An E-Learning System for "The Formal Semantics of Computer Programs" and Its Implementation by Program Generation

Atsushi Igarashi
(Kyoto Univ.)

Plan of The Talk

- Graduate Course on the Formal Semantics of Computer Programs
- Demo of the System
- How the system is designed and implemented
- Concluding Remarks

Formal Semantics of Computer Programs

- Taught in Grad. Schl. of Informatics, Kyoto U. for 12 yrs.
- In 2013:
  - Big-step semantics of core ML, incl.
    - Lists and pattern matching
  - ML type system and HM type inference
  - Metatheory: type soundness, principal types, etc.
  - E-Learning system for self-study (2008-)

E-Learning System (2008-)

- Exercise to write down *concrete* derivations of evaluation and typing relations
  - To get used to derivations
  - To understand the object theory (i.e., ML) better
- Web-based system (open to public)
  - Automated derivation checker(s)
  - 150 questions

Demo

Plan of The Talk

- Graduate Course on the Formal Semantics of Computer Programs
  - Course overview
- Demo of the System
- How the system is designed and implemented
- Concluding Remarks

Goals and Requirements

Goals

- Gentle error messages
- Minimal development time
- No discrepancies btw. textbook and system

Requirements

- Derivation checker
  - No formal metatheory needed
- Handling many different formal systems

Three Options I Considered

- Use of proof assistants (Coq, Twelf, …)
  - Kind of overkill
  - Indirect representation
- Generic derivation checker
- Derivation checker generator

Overview of the System

- CGI frontend in Gauche (Scheme impl.)
- Derivation checkers in ML

Derivation Checker Generator

- Input: Specification of a formal system
- Output: ML functions for derivation checking
  - Algorithm is independent of systems
  - (parsing has to be implemented manually)
  - Formal system description in TeX is also generated
    - LaTeX2MathML

Prelog:
DSL for Specifying Formal Systems

- Prelog: "Prelogical" framework :-)
- Formal system = syntax for objects and judgments + inference rules
- Inference rules *à la* Prolog
  - Typechecking for inference rules
- Side conditions can be written in ML
- c.f. Ott by Swell et al.

A Specification Consists of:

- Syntax section
  - BNF for objects (such as numbers, expressions)
  - Declaration of metavariables
    - Names decide their syntactic categories
- Judgments section
  - (Abstract) syntax for judgments
- Rules section
  - Inference rules in Prolog-like syntax
  - Backquotes for side conditions

Examples

- NatExp: addition and multiplications of Peano natural numbers
- EvalML1: evaluation of simple expressions w/
  - Integers, Booleans
  - Arithmetic and comparison operations
    - Implemented in ML
  - Conditionals

Typechecking Spec.

- Checks if inference rules respect syntactic categories given in the Syntax section
  - Syntactic categories as types
  - injection as subtyping
  - Side-conditions are not checked ;-(

Formal Systems Used in Class

- Addition, multiplication, comparison of Peano numerals
- Big-step semantics of core ML in stages
  - Simple expressions
  - Local declarations (and environments)
  - Functions and recursions
  - Lists and pattern matching
  - References and continuations
- De Brujin index and nameless evaluation
- Simple and HM type systems

Plan of The Talk

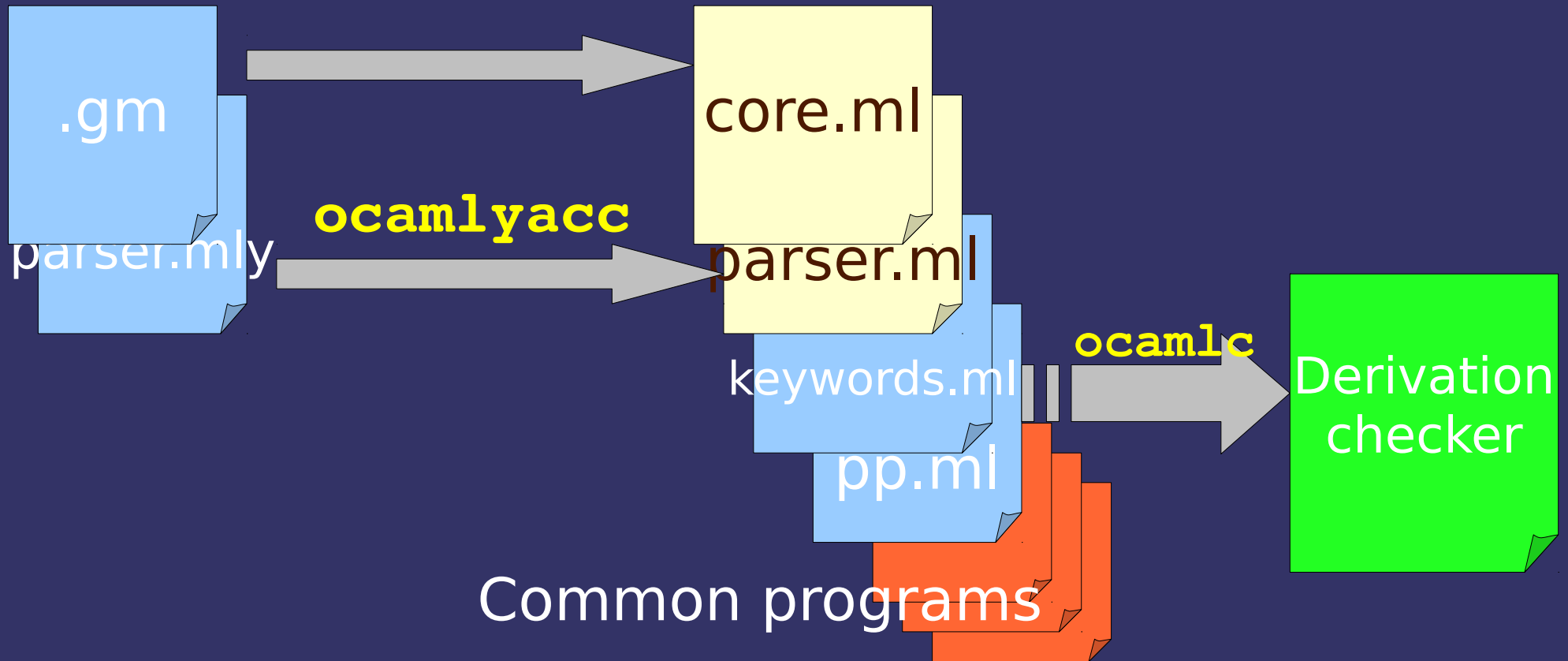
- Graduate Course on the Formal Semantics of Computer Programs
  - Course overview
  - Demo
- How the system is designed and implemented
  - Specification Language "Prelog"
  - Implementation
- Concluding Remarks

Implementation Overview

- Typechecker (for Prelog)
- Prelog-to-OCaml translator in OCaml
  - Datatypes for objects and judgments
  - Function check_deriv: deriv → judgment
    - Case analysis on the last rule used
    - Pattern-matching the last deviration step and the rule
      - Relying mostly on ML
      - Handling multiple occurrences of the same variable

You still have to implement ...

- Parser
- Pretty printer (pp.ml) for debugging



Bonus: Derivation Generator Generator

Why: Useful for debugging questions

How:

- Mode analysis for logic programs
  - Each judgment comes with annotation of which arguments are inputs and which are outputs
  - Checks if inference rules are "consistent" with the annotations
- Search with backtracking

Mode Analysis

- Judgement: J(i; o)
- Rule:

$$\frac{J1(e1, f1) \quad \ldots \quad Jn(en, fn)}{J(e,f)}$$

  - Parameters in ei can use parameters only in e, f1, ..., and fi-1
  - Parameters in f can use parameters only in e, f1, ..., and fn

Summary of Prelog

- First-order syntax
  - No need to implement pattern matching
- Shell-like Backquotes to implement side conditions
  - Once students learn how to formalize addition, they don't want to care about it in later systems
- Typechecking formal system spec.

What I Learned

- ML pattern matching is really GRRRREAT!
- Code generation by printf is much better than nothing but still annoying
- Many duplications in derivation system specs
- Most errors students make are syntax errors
  - Parsers with helpful error messages are important

Why not MetaOCaml?

- No support for generating datatype declarations
  - Universal data structure to sidestep the problem
- No support for offline program generation (until near future…?)

Related Work

- Twelf, Coq
  - Hard for beginners
  - Derivations are encoded (and put behind the scene)
- CAL [Sato,'97-] (implemented in elisp!)
  - Used in an undergraduate course at Kyoto
    - More emphasis on logic
  - Our notations follow this system closely
- Prolog

- Ott [Sewell et al.'07 ICFP]
  - Probably we should integrate our system into this!
- SASyLF [Aldrich et al. '08 FDPE]
  - (Another) implementation of Twelf
  - Ott-like specification language
  - Proof notation is closer to hand-written proofs (than original Twelf)

Summary of the Talk

- E-Learning system for writing down formal derivations
- Prelog: Metalanguage for specifying formal systems
- Generation of derivation checkers and generators

Some Statistics

- More than 100 users
  - Well, # of students every year is only about 15
- 17 people solved all the questions
  - Some people took less than 3 days!