CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes

Hubert Garavel - Frédéric Lang

Radu Mateescu - Wendelin Serwe

INRIA - LIG / VASY

http://vasy.inria.fr

Another view on model checking

- Starting point: Quasar model checker [Queille-Sifakis 1982]
- Combining model checking with:
  - Concurrency theory
  - Formal methods
  - Compiler construction
- Emphasis on:
  - Concurrent languages with a formal semantics
  - Process calculi: minimal concepts, maximal expressiveness
  - Bisimulations, congruence results, temporal logic compatibility
- European school of model checking:
  - CADP (FR), CSP/FDR2 (UK), FSP/LTSA (UK), μCRL/mCRL2 (NL) + CWB-NC, XSB (USA)
  - Not the mainstream approach, but effective
  - Uses also equivalence checking and theorem proving

CADP 2010 (cadp.inria.fr)

- A long-run effort:
  - 25 years of development
  - Initially: only 2 tools (*CAESAR* and *ALDEBARAN*)
  - Today: 50 tools

- A modular toolbox for asychronous systems
  - Generic software components for verification
  - Modular, extensible architecture (APIs)

- Licensed by INRIA - Free for academics

CADP wrt other model checkers

- Concurrent systems (rather than sequential programs)
- High-level languages with a formal semantics (not C/C++)
- Message passing (rather than shared memory (*))
- Dynamic data structures (records, unions, lists, trees…)
- Action-based semantics (rather than state-based)
- Branching-time logic (rather than linear-time)
- Explicit-state verification (rather than symbolic)

(*) *various forms of shared memories can modelled using message passing*

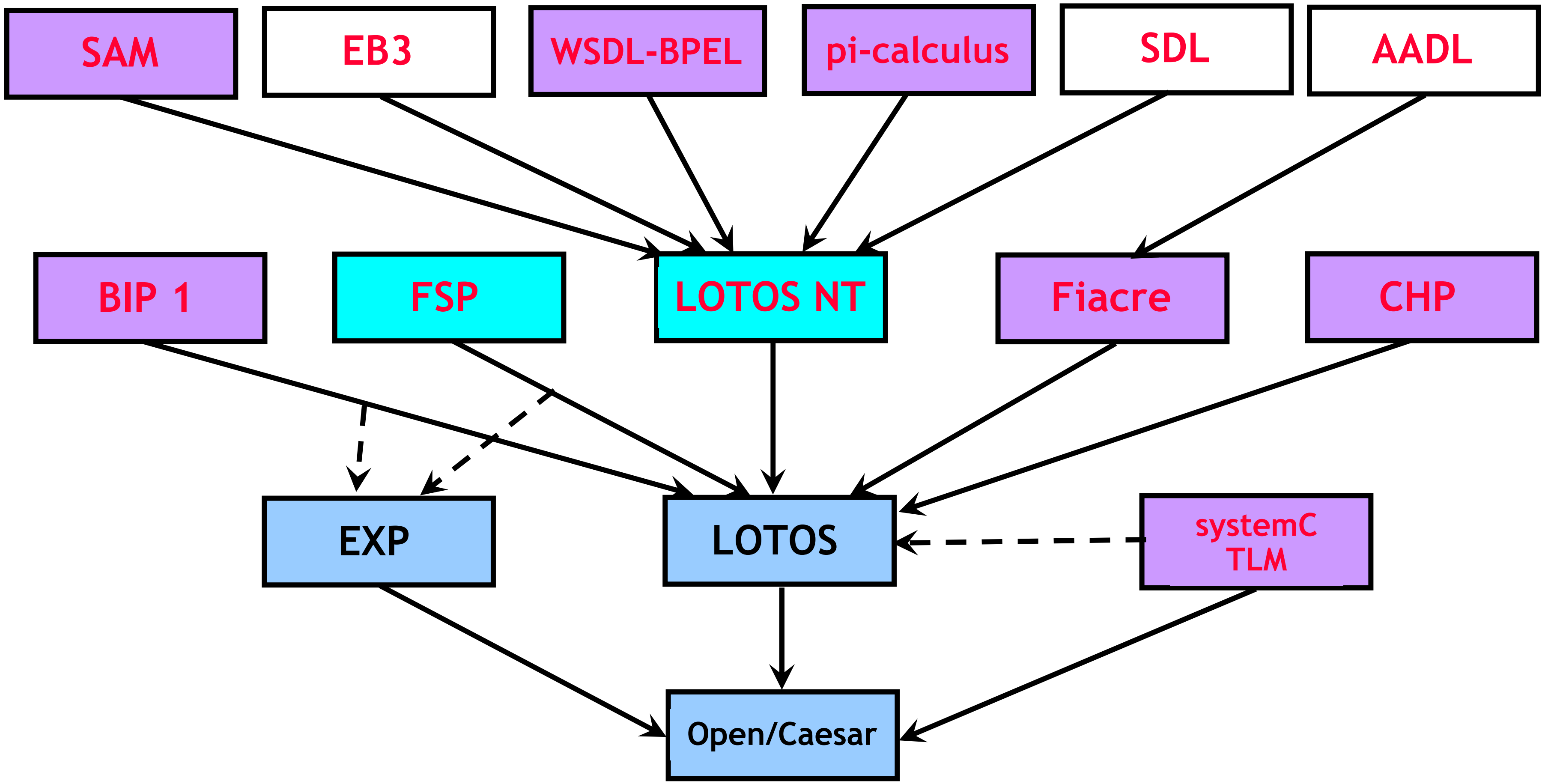
Modelling using LOTOS

- LOTOS (ISO standard 8807):
  - Types: algebraic data types (inductive definitions)
  - Functions: rewrite rules with priorities (or C code import)
  - Processes: calculus based on CCS and CSP
- Tools: efficient LOTOS to C compilers
  - rapid prototyping
  - step by step simulation
  - state space exploration (LTS generation)
- Assessment of LOTOS:
  - really expressive (except quantitative time and mobility)
  - steep learning curve (especially for industry engineers)

Modelling using LOTOS NT

- Goal: replace LOTOS with a better language
- LOTOS NT:
  - a concurrent language developed at INRIA/VASY
  - inspired from E-LOTOS (ISO standard 15437)
  - concrete syntax close to Pascal, Ada, Eiffel
  - very positive industrial feedback
  - Types:
    - inductive types
    - basic types: bool, nat, int, real (= float), char, string
    - ranges, predicate types, arrays, sets, lists, sorted lists, etc.
  - Functions:
    - imperative style (yet strictly functional)
  - Processes:
    - imperative style (yet still a process calculus)
- Tools: translation chain from LOTOS NT to LOTOS

… and many other languages



Model checking (1/2)

RAFMC language

- Alternation-free modal μ-calculus extended with regular expressions
- Action predicates:
  - strings
  - regular expressions
  - not, and, or …
- Path formulas:
  - regular expressions over actions
- State formulas:
  - [Action] φ, <Action> φ
  - [Path] φ, <Path> φ
  - not, and, or …
  - least and greatest fixed points

Evaluator 3.6 model checker

- On-the-fly model checker for the RAFMC language
- Automatic generation of diagnostics (LTS fragments: sequences, trees, or lassos)
- Libraries of standard property patterns of current use

Model checking (2/2)

MCL (Model Checking Language)

- Predefined types
  - boolean, natural, integer, natset, real, character, string
- Extended action formulas
  - value extraction from LTS labels
  - value matching
- Extended path formulas
  - if-then-else, case, let, while, repeat, for, ...
  - enables counting of actions
- Extended state formulas
  - fixed point variables parameterized with typed variables
  - if-then-else, case, let
  - quantifiers over finite domains
  - fairness operators inspired from PDL-Δ to describe cyclic behaviours

Evaluator 4.0 model checker

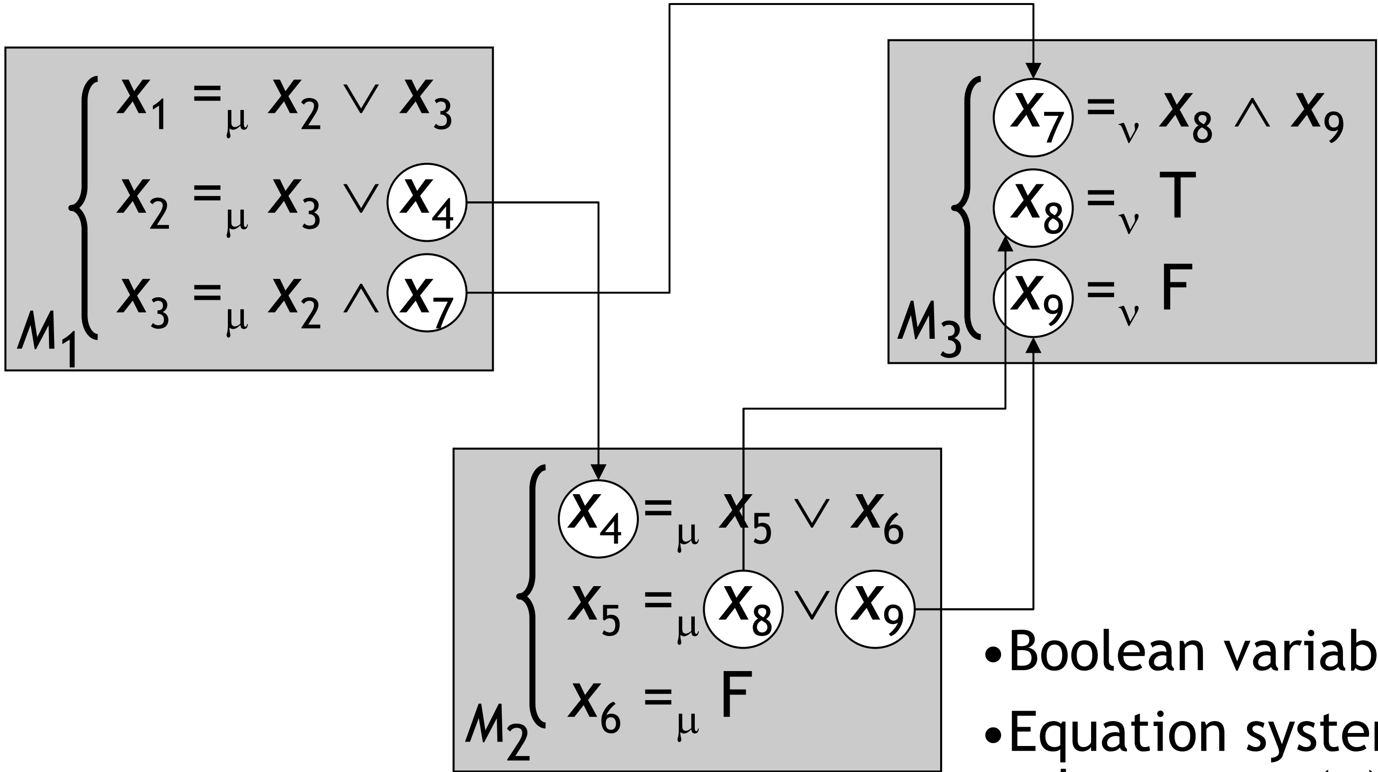
- On-the-fly model checker for the MCL language
- Diagnostic generation
- Expressiveness:
  - MCL subsumes RAFMC
  - temporal formula with data
- Efficiency:
  - "reasonable" model checking complexity
  - linear-time for RAFMC (i.e., dataless formulas)
  - based on Parameterized Boolean Equation Systems

Equivalence checking

- Based on bisimulation theory
- Minimization of an LTS
  - signature-based partition refinement [Blom 2004]
  - handles 10^9 – 10^{10} explicit states
- Comparison of two LTSs
  - on-the-fly comparison + diagnostics generation
  - 7 equivalence relations supported (+ their preorders)
- Projection of an LTS on an "interface"
  - on-the-fly generation of a process constrained by its environment represented as a regular language
  - [Graf-Steffen 1990], [Krimm-Mounier 1997]

Boolean Equation Systems (1/2)

BES: unified framework for model/equivalence checking



- Boolean variables
- Equation systems with least (μ) and greatest (ν) fixed points
- DAG dependencies between systems (alternation-free)

Boolean Equation Systems (2/2)

- CADP provides solvers for :
  - for explicit BES (stored in a file)
  - for implicit BES (built and solved on the fly)
  - 9 resolution algorithms (general vs specialized)
  - diagnostics generation (examples or counter-examples)
  - fully documented API
  - handles billions of variables and equations

- Parameterized Boolean Equation Systems (PBES)
  - "1st order"-extension of BES [Mateescu's thesis, 1998]
  - boolean variables -> predicates with user-defined types
  - a popular model: Grenoble, Eindhoven, Oxford, Twente, …

Ways to fight state explosion

- Bisimulation-preserving optimizations:
  - Petri net structural transformations
  - Petri net reachability analysis using BDDs
  - static analysis on data flow
- Compositional verification:
  - component minimization and recombination (divide and conqueer)
  - component constrained using interfaces
- On-the-fly verification:
  - on-the-fly BES resolution
  - on-the-fly model checking
  - on-the-fly equivalence checking
- Distributed verification using NoWs, clusters, and grids:
  - distributed LTS exploration
  - distributed BES resolution

Explicit-state verification works well with "proper" languages

Performance evaluation

- Analysis of concurrent systems:
  - model checking: qualitative queries (booleans)
  - performance evaluation: quantitative queries (numbers)
- Unifying action-based semantic models:
  - LTS vs (discrete/continuous time) Markov chains
  - bisimulation vs lumpability
  - Hermanns' *Interactive Markov Chains* (IMC)
- CADP tools for performance evaluation:
  - compositional generation of IMCs
  - transient and steady-state Markov solvers
  - Markovian simulator

Conclusion - Facts

- CADP: a bridge between theory and practice
  - Combines concurrency theory and computer-aided verification
  - A significant development effort:
    - 50 tools integrated together
    - 700 pages of documentation
    - Availability: Linux, MacOS X, Solaris, Windows (x86, x64, Itanium, PowerPC, Sparc)
- Dissemination figures
  - 139 case-studies performed using CADP
  - 57 research tools built using CADP
  - 500-700 licenses granted every year
  - Forum: 185 users - 1280 messages

Conclusion - Applicability

- All kinds of concurrent systems
  - no quantitative time
  - limited mobility
- Computer science applications
  - hardware, software, telecom, embedded, security
  - lowest level: asynchronous circuits
  - highest level: cloud computing
- "Exotic" applications
  - bio-informatics (genetic regulation networks)
  - cognition (concurrent models of the human brain)

For more information:

http://cadp.inria.fr