DATA REFINEMENT FOR WORKERS

(or Formal Methods for Casuals)

Theory ArrayHashMap_Impl

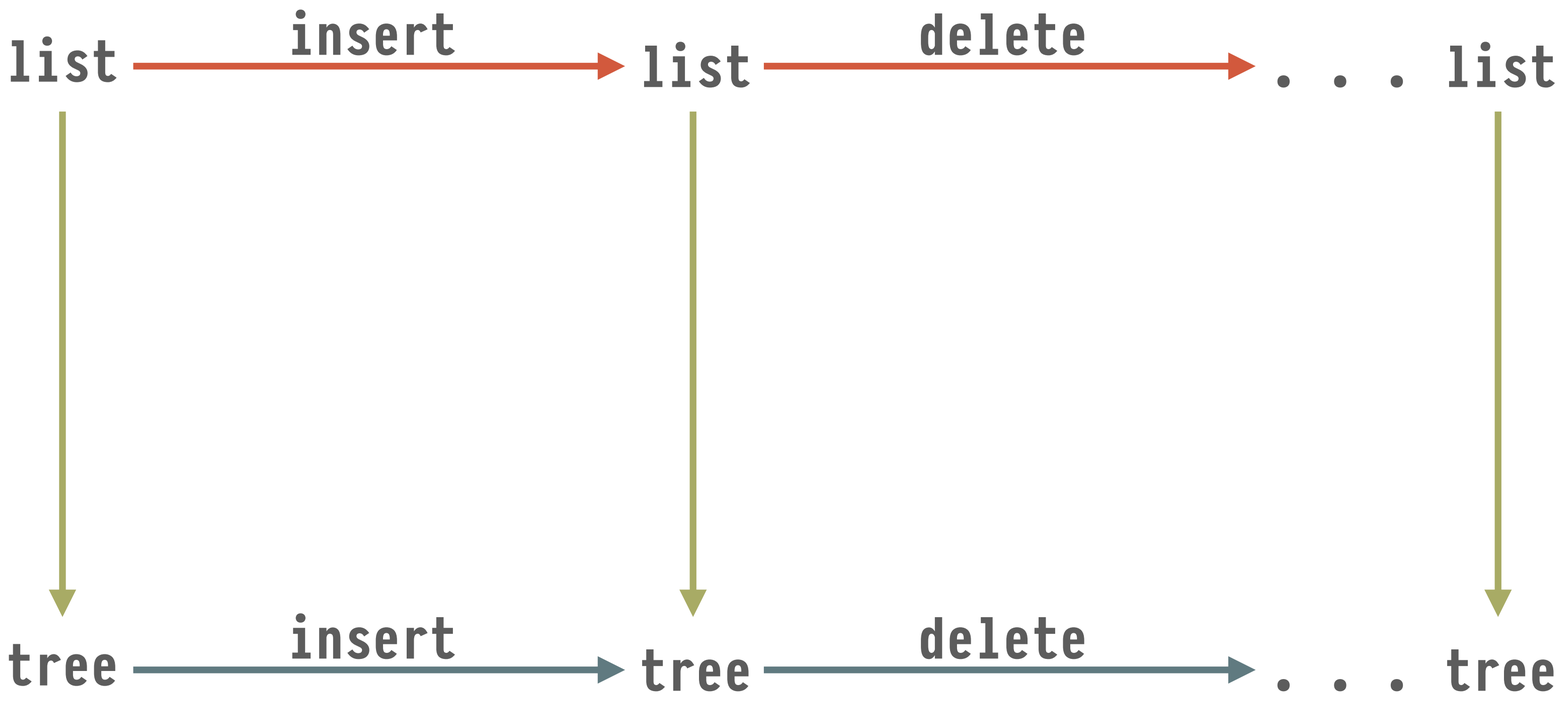
theory *ArrayHashMap_Impl*
imports *HashCode* *ListGA* *ListMapImpl* *Array_Iterator*

```
(*  Title:       Isabelle Collections Library
    Author:      Andreas Lochbihler <andreas dot lochbihler at kit.edu>
    Maintainer:  Andreas Lochbihler <andreas dot lochbihler at kit.edu>
*)
section {* \isaheader{Array-based hash map implementation} *}
theory ArrayHashMap_Impl imports
```

Branch: **master** ▾ **ghc** / libraries / base / Data / **List.hs**

jrraymond Data.List.isSubsequenceOf documentation clarification

18 contributors 


list
insert
list
delete
. . .
list
tree
insert
tree
delete
. . .
tree

What's out there?

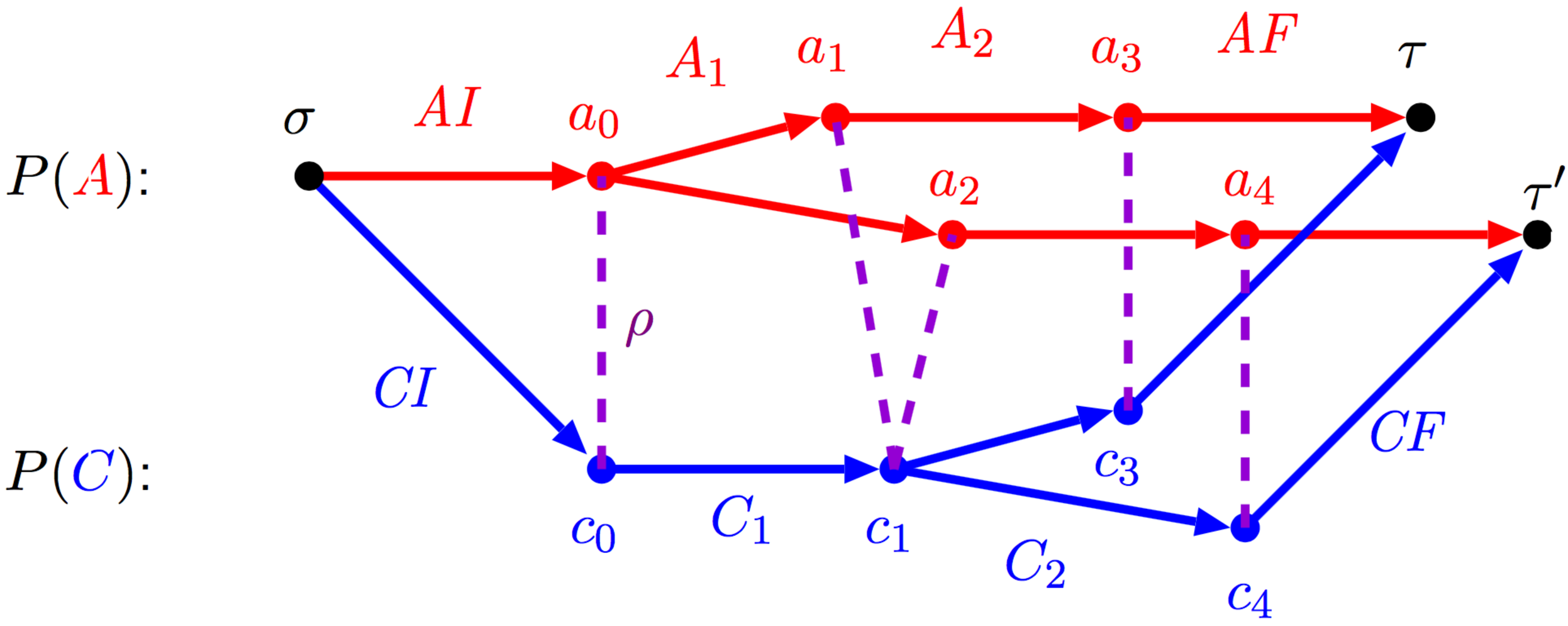
Numerous (formal) methods exist for writing specifications and refining those to implementations:

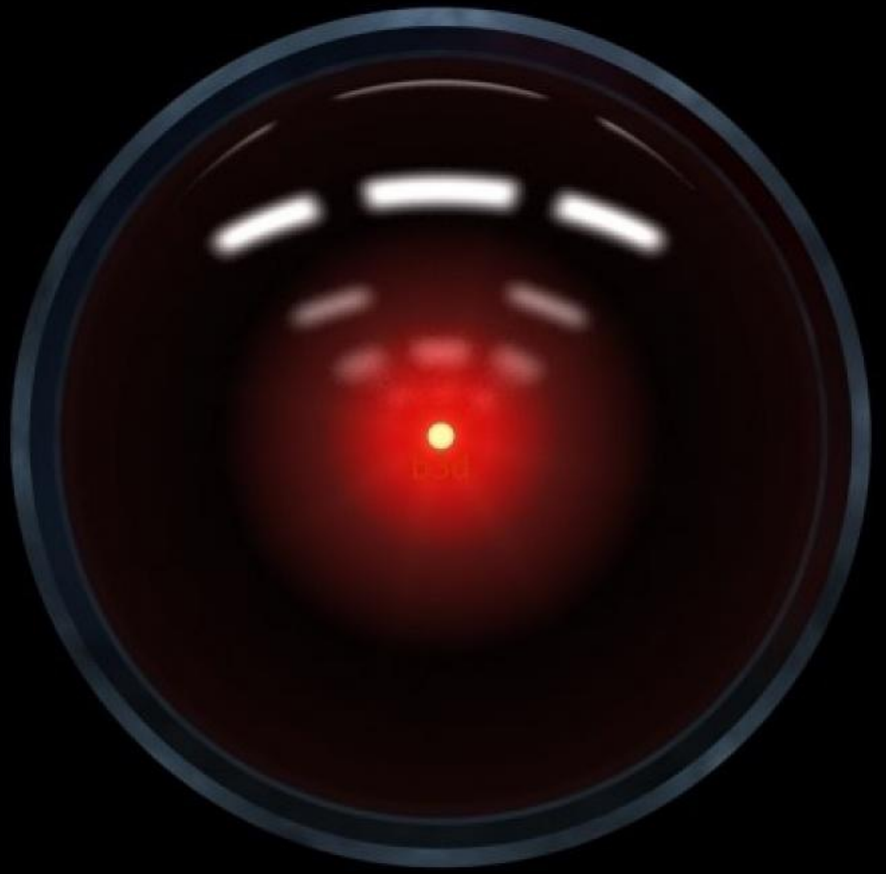
- VDM (Raise, Z, B)
- Reynolds' method
- Refinement Calculi of Back & von Wright, Gardiner & Morgan, Morris
- Hehner's method
- Abadi & Lamport's refinement mappings
- Lynch's possibilities mappings

major development technique: stepwise refinement

All these methods are proved to be related in the Data Refinement book by Kai Engelhardt and me.


THE GOOD
THE BAD
AND THE UGLY


$P(A)$:
$P(C)$:
σ
AI
a_0
A_1
a_1
A_2
a_3
AF
τ
a_2
a_4
τ'
ρ
CI
c_0
C_1
c_1
C_2
c_3
c_4
CF



$$x : \tau$$

Data refinement, the theory and methods:
http://www-verimag.imag.fr/PEOPLE/Nicolas.Halbwachs/SYNCHRON03/Slides/deroever.pdf

A mention of model-based testing for monadic programs in:
http://www.cse.chalmers.se/~rjmh/Papers/QuickCheckST.ps