lec5–7: Natural deduction

Jana Dunfield

February 1, 2022

1 Context

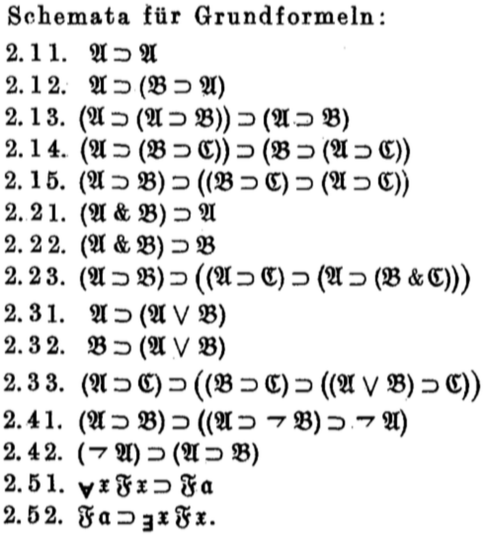
The horizontal-line notation for rules was first used in Gentzen's thesis to develop *natural deduction* and *sequent calculus*. These logical systems were developed before most of computer science existed, but they have deep connections to the static semantics of programming languages.

2 Background

In the early 1900s, the principal efforts towards foundations for mathematical and logical reasoning—*mathematical logic*—focused on developing sets of axioms. Axioms are similar to rules in having meta-variables that can be instantiated (to logical formulas, for example), but differ from rules in having no premises as such. Instead, the premises are encoded as the conditions of implications. For example, the axiom

$$A \supset (B \supset A)$$

says that if A, then: if B, then A. More clearly, it can be read "if A and B, then A." The first occurrence of A plays the role of a premise, as does the B; the second occurrence of A plays the role of the conclusion.



Schemata für Grundformeln:

2.11. $\mathfrak{A} \supset \mathfrak{A}$
2.12. $\mathfrak{A} \supset (\mathfrak{B} \supset \mathfrak{A})$
2.13. $(\mathfrak{A} \supset (\mathfrak{A} \supset \mathfrak{B})) \supset (\mathfrak{A} \supset \mathfrak{B})$
2.14. $(\mathfrak{A} \supset (\mathfrak{B} \supset \mathfrak{C})) \supset (\mathfrak{B} \supset (\mathfrak{A} \supset \mathfrak{C}))$
2.15. $(\mathfrak{A} \supset \mathfrak{B}) \supset ((\mathfrak{B} \supset \mathfrak{C}) \supset (\mathfrak{A} \supset \mathfrak{C}))$
2.21. $(\mathfrak{A} \mathbin{\&} \mathfrak{B}) \supset \mathfrak{A}$
2.22. $(\mathfrak{A} \mathbin{\&} \mathfrak{B}) \supset \mathfrak{B}$
2.23. $(\mathfrak{A} \supset \mathfrak{B}) \supset ((\mathfrak{A} \supset \mathfrak{C}) \supset (\mathfrak{A} \supset (\mathfrak{B} \mathbin{\&} \mathfrak{C})))$
2.31. $\mathfrak{A} \supset (\mathfrak{A} \vee \mathfrak{B})$
2.32. $\mathfrak{B} \supset (\mathfrak{A} \vee \mathfrak{B})$
2.33. $(\mathfrak{A} \supset \mathfrak{C}) \supset ((\mathfrak{B} \supset \mathfrak{C}) \supset ((\mathfrak{A} \vee \mathfrak{B}) \supset \mathfrak{C}))$
2.41. $(\mathfrak{A} \supset \mathfrak{B}) \supset ((\mathfrak{A} \supset \neg \mathfrak{B}) \supset \neg \mathfrak{A})$
2.42. $(\neg \mathfrak{A}) \supset (\mathfrak{A} \supset \mathfrak{B})$
2.51. $\forall \mathfrak{x}\, \mathfrak{F}\mathfrak{x} \supset \mathfrak{F}\mathfrak{a}$
2.52. $\mathfrak{F}\mathfrak{a} \supset \exists \mathfrak{x}\, \mathfrak{F}\mathfrak{x}$.

Such axioms (the above is an image from Gentzen's thesis, based on work by Hilbert and Glivenko) were not user-friendly: the process of instantiating axioms doesn't line up well with the reasoning that mathematicians actually do.

Gentzen's work has several advantages over axiom systems like the above.

First, rules clearly distinguish the premises from the conclusion. I think this is a relatively small advantage: with enough practice, the premises in axiom systems are easy to see.

Second, each of Gentzen's rules of natural deduction has a close analogue to actual mathematical reasoning, whereas axiom systems necessarily include "administrative" axioms like $A \supset (B \supset A)$; no mathematician would write "we know $x > 0$ and $x < y$, therefore, $x > 0$" in a proof. Natural deduction's closeness to mathematical reasoning arises from natural deduction's handling of assumptions; while natural deduction's particular style of modelling assumptions is somewhat awkward, Gentzen also developed *sequent calculus* which models assumptions in a different way.

Third, along with rules Gentzen developed *derivations*. Rather than searching through a proof of the conjunction $A \wedge B$ for the parts contributing to A and the parts contributing to B, a derivation proving $A \wedge B$ clearly separates the proof of A from the proof of B. This compositionality makes derivations easier to deal with as mathematical objects, and—from a computational standpoint—as data structures.

However, a disadvantage of derivations is that they are space-inefficient. (It was once fashionable for PL researchers to include large derivations in their papers; these usually required a figure in "landscape" orientation.) Moreover, they do not resemble traditional mathematical proofs. Some descendants of natural deduction, such as *Fitch-style natural deduction*, adopt many of Gentzen's rules (and his rule notation) but use a line-by-line proof format rather than derivations. Such descendants seem to be a good way to teach logic (which is why we use those systems in CISC 204), and a good way to write careful and detailed proofs, but for our purposes we need to handle proofs as objects (data structures) in their own right.

Gentzen, backwards

Gentzen died in a prison camp in what is now Czechia in 1945. He had been working at the German University in Nazi-occupied Prague, having inexplicably refused to flee to Germany.

Gentzen was a Nazi. This is not hyperbole; he was a member of the Nazi Party. Worse, he joined the SA, an important Nazi paramilitary organization. In his defence (to the extent there can be a defence), he did not attend SA meetings, and claimed (in a letter to Paul Bernays, his doctoral advisor) to have joined the SA solely to advance his academic career. Also in his favour, he corresponded repeatedly with Jewish mathematicians, and did not stop when told to do so.

In 1933, the Nazis fired all the Jewish university professors, including Bernays. Bernays moved to Switzerland and lived until 1977.

3 Introduction and elimination

A feature of natural deduction (inherited by type systems!) is that the rules can be systematically designed, or at least systematically organized. The rules for each *connective*, such as $\wedge$ or $\supset$, can be categorized as (1) rules that *introduce* the connective, and (2) rules that *eliminate* the connective. (The number of rules in each category depends on the connective; one very common connective has zero elimination rules.)

In addition, the rules of natural deduction are *orthogonal*: the rules to introduce and eliminate $\wedge$ only mention $\wedge$, not $\supset$ or any other connective. This makes natural deduction, and type systems based on it, easier to extend: to incorporate a connective, we only have to design its introduction and elimination rules—which may not be easy but can be done without regard for any other connectives. The rules for each connective are a kind of "module" in the rule system.

4 Natural deduction

We begin with a grammar of formulas. We leave the atomic formulas (P, Q) unspecified in the grammar, but we will use things like $a > 0$ as atomic formulas. At this point, we are concerned mainly with the *structure* of logical proofs.

$$\begin{array}{llcll}
\text{atomic formulas} & P, Q & & & \\
\text{formulas} & A, B, C & ::= & P & \text{atomic formula} \\
 & & & \mid A \supset B & \text{implication} \\
 & & & \mid A \wedge B & \text{conjunction (and)} \\
 & & & \mid A \vee B & \text{disjunction (or)} \\
 & & & \mid \forall a : \mathsf{nat}.\ A & \text{universal quantification} \\
 & & & \mid \exists a : \mathsf{nat}.\ A & \text{existential quantification}
\end{array}$$

$\boxed{A\ \mathsf{true}}$ A is true

$$\dfrac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\ {\wedge}\mathrm{Intro} \qquad \dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\ {\wedge}\mathrm{Elim1} \qquad \dfrac{A \wedge B\ \mathsf{true}}{B\ \mathsf{true}}\ {\wedge}\mathrm{Elim2}$$

$$\dfrac{\begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true} \end{array}}{(A \supset B)\ \mathsf{true}}\ {\supset}\mathrm{Intro}^{x} \qquad \dfrac{A \supset B\ \mathsf{true} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\ {\supset}\mathrm{Elim}$$

Rule ⊃Elim is modus ponens, but rule ⊃Intro requires an *assumption*. The assumption "floats" above the subderivation in which it is available; it is available between the floating assumption $x\,[A\ \mathsf{true}]$ to its point of introduction, marked with the superscript x next to the rule name. I have highlighted the x in the rule, but we may have to work without highlighting.

This notation for assumptions is somewhat unfriendly: it's easy to lose track of the scope of the assumption. But this notation may be closer to ordinary mathematical reasoning, and it's historically important, so we'll keep using it for now.

Disjunction usually causes more trouble than conjunction, and this certainly holds for natural deduction. However, there is a duality between conjunction and disjunction. In Boolean logic, you can get a feeling for this by comparing the truth table for AND with the truth table for OR, after swapping "true" and "false" in one of them. In natural deduction, this opposition between conjunction and disjunction shows itself in a similarity between the elimination rules for ∧, which are ∧Elim1/∧Elim2, and the introduction rules for ∨:

$$\dfrac{A\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ {\vee}\mathrm{Intro1} \qquad \dfrac{B\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ {\vee}\mathrm{Intro2}$$

Observe that ∨Intro1 is ∧Elim1 turned upside down, with ∧ changed to ∨.

Sadly, flipping the *introduction* rule ∧Intro upside down doesn't give us a good ∨-elimination rule:

$$\dfrac{A \vee B\ \mathsf{true}}{A\ \mathsf{true} \qquad B\ \mathsf{true}}\ ??{\vee}\mathrm{Elim}??$$

This rule seems to have two conclusions. People sometimes write more than one conclusion as a concise notation for two rules with identical premises—we could combine $\wedge$Elim1 and $\wedge$Elim2, for example. But it's certainly not true that, from "A or B", we should get A *and* B.

Instead, our $\vee$Elim rule will *reason by cases*. If $A \vee B$ then either A, or B. As we can split a proof into cases according to a given grammar ("Case $e = n$…Case $e = (+\ e_1\ e_2)$"), we can have subderivations with different assumptions.

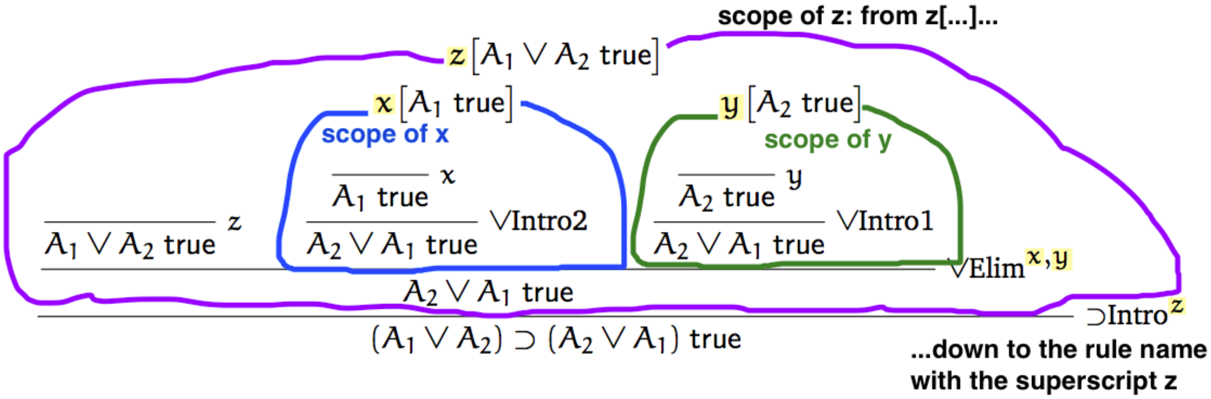
$$\dfrac{A \vee B\ \mathsf{true} \qquad \begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true}\end{array} \qquad \begin{array}{c} y\,[B\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true}\end{array}}{C\ \mathsf{true}}\ \vee\mathrm{Elim}^{x,y}$$

To understand why the conclusion of $\vee$Elim should be $C\ \mathsf{true}$, where C is *any* formula, it may help to think about case analysis in a (line-by-line) proof: we can case-analyze regardless of what our goal is. We might be trying to show that $v = 0$, or that $v_1 = v_2$, or that $e \Downarrow v$; whatever the goal, case analysis works the same way. The only requirement is that each case must show the same goal: if we want to show $v = 0$ we need to show $v = 0$ assuming $e = n$, and $v = 0$ assuming $e = (+\ e_1\ e_2)$.

4.1 Example

$$\begin{array}{c} z\,[A_1 \vee A_2\ \mathsf{true}] \\ x\,[A_1\ \mathsf{true}] \qquad\qquad\qquad y\,[A_2\ \mathsf{true}] \end{array}$$

$$\dfrac{\dfrac{\dfrac{}{A_1 \vee A_2\ \mathsf{true}}\,z \qquad \dfrac{\dfrac{}{A_1\ \mathsf{true}}\,x}{A_2 \vee A_1\ \mathsf{true}}\,\vee\mathrm{Intro2} \qquad \dfrac{\dfrac{}{A_2\ \mathsf{true}}\,y}{A_2 \vee A_1\ \mathsf{true}}\,\vee\mathrm{Intro1}}{A_2 \vee A_1\ \mathsf{true}}\,\vee\mathrm{Elim}^{x,y}}{(A_1 \vee A_2) \supset (A_2 \vee A_1)\ \mathsf{true}}\,\supset\mathrm{Intro}^{z}$$



5 Natural deduction, extended

$$\begin{array}{llcll}
\text{atomic formulas} & P, Q & & & \\
\text{formulas} & A, B, C & ::= & P & \text{atomic formula} \\
 & & & \mid A \supset B & \text{implication} \\
 & & & \mid A \wedge B & \text{conjunction (and)} \\
 & & & \mid A \vee B & \text{disjunction (or)} \\
 & & & \mid \forall a : \mathsf{nat}.\, A & \text{universal quantification} \\
 & & & \mid \exists a : \mathsf{nat}.\, A & \text{existential quantification} \\
 & & & \mid \mathsf{True} & \text{truth}
\end{array}$$

$\boxed{A\ \mathsf{true}}$ A is true

$$\dfrac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\ \wedge\text{Intro} \qquad \dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\ \wedge\text{Elim1} \qquad \dfrac{A \wedge B\ \mathsf{true}}{B\ \mathsf{true}}\ \wedge\text{Elim2}$$

$$\dfrac{\begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true} \end{array}}{(A \supset B)\ \mathsf{true}}\ \supset\text{Intro}^{x} \qquad \dfrac{A \supset B\ \mathsf{true} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\ \supset\text{Elim}$$

$$\dfrac{A\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ \vee\text{Intro1} \qquad \dfrac{B\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ \vee\text{Intro2} \qquad \dfrac{A \vee B\ \mathsf{true} \qquad \begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true} \end{array} \qquad \begin{array}{c} y\,[B\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true} \end{array}}{C\ \mathsf{true}}\ \vee\text{Elim}^{x,y}$$

$$\dfrac{}{\mathsf{True}\ \mathsf{true}}\ \mathsf{True}\text{Intro} \qquad \text{no elimination rules for } \mathsf{True}$$

5.1 True

To design rules for the formula True, it may be helpful to view it as an "and" of nothing—a 0-ary conjunction. Since $\wedge$ is a binary (2-ary) conjunction whose introduction rule $\wedge$Intro has two premises, following that structure leads to the rule TrueIntro, which has zero premises. This seems consistent with an intuitive understanding of True: since True has no subformulas, its truth does not depend on the truth of the subformulas (unlike $\wedge$ where $A \wedge B$ is true only if A and B are true).

For the elimination rule(s), we can also argue by analogy to $\wedge$. However, the argument feels a little different from the argument for the introduction rule.

$$\dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\ \wedge\text{Elim1} \qquad \dfrac{A \wedge B\ \mathsf{true}}{B\ \mathsf{true}}\ \wedge\text{Elim2}$$

The trick is to find something about the above rules related to the number two. But the only thing related to two *within* $\wedge$Elim1 and $\wedge$Elim2 is the formula $A \wedge B$ itself: each rule has one premise. Instead, we must step back and observe that the *number of elimination rules* is two. Since $\wedge$ is 2-ary and True is 0-ary, this suggests that we should have zero elimination rules for True!

We can justify this by analogy to $\wedge$, but more intuitively: the formula $A \wedge B$ combines two facts (A is true and B is true), leading to two elimination rules, each extracting one of those two facts. But no facts are needed to justify True. Therefore, no facts can be extracted. Perhaps we could argue that True itself could be extracted:

$$\frac{\mathsf{True}\ \mathsf{true}}{\mathsf{True}\ \mathsf{true}}$$

But this rule is admissible (that is, redundant): we already have TrueIntro, which has the same conclusion. (Any rule with a premise that is identical to its conclusion is admissible.)

Allowing rules to have multiple conclusions is something that I'm trying to avoid, because it can be confusing, but allowing that leads to another argument for having no elimination rules. If we allow multiple conclusions then we can combine $\wedge$Elim1 and $\wedge$Elim2:

$$\frac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true} \qquad B\ \mathsf{true}}\ \wedge\text{Elim-combined}$$

Since this rule has two conclusions, TrueElim should have zero conclusions. But a rule with no conclusions isn't a rule; even if we tried to bend the definition to allow it, it can't conclude anything because it has no conclusion.

6 Harmony

The above arguments for designing rules for True have "intensional flavour": we argued for our design based on existing internal features—our rules for $\wedge$—of the system (and *then* checked the resulting rules against our intuitive understanding of True).

This seems to run against Carnap's aphorism, "In logic, there are no morals." It suggests that we have some constraints around how the parts of the system fit together.

While I argued (in lecture, not written up here yet) by analogy to true as an identity element for $\wedge$, that wasn't strictly necessary. Whether or not we believe that $\wedge$ comports with common usage of the word "and", we can ask: If $\wedge$ is the 2-ary version of something, what is the 0-ary version? We can choose to call that 0-ary version "True", or "tonk", or even "False" (if we enjoy confusion).

One intensional quality-assurance tool doesn't even need to compare the rules for different connectives: *harmony* checks that the introduction rules and elimination rules *for a single connective* match (are *in harmony* with) each other. Harmony has two parts:

- *Local soundness* holds when the results of applying elimination rules were already used in the introduction rule. On a high level, facts go into an introduction rule; the elimination rules should produce only those facts.
- *Local completeness* holds when the elimination rules can be used to recover *all* of the facts that went into the introduction.

Checking that rules satisfy these two parts of harmony gives us some protection against two possible design mistakes: neglecting to add a necessary rule, and adding a rule that is too powerful. Specifically:

- If local soundness is violated, either an elimination rule is wrong (it is producing something outside the "inputs" to the introduction rule), or we forgot an introduction rule.

- If local completeness is violated, either we forgot an elimination rule, or an introduction rule is wrong.

It's probably easiest to grasp these ideas by considering some relatively clear mistakes in designing rules for conjunction.

$$\dfrac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\wedge\mathrm{Intro} \qquad \dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\wedge\mathrm{Elim1}$$

Suppose the above two rules were our only rules for $\wedge$. Local soundness holds, because our (only) elimination rule $\wedge$Elim1 produces $A\ \mathsf{true}$, which "went into" our use of $\wedge$Intro:

$$\dfrac{\dfrac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\wedge\mathrm{Intro}}{A\ \mathsf{true}}\wedge\mathrm{Elim1}$$

But local completeness fails, because our single elimination rule can't recover the information $B\ \mathsf{true}$. Checking local completeness ensures that we remember to include both elimination rules.

On the other hand, suppose we have both elimination rules but forget one of the premises of $\wedge$Intro.

$$\dfrac{A\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\wedge\mathrm{Intro??} \qquad \dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\wedge\mathrm{Elim1} \qquad \dfrac{A \wedge B\ \mathsf{true}}{B\ \mathsf{true}}\wedge\mathrm{Elim2}$$

Checking local *soundness* will reveal the problem:

$$\dfrac{\dfrac{A\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\wedge\mathrm{Intro??}}{B\ \mathsf{true}}\wedge\mathrm{Elim2}$$

The rule $\wedge$Elim2 derives $B\ \mathsf{true}$, but $B\ \mathsf{true}$ didn't go into our (wrong) introduction rule $\wedge$Intro??, so $\wedge$Elim2 is locally unsound with respect to $\wedge$Intro??.

Local soundness and local completeness are not quite the same as soundness and completeness between different systems, but they are similar in that they depend on keeping *something* in a "fixed position" and comparing other stuff to the fixed thing: Ordinary soundness takes some system as ground truth, and checks that another system stays within that ground truth; ordinary completeness asks whether another system covers everything within that ground truth. Local soundness keeps the introduction rules stationary, and ensures that the elimination rules stay "within the scope" of the introduction rules. Local completeness also keeps the introduction rules stationary, and checks that the elimination rules can recover all of the information used by the introduction rules.

Let's check that our rules for $\supset$ satisfy local soundness and local completeness.

$$\dfrac{\begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true} \end{array}}{(A \supset B)\ \mathsf{true}}\supset\mathrm{Intro}^{x} \qquad \dfrac{A \supset B\ \mathsf{true} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\supset\mathrm{Elim}$$

6.1 Local soundness for ⊃

For each elimination rule for ⊃, we ask if that rule is locally sound. We have one elimination rule for ⊃; is it locally sound?

$$\dfrac{\dfrac{\begin{matrix} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true}\end{matrix}}{(A \supset B)\ \mathsf{true}}\,{\supset}\mathrm{Intro}^{x} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\,{\supset}\mathrm{Elim}$$

This is a little more complicated than ∧, because ⊃Elim has a second premise A true. We can apply ⊃Elim only when A true. So the question becomes: did the information

> “assuming A true [the other premise of ⊃Elim], it holds that B true [the conclusion of ⊃Elim]”

go into the application of ⊃Intro? Yes, because the (only) premise of ⊃Intro derived B true *under the assumption* A true.

6.2 Local completeness for ⊃

$$\dfrac{\dfrac{\begin{matrix} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true}\end{matrix}}{(A \supset B)\ \mathsf{true}}\,{\supset}\mathrm{Intro}^{x} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\,{\supset}\mathrm{Elim}$$

For local completeness, we ask whether all the information going into ⊃Intro can be recovered using one of the elimination rules for ⊃. Since there is only one elimination rule, we ask whether the information going into ⊃Intro can be recovered using ⊃Elim. That information was

“assuming A true [the assumption within the premise of ⊃Intro], it holds that B true [the premise of ⊃Intro].”

(This is the same piece of information that we used in local soundness, but only because our rules really do satisfy local soundness and local completeness!) The argument for local completeness goes like this:

1. Assume we have a derivation of $A \supset B$ true whose concluding rule is ⊃Intro.
2. Assume A true.
3. Our goal is to derive B true.
4. Applying rule ⊃Elim to $A \supset B$ true and A true gives B true.

Note that if we had forgotten to write ⊃Elim, we could not take the last step of this argument.

6.3 Local soundness for True

For each elimination rule for True, we ask if that rule is locally sound. We have no elimination rules for True, so there is nothing to check.

6.4 Local completeness for True

For local completeness, we ask whether all the information going into TrueIntro can be recovered using one of the elimination rules for True. But TrueIntro has no premises, so no information went into it. So there is nothing to check.

6.5 Local soundness for $\lor$

For each elimination rule for $\lor$, we ask if that rule is locally sound. We have only one elimination rule for $\lor$, but it is somewhat complicated:

$$\dfrac{\dfrac{\cdots}{A \lor B\ \mathsf{true}}\,\lor\mathrm{Intro}\ldots \qquad \begin{matrix} x\,[A\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true}\end{matrix} \qquad \begin{matrix} y\,[B\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true}\end{matrix}}{C\ \mathsf{true}}\,\lor\mathrm{Elim}^{x,y}$$

Since we have more than one introduction rule for $\lor$, we don't know which of them was used to derive $A \lor B$ true, so I have included some "...".

For $\supset$ we assumed the other (second) premise of $\supset$Elim. So here, we assume the other (second and third) premises of $\lor$Elim:

- Second premise of $\lor$Elim: "assuming A true, then C true."
- Third premise of $\lor$Elim: "assuming B true, then C true."

We also assume the first premise, the derivation of $A \lor B$ true by one of the $\lor$-introduction rules. Consider cases of which introduction rule was used. Our goal is to show C true.

- Case: $\lor$Intro1 was used.

  By inversion on $\lor$Intro1, A true.

  We know ("Second premise of $\lor$Elim") that, if A true, then C true. Since we know A true, we know C true.

- Case: $\lor$Intro2 was used.

  By inversion on $\lor$Intro2, B true.

  We know ("Third premise of $\lor$Elim") that, if B true, then C true. Since we know B true, we know C true.

6.6 Local completeness for $\lor$

Arguing local completeness for $\lor$ is tricky, because $\lor$Elim does not literally produce the information that went into the introduction rule. Instead, it splits into two cases, each working towards a common goal C true, where C may not be the same as A or B. Instead of trying to get the literal information A true (or B true), we must consider what we could deduce if we knew that either A true holds or B true holds.

1. Assume, as usual, $A \lor B$ true by one of the introduction rules for $\lor$; this is the first premise of $\lor$Elim.
2. Also assume, as usual, the remaining premises of $\lor$Elim.
3. Suppose that C' is a formula that can be deduced assuming A, and can be deduced assuming B. Our goal is now to use $\lor$Elim to derive C' true.
4. We want to apply $\lor$Elim. We don't get to choose A and B; they are determined by the pre-existing derivation of $A \lor B$ true. However, we can choose C.
5. Let C be C'.
6. Either $\lor$Intro1 was used to derive $A \lor B$ true, or $\lor$Intro2 was used to derive it.
    - Case: $\lor$Intro1 was used.
      By inversion on $\lor$Intro1, A true.
      The second premise of $\lor$Elim is that C' true under the assumption A true. That is, C' true can be deduced from A true.
    - Case: $\lor$Intro2 was used.
      By inversion on $\lor$Intro2, B true.
      The second premise of $\lor$Elim is that C' true under the assumption B true. That is, C' true can be deduced from B true.
7. By rule $\lor$Elim, C' true.

The business about C' is needed to reject a possible bug in $\lor$Elim: an insufficiently general conclusion. Consider this specialized version of $\lor$Elim, which can express our example in Section 3.1, but nothing else:

$$\dfrac{A \lor B\ \mathsf{true} \qquad \begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ B \lor A\ \mathsf{true} \end{array} \qquad \begin{array}{c} y\,[B\ \mathsf{true}] \\ \vdots \\ B \lor A\ \mathsf{true} \end{array}}{B \lor A\ \mathsf{true}}\ \lor\text{Elim-swap}^{x,y}$$

If $\lor$Elim-swap were our only elimination rule for $\lor$, step 5—Let C be C'—would work only in the special case of $C' = B \lor A$. Since $\lor$Elim-swap cannot derive other conclusions, including—for example—$(A \lor B) \lor A$, our argument fails, as it should: $\lor$Elim-swap is incomplete because it doesn't work for most possible C'.

Note that $\lor$Elim-swap is locally *sound*: it works for only one conclusion, but for that conclusion, it does not go beyond the premise of the introduction rule.

■ **Exercise 1.** Consider the connective tonk, due to A.N. Prior (1960), who argued (tongue-in-cheek) that tonk would succeed based on its "extreme *convenience*". Translated to our notation, Prior gave two rules for tonk:

$$\frac{A \ \mathsf{true}}{(A \ \mathsf{tonk}\ B) \ \mathsf{true}}\ \mathsf{tonkIntro} \qquad \frac{(A \ \mathsf{tonk}\ B) \ \mathsf{true}}{B \ \mathsf{true}}\ \mathsf{tonkElim}$$

Essentially, this connective steals one of the introduction rules for $\vee$ and one of the elimination rules for $\wedge$.

Argue that local soundness for tonk does not hold.

7 Quantifiers and falsehood

For quantifiers, we need some notation that substitutes a specific natural number for the quantified variable. Suppose wehave

$$\forall a : \mathsf{nat}.\ \big(\mathsf{even}(a) \vee \mathsf{odd}(a)\big)$$

and we want to know that the natural number 5 is either even or odd. We can get

$$\big(\mathsf{even}(5) \vee \mathsf{odd}(5)\big)$$

by looking for a (the quantified variable) throughout the body of the quantifier, and wherever we find a, replacing it with 5.

$$\forall a : \mathsf{nat}.\ \underbrace{\big(\mathsf{even}(a) \vee \mathsf{odd}(a)\big)}_{\text{body of the quantifier}}$$

We will write this use of substitution as

$$[5/a]\big(\mathsf{even}(a) \vee \mathsf{odd}(a)\big)$$

or more generally,

$$[n/a]A \quad = \quad A \textit{ with } n \textit{ replacing each occurrence of } a$$

Substitution is a *meta-level* operation, like writing $n_1 + n_2$ in the rule eval-add. As with addition, it would be better to formally define what $[n/a]A$ means. However, the full definition of substitution has some "interesting" parts, which I don't want to explain just yet.

If you're curious about what the interesting parts might be, consider what should happen if we substitute 5 for b in the following:

$$(b > 2) \wedge \Big(\mathsf{prime}(a) \supset \big(\exists b : \mathsf{nat}.\ (b > a) \wedge \mathsf{prime}(b)\big)\Big)$$

7.1 Mnemonic device

The PL research community has not converged on one standard notation for substitution. I have a number of reasons for preferring the notation above, which I won't bore you with. A shortcoming of my preferred notation is that the order of n (the thing replacing the variable) and a (the variable being replaced) is not immediately clear. Here is a memory trick (more snobbily, a *mnemonic device*):

If we look for all occurrences of a throughout a, and replace a with 5, we write that as

$$[5/a]a \;=\; 5$$

If we creatively reinterpret $5/a$ as a fraction and reinterpret substitution as multiplication, we can "cancel" the a:

$$\frac{5}{a} \cdot a \;=\; \frac{5 \cdot \cancel{a}}{\cancel{a}} \;=\; 5$$

7.2 Substitution as renaming

We don't have to substitute a constant natural number like 5. We could also substitute a variable.

$$\big[a'/a\big]\big((b > a) \wedge \mathsf{prime}(b)\big) \;=\; \big((b > a') \wedge \mathsf{prime}(b)\big)$$

We could then substitute a constant for a':

$$\begin{aligned}\big[2/a'\big]\big[a'/a\big]\big((b > a) \wedge \mathsf{prime}(b)\big) &= \big[2/a'\big]\big((b > a') \wedge \mathsf{prime}(b)\big) \\ &= \big((b > 2) \wedge \mathsf{prime}(b)\big)\end{aligned}$$

A nice feature of this notation is that such "double substitutions" have the same variable in the centre: "2 replaces a' which replaces a, so 2 is replacing a", or (reading right to left) "replace a with a', then replace a' with 2".

7.3 Natural deduction, extended again (2018–02–01)

$$
\begin{array}{llll}
\text{atomic formulas} & P, Q & & \\
\text{formulas} & A, B, C & ::= P & \text{atomic formula} \\
 & & \mid A \supset B & \text{implication} \\
 & & \mid A \wedge B & \text{conjunction (and)} \\
 & & \mid A \vee B & \text{disjunction (or)} \\
 & & \mid \forall a : \mathsf{nat}.\, A & \text{universal quantification} \\
 & & \mid \exists a : \mathsf{nat}.\, A & \text{existential quantification} \\
 & & \mid \mathsf{True} & \text{truth} \\
 & & \mid \mathsf{False} & \text{falsehood}
\end{array}
$$

$\boxed{A\ \mathsf{true}}$ A is true

$$
\dfrac{\begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ B\ \mathsf{true} \end{array}}{(A \supset B)\ \mathsf{true}}\ {\supset}\text{Intro}^{x}
\qquad\qquad
\dfrac{A \supset B\ \mathsf{true} \qquad A\ \mathsf{true}}{B\ \mathsf{true}}\ {\supset}\text{Elim}
$$

$$
\dfrac{}{\mathsf{True}\ \mathsf{true}}\ \mathsf{TrueIntro}
\qquad\qquad
\text{no elimination rules for } \mathsf{True}
$$

$$
\dfrac{A\ \mathsf{true} \qquad B\ \mathsf{true}}{A \wedge B\ \mathsf{true}}\ {\wedge}\text{Intro}
\qquad
\dfrac{A \wedge B\ \mathsf{true}}{A\ \mathsf{true}}\ {\wedge}\text{Elim1}
\qquad
\dfrac{A \wedge B\ \mathsf{true}}{B\ \mathsf{true}}\ {\wedge}\text{Elim2}
$$

$$
\dfrac{\begin{array}{c} x\,[a : \mathsf{nat}] \\ \vdots \\ B\ \mathsf{true} \end{array}}{(\forall a : \mathsf{nat}.\, B)\ \mathsf{true}}\ \forall\text{Intro}^{x}
\qquad
\dfrac{(\forall a : \mathsf{nat}.\, B)\ \mathsf{true} \qquad n : \mathsf{nat}}{[n/a]B\ \mathsf{true}}\ \forall\text{Elim}
$$

$$
\dfrac{A\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ {\vee}\text{Intro1}
\qquad
\dfrac{B\ \mathsf{true}}{A \vee B\ \mathsf{true}}\ {\vee}\text{Intro2}
\qquad
\dfrac{A \vee B\ \mathsf{true} \qquad \begin{array}{c} x\,[A\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true} \end{array} \qquad \begin{array}{c} y\,[B\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true} \end{array}}{C\ \mathsf{true}}\ {\vee}\text{Elim}^{x,y}
$$

$$
\text{no introduction rules for } \mathsf{False}
\qquad\qquad
\dfrac{\mathsf{False}\ \mathsf{true}}{C\ \mathsf{true}}\ \mathsf{FalseElim}
$$

$$
\dfrac{n : \mathsf{nat} \qquad \big([n/a]B\big)\ \mathsf{true}}{(\exists a : \mathsf{nat}.\, B)\ \mathsf{true}}\ \exists\text{Intro}
\qquad
\dfrac{(\exists a : \mathsf{nat}.\, B)\ \mathsf{true} \qquad \begin{array}{c} x\,[a : \mathsf{nat}] \\ y\,[B\ \mathsf{true}] \\ \vdots \\ C\ \mathsf{true} \end{array}}{C\ \mathsf{true}}\ \exists\text{Elim}^{x,y}
$$

7.3.1 Motivation for the new rules

Here we have added rules for $\forall$, $\exists$, and False. In class, I motivated the design through various symmetries:

- True is a 0-ary conjunction, $\wedge$ is a 2-ary (binary) conjunction, $\forall$ is an ∞-ary (infinitary) conjunction.
- False is a 0-ary disjunction, $\vee$ is a 2-ary (binary) disjunction, $\exists$ is an ∞-ary (infinitary) disjunction.

For $\forall$Intro, the 2 premises of $\wedge$Intro for the 2-ary $\wedge$ became "infinite premises", one for each natural number.

$$\dfrac{[0/a]B\ \mathsf{true} \qquad [1/a]B\ \mathsf{true} \qquad [2/a]B\ \mathsf{true} \qquad \cdots}{(\forall a : \mathsf{nat}.\ B)\ \mathsf{true}}\ \forall\text{Intro?}$$

Our actual $\forall$Intro rule represents this infinite set of premises as *one* premise with an assumption that a is a natural number.

For $\forall$Elim, the 2 elimination rules of $\wedge$Elim for the 2-ary $\wedge$ became an infinite number of elimination rules.

$$\dfrac{(\forall a : \mathsf{nat}.\ B)\ \mathsf{true}}{[0/a]B\ \mathsf{true}}\ \forall\text{Elim0} \qquad \dfrac{(\forall a : \mathsf{nat}.\ B)\ \mathsf{true}}{[1/a]B\ \mathsf{true}}\ \forall\text{Elim1} \qquad \dfrac{(\forall a : \mathsf{nat}.\ B)\ \mathsf{true}}{[2/a]B\ \mathsf{true}}\ \forall\text{Elim2} \qquad \cdots$$

Since we cannot directly write all the rules in the infinite set $\{\forall\text{Elim0}, \forall\text{Elim1}, \forall\text{Elim2}, \forall\text{Elim3}, \ldots\}$, we replaced them with one rule that requires a specific n:

$$\dfrac{(\forall a : \mathsf{nat}.\ B)\ \mathsf{true} \qquad n : \mathsf{nat}}{[n/a]B\ \mathsf{true}}\ \forall\text{Elim}$$

If we choose $n = 0$, our $\forall$Elim does the same thing as $\forall$Elim0; if we choose $n = 1$, it does the same thing as $\forall$Elim1, and so forth.

Moving on to $\exists$, I waved my hands about duality between $\forall$ and $\exists$—which suggests that aspects of $\forall$Intro, should find their way into $\exists$'s elimination rule, and that aspects of $\forall$Elim should show up in $\exists$'s introduction rule. I also (perhaps more clearly) used our rules for 2-ary disjunction $\vee$ to inform the rules for the infinitary disjunction $\exists$.

Thus, the 2 introduction rules for the 2-ary $\vee$ became an infinite number of introduction rules

$$\dfrac{\big([0/a]B\big)\ \mathsf{true}}{(\exists a : \mathsf{nat}.\ B)\ \mathsf{true}}\ \exists\text{Intro0} \qquad \dfrac{\big([1/a]B\big)\ \mathsf{true}}{(\exists a : \mathsf{nat}.\ B)\ \mathsf{true}}\ \exists\text{Intro1} \qquad \dfrac{\big([2/a]B\big)\ \mathsf{true}}{(\exists a : \mathsf{nat}.\ B)\ \mathsf{true}}\ \exists\text{Intro2} \qquad \cdots$$

which we coalesced into $\exists$Intro, noting that since the *elimination* rule for $\forall$ has a premise $n : \mathsf{nat}$, duality between $\forall$ and $\exists$ suggests that the *introduction* rule for $\exists$ should also assume $a : \mathsf{nat}$ within a premise.

Our elimination rule for $\exists$ has similar structure to our elimination rule for $\vee$. We arrived at that structure by, first, replicating the two C true premises of $\vee$ into an infinite set gives

$$\dfrac{(\exists a : \mathsf{nat}.\ B)\ \mathsf{true} \qquad \begin{matrix} y_0\,\big[[0/a]B\ \mathsf{true}\big] \\ \vdots \\ C\ \mathsf{true} \end{matrix} \qquad \begin{matrix} y_1\,\big[[1/a]B\ \mathsf{true}\big] \\ \vdots \\ C\ \mathsf{true} \end{matrix} \qquad \begin{matrix} y_2\,\big[[2/a]B\ \mathsf{true}\big] \\ \vdots \\ C\ \mathsf{true} \end{matrix} \qquad \cdots}{C\ \mathsf{true}}\ \exists\text{Elim}^{y_0, y_1, y_2, \ldots}$$

Second, we coalesced the infinite premises into a premise under the assumption $a : \mathsf{nat}$.

7.4 Historical notes (optional reading)

7.4.1 Assumptions and substitution

In Gentzen's natural deduction system ("NJ", essentially a misprint of "NI"), the assumptions $x\big[a : \mathsf{nat}\big]$ are not written out. Moreover, Gentzen's rules rename the variable in the quantifier (e.g. the a in $\forall a : \mathsf{nat}.\ B$) to a new variable a': Instead of assuming $a : \mathsf{nat}$ and $B\ \mathsf{true}$, Gentzen assumes $[a'/a]B\ \mathsf{true}$ and calls the new variable a' an *Eigenvariable*. (*Eigen* is German for "own": the eigenvariable "belongs" to the particular application of the rule.)

This "extra" substitution is painless in Gentzen's notation, because he wrote quantifiers differently: instead of $\forall a : \mathsf{nat}.\ A$, with $\forall a : \mathsf{nat}.\ \mathsf{prime}(a)$ as a concrete example, he wrote (roughly) $\forall a : \mathsf{nat}.\ A(a)$. Then the substitution of a' for a in the body $A(a)$ of the quantifier can be written as $A(a')$. If this course were entirely about natural deduction, I might have used Gentzen's notation since it is more compact than square-bracket substitutions, but Gentzen's notation is not suited for code in programming languages. By using square-bracket substitution, we can use the same notation consistently.

7.4.2 Judgment form

Gentzen did not write true in judgments or derivations. For example, Gentzen's original $\wedge$Intro looked like this:

$$\frac{\mathfrak{A} \qquad \mathfrak{B}}{\mathfrak{A} \wedge \mathfrak{B}}\ \wedge\text{–}I$$

This is the same as the usual CISC 204 textbook by Huth and Ryan. CISC 204 focused on one judgment form—"judging" that a formula is true—so writing true was redundant. By contrast, this course deals with many judgment forms:

- that an expression evaluates to a value;
- that an expression takes one step to another expression;
- that an expression takes zero or more steps to another expression;
- that an expression has a particular type;
- that a formula is true (in natural deduction);
- that a formula is true (in sequent calculus)...

7.4.3 Included connectives

Apart from such notational differences, Gentzen's NJ differs from our development in several ways:

- NJ does not include True. Without True, I think our FalseElim would seem less plausible.
- NJ does include negation, which still lurks on our horizon. Also, while Gentzen has our exact FalseElim, he calls it a negation elimination rule... because he *does* have a way to derive False through a negation-elimination rule.

7.4.4 Local soundness

For Gentzen (1934, §5.13), the introduction rules for a connective are its definition. He informally explained (without using the term) an example of local soundness (for $\supset$), noting that "we need not go into the 'informal sense' of the $\supset$-symbol": local soundness is an intensional (internal) property of the rules. Later, the view that introduction rules define a connective became known as *verificationism* (we "verify" that $A \wedge B$ true is derivable by applying the introduction rule for $\wedge$). An opposing view, known as *pragmatism*, regards the elimination rules as "the" definition: the meaning of $A \wedge B$ true is *what you can do with it*.

References

Gerhard Gentzen. Untersuchungen über das logische Schließen. *Mathematische Zeitschrift*, 39: 176–210, 405–431, 1934. English translation, *Investigations into logical deduction*, in M. Szabo, editor, *Collected papers of Gerhard Gentzen* (North-Holland, 1969), pages 68–131.

A. N. Prior. The runabout inference-ticket. *Analysis*, 21(2):38, 1960.