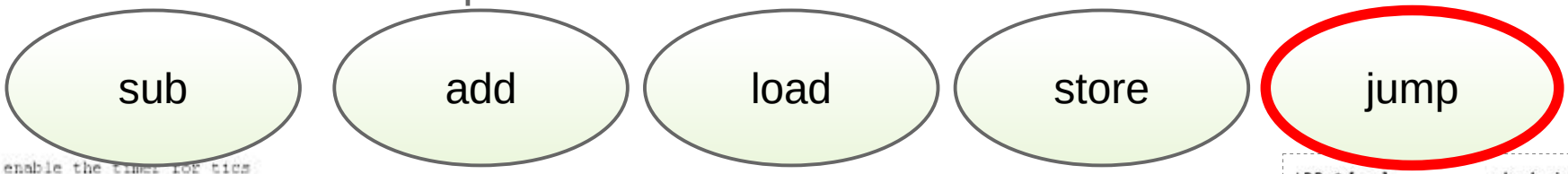
Behavioral Semantics of Languages

digest

Julien Deantoni
Universite Cote d'Azur,
CNRS I3S, INRIA KAIROS
Julien.deantoni@polytech.unice.fr

Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects



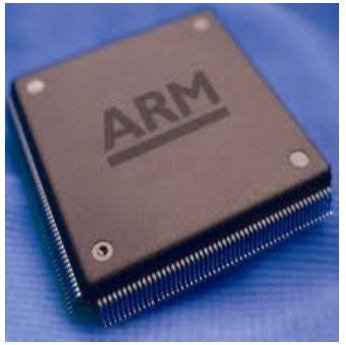

```
;
;  enable the timer for tics
;
ticson:
  ld      a,0
  out0    tmdr01  ; set timer count lo=0
  out0    rldr01  ; set reload
  ld      a,tmrtic
  out0    tmdr0h  ; set reload and count=1e00=60 tics/sec
  out0    rldr0h
  ld      a,11h
  out0    tcr ; enable timer 0 + interrupts
  ret
```

```
{
203FA354: 7C0802A6  mflr    r0
203FA358: 90010004  stw     r0,4(SP)
203FA35C: 9421FFE0  stwu    SP,-32(SP)
203FA360: 39610020  addi    r11,SP,32
203FA364: 4800022D  bl      __save_gpr+0x34 (0x203fa590)
    char    *   pDIPSwitch1;
    char    *   pDIPSwitch2;
    char    *   p7Seg;
    char        byVal;
    int         j;
```

```
LDR r0,[p_a]        ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]        ; load b into r1
ADD r3,r0,r1        ; compute a + b
STR r3,[p_w]        ; w = a + b
LDR r2,[p_c]        ; load c into r2
ADD r0,r2,r3        ; compute c + w, reusing r0 for x
STR r0,[p_x]        ; x = c + w
LDR r0,[p_d]        ; load d into r0
ADD r3,r2,r0        ; compute c + d, reusing r3 for y
STR r3,[p_y]        ; y = c + d
```

Avant....

Automatic generation / compilation



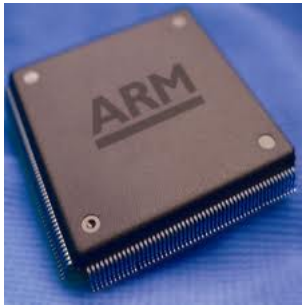
Reifying good concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

```
nodename = getNodename()
label=symbol.sym_name.get(int(ast[0]),ast[0])
print '    %s [label="%s' % (nodename, label),
if isinstance(ast[1], str):
   if ast[1].strip():
      print '= %s"];' % ast[1]
   else:
      print '"]'
else:
   print '"];'
   children = []
   for n, child in enumerate(ast[1:]):
      children.append(dotwrite(child))
   print '    %s -> {' % nodename,
   for name in children:
      print '%s' % name,
```

Avant hier....

Automatic generation / compilation



Reifying **good** concepts

- Modèle représentant un certain code exécutable en abstrayant certains aspects

GOTO ?

At the machine code level, a goto is a form of branch or jump statement.

Although at the pre-ALGOL meeting held in 1959, Heinz Zemanek explicitly threw doubts on the necessity of GOTO statements, at the time no one paid attention to his remark, including Edsger Dijkstra, who would later become the iconic opponent of GOTO. [3] The 1970s and 1980s saw a decline in the use of GOTO statements in favor of the "structured programming" paradigm, with goto criticized as leading to "unmaintainable spaghetti code" (see below).

Source: wikipedia

Reifying **good ?** concepts

Good is a point of view

- Modèle représentant un certain code exécutable en abstrayant certains aspects *dans un but spécifique, pour un domaine particulier*

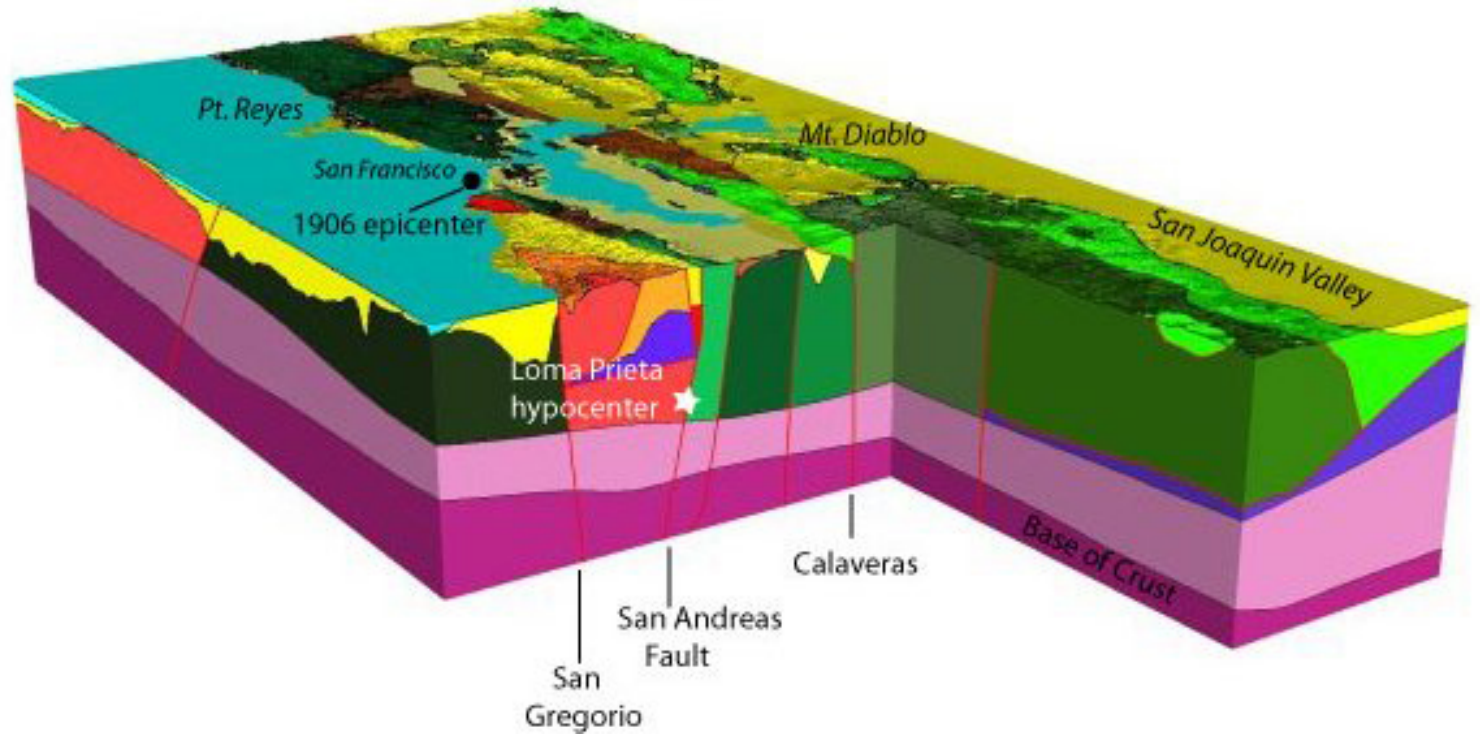
[...]In that letter Dijkstra argued that unrestricted GOTO statements should be abolished from higher-level languages because **they complicated the task of analyzing and verifying the correctness of programs** (particularly those involving loops)

Some programmers, such as Linux Kernel designer and coder Linus Torvalds or software engineer and book author Steve McConnell, also object to Dijkstra's point of view, stating that **GOTOs can be a useful language feature, improving program speed, size and code clearness**, but only when used in a sensible way by a comparably sensible programmer.

Source: wikipedia

Two possible usages of models

- *Descriptive :* Abstraction of an existing reality

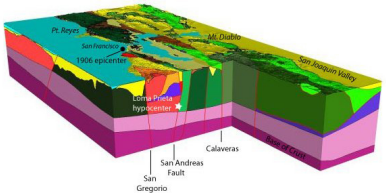


To understand
Analyse
automate
communicate
etc

- *Prescriptive :* Specification of something to be realized

Two usages of models

- *Descriptive :* Abstraction of an existing reality

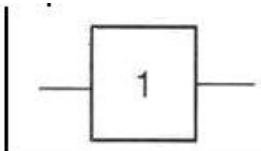
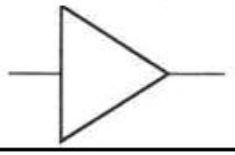
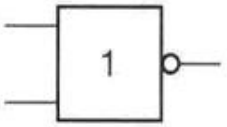
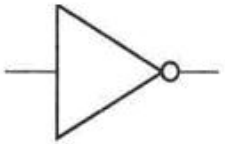
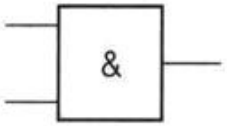
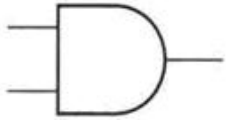
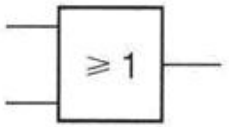
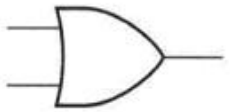


- *Prescriptive :* Specification of something to be realized

To understand
Analyse
automate
communicate
etc

Languages dédiés et modèles

- ...un principe qui a prouvé son efficacité dans de nombreux domaines
- ...permet de communiquer entre personnes comprenant le modèle.

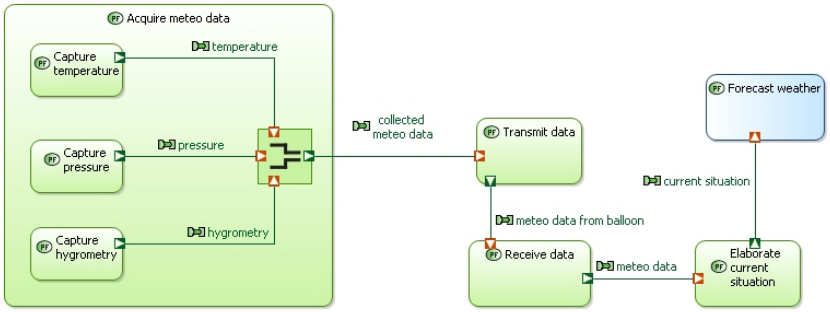
Porte OUI (YES)			entrée sortie 0 0 1 1
Porte NON (NO)			entrée sortie 0 1 1 0
Porte ET (AND)			entrées sortie 0 0 0 0 1 0 1 0 0 1 1 1
Porte OU (OR)			entrées sortie 0 0 0 0 1 1 1 0 1 1 1 1

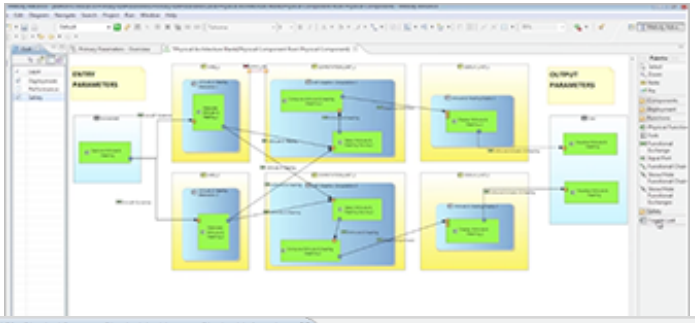
Deux notations pour chaque porte ?!

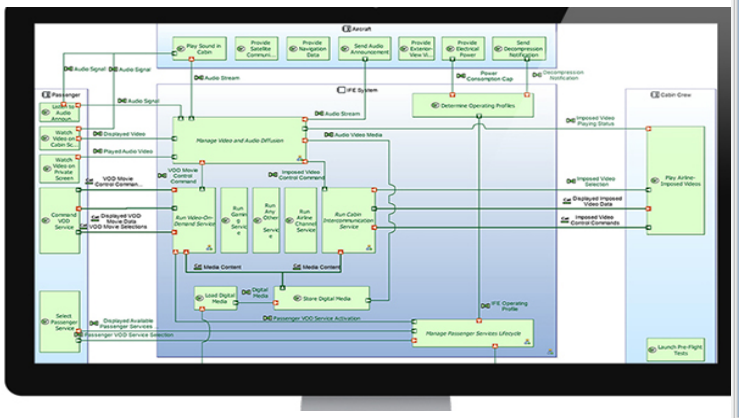
Mais la sémantique est donnée...

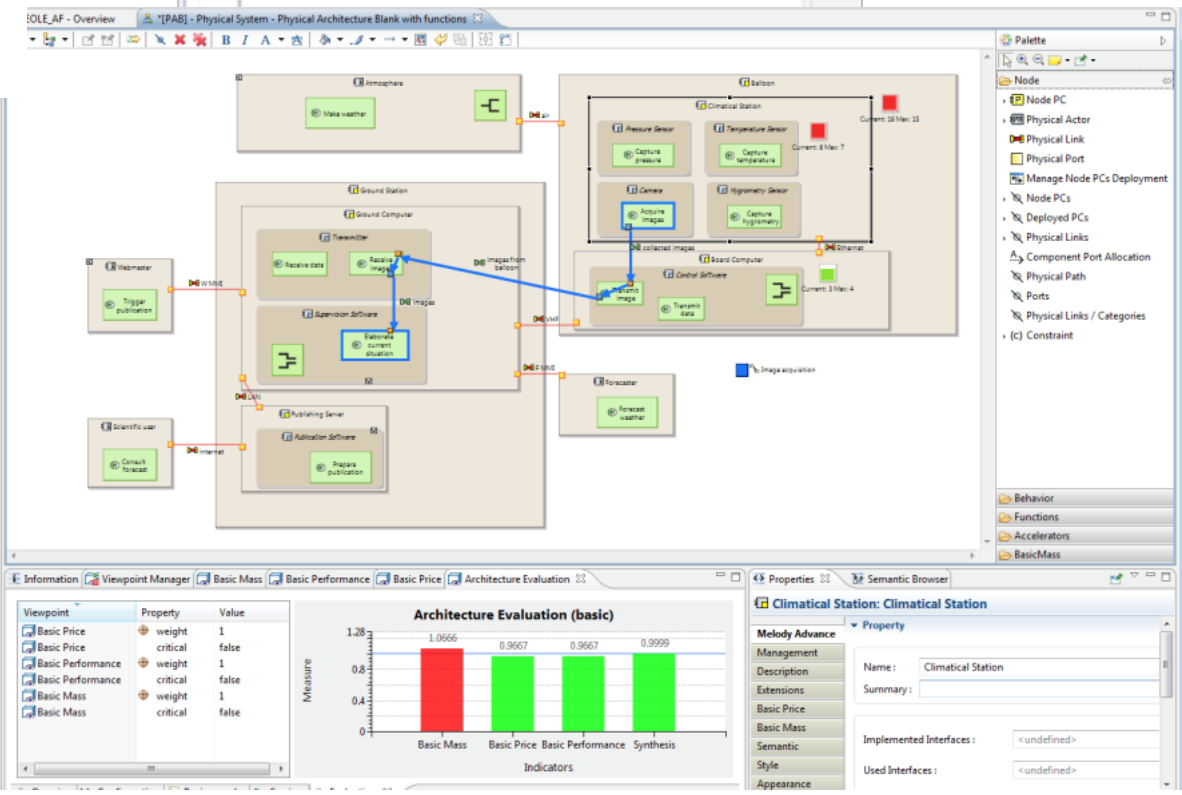
→ toutes les personnes comprenant la logique booléenne ?!

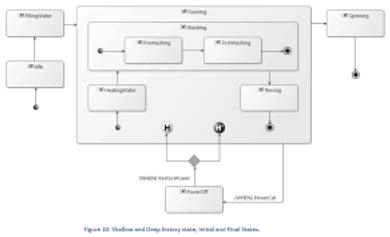
Capella !


Acquire meteo data
Capture temperature
temperature
Capture pressure
pressure
Capture hygrometry
hygrometry
collected meteo data
Transmit data
Forecast weather
current situation
meteo data from balloon
Receive data
meteo data
Elaborate current situation



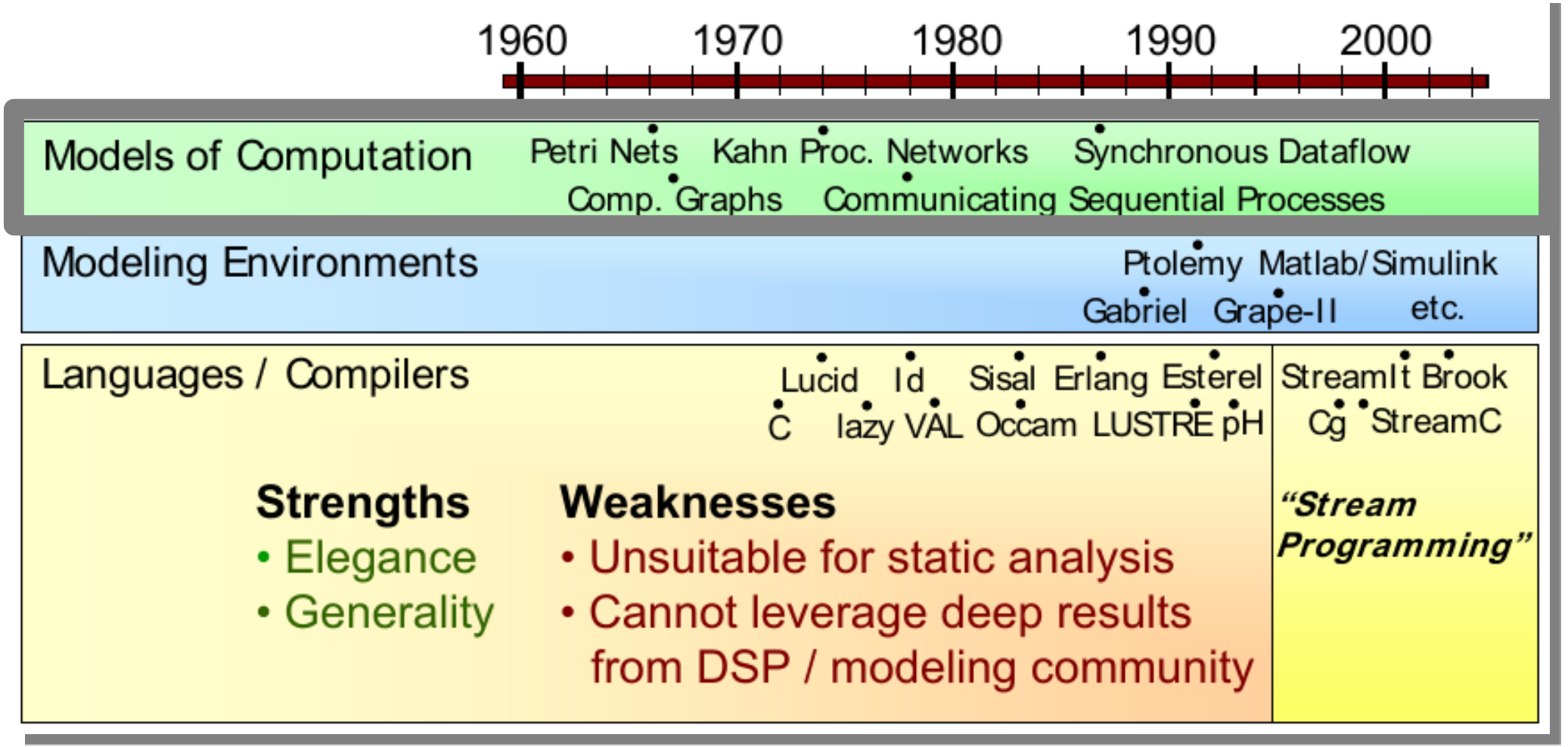



Palette
Node
Node PC
Physical Actor
Physical Link
Physical Port
Manage Node PCs Deployment
Node PCs
Deployed PCs
Physical Links
Component Port Allocation
Physical Path
Ports
Physical Links / Categories
(c) Constraint
Behavior
Functions
Accelerators
BasicMass



Information
Viewpoint Manager
Basic Mass
Basic Performance
Basic Price
Architecture Evaluation
Viewpoint
Property
Value
Basic Price
weight
1
Basic Price
critical
false
Basic Performance
weight
1
Basic Performance
critical
false
Basic Mass
weight
1
Basic Mass
critical
false
Architecture Evaluation (basic)
Measure
1.28
0.8
0.4
0
1.0666
0.9667
0.9667
0.9999
Basic Mass
Basic Price
Basic Performance
Synthesis
Indicators
Properties
Semantic Browser
Climatical Station: Climatical Station
Melody Advance
Management
Description
Extensions
Basic Price
Basic Mass
Semantic
Style
Appearance
Property
Name :
Climatical Station
Summary :
Implemented Interfaces :
<undefined>
Used Interfaces :
<undefined>

Other kinds of models

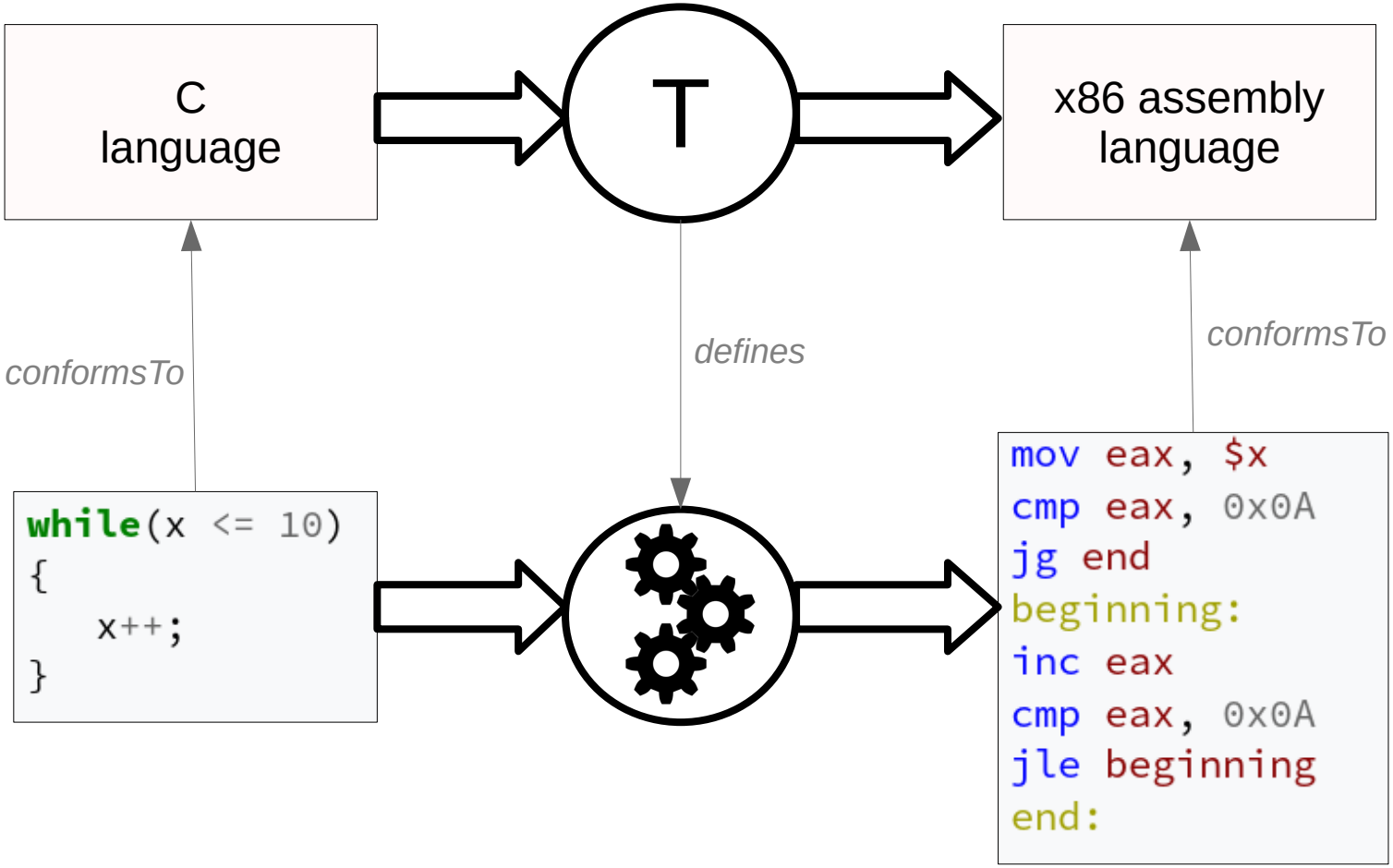


Modèles d'abstraction forte, plus proche de l'analyse du problème que de la description de la solution
→ ***réel impact sur l'informatique... par le côté théorique***

William Thies (phd thesis)

Transformational semantics

- **Transformational :** the semantics is defined by reducing constructs of the language to more elementary ones by means of definitional transformations into a simpler language whose the semantics is already given.



Transformational semantics

https://stackoverflow.com/questions/35716868/gcc-o2-optimize-to-wrong-behavior-is-it-a-bug

```c
#include <stdio.h>

int main(int argc, char **argv)
{
    unsigned long i = 0;
    while (1) {
        if (++i > 0x1fffffffUL) {
            printf("hello\n");
            i = 0;
        }
    }
}
```

gcc -O1

```
Disassembly of section .text:

    00000000 <_main>:
    #include <stdio.h>

    int main(int argc, char **argv)
    {
       0:   55                      push   %ebp
       1:   89 e5                   mov    %esp,%ebp
       3:   83 e4 f0                and    $0xfffffff0,%esp
       6:   83 ec 10                sub    $0x10,%esp
       9:   e8 00 00 00 00          call   e <_main+0xe>
       e:   b8 00 00 00 20          mov    $0x20000000,%eax
        unsigned long i = 0;
        while (1) {
            if (++i > 0x1fffffffUL) {
      13:   83 e8 01                sub    $0x1,%eax
      16:   75 fb                   jne    13 <_main+0x13>
                printf("hello\n");
      18:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
      1f:   e8 00 00 00 00          call   24 <_main+0x24>
      24:   eb e8                   jmp    e <_main+0xe>
      26:   90                      nop
      27:   90                      nop
```

gcc -O2

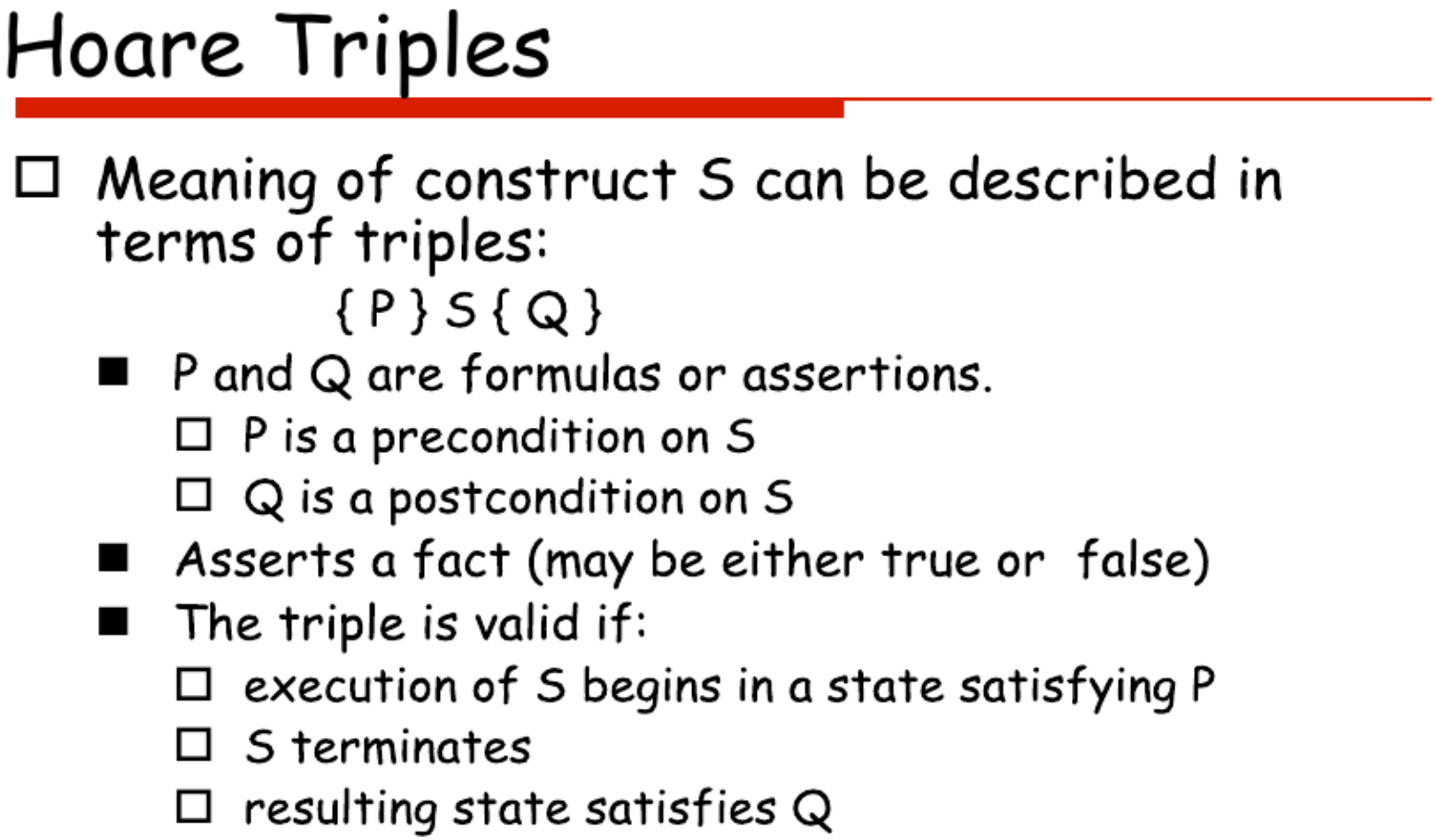
```
Disassembly of section .text.startup:

00000000 <_main>:
#include <stdio.h>

int main(int argc, char **argv)
{
   0:   55                      push   %ebp
   1:   89 e5                   mov    %esp,%ebp
   3:   83 e4 f0                and    $0xfffffff0,%esp
   6:   83 ec 10                sub    $0x10,%esp
   9:   e8 00 00 00 00          call   e <_main+0xe>
   e:   66 90                   xchg   %ax,%ax
  10:   c7 04 24 00 00 00 00    movl   $0x0,(%esp)
  17:   e8 00 00 00 00          call   1c <_main+0x1c>
  1c:   eb f2                   jmp    10 <_main+0x10>
  1e:   90                      nop
  1f:   90                      nop
```

Axiomatic semantics

- **Axiomatic :** the semantics is defined by a logical theory associated to each language elements in order to enable some properties to be proven (the formulae describe, for each statement, the relation between the pre-state and the post-state of the executing the statement)



http://www.cs.purdue.edu/homes/suresh/565-Spring2009/lectures/lecture-6.pdf

Operational semantics

- The operational semantics for a programming language describes how a valid program is interpreted as sequences of computational steps. These sequences then are the meaning of the program.
- Structural Operational Semantics [http://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf]

Condition

Rewriting rule

$\langle n, \sigma \rangle \Downarrow n$ "n in state σ, evaluates to n"

$\langle a, \sigma \rangle \Downarrow n$ "expression a in state σ, evaluates to n"

$\langle X, \sigma \rangle \Downarrow \sigma(X)$ "location X evaluates to its contents in a state"

$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{false}}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \Downarrow \sigma} \quad \text{(while loops)}$$

while (b)
do
    C ;
done

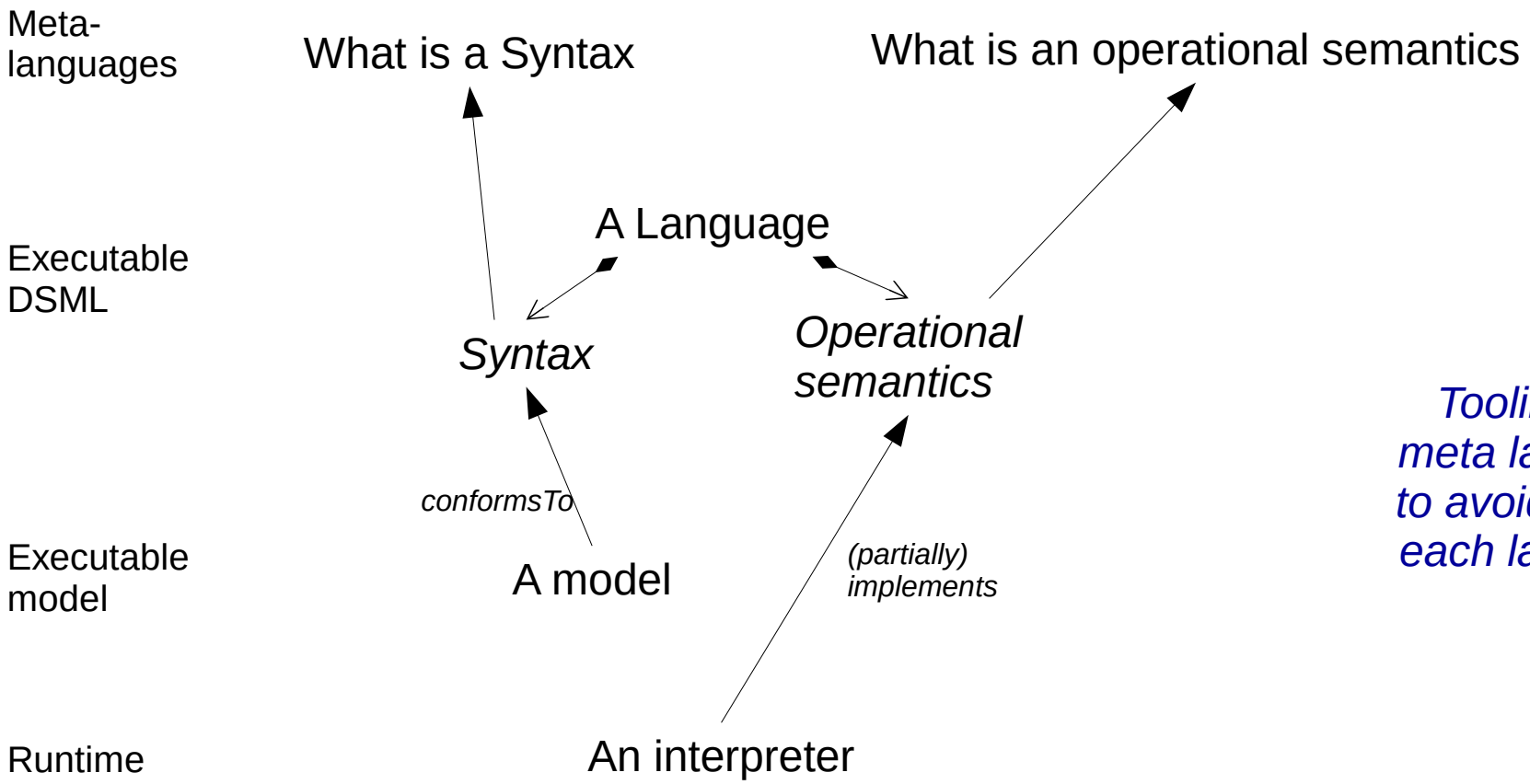
$$\frac{\langle b, \sigma \rangle \Downarrow \mathbf{true} \quad \langle c, \sigma \rangle \Downarrow \sigma'' \quad \langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma'' \rangle \Downarrow \sigma'}{\langle \mathbf{while}\ b\ \mathbf{do}\ c, \sigma \rangle \Downarrow \sigma'}$$

$$\frac{\langle B, s \rangle \Rightarrow \mathbf{true}}{\langle \mathbf{while}\ B\ \mathbf{do}\ C, s \rangle \longrightarrow \langle C; \mathbf{while}\ B\ \mathbf{do}\ C, s \rangle} \qquad \frac{\langle B, s \rangle \Rightarrow \mathbf{false}}{\langle \mathbf{while}\ B\ \mathbf{do}\ C, s \rangle \longrightarrow s}$$

From wikipedia

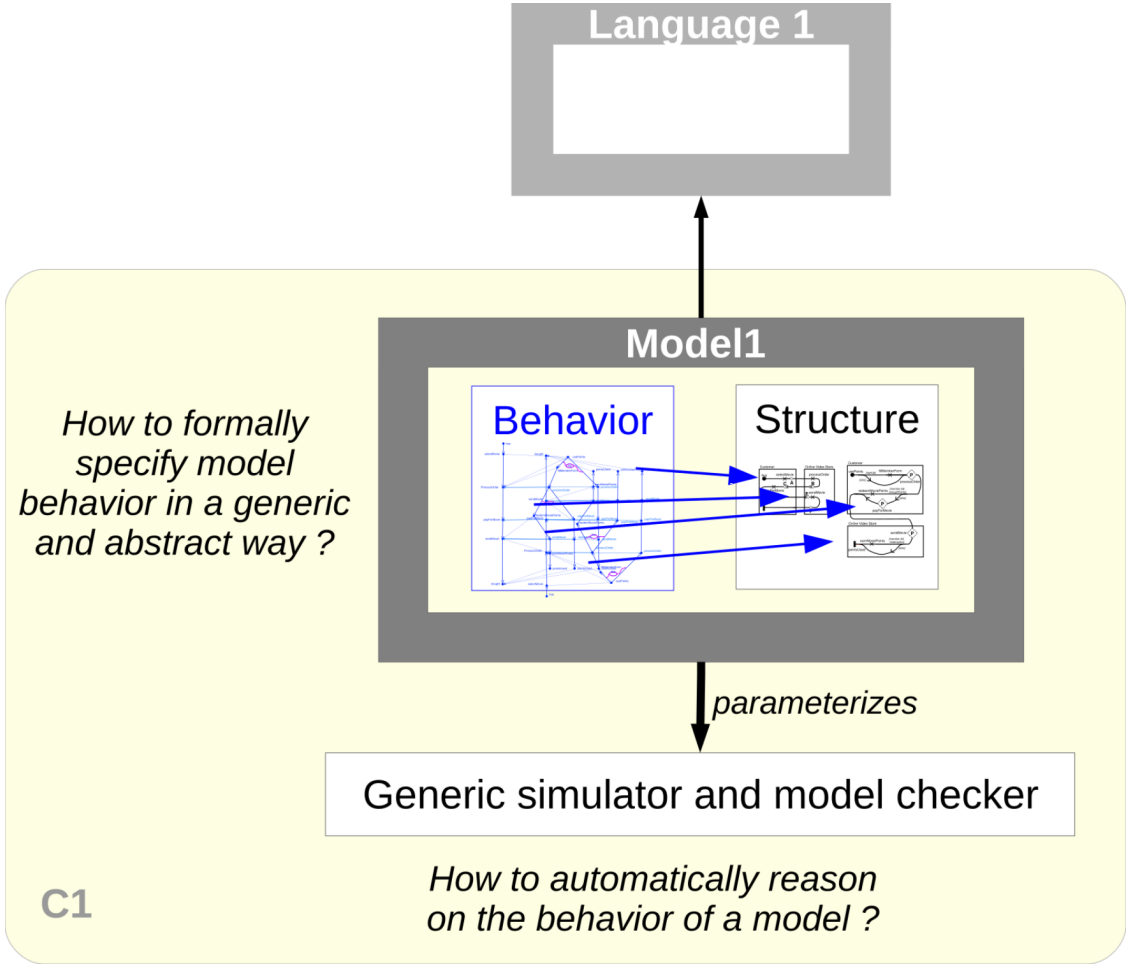
GEMOC approach : context

- We consider models that can be interpreted according to their (concurrent and timed) operational semantics
- We do not want to implement all the tooling for each new language



Tooling the meta language to avoid tooling each language

Globalization of Modelling langages : Challenge 1



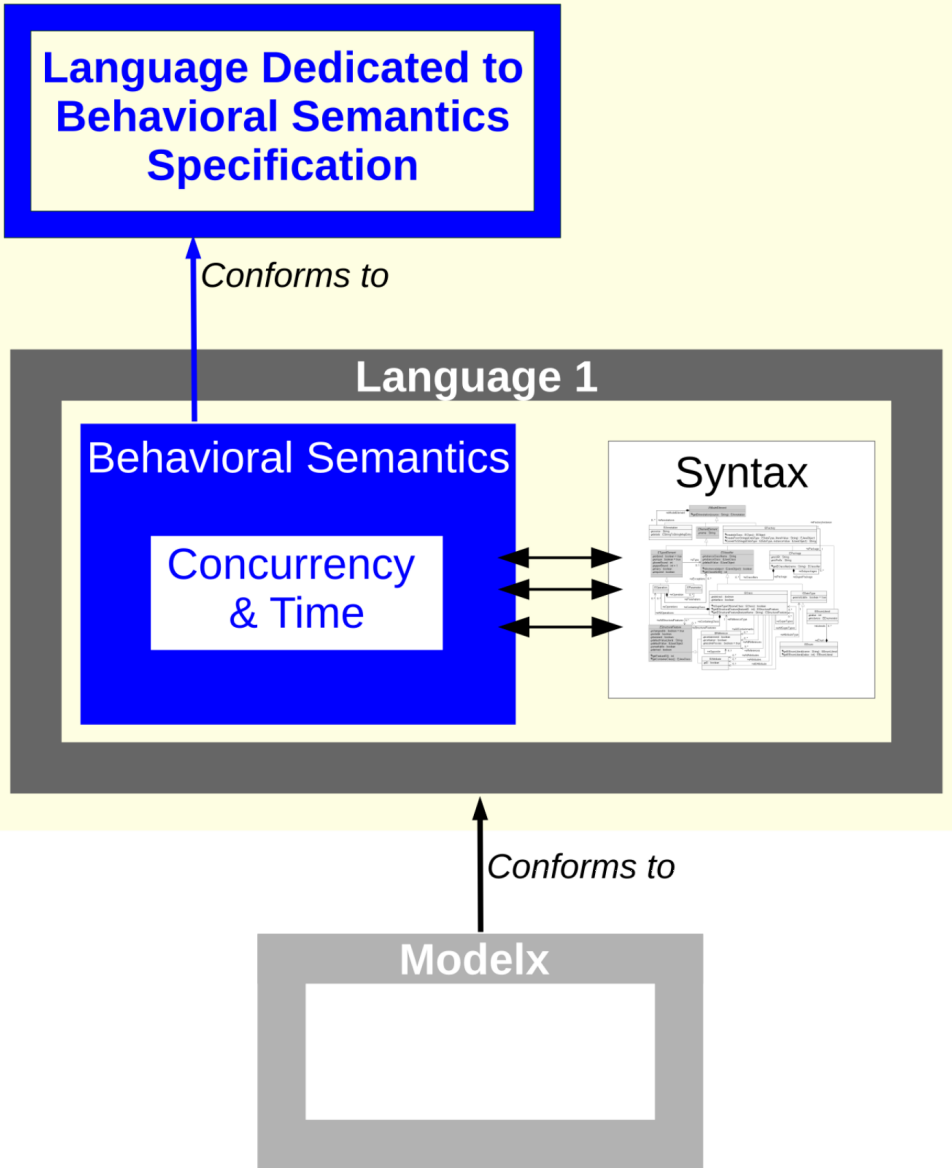
Globalization of Modelling langages : Challenge 3

C3

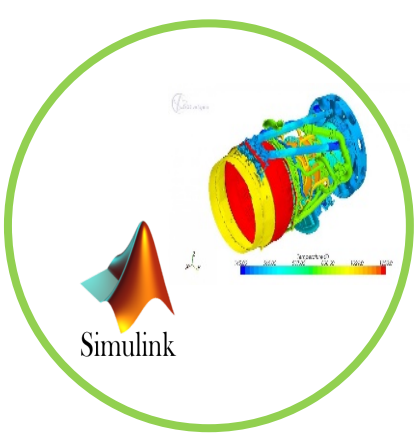
What is (are) the appropriate metalanguage(s) to specify a behavioral semantics ?

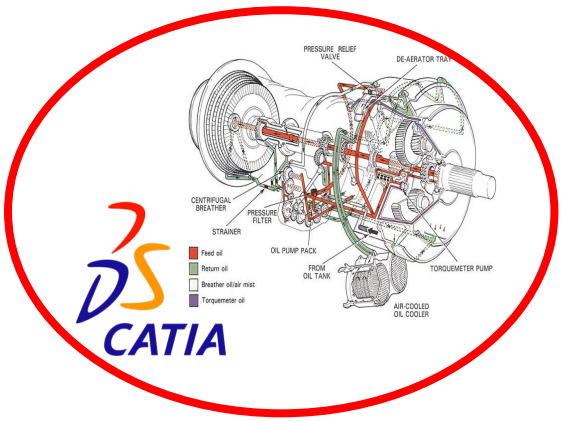
How to make explicit the concurrency and time in the specification of behavioral semantics ?

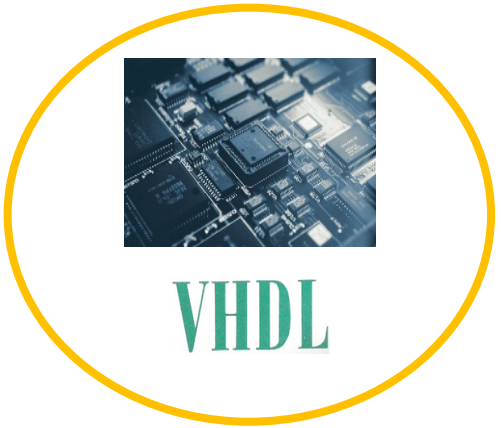
What tooling can be developed to take benefits of a modular description of the semantics ?

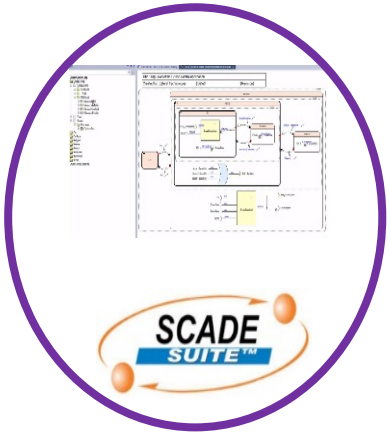


Modeling and Simulation
→ Co-simulation





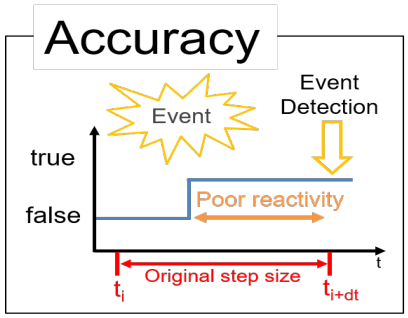




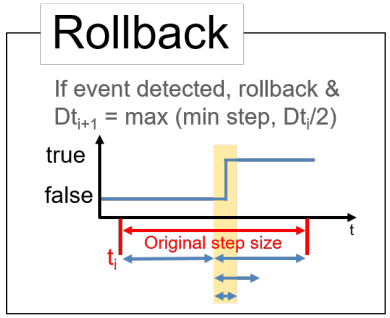
Coordinator

Ask to simulate the individual models
Realize Data propagation
Ensure Time consistency

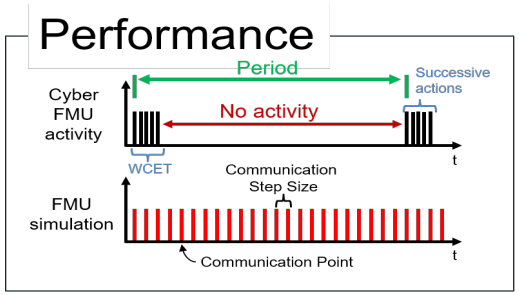
→ Mostly Time triggered and generic coordinators in the State of the Art



(Tavella, 2016)



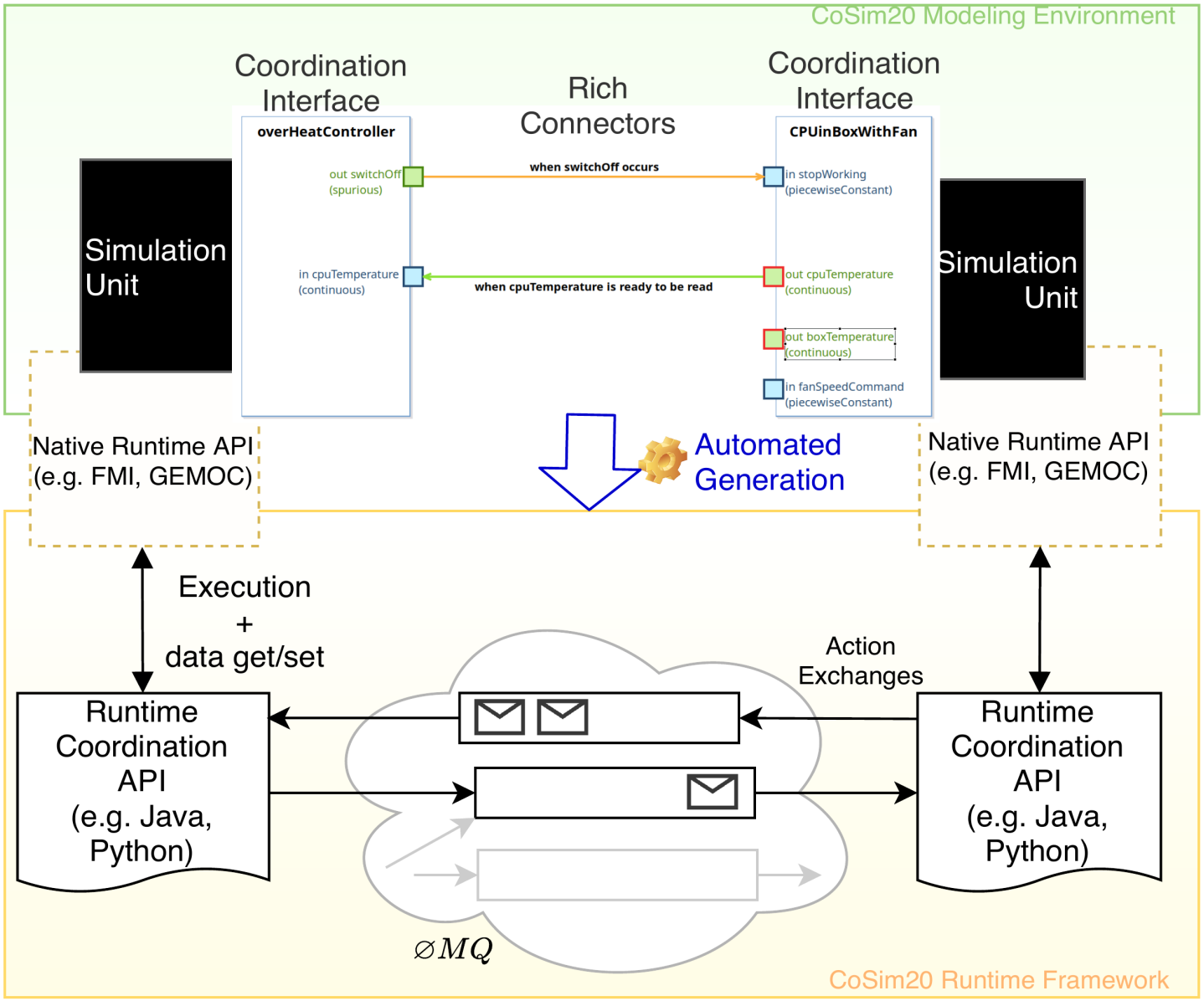
(Tavella, 2016)



(Centomo, 2016)

Modeling and Simulation

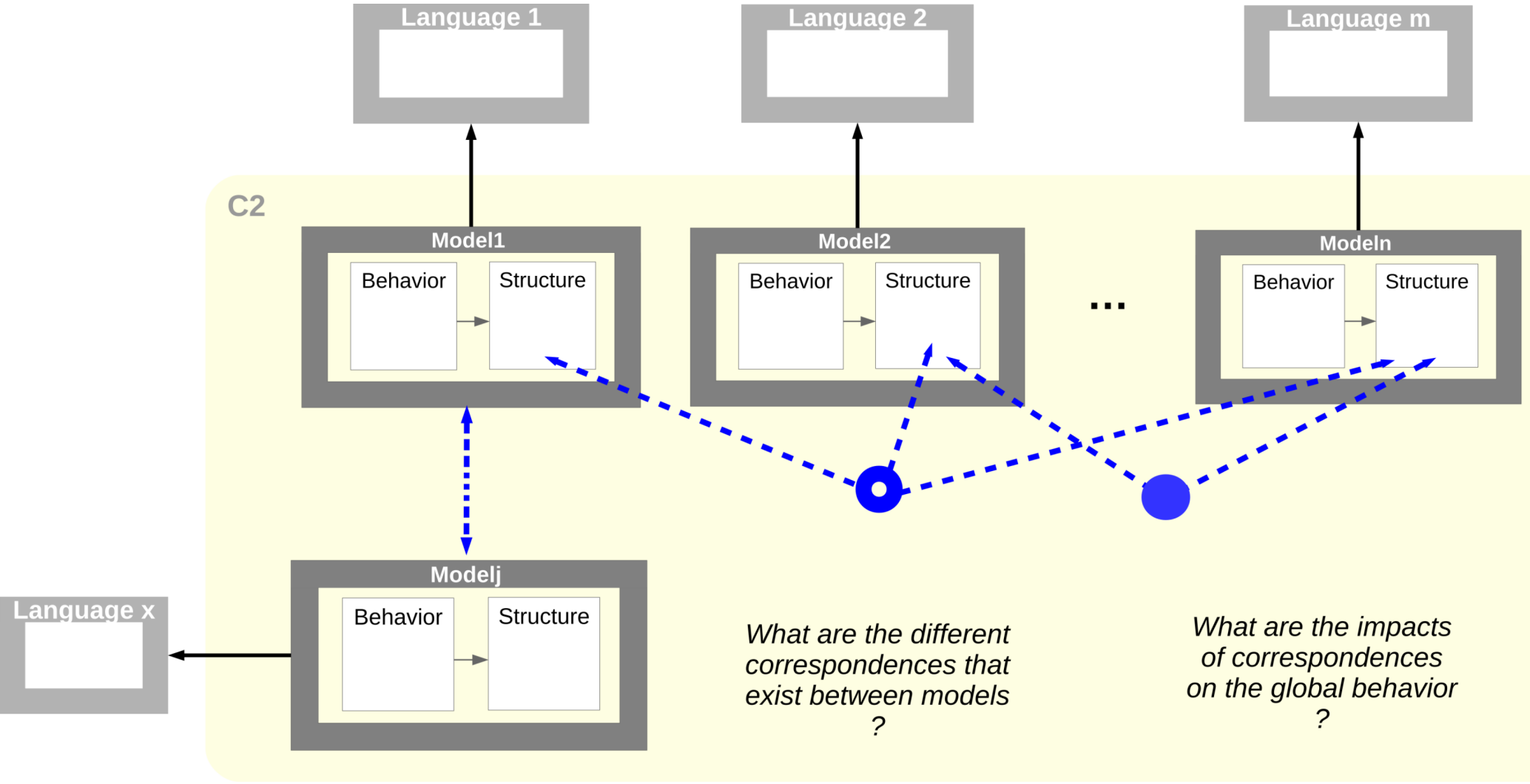
→ the CoSim20 framework (Giovanni Liboni's thesis)



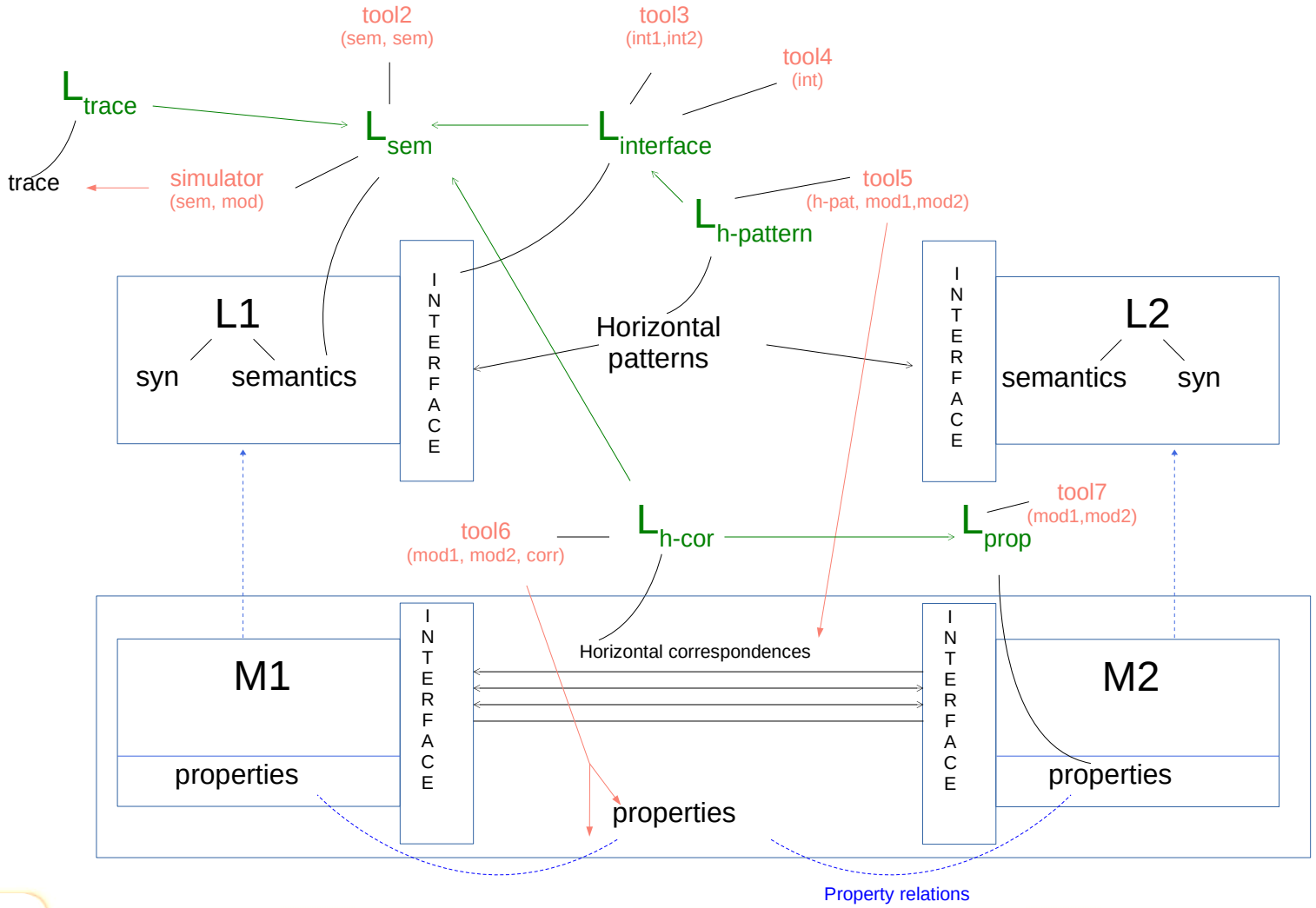
An advanced modeling environment with explicit rich connectors specifying *when* and *how* each data must be coordinated

A distributed runtime framework with a coordinator *tailored* to the system, *mixing timed and event triggered communications*

Globalization of Modelling langages : Challenge 2



Expectations



Expectations