Institut
Mines-Telecom

Static analysis techniques to verify mutual exclusion situations within SysML models

Ludovic Apvrille
ludovic.apvrille@telecom-paristech.fr

Pierre de Saqui-Sannes
pdss@isae.fr

SDL Forum 2013, Montreal, Canada

Outline

Outline

Rationale

Context: Formal verification, SysML

- Formal verification enables early detection of design errors in the life cycle of real-time and distributed systems
- Formal verification tools face state explosion problem when models must be executed
  - SysML models are usually executed to be formally verified

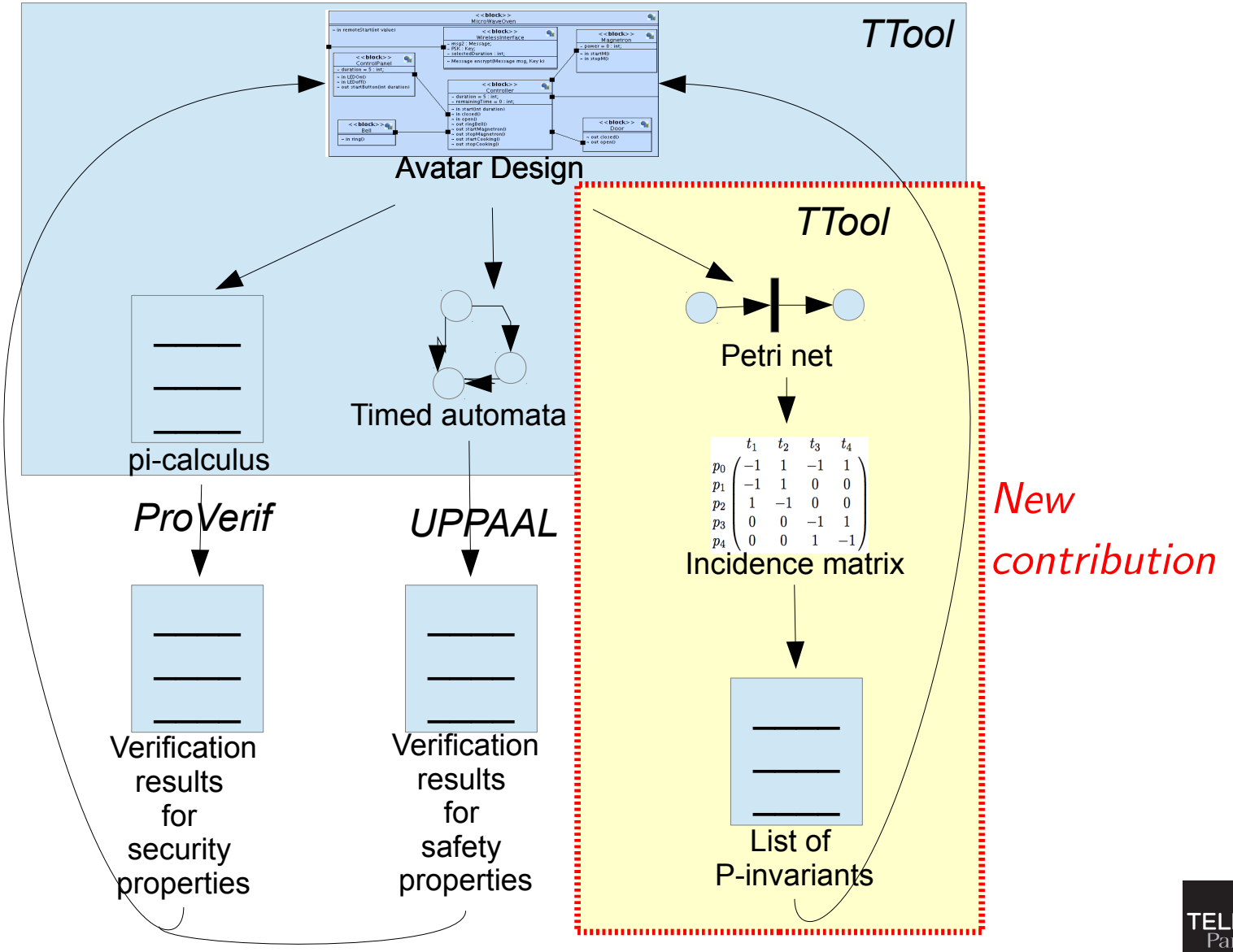
Contribution: static analysis of models

- AVATAR
  - Formal SysML environment
- TTool
  - Toolkit supporting AVATAR
  - Static analysis based on Petri nets invariants

AVATAR and TTool: Method

1. Requirement capture
   - Requirement diagrams: goals, assumptions, requirements
   - Property expression in TEPE (parametric diagrams extension)
2. Analysis
   - Use case driven analysis (use case diagrams)
   - Use cases documented by scenarios (sequence diagrams) and flow charts (activity diagrams)
3. Design
   - Architecture (block instance diagrams)
   - Behaviors (state machines diagrams)
   - Formal verification against the requirements and the properties defined in the AVATAR model
4. Prototyping
   - Generation of C/POSIX executable code
   - Integration with SoCLib

Formal Verification in AVATAR/TTool



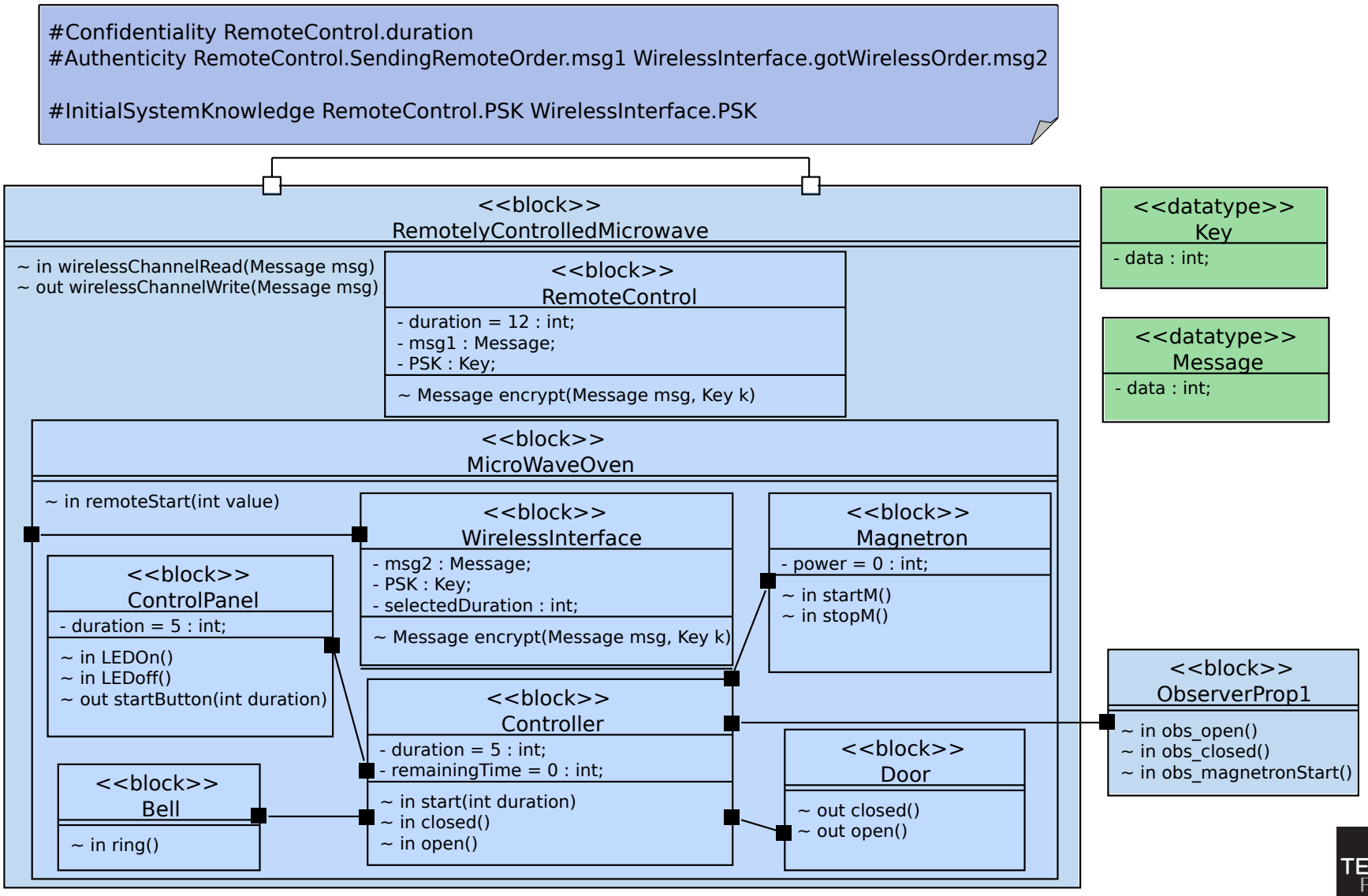
Overview of Contributions

1. AVATAR design model transformation to Petri Net
2. Computation of incidence matrix
3. Computation of minimal P-invariants
   - Farkas algorithm
4. Backtracing to the AVATAR design model

Case study: a microwave oven

Outline

AVATAR Design of a Microwave Oven



Overview of Model Transformation

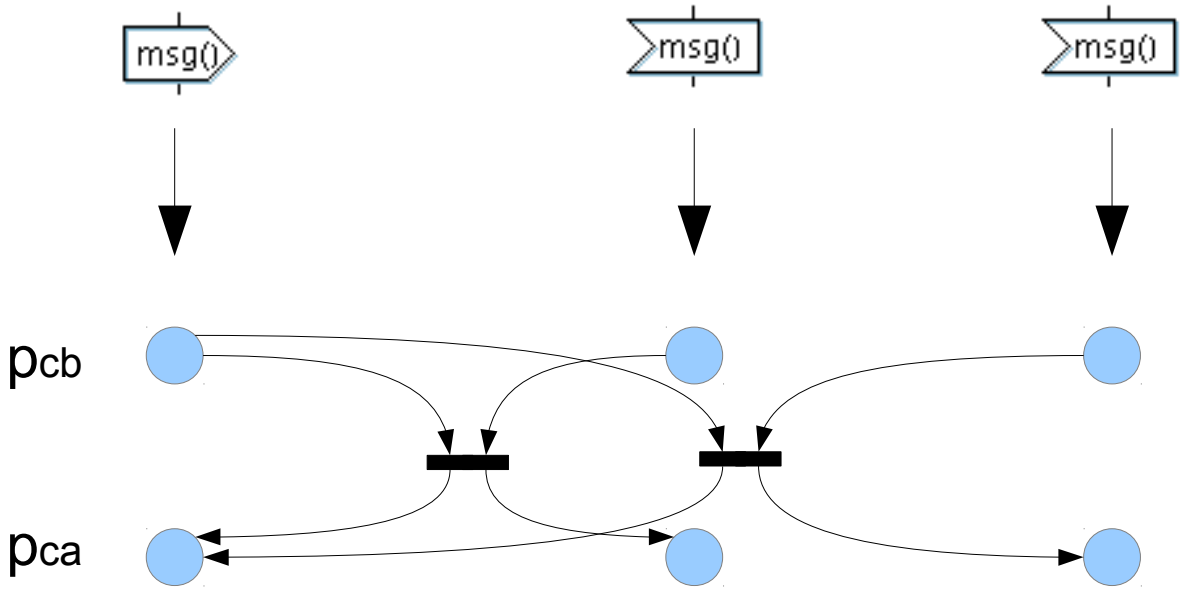
Translation of state machines

- State → PN place, Transition → PN transition
- Non-deterministic choice → PN place from which starts a set of non deterministic PN transitions
- Communication operators → one translation pattern is used for each communication semantics
  - Synchronous, blocking asynchronous, non blocking asynchronous
- Timers, guards and time constraints are ignored

Translation of the architecture

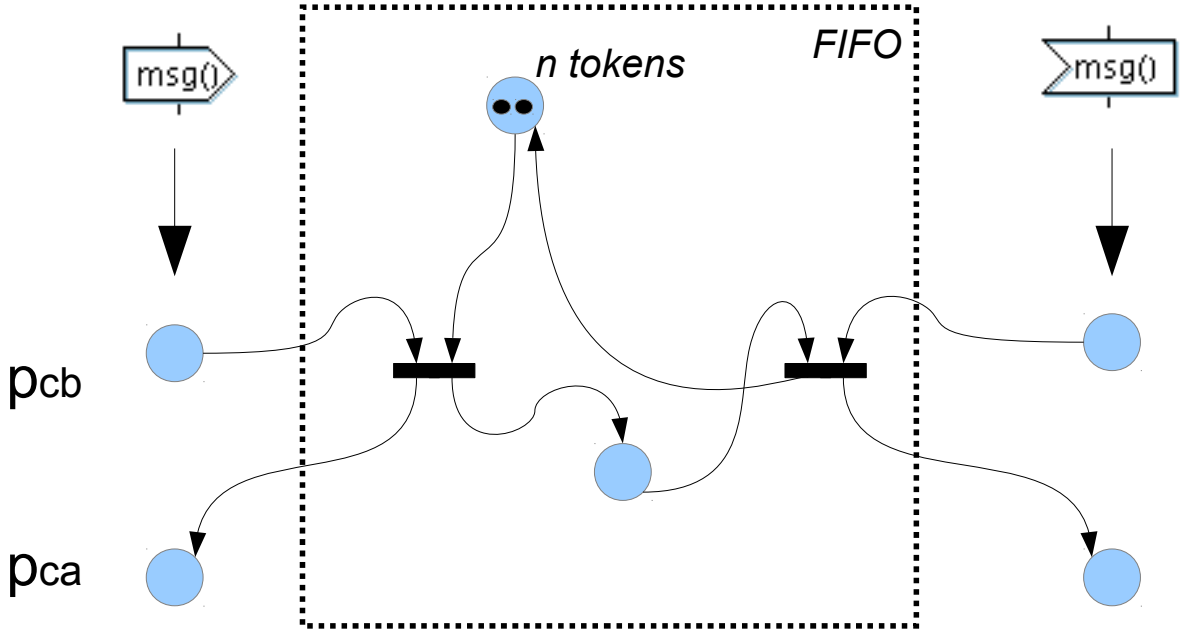
- Variables and methods are ignored
- Communication channels and signals are taken into account by the translation of communication operators

Synchronous Communications

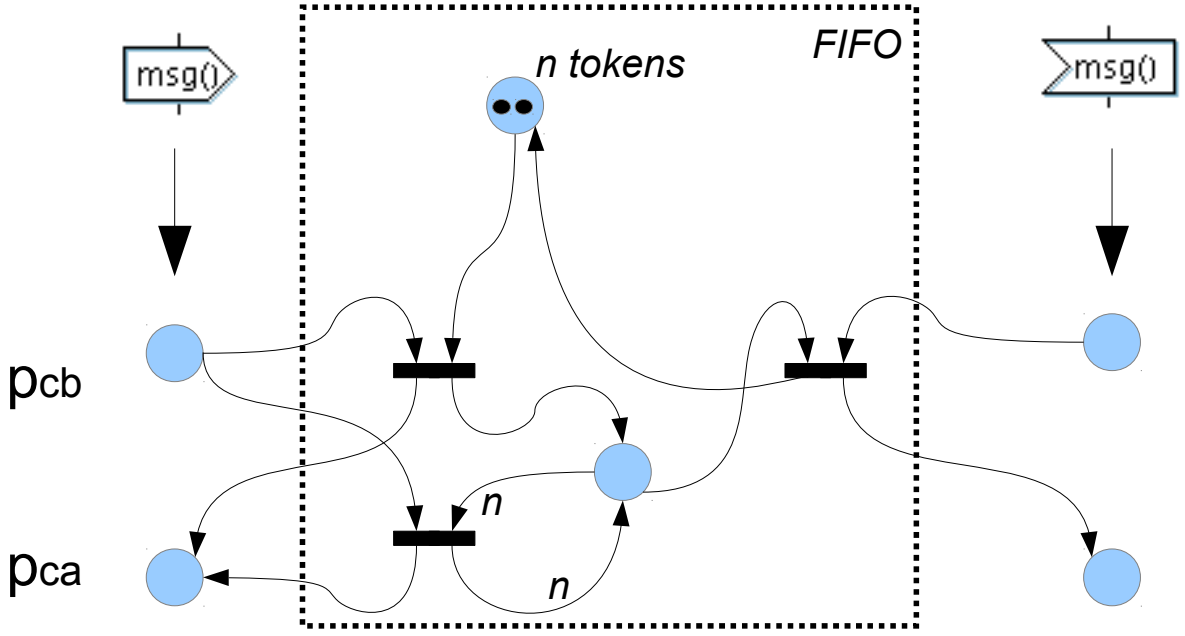


(p_{cb} means "place channel before" and p_{ca} means "place channel after")

Blocking Asynchronous Communications

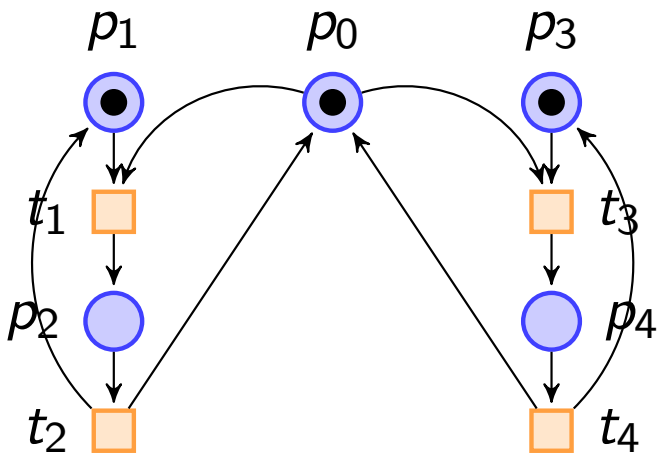


Non Blocking Asynchronous Communications



Petri Net: Example

- Petri Net



- Transposed incidence matrix

$$A^t = \begin{array}{c} \\ p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} -1 & 1 & -1 & 1 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array}$$

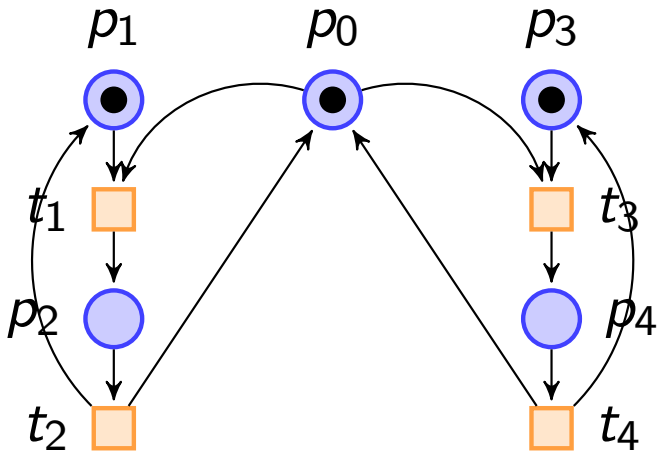
Computation of P-invariants

General approach

- The matrix A^t is made triangular, as for solving a linear system e.g. $\mathcal{W}.A = 0$
  - Lines of A^t can be exchanged, multiplied by a given integer value, or one line can be added to another one
  - Use of the *Farkas* algorithm

$$A^t_{triangular} = \begin{matrix} p_0 \\ p_4 \\ p_0 + p_2 + p_4 \\ p_3 + p_4 \\ p_1 + p_2 \end{matrix} \begin{matrix} t_1 & t_2 & t_3 & t_4 \end{matrix} \begin{pmatrix} -1 & 1 & -1 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

P-invariants of the example PN



Finally, P-invariants are:

- $p_1 + p_2$
- $p_3 + p_4$
- $p_0 + p_2 + p_4$

We are interested in P-invariants

- Whose places do not relate to only one AVATAR block
  - This is an obvious P-invariant since one block has only one flow of execution
- Whose value $= 1 \rightarrow$ all listed places are in mutual exclusion

Complexity

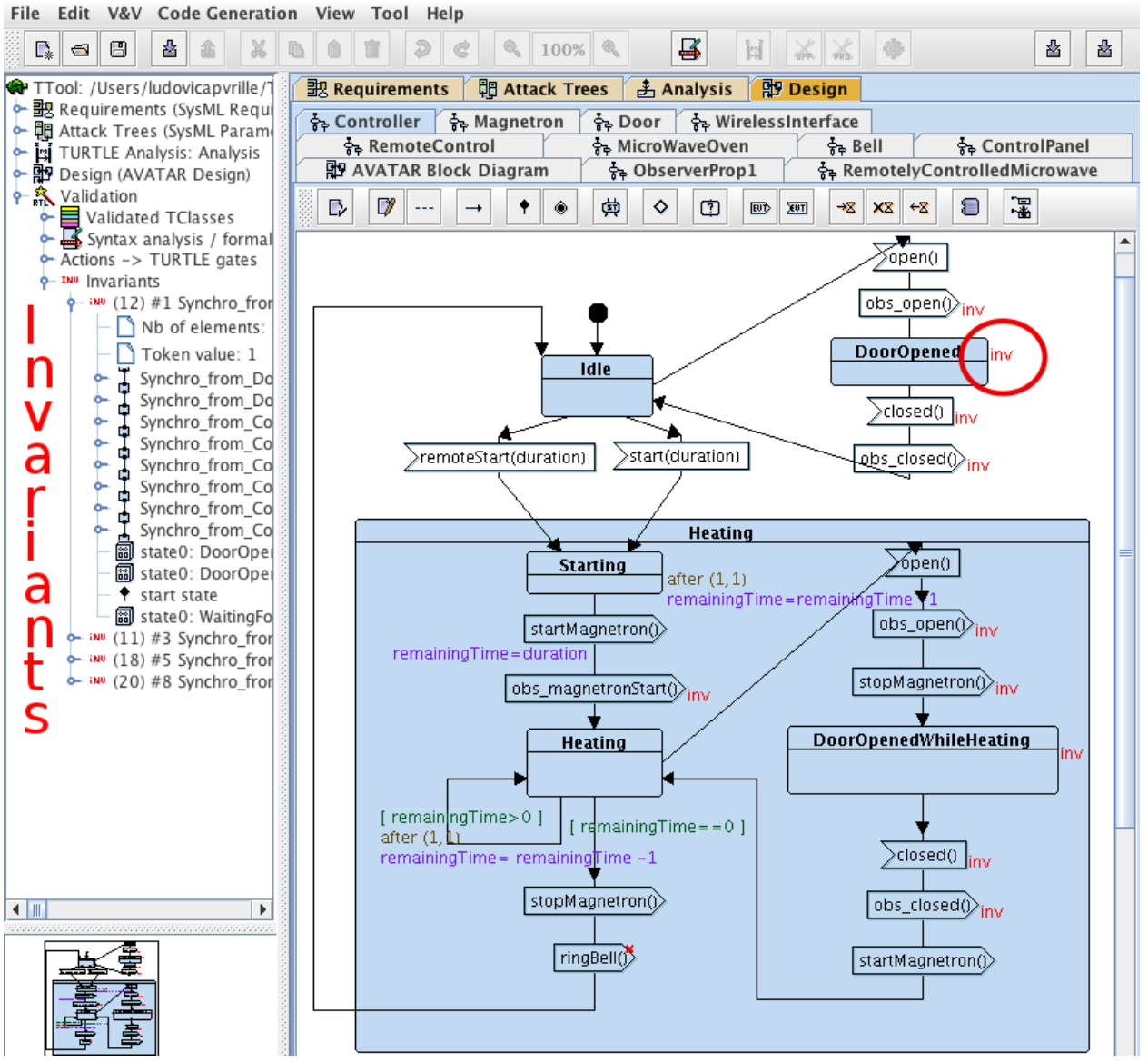
- Farkas: exponential time in the number of places of the net

$\rightarrow$ we have defined heuristics to reduce that complexity

Heuristics

- Memo: We are interested in mutual exclusions i.e. in P-invariants whose value is 1
- When combining lines, each time the line contains two times the same place, we delete the line
  - Unfortunately: we may remove invariants for which all places are listed the same number n of times, with $n > 1$
- Computations of invariants is instantaneous on all AVATAR models on which we have tested these heuristics
  - Without heuristics: up to an hour
  - No difference in the list of computed P-invariants

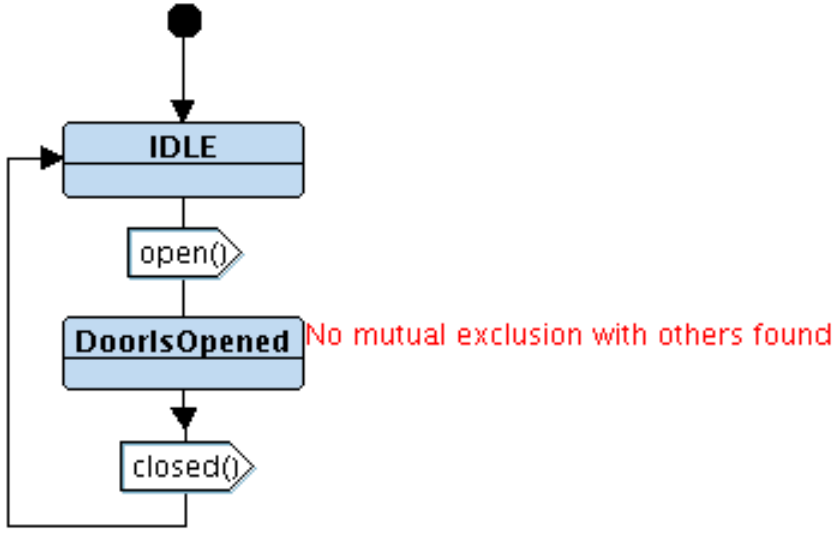
Invariants as displayed by TTool



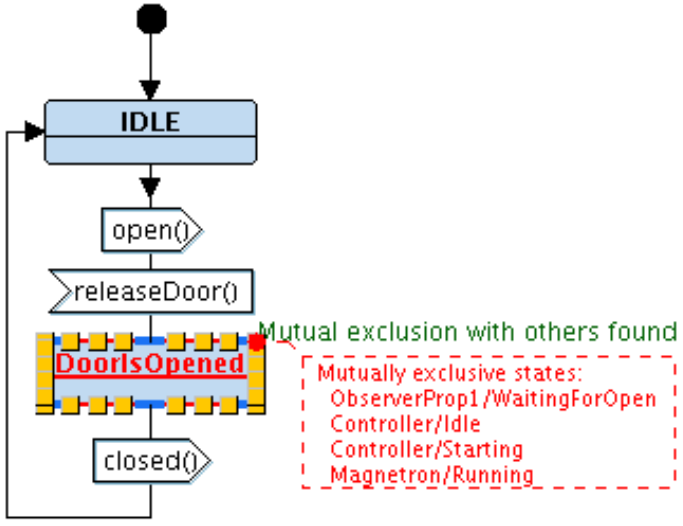
Mutual Exclusion of States

- **Example**: search of the mutual exclusion of the state Door/DoorOpened with the state Magnetron/Running

Version 1



Version 2



Outline

Conclusion

Contributions

- Static analysis of AVATAR models searching for mutually exclusive states
- TTool hides Petri nets and displays results on AVATAR models

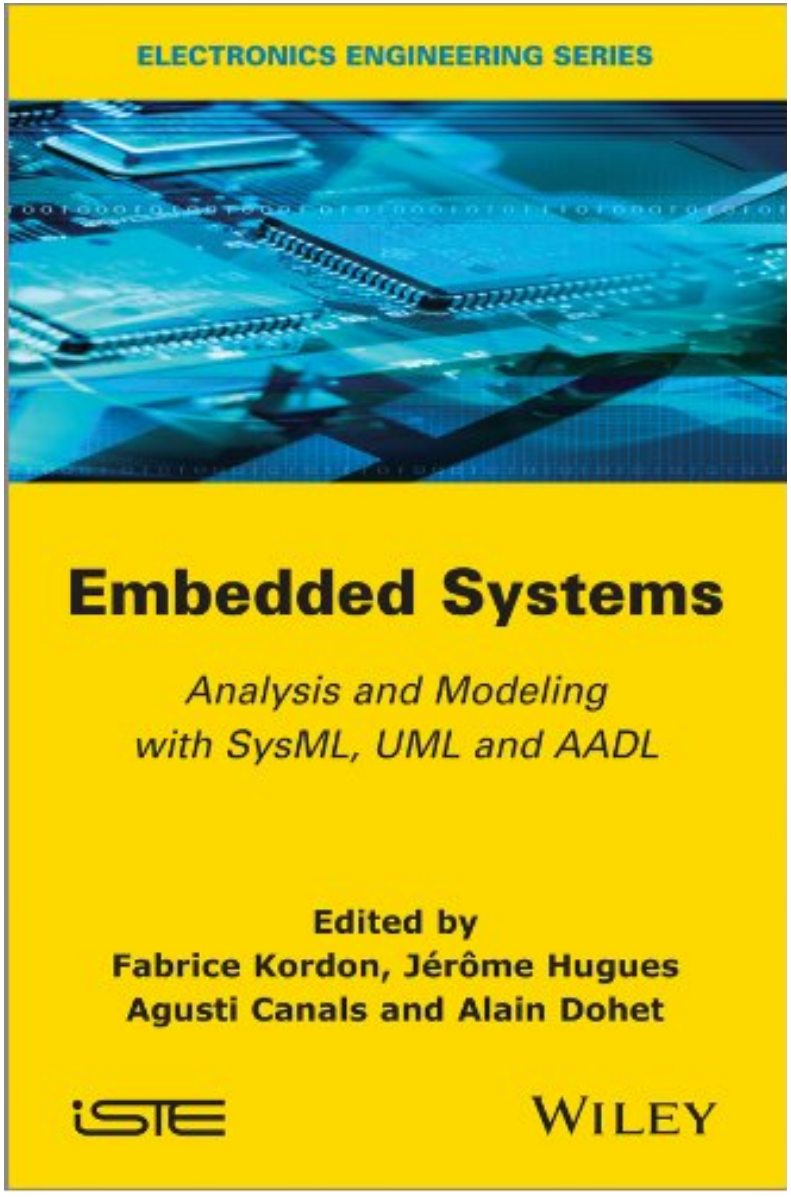
Limitations

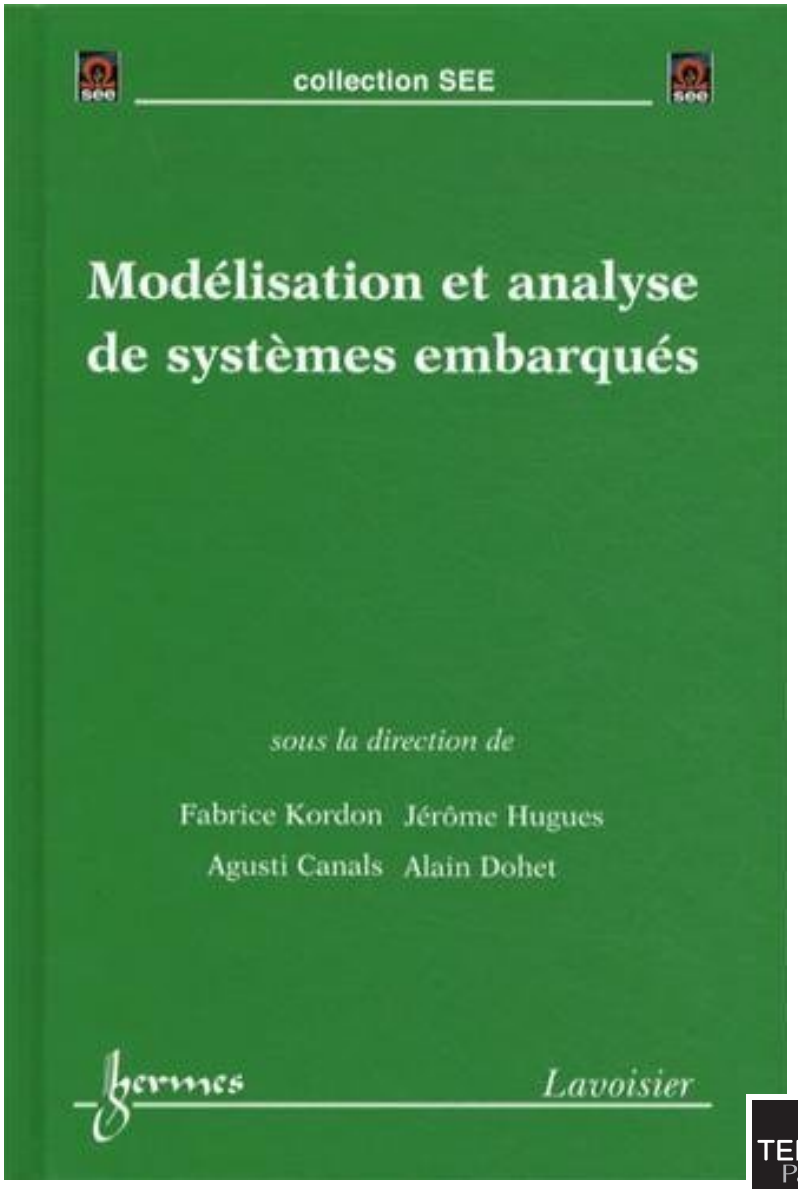
- Basic Petri nets
- No support of H and H* operators
- No support of continuous flows

Future work

- Invariants for extended Petri nets
- Application to other real-time UML profiles

To Go Further . . .





To Go Further: TTool, AVATAR

TTool

- http://ttool.telecom-paristech.fr
- Can be executed on usual operating systems (Windows, Linux, MacOS)
- Supports several profiles (e.g., DIPLODOCUS, AVATAR)
- Open-source, contributions are welcome
- Support from industrial and academic partners

AVATAR

- http://ttool.telecom-paristech.fr/avatar.html
- Tutorials, examples