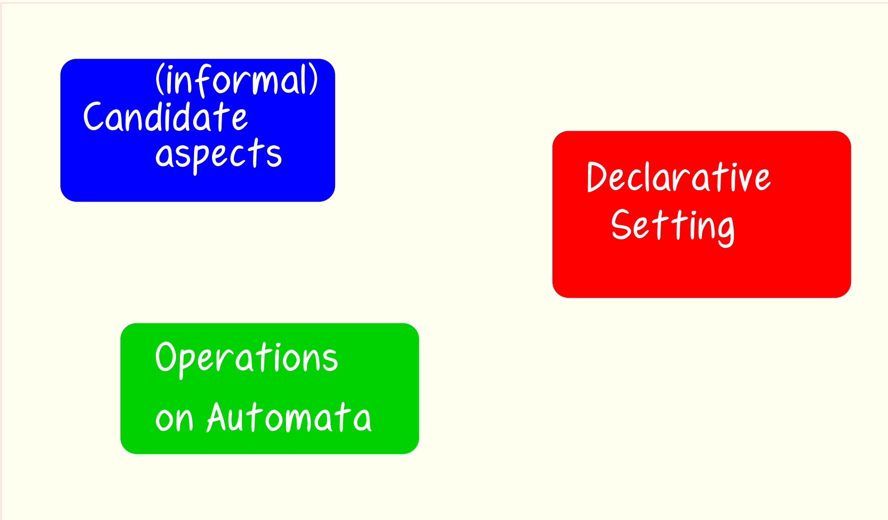
The Global Picture

(informal)
Candidate
aspects

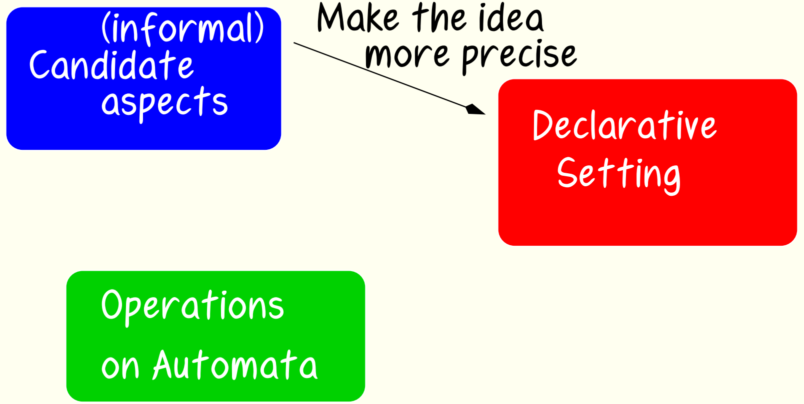
The Global Picture



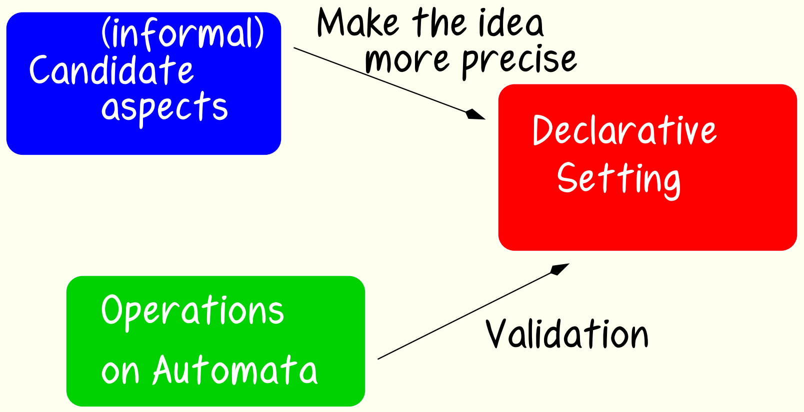
The Global Picture



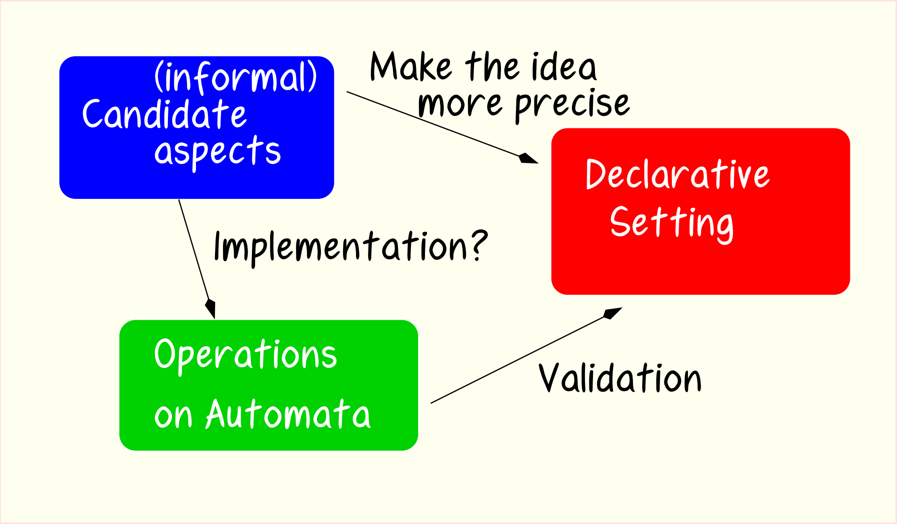
The Global Picture



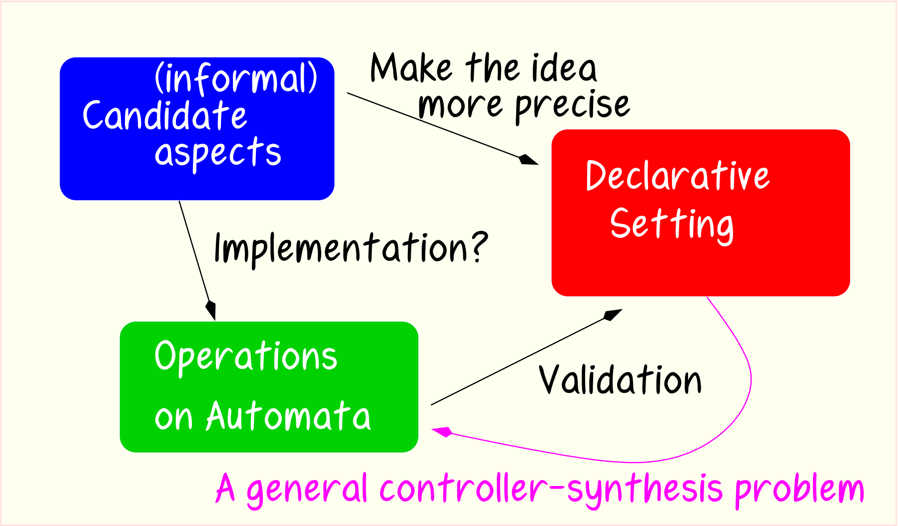
The Global Picture



The Global Picture



The Global Picture



An Open Question

For any of the candidate aspects, would it be possible to implement it using existing constructs ?

An Open Question

For any of the candidate aspects, would it be possible to implement it using existing constructs ?

Given a program P, and a new signal r, is there a context $\mathcal{C}$ of existing operators (parallel, encapsulation) such that:

$\mathcal{C}[P\,]$ behaves as: P reinitialized when r ?

An Open Question

For any of the candidate aspects, would it be possible to implement it using existing constructs ?

Given a program $\boldsymbol{P}$, and a new signal $\mathtt{r}$, is there a context $\mathcal{C}$ of existing operators (parallel, encapsulation) such that:

$\mathcal{C}[\boldsymbol{P}\,]$ behaves as: $\boldsymbol{P}$ reinitialized when $\mathtt{r}$?
(for example, using feedback ?)

An Open Question

For any of the candidate aspects, would it be possible to implement it using existing constructs ?

Given a program P, and a new signal r, is there a context $\mathcal{C}$ of existing operators (parallel, encapsulation) such that:

$\mathcal{C}[P\ ]$ behaves as: P reinitialized when r ?
(for example, using feedback ?)

can be studied on an example, but how to characterize what cannot be implemented with existing constructs?

Conclusion, further work

- The general setting is almost ok
- Try to find a minimal set of automata transformations to implement aspect weaving and validate them according to the declarative setting.
- Find real-life examples that could benefit from the AOP point of view