CSci 8980, Fall 2012
Specifying and Reasoning About Computational Systems
Introduction to the Course

Gopalan Nadathur

Department of Computer Science and Engineering
University of Minnesota

Lectures in Fall 2012

Objectives for the Course

Expose a logic-based approach to the treatment of syntax-directed and rule driven descriptions

In particular, we will gain an understanding of at least the following aspects

- A logic programming approach to encoding rule-based specifications
- A logical approach to capturing the meta-theory of logic programming languages
- Foundational underpinnings of the different logical approaches
- Particular languages and programming systems that realize the above capabilities
- Computational aspects that make the systems work

Motivation for Studying these Topics

Syntax-driven and rule based specifications are commonly used for describing computational notions

For example, the following are typically described in this way

- semantics of programming language constructs
- typing in programming languages
- natural deduction and sequent calculi
- software modelling languages (e.g. π-calculus)

Moreover, these specifications are often used as the basis for (informal) reasoning about the systems

Logics for formalizing such specifications and reasoning can be used to provide computer-based support for these activities

Motivations (Continued)

More specifically, satisfying the agenda will require providing a means to do the following:

- Transparently encoding syntax-based descriptions
  - succinct, declarative representation of objects treated
  - logic-based encoding of all aspects of rules
- Prototyping based on specifications
  - Effectively automatable proof search
  - Reflection of underlying computational structure
- Reasoning through specifications
  - characterizing positive *and* negative behaviour
  - specification properties should reflect actual system properties (adequacy)

These requirements will guide the construction of the logics we discuss

Motivating Example (Formalizing mini-Lazy ML)

Syntax

$ty ::= \texttt{int} \mid ty \to ty \qquad e ::= x \mid e\ e \mid (\texttt{fn}\ x : ty \Rightarrow e)$

Call-by-Name Evaluation ($e \Downarrow v$ means e evaluates to v)

$$\frac{}{(\texttt{fn}\ x : a \Rightarrow r) \Downarrow (\texttt{fn}\ x : a \Rightarrow r)}$$

$$\frac{m \Downarrow (\texttt{fn}\ x : a \Rightarrow r) \qquad r[n/x] \Downarrow v}{(m\ n) \Downarrow v}$$

Typing Rules ($\Gamma \vdash e : ty$ means e has type ty relative to Γ)

$$\frac{\texttt{x} : a \in \Gamma}{\Gamma \vdash \texttt{x} : a} \qquad \frac{\Gamma \vdash m : a \to b \qquad \Gamma \vdash n : a}{\Gamma \vdash (m\ n) : b}$$

$$\frac{\Gamma, x : a \vdash r : b}{\Gamma \vdash (\texttt{fn}\ x : a \Rightarrow r) : a \to b}\ x \notin dom(\Gamma)$$

Motivating Example (Continued)

In the setting of these specifications, we want to be able to provide the following kinds of capabilities:

- An easy to use and transparently correct representation of types and terms
- A formalization of *all* aspects of the different rules, especially the side conditions
- Executability of the formalization
- Reasoning about the formalization

  E.g. we would want to derive properties such as
  - Determinateness of evaluation
  - Preservation of typing by evaluation
  - Uniqueness of typing
- Transference of the properties to the actual system

Realizing the Desired Capabilities

Our realization of the capabilities we desire will rely on the following logical devices:

- a rich, higher-order logic for logic programming
- the use of lambda terms as data structures
- logics of definitions (with induction and co-induction)
- differentiation between intensional and extensional universal quantification
- two-level logic form of reasoning

We will develop an understanding of these notions at different points in the course