

Comp 411 Notes - 14 Jan 2005

BASIC CONCEPTS

Reproduced by Moez A. Abdel-Gawad
(moez@cs.rice.edu)

1 Important points from previous lecture

- Language safety is different from type safety (broader). Both are relative to a specification.
- Different ways to specify sets (recursively): BNF, by construction, semantics??

2 Alternative Math Symbols

In homeworks handed by email, we can use the following text symbols in place of the corresponding mathematical symbols:

Symbol	Text
\in	<code>\in</code>
\subseteq	<code><=</code>
\subset	<code><</code>
\cup	<code>+</code>
\cap	<code>*</code>
A_i	<code>A_i</code>
A^i	<code>A^i</code>
\Rightarrow	<code>==></code>
\rightarrow	<code>--></code>

3 Important Definitions

The following are important definitions used throughout the course.

3.1 Relations

A *Relation* on two sets A and B is a subset of the cross product of A and B ($A \times B$). $R \subseteq A \times B$.

3.2 Functions

A *Function* from set A to set B is a relation F from set A to set B ($F \subseteq A \times B$), with the property that every element of A has at most one corresponding element in B (thru F). That is, $\forall x \in A, \forall y, y' \in B$, we have:

$$((x, y) \in F \wedge (x, y') \in F) \Rightarrow y = y'$$

3.3 Judgements

A foundational concept, or construct, that we'd be using a lot in COMP 411 is the *Judgement*. A judgement is a statement of belonging to a set, *based on a proof (premises, preconditions, or antecedents)*. A judgement is, thus, always associated with a constructive proof (intentional??), which makes it appealing to computer scientists.

3.3.1 Examples

Examples of judgements are in the following *rules*:

$$\frac{}{0 \in \text{Nat}} R_0$$
$$\frac{n \in \text{Nat}}{n+1 \in \text{Nat}} R_+$$

Antecedents (or, *preconditions*) are written above a rule's dividing line, while the *judgement* is written below the line.

A *derivation*, or *proof* as an object, is composed of multiple applications of the rules. A derivation becomes a *derivation tree* when a rule is used that has more than one antecedent.

$$\frac{\frac{}{0 \in \text{Nat}} R_0}{0+1 \in \text{Nat}} R_+}{(0+1)+1 \in \text{Nat}} R_+$$

Bottom-up, and *top-down*, are both approaches used to build derivations, but in this course we'd not get into that. We'd only deal with the final product, the derivation, regardless of where it came from, and how it was constructed.

$\boxed{1 + 2 \leftrightarrow 3}$ is another way to write a judgement. The whole box refers to the proof *object* that we have for the judgement inside, i.e., the whole derivation tree. Same applies for the following judgement $\boxed{x : \text{int}, y : \text{int} \vdash x + y : \text{int}}$

4 BNF

BNF (Backus-Naur Form) is a standard method of specifying CFG (Context Free Grammars).

We can specify the syntax for Nat using BNF as follows:

$$n \in \text{Nat} ::= 0 \mid n + 1$$

Type Systems, which depend on Lambda Calculus (LC), however, need more than context-free grammars.

The *syntax* for LC can be specified by: $e \in LC ::= x | \lambda x. e | ee$, where $x \in Var$

This context free definition, however, includes *open*, as well as *closed*, terms of lambda calculus.

To express closed terms, context-sensitivity is needed. To see why, check proof presented later (using pumping lemma for PDAs).

4.1 Free Variables, and Open Terms

We define open terms (of LC) using a recursive (and context-sensitive) definition of free variables of a term.

$$\begin{aligned} FV(x) &= \{x\} \\ FV(\lambda x. e) &= FV(e) \setminus x \\ FV(e_1 e_2) &= FV(e_1) \cup FV(e_2) \end{aligned}$$

(Note the recursive definition of FV, which follows the recursive definition of e . This would be the case in many proofs and definitions we'd see in this course.)

$$closed(e) \Leftrightarrow FV(e) = \Phi(= \{\})$$

4.2 Examples

Here are some examples of closed and free lambda calculus terms.

- $closed(\lambda x. x) \Leftrightarrow Free(\lambda x. x, \{\}) \Leftrightarrow \cdot \vdash \lambda x. x$
(the inclusion of $\{\}$, or the \cdot , to denote Φ , the empty 'environment', adds "context" – i.e., makes the judgements context-sensitive)
- $\neg closed(\lambda x. y) \Leftrightarrow \neg Free(\lambda x. y, \{\}) \Leftrightarrow \neg(\cdot \vdash \lambda x. y)$

4.3 Environments

Environments add context-sensitivity to judgements.

$\frac{}{P, x, S \vdash x} \Leftrightarrow \frac{x \in L}{L \vdash x}$, where P is the prefix of the environment (subsequence before x), and S is the suffix.

$$\frac{L, x \vdash e}{L \vdash \lambda x. e}$$

$$\frac{L \vdash e_1 \quad L \vdash e_2}{L \vdash e_1 e_2}$$

Extra HW: Prove that $L \vdash e \Leftrightarrow FV(e) \subseteq L$