Advanced Modeling and Verification Techniques Applied to a Cluster File System

Charles Pecheur

RIACS / NASA Ames

pecheur@ptolemy.arc.nasa.gov

Work performed at INRIA Rhône-Alpes (F)

Introduction

Show two advanced formal verification techniques...

- Compositional model generation
- Extensible temporal logic verification

... on a real-size application.

- CFS distributed file system

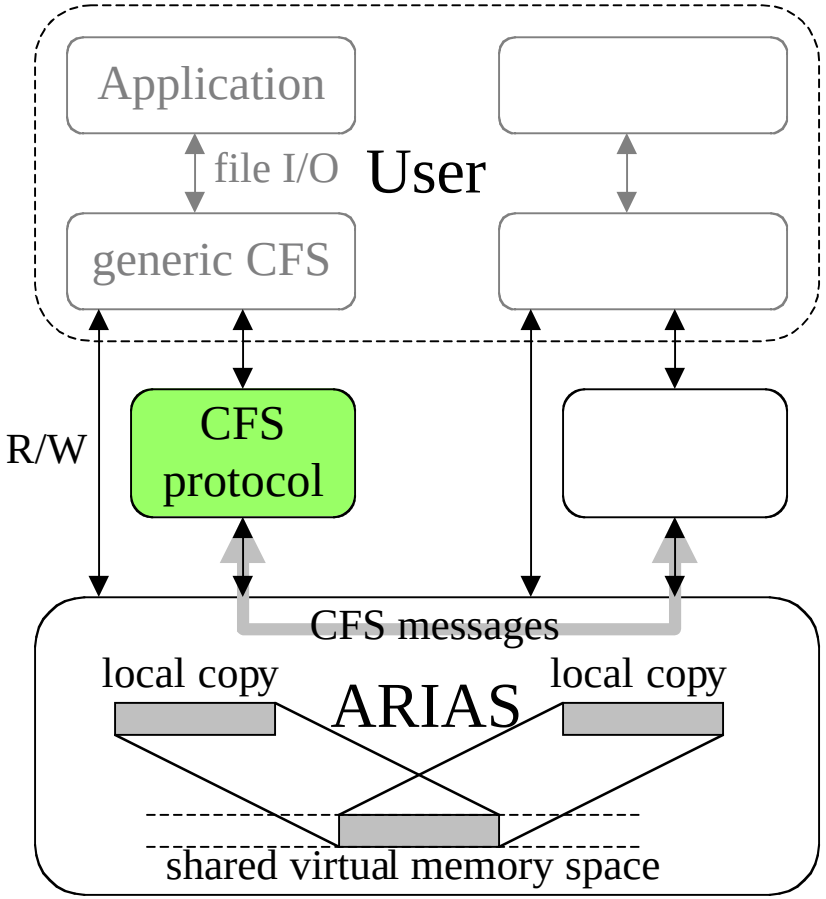
Cluster File System (CFS)

- Distributed file system
- On top of ARIAS:
  shared memory architecture
- From Bull, in AIX
- Dynamic master/slave
- CFS Protocol calls:

  BeginWrite/EndWrite
  BeginRead (no EndRead)
- CFS messages:

  Read/Write Req/Ans, Invalidate

  NB: causes memory updates



LOTOS

- Formal Specification Language (ISO 8807)
  - Control: process algebra (CCS, CSP)
  - Data: algebraic data types (ACT ONE)
    + syntax extensions (APERO)

"the spec(ification)"

- Semantics: *labeled transition system* (LTS)
  = state graph with labels on edges

"the model"

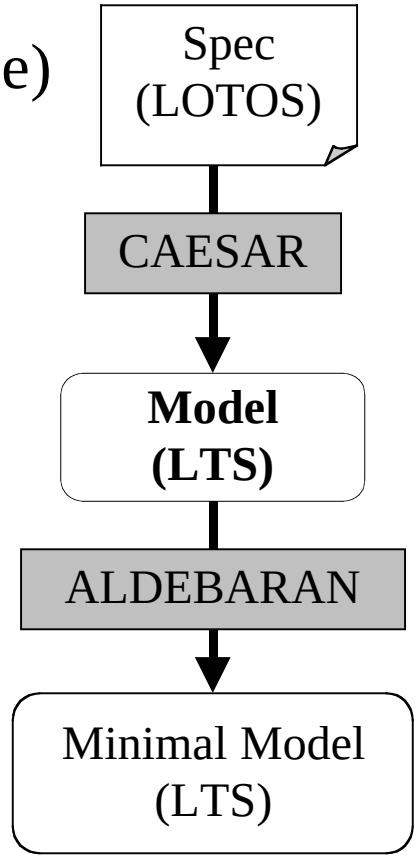
CAESAR/ALDEBARAN Toolset

- From INRIA & VERIMAG (Grenoble)

CAESAR:
Generate models from specs

ALDEBARAN:
Minimize and compare models

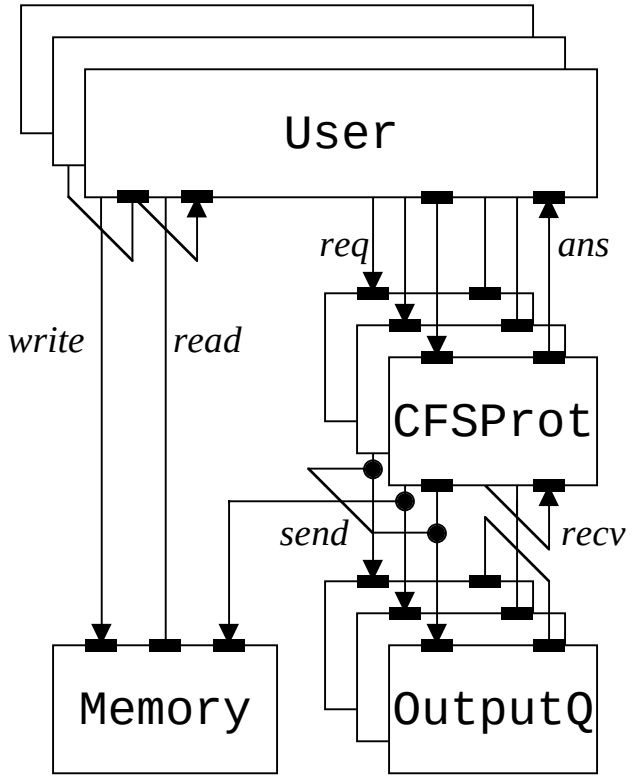
+ GUI + simulator + temporal logic
+ on-the-fly + other formalisms + ...



Specification of CFS

- Describes:
  - CFS protocol
  - ARIAS communications
  - ARIAS memory mgt
  - Users (= environment)

- For model checking
  => keep it small:
  - One memory block
  - 3 sites
  - 3 x one-slot queue

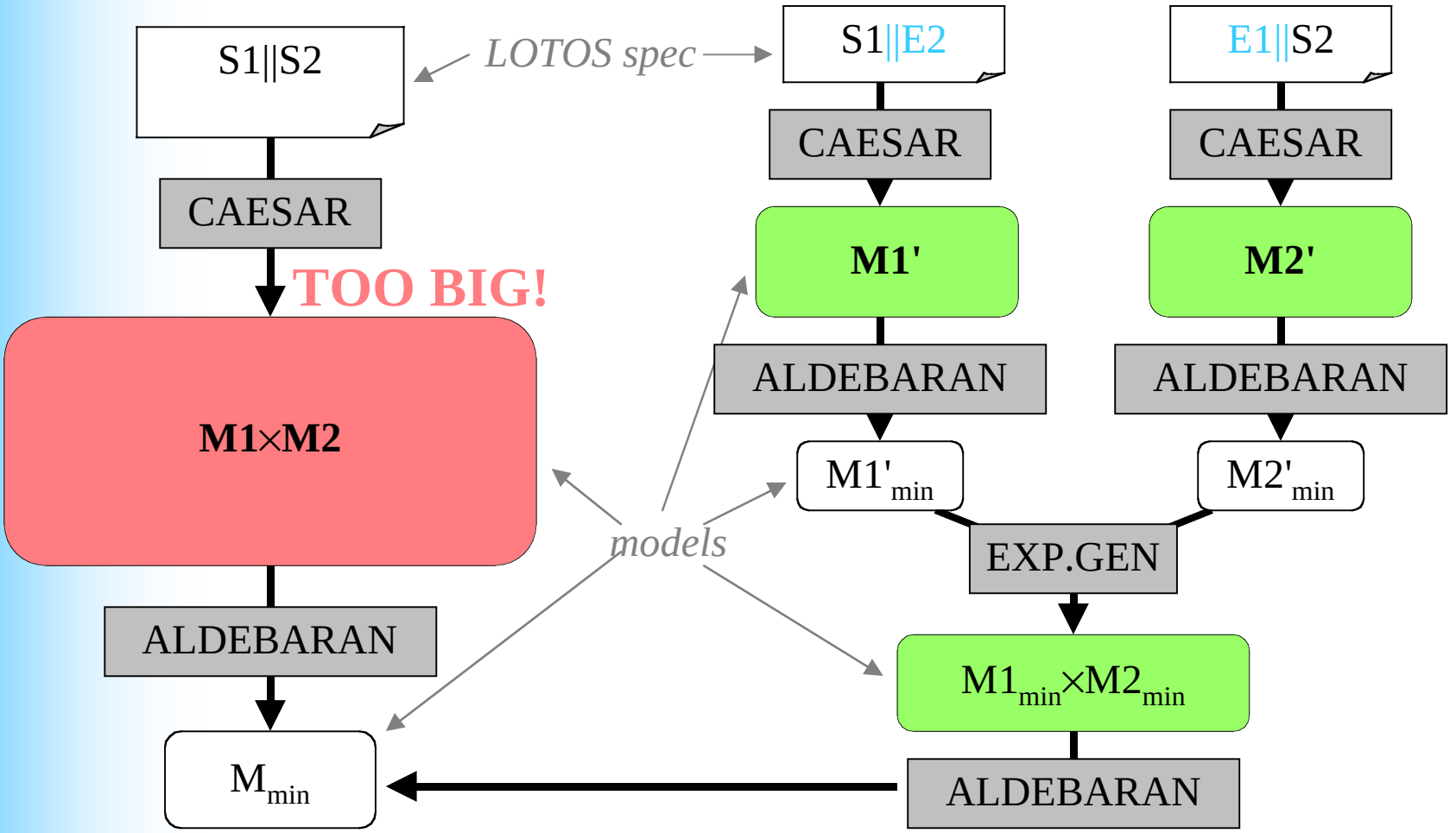


Specification of User

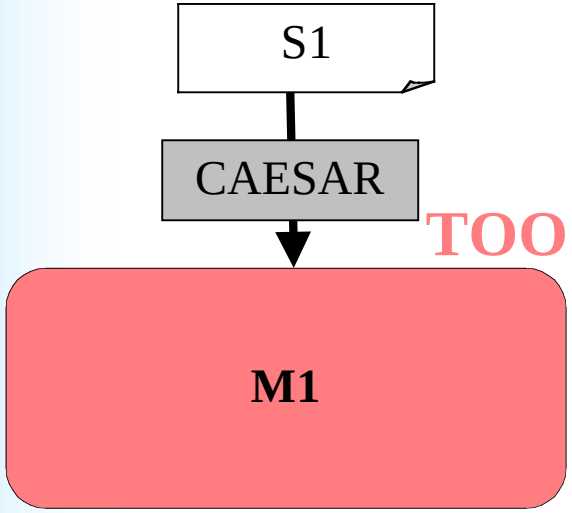
- Enforces correct use of CFS calls
- Otherwise very general – not a test scenario

```
repeat forever     -- NB: this is NOT LOTOS!
  choose either {
      CFSCall(Read);
      read;
  } or {
      CFSCall(BeginWrite);
      write;
      CFSCall(EndWrite);
  } endchoose
endrepeat
```

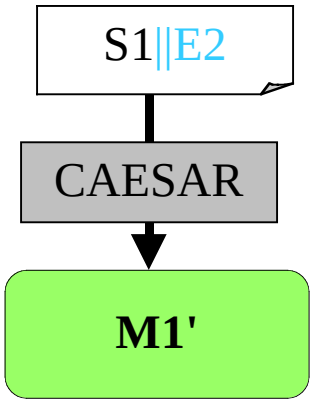
Compositional Model Generation



Environments for Composition



TOO BIG!



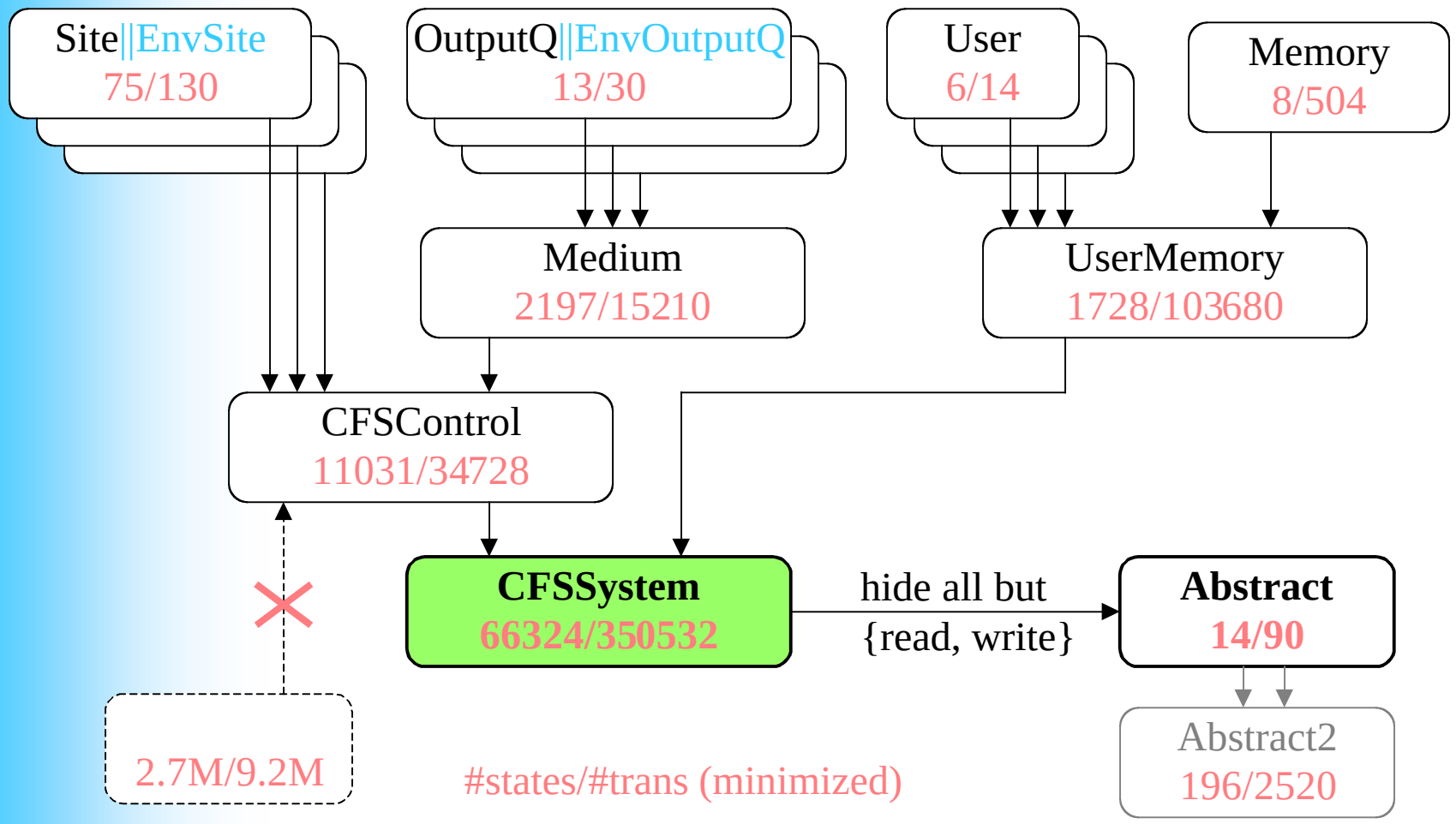
S1, S2 tightly coupled

S1 alone =>
many irrelevant situations

E2 =
conservative approximation
of S2 as seen from S1

(S1||E2)||S2 ≈ S1||S2

Generating a Model of CFS



XTL

- Temporal Logic Checker for LTS
- Programmable => different logics as libraries
- I used ACTL (branching-time logic with actions)
- Example: EX_A F = "F can be reached after A"

```
def EX_A (A : labelset, F : stateset) : stateset =
  { S : state where
    exists T : edge among out(S) in
      (label(T) among A) and
      (target(T) among F)
    end_exists
  }
end_def
```

types and functions
to walk through the model

Printing Explanations in XTL

- Default ACTL library: backward search
  - + Checks if φ holds in M in O(|M|.|φ|)
  - – No explanation if φ does not hold
- New ACTL library: forward search

  Uses macros to simulate 2nd order functions
  - + Prints trace (when it makes sense)
  - + Can bind data (e.g. *send ?x => eventually receive !x*)
  - – Checks if φ holds in M in O($|M|^{|\varphi|}$)

  Worst case is costly, but linear for simple formulas and explanations are very useful.

Properties of CFS

- General Principle:
  1. Verify on `Abstract` (very small)
  2. If needed, generate diagnostic on `CFSSystem` (big)
- No given formal properties to check => explore!
- Simple Properties:
  - No global deadlock (OK)
  - No local deadlock (OK)
  - Deterministic (FAIL because of abstraction)
  - Values propagate (OK with fairness)

Consistency Properties

- Atomic coherency

  All sites see the same history at the same time
  = Distributed data is equivalent to local data

  OK for write/write, FAIL for others

  Expected considering loose sync. on read

- Sequential consistency

  All sites see the same history, possibly with delays

  OK for one block

  FAIL for two blocks (using `Abstract2`)

Conclusions

- Approx. 4 man.month including:
  - Writing the specification
  - Doing compositional generation
  - Developing the new XTL library
  - Doing verification
- Results
  - Verification for program understanding
  - LOTOS specification & tools => basis for future experiments
- Tools
  - Can address real-size problems (with tools, CPU & expertise)
  - XTL: flexible, needs on-the-fly capabilities

Atomic vs. Sequential Consistency