Formal Methods && Tools Group

Stefania Gnesi

Outline

- Overview of the Formal Methods && Tools Lab

Who we are

Research activities

Projects

More info on the Lab:
- http://matrix.iei.pi.cnr.it/FMT

Who we are

Research Staff

Patrizia Asirelli	Tommaso Bolognesi	
Giorgio Faconti	Alessandro Fantechi	External Collaborator-Univ. Firenze
Stefania Gnesi	Diego Latella	
Mieke Massink	Franco Mazzanti	

Maurice ter Beek	ERCIM fellolw
Giuseppe Lami	PHD student (Ingegneria Informatica-Pisa)
Gabriele Lenzini	PHD student (Twente-NL)

Associated Research Staff

Maria Lisa Masseti

Gianluca Trentanni

Formal Specification and Verification of Complex Systems

The *Formal Methods && Tools Group* is active in the fields of *development and application of formal notations, methods and software support tools for the specification, design and verification of complex computer systems*.

These systems often must meet **real-time, security constraints and are used in safety-critical missions where also human factors play a major role.**

Formal Specification and Verification of Complex Systems

We are currently involved in research activities in the areas of:

- Model-checking algorithms,tools and applications
- Quantitative extensions of Process Algebras e related tools
- Formal Approaches to the modeling of Human-Computer Interaction
- Precise UML
- Formal Approaches to Requirements Engineering
- Integration of process-algebraic, state-based and functional specification models

Current Projects activity

The *Formal Methods && Tools Group* is active in several international and national projects:

* AGILE, Architectures for Mobility Information Societies Technology (IST PROGRAMME IST-2001-32747, 2002-2004)
* CAFE, IT EUREKA Project "Information Technology for European Advancement" (ITEA, 2001-2003)
* PRIDE, ambiente di PRogettazione Integrato per sistemi Dependable (Italian Space Agency, 2002-2003)
* SP4- High-Quality Service Software Architectures for Global Computing on Co-operative Wide Area Networks (MURST 5% 2002-2004)
* PROFUNDIS, Proofs of Functionality for Mobile Distributed Systems (IST PROGRAMME IST-2001-33100, 2002-2004)
* QUACK, A Platform for the Quality of New Generation Integrated Embedded Systems (Progetto MURST 40%, 2002-2003)
* COVER (Progetto MURST 40%, 2003-2004)
* MEFISTO, Metodi Formali per la Sicurezza ed il Tempo (Progetto MURST 40%, 2002-2003)

Software Tools development

The *Formal Methods && Tools Group* has developed several verification tools:

* JACK Project (Just Another Concurrency Kit)
  - AMC: ACTL model checker for fc2 automatons
  - BMC: BDD based ACTL+ model checker for networks of automata
  - FMC: (fmc, totab, tofc2) a set of tools for exploration and verification of networks of automata, including an "on the fly" model checker for full μ-calculus (ACTL-compatible)

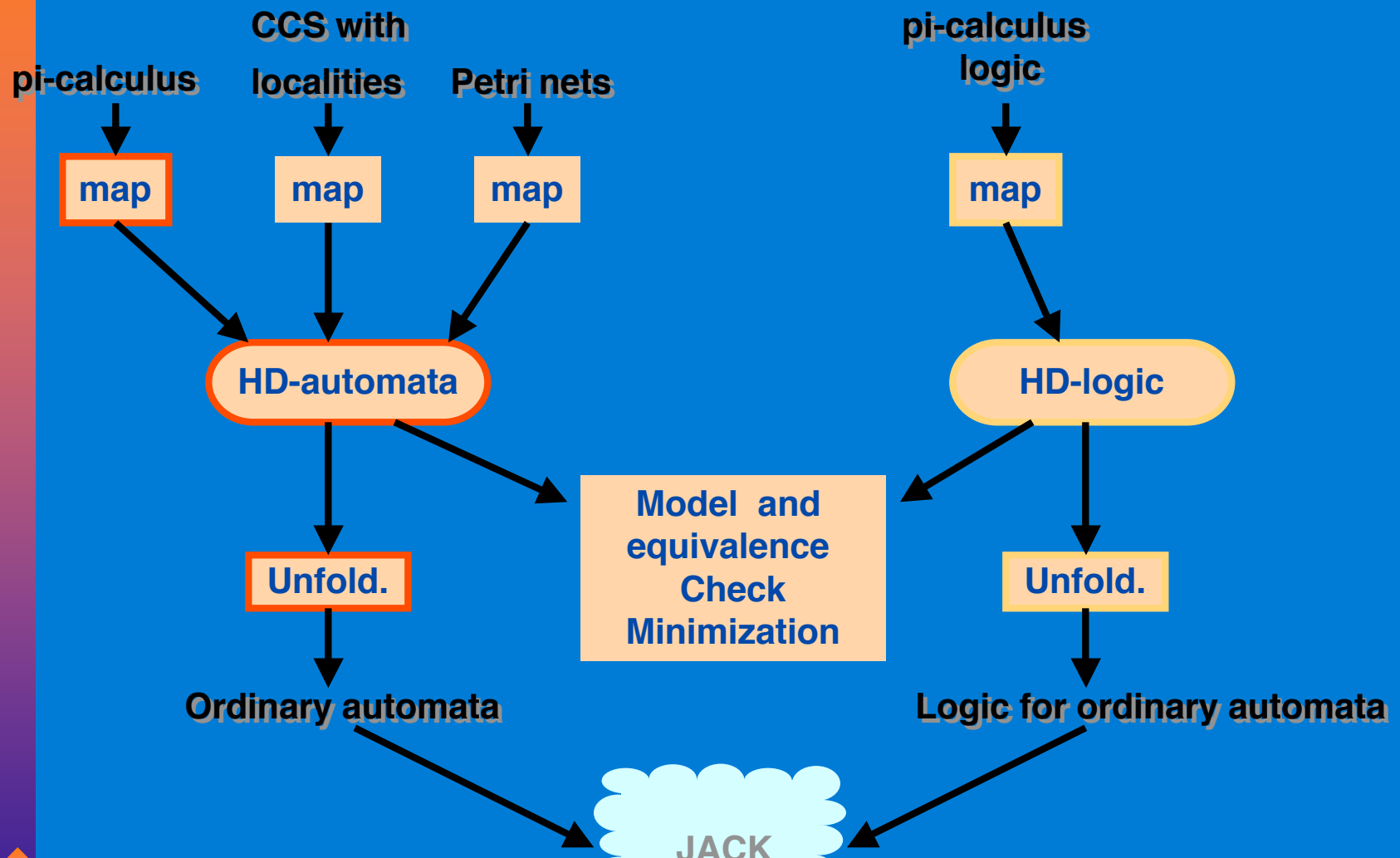
* HAL (History-dependent Analysis Laboratory): μ-calculus verification environment
  - PMC μμlogic model checker
  - HAL on Line: μ-calculus verification environment directly on the web.

•UMCTOOLS: (um, totab, xmi2umc) a set of tools for the exploration and verification of networks of automata, including an "on the fly" model checker for full μ-calculus and UML statemachines

As well as tools for supporting the analysis of NL requirements:

* QuARS Quality Analyzer of Requirements Specifications (in collaboration with CCS)

Model checking mobile systems



-calculus requires an infinite number of states also for very simple agents. The creation of a new name gives rise to an infinite set of transitions: one for each choice of the new name.

In HD-automata names appear explicitly in states, transitions and labels (local names) . Local names do not have a global identity.

In this way, for instance, a single state of the HD-automaton can be used to represent all the states of a system that differ just for a bijective renaming.

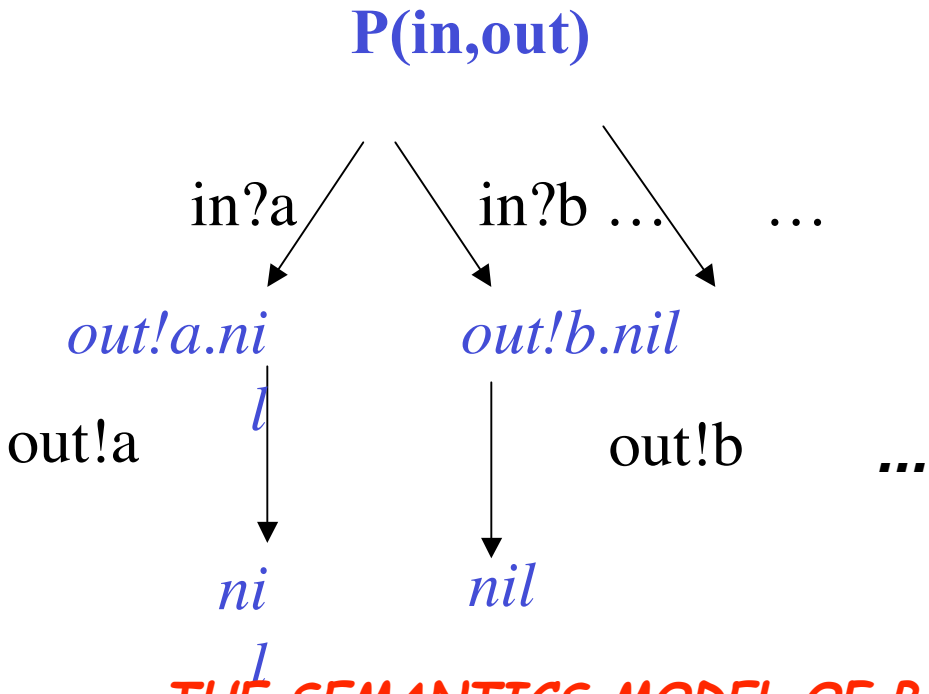
P(in,out) ::= in?(x). out! x nil

x,in,out ∈ N

N infinite sets of names

in, out: channels

x : place holder



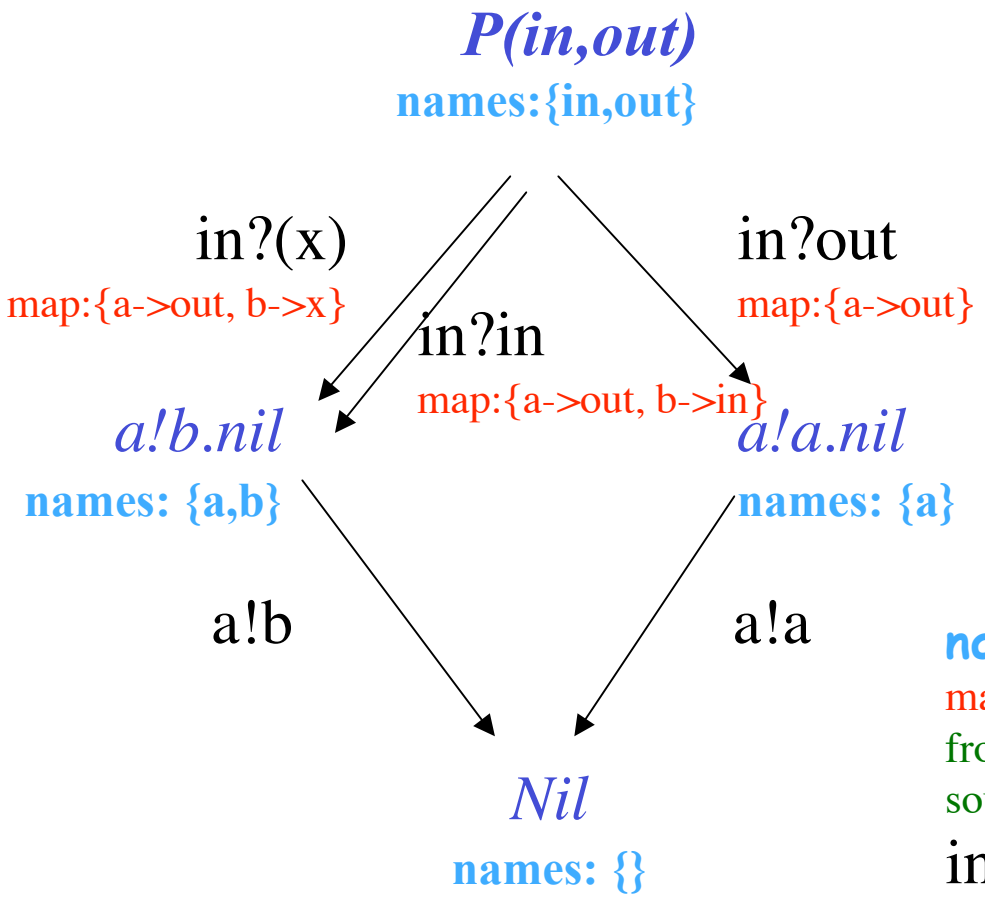
THE SEMANTICS MODEL OF P IS:

INFINITE STATE

INFINITE BRANCHING

FROM π-calculus to HD-AUTOMATA

P(in,out) ::= in?(x). out! x nil



names:{...} local names

map:{a->out, b->x} embedding function from names of the target state to the source state

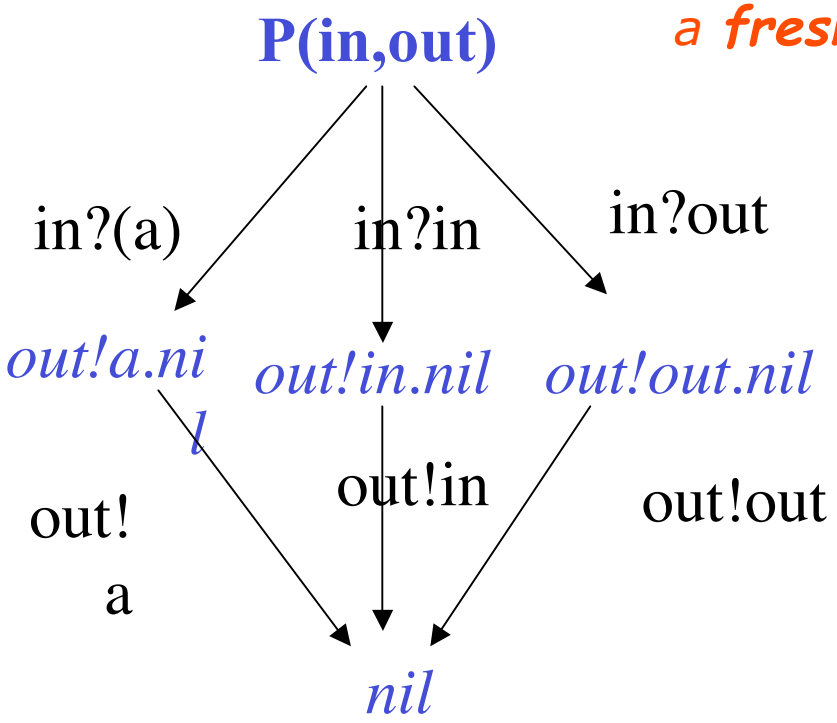
in?(x) **input fresh name**

FROM HD-AUTOMATA TO LTSs

P(in,out) ::= in?(x). out! x nil

in,out **are the active names of P**

a ***fresh name***



The HAL environment: an overview

HAL is written in C++ and compiles with the GNU C++ compiler (the GUI is written in Tcl/Tk).
It is currently running on SUN stations (under SUN-OS) and on PC stations (under Linux).

The □-logic: sintax

□-logic syntax -->

□ ::= *true* | ~ □| □□ □ | □ X{□}□| <□> □| □ F □

□ ::= tau | x!y |x!(y) | x?y

□ X{□}□ strong next

<□> □ weak next

□ F □ eventually

As usual [□] □, AG □ can be defined by duality

□-logic is adequate with respect to strong early bisimulation equivalence

A translation function exists from □-logic to ACTL

soundness : a □-logic formula is satisfied by a □ -calculus agent P if and only if the finite state ordinary automaton associated with P satisfies the corresponding ACTL formula

The translation of a formula is thus not unique, but depends on the agent P. Specifically, it depends on the set S of the fresh names of the ordinary automaton associated with the agent P.

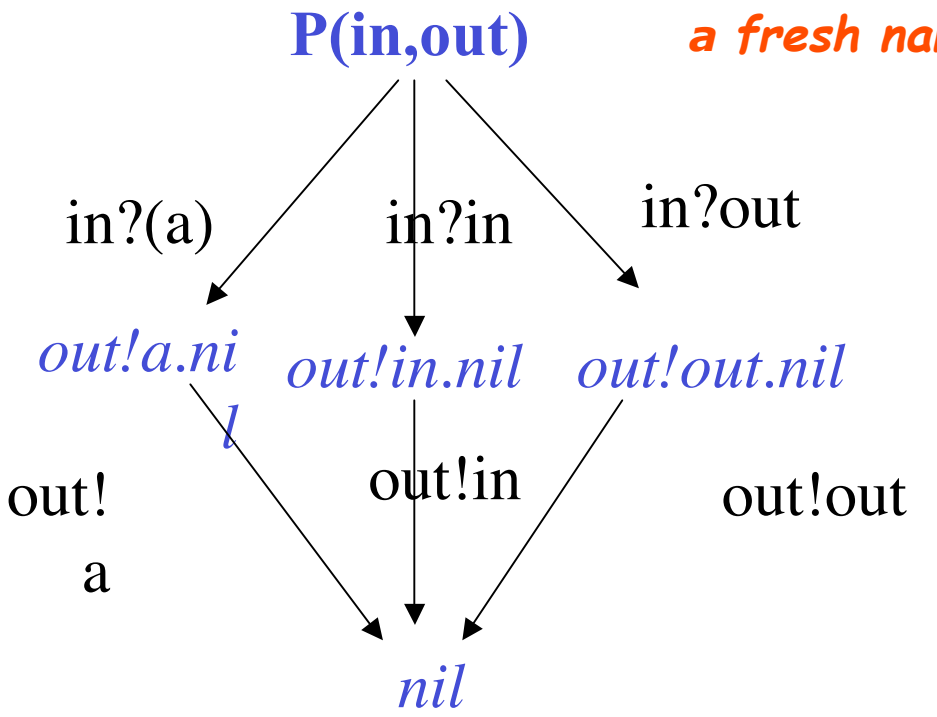
Model checking facilities

P(in,out) ::= in?(x). out! x nil

EX {in?u}EX{out!u} true (□-logic)

EX {in?(a)}EX{out!a} true (ACTL)

a fresh name



Verification on the web

Development of a distributed environment for the verification of properties of distributed, mobile systems.

- Tool developed using different specification and verification methodologies
- Different platforms and languages.

Web application for mobile systems

Web as infrastructure

Specification and verification modules= WEB services

Interaction based on HTTP/XML plus

- remote invocation (e.g. \xmlrpc\, SOAP),
- directory and service binding (e.g. UDDI, trader),
- language to express service features (e.g. WSDL)

It will become the standard functional platform to programming applications over the WEB.

Precise ULM and model checking

A formal operational semantics for a behavioural subset of UML Statechart Diagrams (UMLSDs) including a formal proof of their correctness with respect to major UML semantics requirements concerning behavioural issues

Conceptual issues related to building a tool for both linear and branching Time model-checking, for the automatic verification of formal correctness of UML Multicharts. (Spin,Jack)

Recently we have started a new project aimed at developing an on the fly Model Checker, UMC, for UML communicating state machines.
The current alpha-version of the UMC prototype is accessible "online"
http://matrix.iei.pi.cnr.it/umc/demo.

UML plus Mobility/Locality

A formal operational semantics for a behavioural subset of UML Statechart Diagrams (UMLSDs) extended with mobility a la □-calculs has been defined.

A new extension with localities a la Klaim is ongoing.

UMC can be used also to verified properties of UML statemachines taking into account locality aspects. An extension of the ACTL logic with assertion predicates has been defined to this purpose.

i.e. We can check properties like:

it is true that passengers can eat, only when their plane is flying.

```
AG ( ( EX { eating }  true ) -> ASSERT(Plane1.Status=1) )
```

Challenging research themes

- From Qualitative to Quantitative Formalisms
- Integration of Formal Methods towards unifying paradigms
- Combination of NL analysis techniques and formal methods for the early validation of requirements
- Combination Model Checking and Testing Techniques
- Tools development

Traditional view

Centralized Toolkit

Integration between different tools is given by:

- Common formats (I.e. FC2) & pipelining

– Problems

- Interoperability
- Dynamic reconfiguration

mihda =
http://jordie.di.unipi.it:8080/mihda/hd

hal =
http://matrix.iei.pi.cnr.it:8080/hal