Computer Mathematics, AI and Functional Programming

Moa Johansson
Chalmers University of Technology

Lambda Days
21 February 2019

CHALMERS
UNIVERSITY OF TECHNOLOGY

1950's – The Logic Theorist and the birth of AI

- Proved theorems from *Principia Mathematica.*

- Reasoning as search – **heuristics**.

- Invented a **programming language for list processing** (predecessor to LISP).

```
THE LOGIC THEORY MACHINE
A COMPLEX INFORMATION PROCESSING SYSTEM

by

Allen Newell and Herbert A. Simon

P-868

June 15, 1956
```

"**But, although the program of LT does not change as it accumulates experience in solving problems, learning does take place in one very important respect. The program stores the new theorems it proves, and these theorems are then available as building blocks for proofs of subsequent theorems.**"

1960's: Optimism!

"Within ten years a digital computer will discover and prove an important new mathematical theorem"

(Newell & Simon 1958)

- Important algorithms of the 60's:

  - **DPLL** for SAT-solving (Davis-Putnam-Logemann-Loveland).

  - **Resolution** for first-order logic (Robinson)

    - Led to development of **PROLOG-language**

Axioms, conjecture → Automated Prover Reasoning-as-search → Proved? Y/N

1970's: Boom and bust

Disappointment - what had been promised not achieved.

- No discovery of important new theorems,

- No machine translation,

- No reliable image recognition....

Limitations in computer mathematics:

- Automatic and slow. No proofs, or

- Proof checkers – little or no automation.

1970's: Boom and bust

A step back from full automation of proofs:

Interactive theorem proving - best of both worlds?

Edinburgh LCF (Milner et al)

- Program small *tactics* to automate (parts of) proofs.

- Check proof steps - small trusted core.

- Build up *theories* - theorems and proofs.

- Needed powerful *meta-language*: **ML**

1980-90's

- "Meta-language" soon became a programming language in its own right: **Standard ML**.

- Many decedents of LCF: e.g. Isabelle/HOL, HOL (Light), Coq.

1990's-00's

Increasing amount of mathematics *formalised* in interactive theorem proving libraries:

- **Newtons Principia** (Fleuriot 1998)
  - Small error found.
- **Kepler's Conjecture** (Hales 1998)
  - Flyspeck project (2003 – 2014)
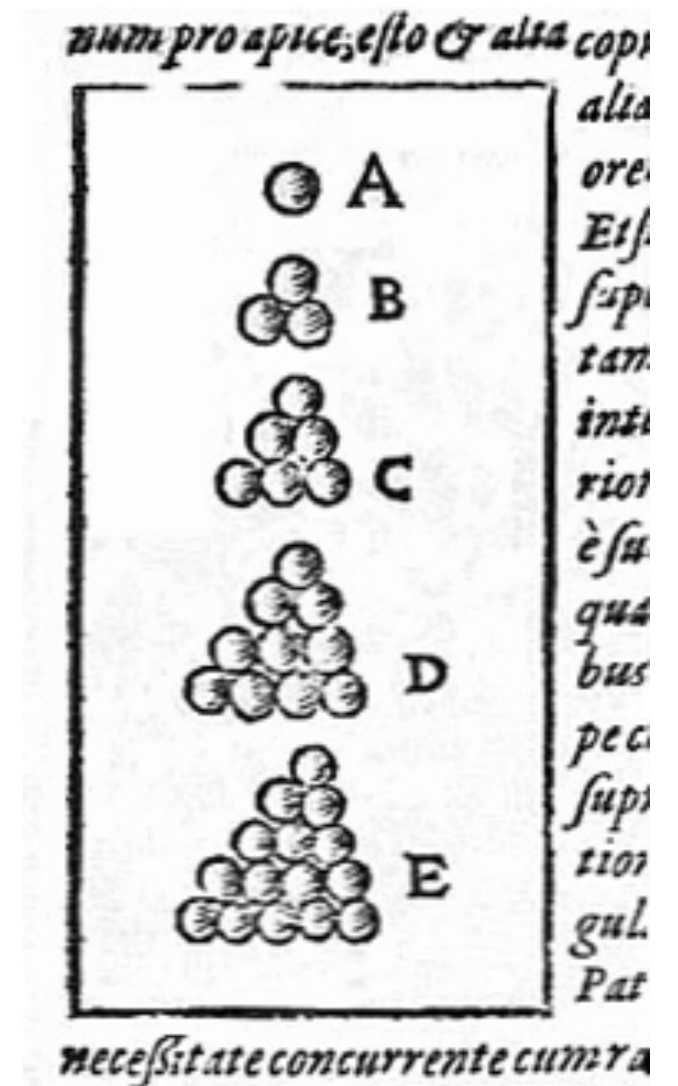
Today

"~~Within ten years~~ a digital computer will ~~discover and~~ prove an important ~~new~~ mathematical theorem"

- Faster computers - come-back of brute force reasoning-as-search.

- **Boolean Pythagorean Triples Problem:** proved automatically SAT-solver (Heule & Kullmann 2016)
  - 200 TB of proof...

- **Hammers** - interactive provers connect to automated.

  - Machine learning selects (likely) relevant facts.

  - External automated prover.

  - Proofs recreated automatically in trusted setting.

- **Challenge:** Can computers discover interesting conjectures themselves?

Demo: automate my homework

```
fun sorted :: "nat list ⇒ bool"
  where "sorted [] = True"
  | "sorted [x] = True"
  | "sorted (x1#x2#xs) = ((x1 ≤ x2) ∧ sorted (x2#xs))"

fun ins :: "nat ⇒ nat list ⇒ nat list"
  where "ins x [] = [x]"
  | "ins x (y#ys) = (if (x ≤ y) then (x#y#ys) else y#(ins x ys))"

fun isort ::  "nat list ⇒ nat list"
  where "isort [] = []"
  | "isort (x#xs) = ins x (isort xs)"

theorem my_homework: "sorted (isort x)"
```
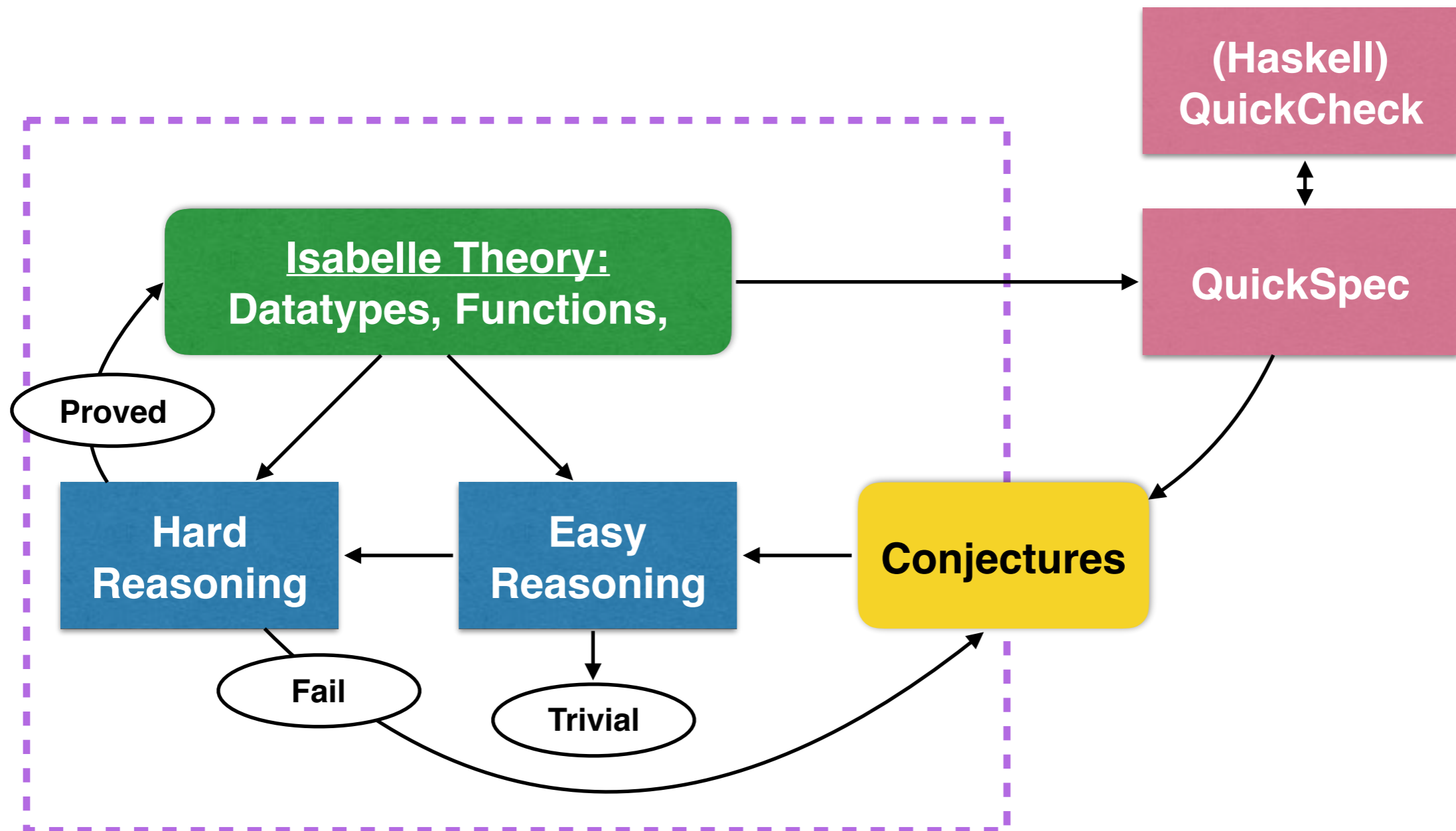
Hipster and bugs

- What happens if we make a mistake in our code?

- **Demo:** unexpected properties...

```
fun len :: "'a list ⇒ nat"
  where
    "len [] = 0"
  | "len (x#xs) = Suc(len xs)"

fun rot :: "nat ⇒ 'a list ⇒ 'a list"
  where
    "rot 0 xs = xs"
  | "rot (Suc n) [] = []"
  | "rot (Suc n) (x#xs) = rot n xs@[x]"
```

Theorem discovery in Hipster

"~~Within ten years~~ a digital computer will discover and prove an ~~important new~~ mathematical theorem"

Try it!

- https://github.com/moajohansson/IsaHipster

- http://hackage.haskell.org/package/quickspec

Thanks to all contributors and collaborators:

Nick Smallbone, Koen Claessen, Irene Lobo Valbuena,
Sólrún Halla Einarsdottir, Maximilian Algehed,
Johannes Åman Pohjola