IF: a validation environment for asynchronous real-time systems

L. MOUNIER

Distributed System Group

VERIMAG
Grenoble

The Context

Application area:

Telecommunication and distributed systems

Main characteristics:

- asynchronous communications
- "real-time" features
- **critical** systems

⇒ introduction of **validation techniques** into the development cycle.

Formal validation: the current situation

Specification formalisms

- use of international standards: Estelle, Lotos, SDL, UML, ...
- a difficult trade-of between:
  - programming facilities (e.g., high-level primitives)
  - validation facilities (e.g., real-time semantics)

Validation tools

- commercial environments:
  - edition, simulation, code generation, test case generation
  - support the existing standards
- academic tools:
  - efficient verification techniques
  - restricted input language

Motivations

IF: an intermediate representation
for timed asynchronous systems

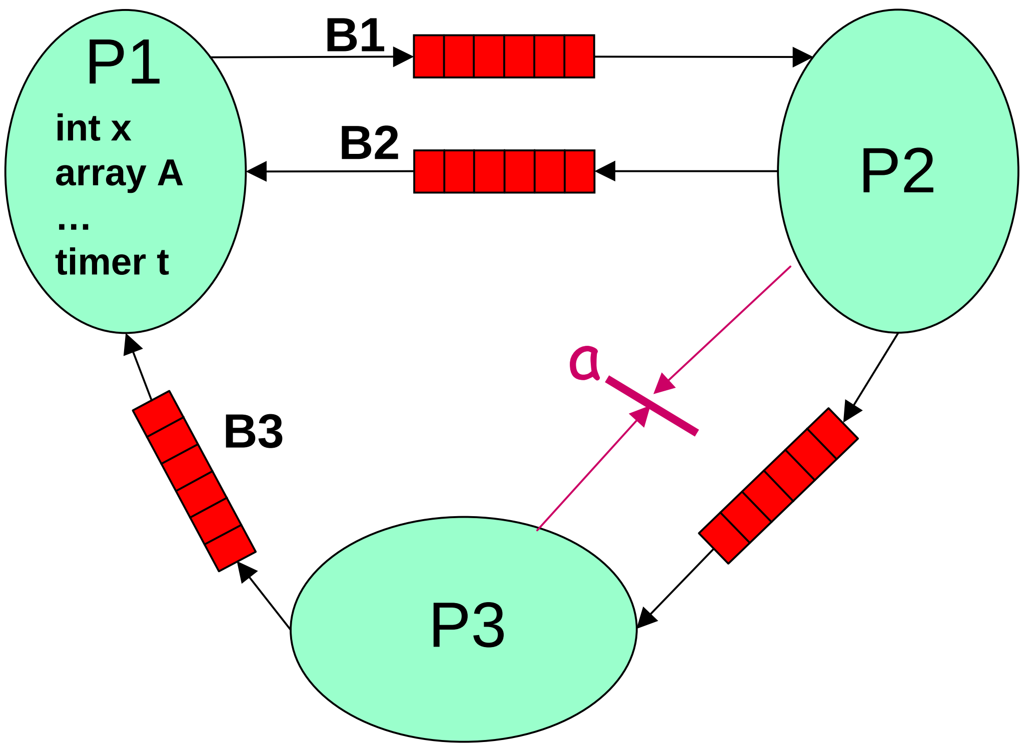
- a **connection** between commercial and academic tools
  $\Rightarrow$ bridge the gap between standarts and low-level formalisms
- a **“source level”** intermediate representation
  $\Rightarrow$ allows efficient optimisation and verification techniques
- relies on a powerful and flexible **time model**
  $\Rightarrow$ a laboratory to study the real-time semantics of high-level formalisms

Outline

- Motivations
- **IF: the language**
- The IF validation environment
- Some case studies
- Conclusion and perspectives

The IF intermediate representation

Communicating extended timed automata (with urgencies)



Communication

-asynchronous message buffers
(reliable/lossy/bounded)
-synchronous rendez-vous
-shared variables

Time model

Timed Automata with
urgency attributes on transitions

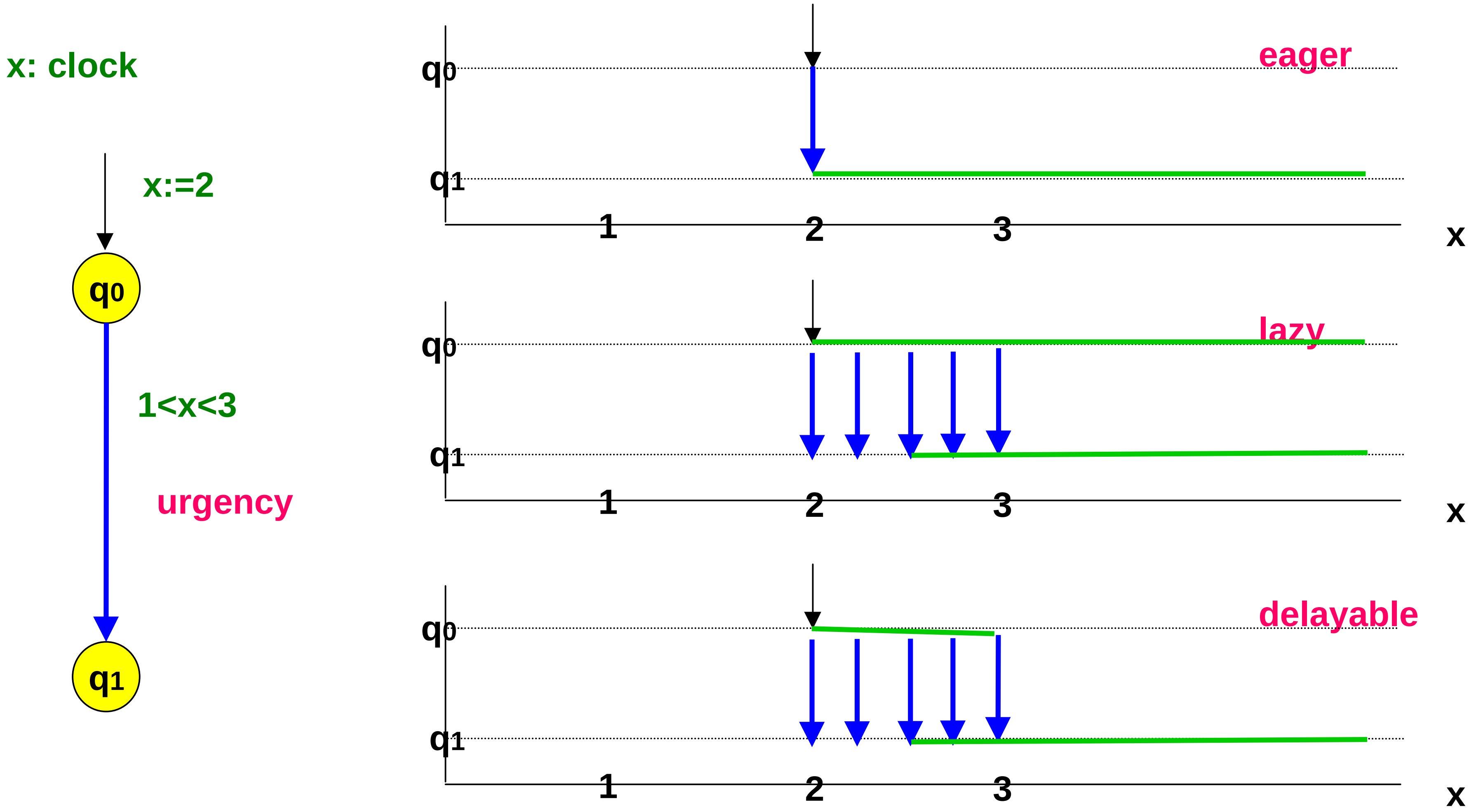
Timed automata with urgency [BornotSifakis96]

Time progress depends on **urgency** of enabled transitions:

- **eager** transitions are urgent as soon as they are enabled and block time progress
- **lazy** transitions never block time progress
- **delayable** transitions allow time progress unless time progress disables it

$\Rightarrow$ allows to express a large spectrum of real-time paradigms.

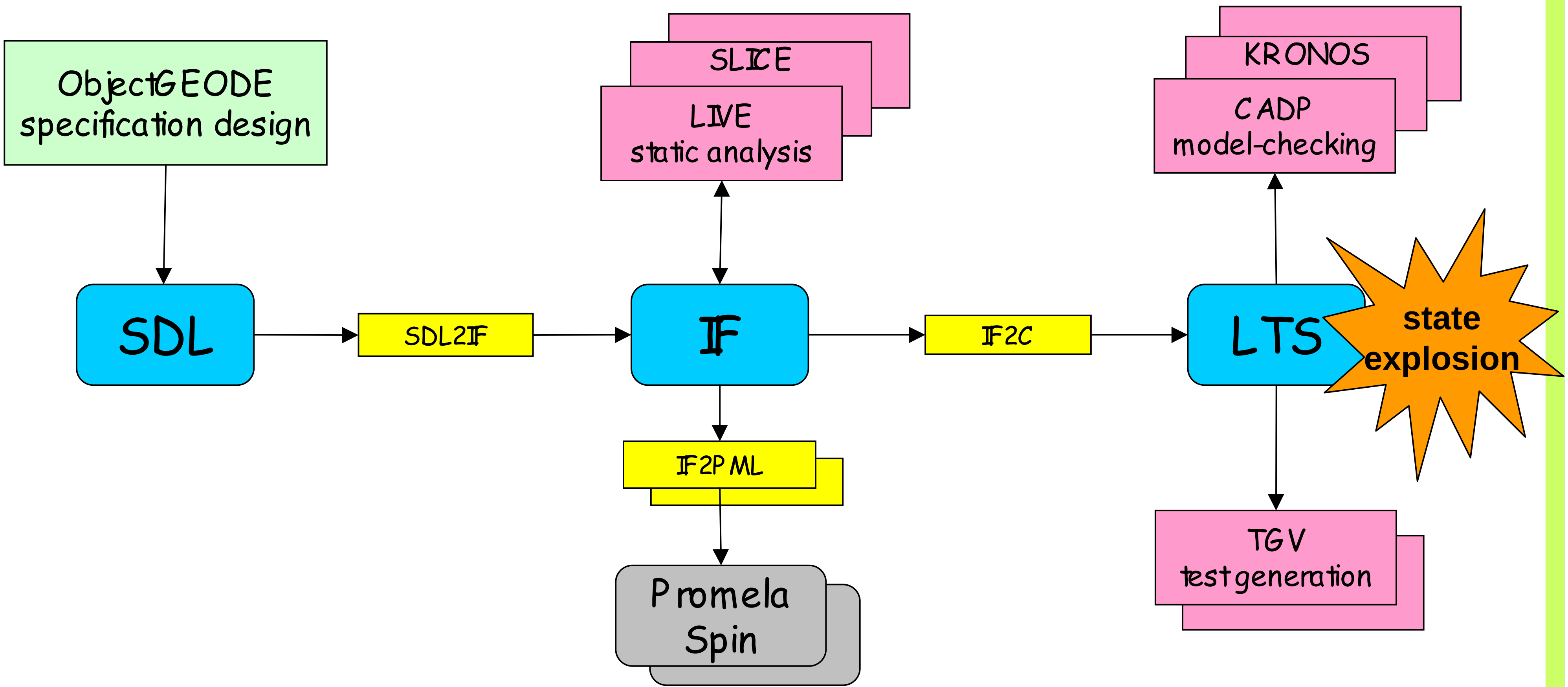
Transition Urgency



Outline

- Context and Motivations
- IF: the language
- **The IF validation environment**
- Some case studies
- Conclusions and perspectives

Architecture of the toolbox



The frontend component: sdl2if

Translation from SDL to IF:

- based on an ObjectGeode API:

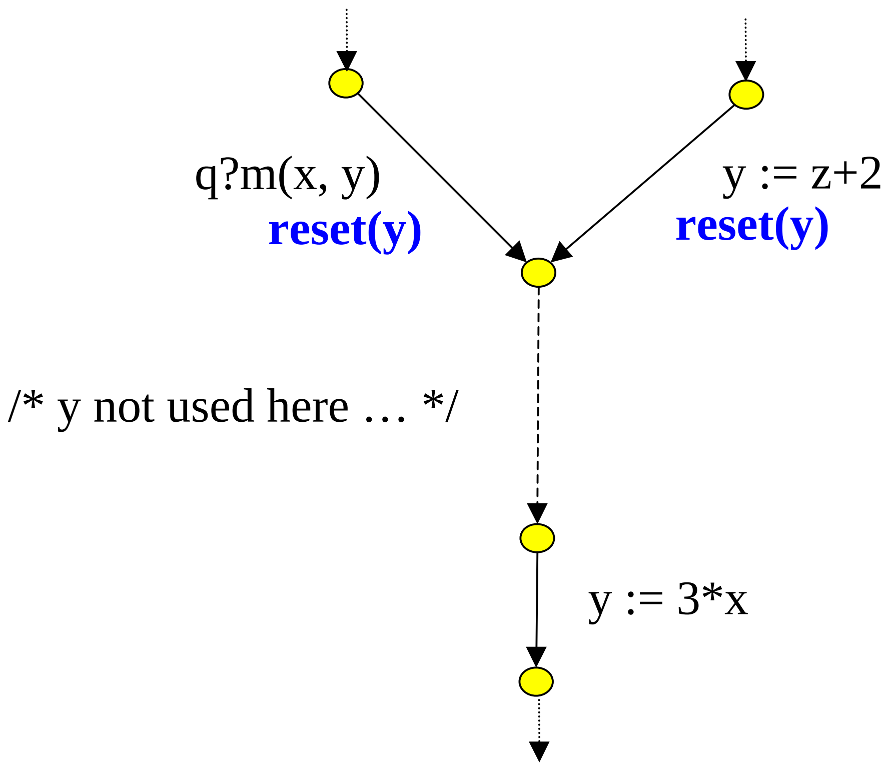
  ⇒ we follow standard evolution of SDL

- supports a static subset of SDL:
  - limited dynamic process creation/destruction
  - procedures are inlined (no recursion)
  - only static data types are fully translated

The IF level components

- **Translation** from IF to other tools and formalisms:
  - Promela-Spin (University of Eindhoven)
  - Lash (University. of Liege)
  - Agatha (CEA-Leti)
  - …
- **Static analysis** and **abstractions**:
  - live variable computation
  - slicing

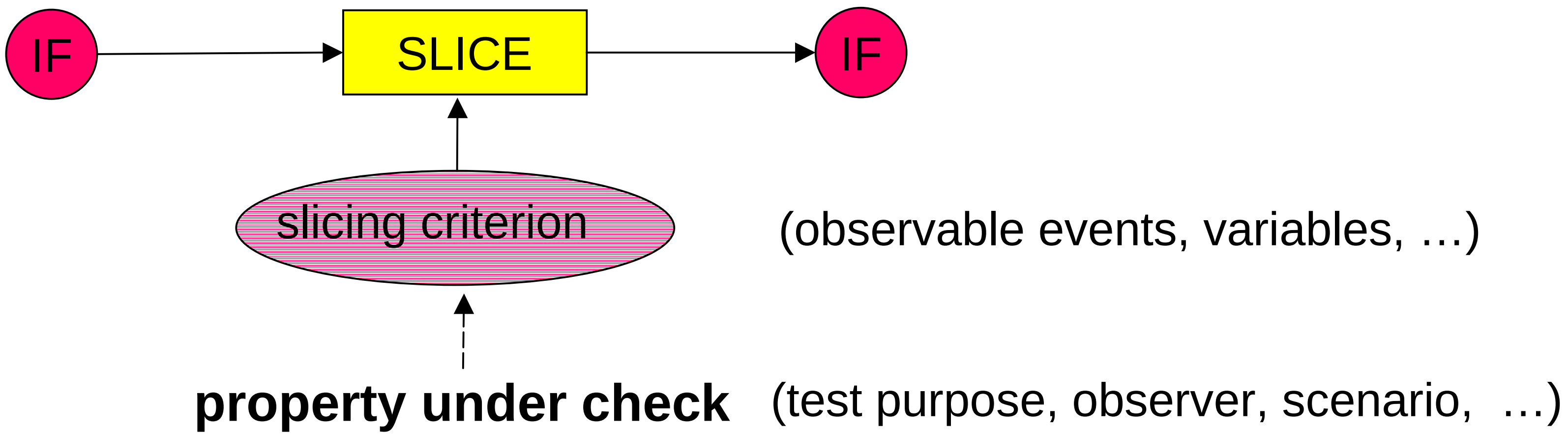
Static analysis (1): live variables computation



- strongly preserve the initial behaviour
- drastically reduce the size of the model (more than 2 orders of magnitude)
- easy to compute ...

Static Analysis (2): slicing

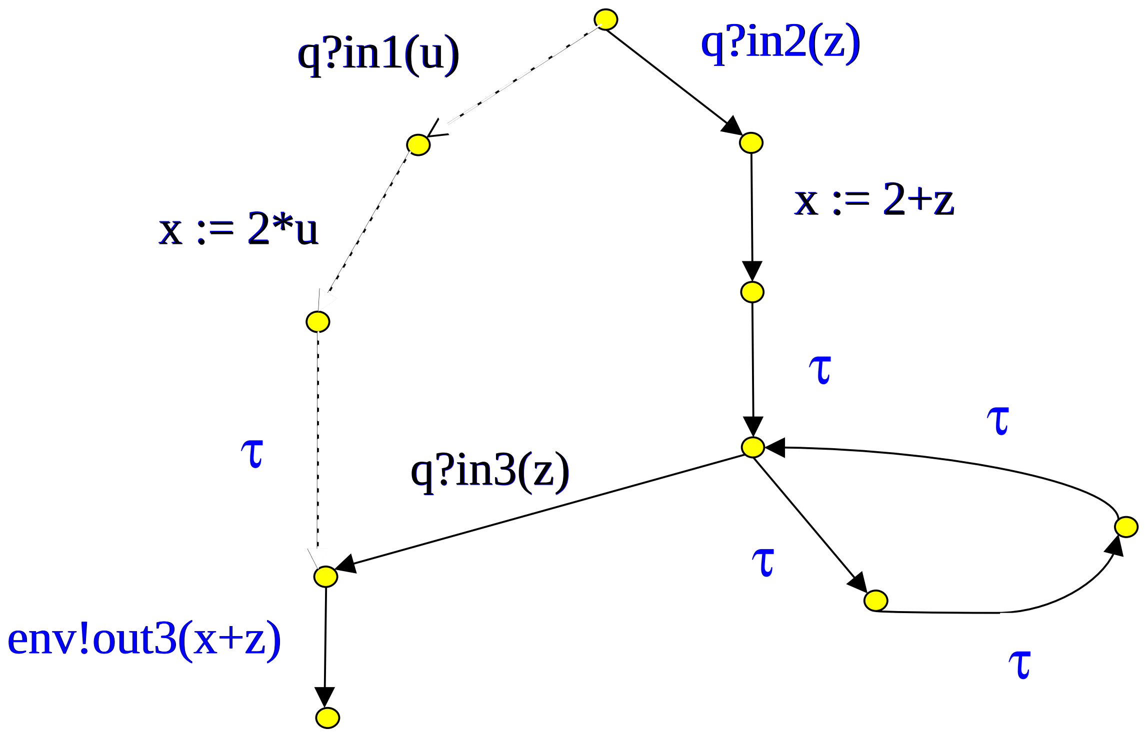
- Extract the relevant part of a specification with respect to a **slicing criterion**:



- Validation can be performed on the simplified specification

Slicing (example)

var: u,w,x,y,z



Slicing criteria:

- observable events: in2, out3

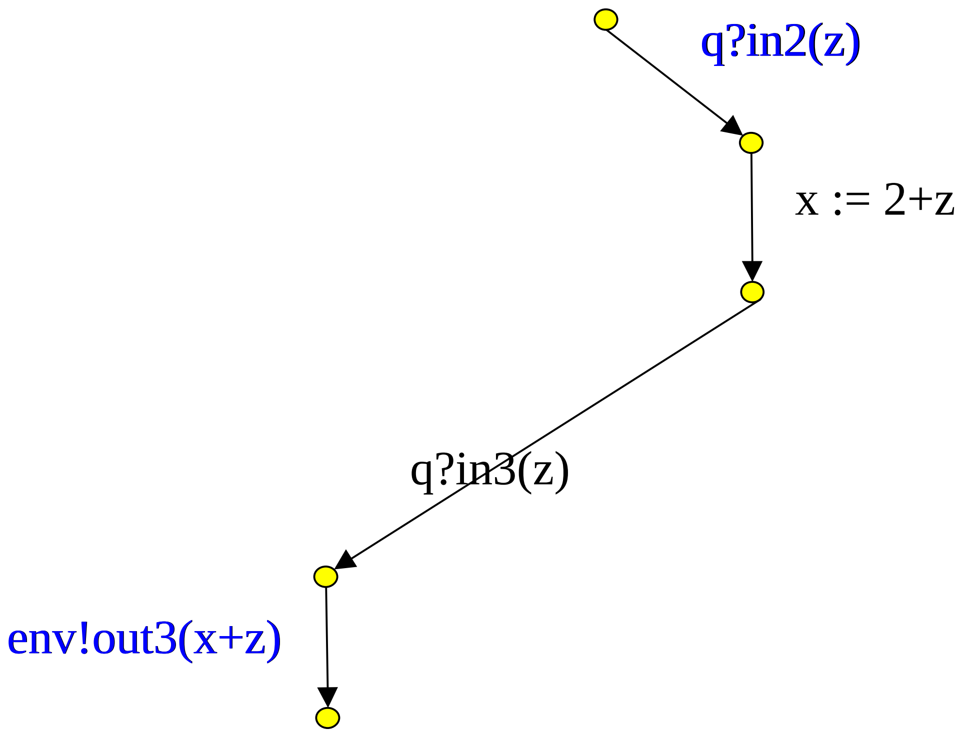
var: u,x,z

- environment: in2, in3, in4

var: x,z

Slicing (example)

var: x,z



Slicing criteria:

- observable events: in2, out3

var: u,x,z

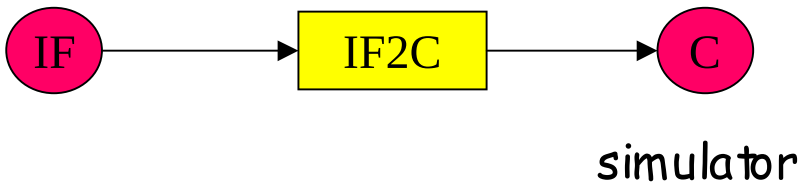
- environment: in2, in3, in4

var: x,z

- weak bisimulation reduction

The LTS level components

- simulator construction:



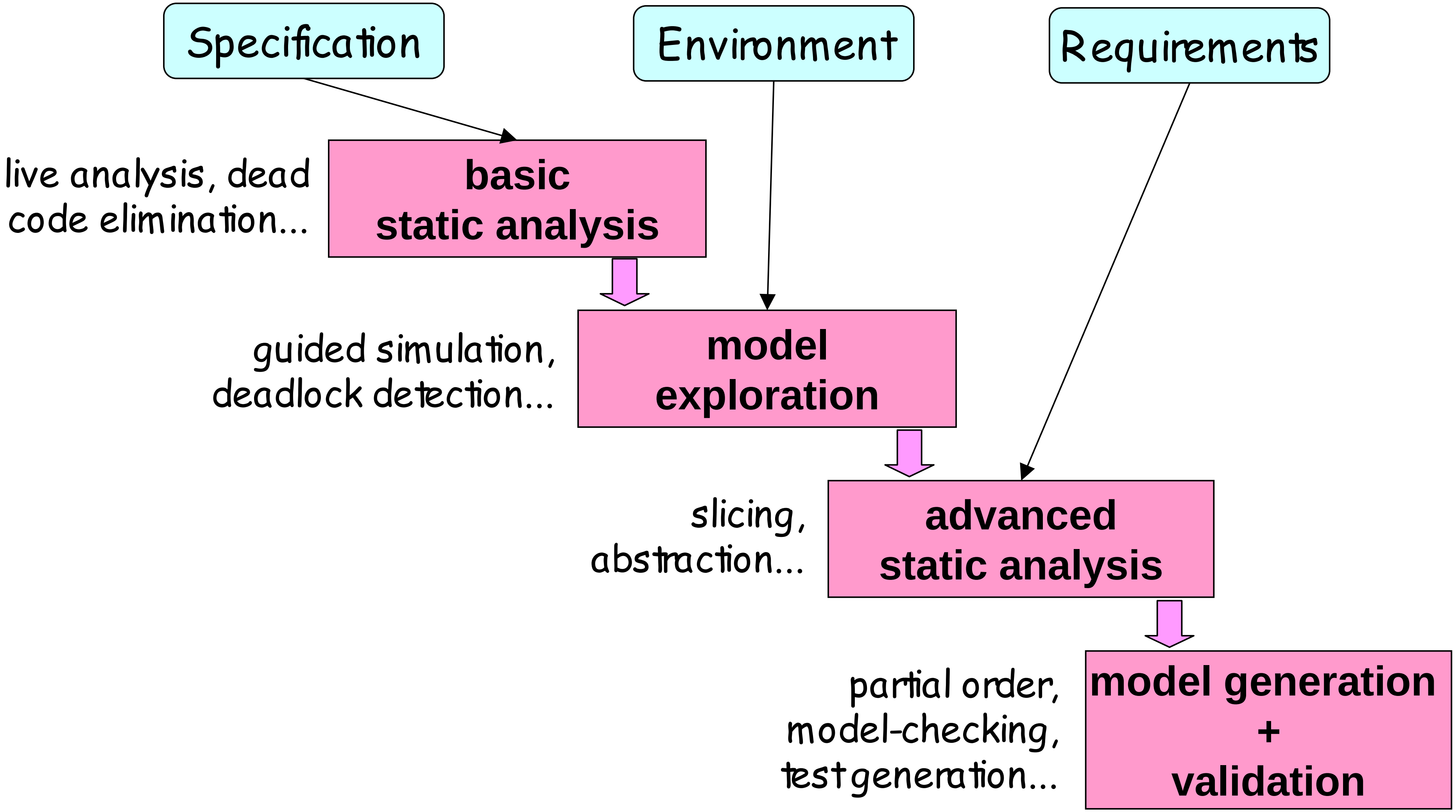
- implements discrete/dense time
- supports **on-the-fly** and **partial order** reductions techniques

- model-checking:
  - temporal-logic properties (Evaluator, Kronos)
  - behavioural specifications (Aldébaran)

  $\Rightarrow$ both including diagnostic capabilities

- test case generation (TGV)

A validation "methodology"



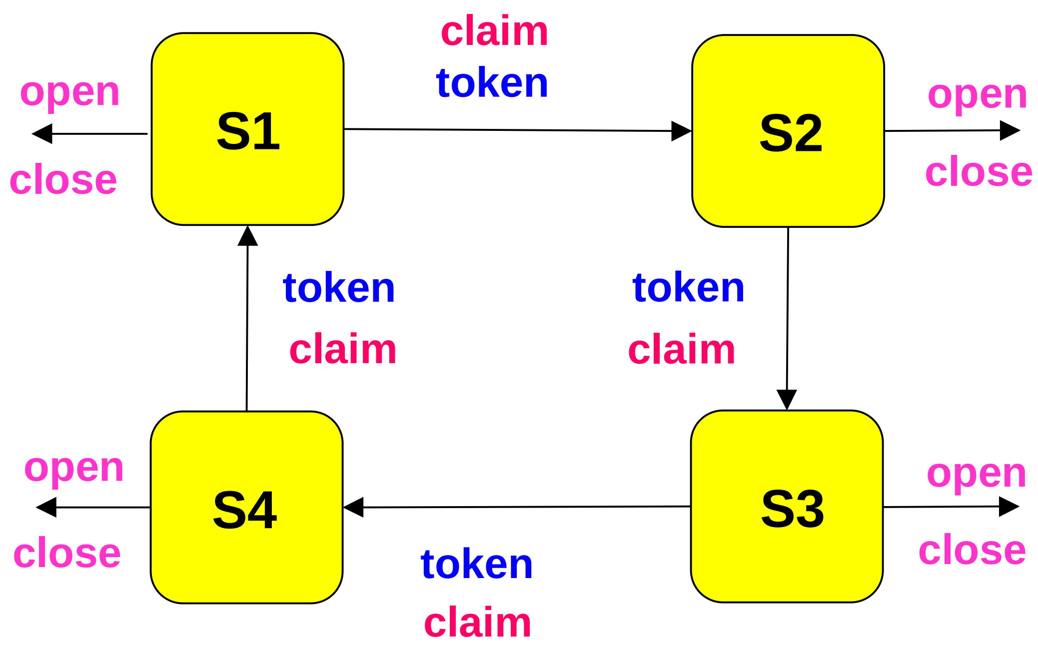
Outline

- Motivations
- IF: the language
- The IF validation environment
- **Some case studies**
  - a distributed leader election algorithm
  - the SSCOP protocol
  - the Ariane-5 flight controller [slides not available here]
- Conclusions and perspectives

A distributed leader election algorithm

mutual exclusion access to a shared resource
on an unreliable circular network:

- approx. 200 lines of SDL
- 4 processes:
  - 3 states
  - 3 variables, 1 timer

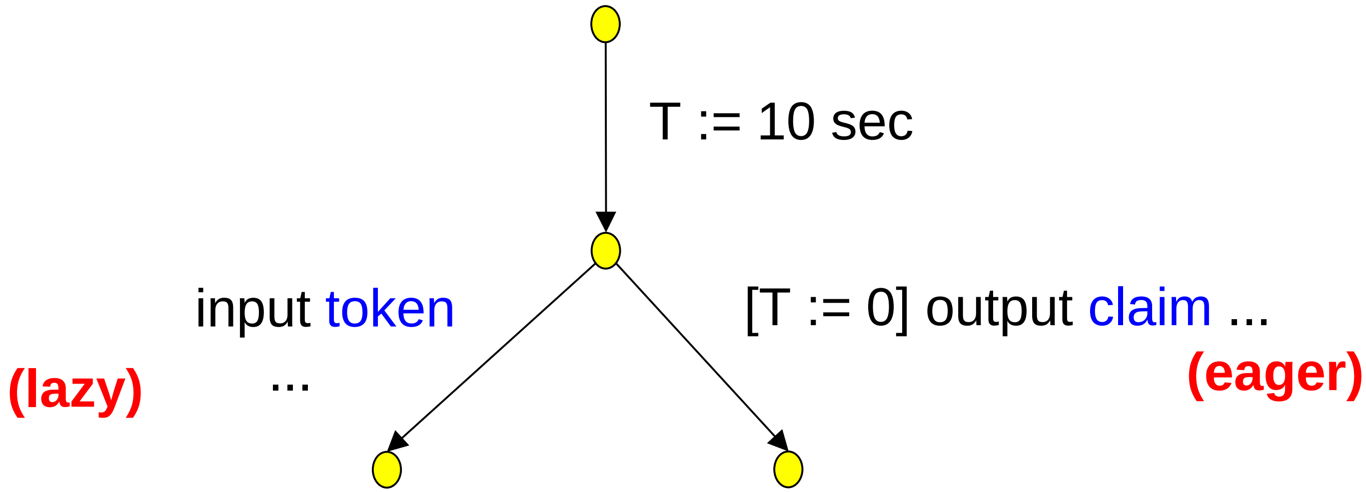


Leader election algorithm: modelling problems

1. modelling unreliable channels: an IF buffer attribute

2. modelling time progress:

timer T



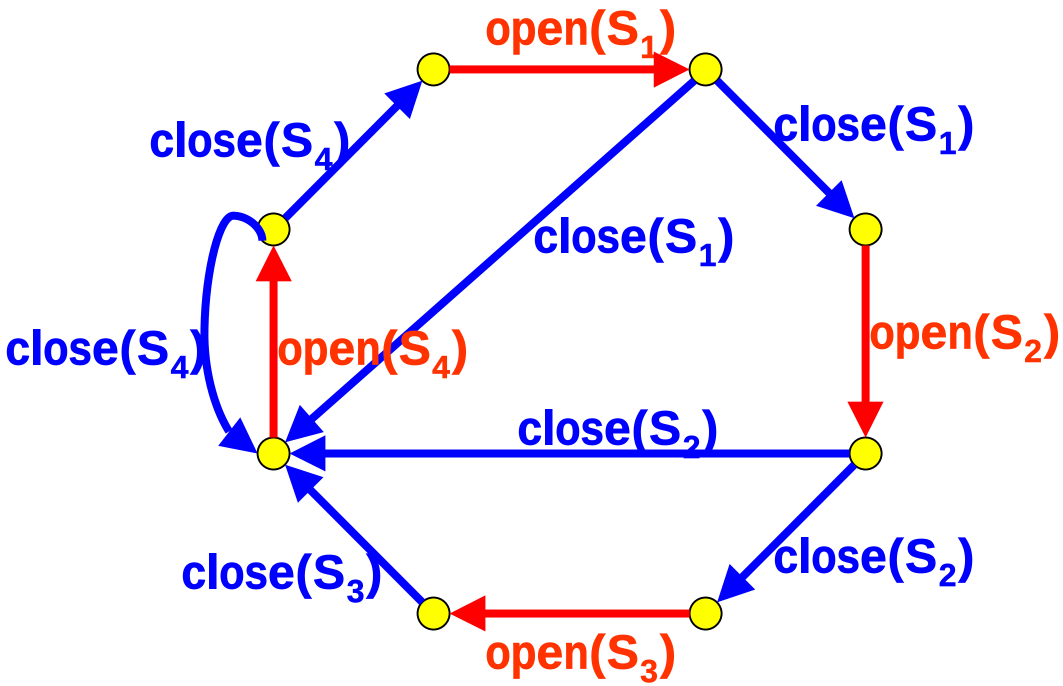
Results for live analysis

	ObjectGEODE	IF	IF + live analysis
reliable channels maximal urgency	1731 st. 3822 tr. 1 sec.	618 st. 1256 tr. 0.4 sec.	292 st. 756 tr. 0.2 sec.
lossy channels maximal urgency	3 018 145 st. 7 119 043 tr. 18 mn 7 sec.	537 891 st. 2 298 348 tr. 9 mn 7 sec.	4943 st. 19664 tr. 4.8 sec.
lossy channels weak urgency	not available	too large !	54591 st. 250 016 tr. 54.9 sec.

Leader election algorithm: verification

Property:

> Accesses to the resource are performed **in mutual exclusion**, i.e., there is always a **close** action between two **open** actions



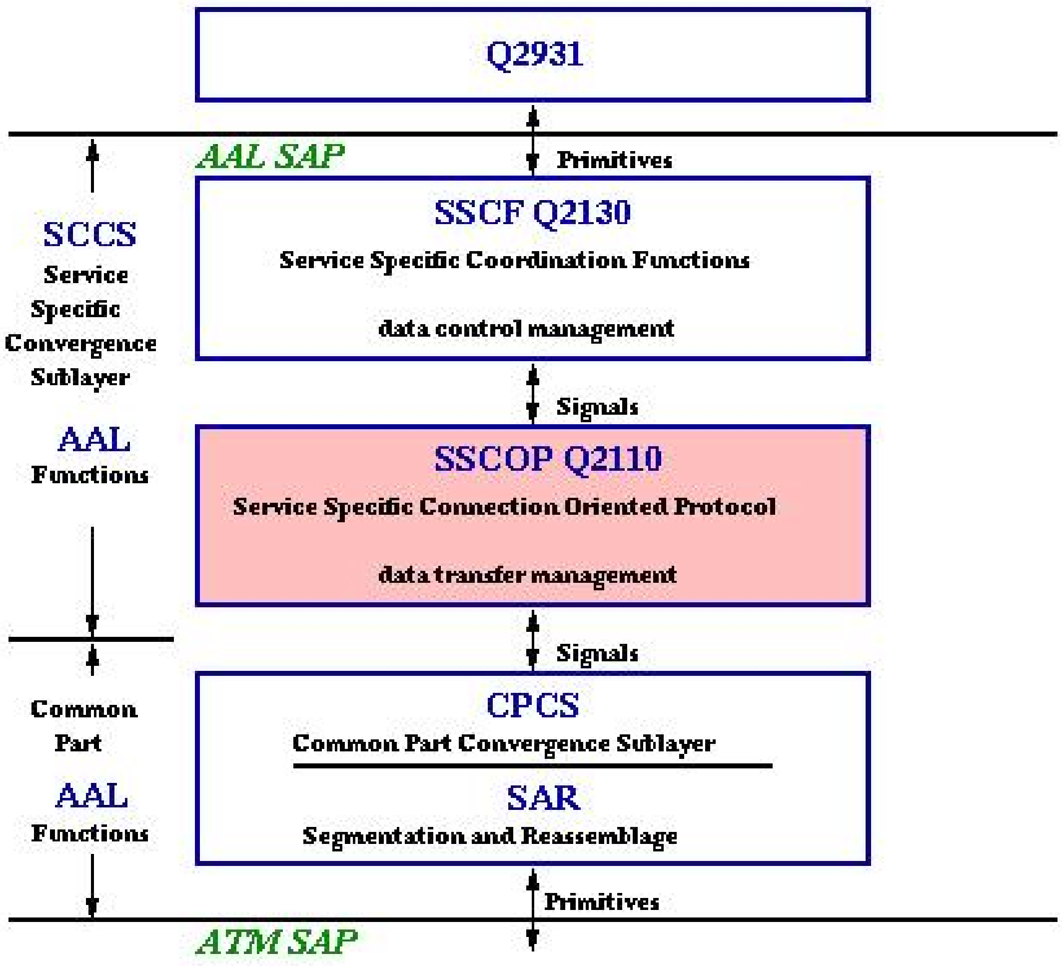
Graph obtained by
weak bisimulation minimisation

Outline

- Motivations
- IF: the language
- The IF validation environment
- Some case studies
  - a distributed leader election algorithm
  - **the SSCOP protocol**
  - the Ariane-5 flight controller [slides not available here]
- Conclusions and perspectives

SSCOP Protocol

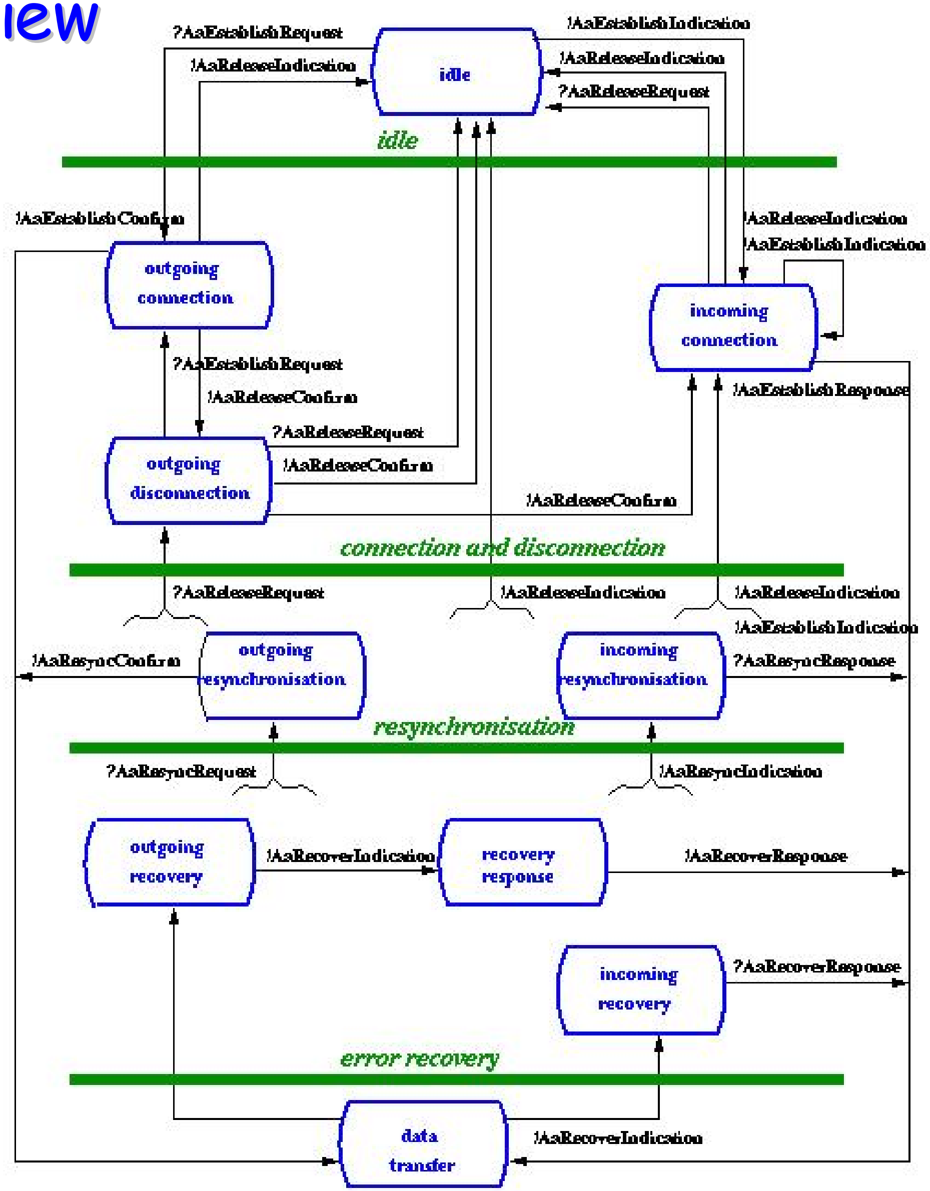
- case-study provided by France Telecom R&D within the FORMA Research Action
- part of ATM Adaptation Layer (AAL), normalised by ITU Q2110
- aims were both formal verification and test generation



SSCOP Protocol: overview

- several services have to be provided
  - connection control (establishment, flow-control, maintenance)
  - data transfer
  - error detection and recovery

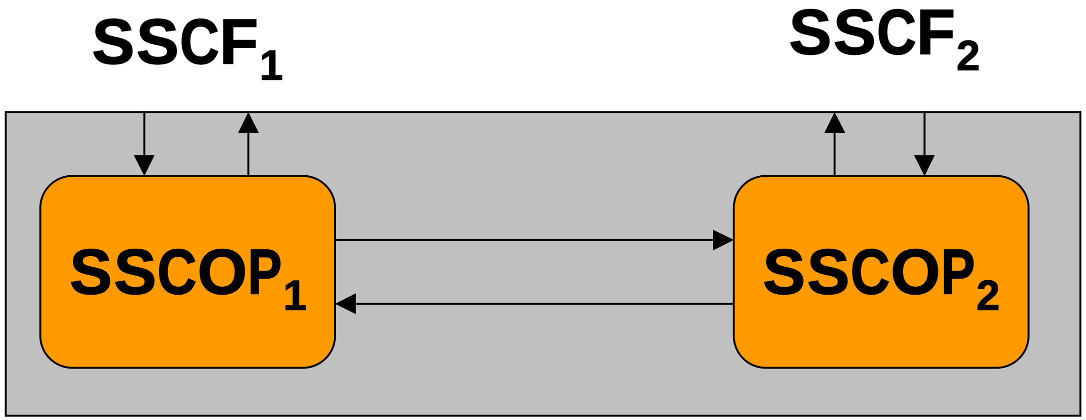
- described as a single SDL process
  - 10 states, 134 variables, 4 timers
  - 2000 lines of code



SSCOP Protocol: Verification Steps

- direct generation using ObjectGEODE fails
  - 2kB /state vector: only 50 000 states could be generated
- static analysis simplifications
  - "aggressive" abstraction by variable elimination
  - variable resetting using live information
  - slicing wrt specific properties
- model generation using ObjectGEODE
  - 0.2 kB /state vector: 1 000 000 could be generated
  - several functioning phases were completely verified

SSCOP Protocol: Verification Steps (cont'd)



Example of property: connection establishment

each connection request input to a SSCOP entity is followed by a connection response output by the same entity

Verification: about 15 000 states generated (2 mn)

Conclusion

IF Validation environment

- **open** validation environment, connecting design and verification tools (ObjectGeode, Spin, CADP, TGV, ...)

- provides automatic **program level** optimisations: static analysis, slicing

- able to deal with **realistic** size case studies ...

Perspectives

- more static analysis (invariant generation, ...)
- general abstractions (InVeSt)
- connection with other tools

- definition of **dynamicIF**, for the description of dynamic and parameterised systems and for **connection with UML and Java**

The END

http://www-verimag.imag.fr/DISTSYS