Natural Deduction and Explicit Proof Objects

Joshua D. Guttman
Worcester Polytechnic Institute

October 12, 2010

Contents

---

1 Consequence Relations

A useful notion that cuts across both semantic (model-oriented) and syntactic (derivation-oriented) issues is the notion of a consequence relation. We will use capital Greek letters like Γ, Δ (Gamma and Delta) to refer to finite

sets of formulas, and lower case Greek letters like ϕ, ψ (phi and psi) to refer to individual formulas. We will save ink by writing Γ, Δ for the set $\Gamma \cup \Delta$, and Γ, ϕ for the set $\Gamma \cup \{\phi\}$, etc.

By $\phi[t_1/x_1, \ldots t_n/x_n]$, we mean the result of plugging in the terms $t_1, \ldots, t_n$ in place of the variables $x_1, \ldots, x_n$. We assume that all the x_i are different variables, and that all of the plugging in happens at once. So, if there are x_2s inside the term t_1, they are not substituted with t_2s. $\Gamma[t_1/x_1, \ldots t_n/x_n]$ means the result of doing the substitutions to all the formulas in Γ.

Definition 1 *Suppose that $\preceq$ is a relation between finite sets of formulas and individual formulas, as in $\Gamma \preceq \phi$. Then $\preceq$ is a* consequence relation *iff it satisfies these properties:*

Reflexivity: $\Gamma, \phi \preceq \phi$*;*

Transitivity: $\Gamma \preceq \phi$ *and* $\Gamma, \phi \preceq \psi$ *imply* $\Gamma \preceq \psi$*;*

Weakening: $\Gamma \preceq \phi$ *implies* $\Gamma, \Delta \preceq \phi$*; and*

Substitution: $\Gamma \preceq \phi$ *implies* $\Gamma[t_1/x_1, \ldots t_n/x_n] \preceq \phi[t_1/x_1, \ldots t_n/x_n]$.

For now, we will focus on formulas with no variables, so **Substitution** will be irrelevant. We will ignore it until later. The **Reflexivity** and **Transitivity** rules ensure that a consequence relation is a partial order, when restricted to sets containing just one assumption. The **Weakening** rule "lifts" this partial order to sets with more members.

We refer to an instance of a relation $\Gamma \preceq \phi$ or any $\Gamma \; R \; \phi$ as a *judgment*.

Both semantic notions such as *entailment* and syntactic notions such as *derivability* give us examples of consequence relations. Suppose we have a notion of *model* such as $\mathbb{M} \models \phi$ as defined in the Dougherty lecture notes, Def. 2.2.2.[1] Then we have a corresponding notion of *(semantic) entailment* defined:

Definition 2 Γ entails ϕ*, written* $\Gamma \Vdash \phi$*, holds iff, for all models* $\mathbb{M}$*:*

If *for each* $\psi \in \Gamma$, $\mathbb{M} \models \psi$,

then $\mathbb{M} \models \phi$.

That is, $\Gamma \Vdash \phi$ means that every model that makes all of the formulas in Γ true makes ϕ true too.

[1] Available at URL `http://web.cs.wpi.edu/~guttman/cs521_website/Dougherty_lecture_notes.pdf`.

$$\frac{\Gamma \vdash \phi \qquad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}$$

Figure 1: ND Introduction and Elimination Rules for $\wedge$

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash \phi \to \psi} \qquad \frac{\Gamma \vdash \phi \to \psi \qquad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

Figure 2: ND Introduction and Elimination Rules for $\to$

Lemma 3 *Entailment is a consequence relation, i.e.* $\Vdash$ *satisfies reflexivity, transitivity, and weakening in Def. 1:*

1. $\Gamma, \phi \Vdash \phi$*;*
2. $\Gamma \Vdash \phi$ *and* $\Gamma, \phi \Vdash \psi$ *imply* $\Gamma \Vdash \psi$*; and*
3. $\Gamma \Vdash \phi$ *implies* $\Gamma, \Delta \Vdash \phi$*.*

We turn next to showing that a particular set of rules for constructing proofs is also a consequence relation.

2 A Derivation System for "Natural Deduction"

We consider the rules suggested by Gerhart Gentzen as a "natural" form of deduction [3]. Gentzen considered these rules natural because they seemed to match directly the meaning of each logical operator.

Each logical operator has one or a couple of rules that allow you to prove formulas containing it as the outermost operator. These are called *introduction* rules. Each operator also has one or a couple of rules that allow you to prove other formulas by extracting the logical content in a formula containing it as outermost operator. They are called *elimination* rules. The introduction rules push formulas up in the partial ordering, while the elimination rules hold them down. Between them, the introduction and elimination rules *fix the meaning* of the logical operators purely in terms of their *deductive power*.

All of this extends to much richer logics, as we will see.

The rules are spread out through Figs. 1–4.

Definition 4 *A* natural deduction derivation *is a tree, conventionally written with the conclusion, the root, at the bottom, such that each judgment is the conclusion of a rule.*

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \phi \vee \psi} \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \phi \vee \psi}$$

$$\frac{\Gamma \vdash \phi \vee \psi \qquad \Gamma, \phi \vdash \chi \qquad \Gamma, \psi \vdash \chi}{\Gamma \vdash \chi}$$

Figure 3: ND Introduction and Elimination Rules for $\vee$

$$\frac{}{\Gamma, \phi \vdash \phi} \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash \phi}$$

Figure 4: ND Axioms and Rule for $\bot$

A derivation is a natural deduction derivation in intuitionist propositional logic *if each rule is one of those shown in Figs. 1–4.*

An example derivation is shown in Fig. 5. It proves $\vdash (p \wedge q) \to (p \vee q)$. There are two questions we'd immediately like answers to. First, do the derivable judgments form a consequence relation? That is, if $\Gamma \preceq \phi$ means that there is a derivation of $\Gamma \vdash \phi$ using our rules, then is $\preceq$ a consequence relation?

Second, how do derivable judgments relate to entailment? If $\Gamma \vdash \phi$ is derivable, then is $\Gamma \Vdash \phi$ true? If $\Gamma \Vdash \phi$ then is $\Gamma \vdash \phi$ derivable?

We can answer the first question affirmatively.

Lemma 5 *The set of derivable judgments* $\Gamma \vdash \phi$ *form a consequence relation.*

Proof: 1. **Reflexivity** holds because $\dfrac{}{\Gamma, \phi \vdash \phi}$ is always a derivation.
2. **Transitivity** holds by Fig. 6.
3. **Weakening** holds by *induction on derivations*:

Base Case Suppose that there is a derivation of $\Gamma \vdash \phi$ consisting only of an application of the Axiom rule. That is, $\phi \in \Gamma$. Thus, $\phi \in \Gamma, \Delta$, so $\dfrac{}{\Gamma, \Delta \vdash \phi}$ is an application of the Axiom rule.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p \wedge q \vdash p \wedge q}}{p \wedge q \vdash p}}{p \wedge q \vdash p \vee q}}{\vdash (p \wedge q) \to (p \vee q)}$$

Figure 5: An Example Derivation

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ d_1\\ \vdots\\ \Gamma, \phi \vdash \psi\end{array}}{\Gamma \vdash \phi \to \psi} \qquad \begin{array}{c}\vdots\\ d_2\\ \vdots\\ \Gamma \vdash \phi\end{array}}{\Gamma \vdash \psi}$$

Figure 6: Composing Derivations for Transitivity

Induction Step Suppose that we are given a derivation d where the last step is an application of one of the rules from Figs. 1–4, and the previous steps generate one or more subderivations d_i, each with conclusion $\Gamma_i \vdash \psi_i$.

Induction hypothesis. Assume that for each of the subderivations d_i, there is a weakened subderivation $W(d_i)$ such that $W(d_i)$ has conclusion $\Gamma_i, \Delta \vdash \psi_i$.

Construct the desired derivation of $\Gamma, \Delta \vdash \phi$ by combining the weakened subderivations $W(d_i)$ using the *same* rule of inference.

□

One part of the second question is easy to answer.

Lemma 6 $\vdash \subseteq \Vdash$.
That is, if $\Gamma \vdash \phi$ *is derivable, then* $\Gamma \Vdash \phi$.

Proof: By induction on derivations. □

On the other hand, $\vdash \subsetneq \Vdash$. There are entailment relations that cannot be derived using these rules.

Challenge. Find a Γ, ϕ such that $\Gamma \Vdash \phi$ but $\Gamma \vdash \phi$ is not derivable using our rules. How would one prove it not derivable?

Question. If these rules do not characterize the semantic entailment relation generated from the classical $\models$, what do they characterize?

3 Derivations with Explicit Proof Objects

In this section we annotate our derivations with a representation of the proofs the derivations construct. This will allow us to manipulate the forms

$$\frac{\Gamma \vdash s : \phi \qquad \Gamma \vdash t : \psi}{\Gamma \vdash \langle s, t \rangle : \phi \wedge \psi}$$

$$\frac{\Gamma \vdash s : \phi \wedge \psi}{\Gamma \vdash \mathsf{fst}(s) : \phi} \qquad \frac{\Gamma \vdash s : \phi \wedge \psi}{\Gamma \vdash \mathsf{scd}(s) : \psi}$$

Figure 7: Rules for Conjunction, with Explicit Proof Objects

$$\frac{\Gamma \vdash s : \phi}{\Gamma \vdash \langle \mathsf{lft}, s \rangle : \phi \vee \psi} \qquad \frac{\Gamma \vdash s : \psi}{\Gamma \vdash \langle \mathsf{rgt}, s \rangle : \phi \vee \psi}$$

$$\frac{\Gamma \vdash s : \phi \vee \psi \qquad \Gamma, x : \phi \vdash t : \chi \qquad \Gamma, y : \psi \vdash r : \chi}{\Gamma \vdash \mathsf{cases}(s, \lambda x \,.\, t, \lambda y \,.\, r) : \chi}$$

Figure 8: Rules for Disjunction, with Explicit Proof Objects

of proofs, and also to treat proof and computation in an overlapping way. The remaining presentation draws heavily on [1, 2, 5].

Definition 7 *By a* context Γ*, we mean a set of pairs consisting of a variable and a formula, such that no variable appears more than once. We write members of* Γ *in the form* $v : \phi$*, so a context takes the form:*

$$v_1 : \phi_1, \ldots v_i : \phi_i.$$

The empty context is permitted, i.e. when $i = 0$ *there are no* $v : \phi$ *pairs.*

Thus, a context is a finite partial function from variables to formulas. Henceforth, we will use Γ for contexts rather than simply sets of formulas.

We annotate each of the rules of Figs. 1–4 with explicit proof objects in such a way that the introduction rules and elimination rules cancel out, yielding Figs.7–10. If a derivation ends with an introduction rule followed by an elimination rule, then we should be able to extract the proof object for the subderivation before the two redundant steps. For instance, the conjunction introduction rule constructs a pair object, from which the two elimination rules can extract the components, i.e. recover the embedded proofs of the individual conjuncts. Likewise, the disjunction introduction rules tag their subderivation $s : \phi$ or $s : \psi$, so that the *cases* construct can

$$\frac{\Gamma, x : \phi \vdash s : \psi}{\Gamma \vdash \lambda x \,.\, s : \phi \to \psi} \qquad \frac{\Gamma \vdash s : \phi \to \psi \qquad \Gamma \vdash t : \phi}{\Gamma \vdash (st) : \psi}$$

Figure 9: Rules for Implication, with Explicit Proof Objects

$$\frac{}{\Gamma, x\colon \phi \ \vdash \ x\colon \phi} \qquad\qquad \frac{\Gamma \ \vdash \ s\colon \bot}{\Gamma \ \vdash \ \mathsf{emp}(s)\colon \phi}$$

Figure 10: Rules for Axioms and Falsehood, with Explicit Proof Objects

$$\begin{array}{rcllllll} s,t,r & ::= & v & | & & & & \\ & & \langle s,t\rangle & | & \mathsf{fst}(s) & | & \mathsf{scd}(s) & | \\ & & (\lambda v\ .\ s) & | & s\ t & | & & \\ & & \langle \mathsf{lft}, s\rangle & | & \langle \mathsf{rgt}, s\rangle & | & \mathsf{cases}(s,t,r) & \\ v & ::= & x & | & y & | & v' & \ldots \end{array}$$

Figure 11: Syntax of Proof Terms

insert it into the appropriate branch proving χ. The rule for falsehood has a less symmetrical role: We call it the "empty promise" rule, on the grounds that it should never be possible to apply it to a closed (variable-free) term s. We give the syntax of explicit proof terms in Fig. 11. Compound terms of certain forms may be *reduced* as indicated in the *reduction rules* shown in Fig. 12. We regard Fig. 12 as giving a kind of operational semantics for an extremely simple programming language. Unfortunately, the most important reduction rule, the "beta rule" β, is not as simple as it looks, as we will explain in Section 4. We will also use some "compile-time" reduction rules that tell us how to push the destructuring operators fst, scd, cases, and function application through a cases operator. They look like rules for code transformation used in some compilers, sometimes called partial evaluation rules. They are needed to give "normal forms" the logical interpretation we want. In proof theory, they are due to Dag Prawitz [6]. These rules are shown in Fig. 13. Their purpose is to enable further reductions. That is, if the consequent t or alternative r contains a constructor, then these rules will

$$\begin{array}{lcll} \mathsf{fst}(\langle s, s'\rangle) & \longrightarrow_r & s & \\ \mathsf{scd}(\langle s, s'\rangle) & \longrightarrow_r & s' & \\ \mathsf{cases}(\langle \mathsf{lft}, s\rangle, t, r) & \longrightarrow_r & t\ s & \\ \mathsf{cases}(\langle \mathsf{rgt}, s\rangle, t, r) & \longrightarrow_r & r\ s & \\ (\lambda v\ .\ s)\ t & \longrightarrow_r & s[t/v] & (\beta) \end{array}$$

Figure 12: Reduction Rules for Proof Terms: Local Rules

$$
\begin{array}{lcl}
\mathsf{fst}(\mathsf{cases}(s,t,r)) & \longrightarrow_r & \mathsf{cases}(s, \mathsf{fst} \circ t, \mathsf{fst} \circ r) \\
\mathsf{scd}(\mathsf{cases}(s,t,r)) & \longrightarrow_r & \mathsf{cases}(s, \mathsf{scd} \circ t, \mathsf{scd} \circ r) \\
\mathsf{cases}(\mathsf{cases}(s,t,r),u,w) & \longrightarrow_r & \mathsf{cases}(s, \lambda v \,.\, \mathsf{cases}(t(v),u,w), \\
 & & \qquad \lambda v \,.\, \mathsf{cases}(r(v),u,w)) \\
u\ (\mathsf{cases}(s,t,r)) & \longrightarrow_r & \mathsf{cases}(s, u \circ t, u \circ r)
\end{array}
$$

Figure 13: Reduction Rules for Proof Terms: Compile-time Rules

allow it to be canceled at compile time, even if it is not yet known whether the test s will be filled in (at run-time) with a value $\langle \mathsf{lft}, s' \rangle$ or $\langle \mathsf{rgt}, s' \rangle$.

4 How to Beta-Reduce

The rule for β-reduction is complicated by the need to rename bound variables when evaluating $s[t/v]$, i.e. plugging a term t into a term s in place of v. The problems arise when t contains a free occurrence of a variable x, but within s, v has an occurrence in the body of some λ-binder $\lambda x \,.\, r$. In this situation, x's free occurrences in t would be "captured" by this λ-binder.

As an example, consider

$$(\lambda v \,.\, (\lambda x \,.\, (x\ v)))\ (x).$$

If we reduce it by plugging in x in place of v, we should *not* obtain $(\lambda x \,.\, (x\ x))$. Nothing here asks us to accept a function x and apply it to itself.

Instead, we would like to change the name of the bound variable x in $\lambda x \,.\, (x\ v)$ before plugging in the meaningful, externally chosen value for x in place of v. Thus, we first *α-convert* $\lambda x \,.\, (x\ v)$ by renaming its x to some new variable z. Now there is no danger and we plug in the free x in place of v without risk of capturing x, obtaining $\lambda z \,.\, (z\ x)$. After all, the renaming can do no harm: the bound variable x or z will really only get its value later when we apply this term to some argument, which will furnish its value.

For this reason, we first explain how to α-convert.

4.1 How to α-Convert

Definition 8 (Free and bound variables) *We define the* free variables *and the* bound variables *of a term recursively:*

$$\begin{aligned} \mathsf{fv}(v) &= \{v\} & \mathsf{bv}(v) &= \emptyset \\ \mathsf{fv}(s\ t) &= \mathsf{fv}(s) \cup \mathsf{fv}(t) & \mathsf{bv}(s\ t) &= \mathsf{bv}(s) \cup \mathsf{bv}(t) \\ \mathsf{fv}(\lambda v\ .\ s) &= \mathsf{fv}(s) \setminus \{v\} & \mathsf{bv}(\lambda v\ .\ s) &= \mathsf{bv}(s) \cup \{v\} \end{aligned}$$

The variables *of s are those of both kinds:* $\mathsf{vars}(s) = \mathsf{fv}(s) \cup \mathsf{bv}(s)$.

Be careful: there are terms in which a variable v occurs both free and bound. For instance, letting s be $(\lambda v\ .\ x\ v)(v)$, $v \in \mathsf{fv}(s) \cup \mathsf{bv}(s)$.

We call an object $\sigma = v \mapsto v'$ a (one-variable) *replacement*. Suppose that $\lambda v\ .\ s$ is a lambda expression whose topmost bound variable v is the same as the argument of σ. Then the *result* of σ on $\lambda v\ .\ s$ is $\lambda v'\ .\ s'$, where s' arises by replacing every free occurrence of v by v' throughout s.

For instance, the result of $\sigma = v \mapsto v'$ on the term

$$\lambda v\ .\ ((\lambda v\ .\ v\ x)\ v)$$

is

$$\lambda v'\ .\ ((\lambda v\ .\ v\ x)\ v').$$

The body of the first term has only one *free* occurrence of v, which has been replaced by v'.

If $v' \notin \mathsf{fv}(\lambda v\ .\ s)$, then the result of $\sigma = v \mapsto v'$ on $\lambda v\ .\ s$ will represent the same function as $\lambda v\ .\ s$. We cannot prove this currently, since we haven't formalized "the function that s represents." However, the claim is a reasonable constraint on any formalization we would give: The name of the bound variable v doesn't matter, since it is gone as soon as we plug in an argument for v. Moreover, we will plug in the argument in the same locations in the result $\lambda v'\ .\ s'$ as in $\lambda v\ .\ s$.

Two terms are α-equivalent if one can be obtained from the other by applying this operation to their parts.

Definition 9 *α-equivalence, written $\equiv_\alpha$, is the smallest relation such that:*

1. *$s \equiv_\alpha s$;*
2. *if $s \equiv_\alpha s'$ and $t \equiv_\alpha t'$, then $(s\ t) \equiv_\alpha (s'\ t')$;*
3. *if $s \equiv_\alpha s'$, then $(\lambda v\ .\ s) \equiv_\alpha (\lambda v\ .\ s')$;*

4. *if* $v' \notin \mathsf{fv}(\lambda v\ .\ s)$, *and* $\lambda v'\ .\ s'$ *is the result of* $v \mapsto v'$ *on* $\lambda v\ .\ s$, *then*

$$(\lambda v\ .\ s) \equiv_\alpha (\lambda v'\ .\ s').$$

Lemma 10 *Let* s, t *be terms.*

1. $\mathsf{bv}(s)$ *and* $\mathsf{fv}(s)$ *are finite.*
2. *If* t *is a subterm of* s, *then* $\mathsf{fv}(t) \subseteq \mathsf{vars}(s)$.
3. *Let* $F \subseteq \mathcal{V}$ *be a finite set of variables. There is an* s' *such that* $s' \equiv_\alpha s$ *and* $\mathsf{bv}(s') \cap F = \emptyset$.
4. *There is a* s' *such that* $s' \equiv_\alpha s$ *and* $\mathsf{bv}(s') \cap \mathsf{fv}(t) = \emptyset$.
5. *There is an* s' *such that* $s' \equiv_\alpha s$ *and: (i) any two* λ*-expressions within* s' *bind different variables and (ii)* $\mathsf{bv}(s') \cap \mathsf{fv}(s') = \emptyset$.

A term s' satisfying the conditions (i,ii) in Clause 5 is said to be in "standard form." People frequently assume, in the middle of a proof, that any term they're working with is in standard form. Clause 5 justifies this, because even if the term *isn't* in standard form, some α-equivalent term is.

Proof: 1. By induction on the structure of terms. *Base case.* If s is a variable v, then $\mathsf{bv}(v) = \emptyset$ and $\mathsf{fv}(v) = \{v\}$, both of which are finite.

Induction step. Suppose (i.h.) that $\mathsf{bv}(s)$ and $\mathsf{fv}(s)$ are finite, and so are $\mathsf{bv}(t)$ and $\mathsf{fv}(t)$.

Then so are $\mathsf{fv}(s\ t)$ and $\mathsf{bv}(s\ t)$, since the union of two finite sets is finite. $\mathsf{fv}(\lambda v\ .\ s)$ is a subset of $\mathsf{fv}(s)$, so finite. $\mathsf{bv}(\lambda v\ .\ s)$ is the union of the finite set $\{v\}$ with a finite set, so also finite.

2. For every path p to an occurrence of a free variable v in t, consider $q^\frown p$, where q is a path in s to an occurrence of t. Then $q^\frown p$ is a path to v. It is a path to a bound occurrence if q traverses some λv, and otherwise it is a path to a free occurrence.

3. Let $\langle v_i \rangle_{i \in \mathbb{N}}$ be an infinite sequence that enumerates all the variables $v \in \mathcal{V}$. For any term t, let $\mathsf{fresh}(t) = \max\{i\colon v_i \in F \cup \mathsf{vars}(t)\}$: so $\mathsf{fresh}(t)$ is well-defined because F and $\mathsf{vars}(t)$ are finite. Now argue by induction on the structure of terms.

Base case. If s is a variable v, then $\mathsf{bv}(v) = \emptyset$, so s' can be v.

Induction step. Suppose given t_0 and t_1. Assume (i.h.) that there are t_0' and t_1' such that $t_i' \equiv_\alpha t_i$ and $\mathsf{bv}(t_i') \cap F = \emptyset$.

If $s = (t_0\ t_1)$, then the desired s' is $s' = (t_0'\ t_1')$.

If $s = \lambda v\,.\,t_0$, and $v \notin F$, then the desired s' is $s' = \lambda v\,.\,t_0'$.
Otherwise, let $v' = \mathsf{fresh}(t_0')$, and let t' result from s' under $v \mapsto v'$.
4. Apply Clause 3 to $F = \mathsf{fv}(t)$.
5. Use $v' = \mathsf{fresh}(t_0')$ as in Clause 3, starting with $F = \mathsf{vars}(s)$. □

4.2 How to β-Reduce

Terminology varies slightly among the different notions introduced in the next definition.

Definition 11 *Let $s = \lambda v\,.\,s_0$, and let t be a term. Let $s' = \lambda v'\,.\,s_0'$ be chosen in some canonical way such that $s' \equiv_\alpha s$ and $\mathsf{bv}(s') \cap \mathsf{fv}(t) = \emptyset$. Then $s_0'[t/v']$ is defined to be the result of replacing every free occurrence of v' with the term t.*

We say $(s\ t)$ β-reduces to *r iff $r \equiv_\alpha s_0'[t/v']$, and we write $(s\ t) \to_b r$.*
We write $s \to_{\beta 1} t$ for the least relation such that:

1. *$s \to_b t$ implies $s \to_{\beta 1} t$;*
2. *$s_0 \to_{\beta 1} s_1$ implies:*
   (a) *$(s_0\ t) \to_{\beta 1} (s_1\ t)$;*
   (b) *$(t\ s_0) \to_{\beta 1} (t\ s_1)$; and*
   (c) *$\lambda v\,.\,s_0 \to_{\beta 1} \lambda v\,.\,s_1$.*

$\to_\beta$ is the least relation which is transitive and closed under $\equiv_\alpha$ and includes $\to_{\beta 1}$. Thus, $s \to_\beta t$ holds whenever $s \equiv_\alpha t$, or $s \to_{\beta 1} t$, or there is an r such that $s \to_\beta r$ and $r \to_\beta t$.

The relation $\to_b$ gives the correct meaning for the last line of Fig. 12.

5 Reduction and Typing for Proof Terms

Throughout Section 4, we have ignored the typing discipline and the other forms of proof terms. The whole discussion there extends mechanically to the other proof terms, and remains equally applicable if typing is altered or discarded.

5.1 Typing

The type structure ensures interesting additional properties. We here follow Barendregt [1, Section 3.1].

We will write henceforth $\Gamma \vdash s\colon \phi$ to mean that there is a derivation using the explicit proof rules (Figs. 7–10) where $\Gamma \vdash s\colon \phi$ is the last line of the derivation (the root of the tree).

Lemma 12 (Context Lemma) *Suppose that Γ, Γ' are contexts.*

1. *If $\Gamma \subseteq \Gamma'$, then $\Gamma \vdash s\colon \phi$ implies $\Gamma' \vdash s\colon \phi$.*
2. *If $\Gamma \vdash s\colon \phi$, then $\mathsf{fv}(s) \subseteq \mathsf{dom}(\Gamma)$.*
3. *$\Gamma \vdash s\colon \phi$, and $\Gamma' = \Gamma \restriction \mathsf{fv}(s)$, then $\Gamma' \vdash s\colon \phi$*

Proof: By induction on derivations. Clause 1 is the analog to Weakening (Lemma 5) in the explicit proof formalism. □

Clause 2 of this lemma has a significant consequence: it says that the closed proof objects—the ones with no free variables—are very important. They are the only ones that prove a conclusion with no premises, i.e. with $\Gamma = \emptyset$. The other clauses say that premises in Γ with variables not appearing free in the right hand side do no good (Clause 3) and do no harm (Clause 1).

We next summarize what must be true when $\Gamma \vdash s\colon \phi$, as a function of the syntactic form of s as a proof term. Naturally, for a given proof term, we can use this lemma repeatedly on its subexpressions to unfold the derivation from the bottom up.

Lemma 13 (Generation) *Suppose that $\Gamma \vdash s_0\colon \phi$. If s_0 is of the form:*

v, **then** *$x\colon \phi \in \Gamma$;*

$\langle s, t\rangle$, **then** *$\phi = \phi_1 \wedge \phi_2$ and $\Gamma \vdash s\colon \phi_1$ and $\Gamma \vdash t\colon \phi_2$;*

$\mathsf{fst}(s)$, **then** *for some ψ, $\Gamma \vdash s\colon \phi \wedge \psi$;*

$\mathsf{scd}(s)$, **then** *for some ψ, $\Gamma \vdash s\colon \psi \wedge \phi$;*

$\lambda v\,.\,s$, **then** *$\phi = \phi_1 \to \phi_2$ and $\Gamma, v\colon \phi_1 \vdash \phi_2$;*

$(s\ t)$, **then** *for some ψ, $\Gamma \vdash s\colon \psi \to \phi$ and $\Gamma \vdash t\colon \psi$;*

$\langle \mathsf{lft}, s\rangle$, **then** *$\phi = \phi_1 \vee \phi_2$ and $\Gamma \vdash s\colon \phi_1$;*

$\langle \mathsf{rgt}, s\rangle$, **then** *$\phi = \phi_1 \vee \phi_2$ and $\Gamma \vdash s\colon \phi_2$;*

$\mathsf{cases}(s,t,r)$**, then** *for some disjunction* $\psi_1 \vee \psi_2$, $\Gamma \vdash s\colon \psi_1 \vee \psi_2$, *and* $\Gamma \vdash t\colon \psi_1 \to \phi$, *and* $\Gamma \vdash r\colon \psi_2 \to \phi$.

Proof: The proof is essentially by pattern matching the form of each rule's conclusion. The only wrinkle is in the last clause, where we have an implication $\psi_1 \to \phi$ because we have lambda-bound the variable in each subsidiary derivation in the or-elimination rule. □

Lemma 14 (Subterms typable) *Suppose that* $\Gamma \vdash s\colon \phi$, *and* t *is a subterm of* s. *Then for some* Γ' *and some* ψ, $\Gamma' \vdash t\colon \psi$.

Indeed, with our current rules, $\Gamma' \supseteq \Gamma$ *and* ψ *is a subformula of* ϕ.

Proof: By induction on the derivations. □

Note, however, that there are some terms that are not typable with any Γ and ϕ. An example is $(x\ x)$. So no term with any part of this form is typable in this system.

The following lemma shows that proof terms are generic. Derivations do not depend on the fact that any formula is atomic. Thus, the same explicit proof term can have many typings (and prove many theorems) if an atomic formula part is replaced by a compound.

Lemma 15 (Generic proof objects) *Suppose for some atomic formula* p, $\Gamma, x\colon p \vdash s\colon \phi$. *Then for any* ψ, $\Gamma, x\colon \psi \vdash s\colon \phi'$, *where* ϕ' *results from* ϕ *by replacing every occurrence of* p *with the formula* ψ.

Lemma 16 *Suppose that* $\Gamma, x\colon \phi \vdash s\colon \psi$ *and* $\Gamma \vdash t\colon \phi$. *Then* $\Gamma \vdash s[t/x]\colon \psi$.

This leads directly to a crucial theorem about the reduction relation, relating reduction to derivations (or type judgments, which is the same thing).

5.2 Reduction

Throughout Section 4, we ignored the other proof terms, generated using pairing $\langle \cdot, \cdot \rangle$ and the functions $\mathsf{fst}, \mathsf{scd}, \mathsf{cases}$. The discussion there can be easily extended to the larger language. In particular, the definition of $\equiv_\alpha$ extends mechanically through the remaining proof terms, with the same properties. The reduction relation takes the form:

Definition 17 *The one-step reduction relation* $\longrightarrow_1$ *is the smallest relation such that:*

1. *If* $s \longrightarrow_r t$ *as defined in Figs. 12–13, then* $s \longrightarrow t$*;*

2. $s_0 \longrightarrow s_1$ *implies:*

(i)	$(s_0\ t) \longrightarrow (s_1\ t)$;	*(ii)*	$(t\ s_0) \longrightarrow (t\ s_1)$;
(iii)	$\lambda v\ .\ s_0 \longrightarrow \lambda v\ .\ s_1$;	*(iv)*	$\mathsf{fst}(s_0) \longrightarrow \mathsf{fst}(s_1)$;
(v)	$\mathsf{scd}(s_0) \longrightarrow \mathsf{scd}(s_1)$;	*(vi)*	$\langle s_0, t\rangle \longrightarrow \langle s_1, t\rangle$;
(vii)	$\langle t, s_0\rangle \longrightarrow \langle t, s_1\rangle$	*(viii)*	$\mathsf{cases}(s_0, t, r) \longrightarrow \mathsf{cases}(s_1, t, r)$;
(ix)	$\mathsf{cases}(t, s_0, r) \longrightarrow \mathsf{cases}(t, s_1, r)$;	*(x)*	$\mathsf{cases}(t, r, s_0) \longrightarrow \mathsf{cases}(t, r, s_1)$.

The reduction relation $\longrightarrow^*$ *is the least relation which is transitive and closed under* $\equiv_\alpha$ *and includes* $\longrightarrow$*.*

Thus, $s \longrightarrow^* t$ holds whenever $s \equiv_\alpha t$, or $s \longrightarrow t$, or there is an r such that $s \longrightarrow^* r$ and $r \longrightarrow^* t$.

Theorem 18 (Subject Reduction) *Suppose that* $s \longrightarrow^* t$ *and* $\Gamma \vdash s\colon \phi$*. Then* $\Gamma \vdash t\colon \phi$ *also.*

This is also known as a *type preservation theorem*, since it says that the type ϕ is preserved no matter how we reduce s. The phrase "subject reduction" comes from the idea that in the judgment $s\colon \phi$, s is the "subject" and ϕ is the "predicate." It says that when the subject reduces, the predicate still applies. Theorems of this form are ubiquitous in programming language semantics.

6 Normalization

A term t is *a normal form* with respect to a reduction relation such as our $\longrightarrow$ if it has no parts that can be reduced, i.e. there is no $s \neq t$ such that $t \longrightarrow s$. If $s \longrightarrow t$ and t is a normal form, then we think of t as a value that the program s computes. If in addition t is closed—it has no free variables—this is especially appropriate. Particularly in our context where it means that t proves a theorem ϕ with no premises: $\emptyset \vdash t\colon \phi$. In this section, we will prove that every well-typed term has a normal form.

Theorem 19 (Normal Form) *If* $\Gamma \vdash s\colon \phi$*, then there is a normal form* t *such that* $s \longrightarrow^* t$*.*

A term t is *closed* if it has no free variables: $\mathsf{fv}(t) = \emptyset$.

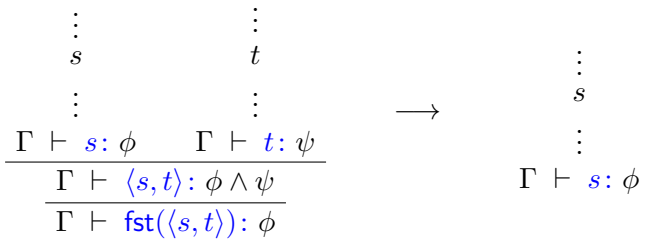

Figure 14: Reducing an Unnecessary $\wedge$-Introduction/Elimination

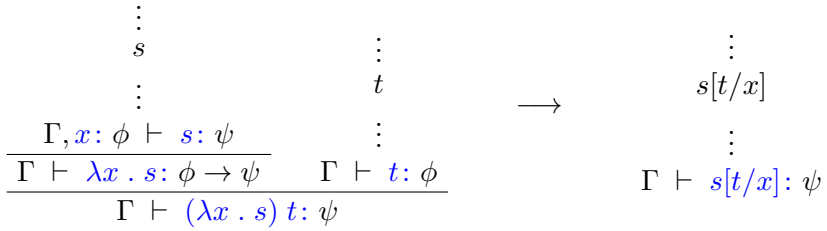

Figure 15: Reducing an Unnecessary $\rightarrow$-Introduction/Elimination

Corollary 20 *If ϕ is derivable from Γ, then for some normal t, $\Gamma \vdash t : \phi$. If ϕ is derivable from $\emptyset$, then for some closed normal t, $\vdash t : \phi$.*

If a derivation system satisfies the analogue of Thm. 19, then a great deal of information about what it proves follows by considering the forms of its closed normal forms. This is a crucial technique for proving consistency theorems, and also for proving that one derivation system is a conservative extension of another.

We first illustrate the proof-theoretic meaning of the reduction steps. Each reduction step determines a proof simplification. As a simplest example, the rule $\mathsf{fst}(\langle s, s' \rangle) \longrightarrow_r s$ says that a proof that first introduces a conjunction and then eliminates it simplifies to the embedded proof of the conjunct (Fig. 14). Eliminating an implication is somewhat more complex, because there is an implicit induction on the proof $s : \psi$ to ensure that anywhere the premise $x : \phi$ may have been used, the proof $t : \phi$ may in fact be used instead (Fig. 15).

We can also illustrate the compile-time rules for cases. In Fig. 16, we see that a derivation of the form $\mathsf{fst}(\mathsf{cases}(s, \lambda x \,.\, t, \lambda y \,.\, r))$—which definitely has no reduction, except possibly within s, t, or r—is replaced with a re-

$$\dfrac{\dfrac{\Gamma \vdash s\colon \phi \vee \psi \qquad \Gamma, x\colon \phi \vdash t\colon \chi_1 \wedge \chi_2 \qquad \Gamma, y\colon \psi \vdash r\colon \chi_1 \wedge \chi_2}{\Gamma \vdash \mathsf{cases}(s, \lambda x\,.\,t, \lambda y\,.\,r)\colon \chi_1 \wedge \chi_2}}{\Gamma \vdash \mathsf{fst}(\mathsf{cases}(s, \lambda x\,.\,t, \lambda y\,.\,r))\colon \chi_1}$$

$$\longrightarrow$$

$$\dfrac{\Gamma \vdash s\colon \phi \vee \psi \qquad \dfrac{\Gamma, x\colon \phi \vdash t\colon \chi_1 \wedge \chi_2}{\Gamma, x\colon \phi \vdash \mathsf{fst}(t)\colon \chi_1} \qquad \dfrac{\Gamma, y\colon \psi \vdash r\colon \chi_1 \wedge \chi_2}{\Gamma, y\colon \phi \vdash \mathsf{fst}(r)\colon \chi_1}}{\Gamma \vdash \mathsf{cases}(s, \lambda x\,.\,\mathsf{fst}(t), \lambda y\,.\,\mathsf{fst}(r))\colon \chi_1}$$

Figure 16: Disjunction Elimination Followed by Conjunction Elimination

duction that can reduce, if either t or r is of the form $\langle u_1, u_2\rangle$. The most interesting case of this kind is definitely one disjunction elimination followed by another. By this we mean, a disjunction elimination which furnishes the *major premise*, the disjunction that will get eliminated, for a subsequent disjunction elimination (Fig. 17). Observe that in the "reduced" derivation, the disjunction $\phi \vee \psi$ no longer *separates* $\chi_1 \vee \chi_2$ from the top of the derivation. Thus, if we apply this transformation everywhere, no eliminated disjunction will separate any other eliminated disjunction from the top of the derivation.

This is in fact very important. It establishes a *subformula* property.

Definition 21 *A derivation d with conclusion $\Gamma \vdash s\colon \phi$ has the subformula property if for every judgment $\Delta \vdash t\colon \rho$ appearing in d, either*

1. *ρ is a subformula of ϕ, or*
2. *for some $v\colon \psi \in \Gamma$, ρ is a subformula of ψ.*

The minor premises *of the disjunction elimination rule are the judgments $\Gamma, x\colon \phi \vdash \chi$ and $\Gamma, y\colon \psi \vdash \chi$ in its premises. The* minor premise *of the implication elimination rule is the derivation of $\Gamma \vdash t\colon \phi$. All other premises are* major premises*.*

Thus, all the premises of an introduction rule are major, and a premise of an elimination rule is major if it contains the connective being eliminated. We note several properties of normal derivations:

Lemma 22 *Suppose that d is a normal derivation of $\Gamma \vdash s\colon \phi$, and p is a path upwards between any two judgments in the tree.*

$$\dfrac{\dfrac{\Gamma \vdash s\colon \phi \vee \psi \qquad \dfrac{\Gamma, x\colon \phi \vdash t\colon \chi_1 \vee \chi_2 \qquad \Gamma, y\colon \psi \vdash r\colon \chi_1 \vee \chi_2}{\Gamma \vdash \mathsf{cases}(s, \lambda x\,.\,t, \lambda y\,.\,r)\colon \chi_1 \vee \chi_2}}{\Gamma \vdash f\colon \rho} \qquad \begin{array}{c}\Gamma, z\colon \chi_1 \vdash u\colon \rho \\ \Gamma, v\colon \chi_2 \vdash w\colon \rho\end{array}}{}$$

$$\longrightarrow$$

$$\dfrac{\Gamma \vdash s\colon \phi \vee \psi \qquad \dfrac{\Gamma, x\colon \phi \vdash t\colon \chi_1 \vee \chi_2 \qquad \begin{array}{c}\Gamma, x\colon \phi, z\colon \chi_1 \vdash u\colon \rho \\ \Gamma, x\colon \phi, v\colon \chi_2 \vdash w\colon \rho\end{array}}{\Gamma, x\colon \phi \vdash g_1\colon \rho}}{\Gamma \vdash f'\colon \rho}$$

$$f = \mathsf{cases}(\mathsf{cases}(s, \lambda x\,.\,t, \lambda y\,.\,r), \lambda z\,.\,u, \lambda v\,.\,w)$$

$$g_1 = \mathsf{cases}(t, \lambda z\,.\,u, \lambda v\,.\,w) \quad g_2 = \mathsf{cases}(r, \lambda z\,.\,u, \lambda v\,.\,w) \quad f' = \mathsf{cases}(u, g_1, g_2)$$

Figure 17: Disjunction Elimination Followed by Disjunction Elimination (analogous derivation of $\Gamma, y\colon \psi \vdash g_2\colon \rho$ is omitted; $x, y \notin \mathsf{fv}(u, w)$)

1. *If p traverses only major premises, then p never traverses any instance of an introduction rule after any instance of an elimination rule.*
2. *If p traverses only elimination rules, and p traverses a disjunction elimination inference, then it is below any other elimination rule.*
3. *If p traverses only elimination rules, then p traverses at most one disjunction elimination major premise.*
4. *If p traverses only introduction rules, then each successive right hand side is a subformula of the one below it.*

From this the subformula property follows:

Theorem 23 *Every normal derivation has the subformula property.*

Proof: 1. Conclusion of an introduction rule: subformula of ϕ.
2. Major premise of an elimination rule: subformula of some ψ in Γ.
3. Minor premise of $\rightarrow$-elimination: subformula of the major premise.
4. Minor premise of $\vee$-elimination: subformula of ϕ. □

Corollary 24 *These rules are* consistent*: there is no derivation $\vdash s\colon \bot$.*

Proof: Suppose that $\vdash s_0 : \bot$, and let t_0 be minimal in the subtree ordering among all normal t such that $\vdash t : \bot$.

The last inference in t_0 is not an elimination rule, because no elimination rule has a major premise that is a subformula of $\bot$. The last inference is not an introduction rule apart from emp, because the conclusion contains no connective. The last inference is not emp, because then we would have:

$$\frac{\vdash s : \bot}{\vdash t_0 : \bot}$$

contradicting the minimality of t_0.

Finally, $\vdash t_0 : \bot$ is not an instance of the Axiom Rule. □

7 Richer Proof Objects: Primitive Recursion

In this section, we stipulate that there is a particular atomic proposition N. Think of N as saying that there are natural numbers. Since this is true, we will enrich our proof objects to provide proofs of N. In fact, rather a lot of proofs of N. The interest of this proof system is not in what you can prove, but instead in what the proofs are, since the proofs provide a much more interesting notion of computation than the previous sections. This system is officially known as "Gödel's system T," since Kurt Gödel introduced it in a paper from 1958 [4].

The new constants in the syntax of proof terms are $0, S$, and ρ. 0 and S represent 0 and the successor function. 0 is a proof-by-exhibit that there are natural numbers, and if t is such a proof-by-exhibit, then so is $S(t)$.

The operator ρ provides proof by primitive recursion. That is, $rho(0, a, f)$ evaluates to the "seed value" a, and $\rho(S(t), a, f)$ evaluates to the result of applying f once more than $\rho(t, a, f)$. The trick to the current system is that f may be a function $N \to N$, but it may also be a function of "higher type" such as $(N \to N) \to (N \to N)$, and primitive recursion at higher types is rather powerful. It allows us to define functions that would not be definable using ρ only at the lowest type $N \to N$, including fast-growing functions such as Ackermann's function.

The typing rules for $0, S, \rho$ are in Fig. 18. Observe that ρ produces values of any type σ, and requires only that the seed value a is of that type, and the iterated function $f : \sigma \to \sigma$ preserves that type. The first argument t must be a term of type N, saying how many times to iterate f starting from a. Of course, t may contain free variables. The reduction rules in Fig. 19 for $0, S, \rho$ formalize the explanation we have given for ρ's meaning. Using

$$\frac{}{\Gamma \vdash 0 : N} \qquad \frac{\Gamma \vdash t : N}{\Gamma \vdash S(t) : N}$$

$$\frac{\Gamma \vdash t : N \qquad \Gamma \vdash a : \sigma \qquad \Gamma \vdash f : \sigma \to \sigma}{\Gamma \vdash \rho(t, a, f) : \sigma}$$

Figure 18: Typing Rules for $0, S, \rho$

$$\begin{array}{lcl} \rho(0, a, f) & \longrightarrow_r & a \\ \rho(S(t), a, f) & \longrightarrow_r & f(\rho(t, a, f)) \end{array}$$

Figure 19: Reduction Rules for ρ

these operators, we can define addition, multiplication, and exponentiation:

$$\begin{array}{rcl} m + n & = & \rho(n, m, S) \\ m * n & = & \rho(n, 0, (\lambda v \,.\, v + m)) \\ m^n & = & \rho(n, 1, (\lambda v \,.\, v * m)) \end{array}$$

Now we can compute e.g. $S(S(0)) + S(S(S(0)))$ as:

$$\begin{array}{l} \rho(S(S(S(0))), S(S(0)), S) \longrightarrow S(\rho(S(S(0)), S(S(0)), S)) \longrightarrow \\ S(S(\rho(S(0), S(S(0)), S))) \longrightarrow S(S(S(\rho(0, S(S(0)), S)))) \longrightarrow S(S(S(S(S(0))))) \end{array}$$

meaning that $2 + 3 = 5$.

In this system, we are not restricted to definitions like this, where the function argument to ρ is of type $N \to N$. We can also, for instance, construct functions by primitive recursion in which the function argument is of type $(N \to N) \to (N \to N)$, which we can then apply to a function $f : N \to N$ and a value $n : N$ to obtain a result of type N. For instance, the function

$$\lambda g \,.\, \lambda n \,.\, \rho(n, 1, g)$$

will iterate its function argument g starting from 1, applying it n times. Thus, if we write:

$$\lambda x \,.\, \rho(x, (\lambda m \,.\, m * 2), (\lambda g n \,.\, \rho(n, 1, g))),$$

then this is a function that—given x—produces a function doing x-nested

doubling. That is,

$$\begin{array}{rcl}
\rho(0,(\lambda m\,.\,m*2),(\lambda gn\,.\,\rho(n,1,g))) & \longrightarrow & \lambda m\,.\,m*2 \\
\rho(1,(\lambda m\,.\,m*2),(\lambda gn\,.\,\rho(n,1,g))) & \longrightarrow & \lambda n\,.\,\rho(n,1,\lambda m\,.\,m*2) \\
 & \longrightarrow & \lambda n\,.\,2^n \\
\rho(2,(\lambda m\,.\,m*2),(\lambda gn\,.\,\rho(n,1,g))) & \longrightarrow & \lambda n\,.\,\rho(n,1,\rho(1,(\lambda m\,.\,m*2), \\
 & & \qquad\qquad (\lambda gn\,.\,\rho(n,1,g)))) \\
 & = & \lambda n\,.\,\rho(n,1,\lambda m\,.\,2^m).
\end{array}$$

As the first argument increases, this quickly becomes a very fast-growing function of the second argument. Indeed,

$$\lambda k\,.\,(\rho(k,(\lambda m\,.\,m*2),(\lambda gn\,.\,\rho(n,1,g)))\;k)$$

can be shown to grow faster than any function that can be defined using only ρ applied to functions $f\colon N \to N$ (no matter how many times).

Although this system encodes powerful computations, as a logic it is expressively puny. It contains many interesting proofs, but they all prove a small number of boring theorems, such as N, which says that there are natural numbers.

8 Predicate Logic

We turn now to a logic that expresses more interesting propositions, which we will combine with numbers and recursion in the next section.

Definition 25 *Let $\mathcal{V}$ be an infinite set of objects called* variables*, disjoint from all the sets introduced below.*

1. *A* signature Σ *consists of two disjoint sets:*

   Function constants $\mathcal{F}$*, written c_0, c_1, f_0, f_1, g, etc. and*

   Relation constants $\mathcal{R}$*, written P_0, Q_1, R_0, etc.*

   together with a function $\Sigma\colon \mathcal{F} \cup \mathcal{R} \to \mathbb{N}$, the natural numbers.

   $\Sigma(f)$ or $\Sigma(R)$ is called the arity *or* degree *of f, R. If a function constant has degree 0, it is called an* individual constant*. If a relation constant has degree 0, it is called a* propositional constant*.*

2. *The* terms over Σ *are defined recursively:*

A variable $v \in \mathcal{V}$ *is a term;*

A function symbol $f \in \mathcal{F}$*, together with* n *terms* $t_1, \ldots, t_n$ *form a term* $f(t_1, \ldots, t_n)$*, if* $n = \Sigma(f)$*. In particular, if* $0 = \Sigma(c)$*, then* $c()$ *is a term, which we will in fact write* c*.*

3. *The* atomic formulas over Σ *are defined:*

   A relation symbol $R \in \mathcal{R}$ *together with* n *terms* $t_1, \ldots, t_n$ *form a term* $R(t_1, \ldots, t_n)$*, if* $n = \Sigma(R)$*. In particular, if* $0 = \Sigma(P)$*, then* $P()$ *is a term, which we will in fact write* P*.*

4. *The* formulas over Σ *are defined recursively:*

   An atomic formula $R(t_1, \ldots, t_n)$ *is a formula;*

   A conjunction $\phi \wedge \psi$ *of formulas* ϕ, ψ *is a formula;*

   A disjunction $\phi \vee \psi$ *of formulas* ϕ, ψ *is a formula;*

   An implication $\phi \to \psi$ *of formulas* ϕ, ψ *is a formula;*

   A universal quantification $\forall v \,.\, \phi$ *is a formula, assuming* $v \in \mathcal{V}$ *and* ϕ *is a formula; and*

   An existential quantification $\exists v \,.\, \phi$ *is a formula, assuming* $v \in \mathcal{V}$ *and* ϕ *is a formula.*

   We use parentheses as needed to avoid ambiguity, e.g. $\phi \wedge (\psi \vee \chi)$*.*

5. *A variable* $v \in \mathcal{V}$ occurs free *in the* term v*, and it occurs free in* $f(t_1, \ldots, t_n)$ *if it occurs free in any of the* t_i*. No variable* occurs bound *in any term.*

6. v occurs free *in the* formula $R(t_1, \ldots, t_n)$ *if it occurs free in any of the terms* t_i*. It occurs free in* $\phi \wedge \psi$*,* $\phi \vee \psi$*,* $\phi \to \psi$ *if it occurs free in either* ϕ *or* ψ*.*

   It occurs free in $\forall x \,.\, \phi$ *or* $\exists x \,.\, \phi$ *if it occurs free in* ϕ *and* v *and* x *are different variables.*

7. v occurs bound *in* formula $\phi \wedge \psi$*,* $\phi \vee \psi$*,* $\phi \to \psi$ *if it occurs free in either* ϕ *or* ψ*.* v *does not occur bound in* $R(t_1, \ldots, t_n)$*.*

   It occurs bound in $\forall x \,.\, \phi$ *or* $\exists x \,.\, \phi$ *if it occurs bound in* ϕ *or if* v *and* x *are the same variable.*

8. *We write* $\mathsf{fv}(\phi)$ *and* $\mathsf{bv}(\phi)$ *for the free and bound variables (resp) of* ϕ*.*

$$\frac{\Gamma \vdash s : \phi}{\Gamma \vdash (\lambda x \,.\, s) : \forall z \,.\, \phi[z/x]}\ (1) \qquad \frac{\Gamma \vdash s : \forall x \,.\, \phi}{\Gamma \vdash (s\ t) : \phi[t/x]}$$

Figure 20: Rules for $\forall$. (1) requires $x \notin \mathsf{fv}(\Gamma)$, and either z is x or $z \notin \mathsf{fv}(\phi)$

$$\frac{\Gamma \vdash s : \phi[t/x]}{\Gamma \vdash \langle E, t, s\rangle : \exists x \,.\, \phi} \qquad \frac{\Gamma \vdash s : \exists x \,.\, \phi \qquad \Gamma, v : \phi \vdash t : \psi}{\Gamma \vdash \mathsf{ecase}(s, (\lambda x v \,.\, t)) : \psi}\ x \notin \mathsf{fv}(\Gamma, \psi)$$

Figure 21: Rules for $\exists$

Notice that $v \in \mathsf{fv}(\phi) \cap \mathsf{bv}(\phi)$ is possible, for instance $P(v) \wedge (\forall v \ . \ \psi)$. Propositional logic is certainly a special case of this, namely where $\Sigma(R) = 0$ for each $R \in \mathcal{R}$. We add introduction and elimination rules for $\forall$ and $\exists$ to the rules available in propositional logic, as shown in Figs. 20–21.

The proof terms for the universal quantifier simply reuse the λ operator.[2] A proof of a universal formula $\forall x \,.\, \phi$ is a function that, given an instance t for x, produces a proof of $\phi[t/x]$. For instance, a proof that ends with an application of $\forall$-introduction, if followed by an application of $\forall$-elimination using t, really does yield a proof of $\phi[t/x]$.

Namely, we take the subderivation up until immediately before the $\forall$-introduction, and substitute t in place of x throughout the that derivation. Since x does not appear free in Γ, the resulting object remains a derivation. Hence, this syntactic operation constructs direct proofs of $\phi[t/x]$ given a direct proof of $\forall x \,.\, \phi$ and a choice of t.

The proof term for an existential quantifier is however something new, although reminiscent of disjunction proof terms. We have the additional local reduction rule shown in Fig. 22.

References

[1] Henk Barendregt. Lambda calculi with types. *Handbook of logic in computer science*, 2:117–309, 1992.

[2] We here ignore the $[z/x]$ substitution, i.e. we assume x and z are the same variable.

$$\mathsf{ecase}(\langle \mathsf{E}, t, s\rangle, r) \quad \longrightarrow_r \quad (r\ t)\ s$$

Figure 22: Local reduction rule for $\exists$

[2] Henk Barendregt and Silvia Ghilezan. Lambda terms for natural deduction, sequent calculus and cut elimination. *Journal of Functional Programming*, 10(01):121–134, 2000.

[3] Gerhard Gentzen. Investigations into logical deduction. In Manfred Szabo, editor, *Complete Works of Gerhard Gentzen*. North Holland, 1969. Originally published in *Mathematische Zeitschrift*, 1934–1935.

[4] K. Godel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12(280-287):12, 1958. Translated in *Collected Works, v. 2*, 1990, Oxford University Press.

[5] Ralph Loader. Notes on simply typed lambda calculus. Technical Report ECS-LFCS-98-381, University of Edinburgh, 1998. At URL `http://www.lfcs.inf.ed.ac.uk/reports/98/ECS-LFCS-98-381/`.

[6] Dag Prawitz. *Natural Deduction. A Proof-Theoretic Study*. Almqvist and Wiksell, Stockholm, 1965.