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/

Talk overview

- Part I: What is CADP?
- Part II: What is new in CADP 2010?
- Conclusion

What is CADP?

CADP

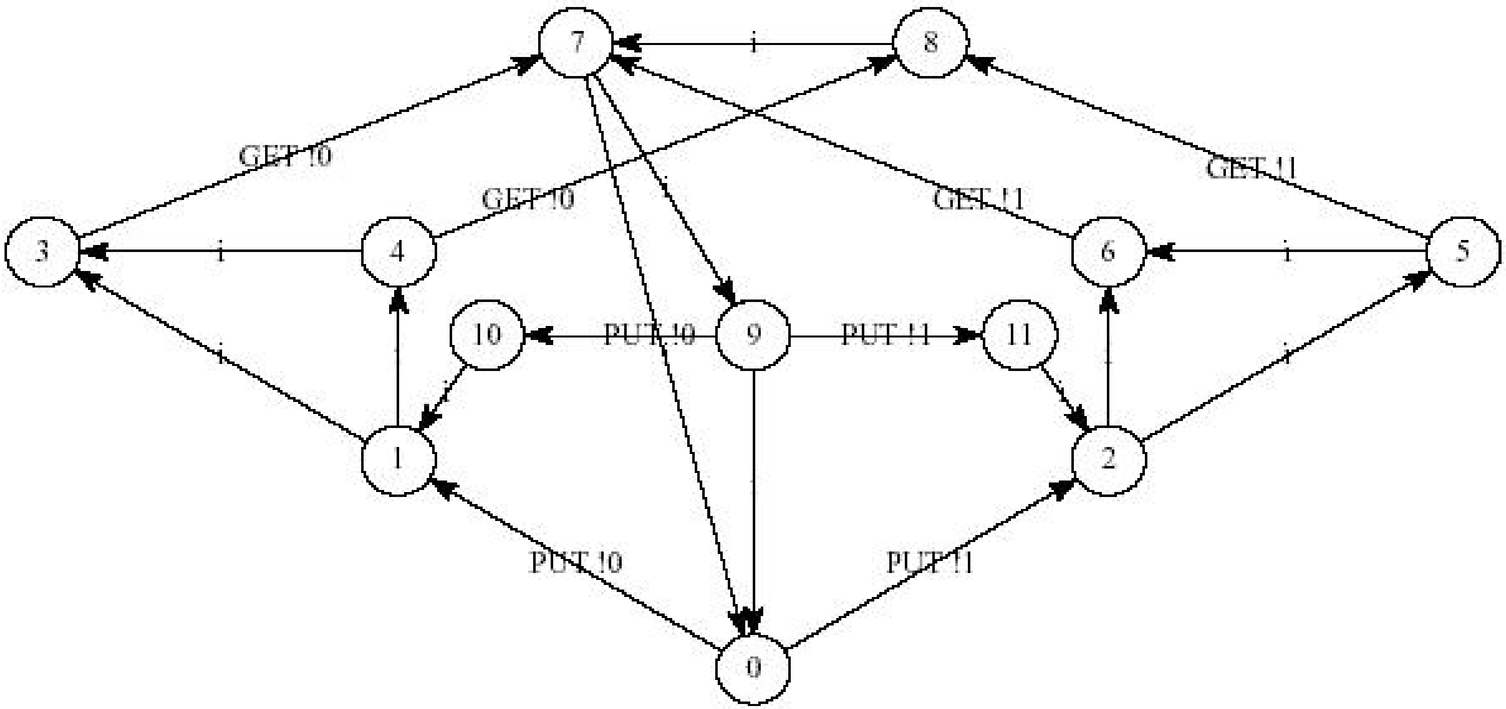
- A modular toolbox for asychronous systems
- At the crossroads between:
  - concurrency theory
  - formal methods
  - computer-aided verification
  - compiler construction
- A long-run effort:
  - development of CADP started in the mid 80s
  - initially: only 2 tools (CAESAR and ALDEBARAN)
  - last stable version: CADP 2006
  - today: nearly 50 tools in CADP 2010

CADP main features

- Formal specification languages
- Verification paradigms:
  - Model checking (modal μ-calculus)
  - Equivalence checking (bisimulations)
  - Visual checking (graph drawing)
- Verification techniques:
  - Reachability analysis
  - On-the-fly verification
  - Compositional verification
  - Distributed verification
  - Static analysis
- Other features:
  - Step-by-step simulation
  - Rapid prototyping
  - Test-case generation
  - Performance evaluation

Verification technology: LTS (1/2)

- Labelled Transition Systems
- LTS = state-transition graph
  - no information attached to states (except the initial state)
  - information ("labels" or "actions") attached to transitions

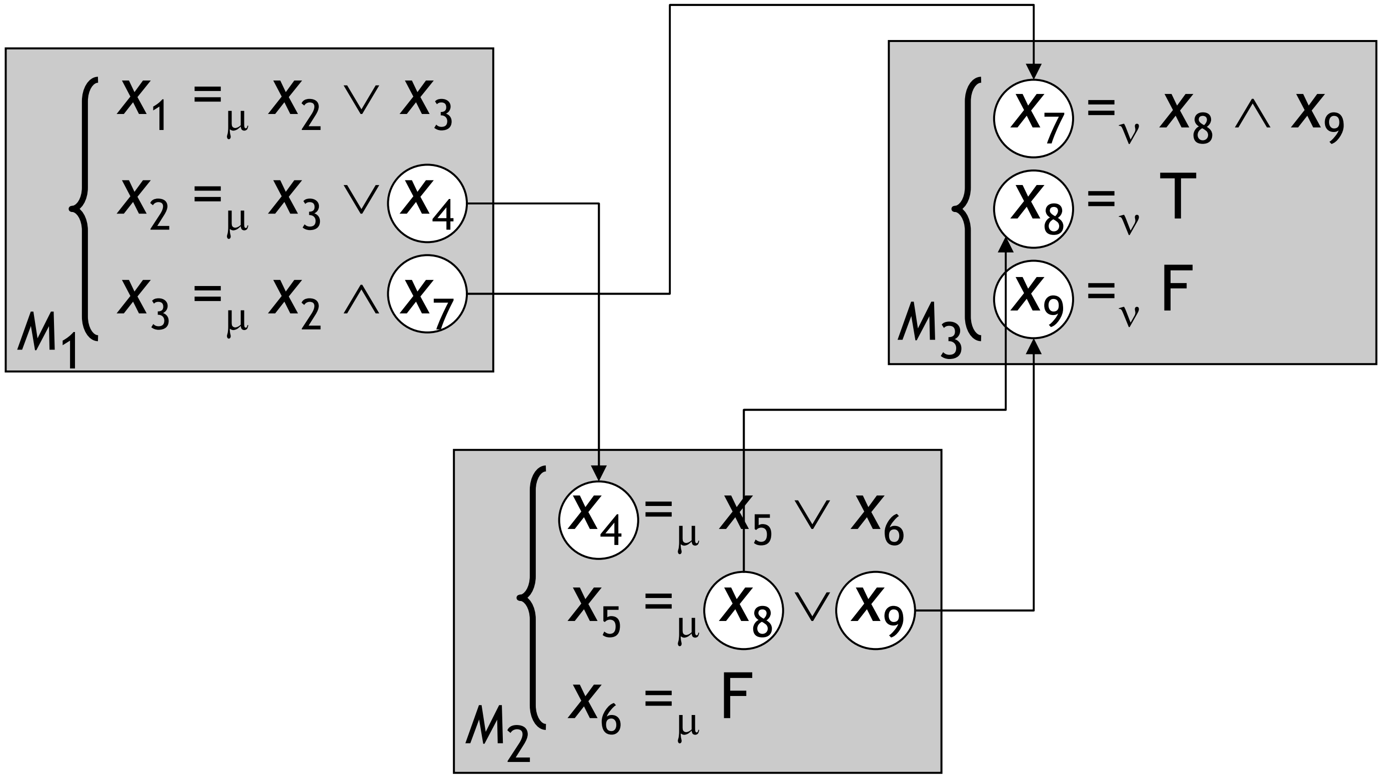


Verification technology: LTS (2/2)

- "Explicit" LTS *(enumerative, global)*:
  - comprehensive sets of states, transitions, labels
  - BCG: a file format for storing large LTSs
  - a set of tools for handling BCG files
  - CADP 2010: BCG limits extended from 2^{29} to 2^{44}
- "Implicit" LTS *(on the fly, local)*:
  - defined by initial state and transition function
  - Open/Caesar: a language-independent API
  - many languages connected to Open/Caesar
  - many tools developed on top of Open/Caesar

Verification technology: BES (1/2)

- Boolean Equation Systems
- least (μ) and greatest (ν) fix points
- DAG of equation systems (no cycles – alternation-free)



Verification technology : BES (2/2)

- BES can be given:
  - explicitly (stored in a file)
  - or implicitly (generated on the fly)
- CAESAR_SOLVE: a solver for implicit BES
  - works on the fly: explores while solving
  - translates dynamically BES into Boolean graphs
  - implements 9 resolution algorithms A0-A8 (general vs specialized)
  - generates diagnostics (examples or counter-examples)
  - fully documented API
- BES_SOLVE: a solver for explicit BES

What is new in CADP 2010?

Specification: support for LOTOS

- LOTOS (ISO standard 8807):
  - Types/functions: algebraic data types
  - Processes: process algebra based on CCS and CSP
- Tools: CAESAR, CAESAR.ADT, CAESAR.OPEN, etc.
- New features:
  - 64-bit support (as for all tools of CADP 2010)
  - Structured types (tuples, unions, lists, trees, strings, sets, etc.) can be stored canonically using bounded hash tables
  - Enhanced data flow analysis for further state space reductions
  - Dynamically resizable state tables
  - Code specialization according to the amount of RAM

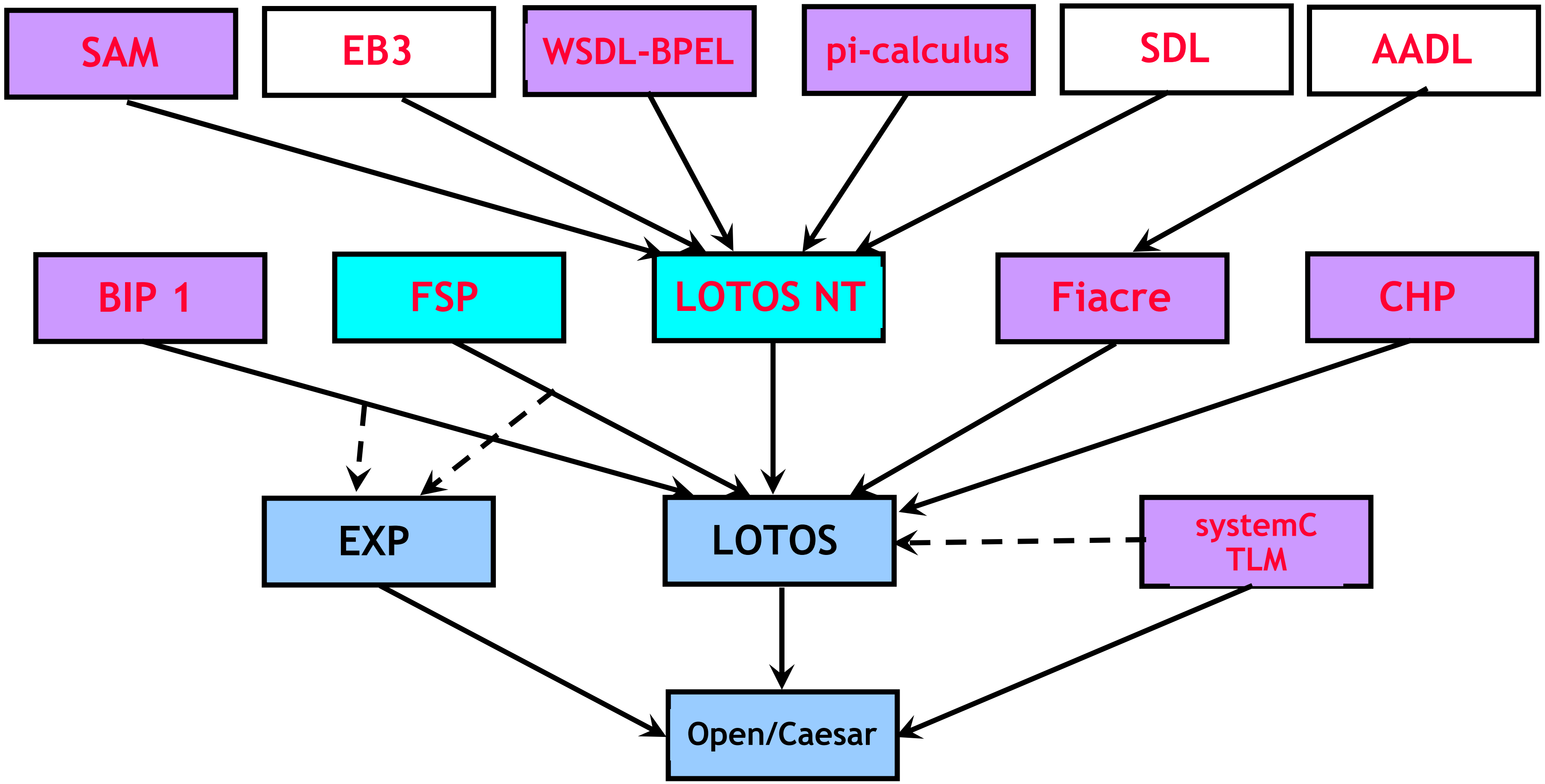
Specification: support for FSP

- FSP *(Finite State Processes)* [Magee-Kramer]
  - A simple, concise process calculus
  - Supported by the LTSA tool
- New tools: FSP2LOTOS and FSP.OPEN
  - Translation from FSP to LOTOS + EXP + SVL
  - On-the-fly state space generation for FSP
  - Benefits wrt LTSA:
    - Non-guarded process recursion is handled
    - Larger LTSs can be generated (64-bit support)
    - Easy interfacing with all other CADP tools

Specification: support for LOTOS NT

- Goal: replace LOTOS with a better language
- LOTOS NT:
  - easier to learn than LOTOS
  - more concise than LOTOS
  - coherent synthesis of LOTOS, ESTELLE, SDL, and Promela
- Key ideas:
  - types and functions: functional languages (first-order only)
  - processes: process algebras
  - with imperative-style syntax to be close to mainstream languages
- New tools: LPP, LNT2LOTOS, LNT.OPEN
  - Translation from LOTOS NT to LOTOS
  - On-the-fly state space generation for LOTOS NT

Specification: other languages



Model checking: Evaluator 3.6

- An on-the-fly model checker for μ-calculus built-on top of Open/Caesar and Caesar_Solve library for BES
- Automatic generation of diagnostics (LTS fragments: sequences, trees, or graphs with cycles)
- Libraries of standard property patterns

Formula language:

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

Model checking: Evaluator 4.0

- An on-the-fly model checker for the MCL language (more powerful than the language of Evaluator 3.6)
- Supports temporal formula with data (LTS actions contain typed values)
- Based on PBES (Parameterized Boolean Equation Systems) rather than BES
- Reasonable model checking complexity (linear-time for formulas without data)

MCL (*Model Checking Language*)

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

Equivalence checking

- Minimizing and comparing LTSs
- Bisimulations relations: strong, branching…
- Compositional generation of large LTSs
- Tools: EXP.OPEN, PROJECTOR, REDUCTOR, SVL
- (Almost) new tool: BISIMULATOR 1.1
  - on-the-fly comparison of two LTSs
  - 7 equivalence relations supported with their preorders
  - diagnostics (common LTS fragment before divergences)
- New tool: BCG_MIN 2.0
  - new signature-based minization algorithm
  - support for large LTSs with 10^9 – 10^{10} states
- New tool: PROJECTOR 3.0
  - enhanced algorithm (3 times faster, 1.5 times less memory)

Distributed verification

- Exploit NoWs, clusters and grids
- Cumulate CPU and RAM across the network
- Step 1: distributed LTS exploration
  - The LTS is built and partitioned on the fly
  - Fragments are sets of states and transitions
  - PBG = LTS consisting of remote graph fragments
  - Tools: DISTRIBUTOR and BCG_MERGE
  - New tools: PBG_CP, PBG_INFO, PBG_MV, PBG_OPEN, PBG_RM
- Step 2: distributed BES resolution
  - The BES is built, partitioned, and solved on the fly
  - Fragments are sets of boolean variables and logical dependencies between variables
  - New tool: distributed BES solver available in BES_SOLVE

Performance evaluation

- Combining functional verification and performance evaluation
- Based on Hermanns' *Interactive Markov Chains* (IMCs)
- Step 1: Compositional generation of IMCs
  - Tools: BCG_MIN, DETERMINATOR, EXP.OPEN, SVL
- Step 2: Markov solvers for IMCs
  - Tools: BCG_STEADY and BCG_TRANSIENT
- Step 3: Markov simulation (for big models)
  - New Tool: CUNCTATOR
    - on-the-fly simulator for IMCs
    - on-the-fly hiding/renaming of labels
    - various scheduling strategies
    - save/restore features

Conclusion

Conclusion

- CADP: a bridge between theory and practice
  - 139 case-studies performed using CADP
  - 57 research tools built using CADP
  - Forum with 160 users and ~1100 messages
- CADP 2010: a significant development effort
  - Comprehensive tool set: ~50 tools
  - Many architectures supported (full 64-bit support)
    - Processors: Itanium, PowerPC, Sparc, x86, x64
    - Operating systems: Linux, MacOS X, Solaris, Windows
    - Compilers: gcc3, gcc4, Intel C, Sun C
  - Large documentation
  - Significant testing effort (+ Contributor tool)