The Theory behind TheoryMine

Alan Bundy

(Joint work with Flaminia Cavallo, Lucas Dixon, Moa Johansson & Roy McCasland)

School of Informatics,
University of Edinburgh

Cambridge 8.3.12

Outline

1. TheoryMine
2. IsaWannaThm
3. IsaCoSy
4. IsaPlanner
5. Isabelle
6. Discussion

TheoryMine

theory[mine]

- TheoryMine is a spin-out company in the novelty gift market.
  - www.theorymine.co.uk
- Generates novel theorems for customers to name.
- Theorems are inductive consequences of recursively defined functions and types.
- Certificate summarises: theorem plus type and function definitions.
  - Plus customised explanatory brochure.
  - Also tee-shirts, mouse mats and mugs.
- Theorem purchase not new, *cf* l'Hospital's Rule.

Underlying Technology

TheoryMine uses a tower of four systems:

IsaWannaThm: generates novel recursive types and functions to form new recursive theories.

IsaCoSy: given a recursive theory, generates inductive conjectures in that theory.

IsaPlanner: given an inductive conjecture, tries to prove it using an inductive proof plan.

Isabelle: is guided through proof by IsaPlanner's proof plan.

Example Certificate

TheoryMine

CERTIFICATE OF REGISTRY

Quentin's Theorem:

Let

$T = C_a(\text{bool}, \text{bool}) \mid C_b(T)$

$f_\alpha : T \times T \rightarrow T$

$f_\alpha(C_a(x,y),z) = z$

$f_\alpha(C_b(x),y) = C_b\ (f_\alpha(x,y))$

then

$$f_\alpha(y, f_\alpha(x,z)) = f_\alpha(x, f_\alpha(y,z))$$

Proof outline: induction on y

THIS THEOREM HAS BEEN NAMED AND RECORDED IN THE THEORYMINE DATABASE

19 Dec 2010

www.theorymine.co.uk

Explanation of the Certificate

New Recursive Data-Type:

$$T = C_a(bool, bool) \mid C_b(T)$$

Four-coloured naturals: 0,1,2,..., 0,1,2,..., ...

New Defined Function:

$$\begin{aligned} T \times T &\rightarrow T \\ f_\alpha(C_a(x, y), z) &= z \\ f_\alpha(C_b(x), y) &= C_b(f_\alpha(x, y)) \end{aligned}$$

A coloured version of addition: $f_\alpha(2, 3) = 5$

NB - number inherits colour of second argument.

New Theorem:

$$f_\alpha(x, f_\alpha(y, z)) = f_\alpha(y, f_\alpha(x, z))$$

A contextual commutativity.

f_α is not commutative: $f_\alpha(2, 3) = 5 \neq 5 = f_\alpha(3, 2)$

Generating Recursive Theories

- IsaWannaThm was UG project of Flaminia Cavallo.
  - But was completely overhauled by Lucas Dixon.
- It generates a set of recursive types.
- It generates some recursive functions over these types.
- These two choices provide a recursive theory.
  - Note that theories are purely definitional, so consistent.
  - Type and function spaces are both infinite,
  - but size limits imposed to ensure tractability.
- It uses IsaCoSy to generate inductive conjectures in this theory.
  - Note that all functions must appear in each conjecture.
- IsaCoSy uses IsaPlanner to prove them.

Generating Recursive Types

- Example type definition (from certificate):

$$T \quad = \quad C_a(bool, bool) \mid C_b(T)$$

- In general:

$$\tau \quad ::= \quad \ldots \mid c(\tau_1, \ldots, \tau_n) \mid \ldots$$

  - where τ is type being defined,
  - where c is typical constructor function,
  - where τ_i might be τ or a non-recursive argument.
- Grammar for generating novel types:
  - From initial set of types, e.g., *Bool* and $\mathbb{N}$.
  - Vary number of constructor functions.
  - Vary their arity and types of their arguments.
  - Filter out any already in Isabelle library.

Generating Recursive Functions

- Example function definition (from certificate):

$$
\begin{aligned}
T \times T &\rightarrow T \\
f_\alpha(C_a(x,y),z) &= z \\
f_\alpha(C_b(x),y) &= C_b(f_\alpha(x,y))
\end{aligned}
$$

- Within resource limits, incrementally generates all possible function types.
    - With one recursive argument (wlog, the first).
    - Non-recursive argument types can include *bool* and $\mathbb{N}$.
    - Avoid associative or commutative variants.
- Within resource limits, incrementally generates all possible structurally recursive functions.
    - Use IsaCoSy to generate function bodies.
    - Can use initial and previously defined functions.
    - Reject non-terminating functions using Isabelle's function package.

IsaCoSy

- IsaCoSy was PhD project of Moa Johansson.
- It generates irreducible terms. In particular, conjectures.
- Used by TheoryMine to generate inductive conjectures for any recursive theory, e.g.,

$$f_\alpha(x, f_\alpha(y, z)) \quad = \quad f_\alpha(y, f_\alpha(x, z))$$

- TheoryMine conjectures are quantifier-free equations between irreducible terms.
- Within resource limits, incrementally generates all possible such conjectures.
- Non-theorems filtered out by quickcheck counter-example finder.
- Survivors sent to IsaPlanner to be proved.

IsaCoSy's Irreducible Term Generation

- Irreducibility ensures that conjectures:
  - Are in normal form, so simplest expression of result.
  - Are not rewrite consequence of definitions and previous theorems.
  - That is, induction (or rewriting backwards) is required to prove them.
  - Therefore, have some intrinsic merit.
- Constraints ensure reducible terms are not generated.
  - Definitions and theorems are oriented as rewrite rules, e.g., $f(c(x)) \Rightarrow t$.
  - A new constraint is then imposed to ban generation of (sub-)terms of form $f(c(\ldots))$.
  - Constraint set grows with new definitions and theorems.
- IsaCoSy also generates bodies of function definitions.

IsaCoSy Results

- Very good precision/recall results against Isabelle libraries.
- Typical IsaCoSy theorems in regular theories.

$$
\begin{aligned}
a \times b &= b \times a \\
(a + b) + c &= a + (b + c) \\
(a \times b) + (c \times b) &= (a + c) \times b \\
rev(map\ a\ b) &= map\ a(rev\ b) \\
foldl\ a\ (foldl\ a\ b\ c)\ d &= foldl\ a\ b\ (c @ d)
\end{aligned}
$$

The Quickcheck Counter-Example Finder

- Quickcheck is an Isabelle tool being developed by Lukas Bulwahn.
- IsaCoSy sends it equations of form $t_1(\vec{x}) = t_2(\vec{x})$, where $\vec{x} : \vec{\tau}$.
- Quickcheck exhaustively generates all small $\vec{c} : \vec{\tau}$.
- It turns the t_i into ML programs and evaluates them on these $\vec{c}$.
- If $t_1(\vec{c}) \neq t_2(\vec{c})$ for some $\vec{c}$ then conjecture is false.

Limitations of Quickcheck

Conditionals: $P(\vec{x}) \implies t_1(\vec{x}) = t_2(\vec{x})$.

- If $P(\vec{x})$ is rarely true then $P(\vec{c})$ is usually false and conditional true.
- Need some way to generate only $\vec{c}$s for which $P(\vec{c})$ is true.

Existentials: $\exists \vec{x} : \vec{\tau}.\ P(\vec{x})$

- Need to prove $\forall \vec{x} : \vec{\tau}.\ \neg P(\vec{x})$ to disprove conjecture.
- Quickcheck can only do this when $\vec{\tau}$ is finite.

IsaPlanner

- IsaPlanner was originally PhD project of Lucas Dixon.
- Uses proof planning to guide Isabelle on inductive conjectures.
  - Uses rippling and proof critics.
  - Proofs entirely automatic.
  - High success rate on simple theorems.
- Example theorems for $T + f_\alpha$ theory:

  The Richard Scott Russell Theorem:

$$f_\alpha(x, C_b(y)) = C_b(f_\alpha(x, y))$$

  The Herdman Theorem:

$$f_\alpha(f_\alpha(x, y), z) = f_\alpha(x, f_\alpha(y, z))$$

  Quentin's Theorem

$$f_\alpha(y, f_\alpha(x, z)) = f_\alpha(x, f_\alpha(y, z))$$

Inductive Proof of Quentin's Theorem

Quentin's Theorem: $f_\alpha(y, f_\alpha(x, z)) = f_\alpha(x, f_\alpha(y, z))$

Rewrite Rules:

$$
\begin{aligned}
f_\alpha(C_a(x, y), z) &\Rightarrow z && (1)\\
f_\alpha(C_b(x), y) &\Rightarrow C_b(f_\alpha(x, y)) && (2)\\
f_\alpha(x, C_b(y)) &\Rightarrow C_b(f_\alpha(x, y)) && (3)
\end{aligned}
$$

Base Case:

$$
\begin{aligned}
f_\alpha(C_a(b_1, b_2), f_\alpha(x, z)) &= f_\alpha(x, f_\alpha(C_a(b_1, b_2), z)) &&\\
f_\alpha(x, z) &= f_\alpha(x, z) && \text{by } 2\times(1)
\end{aligned}
$$

Step Case:

$$
\begin{aligned}
f_\alpha(C_b(y), f_\alpha(x, z)) &= f_\alpha(x, f_\alpha(C_b(y), z)) &&\\
C_b(f_\alpha(y, f_\alpha(x, z))) &= f_\alpha(x, C_b(f_\alpha(y, z))) && \text{by } 2\times(2)\\
C_b(f_\alpha(y, f_\alpha(x, z))) &= C_b(f_\alpha(x, f_\alpha(y, z))) && \text{by } (3)\\
C_b(f_\alpha(x, f_\alpha(y, z))) &= C_b(f_\alpha(x, f_\alpha(y, z))) && \text{by hyp}
\end{aligned}
$$

Isabelle

- Generic, interactive proof assistant from Cambridge and Munich.
- Classical, higher-order logic most popular theory,
    - which is what we use.
- LCF-style prover with small, trusted core of logical rules,
    - provides very high level assurance of correctness.
- Tactic-driven by IsaPlanner.
- Also includes quickcheck counter-example finder.

Explanation of Design Decisions

The following TheoryMine features are crucial to its business model:

- Restriction to purely definitional theories ensures consistency.
- Isabelle's LCF-style architecture ensures correctness.
- IsaPlanner's proof planning provides automatic proof.
- IsaCoSy's irreducibility heuristic ensures intrinsic merit of theorems.
- IsaWannaThm's meta-grammars generate a huge number of novel theories,
    - and hence theorems: initially estimated at 10^{16}.

Conclusion

- Successfully generates huge numbers of novel theorems of some intrinsic merit.
- Design decisions motivated by business model.
- Recursive types, functions and theorems are simple.
- Many open conjectures generated.
- TheoryMine slows as theories become more complex.

Further Work

- More user involvement.
- Extend product offering:
    - Have English, Mandarin and Spanish language — plan more.
    - On-line journal with proof summary.
- Extend space of types, e.g., non-free, mutual, higher-order.
- Extend space of functions, e.g., simultaneous, mutual, higher-order.
- Extend space of theorems, e.g., conditional, existentials, higher-order.
- Make IsaCoSy more efficient.
- Improve proof-power of IsaPlanner.
- Open conjectures as resource.