Specification, Model Generation, and Verification of Distributed Applications

E. Madelaine

Oasis team

INRIA -- CNRS / I3S -- Univ. of Nice Sophia-Antipolis

My Background :
process algebras and verification tools

PhD (1983): Correction proof of compilers**:** axiomatic semantics, rewriting techniques, early theorem proving techniques, LCF system

MEIJE team (1983-99): process algebras, structural operational semantics (finiteness results, ECRINS and PAC tools);
model- and equivalence- checking engines (AUTO, MAUTO)
graphical formalisms, semantic formalisms (ATG, FC2)

Joining the OASIS team (2000)

Challenge :

“ Can we use existing formalisms and existing semantic models to lift verification methods from “academic" calculi (process algebras and their natural LTS-based behavioural semantics), to real languages, to support the analysis of Java/ProActive applications ? ”

Correlated question:

“ Can we provide analysis/verification methods and tools to the non-specialist developer ? ”

Agenda

- Related work: A Fast Moving Landscape
- Running example
- Behavioural semantics:
    - The pNets model
    - Semantics of GCM applications
- Tool platform:
    - Formalisms
    - Scaling up
- Conclusion and Perspectives

Landscape

(1) (Semantic) models of distributed applications

- automata-based (ASM, STS, …) + equational (LOTOS)
- dynamic calculi (Pi*, chemical machine)
- probabilistic, timed, synchronous. Mixed synch/asynch (GALS / PALS)

(2) (Programming) models for distributed and component-based systems

- CCA, CCM, SCA, …
- **Fractal**: encapsulation + interfaces, hierarchy, separation of concerns
- **GCM** (Grid Component Model): Fractal extension with asynchronous communication, transparent futures, collective interfaces

Landscape

(3) Model-checking engines, a lot of progress in the last 10 years:

- Improvements of classical engines : SPIN, SMV, UPAAL, …
- Progress in SAT-solvers (see SAT-Race yearly competitions):

  MiniSat, ManySat, Psolver, ...
- Satisfiability Modulo Theories (SMT), e.g. SAT + linear integer arithmetic + uninterpreted functions + satellite theories… (see SMT-COMP yearly competitions)

  Z3 (Microsoft), Yices (SRI), OpenSMT (U. Lugano), ...

Landscape

(4) Parallel State-space Generation

Explicit/Distributed: ~linear speedup; hash function, buffers, ...

Explicit/Shared: ~linear speedup up to 16 processors/cores; work stealing

Implicit/Distributed: difficult and not very efficient; vertical/horizontal partitioning of BDD/MDD trees, speculative computation, etc.

Implicit/Shared: difficulties due to the overload of locking mechanisms; uncertain experimental results.

[Invited Survey by Gianfranco Ciardo, U. Of California Riverside, PDMC’09]

Landscape

CADP (INRIA Grenoble) :

new boolean equation solvers,
new logics,
new compositional/contextual tools
distributed state-space representation, distributed engines

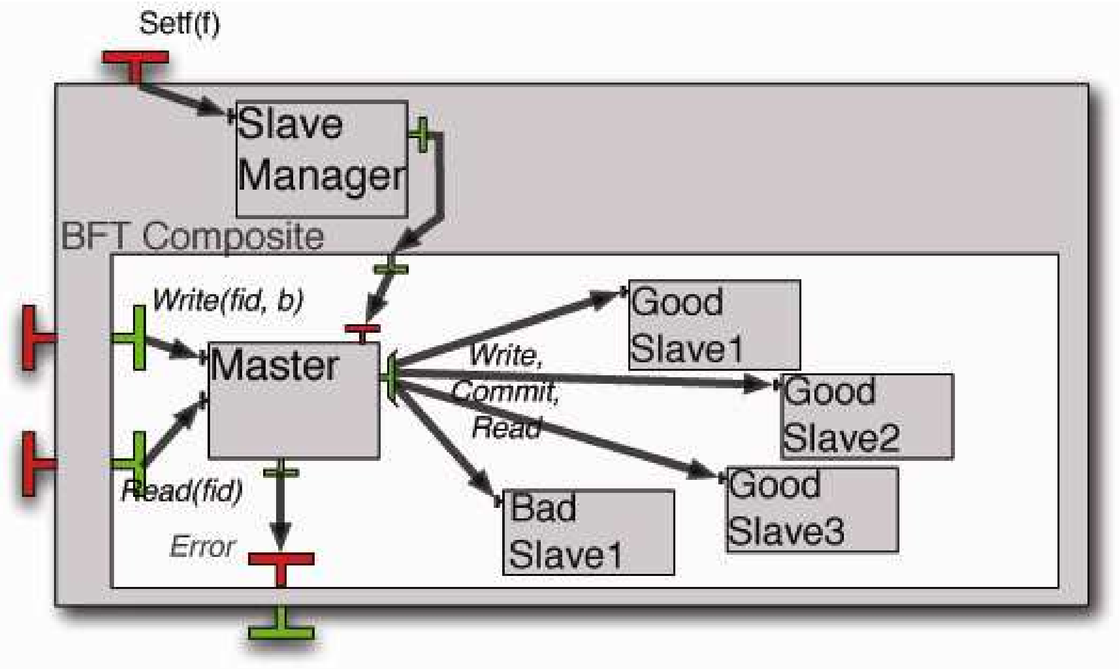
Agenda

- Related work: A Fast Moving Landscape
- **Running example**
- Behavioural semantics:
  - The pNets model
  - Semantics of GCM applications
- Tool platform:
  - Formalisms
  - Scaling up
- Conclusion and Perspectives

Running example: BFT system *[FACS '11]*

- **1 composite component** presenting 2 external services Read/Write.

- The service requests are delegated to a **Master component.**



- 1 multicast interface sending write/read/commit requests to all **Slave components**.
- The salves reply asynchronously, the master only needs 2f+1 coherent answers to terminate.

Agenda

- Related work: A Fast Moving Landscape
- Running example
- **Behavioural semantics:**
  - **The pNets model**
  - **Semantics of GCM applications**
- Tool platform:
  - Formalisms
  - Scaling up
- Conclusion and Perspectives

Semantic Formalism : the pNet model

Compromise:

- **Flexible**: accommodate a wide choice of communication / synchronization mechanisms
- **Opened** to convenient “abstractions” towards specific classes of decidable models (finite, regular, etc.)

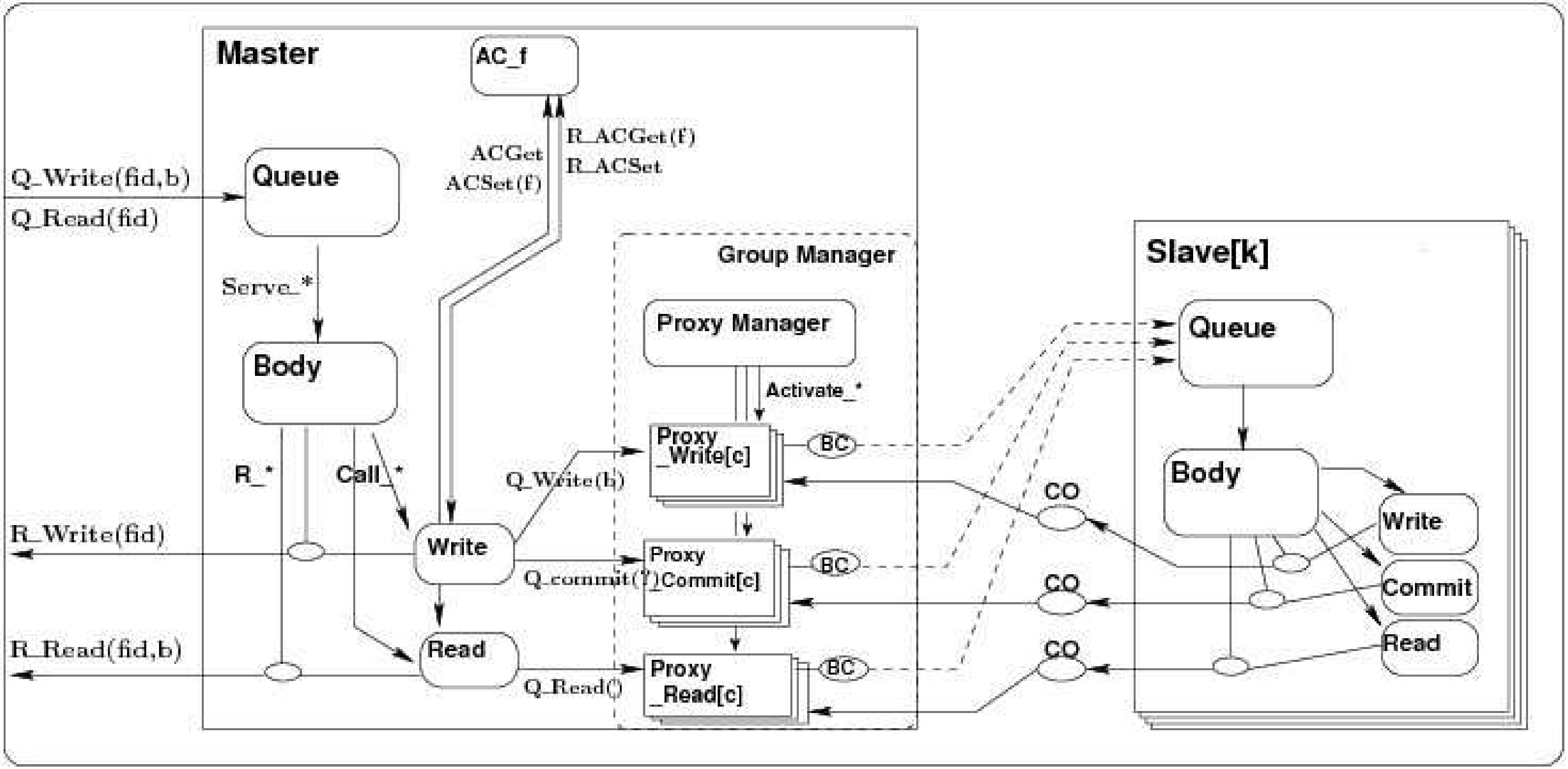
Solution: **[Forte’04, Annals of Telecoms 2008]**

- LTS with explicit data handling (value-passing) with 1st order types
- Parallelism and hierarchy using extended synchronization vectors, with parameterized topology.

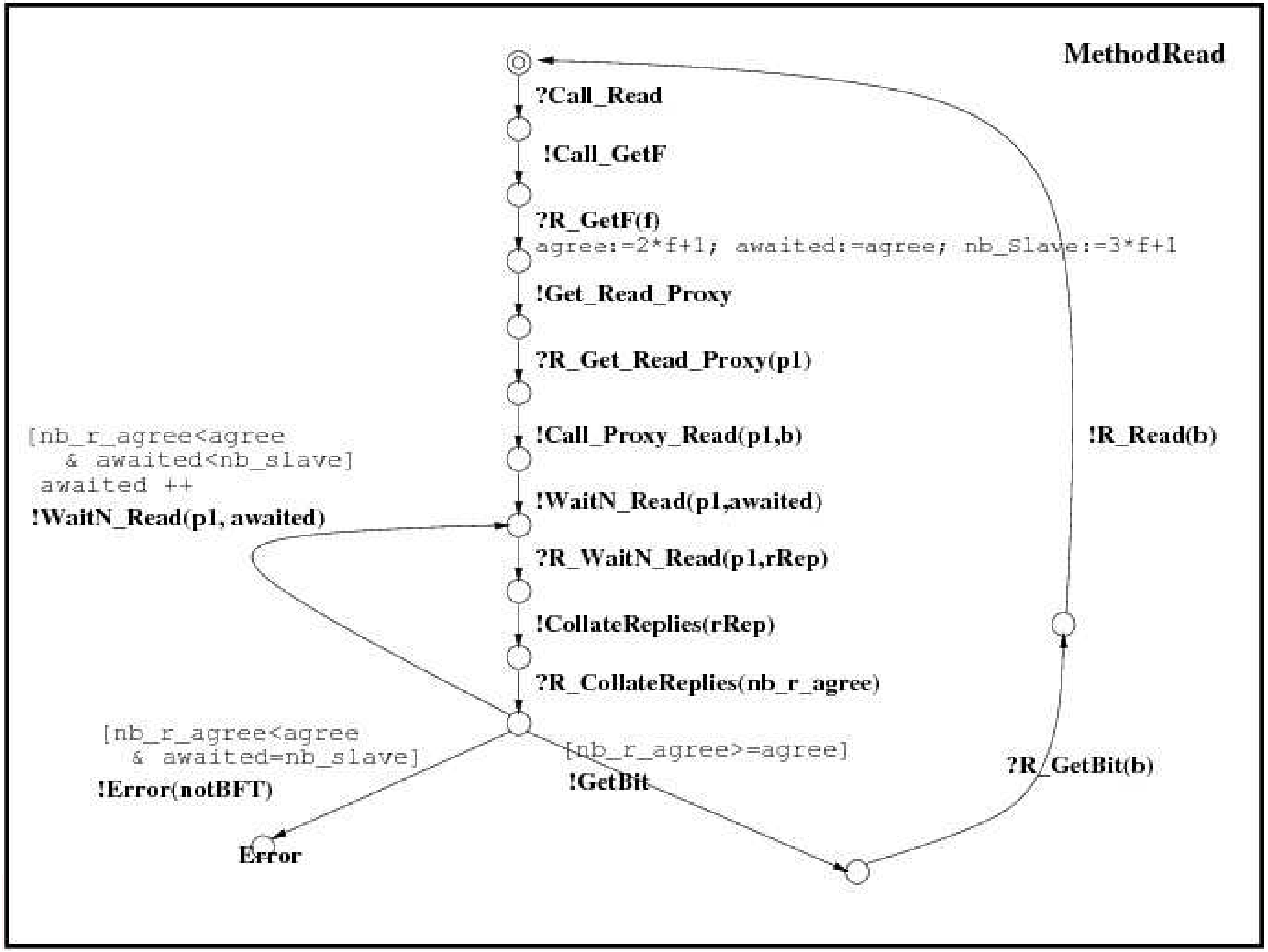
Graphical pNets:

Hierarchical structure of networks

Arrows represent communication/synchronization



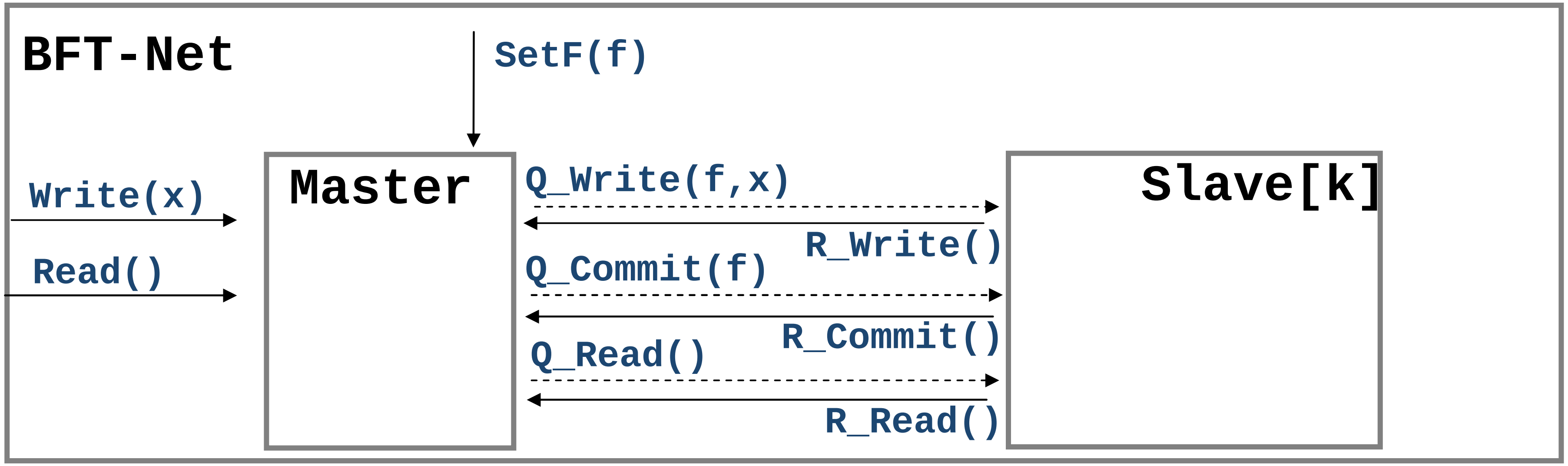
pLTS: parameterized Labelled Transition Systems



Labelled transition systems, with:

- Value passing
- Local variables
- Guards and effects

Synchronization Vectors : generalized parallel operator



Network structure:

BFT-Net : < Master, Slave_1, … ,Slave_n > k ∊ [1:n]

synchronisation vectors :

<?Write(x), - , …, - > => ?Write(x)

<!Q_Write(f,x), ?Q_Write(f,x) , …, ?Q_Write(f,x) > => Q_Write(f,x)

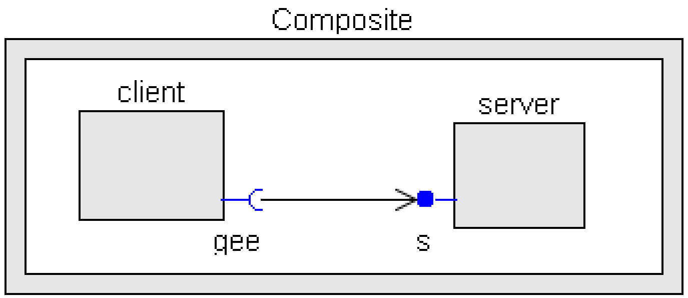
∀k <?R_Write(f,k), - , …, !R_Write(f), …, - > => R_Write(f,k)

Semantics of distributed constructs

The pNets model provides a flexible mechanism for expressing communication and synchronisation operators. We have used it to define a behavioural semantics for:

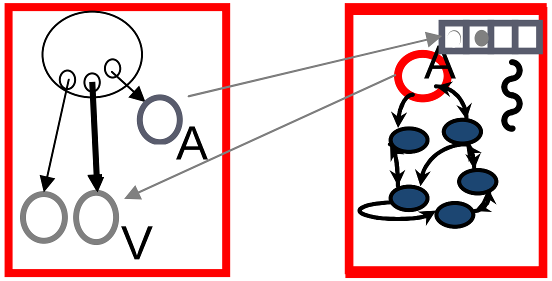
- **Asynchronous active objects**
- **Components**
  - hierarchy, interfaces, bindings
  - Fractal Non-functionnal controllers (life-cycle, binding controller...)
- **GCM components**
  - future proxies, proxy managers
  - first class futures
  - multicast / gathercast interfaces, group controllers

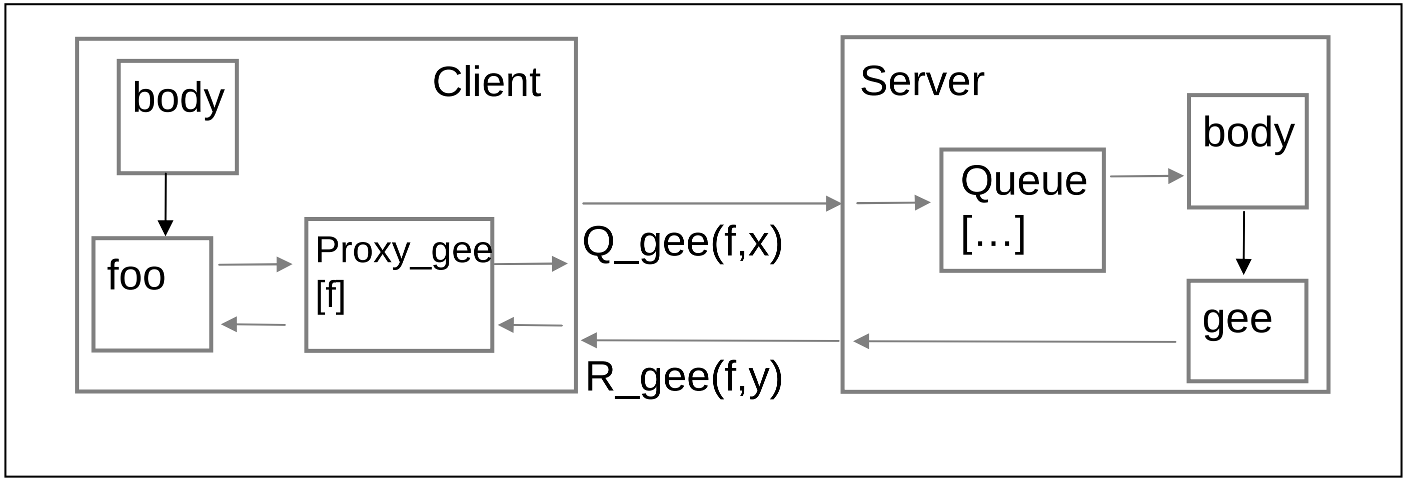
Behavioural Semantics of the GCM (1) : asynchronous communication



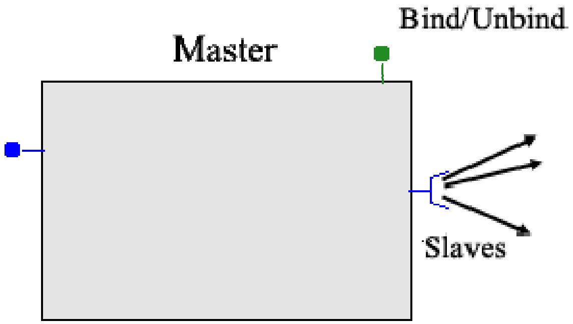
Structural semantic definition:

$$[|\,\text{Comp}\,(\,\{C,S\},\ \text{bindings})\,|] \;=\; \text{pNet}\,(<[|C|],[|S|]>,\ \text{map}\ [|.|]_{SV}\ \text{bindings})$$



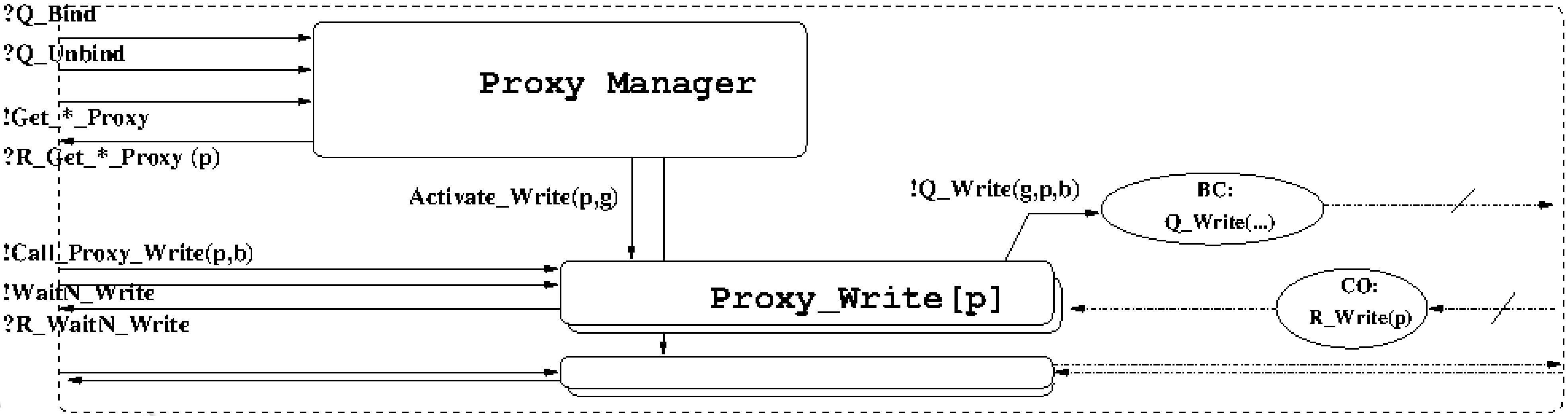


Behavioural Semantics of the GCM (2) : group communication



pNets for a Multicast Interface:

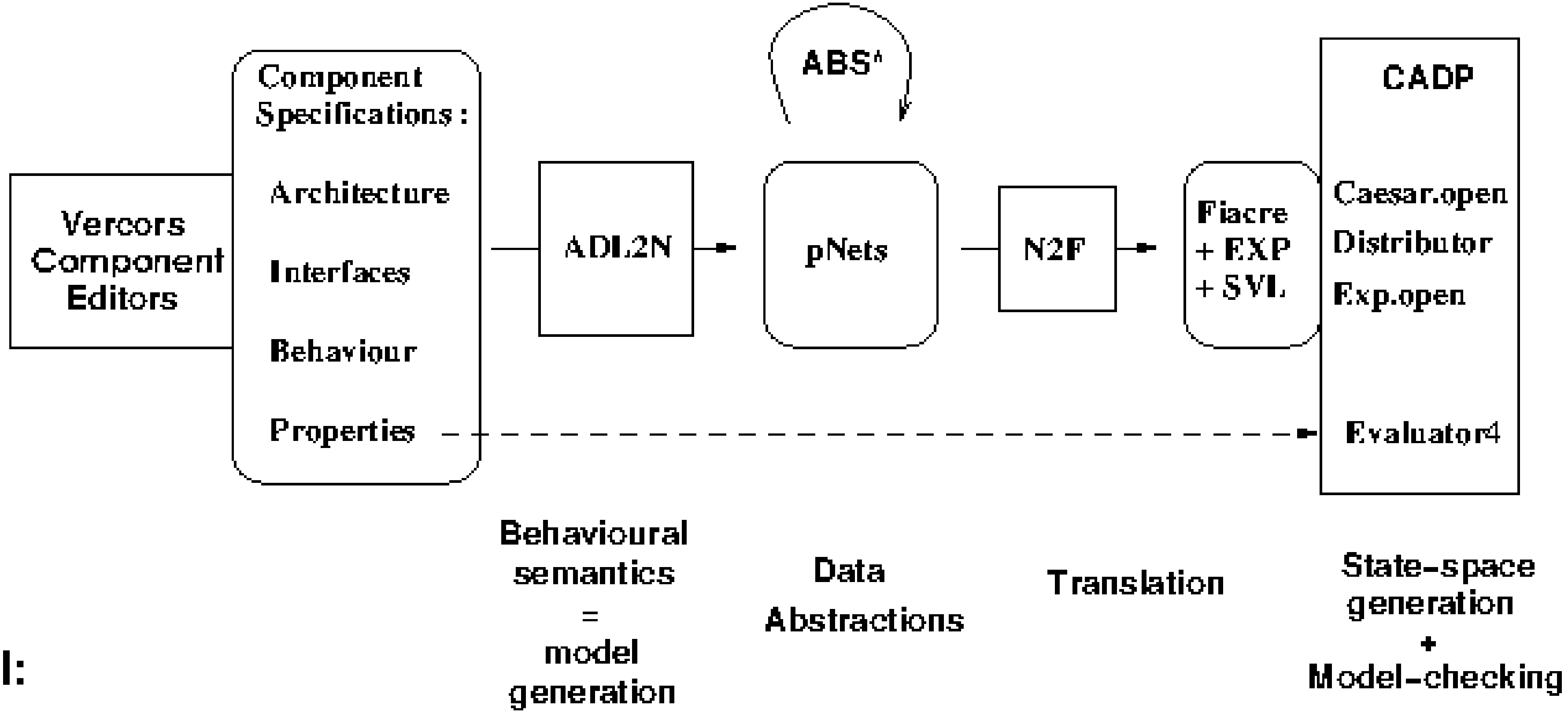
- One proxy family for each method in the interface
- One proxy instance for each call to the method
- Sending the request is Broadcast, collecting the results is asynchronous
- One Group Manager dealing with adding/removing bindings



Agenda

- Related work: A Fast Moving Landscape
- Running example
- Behavioural semantics:
  - The pNets model
  - Semantics of GCM applications
- **Tool platform:**
  - **Formalisms**
  - **Scaling up**
- Conclusion and Perspectives

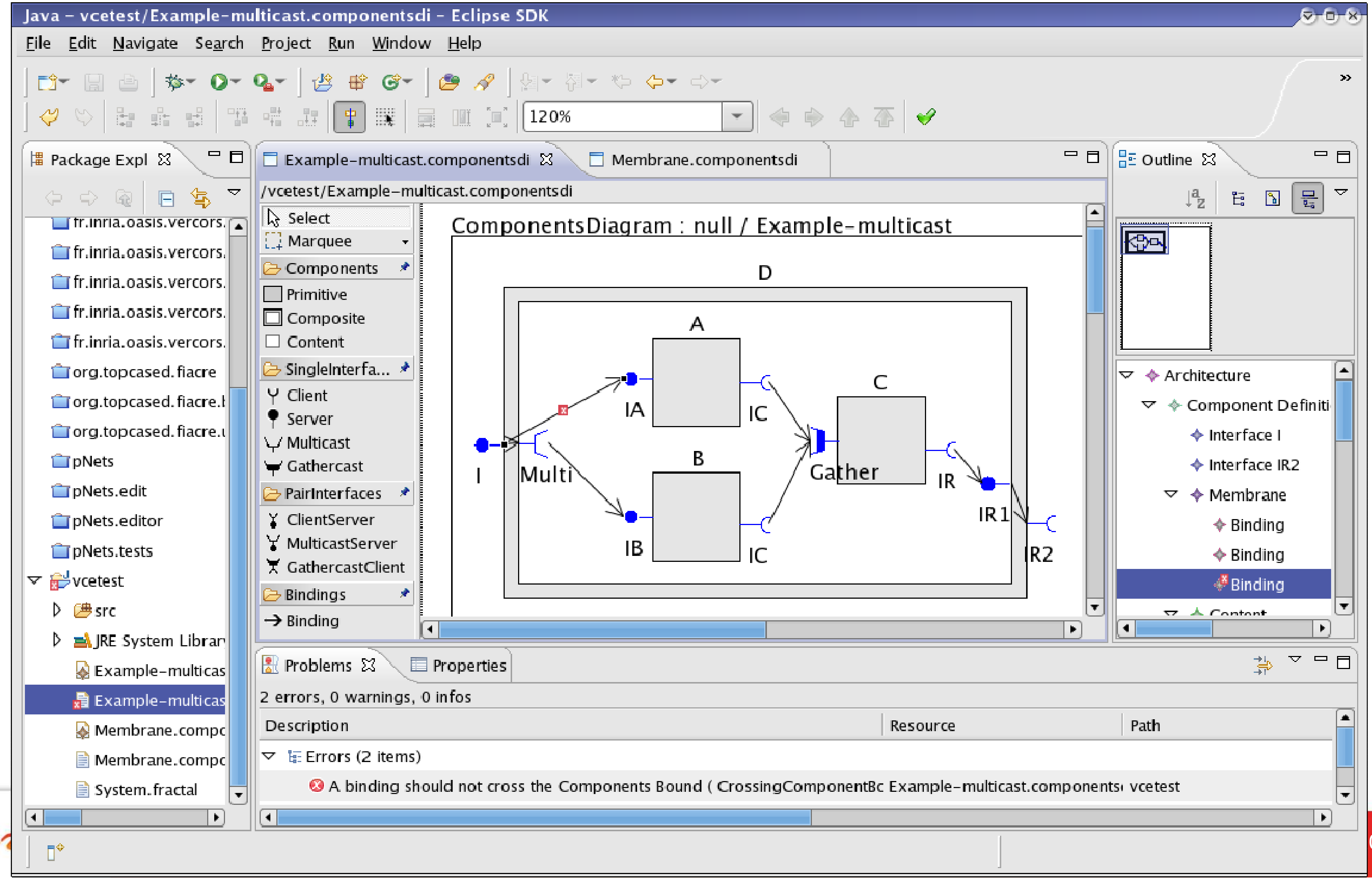
Tool Chain



Goal:

fully automatic chain

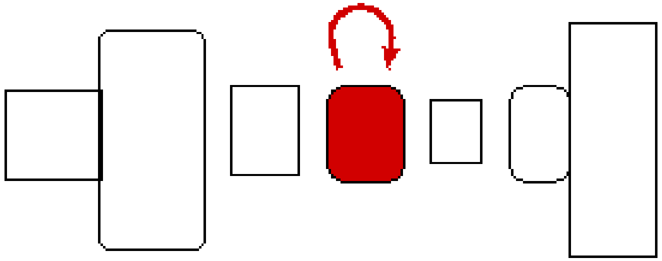
Graphical editors: VCE



2 errors, 0 warnings, 0 infos

Description	Resource	Path
Errors (2 items)		
A binding should not cross the Components Bound (CrossingComponentBc	Example-multicast.components	vcetest

Abstraction



Abstract Model-Checking [Clarke et al, TOPLAS'94]:

- Abstraction of a system into a model is made explicit
- MC is sound & complete on the model, but may be unsound on the system, and often incomplete.

Data Abstraction for Transition Systems [Cleaveland & Riely, CONCUR'94]

- From abstract interpretations for data domains of **value-passing processes,**

➢ builds abstract processes **preserving safety and liveness properties** of the ground process interpretations.

Taming State-Space Explosion

(1) **Data abstraction** (through abstract interpretation):

integers => small intervals

records => structural abstraction

arrays ??? => open question.

(2) Partitioning,

and **minimizing by (branching) bisimulation** + **context** specification

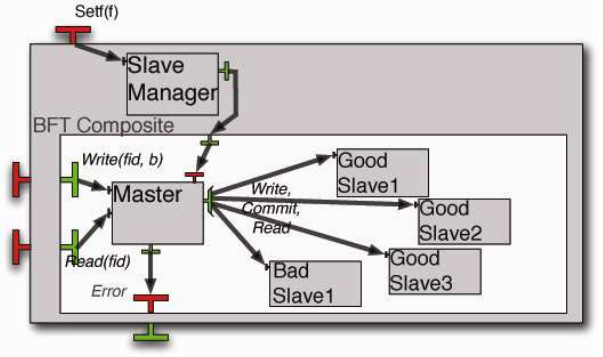
Natural partitioning at components borders

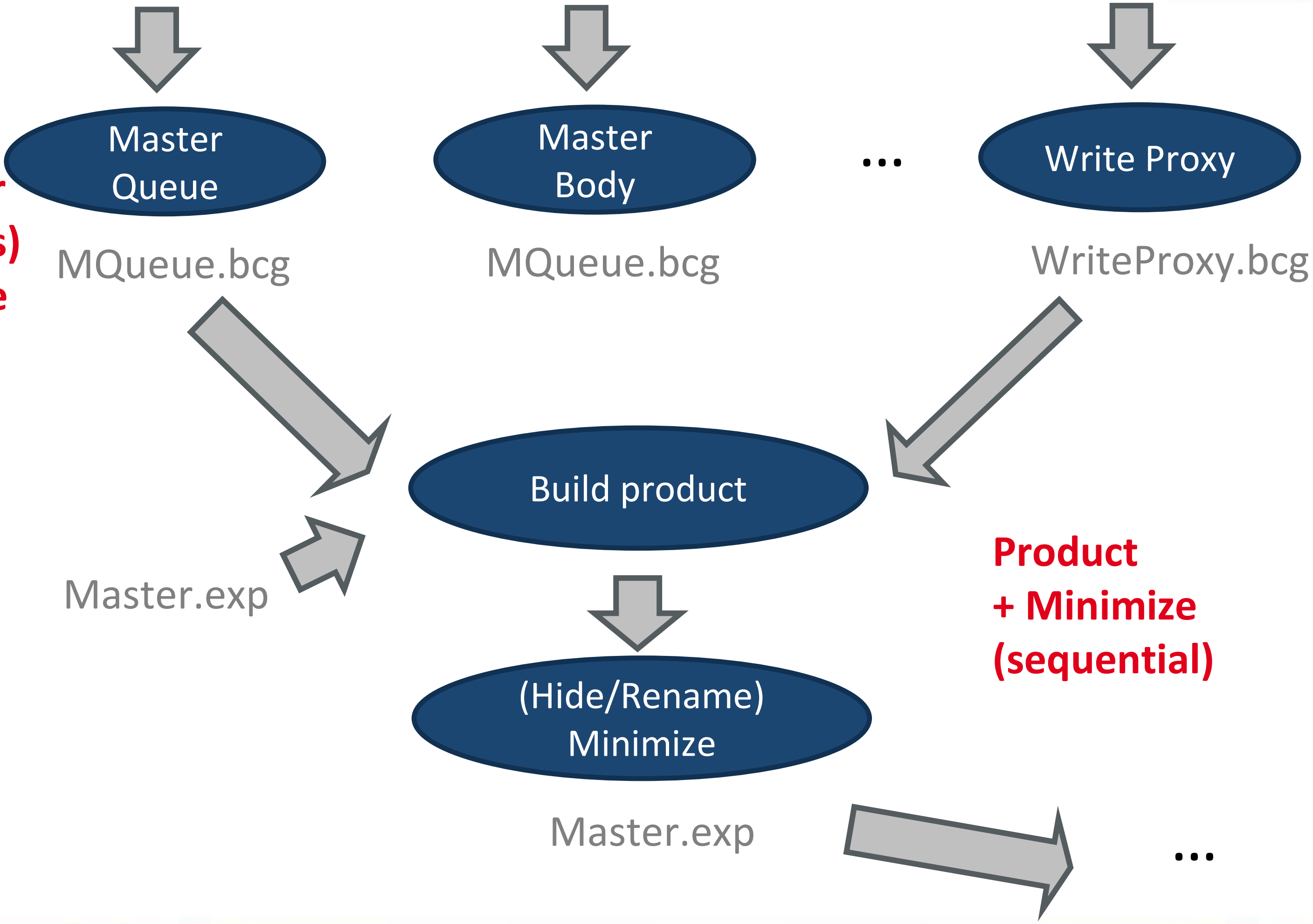
(3) **Distributed verification**.

Only partially available (state-space generation, but no M.C. yet; the bottle-neck is the state-space merging phase).

Example infrastructure: PacaGrid: 1300 cores, 3+ Tbytes of RAM

Model generation workflow

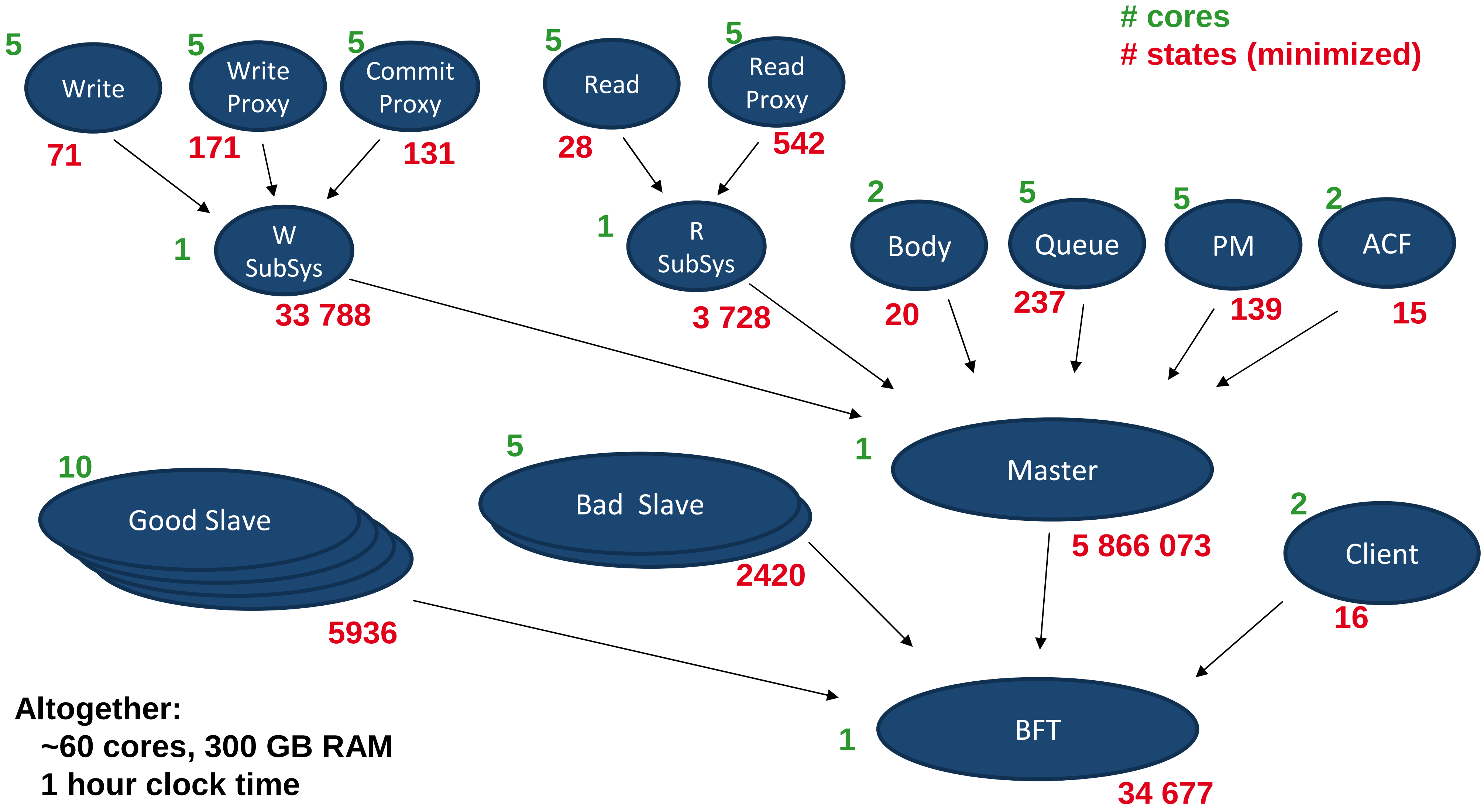




Flac + Distributor (on 5 cores) + Minimize

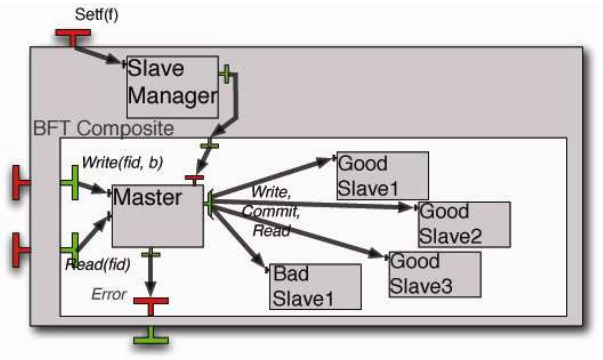
Product + Minimize (sequential)

Model generation workflow



Altogether:
~60 cores, 300 GB RAM
1 hour clock time

Temporal logics properties



Reachability(*):

1- The Read service can terminate

∀ fid:nat among {0...2}. <true* . {!R_Read !fid ?any of bool}> true

2- Is the BFT hypothesis respected by the model ?

< true* . 'Error (NotBFT)'> true

Termination:

After receiving a Q_Write(x) request, it is (fairly) inevitable that the Write services terminates with a R_Write(f) answer, or an Error is raised.

Functional correctness:

After receiving a ?Q_Write(x), and before the next ?Q_Write, a ?Q_Read requests raises a !R_Read(y) response, with y=x

(*) Model Checking Language (MCL), Mateescu et al, FM'08

Conclusion: Summary

- A Behavioural Semantic model: pNets

  flexible, compact, expressive,

  applied to many distributed system features

- A prototype tool platform: Vercors

  editors for specification formalisms, tools for model-generation and abstraction,

  bridges to various verification engines,

  experiments with distributed state-space generation,

- A series of case-studies / scalability tests

Perspectives
(1) Verifying Dynamic Distributed Systems

- **Extend the GCM model generation rules to reconfiguration operations**
- **Identify high-level reconfiguration sequences that have good properties**
- **Use a combination of Theorem-proving, model-checking and runtime**

2 PhD subjects open in the context of 2 industrial collaboration projects:

- Spinnaker: (OSEO funded) “Integrated, Autonomic, and Reliable Deployment and Management for SaaS composite applications ”

- CloudForce: (FUI funded) “Formal Validation of Dynamic Component-based Cloud Applications: Methods and Software Tools”

Perspectives
(2) Code Generation

Generative methods can produce executable code while guaranteeing its properties,

Proposal:

- specify abstract models at early stage of development, generate behavioral models, prove properties of the models,
- generate code skeletons implementing the abstract model and architecture, and run-time validation code (assertions or run-time verification), checking the validity of the implementation wrt. the abstraction.

PhD subject open

➢ Collaboration with the CIRIC lab in Santiago.

Perspectives
(3) *Novel Verification Techniques*

1. **Infinite Systems:**
   - Unbounded Fifo Queues
   - Arithmetic counters
2. **Combining models/algorithms (aka SMT)**
3. **Runtime Verification**
   - For dynamic discovery / adaptation / reconfiguration

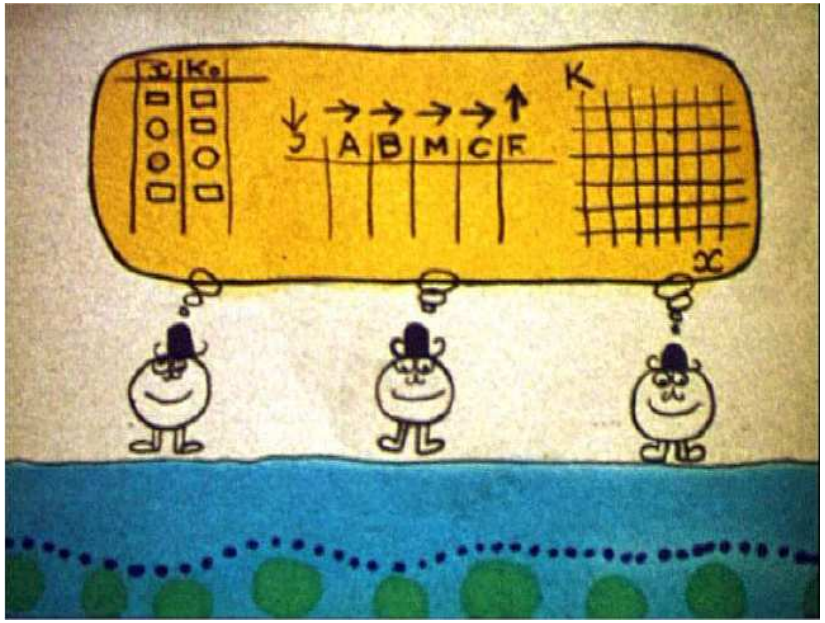
Open Questions

1. **More on data abstraction:**
   - symmetry in useful data structures (intervals, arrays, …),

2. **Context constraints:**
   - ad-hoc correctness proofs (e.g. through proof obligations),
   - links with assume-guaranty approaches, with behavioural typing.

3. **More Tooling**:
   - Proper display of diagnostics
   - Assisted definition of (valid) abstractions.
   - Assisted definition of MC partitioning and strategies.

The origin of distributed applications



The collective mind of the Gibis

« This distributed intelligence is an anticipated plagiarism of Internet collaborative processing … »

in : [« Les Shadoks sont ils décervelables? » G. Berry, Déformaticien [1] au Collège de 'Pataphysique, 2009]

[1] L'informatique c'est la science de l'information, la Déformatique, c'est le contraire.

How far can Gibis scale ????

Thank you

PhDs directions:

Didier Vergamini (1987)
Rabéa Boulifa (2004)
Tomás Barros (2005)
Antonio Cansado (2008)

Recent Collaborative Projects:

ACI sécurité Fiacre (2005-07)
FP6 GridComp (2006-08)
Stic-Amsud ReSeCo (2007-09)
ACI Int. MCorePhP (2010-12)
Oséo Spinnaker (2011 -)
FUI CloudForce (2012 -)

Papers, Use-cases, and Tools at :

http://www-sop.inria.fr/members/Eric.Madelaine

http://www-sop.inria.fr/oasis/Vercors

Distributed State Generation

Abstract model:
f=1, (=> 4 slaves), |data|= 2, |proxies|=3*3, |client requests|=3

Master queue size = 2

~100 cores, max 300 GB RAM

System parts sizes (states/transitions):

Queue	Largest intermediate	Master	Good Slave	Global
237/3189	524/3107	5M/103M	5936/61K	34K/164K

Time
59’

Estimated brute force state spaces :

10^{18}	6.10^{3}	$\sim 10^{32}$