

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.

VERIFYING INTEL'S FLASH FILE SYSTEM CORE
Miguel Ferreira and Samuel Silva
University of Minho
(pg10961.pg11034)@alunos.uminho.pt

Deep Space lost contact with Spirit on 21 Jan 2004, just 17 days after landing.

Initially thought to be due to thunderstorm over Australia.

Spirit transmitted an empty message and missed another communication session.

After two days controllers were surprised to receive a relay of data from Spirit.

Spirit didn't perform any scientific activities for 10 days.

This was the most serious anomaly in four-year mission.

Fault caused by Spirit's FLASH memory subsystem

Why formal methods?
Software bugs cost millions of dollars.

What we can do?
Build abstract models (VDM).
Gain confidence on models (Alloy).
Proof correctness (HOL & PF-Transform).

Acknowledgments:
Thanks to José N. Oliveira for his valuable guidance and contribution on Point-Flow Transformation.
Thanks to Sander Vermeulen for VDM to HOL translator support.
Thanks to Peter Gorm Larsen for VDMtools support.



Logos for Intel and NASA are visible in the bottom left corner. Two red icons, a star and a hexagon, are in the bottom right corner.

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

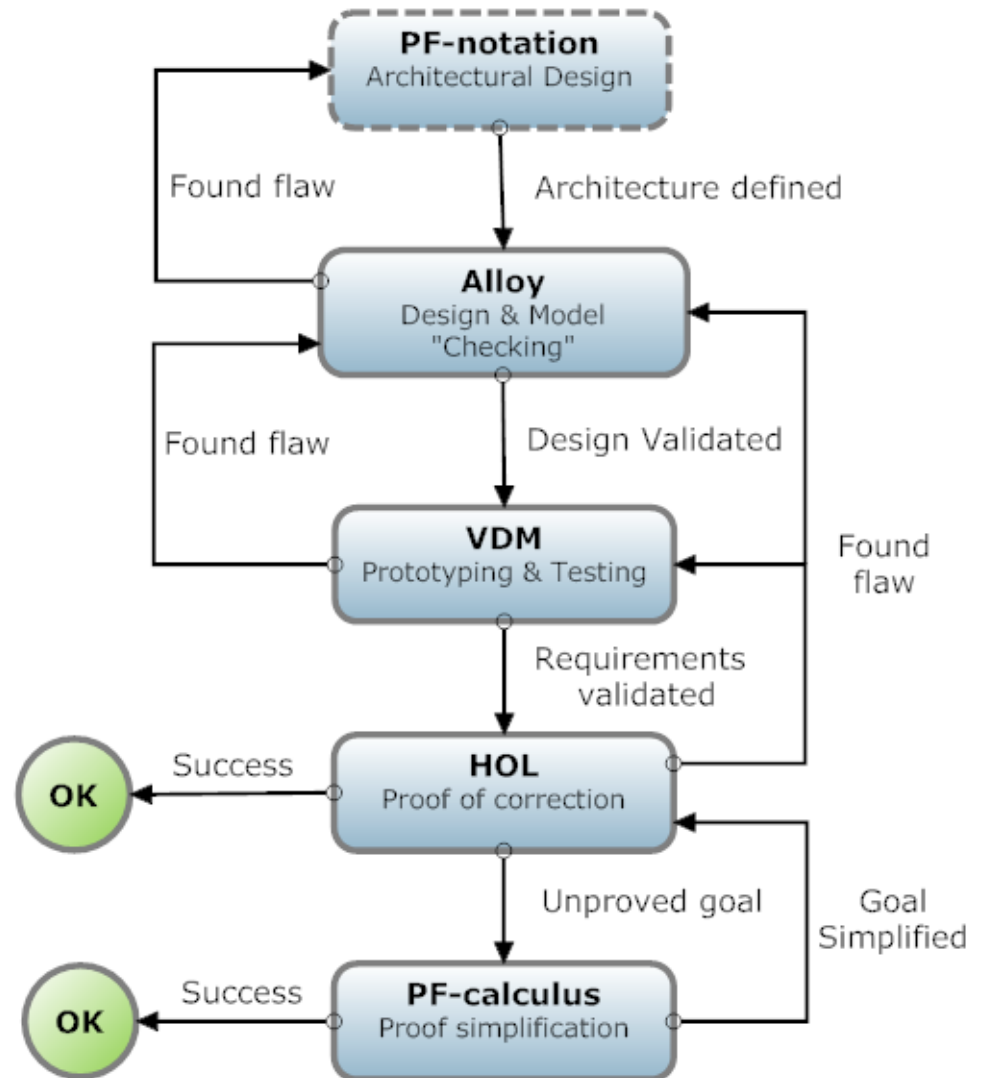
- Refinement of:
 - The file store model
 - Addition of a journaling mechanism (higher performance and reliability)
 - The delete operation

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

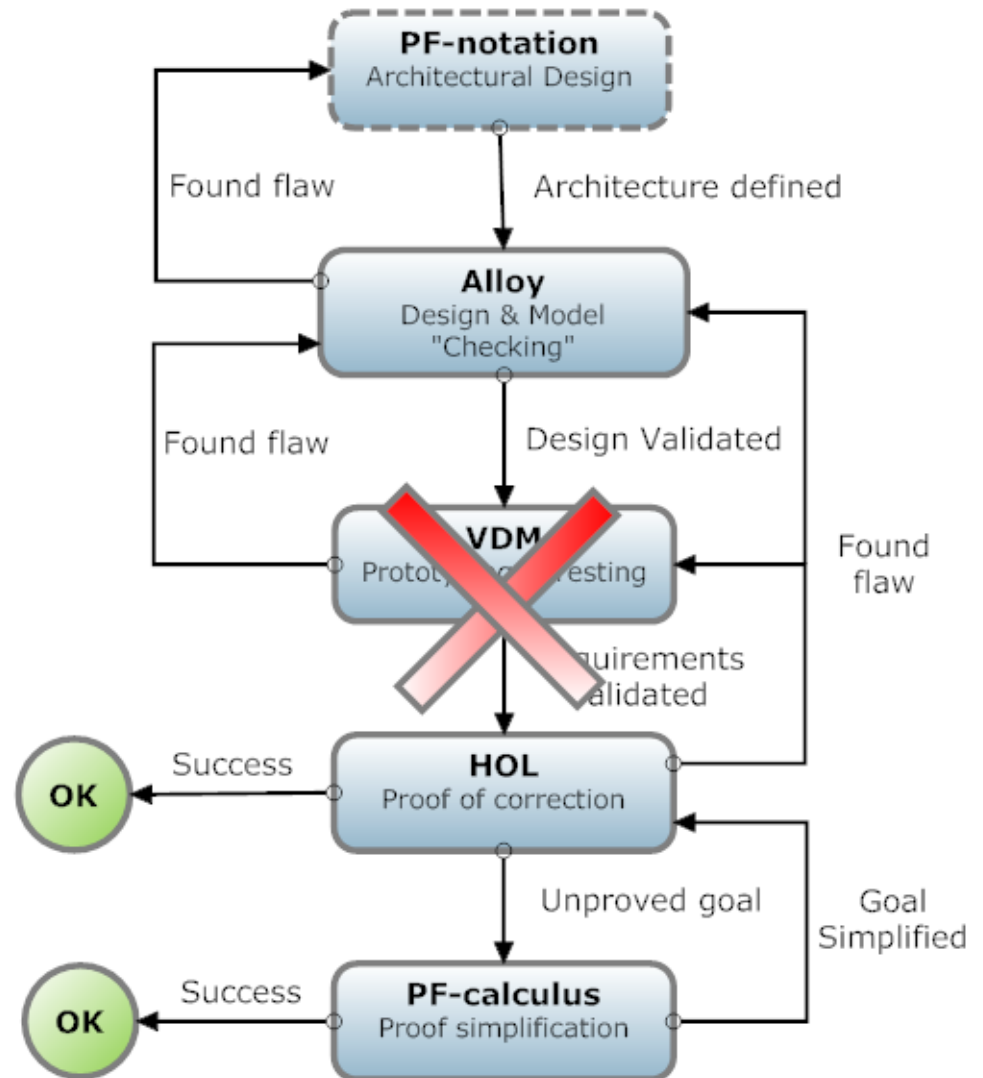
- Satisfiability



Tool-Chain

○ Refinement Proofs

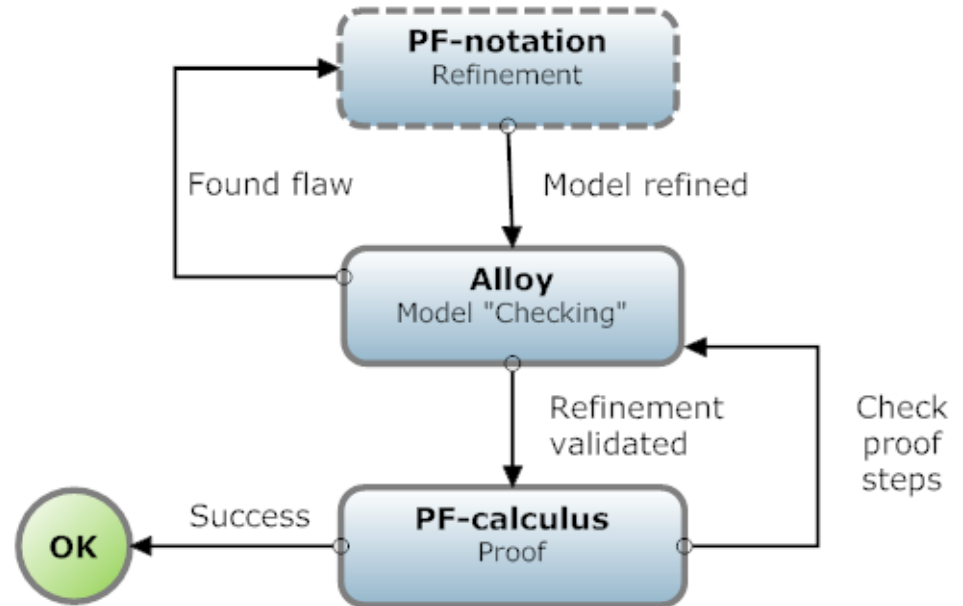
- No subjective user requirements
- Rendering execution and animation unnecessary
- Remove VDM++
 - Problems sewing Alloy to HOL



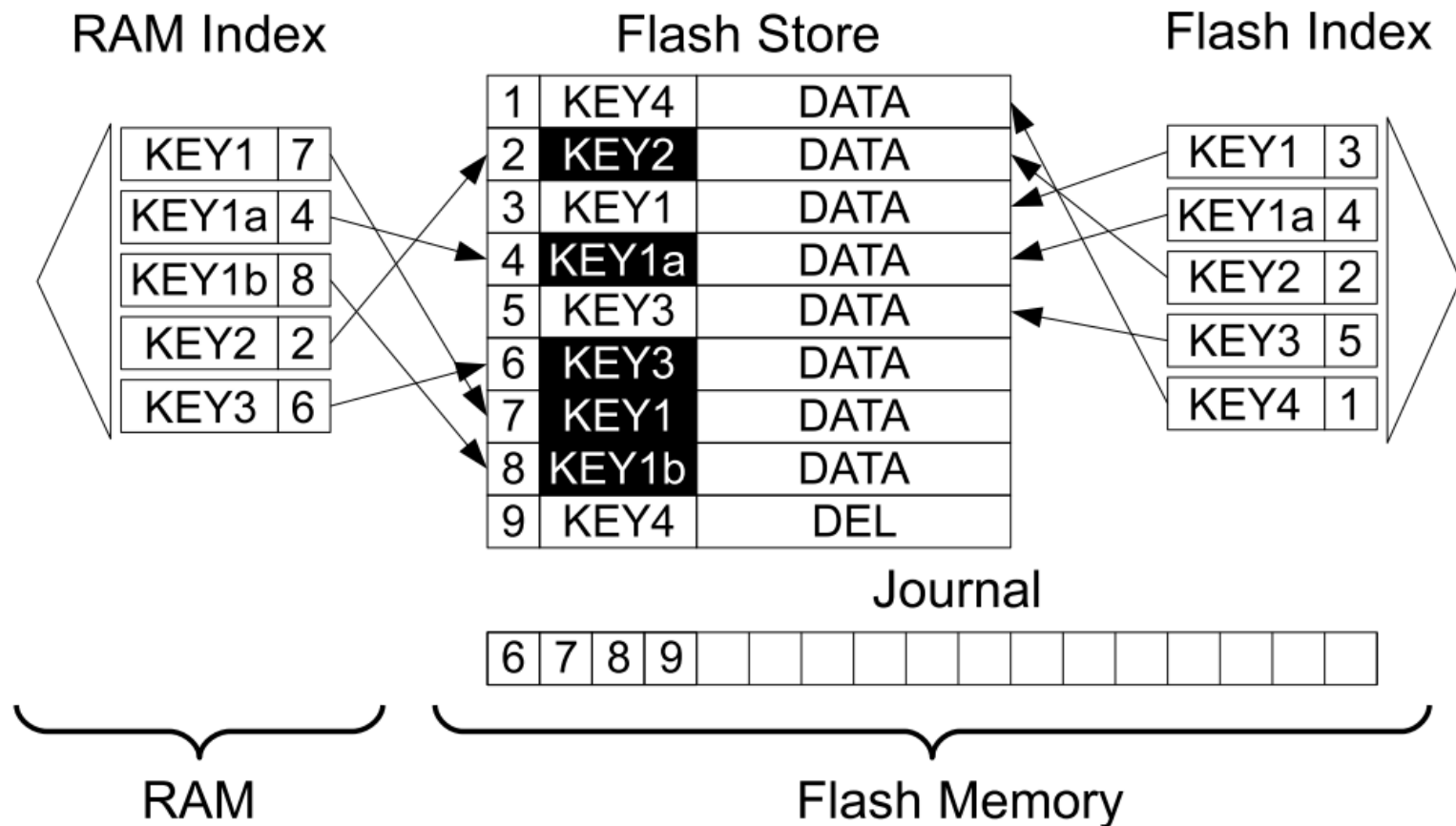
Tool-Chain

○ Simplified Version

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



Journalled file system



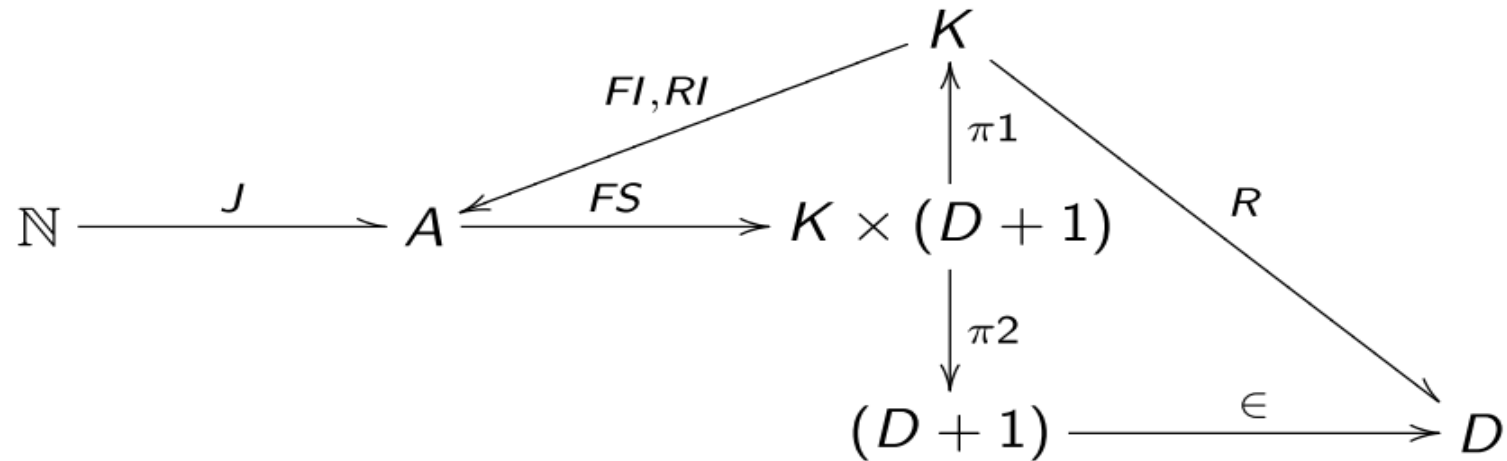
Journalled file system

- 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**

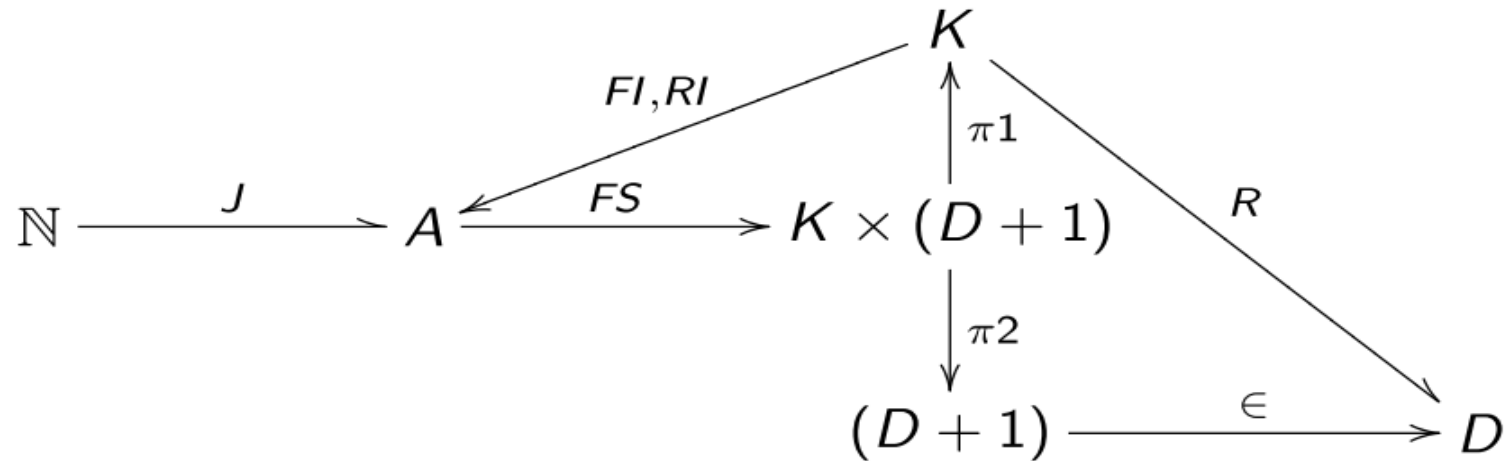
Journalled file system



- Data Structures

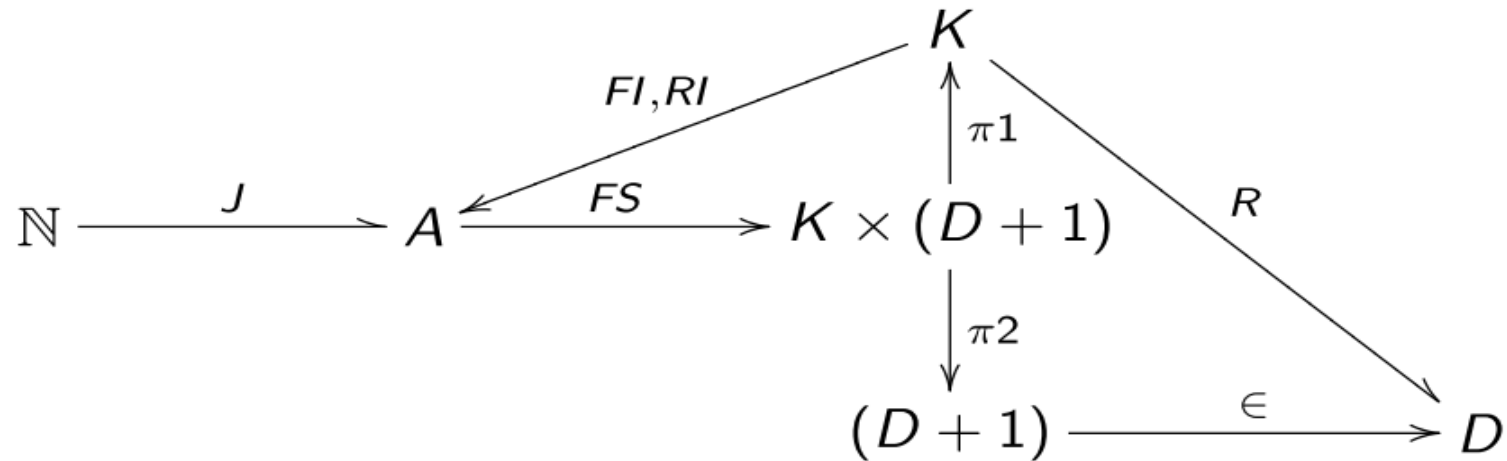
- **FI** (Flash Index)
- **J** (Journal – Sequence of addresses)
- **RI** (RAM Index)
- **FS** (Flash Store)

Journalled file system



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

Abstraction Invariant



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

where abstraction function is

$$af(FS, FI, RI, J) \triangleq (\text{active } FS) \cdot RI$$

for

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

Delete Operation

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

S – paths to be deleted

$\Phi_{(\notin S)}$ - correlexive 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* (\uparrow)
 - Converts a given relation R into a smaller, simple relation by looking at particular elements of its range relative to some ordering.

Conclusions

- Successfully 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