Chapter 1

Logic and Proofs [Lecture 1]

Version of February 8, 2021

Summary Throughout this course, we will need to rigorously show a number of statements involving complicated logical formulas. To aid in this, we give here a short introduction to logic and semi-formal proofs. At the same time, this introduction will serve as a first example of presenting a deductive system by means of inference rules.

A *semi-formal proof* is a proof that is written in natural language, but is implicitly guided by the formal proof rules of natural deduction. This means that a proof written in this style can usually be transcribed relatively directly into a mechanized proof verifier, such as Twelf.

These notes give a brief overview of natural deduction and semi-formal proofs. They are meant as an aid to students who are a bit unsure about what exactly constitutes a proper logical argument, especially when quantifiers and nested implications are involved. They are not meant as a comprehensive treatment of first-order logic.

1.1 Syntax and semantics

1.1.1 Sorts

We will generally use a many-sorted formulation of first-order logic, where individual variables intrinsically range only over particular *sorts* (integers, expressions, derivation trees, etc.), rather than over all objects in the universe. A sort is essentially the same as a set, except that we usually work with a fixed, countable (and often finite) collection of sorts in any particular domain. In particular, not every mathematically well-defined set will automatically qualify as a sort. Rather, it is usually more appropriate to think of sorts in logic as corresponding to *types* in a programming language.

The following are examples of mathematical sets that we will often want to use as sorts in relevant application domains:

- The sets of natural numbers with zero ($\mathbf{N}$), integers ($\mathbf{Z}$), rational numbers ($\mathbf{Q}$), and real numbers ($\mathbf{R}$).
- The set of lexically well-formed *identifiers* (**Ide**) of a given programming language.

- The sets of syntactically well-formed *phrases* of the programming language, such as the set of well-formed arithmetic expressions (**AExp**), well-formed commands (**Com**), etc.

On the other hand, we usually wouldn't consider, for example, the set of all *even* numbers to be a sort; rather, being even is a *property* (predicate) that may or may not hold for elements of sort **N** or **Z**, but doesn't really make sense for **Q** and **R**, and definitely not for, say, **Com**.

It is common to let the name of a mathematical variable indicate its sort. For example, unless otherwise specified, i and j (and their *decorations* such as i_0 or j') typically range only over natural numbers, m and n may range over integers, and r and s may range over reals. When the sorts are syntactic categories specific to a particular language we usually explicitly state the conventions up front, for example saying that a ranges over **AExp**, and c over **Com**. In that case, we must be careful to stick to the conventions – i.e., not talk about, say, an expression of the form "$a + b$", but rather "$a_0 + a_1$", or "$a + a'$".

For technical convenience, we require all sorts to be *non-empty*.

1.1.2 Operations and terms

Every sort S contains a number of basic elements or *individuals*; for example, **N** contains $0, 1, 2, ...$). Additionally, we may construct elements of the sort by means of a countable (and often finite) collection of *operators* or *function symbols*, such as $+$, $\times$, log, etc. All operators have fixed source and target sorts; the *sorting*

$$f : S_1 \times \cdots \times S_n \to S$$

means that operator f accepts n arguments of sorts S_1,...,S_n, and returns an element of sort S. The number n is called the *arity* of the operator. Many binary operators (i.e., with arity 2) are conventionally written infix, e.g., $2+3$ instead of $+(2, 3)$. With multiple binary operators in play, we must specify their associativity and precedence conventions as usual, or insert extra parentheses for disambiguation.

For sorts representing program phrases, the syntactic constructors of the language grammar act as operators. For example, we may have an operator

$$\texttt{if} : \mathbf{BExp} \times \mathbf{Com} \times \mathbf{Com} \to \mathbf{Com}$$

for building a conditional command out of a Boolean test expression and two commands for the branches. It is helpful to think of such operators as *datatype constructors* in a functional language.

Note that function symbols are often *overloaded*. For example, when talking about the syntax and semantics of a programming language, we may use $+$ for both the mathematical addition function (of sort, e.g., $\mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$) and as a syntax constructor (of sort $\mathbf{AExp} \times \mathbf{AExp} \to \mathbf{AExp}$); it should always be clear from the context, which variant is meant in any particular situation. Sometimes, for extra emphasis, we may write the two uses in different fonts, e.g., $n_0 + n_1$ vs. $a_0 \texttt{+} a_1$.

Given the collection of function symbols, we may talk about the set of well-formed *terms* belonging to a given sort. Basically, a term is either a parameter variable (introduced below), or a function applied to some arguments:

$$t ::= x \mid f(t_1, \ldots, t_n)$$

The number of arguments and their sorts must, of course, match the sorting of the function. When $n = 0$, we write f instead of $f()$; in particular, individual elements of sorts (e.g., numbers) can be conveniently denoted by particular zero-ary operators, but there is no requirement that every element of the sort must correspond to a zero-ary operator, or even be expressible with a finite combination of operators. For example, there are uncountably many real numbers, but only countably many terms that could potentially denote such numbers.

In addition to the basic function symbols of the domain, we may introduce auxiliary, *defined* function symbols, in the format:

$$f(x_1, ..., x_n) \stackrel{\text{def}}{\equiv} t\,,$$

where the term t may (but need not) contain the variables $x_1, ..., x_n$. Such definitions are purely abbreviational; in particular, the defining term t *must not* use f recursively. (It may, however, refer to previously defined functions.) Auxiliary definitions are only used for conciseness or readability; they do not materially change the logical system. We use the symbol $\equiv$ instead of $=$ to emphasize the essentially syntactic, macro-like nature of such definitions.

1.1.3 Predicates and formulas

While function symbols f build up terms from other terms, *relation symbols* (or *predicates*) p build up basic (or *atomic*) *logical formulas* from terms. Like function symbols, predicates require their arguments to be of correct sorts; we will occasionally write predicate sortings concisely in the form

$$p : S_1 \times \cdots \times S_n \to \Omega$$

to say that the predicate p takes n arguments of the specified sorts. (Here, the special symbol Ω can be thought of as the notion of truth in the logic; it is *not* a sort.) Like operators, binary relations are often written infix. For example, applications of the binary relation symbol $\leq : \mathbf{Z} \times \mathbf{Z} \to \Omega$ are written as $n_1 \leq n_2$, with the usual meaning, while a relation symbol $\hookrightarrow : \mathbf{AExp} \times \mathbf{Z} \to \Omega$ might be used in the formula $a \hookrightarrow n$ to say that the syntactic expression a evaluates to the number n. (Again, beware of overloading: we may also have a *function* symbol $\leq : \mathbf{AExp} \times \mathbf{AExp} \to \mathbf{BExp}$, constructing a boolean expression out of two arithmetic ones; for extra emphasis we may write that one as a_1 `<=` a_2.)

From atomic formulas, we build up *compound formulas*, according to the following grammar:

$$P, Q, R ::= p(t_1, \ldots, t_n) \mid P_1 \wedge P_2 \mid P_1 \vee P_2 \mid P_1 \Rightarrow P_2 \mid \top \mid \bot \mid \neg P \mid \forall x{:}\, S.\, P \mid \exists x{:}\, S.\, P$$

The symbols $\wedge$, $\vee$, $\Rightarrow$, and $\neg$ are called *logical connectives*. (In texts specifically on formal logic, $\Rightarrow$ is often written as $\supset$, but here we'll use the more familiar symbol from general mathematics.) In the *quantifiers* $\forall$ and $\exists$, we often omit the "$: S$" when S is clear from the context, or from the name of the variable x. We will use the common conventions for

connective and quantifier precedences, from highest to lowest:

$$\begin{array}{c} \neg \\ \wedge \\ \vee \\ \Rightarrow \\ \forall, \exists \end{array}$$

For example, $\neg P \vee Q \wedge R$ means $(\neg P) \vee (Q \wedge R)$. Occasionally we will write formulas with redundant parentheses for extra clarity. When used as a connective, $\Rightarrow$ is considered to be *right*-associative, so e.g., $P \Rightarrow Q \Rightarrow R$ means $P \Rightarrow (Q \Rightarrow R)$, not $(P \Rightarrow Q) \Rightarrow R$ (nor $(P \Rightarrow Q) \wedge (Q \Rightarrow R)$). Quantifiers having the lowest precedence also means that they range over as much of the following formula as syntactically possible, e.g.,

$$\forall x.\, p(x) \Rightarrow q$$

means $\forall x.\, (p(x) \Rightarrow q)$, not $(\forall x.\, p(x)) \Rightarrow q$. (Be aware that some other texts may use different syntactic conventions!)

Also, as "syntactic sugar", we introduce the abbreviation $P \Leftrightarrow Q$ for the formula $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$, and we sometimes compress sequences of consecutive quantifiers of the same kind, so that, e.g., $\forall x, y, z.\, P$ abbreviates $\forall x.\, \forall y.\, \forall z.\, P$, and analogously for existential quantification.

Like for operators, we may define additional predicates, again non-recursively, as abbreviations:

$$p(x_1, ..., x_n) \stackrel{\text{def}}{\equiv} P\,.$$

Note that a formula may, in general, contain *free* variables (i.e., ones not bound by an quantifier inside the formula), such as the formula $0 \leq x \wedge x < 10$. We sometimes write $P(x)$ to emphasize that the formula P may (but need not) contain occurrences of x. Whether such a formula holds or not, depends on the value assigned to the variable x by the context.

1.2 Proof structure

A formal *proposition* is a true statement of the form

> for all x_1 in S_1, ..., and x_n in S_n, if P_1, P_2, ..., and P_m all hold, then so does Q.

Here $x_1, ..., x_n$ are called the *parameters* of the proposition; $P_1, ..., P_m$ are its *assumptions* or *hypotheses*; and Q is the *conclusion* or *consequence*. The formulas P_i and Q may not contain any free variables beyond x_1, ..., x_n, and thus the proposition will always have a definite truth value.[1]

[1] In the full generality of formal logic, the truth of a proposition also depends on the *interpretations* of the sorts, function symbols and predicate symbols occurring in it, e.g., that the function symbol $+ : \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$ actually represents mathematical addition. A proposition that is true in *any* interpretation is called *valid*. In almost all contexts, however, there will be only one relevant interpretation (the "standard" one), so we will generally not be concerned about the distinction between truth and validity.

Assuming that the sorts of the parameters x_1, ..., x_n are clear from their names (in which case the corresponding "for all" part of the statement of the the proposition may be unambiguously elided), we write that a proposition is (semantically) true as

$$P_1, \ldots, P_m \models Q\,.$$

Note that the truth of the original proposition is equivalent to the truth of the related assumption- and parameter-less proposition,

$$\models \forall x_1\colon S_1.\ \ldots \forall x_n\colon S_n.\, P_1 \wedge \cdots \wedge P_m \Rightarrow Q$$

(i.e., the above formula must hold without any further assumptions), but for actually arguing its truth, it will prove convenient to have the parameters and assumptions "unpacked".

In principle, it is simple to verify whether a proposition is true: consider all possible combinations of appropriately sorted values for the parameters $x_1, ..., x_n$, and check that every such combination that makes all of the P_i hold, also makes Q hold. However, if one or more of the sorts S_i is infinite, it would take an infinite amount of work to actually perform these checks. Therefore, we need a way of establishing the truth of the proposition in a finitary way: we replace the semantic condition of *truth* with the stronger, finitely checkable notion of *provability*.

Again assuming that the sorts of the variables are clear from their names, we express the notion of provability as the *sequent* or *consequence judgment*,

$$\boxed{P_1, \ldots, P_m \vdash Q}$$

Here the formulas P_i (often collectively written as Γ) are sometimes called the *antecedents* of the sequent, while Q is called the *consequent*. The order of the formulas P_i doesn't matter for provability of the judgment, but we still think of Γ as an ordered list, to be able to uniquely identify assumptions by number.

A *proof* of a sequent is a structured argument designed to convince the reader that, no matter how the parameters are instantiated, if all the antecedents hold for that instantiation, then so does the consequent. A proposition supported by such a proof is called a *theorem* of the logic. A proof of $\Gamma \vdash Q$ is often referred to as "a proof of Q from assumptions Γ". If the proof rules are sensible (i.e., *sound*), we will only be able to prove true propositions, i.e., $\Gamma \vdash Q$ will guarantee $\Gamma \models Q$.

A proof of a sequent is conceptually a tree, showing how subarguments establishing auxiliary propositions are combined into a formal derivation of the original proposition. Unlike many trees in computer science, logic proof trees are actually written "right side up", i.e, with the leaves towards the top of the page, and the root towards the bottom. The nodes of the tree are the individual inference steps, and are often depicted as horizontal bars, with the subtrees written above the bar, and the proposition that was proved below the bar. A proof tree $\mathcal{P}$, showing $\Gamma \vdash Q$, could look like this:

$$\left.\dfrac{\begin{array}{ccc}\left.\begin{array}{c}\vdots\\ \Gamma_1 \vdash Q_1\end{array}\right\}\mathcal{P}_1 & \cdots & \left.\begin{array}{c}\vdots\\ \Gamma_n \vdash Q_n\end{array}\right\}\mathcal{P}_n\end{array}}{\Gamma \vdash Q}\right\}\mathcal{P}\,.$$

Here $\mathcal{P}_1$ represents a subproof of $\Gamma_1 \vdash Q_1$, and so on. We will usually write the proof tree above slightly more compactly as:

$$\mathcal{P} = \frac{\begin{array}{ccc}\mathcal{P}_1 & & \mathcal{P}_n \\ \Gamma_1 \vdash Q_1 & \cdots & \Gamma_n \vdash Q_n\end{array}}{\Gamma \vdash Q}\,.$$

(Note that, even in this notation, each subtree $\mathcal{P}_i$ is considered to *include* its root proposition $\Gamma_i \vdash Q_i$.)

A key modularity property of the proof rules is that their applicability depends solely on *what* has been shown in the subproofs, not on *how* it is shown. Thus, for the proof $\mathcal{P}$ above, we would still have a valid proof if we were to replace the subproof $\mathcal{P}_1$ with a different $\mathcal{P}_1'$ that established the same proposition $\Gamma_1 \vdash Q_1$. This means that we can specify every proof rule in the format:

$$\frac{\Gamma_1 \vdash Q_1 \quad \cdots \quad \Gamma_n \vdash Q_n}{\Gamma \vdash Q}\,.$$

The sequents above the line are called the *premises* (or *premisses*; singular: *premiss*) of the rule; they should not be confused with the antecedents or assumptions in Γ. Likewise, the *conclusion* of the rule is the sequent $\Gamma \vdash Q$; it should not be confused with the various consequents Q and $Q_1, ..., Q_n$, although these are sometimes called conclusions too. All of our proof rules will be expressed as very simple, and purely syntactical, relationships between their premises and conclusions.

Some proof rules involve no premises (i.e., $n = 0$); they are called *axioms*. In particular, the *assumption axiom* says that we can immediately conclude $\Gamma \vdash Q$, if Q is identical to one of the formulas in Γ.

In the standard proof rules for first-order logic, each antecedent list Γ_i in the premises is either exactly the same as the Γ in the conclusion, or extends Γ with one or more formulas that become available as additional assumptions in the corresponding subproof $\mathcal{P}_i$ only. Many presentations therefore omit the Γ's everywhere, and instead introduce special notation for when the set of available assumptions is locally extended (see Section 1.5.1). We keep the assumption lists explicit for now, to emphasize that reasoning always happens in a context.

A simple proof rule with premises is the rule of and-*introduction*:

$$\frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \wedge P_2}\,.$$

It says that, to establish that the formula $P_1 \wedge P_2$ follows from some assumptions Γ, it suffices to establish by separate subarguments that P_1 follows from Γ and that P_2 follows from Γ. Note that there might also be other ways of proving the sequent; for example $P_1 \wedge P_2$ might already be one of the antecedents in Γ, in which case we could simply use the assumption axiom to get that particular consequent.

Introduction rules focus on the consequent of the rule conclusion, showing how it follows from premises with simpler consequents. Conversely, *elimination* rules focus on how a consequent of one the premises can be exploited. Again, one of the simplest such rules is and-elimination:

$$\frac{\Gamma \vdash P_1 \wedge P_2}{\Gamma \vdash P_1}\,.$$

(There is a symmetric rule for obtaining P_2.) This one states that, if we can establish $P_1 \wedge P_2$ from assumptions Γ (often, but necessarily, by noting that $P_1 \wedge P_2$ is already an assumption in Γ), then we can also immediately conclude that P_1 follows from Γ.

A "proof narrative" is a linearized traversal of a proof tree, presenting it for a human reader. It is important to keep in mind that the exact same proof may be narrated in superficially quite different ways:

Forward presentation This style of presentation, corresponding to a *post-order* traversal of the proof tree, starts with the leaf axioms, incrementally deriving intermediate subconclusions from them, until the desired final conclusion is reached. A typical phrase in the narrative could be, "Having shown ... and ..., we can now conclude ...". It has the advantage that it is quite easy to check mechanically: essentially, every statement used in the proof must either be an initial assumption, or follow from those assumptions and previously established statements by a proper rule of inference.

The disadvantage of a purely forward presentation is that it makes it hard to follow the intuition behind the proof: the argument just consists of a series of seemingly arbitrary steps, which "fortunately" happen to end up showing exactly what was wanted. Quite often, it is easier to actually read such a proof starting with the end, but that defeats the whole purpose of writing the narrative in natural language.

Backward (or goal-directed) presentation Here, we do a *pre-order* traversal of the proof tree, starting with the desired conclusion and keeping track of what still needs to be proved. That is, we often use a phrase like, "In order to show ..., it clearly suffices to show ... and ...".

A purely backwards presentation has the disadvantage that sometimes we have to invoke seemingly arbitrary subgoals; for example, in order to prove Q, we may decide to prove – for some P – both $P \Rightarrow Q$ and P, which again looks like an unmotivated step, even if it'll turn out to work. In almost all such cases, though, the truth of either $P \Rightarrow Q$ or P will either be self-evident, or previously established.

The most common mathematical style of presentation switches back and forth between these two extremes. The top-level development of a non-trivial result is often structured in forward style as a series of named or numbered lemmas, leading up to the main theorem. Each lemma or theorem proof, on the other hand, frequently uses a backward presentation of the big lines, but sometimes locally shifting to a run of forwards reasoning, where we work from the assumptions we have accumulated to the current subgoal. In such multi-directional presentations, it is absolutely crucial to always be very clear and explicit about what *has* been proved (or assumed), and what still *needs* to be proved. Merely writing down the relevant formulas, without clearly stating what follows from what, at best makes the proof needlessly hard to read, but more commonly simply obscures bugs or holes in the argument.

1.3 Proof rules

We now give a systematic presentation of all the standard proof rules of first-order logic.

1.3.1 Assumption

As already mentioned, the assumption rule reads, more formally:

$$\text{Ass}_i : \frac{}{P_1, \ldots, P_n \vdash P_i} \, (i \in \{1, ..., n\}) \, .$$

Here, Ass_i is just the name of the rule for future reference; in complicated proofs it is sometimes useful to be able to identify exactly which rule was used in a particular reasoning step. The *side condition* $(i \in \{1, ..., n\})$ is an additional formal requirement for the rule to be applicable in the given situation; here it says explicitly that the consequent formula P_i must be one of the antecedent formulas P_1 through P_n. In a proof narrative, a use of the assumption rule is typically indicated with the phrase like "... which we had previously assumed", if it is mentioned at all. Sometimes, for extra clarity, the assumptions may even be given explicit names or numbers.

It should be clear that adding extra assumptions to Γ cannot invalidate a use of the assumption rule. In fact, because all the rules uniformly pass on the assumptions in the conclusion to the premises (possibly with some extra assumptions added), the system as a whole enjoys the property of *weakening*: if $\Gamma \vdash Q$, then also $\Gamma' \vdash Q$ for any extension Γ' of Γ.

1.3.2 Domain-specific axioms

For any particular domain of application, there will usually be a collection of proof rules stating connections between the various function and predicate symbols that may occur in formulas. These rules are typically formulated as axioms, i.e., with no further premises:

$$\text{Axiom}_P : \frac{}{\Gamma \vdash P} \, (P \text{ is one of the domain axioms}) \, .$$

For example, a domain axiom may say that the relation $\leq$ between numbers is reflexive $(\forall x\colon \mathbf{Z}.\, x \leq x)$, or that addition is commutative $(\forall x\colon \mathbf{Z}.\, \forall y\colon \mathbf{Z}.\, x + y = y + x)$. However, as far as the logic is concerned, there is no essential difference between adding such facts as axioms, vs. as additional assumptions to the top-level sequent we are trying to prove.

In fact, it is possible to eliminate all uses of domain axioms from a proof, and instead list them all explicitly in the statement of the proposition. This makes the resulting theorem universally applicable (or valid); e.g., rather than proving something about integer addition specifically, we may be able to state and prove a more general result that would hold for *any* commutative function on the integers. Note, however, that we cannot explicitly quantify over sorts, operators, or predicates in the statement or proof of a formal theorem; that requires *higher-order logic*.

1.3.3 Internal lemma

Another frequently used proof rule is:

$$\text{Lemma} : \frac{\Gamma \vdash P \qquad \Gamma, P \vdash Q}{\Gamma \vdash Q} \, .$$

The rule says that, in order to prove Q from some assumptions, it is allowable to first establish another formula P, and then use P as an additional assumption in the proof

of Q. This is useful either when P is used multiple times in the proof of Q, or simply for giving a clearer presentation of the proof. As a special case, when Γ is empty, this proof rule corresponds to introducing and proving a named lemma in the development, and then using it later as a additional fact, much like a domain axiom.

Often, the proof of $\Gamma, P \vdash Q$ is significantly simpler than the proof of $\Gamma \vdash P$. When the truth of the former sequent is obvious, its explicit justification is often left out, and the proof narrative proceeds (in a forward style) as, "we have shown P, and therefore also Q". It is common, but incorrect, to state this reasoning step as "$P \Rightarrow Q$". To avoid confusion, use "hence", "so", "thus", or the symbol "$\therefore$". E.g., "... so $x + 2 = 5$, hence $x = 3$, and thus $x \geq 0$, as required."

1.3.4 Conjunction

As already mentioned, the rule of and-introduction reads,

$$\wedge\mathrm{I} : \frac{\Gamma \vdash P_1 \qquad \Gamma \vdash P_2}{\Gamma \vdash P_1 \wedge P_2} .$$

That is, to prove a conjunction, we must prove both conjuncts separately.

Conversely, there are two and-elimination rules, for extracting the left or the right subformula of a conjunction:

$$\wedge\mathrm{E}_1 : \frac{\Gamma \vdash P_1 \wedge P_2}{\Gamma \vdash P_1} \quad \text{and} \quad \wedge\mathrm{E}_2 : \frac{\Gamma \vdash P_1 \wedge P_2}{\Gamma \vdash P_2} .$$

Normally, the premise proof of $P_1 \wedge P_2$ is direct by assumption, or uses another elimination rule. For example, to show Q from $P \wedge (Q \wedge R)$, we first use the $\wedge\mathrm{E}_2$ rule to conclude $Q \wedge R$, and then $\wedge\mathrm{E}_1$ to get the desired Q. While it would not be formally incorrect to show the premise $\Gamma \vdash P_1 \wedge P_2$ of the $\wedge\mathrm{E}_1$ rule by a use of $\wedge\mathrm{I}$, doing so would be silly, since one of the premises of the and-introduction would already be precisely $\Gamma \vdash P_1$, which is what we ultimately wanted to show.

All of the conjunction rules are so simple and natural that one often doesn't even mention them in a proof narrative. That is, if often goes without saying that once we have established both P_1 and P_2, we also have $P_1 \wedge P_2$, and vice versa.

1.3.5 Disjunction

There are two introduction rules for disjunction:

$$\vee\mathrm{I}_1 : \frac{\Gamma \vdash P_1}{\Gamma \vdash P_1 \vee P_2} \quad \text{and} \quad \vee\mathrm{I}_2 : \frac{\Gamma \vdash P_2}{\Gamma \vdash P_1 \vee P_2} .$$

The rules say that, in order to prove $P_1 \vee P_2$ from some assumptions, it suffices to prove P_1 (or P_2) from the same assumptions. In a proof narrative, especially in backwards style, it is important to make clear which of the rules was used. For example, if we choose to prove $P_1 \vee P_2$ by proving P_2, it must be clearly said that we pursue the right alternative, and hence ignore the left one.

The or-elimination rule is a bit more complicated:

$$\vee\mathrm{E} : \frac{\Gamma \vdash P_1 \vee P_2 \qquad \Gamma, P_1 \vdash Q \qquad \Gamma, P_2 \vdash Q}{\Gamma \vdash Q} .$$

It is known as *proof by cases*: if we can show that $P_1 \vee P_2$ holds, then it remains to show that Q must hold if we assume P_1, and that it must also hold if we assume P_2. (And then it will of course also hold if both P_1 and P_2 happen to be true.) Again, often $P_1 \vee P_2$ will be simply an assumption or otherwise immediate, so the narrative will have the shape: "We know that $P_1 \vee P_2$. Suppose P_1 holds; then we get Q as follows: .... On the other hand, if P_2 holds, then Q follows because ...".

1.3.6 Implication

The introduction rule for implication is:

$$\Rightarrow\text{I} : \frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q} .$$

It says that, to show an implication, we take the formula on the left of the arrow as an additional assumption, and use that to show the formula on the right. For example. in a backwards proof narrative, "We want to show $P \Rightarrow Q$; so we assume P, and we must show Q." It important that P is only available as an extra assumption for the duration of showing Q, not elsewhere in the proof. This is easy to keep track of in a formal, tree-structured derivation, but it is a common mistake in textual proofs to accidentally use assumptions outside of their scope.

The implication-elimination rule is also straightforward:

$$\Rightarrow\text{E} : \frac{\Gamma \vdash P \Rightarrow Q \qquad \Gamma \vdash P}{\Gamma \vdash Q} .$$

That is, if we can show both $P \Rightarrow Q$ and P from some assumptions, we can conclude Q from those assumptions. Note that, if we further choose to prove $P \Rightarrow Q$ by implies-introduction, the net effect is just like an instance of the Lemma rule from before, i.e., to show $\Gamma \vdash Q$, it suffices to show $\Gamma \vdash P$ and $\Gamma, P \vdash Q$.

1.3.7 Truth and falsehood

The trivially true formula is also trivially provable from any assumptions:

$$\top\text{I} : \frac{}{\Gamma \vdash \top} .$$

On the other hand, there is no dedicated true-elimination rule, i.e., a proof rule with a premise $\Gamma \vdash \top$: because $\top$ is so easy to show, we cannot non-trivially use it to show anything else.

For falsehood, there is no introduction rule, precisely because there should be no way to show the false proposition. (However, this does *not* mean that $\Gamma \vdash \bot$ is never derivable; for example, it might follow if Γ actually contained $\bot$, or a collection of other formulas from which $\bot$ was derivable, such as both p and $p \Rightarrow \bot$.) On the other hand, there is a false-elimination rule,

$$\bot\text{E} : \frac{\Gamma \vdash \bot}{\Gamma \vdash Q} ,$$

which says that, from falsehood, anything follows – or, more precisely, that whenever all of the assumptions Γ are mutually contradictory, then we can correctly conclude anything.

Typically, this happens when we have extended an originally consistent set of assumptions by another formula (in an implication-introduction or an or-elimination) that makes the whole set contradictory. We do *not* refer to $\bot$E as "proof by contradiction"; that is something else, covered below (in Section 1.3.12).

Note that the truth and falsehood rules can be seen as the zero-ary versions of the conjunction and disjunction rules, respectively: We had an and-introduction rule with two premises, and two and-elimination rules; in the zero-ary analog, we have a true-introduction rule with zero premises, and zero elimination rules. Conversely, for disjunction, we had two introduction rules, and an elimination rule with two additional premises (besides the one showing the disjunction). And for falsehood, we now have zero introduction rules, and an elimination rule with zero additional premises.

1.3.8 Negation

In the context of natural deduction proofs, negation is best seen as a special case of implication, where $\neg P$ is effectively an abbreviation for $P \Rightarrow \bot$. Thus, the negation-introduction rule reads:

$$\neg\mathrm{I} : \frac{\Gamma, P \vdash \bot}{\Gamma \vdash \neg P}\,.$$

That is, to show $\neg P$ from Γ, we must show that P cannot hold if all of Γ already hold, i.e., that Γ and P together would lead to a logical contradiction. This is not "proof by contradiction", either; it's just proving a negative.

The elimination rule for negation is:

$$\neg\mathrm{E} : \frac{\Gamma \vdash \neg P \qquad \Gamma \vdash P}{\Gamma \vdash \bot}\,.$$

It simply says that, if our assumptions let us show that both P and $\neg P$ hold for some specific P, they must be contradictory.

1.3.9 Universal quantification

The rule for introducing a universal quantifier is:

$$\forall\mathrm{I} : \frac{\Gamma \vdash P[x'/x]}{\Gamma \vdash \forall x{:}\, S.\, P}\,(x' \text{ does not occur in } \Gamma \text{ or } P)\,.$$

It says that, in order to show that a formula P holds for all elements of S, we introduce a new parameter x' ranging over S, and show that P holds if we substitute all occurrences of x (except those governed by an *inner* quantification over x, see below) with that new parameter x'. The requirement that x' is "new", or *fresh*, means that we must not already have assumed anything about it elsewhere in the proof. Often, but not always, we can just use x directly as the x'; the renaming is just needed in the case of name clashes.

Such clashes actually arise fairly commonly when working with abbreviations. For example, suppose we have defined the predicate $odd : \mathbf{Z} \to \Omega$ by $odd(x) \equiv \forall y{:}\, \mathbf{Z}.\, y{+}y \neq x$. Clearly we wouldn't want to conclude $y > 2 \vdash odd(4)$, but if we neglect to properly introduce a fresh parameter for the inner y in the introduction rule, we would incorrectly reduce the goal to proving $y > 2 \vdash y + y \neq 4$, which actually is true – rather than $y > 2 \vdash y' + y' \neq 4$, which evidently is not.

In a proof narrative, forall-introduction is commonly expressed as, "We must show $\forall x\colon S.\, p(x)$. Let x' be an arbitrary element of S; then we must show $p(x')$." Forall-introduction is often combined with implies-introduction, e.g., "To show $\forall x.\, P(x) \Rightarrow Q(x)$, let x be arbitrary such that $P(x)$ holds; we must show $Q(x)$.".

The forall-elimination rule is also relatively simple:

$$\forall \mathrm{E}_t : \frac{\Gamma \vdash \forall x\colon S.\, P}{\Gamma \vdash P[t/x]} \,(t \text{ a term of sort } S)\,.$$

where the *instantiation* term t may contain previously introduced parameters, in addition to function symbols. It captures the intuition that if we have established that $P(x)$ holds for all x of sort S, it must also hold for any particular t of that sort.

When performing the substitution of t for x in P, we must take care to leave alone any occurrences of x that are governed by quantifiers inside P; for example if P is $\exists x.\, x > 5$, then $P[3/x]$ is just P itself – not $\exists 3.\, 3 > 5$ (which is syntactically meaningless), nor $\exists x.\, 3 > 5$ (which is well-formed, but means something else). Conversely, and more subtly, we must also avoid *capturing* variables in t by quantifiers in P. We achieve this by *renaming* the bound variables of such quantifiers as necessary. For example, suppose the premise of the $\forall$E rule is obtained by the domain-specific axiom $\vdash \forall x.\, \exists y.\, x < y$ (where all variables range over integers), so P itself is the formula $\exists y.\, x < y$. If we then take the term t to be $y + 1$, the result of $P[t/x]$ is *not* $\exists y.\, y + 1 < y$ (which is not true), but rather $\exists y'.\, y + 1 < y'$ (which is).

In a proof narrative, we can say, e.g., "We have $\forall x.\, p(x)$; taking x as t, we therefore get $p(t)$.". It is very common for the $\forall x.\, P$ to be either a domain-specific axiom, or a previously established general lemma.

1.3.10 Existential quantification

Much like the two $\vee$I rules essentially inverted the $\wedge$E ones, the introduction rule for existentials mirrors forall-elimination:

$$\exists \mathrm{I}_t : \frac{\Gamma \vdash P[t/x]}{\Gamma \vdash \exists x\colon S.\, P} \,(t \text{ a term of sort } S)\,.$$

Like in $\forall$E, the substitution $P[t/x]$ must be done in a capture-avoiding way. The term t, which again may contain previously introduced parameters, is called the *witness* for the existential formula holding. In a proof narration, exists-introduction could be written as, "To show $\exists x.p(x, y)$, we take x to be t, and show $p(t, y)$." It is very important to state unambiguously exactly what t is, before proceeding.

The exists-elimination rule is perhaps the most complicated of all:

$$\exists \mathrm{E} : \frac{\Gamma \vdash \exists x\colon S.\, P \qquad \Gamma, P[x'/x] \vdash Q}{\Gamma \vdash Q} \,(x' \text{ does not occur in } \Gamma,\, P \text{ or } Q)\,.$$

It says that, if we know that $\exists x.P$ holds, and we can show for some otherwise unconstrained x', that whenever $P[x'/x]$ holds then Q holds, then Q must in fact hold. The proof narrative reads like "we have $\exists x.P(x)$, and we must show Q. Let therefore x' be an arbitrary element of S such that $P(x')$ holds, and we show Q under that assumption." Again, it is crucial that x' is "fresh": we don't know anything about it other than that it makes P hold.

1.3.11 Equality

It is common to treat the *equality predicate* $=_S : S \times S \to \Omega$ as an additional formula constructor with dedicated introduction and elimination rules, rather than as just another relation symbol equipped with a collection of domain-specific axioms (symmetry, transitivity, etc.). We also normally omit the subscript S, when it is clear from the context. The equals-introduction rule is seemingly trivial; it just says that $=$ is reflexive:

$$=\text{I} : \frac{}{\Gamma \vdash t = t} \,.$$

(In some more refined variants of first-order logic, where operators may denote *partial* functions, there is an additional requirement that the term t must not only be syntactically well-formed and well-sorted, but also semantically well-defined; for instance we would *not* be able to conclude by reflexivity that $1/0 =_{\mathbf{R}} 1/0$.)

The elimination rule for equality says that if $t_1 = t_2$ holds, we can replace any use of t_1 in a formula P with t_2, without affecting its truth value. Formally, we have:

$$=\text{E} : \frac{\Gamma \vdash t_1 = t_2 \qquad \Gamma \vdash P[t_1/x]}{\Gamma \vdash P[t_2/x]} \,.$$

This is a surprisingly powerful reasoning principle. For example, we can use it to give a formal proof that $=$ is a symmetric relation, i.e., that if $t_1 = t_2$, then also $t_2 = t_1$:

$$\dfrac{\dfrac{\text{Ass}_1\,\dfrac{}{t_1 = t_2 \vdash t_1 = t_2} \qquad \begin{array}{c} =\text{I}\,\dfrac{}{t_1 = t_2 \vdash t_1 = t_1} \\ \vert\vert\vert \\ t_1 = t_2 \vdash (x = t_1)[t_1/x] \end{array}}{=\text{E}\,\dfrac{}{t_1 = t_2 \vdash (x = t_1)[t_2/x]}}}{\begin{array}{c} \vert\vert\vert \\ t_1 = t_2 \vdash t_2 = t_1 \end{array}} \,.$$

Here, we have taken the formula P in $=$E to be $(x = t_1)$, where x is some variable not used anywhere else in the proof. (The "steps" indicated by $\vert\vert\vert$ are not proper reasoning steps, but simply represent writing out the term substitutions; that is, the sequents above and below each $\vert\vert\vert$ are actually *the same*.) Similar arguments show that $=$ is transitive (if $t_1 = t_2$ and $t_2 = t_3$, then also $t_1 = t_3$) and a congruence (if $t_1 = t_2$, then also $f(t_1) = f(t_2)$ for any operator f). In proof narratives, we almost always use these familiar properties of $=$ without explicit mention.

1.3.12 Proof by contradiction

The rules covered so far make up a so-called *intuitionistic logic*; they always produce what are called *constructive* proofs. In some branches of mathematics (but very rarely in proofs related to programming language theory), one also employs the following rule of *indirect proof*, or *proof by contradiction*, giving *classical* logic:

$$\text{PBC} : \frac{\Gamma, \neg P \vdash \bot}{\Gamma \vdash P} \,.$$

It says that, if from the additional assumption $\neg P$, we can derive a contradiction, then P must hold. The proof narrative goes, "To show P, we assume by contradiction that P does not hold, i.e., that $\neg P$ holds, and show that that would lead to a contradiction."

Note that this rule can be seen as a combination of two other rules that are sometimes incorrectly referred to as proof by contradiction, namely the $\bot$E and $\neg$I rules from above. Compared to the former, PBC gives us the additional hypothesis $\neg P$ to derive $\bot$, so anything provable using false-elimination is also immediately provable by PBC, but not conversely. On the other hand, compared to not-introduction, here we are establishing P by showing that $\neg P$ would lead to a contradiction, rather than that getting $\neg P$ by showing that P would cause a contradiction. These two principles are not quite saying the same thing, because – in the absence of PBC – the formulas P and $\neg\neg P$ are *not* logically equivalent. In fact, an alternative way of extending intuitionistic logic to classical is by adding the following rule of *double-negation elimination*:

$$\text{DNE} : \frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P} .$$

(On the other hand, we *can* conclude $\neg\neg P$ from P using just the previously introduced intuitionistic rules.)

A third way of getting the power of classical logic is by the *law of excluded middle*,

$$\text{LEM} : \frac{}{\Gamma \vdash P \vee \neg P} .$$

That is, for any formula P, we can immediately conclude that either P or its negation must hold.

Given any of the rules (PBC, DNE, or LEM), we can derive the other two. The reason why one does not always consider them an inseparable part of the logical system is that they may break the useful property of *constructivity*.

In particular, from a correct constructive proof of $\vdash \exists x.p(x)$ one can always extract the relevant witness, i.e., a term t such that $p(t)$ holds. But with the non-constructive principles in this section, this is not always the case: we can construct rigorous proofs that an element with some interesting property must exist, without the proof providing any help in determining what that element might be.

1.3.13 Induction

In addition to domain-specific axioms, the logic may also contain domain-specific inference rules. The prime example of such rules are *induction principles*, which provide an alternative way of showing a universally quantified formula. For instance, the principle of *mathematical induction*, or *induction over the natural numbers*, can be formalized as the following rule:

$$\text{IND}_{\mathbf{N}} : \frac{\Gamma \vdash P[0/x] \qquad \Gamma, P[x'/x] \vdash P[(x'+1)/x]}{\Gamma \vdash \forall x\colon \mathbf{N}.\, P} \; (x' \text{ does not occur in } \Gamma \text{ or } P) .$$

That is, to show that some formula P holds for all natural numbers x, it suffices to show that it holds for 0, and that, from the additional assumption (the so-called *induction hypothesis*) that it holds for some fresh parameter x', we can show that it also holds for $x' + 1$.

In this course, we will be heavily using induction principles – not only over natural numbers, but also over syntactic structures, and even over entire formal derivation trees.

1.4 Some larger derivations

Let us consider a few examples of complete proofs, both in formal and semi-formal form.

Example If $(P \vee Q) \wedge R$, then $(P \wedge R) \vee (Q \wedge R)$. Proof tree:

$$\dfrac{\dfrac{\dfrac{\overset{\text{Ass}_1}{L \vdash (P \vee Q) \wedge R}}{L \vdash P \vee Q}\,\wedge\text{E}_1 \qquad \dfrac{\dfrac{\overset{\text{Ass}_2}{L, P \vdash P} \qquad \dfrac{\overset{\text{Ass}_1}{L, P \vdash (P \vee Q) \wedge R}}{L, P \vdash R}\,\wedge\text{E}_2}{L, P \vdash P \wedge R}\,\wedge\text{I}}{L, P \vdash (P \wedge R) \vee (Q \wedge R)}\,\vee\text{I}_1 \qquad \vdots \qquad L, Q \vdash (P \wedge R) \vee (Q \wedge R)}{\underbrace{(P \vee Q) \wedge R}_{L} \vdash (P \wedge R) \vee (Q \wedge R)}\,\vee\text{E}$$

(The missing derivation symbolized by $\vdots$ has essentially the same structure as the middle one, only using the $\vee\text{I}_2$-rule.)

Sample proof narrative (in forward style, with the implicit rule uses and assumption numbers noted in brackets): "First, from the initial assumption [1] that $(P \vee Q) \wedge R$ holds, we in particular have [by $\wedge\text{E}_1$] that $P \vee Q$ holds. We proceed with a proof by cases. First, suppose that P holds [2]. Since from our assumption [1], [by $\wedge\text{E}_2$] we also have R, [by $\wedge I$] we get $P \wedge R$, and therefore [by $\vee\text{I}_1$] also the desired conclusion $(P \wedge R) \vee (Q \wedge R)$. Similarly, suppose Q holds; then by a symmetric argument, we obtain first $Q \wedge R$, and then again $(P \wedge R) \vee (Q \wedge R)$. Thus, since we have already established that at least one of P and Q holds, [by $\vee E$] we have $(P \wedge R) \vee (Q \wedge R)$, which was what we needed to show."

Example If $\neg\exists x\colon S.\, P(x)$, then $\forall y\colon S.\, \neg P(y)$. Proof tree:

$$\dfrac{\dfrac{\dfrac{\overset{\text{Ass}_1}{\neg\exists x\colon S.P(x), P(y') \vdash \neg\exists x\colon S.\, P(x)} \qquad \dfrac{\overset{\text{Ass}_2}{\neg\exists x\colon S.P(x), P(y') \vdash P(y')}}{\neg\exists x\colon S.P(x), P(y') \vdash \exists x\colon S.\, P(x)}\,\exists\text{I}_{y'}}{\neg\exists x\colon S.P(x), P(y') \vdash \bot}\,\neg\text{E}}{\neg\exists x\colon S.P(x) \vdash \neg P(y')}\,\neg\text{I}}{\neg\exists x\colon S.P(x) \vdash \forall y\colon S.\, \neg P(y)}\,\forall\text{I}$$

Sample proof narrative (in backward style): "We assume $\neg\exists x.P(x)$ [1], and must prove $\forall y.\neg P(y)$. So, [for $\forall I$] let y' be completely arbitrary; we must show $\neg P(y')$. To do that, let us assume [for $\neg I$] that $P(y')$ holds [2], and use this to derive a contradiction. And to get that contradiction, since we already have assumed $\neg\exists x.P(x)$ [1], [by $\neg E$], we just need to also show $\exists x.P(x)$. Pick the witness x as y'; then [for $\exists I$] we must show $P(y')$ – but that is also already one of our assumptions [2], so we are done."

1.5 Alternative presentations

The sequent-style presentation of natural deduction in the previous section strikes a balance between generality and compactness. While we explicitly list all the hypotheses of a sequent in every proof step, the treatment of parameters is largely implicit. In this section, we will consider how both of those choices can be made differently.

1.5.1 Implicit hypotheses

It is easy to see that all the proof rules rules have the property that assumptions are only added to Γ, when moving upwards in the tree. Likewise, assumptions in Γ may be used any number of times, and in any order, in the derivation. Because of these structural properies of the system, it is possible to highlight only the places in the derivation at which Γ is actually changed.

In this alternative presentation, the main judgment form is just $\boxed{\vdash P}$, stating that P is provable. (Many styles also omit the $\vdash$, writing just the formula P as the conclusion of the derivation.) Any initial assumptions may now appear as leaves in the tree, *without* an overbar. However, any assumptions added locally in the derivation, such as in the $\Rightarrow$I rule, may be *discharged*, meaning that they are not included in the set of hypotheses on which the conclusion of the derivation rests.

Most of the rules (e.g., $\Rightarrow$E) in the new system look just as before, only without the Γ on the left of $\vdash$. However, the rules that introduce additional assumptions use a new convention:

$$\Rightarrow\mathrm{I}_u : \frac{\begin{array}{c}{}^{u}\overline{\vdash P}\\ \vdots\\ \vdash Q\end{array}}{\vdash P \Rightarrow Q} \qquad \vee\mathrm{E}_u : \frac{\vdash P_1 \vee P_2 \qquad \begin{array}{c}{}^{u_1}\overline{\vdash P_1}\\ \vdots\\ \vdash Q\end{array} \qquad \begin{array}{c}{}^{u_2}\overline{\vdash P_2}\\ \vdots\\ \vdash Q\end{array}}{\vdash Q} \qquad \mathrm{PBC}_u : \frac{\begin{array}{c}{}^{u}\overline{\vdash \neg P}\\ \vdots\\ \vdash \bot\end{array}}{\vdash P}$$

Here, the $\vdots$ notation indicates that the given subderivation may use the formula on top as if it were an extra axiom. (Some presentations styles instead write $[\vdash P]$ for this.) The u is some unique id, simplifying verification of the proof by identifying the exact rule instance that introduced the extra assumption.

For an example of a derivation in this style, here is the the distributivity result from the previous section (now fitting all three subderivations in the $\vee$E rule):

$$\vee\mathrm{E}_u \frac{\wedge\mathrm{E}_1 \dfrac{\vdash (P \vee Q) \wedge R}{\vdash P \vee Q} \qquad \vee\mathrm{I}_1 \dfrac{\wedge\mathrm{I} \dfrac{{}^{u_1}\overline{\vdash P} \qquad \wedge\mathrm{E}_2 \dfrac{\vdash (P \vee Q) \wedge R}{\vdash R}}{\vdash P \wedge R}}{\vdash (P \wedge R) \vee (Q \wedge R)} \qquad \vee\mathrm{I}_2 \dfrac{\wedge\mathrm{I} \dfrac{{}^{u_2}\overline{\vdash Q} \qquad \wedge\mathrm{E}_2 \dfrac{\vdash (P \vee Q) \wedge R}{\vdash R}}{\vdash Q \wedge R}}{\vdash (P \wedge R) \vee (Q \wedge R)}}{\vdash (P \wedge R) \vee (Q \wedge R)}$$

It is a straightforward exercise to convert well-formed derivations in one of the styles to the other. While the implicit style is more compact, it doesn't scale as nicely to defining more complicated logics (such as linear, or modal), which impose additional restrictions on where assumptions may be used in a derivation. Also, it somewhat complicates the notion of *induction on derivations*, which will be a central tool for reasoning about formal systems in the following chapters

1.5.2 Explicit parameters

In the main presentation, the *parameters* of the derivation, both global and locally introduced, were kept essentially implicit. In an even more verbose presentation style, the

set of available parameters is also made explicit as a list of parameter variables and their sorts:

$$\Delta ::= x_1 \colon S_1, ..., x_n \colon S_n$$

where all the x_i are distinct. Recognizing that a Δ is effectively a finite partial function from variables to sorts, we write $\operatorname{dom} \Delta = \{x_1, ..., x_n\}$. We can now also define an explicit sorting judgment for terms, $\boxed{\Delta \triangleright t : S}$, which says that t is a well-formed term of sort S, possibly containing variables from Δ.

Then the judgment form for provability becomes $\boxed{\Gamma \vdash_\Delta Q}$, where all variables occurring inside terms in Γ and Q belong to $\operatorname{dom} \Delta$ (unless bound by inner quantifications in Γ or Q). Most proof rules are not modified, except that $\vdash$ is replaced by $\vdash_\Delta$ in both premises and conclusion. However, the introduction and elimination rules for universal quantification now read:

$$\forall \mathrm{I} : \frac{\Gamma \vdash_{\Delta, x' : S} P[x'/x]}{\Gamma \vdash_\Delta \forall x \colon S.\, P} \; (x' \notin \operatorname{dom} \Delta) \qquad \forall \mathrm{E}_t : \frac{\Gamma \vdash_\Delta \forall x \colon S.\, P \qquad \Delta \triangleright t : S}{\Gamma \vdash_\Delta P[t/x]}$$

And for existential quantification:

$$\exists \mathrm{I}_t : \frac{\Gamma \vdash_\Delta P[t/x] \qquad \Delta \triangleright t : S}{\Gamma \vdash_\Delta \exists x \colon S.\, P} \qquad \exists \mathrm{E} : \frac{\Gamma \vdash_\Delta \exists x \colon S.\, P \qquad \Gamma, P[x'/x] \vdash_{\Delta, x' : S} Q}{\Gamma \vdash_\Delta Q} \; (x' \notin \operatorname{dom} \Delta)$$

We also have a weakening principle for parameters: If $\Delta \subseteq \Delta'$ (i.e., if Δ' contains all the parameter assumptions in Δ, and possibly some additional ones), then whenever $\Gamma \vdash_\Delta Q$, we also have $\Gamma \vdash_{\Delta'} Q$.

For example, our second sample derivation now reads:

$$\dfrac{\dfrac{\dfrac{}{\neg\exists x \colon S.\, P(x), P(y') \vdash_{y' : S} \neg\exists x \colon S. P(x)}\,\text{Ass}_1 \qquad \dfrac{\dfrac{\dfrac{}{\neg\exists x \colon S.\, P(x), P(y') \vdash_{y' : S} P(y')}\,\text{Ass}_2 \qquad \dfrac{}{y' \colon S \triangleright y' : S}}{\neg\exists x \colon S.\, P(x), P(y') \vdash_{y' : S} \exists x \colon S. P(x)}\,\exists \mathrm{I}_{y'}}{\neg\exists x \colon S.\, P(x), P(y') \vdash_{y' : S} \bot}\,\neg\mathrm{E}}{\neg\exists x \colon S.\, P(x) \vdash_{y' : S} \neg P(y')}\,\neg\mathrm{I}}{\neg\exists x \colon S.\, P(x) \vdash \forall y \colon S. \neg P(y)}\,\forall \mathrm{I}$$

This style of presentation scales better to logics with a more complicated sort structure, including in particular *dependent sorts*, where we can talk, e.g., of the sort of all integer lists of length exactly n, where n is itself a parameter. It also makes it feasible to lift the restriction that sorts must be non-empty.

1.6 Exercises

1. Argue in precise natural language, preferably without appealing to non-constructive reasoning principles (indirect proof or the law of excluded middle), that each of the following propositions is true, for any choice of concrete formulas P, Q, and R. Then, sketch the formal proof tree using the natural deduction rules.

   Also, for each one, is its converse (i.e., with assumption and conclusion swapped) provable, perhaps only non-constructively? Argue why or why not.

(a) If $P \wedge (Q \wedge R)$, then $Q \wedge P$.

(b) If $Q \vee P$, then $P \vee (Q \vee R)$.

(c) If $(P \wedge Q) \vee R$, then $(P \vee R) \wedge (Q \vee R)$.

(d) If $\neg P \vee Q$, then $P \Rightarrow Q$.

(e) If $(\exists x.\, P(x)) \Rightarrow Q$, then $\forall x.\, P(x) \Rightarrow Q$. ($x$ may occur in P, but not in Q.)

(f) If $\exists x\colon S.\, \forall y\colon T.\, P(x, y)$, then $\forall y\colon T.\, \exists x\colon S.\, P(x, y)$.

2. Let S be a sort, for example representing the (evidently non-empty) set of students taking *Semantics and Types* this year, with all variables ranging over elements of S. Let $p : S \to \Omega$ be a predicate, for example with the interpretation that $p(s)$ means that student s passes the course. Consider the following mystery formula:

$$M \;\equiv\; \exists x.\, p(x) \Rightarrow \forall y.\, p(y)$$

[Remember that the implicit parentheses go like $\exists x.(p(x) \Rightarrow \forall y.p(y))$.] In our sample interpretation, M says that there is a some very special student, with the property that if he or she passes the course, then all students pass it!

Argue as rigorously as you can – preferably, but not necessarily, using the proof rules – that M is provable without any additional assumptions or domain-specific axioms (i.e., knowledge of the actual set of SaT students and their performance). **Hint:** you will need to use one or more of the non-constructive principles.