Logic and Language, Proposition and Types, Proofs and Computation

The Particle Physics of Computer Science

Andreas Abel

Department of Computer Science and Engineering
Chalmers and Gothenburg University

Introduction for 1st year students
Lunch Seminar, HC2, Chalmers
21 April 2015

Programming Logic

- ProgLog group
- Professors: Thierry Coquand, Peter Dybjer, Bengt Nordström
- Permanent staff: Andreas Abel, Ana Bove, Nils Anders Danielsson, Ulf Norell
- Overall goal: Correctness of programs through logical means
    - Foundations of programming
    - Foundations of logics and mathematics

Correctness of programs

- Example: compiler correctness.
- Produced JVM byte code should faithfully represent Java source code.

```
JAVA:  y = x + 5

JVM:   iload  1
       bipush 5
       iadd
       istore 2
```

- There are infinitely many possible Java programs; we cannot test the compiler on all.

Compiler correctness

- Compiler is a function, inputs Java, outputs JVM.

$$\mathsf{compile} : \mathsf{Java} \to \mathsf{JVM}$$

- Correctness means that compilation preserves meaning of code.
- Meaning of target: behavior of JVM code when run (executed in bytecode interpreter).
- Meaning of source: behavior of Java program when executed in an interpreter.

$$\forall (p : \mathsf{Java}) \to \mathsf{interpret}(p) = \mathsf{run}(\mathsf{compile}(p))$$

Compiler correctness

- What is a Java program, mathematically?
  - A sentence (string) following the Java *grammar*. [Languages, grammars, parsing]
  - Representable as abstract syntax tree. [Data structures, recursion]
- What is JVM code, mathematically?
- What is the meaning of a Java program? [Interpreter, semantics]
- What is the meaning of JVM code? [Machines, execution]
- What does "equal behavior" mean? [Relations, models of computation]
- How can we prove something for all Java programs? [Logic, induction]
- How can we be sure our proof is correct? [Proof theory, machine-assisted verification]

A simpler example

- Say we have a list l of natural numbers.

i	0	1	2	...	i	...	$n-1$
l	2	3	5	...	$\mathsf{lookup}\, l\, i$	...	$\mathsf{lookup}\, l\, (n-1)$
$\mathsf{incr}\ l$	3	4	6	...	$1 + \mathsf{lookup}\, l\, i$	...	$1 + \mathsf{lookup}\, l\, (n-1)$

- The following two should be equivalent.
  1. Making a copy of the list with each element increased by 1 (incr) and then taking the ith element (lookup).
  2. Taking the ith element (lookup) and increase it by 1 (suc).

$$\forall (l : \mathsf{List}\, \mathbb{N})(i : \mathbb{N}) \to \mathsf{lookup}\, (\mathsf{incr}\, l)\, i \equiv \mathsf{suc}\, (\mathsf{lookup}\, l\, i)$$

Modelling our example

- Data structures: natural numbers and lists
  [choice, composition, recursion]
- Functions: traversing a list
  [case distinction, recursion]
- Logic: proof of universal ($\forall$) statement
  [induction $=$ case distinction $+$ recursion]

Curry-Howard-Isomorphismus

$$\begin{array}{rcl} \text{Proposition} & \cong & \text{Set} \\ \text{proof} & \cong & \text{program/data} \end{array}$$

- Discovered in 1950s.
- Logic inspires programming language research.
- Programming language constructs find logical interpretations.

Particles of Computer Science

A logical approach to information and computation.

With quotes from L. & A. Wachowski,
The Matrix Reloaded

Causality (Implication)

MEROVINGIAN: You see, there is only one constant, one universal,
It is the only real truth: *causality*.
Action. Reaction.
Cause and effect.

Functions. Transforming input to output.
Implication. Conclusions from premises.

$$\mathsf{incr} : \mathsf{List}\,\mathbb{N} \to \mathsf{List}\,\mathbb{N}$$

$$\mathsf{lookup\text{-}incr} : (l : \mathsf{List}\,\mathbb{N})(i : \mathbb{N}) \to \mathsf{lookup}\,(\mathsf{incr}\,l)\,i \equiv \mathsf{suc}\,(\mathsf{lookup}\,l\,i)$$

$$(\mathsf{Even}(n) \wedge \mathsf{Prime}(n)) \to n \equiv 2$$

Structure (Conjunction)

KEYMAKER: The system is based on the rules of a building.
One system built on another.

If one fails, all fail.

Tuples: several things put together.
E.g. the cons of lists, pairing head (1st element) and tail (rest).
Conjunction: 2 is an odd prime number.

$$(1, 2)$$
$$\text{head} :: \text{tail}$$
$$\text{Odd}(2) \land \text{Prime}(2)$$

Choice (Disjunction)

THE ORACLE: We can never see past the choices we don't understand.

MORPHEUS: Everything begins with choice.

NEO: Choice. The problem is choice.

Bits: false or true, zero or successor, empty list or cons.
Each natural number is either even or odd.

bit	0	1
Bool	false	true
$\mathbb{N}$	zero	suc
List	[]	_ :: _

$$\mathsf{Even}(n) \vee \mathsf{Odd}(n)$$

Recursion

AGENT JACKSON: You.
SMITH: Yes me. Me, me, me!
AGENT JACKSON/SMITH: Me too!

SMITH: Go ahead, shoot. The best thing about being me—
there's so many me.

Recursive data types (e.g. lists).
Recursive functions (e.g. incr, lookup).
Recursive proofs (induction, e.g. lookup-incr).

$$\begin{array}{rcl} (n : \mathbb{N}) :: (l : \mathsf{List}\,\mathbb{N}) & : & \mathsf{List}\,\mathbb{N} \\ \mathsf{lookup}\,(n :: l)\,(\mathsf{suc}\,i) & = & \mathsf{lookup}\,l\,i \end{array}$$

Agda

- Haskell-like programming language
- Based on the Curry-Howard-Isomorphismus
- Agda 2 developed at Chalmers since 2006
- Precursors since 1980s (ALF, Half, Alfa, Agda)

ProgLog Courses

DAT060 Logic in computer science (Coquand)

- Proof calculi, applications of logic

TMV027 Finite automata and formal languages (Bove)

- Grammars, parsing

DAT140 Types for proofs and programs (Dybjer)

- Programming language theory
- Type theory and Agda

TDA183 Models of computation (Nordström)

- Lambda calculus, Turing machines
- Undecidability