Variations on an Alloy-centric Tool-Chain in Verifying a Journaled File System Model

Authors: Miguel A. Ferreira and
José N. Oliveira
Presented by: Carlos Eduardo Silva

Introduction

- Grand Challenge(GC)
- Verified File System (VFS):
  - Verified subset of POSIX suitable for flash-memory hardware with strict fault-tolerant requirements to be used by forthcoming NASA's JPL missions.

Introduction

- Tool-chain:
  - Promote incremental development and verification of specifications;
  - Be agile enough to encourage users to verify even the smallest unit of their specifications;
  - Be capable of producing immediate feedback on the problems unveiled;
  - Be capable of performing fully automated proofs;
  - Be amenable to automatic code generation.
- Presented in M. A. Ferreira and J.N. Oliveira - An integrated formal methods tool-chain and its application in verifying a file system model. In Oliveira and Woodcock.
- This paper contribution is towards catering for refinement proofs.

Introduction

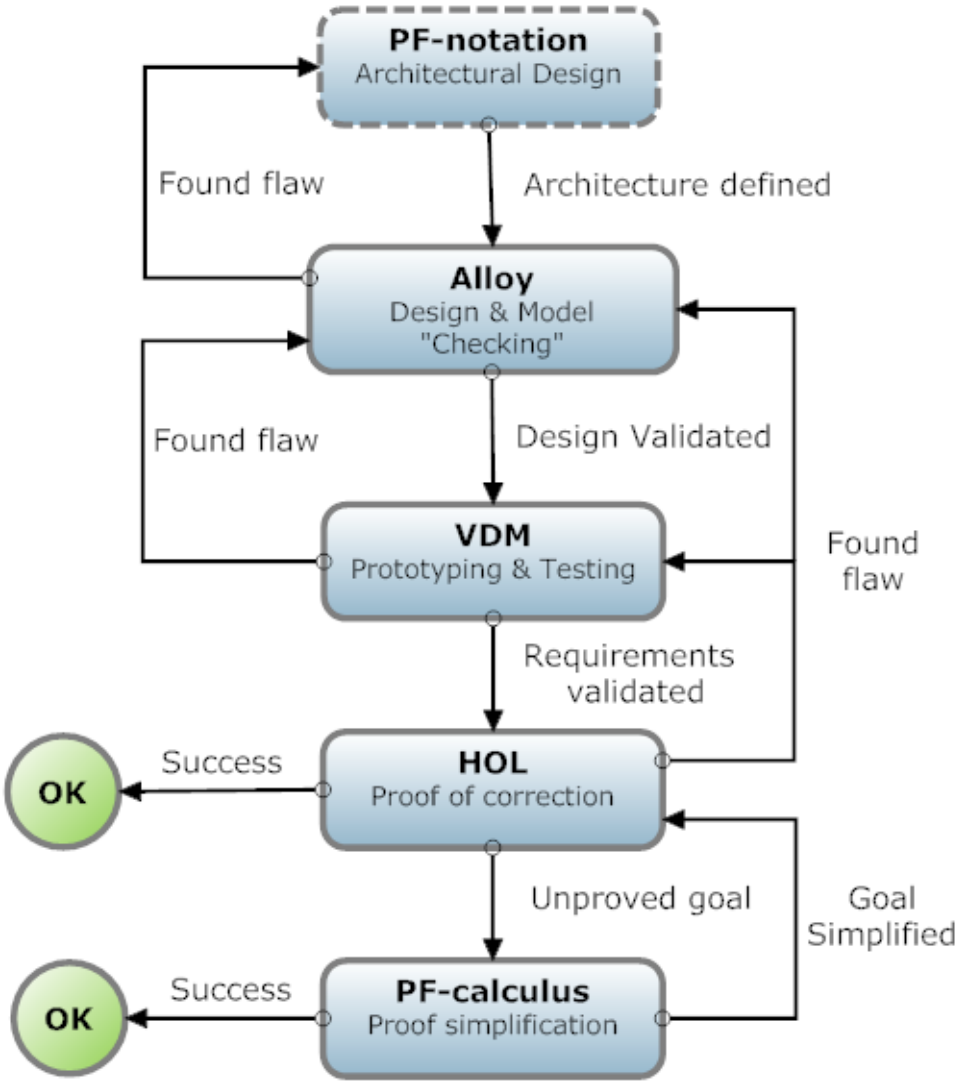
- Refinement of:
  - The file store model
    - Addition of a journaling mechanism (higher performance and reliability)
  - The delete operation

Tool-Chain

- Relational algebra
  - Pointfree notation
  - Pen and paper proof strategy
- Alloy
  - Model Checker to generate uninterpreted, unexpected counter examples
- VDMTools
  - Interpreter for semantically meaningful animation and testing
- HOL
  - Theorem prover (overture proof obligations system)

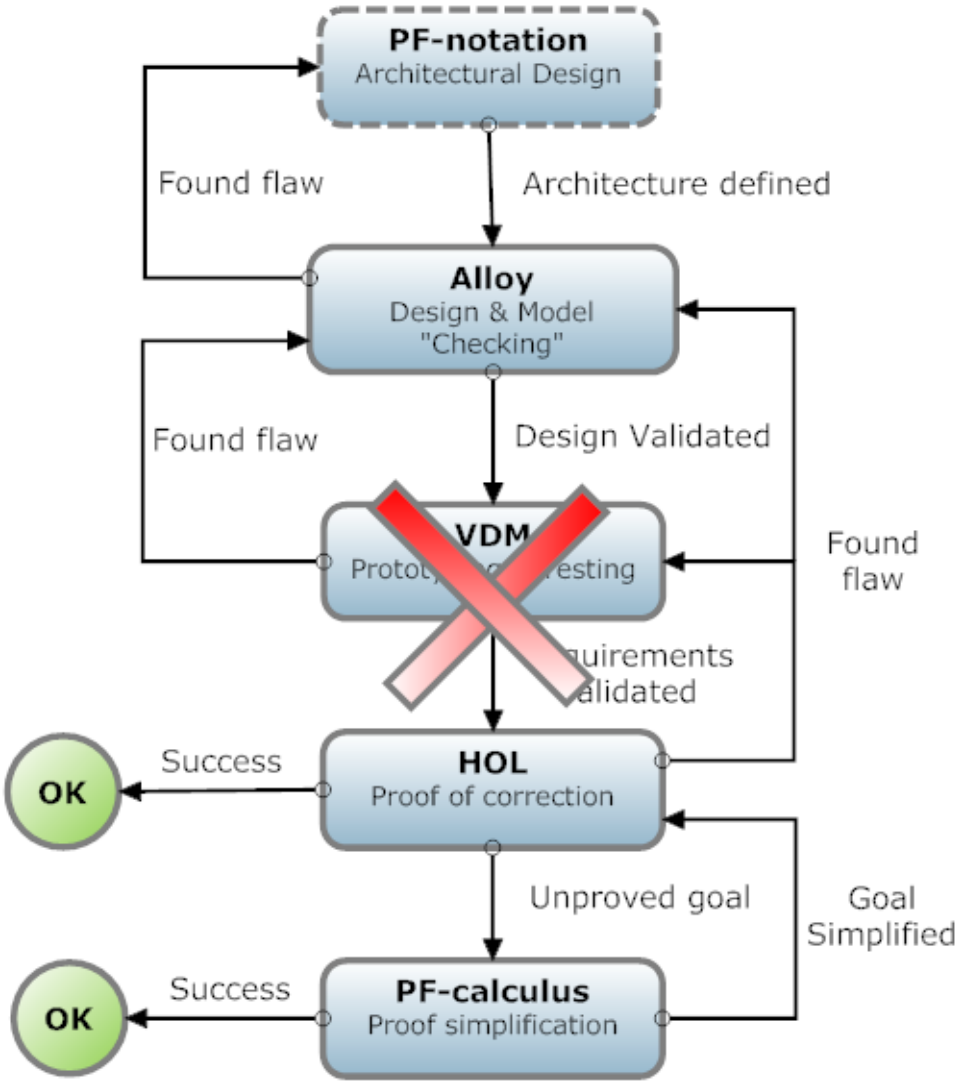
Tool-Chain

- Satisfiability



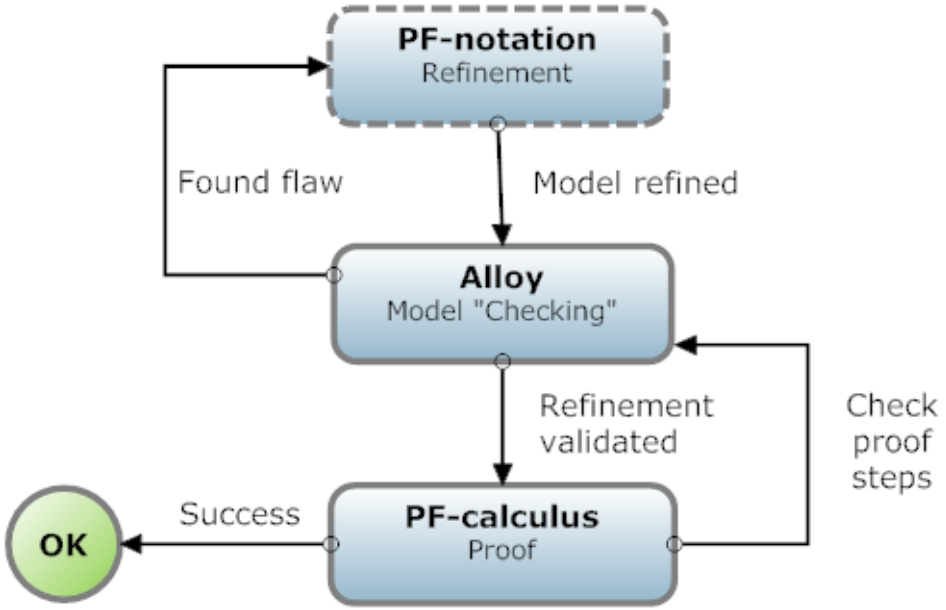
Tool-Chain

- Refinement Proofs
  - No subjective user requirements
  - Rendering execution and animation unnecessary
- Remove VDM++
  - Problems sewing Alloy to HOL

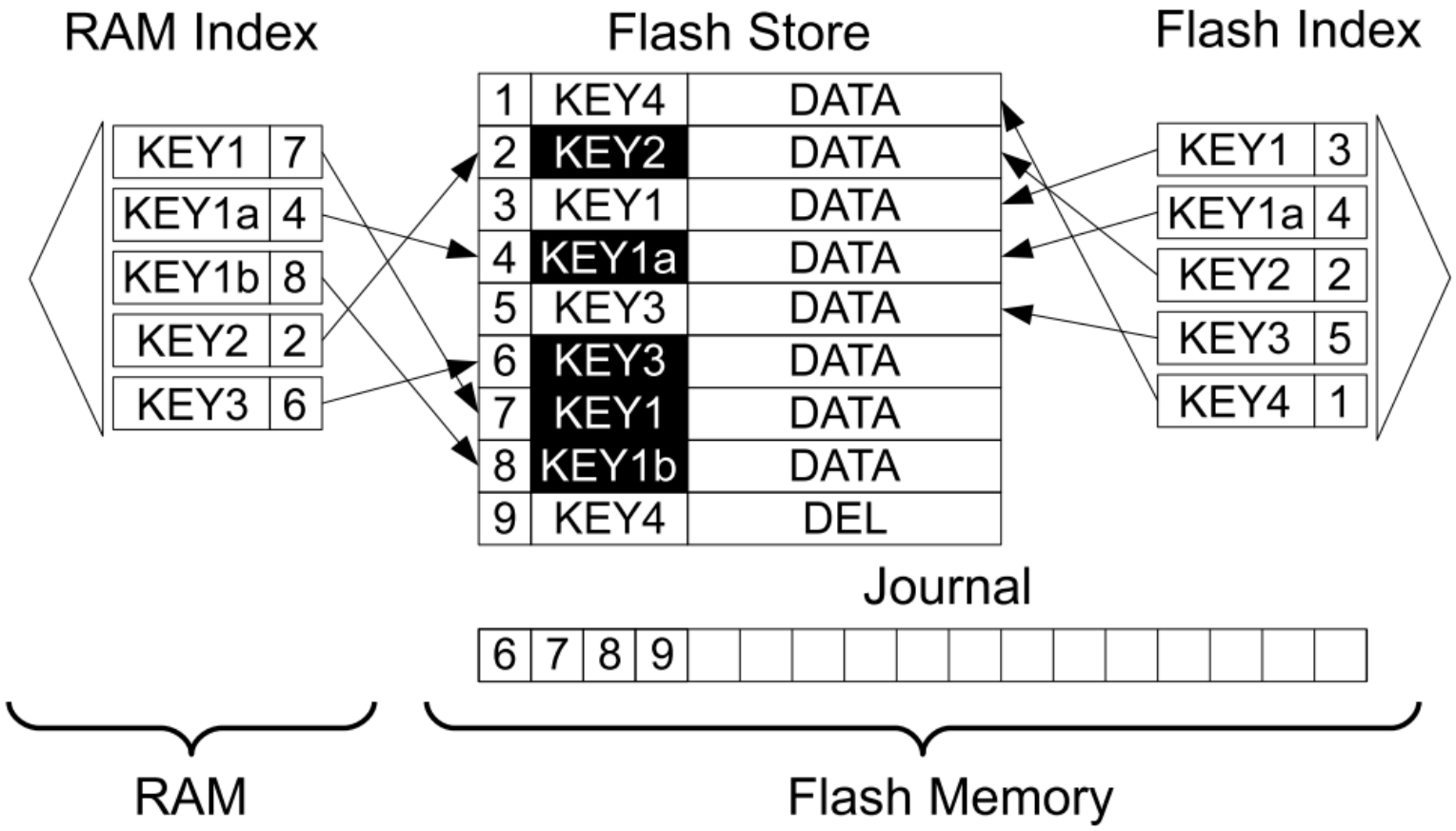


Tool-Chain

- Simplified Version
  - PF-calculation less error prone
  - Difficult steps (lemmas) are model-checked in Alloy



Journaled file system

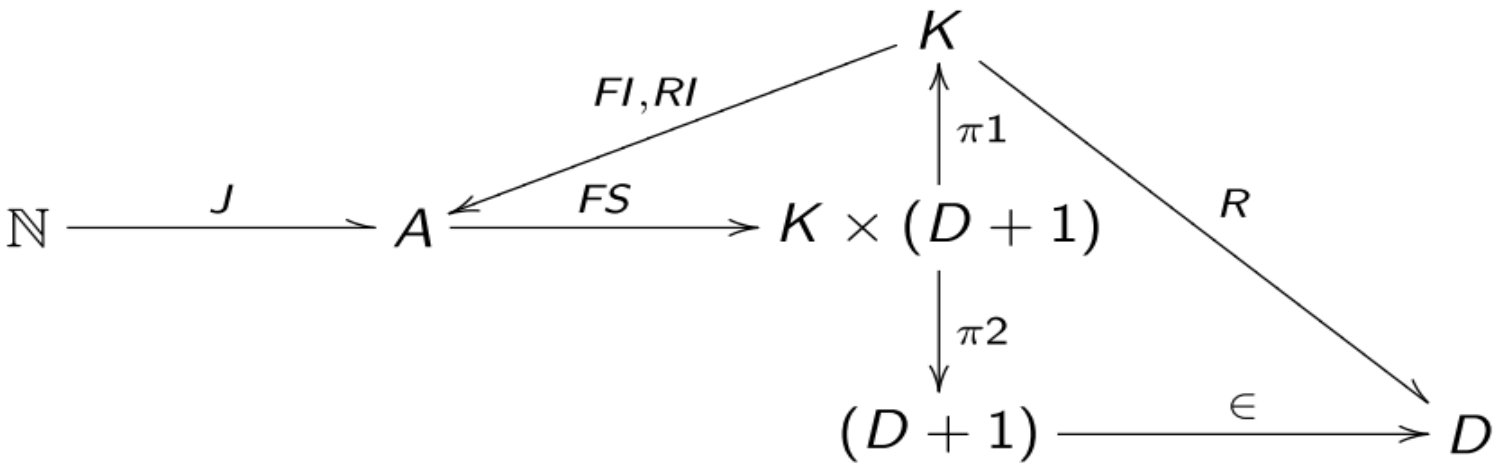


Journaled file system

- Backlog of the operations (journal)
  - Rebuild meta-data as before the fault
  - Added complexity to the model and invariant
- File Store
  - Finite and univocal (simple) relation between paths and files

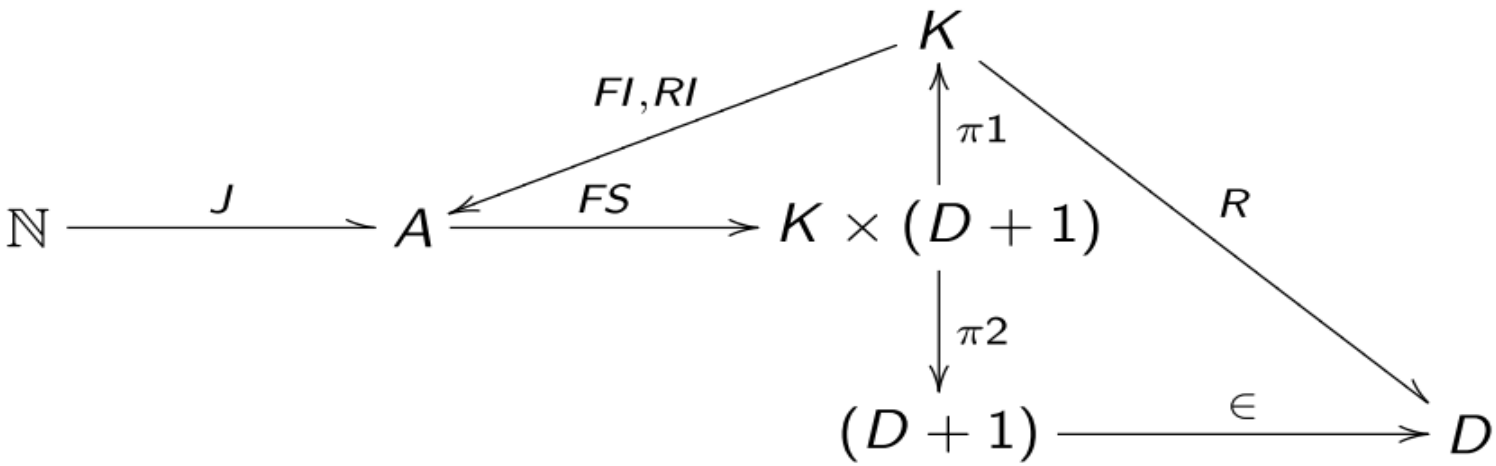
  - fileStore is depicted as relation R
  - Path is defined by K
  - File is defined by D

Journaled file system



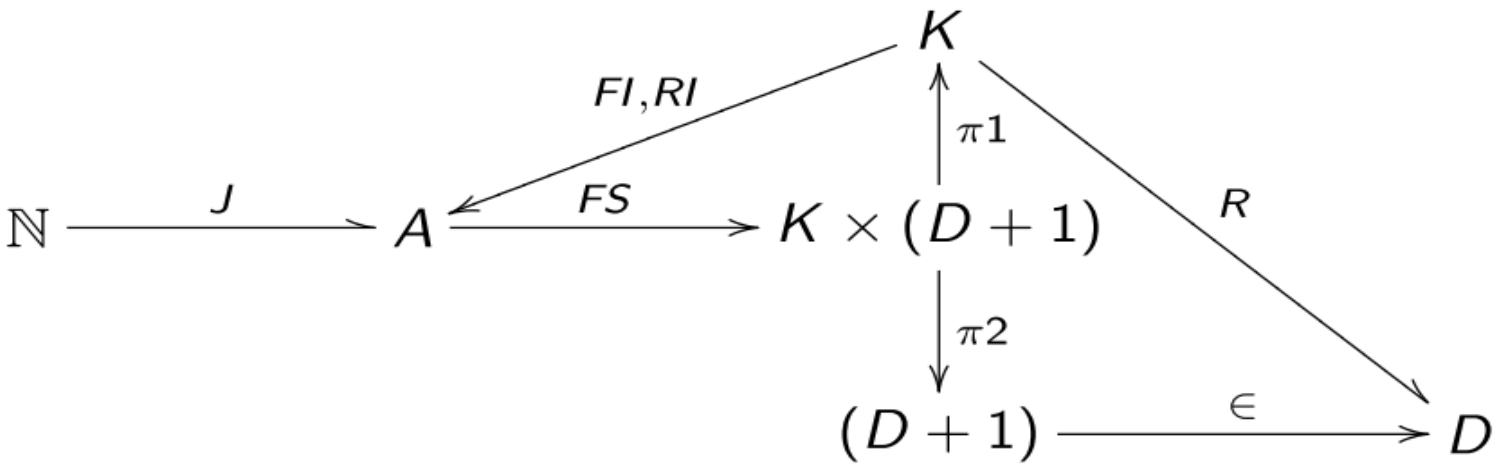
- Data Structures
  - FI (Flash Index)
  - J (Journal – Sequence of addresses)
  - RI (RAM Index)
  - FS (Flash Store)

Journaled file system



- A – Memory addresses
- D+1 – Maybe data or DEL mark (1)
- K×(D+1) – Pairs of keys and maybe values

Abstraction Invariant



$$R \quad = \quad af(FS, FI, RI, J)$$

where abstraction function is

$$af(FS, FI, RI, J) \quad \triangleq \quad (active\ FS) \cdot RI$$

for

$$A \xrightarrow{active\ FS} D \quad \triangleq \quad \in \cdot \pi_2 \cdot FS$$

Delete Operation

Post-FS_DeleteFileDir_Fstore(S,R',R) $\triangleq$ R' = R · $\Phi_{(\notin S)}$

S – paths to be deleted
$\Phi_{(\notin S)}$ - correflexive relation associated to predicate $x \notin S$

Not covering:

- Wear levelling
- Power loss recovery

Conclusions

- Refinement steps
  - Can be carried out in a shorter, reduced life-cycle
- Added model complexity
  - Non functional requirements
    - Wear levelling
    - Power-loss recovery
- Cost to be paid by simplification
  - Seamless integration of Alloy models with PF math
- New combinator *thinning* (↾)
  - Converts a given relation R into a smaller, simple relation by looking at particular elements of its range relative to some ordering.

Conclusions

- Sucessfully proved the refinement of an abstract file store model into its journaled implementation
  - No need for mechanical theorem proving
  - Using the simplified tool-chain
  - Use of combinators was central