Lecture 4: Simple Type Theory: Syntax and Axiomatic Semantics

Lecturer: Max S. New
Scribe: Yanjun Chen

Jan. 23, 2023

Changes from class: all congruence rules are included.

1 Introduction

Let us consider a simple judgment in IPL: $\cdot \vdash \top \vee \top$. This is certainly provable, and in fact there are many ways to prove it. For example,

$$\dfrac{\dfrac{}{\cdot \vdash \top}\,\top I}{\cdot \vdash \top \vee \top}\,\vee I_1 \qquad \dfrac{\dfrac{}{\cdot \vdash \top}\,\top I}{\cdot \vdash \top \vee \top}\,\vee I_2 \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\cdot \vdash \top}\,\top I \quad \dfrac{}{\cdot \vdash \top}\,\top I}{\cdot \vdash \top \wedge \top}\,\wedge I}{\cdot \vdash \top}\,\wedge E_1}{\cdot \vdash \top \vee \top}\,\vee I_1$$

Which of these should we consider to be *the same* proof? The first two have analogous structure, but the first one uses a $\vee$ introduction of the left $\top$ while the second one uses $\vee$ introduction of the right $\top$. The third is structurally different, but ultimately proves the theorem by using $\vee$ introduction for the left $\top$. We might say that the third is essentially the same argument as the first, but takes an unnecessary "detour" to prove $\top\top$. Today we will see a system for deciding when two proofs are equal such that the first and third are equal but the second proof here is considered to be different. Furthermore, in this system we will eventually show that *every* proof of $\cdot \vdash \top \vee \top$ is ultimately equal to the first or second proof.

This is related to a more general property, the *disjunction* property for IPL.

Theorem 1 (Disjuction Property)**.** *If $\cdot \vdash A \vee B$, then $\cdot \vdash A$ or $\cdot \vdash B$.*

We will delay proving this for a few sessions as we will develop some tools to assist us.

This theorem tells us that whenever we have $\cdot \vdash A \vee B$, then there is a proof that ends with an introduction rule:

$$\dfrac{\dfrac{}{\cdot \vdash A}}{\cdot \vdash A \vee B}\,\vee I_1 \qquad \dfrac{\dfrac{}{\cdot \vdash B}}{\cdot \vdash A \vee B}\,\vee I_2$$

Furthermore this is a *constructive* proof, we can construct a computer program that, given a syntactic description of the derivation of $\cdot \vdash A \vee B$, one can produce a derivation for $\cdot \vdash A$ or $\cdot \vdash B$. And in our system of

Note: this theorem is not trivial to prove. For example, the derivation shown below can be used to extract proof of A or B.

$$\dfrac{\dfrac{\vdots}{\cdot \vdash (A \vee B) \wedge C}}{\cdot \vdash A \vee B} \wedge E_1$$

Note: if the context is not empty, the disjunction property does not hold. For example, $A \vee B \vdash A \vee B$ is provable by assumption, but we can't necessarily extract a proof of $A \vee B \vdash A$ or $A \vee B \vdash B$. As a specific counter-example take $X \vee (\neg X) \vdash X \vee (\neg X)$.

From the disjunction property and the soundness theorem, it also follows that the law of excluded middle is not derivable. If we take IPL with one propositional variable, the law of excluded middle would give us that $\cdot \vdash X \vee \neg X$, and therefore we would have either *cdot* $\vdash X$ or $\cdot \vdash \neg X$. But in IPL, $\cdot \vdash X$ is not provable because by the soundness theorem we can construct a model in booleans where X is interpreted as 0 and $\cdot \vdash \neg X$ is also not provable because we can construct a boolean model where X is interpreted as 1.

2 Simple Type Theory

The system we will use that gives us a notion of when two proofs are equal is called *Simple Type Theory*.

In simple type theory, instead of considering the theorem like $\Gamma \vdash A$ in IPL, we have a new judgement $\Gamma \vdash M : A$ where Γ is context, M is called term or program, A is called a type, the analogue of a proposition in IPL.

In the context, there exists a finite list of variables with type like $[x_1 : A_1, x_2 : A_2, \cdots]$. The basic assumption for variable is that the variables with different names are different variables. The rule of variable is given below,

$$\dfrac{x : A \in \Gamma}{\Gamma \vdash x : A} \text{ Var}$$

This has the same structure as the assumption rule in IPL, but now we keep track of *which* assumption we used by referring to it by name x.

We then provide rules in simple type theory. The conjunction $A \wedge B$ becomes the type of ordered pairs, which we write as $A \times B$,

$$\dfrac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B} \times I$$

The notation for conjunction is not a meet $\wedge$ but a Cartesian product $\times$, which emphasis that the $A \times A$ is different from A whereas in a poset, $a \wedge a = a$. The (M, N) is a pair, which is constructor of term. Again, there are two elimination rules,

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \pi_1 M : A_1} \times E_1 \qquad \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \pi_2 M : A_2} \times E_2$$

The π_i is an constructor called projection that eliminates the term. We have a new judgement called equations of form $\Gamma \vdash M = N : A$ which is a judgement for whether the proof equals. These rules give a way to "run" the program or simplify the terms. Since this is an equality, there are three rules for equivalent relation,

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M = M : A} \text{ refl} \quad \frac{\Gamma \vdash N = M : A}{\Gamma \vdash M = N : A} \text{ sym} \quad \frac{\Gamma \vdash M = N : A \quad \Gamma \vdash N = P : A}{\Gamma \vdash M = P : A} \text{ trans}$$

Additionally, for every new term constructor, we add a corresponding congruence rule. For instance for the product introduction and elimination rules,

$$\frac{\Gamma \vdash M_1 = M_1' : A_1 \quad \Gamma \vdash M_2 = M_2' : A_2}{\Gamma \vdash (M_1, M_2) = (M_1', M_2') : A_1 \times A_2} \times I \ cong$$

$$\frac{\Gamma \vdash M = N : A_1 \times A_2}{\Gamma \vdash \pi_1 M = \pi_1 N : A_1} \times E_1 \ cong \qquad \frac{\Gamma \vdash M = N : A_1 \times A_2}{\Gamma \vdash \pi_2 M = \pi_2 N : A_1} \times E_2 \ cong$$

Now that we care additionally about equality between terms, we have that every connective comes with introduction and elimination rules but also equality rules called β and η rules. The β-rules are a kind of computation rule to simplify the program. The β rule says if we apply an elimination rule after an introduction rule, the term simplifies away the intermediate step and we get out what we put in in the introduction rule. So for pairs, if we construct a pair (M_1, M_2) and then we do a projection, we get out the corresponding field of the pair:

$$\frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \pi_1(M_1, M_2) = M_1 : A_1} \times \beta_1 \quad \frac{\Gamma \vdash M_1 : A_1 \quad \Gamma \vdash M_2 : A_2}{\Gamma \vdash \pi_2(M_1, M_2) = M_2 : A_2} \times \beta_2$$

The other rule is called the η-rule which can be thought of as a rule for extensional reasoning about equality between terms that have the type. One way to formulate them is that they say that any term of the type $A \times B$ can be reconstructed as an introduction rule after the elimination rules. For conjunction, the rules are give below,

$$\frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash M = (\pi_1 M, \pi_2 M) : A_1 \times A_2} \times \eta$$

This says that any pair $M : A_1 \times A_2$ is equal to one re-constructed using the introduction rule.

An alternative, equivalent way to phrase the η rule is to say that two terms $M, N : A \times B$ are equal if they are equal when using each of the elimination forms.

$$\frac{\Gamma \vdash M : A_1 \times A_2 \quad \Gamma \vdash N : A_1 \times A_2 \quad \Gamma \vdash \pi_1 M = \pi_1 N : A_1 \quad \Gamma \vdash \pi_2 M = \pi_2 N : A_2}{\Gamma \vdash M = N : A_1 \times A_2} \times \eta'$$

We can compare the IPL and this simple type theory shown in the table.

IPL	Simple Type Theory
A proposition	A type
$\dfrac{\mathcal{D}}{\Gamma \vdash A}$ proof	$\dfrac{\mathcal{D}}{\Gamma \vdash M : A}$ program
-	$\dfrac{}{\Gamma \vdash M = N : A}$ equation
Γ: a sequence of prop	Γ: a sequence of form $x : A$

We then consider the $\top$ and $\bot$ in our system.

$$\frac{}{\Gamma \vdash () : 1}\ 1I$$

The () is considered as a tuple of zero elements. And 1 is the $\top$ in IPL which indicates the identity of multiplication. There is no β-rule since no elimination exists. For η-rule,

$$\frac{\Gamma \vdash M : 1}{\Gamma \vdash M = () : 1}\ 1\eta$$

or in the alternative form

$$\frac{\Gamma \vdash M : 1 \Gamma \vdash N : 1}{\Gamma \vdash M = N : 1}\ 1\eta'$$

Similarly, for $\bot$, we use 0 to mean the empty type, and to indicate the identity of addition. There are no introduction forms as it is the empty type. The elimination rule and η-rule is given below.

$$\frac{\Gamma \vdash M : 0}{\Gamma \vdash \textbf{case}\ M\{\} : C}\ 0E$$

The elimination rule is a case-split on the possible introduction forms. For 0 there are no introduction forms, so we provide no cases. We also add a corresponding congruence rule:

$$\frac{\Gamma \vdash M = N : 0}{\Gamma \vdash \textbf{case}\ M\{\} = \textbf{case}\ N\{0\} : C}\ 0\text{Econg}$$

The η principle is slightly different than the ones for $1, \times$. Rather than being about terms of type 0, it is about terms that have a *free variable* of type 0. It says that any term that has a free variable $x : 0$ is equivalent to one that immediately case splits on x.

$$\frac{x : 0 \in \Gamma \quad \Gamma \vdash M : C}{\Gamma \vdash M = \textbf{case}\ x\{\} : C}\ 0\eta$$

$$\frac{x : 0 \in \Gamma \quad \Gamma \vdash M : C}{\Gamma \vdash M = \textbf{case}\ x\{\} : C}\ 0\eta$$

Alternatively, we can say that any two terms that have a free variable $x : 0$ are equal:

$$\frac{x : 0 \in \Gamma \qquad \Gamma \vdash M : C \qquad \Gamma \vdash N : C}{\Gamma \vdash M = N : C}\ 0\eta'$$

From a programming/optimization perspective this is simply saying that any two programs are equivalent if they are dead code (i.e., never run).

For disjunction $A \vee B$ we instead have the type $A + B$ which we can think of as a type whose elements are an enumerated datatype with two cases: we have a tag bit, which is either 0 and we additionally have an A or the tag bit is 1 and we have a B. We write this tagging as either $i_1 M$ or $i_2 M$

$$\frac{\Gamma \vdash M : A_1}{\Gamma \vdash i_1 M : A_1 + A_2}\ +I_1 \qquad \frac{\Gamma \vdash M : A_2}{\Gamma \vdash i_2 M : A_1 + A_2}\ +I_2$$

Then the elimination form is pattern matching against the tag:

$$\frac{\Gamma \vdash M : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash N_1 : C \qquad \Gamma, x_2 : A_2 \vdash N_2 : C}{\Gamma \vdash \textbf{case}\ M\{i_1 x_1 \to N_1 | i_2 x_2 \to N_2\} : C}\ +E$$

Then we need congruence rules for each of these as well:

$$\frac{\Gamma \vdash M = N : A_1}{\Gamma \vdash i_1 M = i_1 N : A_1 + A_2}\ +I_1\text{cong} \qquad \frac{\Gamma \vdash M : A_2}{\Gamma \vdash i_2 M = i_2 N : A_1 + A_2}\ +I_2\text{cong}$$

$$\frac{\Gamma \vdash M = M' : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash N_1 = N_1' : C \qquad \Gamma, x_2 : A_2 \vdash N_2 = N_2' : C}{\Gamma \vdash \textbf{case}\ M\{i_1 x_1 \to N_1 | i_2 x_2 \to N_2\} = \textbf{case}\ M'\{i_1 x_1 \to N_1' | i_2 x_2 \to N_2'\} : C}\ +E\text{cong}$$

The β-rule and η-rule for case constructor are given below. The $N_1[M/x_1]$ is a notation for replacing all the occurrence of x_1 in N_1 into M. One may give a formal definition of it recursively.

$$\frac{}{\Gamma \vdash \textbf{case}\ i_1 M\{i_1 x_1 \to N_1 | i_2 x_2 \to N_2\} = N_1[M/x_1]}\ +\beta_1$$

$$\frac{}{\Gamma \vdash \textbf{case}\ i_2 M\{i_1 x_1 \to N_1 | i_2 x_2 \to N_2\} = N_2[M/x_2]}\ +\beta_2$$

$$\frac{x : A_1 + A_2 \in \Gamma \qquad \Gamma \vdash M : C}{\Gamma \vdash M = \textbf{case}\ x\{i_1 x_1 \to M[i_1 x_1/x] | i_2 x_2 \to M[i_2 x_2/x]\} : C}\ +\eta$$

$$\frac{\begin{array}{c} x : A_1 + A_2 \in \Gamma \qquad \Gamma \vdash M : C \qquad \Gamma \vdash N : C \\ \Gamma, x_1 : A_1 \vdash M[i_1 x_1/x] = N[i_1 x_1/x] : C \\ \Gamma, x_2 : A_2 \vdash M[i_2 x_2/x] = N[i_2 x_2/x] : C \end{array}}{\Gamma \vdash M = N : C}\ +\eta'$$

For implication, we introduce a new notation $\lambda x : A.M$ which is like a function of $x : A$ and M is the function body. The A is the type of x which can be omitted if the context makes the type clear. The notation for implication is $\Longrightarrow$.

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \implies B} \implies I$$

Then the elimination form is applying the function $M : A \implies B$ to an input $N : A$ gives us a term $MN : B$. So the logical notion of modus ponens is generalized to the mathematical notion of function application.

$$\frac{\Gamma \vdash M : A \implies B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \implies E$$

These come with corresponding congruence rules

$$\frac{\Gamma, x : A \vdash M = N : B}{\Gamma \vdash (\lambda x : A.M) = (\lambda x : A.N) : A \implies B} \implies I\text{cong}$$

$$\frac{\Gamma \vdash M = M' : A \implies B \quad \Gamma \vdash N = N' : A}{\Gamma \vdash MN = M'N' : B} \implies E\text{cong}$$

The β rule says that applying a λ-function to a term is equivalent to evaluating the body of the function with the input substituted for all the occurrences of the variable x.

$$\frac{\Gamma, x : A \vdash M : B \quad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x : M)N = M[N/x] : B} \implies \beta$$

The η rules says that every function is equivalent to a lambda function that passes the input to it.

$$\frac{\Gamma \vdash M : A \implies B}{\Gamma \vdash M = (\lambda x : Mx) : A \implies B} \implies \eta$$

The alternative lambda rule says that to prove two functions

$$\frac{\Gamma \vdash M : A \implies B \quad \Gamma \vdash N : A \implies B \quad \Gamma, x : A \vdash Mx = Nx : B}{\Gamma \vdash M = N : A \implies B} \implies \eta'$$

Here, the function application is left associative, hence, $M\ x_1\ x_2 = (M\ x_1)\ x_2$. Therefore, we usually don't write parenthesis.

3 Example

Let us consider $\cdot \vdash \top \vee \top$ again. In simple type theory, the theorem is $\cdot \vdash M : 1 + 1$. The three proves given before become,

$$\frac{\dfrac{}{\cdot \vdash () : 1}\, 1I}{\cdot \vdash i_1() : 1 + 1}\, +I_1 \qquad \frac{\dfrac{}{\cdot \vdash () : 1}\, 1I}{\cdot \vdash i_2() : 1 + 1}\, +I_2$$

$$\frac{\dfrac{\dfrac{\dfrac{}{\cdot \vdash () : 1}\, 1I \quad \dfrac{}{\cdot \vdash () : 1}\, 1I}{\cdot \vdash ((), ()) : 1 + 1}\, \times I}{\cdot \vdash \pi_1((), ()) : 1}\, \times E_1}{\cdot \vdash i_1(\pi_1((), ())) : 1 + 1}\, +I_1$$

Now we have a formal sense in which the first and the third proofs are the same: their corresponding terms are equal in our equational theory: $\cdot \vdash i_1(\pi_1((),())) = i_1() : 1+1$. Two proofs are given below,

$$\dfrac{\dfrac{\dfrac{\vdots}{\cdot \vdash \pi_1((),()) : 1}}{\cdot \vdash \pi_1((),()) = () : 1}\,1\eta}{\cdot \vdash i_1(\pi_1((),())) = i_1() : 1+1}\,i_1\ cong \qquad \dfrac{\dfrac{\dfrac{}{\cdot \vdash () : 1}\,1I \quad \dfrac{}{\cdot \vdash () : 1}\,1I}{\cdot \vdash \pi_1((),()) = () : 1}\,\times\beta_1}{\cdot \vdash i_1(\pi_1((),())) = i_1() : 1+1}\,i_1\ cong$$

We can then reformulate a stronger version of the disjunction property for STT: given any term $\cdot \vdash M : 1+1$, either $\cdot \vdash i_1() = M : 1$ or $\cdot \vdash i_2() = M : 1$[1]. Again, non-empty context doesn't guarantee it. For example, if M has a free variable $x : 1+1$, i.e., $x : 1+1 \vdash M : 1+1$ then it is not necessarily the case that $M = i_1()$ or $M = i_2()$. It is possible that $M = x$, i.e., the identity function or $M =$ **case** $x\{i_1x_1 \to i_2() | i_2x_2 \to i_1()\}$, which is the negation of booleans.

4 Takeaway Questions

In this system, there are at least two important questions:

- Consistency of the equational theory. Given $\cdot \vdash i_1() : 1+1$ and $\cdot \vdash i_2() : 1+1$, is can we show that $\cdot \vdash i_1() = i_2() : 1+1$ is not provable?

  This is a reasonable analog of the consistency theorem for IPL (no proof of $\bot$), because if it were the case that $i_1() = i_2()$, then any two programs would be equal:

$$\begin{aligned} M &= \textbf{case}\ i_1()\{i_1x_1 \to M | i_2x_2 \to N\} && (+\beta 1)\\ &= \textbf{case}\ i_2()\{i_1x_1 \to M | i_2x_2 \to N\} && (+\beta 2)\\ &= N \end{aligned}$$

- Canonicity. Given $\cdot \vdash M : 1+1$, can we get either $\cdot \vdash M = i_1() : 1+1$ or $\cdot \vdash M = i_2() : 1+1$? More generally, given $\cdot \vdash M : A+B$ can we find that either there is an M_1 such that $\cdot \vdash M = i_1M_1 : A+B$ or an M_2 with $\cdot \vdash M = i_2M_2 : A+B$?

[1] in fact we need only β rules not η rules to prove this