PART 1: TYPE THEORY · CHAPTER 3

Natural Deduction

Alexander V. Gheorghiu
COMP0009 Logic · University College London

CHAPTER 3

Natural Deduction

A logic in which the clerk may suppose.

OPENING

Ten past ten: a support ticket

CASE: THE TICKET

Baker works on the payments service of an online shop. At ten past ten a support ticket lands in her queue: a customer has been charged twice for one order. She opens the runbook.

LINE	STATEMENT	JUSTIFICATION
1.	The first attempt timed out.	supposed
2.	If an attempt times out, the app retries it.	runbook
3.	So the app retried.	from 2 and 1
4.	If the app retries, the card is charged a second time.	runbook
5.	So the card was charged twice.	from 4 and 3
6.	So, *if* the first attempt timed out, the card was charged twice.	from 1–5: paid off

OPENING

Line 1, and then line 6

Baker has not found out whether the first attempt timed out. She has found out what follows *if* it did, and line 6 says so without depending on line 1 at all.

It is a rule. She files it in the runbook and goes to look for a timeout in the gateway's log.

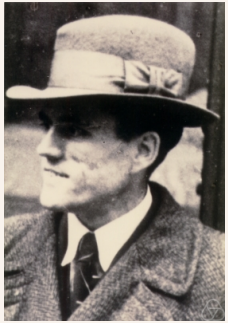
That's how people reason when they don't know the cause: suppose one, follow it to something you can check, and keep the if *when you're done with the supposition.*

OPENING

Gentzen, 1934

In 1934 a young man in Göttingen, Gerhard Gentzen, published a different account of what a proof is, in which line 1 is a permitted move and line 6 is the rule that pays it off.

> “I intended first to set up a formal system which comes as close as possible to actual reasoning. The result was a ‘calculus of natural deduction’.”
>
> Gentzen, 1934



Gerhard Gentzen, 1909–1945

OPENING

What he turned away from

> "The formalization of logical deduction, especially as it has been developed by Frege, Russell, and Hilbert, is rather far removed from the forms of deduction used in practice in mathematical proofs. Considerable formal advantages are achieved in return."
>
> Gentzen, 1934, the sentences before the ones on the last slide

Frege, Russell and Hilbert are the three names behind the clerk, and "considerable formal advantages" is a fair description of what the clerk has: three kinds of line and one rule. Gentzen gave those up.

Nearly everything proof theory has done since 1934 has been done in his systems.

OPENING

Where we are going

SECTION	THE QUESTION
3.1	Derivations in NJ: what are the rules, and how is a supposition paid off?
3.2	Equivalence with the Hilbert system: is anything lost?
3.3	Folding up proofs: what notation shows a proof's structure?
3.4	Removing detours: what is a term good for?
3.5	The simply typed λ-calculus: what does a proof with no detour look like?
3.6	Normalisation: does every proof straighten out?
3.7	Programs as proofs: where do the combinators fit?
3.8	Dropping the types: what do the types buy?
3.9	Fitch proofs (optional): whose boxes have you been drawing?
	History: twenty people, one diamond.

Running example: Baker, on the payments desk, all morning.

SECTION 3.1

Derivations in NJ

The rules, and a bracket round a leaf as a receipt.

3.1 · DERIVATIONS IN NJ

How Baker reasons with *and*

CASE: TWENTY PAST TEN

The runbook says: a charge has settled when the card was authorised (a) *and* the capture was posted (p). Baker has checked each half.

WHAT THE CLERK MUST WRITE

LINE	FORMULA	JUSTIFICATION
	$a,\ p$	
1.	a	hypothesis
2.	p	hypothesis
3.	$a \to p \to a \wedge p$	(pair)
4.	$p \to a \wedge p$	MP 3, 1
5.	$a \wedge p$	MP 4, 2

LINE	HOW BAKER REASONS	WHY
1.	The card was authorised.	checked
2.	The capture was posted.	checked
3.	So both.	1 and 2

Is Baker's line 3 legitimate? The Deduction Theorem says it is.

3.1 · DERIVATIONS IN NJ

Reading (pair) backwards

The schema (pair) is an axiom, and an axiom is a one-line proof, so $\Gamma \vdash \varphi \to (\psi \to \varphi \wedge \psi)$ for every Γ at all. Recall the Deduction Theorem: $\Gamma, \varphi \vdash \chi$ exactly when $\Gamma \vdash \varphi \to \chi$. Apply it twice:

$$\begin{aligned} & \Gamma \vdash \varphi \to (\psi \to \varphi \wedge \psi), \\ \text{so} \quad & \Gamma, \varphi \vdash \psi \to \varphi \wedge \psi, \\ \text{so} \quad & \Gamma, \varphi, \psi \vdash \varphi \wedge \psi. \end{aligned}$$

The arrows have come out of the formula and gone above the line, among the hypotheses.

Γ, φ is short for $\Gamma \cup \{\varphi\}$.

3.1 · DERIVATIONS IN NJ

Splice in the proofs

Now suppose Baker has proved both halves, $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$. Transitivity says a hypothesis you can prove may be replaced by its proof, so she splices each proof in for its hypothesis. What is left is:

$$\text{if } \Gamma \vdash \varphi \text{ and } \Gamma \vdash \psi, \text{ then } \Gamma \vdash \varphi \wedge \psi.$$

The axiom has become a rule*: something you do rather than something you're told. Baker's line 3 is that rule.*

3.1 · DERIVATIONS IN NJ

Proposition 3.12: derived rules

PROPOSITION 3.12 · DERIVED RULES

For all sets of formulas Γ and all formulas φ, ψ, χ, in the Hilbert system:

1. if $\varphi \in \Gamma$ then $\Gamma \vdash \varphi$; — reflexivity
2. if $\Gamma \vdash \varphi \to \psi$ and $\Gamma \vdash \varphi$ then $\Gamma \vdash \psi$; — modus ponens
3. if $\Gamma, \varphi \vdash \psi$ then $\Gamma \vdash \varphi \to \psi$; — the Deduction Theorem
4. if $\Gamma \vdash \varphi$ and $\Gamma \vdash \psi$ then $\Gamma \vdash \varphi \wedge \psi$; — from (pair)
5. if $\Gamma \vdash \varphi \wedge \psi$ then $\Gamma \vdash \varphi$; — from (fst)
6. if $\Gamma \vdash \varphi \wedge \psi$ then $\Gamma \vdash \psi$; — from (snd)

Proposition 3.12, continued

PROPOSITION 3.12 · DERIVED RULES, CONTINUED

7. if $\Gamma \vdash \varphi$ then $\Gamma \vdash \varphi \vee \psi$; from (left)
8. if $\Gamma \vdash \psi$ then $\Gamma \vdash \varphi \vee \psi$; from (right)
9. if $\Gamma \vdash \varphi \vee \psi$, and $\Gamma, \varphi \vdash \chi$, and $\Gamma, \psi \vdash \chi$, then $\Gamma \vdash \chi$; from (either)
10. if $\Gamma \vdash \bot$ then $\Gamma \vdash \varphi$; from (absurd)
11. if $\Gamma \vdash \varphi$ then $\Gamma \vdash \psi \to \varphi$; and if $\Gamma \vdash \varphi \to \psi \to \chi$ and $\Gamma \vdash \varphi \to \psi$ then $\Gamma \vdash \varphi \to \chi$. from (I), (II)

3.1 · DERIVATIONS IN NJ

In Baker's terms

CASE: THE MORNING SO FAR

t: the first attempt timed out. r: the app retried. d: the card was charged twice. a: the card was authorised. p: the capture was posted.

THE FACT	IN THE MORNING'S REASONING
if $\varphi \in \Gamma$ then $\Gamma \vdash \varphi$	what the runbook says may be used
from $\Gamma \vdash t \to r$ and $\Gamma \vdash t$, get $\Gamma \vdash r$	her lines 3 and 5: modus ponens
from $\Gamma, t \vdash d$, get $\Gamma \vdash t \to d$	her line 6: the supposition paid off
from $\Gamma \vdash a$ and $\Gamma \vdash p$, get $\Gamma \vdash a \wedge p$	twenty past ten: each checked, so both

Line 6 is the thing the clerk couldn't do, now a fact about what the clerk can do.

3.1 · DERIVATIONS IN NJ

Two facts that discharge

In two of the facts, a hypothesis appears in a condition and is gone from the conclusion.

$$\text{if } \Gamma, \varphi \vdash \psi \text{ then } \Gamma \vdash \varphi \to \psi$$

$$\text{if } \Gamma \vdash \varphi \vee \psi, \text{ and } \Gamma, \varphi \vdash \chi, \text{ and } \Gamma, \psi \vdash \chi, \text{ then } \Gamma \vdash \chi$$

The hypothesis is discharged: above, it is supposed; below, it is gone. Every other fact keeps its Γ *unchanged.*

3.1 · DERIVATIONS IN NJ

Gentzen's observation

The facts are how people actually reason: to establish both, establish each; to use an implication, apply it; to prove one, suppose and discharge. In the Hilbert system they are theorems *about* proofs. Gentzen wanted a notation in which they *are* the proofs: the facts written as rules, and no axiom ever cited.

CONNECTIVE	TO PROVE IT	TO USE IT
$\wedge$	prove both halves	take either half
$\vee$	prove one side	handle both cases
$\rightarrow$	suppose, then discharge	modus ponens
$\bot$	—	conclude anything

The first kind he called introduction rules, the second elimination rules. Nine rules and no axioms: the system NJ.

3.1 · DERIVATIONS IN NJ

A fact, drawn as a bar

THE FACT, ABOUT THE HILBERT SYSTEM

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

GENTZEN'S BAR

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi}\ \wedge\mathrm{I}$$

Premisses above, conclusion below, the rule on the right. The Γ is not written: it is whatever sits at the leaves.

3.1 · DERIVATIONS IN NJ

The same bar, a different job

CHAPTER 1, AND THE FACT ON THE LEFT

$$\frac{P \rightsquigarrow P'}{P\,Q \rightsquigarrow P'\,Q}\ \mathrm{app}_1$$

Whenever the above holds, so does the below. Nobody draws the stack of rules.

GENTZEN'S BAR

$$\frac{\varphi \to \psi \qquad \varphi}{\psi}\ {\to}\mathrm{E}$$

An instruction for building a tree: trees for the premisses, a bar under them, the conclusion below. The tree is the proof.

The intuition carries over; after Definition 3.3 the "whenever" reading returns, one level up.

3.1 · DERIVATIONS IN NJ

Baker's lines 1 to 5, as a tree

Bars stacked on bars make a tree, read downwards. On the left, the facts with the hypotheses written at every node, $\Gamma = \{t,\ t \to r,\ r \to d\}$; on the right, Gentzen's tree.

WITH THE HYPOTHESES WRITTEN

$$\dfrac{\Gamma \vdash r \to d \qquad \dfrac{\Gamma \vdash t \to r \qquad \Gamma \vdash t}{\Gamma \vdash r}\,{\to}\mathrm{E}}{\Gamma \vdash d}\,{\to}\mathrm{E}$$

GENTZEN'S TREE

$$\dfrac{r \to d \qquad \dfrac{t \to r \qquad t}{r}\,{\to}\mathrm{E}}{d}\,{\to}\mathrm{E}$$

The Γ *on the left* is *the leaves on the right:* t*, which she supposed, and* $t \to r$*,* $r \to d$*, from the runbook. The tree proves its root from its leaves.*

3.1 · DERIVATIONS IN NJ

Line 6: discharge is the Deduction Theorem

Put one more bar under the tree. Now $\Gamma = \{t \to r,\ r \to d\}$, the runbook alone.

THE DEDUCTION THEOREM

$$\frac{\Gamma, t \vdash d}{\Gamma \vdash t \to d}$$

GENTZEN'S TREE

$$\dfrac{\dfrac{r \to d \qquad \dfrac{t \to r \qquad [t]^1}{r}\,{\to}\mathrm{E}}{d}\,{\to}\mathrm{E}}{t \to d}\,{\to}\mathrm{I},1$$

The brackets round t are a receipt: this was supposed, and the bar marked 1 has paid for it. *A leaf in brackets has left the hypotheses, as t leaves the left of the turnstile.*

3.1 · DERIVATIONS IN NJ

And that's all the notation there is

- A proof is a **tree**. Each bar is one of the facts of Proposition 3.12, drawn as a rule.
- A leaf is **open** until a bar below it pays for it, and only two kinds of bar can: ($\to$I) and ($\vee$E).
- What the tree proves is its **root**, *from* its open leaves.

3.1 · DERIVATIONS IN NJ

Definition 3.1: derivations

The derivations are defined inductively; each has a conclusion, the formula at its root.

1. (*hypothesis*) For every formula φ, the one-node tree φ is a derivation, with conclusion φ.
2. ($\to$I) If $\mathcal{D}$ has conclusion ψ, then for every formula φ

$$\dfrac{\begin{array}{c}[\varphi]\\ \mathcal{D}\\ \psi\end{array}}{\varphi \to \psi}\,{\to}\mathrm{I}$$

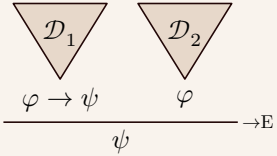
is a derivation.

Each derivation already built is drawn as a triangle, its leaves along the top and its conclusion at the point; a formula in brackets on its top edge is one the new bar discharges.

Definition 3.1, continued: implication, conjunction

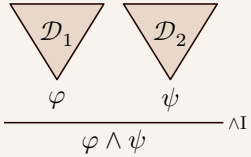
Two trees side by side, one conclusion beneath. Neither bar discharges anything.

3. ($\to$E) If $\mathcal{D}_1$ has conclusion $\varphi \to \psi$ and $\mathcal{D}_2$ has conclusion φ, then



is a derivation, with $\mathrm{OH} = \mathrm{OH}(\mathcal{D}_1) \cup \mathrm{OH}(\mathcal{D}_2)$.

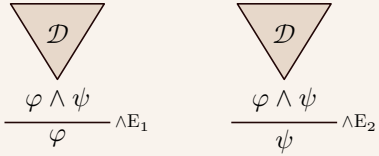
4. ($\wedge$I) If $\mathcal{D}_1$ concludes φ and $\mathcal{D}_2$ concludes ψ, then



is a derivation, with $\mathrm{OH} = \mathrm{OH}(\mathcal{D}_1) \cup \mathrm{OH}(\mathcal{D}_2)$.

Definition 3.1, continued: conjunction, disjunction

5. ($\wedge$E) If $\mathcal{D}$ concludes $\varphi \wedge \psi$, then

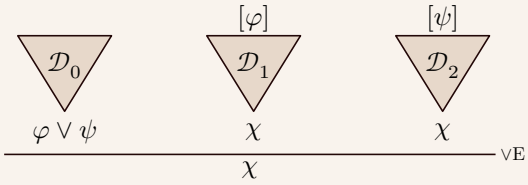


   are derivations.

6. ($\vee$I) If $\mathcal{D}$ concludes φ, then for every formula ψ the tree with root $\varphi \vee \psi$ and immediate subtree $\mathcal{D}$ is a derivation, labelled $\vee I_1$; symmetrically from a derivation of ψ, labelled $\vee I_2$. In both cases $\mathrm{OH} = \mathrm{OH}(\mathcal{D})$.

Definition 3.1, continued: cases, absurdity

7. ($\vee$E) If $\mathcal{D}_0$ concludes $\varphi \vee \psi$, and $\mathcal{D}_1$ and $\mathcal{D}_2$ both conclude χ, then three trees combine into one:



8. ($\bot$E) If $\mathcal{D}$ concludes $\bot$, then for every formula χ the tree with root χ and immediate subtree $\mathcal{D}$ is a derivation.
9. Nothing else is a derivation. (Negation is an abbreviation: $\neg\varphi$ is $\varphi \to \bot$.)

3.1 · DERIVATIONS IN NJ

Definition 3.3: open hypotheses

A tree proves its root from the leaves no bar below them has paid for. Which leaves those are is computed by cases on the last rule, with $\mathcal{D}'$ the one subderivation of a one-premiss rule, $\mathcal{D}_1$, $\mathcal{D}_2$ those of a two-premiss rule, $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_2$ those of ($\vee$E):

$$\mathrm{OH}(\mathcal{D}) := \begin{cases} \{\varphi\} & \text{a leaf } \varphi, \\ \mathrm{OH}(\mathcal{D}') \setminus \{\varphi\} & (\to\mathrm{I}), \text{ discharging } \varphi, \\ \mathrm{OH}(\mathcal{D}_1) \cup \mathrm{OH}(\mathcal{D}_2) & (\to\mathrm{E}),\ (\wedge\mathrm{I}), \\ \mathrm{OH}(\mathcal{D}') & (\wedge\mathrm{E}),\ (\vee\mathrm{I}),\ (\bot\mathrm{E}), \\ \mathrm{OH}(\mathcal{D}_0) \cup (\mathrm{OH}(\mathcal{D}_1) \setminus \{\varphi\}) & (\vee\mathrm{E}), \text{ discharging } \varphi,\ \psi. \\ \quad \cup\, (\mathrm{OH}(\mathcal{D}_2) \setminus \{\psi\}) & \end{cases}$$

Only the ($\to$I) and ($\vee$E) lines take something away: the two rules that discharge.

Definition 3.3, continued: derivability

We write $\Gamma \vdash_{\mathrm{N}} \varphi$, and say φ is derivable from Γ, when there is a derivation whose conclusion is φ and whose open hypotheses are a **subset** of Γ: $\mathrm{OH} \subseteq \Gamma$. Such a derivation witnesses $\Gamma \vdash_{\mathrm{N}} \varphi$.

BAKER'S LINES 1 TO 5

$$\dfrac{\dfrac{r \to d \qquad \dfrac{t \to r \qquad t}{r}\,{\to}\mathrm{E}}{d}\,{\to}\mathrm{E}}{}$$

It witnesses $t,\ t \to r,\ r \to d \vdash_{\mathrm{N}} d$.

BAKER'S LINE 6

$$\dfrac{\dfrac{\dfrac{r \to d \qquad \dfrac{t \to r \qquad [t]^{1}}{r}\,{\to}\mathrm{E}}{d}\,{\to}\mathrm{E}}{t \to d}\,{\to}\mathrm{I},1}{}$$

It witnesses $t \to r,\ r \to d \vdash_{\mathrm{N}} t \to d$.

A subset, not the whole: the tree p witnesses $p,\ q \vdash_{\mathrm{N}} p$. Γ is listed without braces; the N keeps the relation apart from the Hilbert system's $\vdash$ until Section 3.3.

3.1 · DERIVATIONS IN NJ

The bar, read through the turnstile

One level up, each rule says what it does to derivability: whenever the above, then the below.

$$\frac{\Gamma, \varphi \vdash_{\mathrm{N}} \psi}{\Gamma \vdash_{\mathrm{N}} \varphi \to \psi}\ {\to}\mathrm{I} \qquad \frac{\Gamma \vdash_{\mathrm{N}} \varphi \to \psi \qquad \Gamma \vdash_{\mathrm{N}} \varphi}{\Gamma \vdash_{\mathrm{N}} \psi}\ {\to}\mathrm{E}$$

- Left: from a proof of ψ from Γ together with φ, to a proof of $\varphi \to \psi$ from Γ alone. The bar has discharged φ.
- Right: modus ponens.
- Why the two readings agree: $\Gamma \vdash_{\mathrm{N}} \varphi$ means a tree exists. Put the trees for the premisses under one more bar; the ($\to$I) and ($\to$E) lines of Definition 3.3 keep OH inside Γ.

3.1 · DERIVATIONS IN NJ

Discharge, read literally

The ($\to$I) and ($\vee$E) lines of Definition 3.3 are the only ones that take something *away* from OH. The ($\to$I) line removes φ from a set, and three things follow. (The numbers only pair bars with brackets.)

- **Multiple discharge.** Every open leaf φ goes at once: a supposition used three times sits at three leaves, and one bar discharges all three.
- **Vacuous discharge.** If there is no open leaf φ, nothing is removed, and the tree is still a derivation.
- **Which bar takes a leaf.** The first bar below it, on its path to the root, that discharges its formula from the part of the tree it stands in. A bar further down finds that leaf already discharged, though it may take other copies.
- ($\vee$E) does the same in each branch: φ leaves the second premiss and ψ the third, and nothing in the first premiss is discharged.

3.1 · DERIVATIONS IN NJ

Example 3.6: $\vdash_{\mathrm{N}} p \to p$, Hilbert then Gentzen

A HILBERT PROOF

LINE	FORMULA	JUSTIFICATION
	no hypotheses	
1.	$(p \to (p \to p) \to p) \to (p \to (p \to p)) \to p \to p$	(II)
2.	$p \to (p \to p) \to p$	(I)
3.	$(p \to (p \to p)) \to p \to p$	MP 1, 2
4.	$p \to (p \to p)$	(I)
5.	$p \to p$	MP 3, 4

A GENTZEN DERIVATION

$$\dfrac{[p]^1}{p \to p}\ {\to}\mathrm{I}, 1$$

Five lines, the first of which nobody would write unprompted; and one bar.

Example 3.6: $\vdash_{\mathrm{N}} p \to p$, the open hypotheses

Here is a derivation witnessing $\vdash_{\mathrm{N}} p \to p$:

$$\dfrac{[p]^1}{p \to p}\ {\to}\mathrm{I}, 1$$

COMPUTING THE OPEN HYPOTHESES

- The leaf p: $\mathrm{OH} = \{p\}$ — a leaf
- The bar: $\mathrm{OH} = \{p\} \setminus \{p\} = \emptyset$ — the $(\to\mathrm{I})$ line

$\mathrm{OH} = \emptyset \subseteq \Gamma$ *for every* Γ*: it depends on nothing at all. That's what a theorem is.*

RULES USED

(*hypothesis*) φ, with $\mathrm{OH} = \{\varphi\}$

$$(\to\mathrm{I})\ \dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi \to \psi}\ {\to}\mathrm{I}$$

OH loses φ

Example 3.7: $p \wedge q \vdash_{\mathrm{N}} q \wedge p$

A HILBERT PROOF

LINE	FORMULA	JUSTIFICATION
	$p \wedge q$	
1.	$p \wedge q$	hypothesis
2.	$p \wedge q \to q$	(snd)
3.	q	MP 2, 1
4.	$p \wedge q \to p$	(fst)
5.	p	MP 4, 1
6.	$q \to p \to q \wedge p$	(pair)
7.	$p \to q \wedge p$	MP 6, 3
8.	$q \wedge p$	MP 7, 5

A GENTZEN DERIVATION

$$\dfrac{\dfrac{p \wedge q}{q}\,\wedge\mathrm{E}_2 \qquad \dfrac{p \wedge q}{p}\,\wedge\mathrm{E}_1}{q \wedge p}\,\wedge\mathrm{I}$$

Its leaves are both $p \wedge q$, so $\mathrm{OH} = \{p \wedge q\}$: it witnesses $p \wedge q \vdash_{\mathrm{N}} q \wedge p$.

Eight lines; three bars, one per step of the argument.

3.1 · DERIVATIONS IN NJ

Example 3.7: $\vdash_{\mathrm{N}} p \wedge q \to q \wedge p$

One more bar, discharging $p \wedge q$:

$$\dfrac{\dfrac{\dfrac{[p \wedge q]^1}{q}\,\wedge\mathrm{E}_2 \qquad \dfrac{[p \wedge q]^1}{p}\,\wedge\mathrm{E}_1}{q \wedge p}\,\wedge\mathrm{I}}{p \wedge q \to q \wedge p}\,\to\mathrm{I},\,1$$

COMPUTING THE OPEN HYPOTHESES

- Above the last bar: $\{p \wedge q\}$.
- The bar removes $p \wedge q$: $\mathrm{OH} = \emptyset$.

FLAG: MULTIPLE DISCHARGE

The bar marked 1 discharges both leaves $p \wedge q$. However often a supposition is used, it is one supposition, and every open copy goes at once.

The Hilbert system needs the Deduction Theorem here.

3.1 · DERIVATIONS IN NJ

Example 3.9: $\vdash_{\mathrm{N}} p \to p \to p$

Two trees that look like mistakes. The first supposes p and discharges it twice.

$$\dfrac{\dfrac{[p]^{1}}{p \to p}\,{\to}\mathrm{I},1}{p \to p \to p}\,{\to}\mathrm{I},2$$

Both bars discharge p, and there is one leaf. Clause 2 settles it: bar 1 removes every open p, so bar 2 finds none and discharges p vacuously.

FLAG: A DISCHARGE CANNOT BE FORGOTTEN

The brackets record what Definition 3.3 computes. Leave them off a leaf that a bar below it takes, and you get the same derivation, drawn wrongly.

open hypotheses $\emptyset$: a theorem, derivable from any Γ

3.1 · DERIVATIONS IN NJ

Example 3.10: $p \wedge q \vdash_{\mathrm{N}} p \to q \wedge p$

The second: the swap tree, with its last bar discharging p instead.

$$\dfrac{\dfrac{\dfrac{p \wedge q}{q}\,\wedge\mathrm{E}_2 \qquad \dfrac{p \wedge q}{p}\,\wedge\mathrm{E}_1}{q \wedge p}\,\wedge\mathrm{I}}{p \to q \wedge p}\,\to\mathrm{I},\,1$$

It looks wrong, but the ($\to$I) clause asks only that the subtree conclude $q \wedge p$.

FLAG: VACUOUS DISCHARGE

The bar discharges p, and there is no leaf p: no brackets appear, and OH is unchanged. To prove "if p then $q \wedge p$" it is enough to have $q \wedge p$.

open hypotheses $\{p \wedge q\} \subseteq \{p \wedge q\}$

3.1 · DERIVATIONS IN NJ

Example 3.8: $n \vee o,\ n \to l,\ o \to l \vdash_{\mathrm{N}} l$

CASE: HALF PAST TEN

The gateway answered the second attempt, declined (n) or approved (o); the log doesn't say which. Either way, says the runbook, the attempt is in the ledger (l).

$$\dfrac{n \vee o \qquad \dfrac{n \to l \qquad [n]^{1}}{l}\,{\to}\mathrm{E} \qquad \dfrac{o \to l \qquad [o]^{2}}{l}\,{\to}\mathrm{E}}{l}\ {\vee}\mathrm{E}, 1, 2$$

FLAG: TWO FORMULAS, ONE BAR

($\vee$E) discharges n in its second premiss and o in its third, each under its own number.

open hypotheses $\{n \vee o,\ n \to l,\ o \to l\} \subseteq \{n \vee o,\ n \to l,\ o \to l\}$

3.1 · DERIVATIONS IN NJ

Example 3.11: $\vdash_{\mathrm{N}} p \vee p \to p$

Discharging in one branch. First, both cases suppose the same formula, and each branch is a single leaf.

$$\dfrac{\dfrac{[p \vee p]^{1} \qquad [p]^{2} \qquad [p]^{3}}{p}\ \vee\mathrm{E}, 2, 3}{p \vee p \to p}\ \to\mathrm{I}, 1$$

The one-node tree p is a tree from p to p; once the bar discharges it, it leaves nothing open.

FLAG: TWO DISCHARGES, ONE BAR

Both suppositions are p, but the ($\vee$E) bar discharges them separately, one per branch, under the numbers 2 and 3.

open hypotheses $\emptyset$: a theorem, derivable from any Γ

3.1 · DERIVATIONS IN NJ

Example 3.11: $p \vee q,\ p \vdash_{\mathrm{N}} p$

Second, the same letter has two fates.

$$\dfrac{p \vee q \qquad [p]^{1} \qquad p}{p}\ \vee\mathrm{E}, 1, 2$$

The third premiss must go from q to p, and the one-node tree p does. It uses no q, so discharge 2 is vacuous, and its p stays open: ($\vee$E) removes p only from the second premiss.

FLAG: VACUOUS IN ONE BRANCH

One bar can discharge properly in one branch and vacuously in the other, and a formula discharged in one branch may stand open in another.

open hypotheses $\{p \vee q,\ p\} \subseteq \{p \vee q,\ p\}$

3.1 · DERIVATIONS IN NJ

$\vdash_{\mathrm{N}} p \vee q \to q \vee p$

The shape of Example 3.8, with the disjunction itself discharged at the end, so that nothing is left open.

$$\dfrac{\dfrac{[p \vee q]^{1} \qquad \dfrac{[p]^{2}}{q \vee p}\,\vee\mathrm{I}_{2} \qquad \dfrac{[q]^{3}}{q \vee p}\,\vee\mathrm{I}_{1}}{q \vee p}\,\vee\mathrm{E},2,3}{p \vee q \to q \vee p}\,\to\mathrm{I},1$$

Three suppositions, three receipts.

open hypotheses $\emptyset$: a theorem, derivable from any Γ

3.1 · DERIVATIONS IN NJ

$b \to g,\ \neg g \vdash_{\mathrm{N}} \neg b$

CASE: THE RIVAL EXPLANATION

Perhaps the customer clicked "pay" twice (b). Then there would be two order numbers (g); the ledger shows one. She has $b \to g$ and $\neg g$, and wants $\neg b$.

$$\dfrac{\dfrac{\neg g \qquad \dfrac{b \to g \qquad [b]^{1}}{g}\,{\to}\mathrm{E}}{\bot}\,{\to}\mathrm{E}}{\neg b}\,{\to}\mathrm{I},1$$

Modus tollens: suppose the cause, derive the symptom, see it absent.

open hypotheses $\{b \to g,\ \neg g\} \subseteq \{b \to g,\ \neg g\}$

3.1 · DERIVATIONS IN NJ

$h,\ h \to m,\ \neg m \vdash_{\mathrm{N}} l$

CASE: THE LOGS DISAGREE

The retry was sent (h), says the app's log. The runbook says a sent retry is received by the gateway (m). The gateway's log says nothing was received. From these Baker can conclude anything.

$$\dfrac{\dfrac{\dfrac{\neg m \qquad \dfrac{h \to m \qquad h}{m}\,{\to}\mathrm{E}}{\bot}\,{\to}\mathrm{E}}{l}\,{\bot}\mathrm{E}}$$

All three leaves are open. Correct and useless, and that's the point.

open hypotheses $\{h,\ h \to m,\ \neg m\} \subseteq \{h,\ h \to m,\ \neg m\}$

3.1 · DERIVATIONS IN NJ

$\vdash_{\mathrm{N}} (p \wedge q \to r) \to p \to q \to r$

Currying: turning a function on pairs into one that takes its two arguments one at a time.

$$\dfrac{\dfrac{\dfrac{\dfrac{[p \wedge q \to r]^1 \qquad \dfrac{[p]^2 \qquad [q]^3}{p \wedge q}\,\wedge\mathrm{I}}{r}\,\to\mathrm{E}}{q \to r}\,\to\mathrm{I},3}{p \to q \to r}\,\to\mathrm{I},2}{(p \wedge q \to r) \to p \to q \to r}\,\to\mathrm{I},1$$

One discharge per arrow in the conclusion.

open hypotheses $\emptyset$: a theorem, derivable from any Γ

3.1 · DERIVATIONS IN NJ

$\vdash_{\mathrm{N}} (p \to q \to r) \to p \wedge q \to r$

Currying the other way: one supposition taken apart twice.

$$\dfrac{\dfrac{\dfrac{\dfrac{[p \to q \to r]^1 \qquad \dfrac{[p \wedge q]^2}{p}\,\wedge\mathrm{E}_1}{q \to r}\,\to\mathrm{E} \qquad \dfrac{[p \wedge q]^2}{q}\,\wedge\mathrm{E}_2}{r}\,\to\mathrm{E}}{p \wedge q \to r}\,\to\mathrm{I},2}{(p \to q \to r) \to p \wedge q \to r}\,\to\mathrm{I},1$$

FLAG: MULTIPLE DISCHARGE

The bar marked 2 discharges both leaves $p \wedge q$. Again one discharge per arrow.

open hypotheses $\emptyset$: a theorem, derivable from any Γ

3.1 · DERIVATIONS IN NJ

Checking a tree

- Every bar has premisses of the shape its clause requires.
- Every bracket carries a number, and the bar with that number is the *first* bar below the leaf, on its path to the root, that discharges its formula from that part of the tree.
- An ($\to$I) bar discharges every open copy of its formula above it, and is vacuous only if there is none. An ($\vee$E) bar discharges the left side in its second premiss and the right side in its third, and nothing in its first.
- The leaves left without brackets are the open hypotheses. The tree witnesses $\Gamma \vdash_{\mathrm{N}} \varphi$ for every Γ that contains them.

That's the clerk's job again, done on a tree instead of a list.

SECTION 3.2

Equivalence with the Hilbert system

Gentzen's rules lose nothing.

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

Theorem 3.14: equivalence with the Hilbert system

We can now state the theorem promised at the start of the chapter. NJ has no axioms and nine rules; the Hilbert system has nine schemas and one rule. They prove the same things.

THEOREM 3.14 · EQUIVALENCE WITH THE HILBERT SYSTEM

For every set of formulas Γ and every formula φ built from letters and $\bot$ with $\to$, $\wedge$ and $\vee$:

$$\Gamma \vdash_{\mathrm{N}} \varphi \quad \text{if and only if} \quad \Gamma \vdash \varphi.$$

Right to left: induction on the Hilbert proof. Left to right: rule induction on the derivation.

Right to left: from $\vdash$ to $\vdash_{\mathrm{N}}$

Take a Hilbert proof $\varphi_1, \ldots, \varphi_n$ of φ from Γ. Going down it, show $\Gamma \vdash_{\mathrm{N}} \varphi_k$ for every line k.

- **Hypothesis**, $\varphi_k \in \Gamma$: the one-node tree φ_k has $\mathrm{OH} = \{\varphi_k\} \subseteq \Gamma$, so $\Gamma \vdash_{\mathrm{N}} \varphi_k$.
- **Modus ponens** on earlier lines: put an ($\to$E) bar under the trees already built. Its OH is the union of theirs, still inside Γ:

$$\frac{\Gamma \vdash_{\mathrm{N}} \varphi_j \to \varphi_k \qquad \Gamma \vdash_{\mathrm{N}} \varphi_j}{\Gamma \vdash_{\mathrm{N}} \varphi_k}$$

- **Axiom**: every instance of the nine schemas has a derivation with $\mathrm{OH} = \emptyset$, so $\vdash_{\mathrm{N}}$ it, and so $\Gamma \vdash_{\mathrm{N}}$ it. The next three slides.

At the last line, $\Gamma \vdash_{\mathrm{N}} \varphi$.

The axioms: (I)

IN THE HILBERT SYSTEM

LINE	FORMULA	JUSTIFICATION
	no hypotheses	
1.	$\varphi \to \psi \to \varphi$	(I)

IN NATURAL DEDUCTION

$$\dfrac{\dfrac{[\varphi]^1}{\psi \to \varphi}\,{\to}\mathrm{I},2}{\varphi \to \psi \to \varphi}\,{\to}\mathrm{I},1$$

Suppose φ, suppose ψ, conclude φ; then discharge ψ, and then φ. The bar marked 2 discharges something that never appears.

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

The axioms: (I), the open hypotheses

$$\dfrac{\dfrac{[\varphi]^1}{\psi \to \varphi}\,{\to}\mathrm{I},2}{\varphi \to \psi \to \varphi}\,{\to}\mathrm{I},1$$

- The leaf: $\{\varphi\}$.
- Bar 2 removes ψ. There is none, so still $\{\varphi\}$.
- Bar 1 removes φ: $\mathrm{OH} = \emptyset$.

So $\vdash_{\mathrm{N}} \varphi \to \psi \to \varphi$. A supposition you never used can still be paid off.

(When ψ is φ, bar 2 takes the leaf and bar 1 is the vacuous one, as in Example 3.9; $\mathrm{OH} = \emptyset$ either way.)

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

The axioms: (II)

Three suppositions and three discharges, the first place the numbers on the bars earn their keep:

$$\dfrac{\dfrac{\dfrac{\dfrac{[\varphi \to \psi \to \chi]^1 \qquad [\varphi]^3}{\psi \to \chi}\,{\to}\mathrm{E} \qquad \dfrac{[\varphi \to \psi]^2 \qquad [\varphi]^3}{\psi}\,{\to}\mathrm{E}}{\chi}\,{\to}\mathrm{E}}{\varphi \to \chi}\,{\to}\mathrm{I},\,3}{(\varphi \to \psi) \to \varphi \to \chi}\,{\to}\mathrm{I},\,2}{(\varphi \to \psi \to \chi) \to (\varphi \to \psi) \to \varphi \to \chi}\,{\to}\mathrm{I},\,1$$

$\mathrm{OH} = \emptyset$, *so* $\vdash_{\mathrm{N}} (\varphi \to \psi \to \chi) \to (\varphi \to \psi) \to \varphi \to \chi$. *$\varphi$ is used twice and discharged once, by bar 3.*

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

The axioms: the others

Each of the other schemas has a derivation with no open hypotheses, built the same way: suppose the antecedents, use the rules, discharge. Three of them:

(PAIR)

$$\dfrac{\dfrac{\dfrac{[\varphi]^1 \qquad [\psi]^2}{\varphi \wedge \psi}\,\wedge\mathrm{I}}{\psi \to \varphi \wedge \psi}\,\to\mathrm{I},2}{\varphi \to \psi \to \varphi \wedge \psi}\,\to\mathrm{I},1$$

(SND)

$$\dfrac{\dfrac{[\varphi \wedge \psi]^1}{\psi}\,\wedge\mathrm{E}_2}{\varphi \wedge \psi \to \psi}\,\to\mathrm{I},1$$

(RIGHT)

$$\dfrac{\dfrac{[\psi]^1}{\varphi \vee \psi}\,\vee\mathrm{I}_2}{\psi \to \varphi \vee \psi}\,\to\mathrm{I},1$$

Every axiom of the Hilbert system is a theorem of NJ.

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

Left to right: from $\vdash_{\mathrm{N}}$ to $\vdash$

Show, by induction on the derivation: if a derivation of ψ has open hypotheses Δ, then $\Delta \vdash \psi$ in the Hilbert system. Then $\mathrm{OH} \subseteq \Gamma$ and monotonicity give $\Gamma \vdash \varphi$.

- **One node**, φ: its open hypotheses are $\{\varphi\}$, and $\{\varphi\} \vdash \varphi$ by reflexivity.
- **($\to$E)**, under derivations of $\varphi \to \psi$ and φ with open hypotheses Δ_1 and Δ_2: the new tree has open hypotheses $\Delta_1 \cup \Delta_2$, and monotonicity and modus ponens give

$$\frac{\Delta_1 \vdash \varphi \to \psi \qquad \Delta_2 \vdash \varphi}{\Delta_1 \cup \Delta_2 \vdash \psi}$$

- **($\wedge$I), ($\wedge$E), ($\vee$I), ($\bot$E)**: the same, each with the fact for its connective from Proposition 3.12.

3.2 · EQUIVALENCE WITH THE HILBERT SYSTEM

Left to right: the bars that discharge

($\to$I). Let the derivation $\mathcal{D}$ of ψ have open hypotheses Δ; it may or may not contain φ.

THE TREE

$$\dfrac{\begin{array}{c}[\varphi]\\ \mathcal{D}\\ \psi\end{array}}{\varphi \to \psi}\ \to\text{I}$$

Open hypotheses: Δ above the bar, $\Delta \setminus \{\varphi\}$ below it.

IN THE HILBERT SYSTEM

	$\Delta \vdash \psi$	induction hypothesis
so	$(\Delta \setminus \{\varphi\}) \cup \{\varphi\} \vdash \psi$	monotonicity
so	$\Delta \setminus \{\varphi\} \vdash \varphi \to \psi$	Deduction Theorem

Both end with $\Delta \setminus \{\varphi\}$: the bracket on φ is the Deduction Theorem at work.

($\vee$E) is the same in each branch. So $\vdash_{\mathrm{N}}$ and $\vdash$ agree; from here on, $\vdash$ for both.

SECTION 3.3

Folding up proofs

Which programs are these proofs?

3.3 · FOLDING UP PROOFS

Proofs are programs. Which programs?

WHAT WE HAD

A Hilbert proof is a typed combinator program with the program rubbed out: (I) is K, (II) is S, a hypothesis is a variable, modus ponens is application.

WHAT HAS CHANGED

We have changed what a proof is: a tree, with suppositions that are made and then discharged. So what is the matching notion of **typed program**?

3.3 · FOLDING UP PROOFS

The Deduction Theorem, as a program

The new proofs have one move the Hilbert system lacks as a rule: discharge, the Deduction Theorem. The new programs need a term constructor for it.

THE DEDUCTION THEOREM

$$\frac{\Gamma,\ A \vdash B}{\Gamma \vdash A \to B}$$

In the Hilbert system a theorem; its proof builds the program out of K and S.

WITH THE PROGRAMS WRITTEN ON

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

A single step: the constructor λ.

Above the line, a program M with a parameter $x : A$; below, the function `\x -> M`.

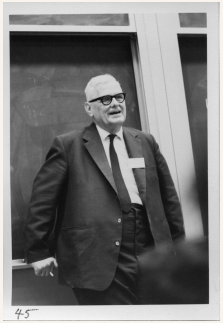
3.3 · FOLDING UP PROOFS

λ came first: Church, 1932

The notation predates all this. In 1932 Alonzo Church, at Princeton, published a logic built from functions, in which every variable is bound by a λ.

His students Kleene and Rosser soon showed the logic inconsistent. The calculus of functions survived.

> "We do not attach any character of uniqueness or absolute truth to any particular system of logic."
>
> Church, 1932



Alonzo Church, 1903–1995

3.8 · DROPPING THE TYPES

Definition 3.46: λ-terms

Fix a stock of variables $x, y, z, f, \ldots$. The terms are defined inductively:

1. every variable x is a term;
2. if M and N are terms then $M\,N$ is a term, the application of M to N;
3. if x is a variable and M is a term then $\lambda x.\, M$ is a term, the abstraction of M over x: the function that takes x and returns M;
4. nothing else is a term.

IN ONE LINE

$$M, N \;::=\; x \;\mid\; M\,N \;\mid\; \lambda x.\, M$$

Application groups to the left. A λ reaches as far right as it can: $\lambda x.\, M\,N$ is $\lambda x.\, (M\,N)$.

3.8 · DROPPING THE TYPES

Example: you have written the third form

LANGUAGE	THE FUNCTION THAT TAKES x AND RETURNS M
Haskell	`\x -> M`
Python	`lambda x: M`
JavaScript	`x => M`
any of them	`def f(x): return M`, a λ with a name attached

These terms, with one rule of computation, are Church's λ-calculus.

3.4 · REMOVING DETOURS

Definition 3.17: free variables

The λ in $\lambda x.\, M$ *binds* the parameter x; its *scope* is the body M. An occurrence of a variable is bound if it lies in the scope of a λ on that variable, and free otherwise.

BY RECURSION ON THE SHAPE

$$\mathrm{FV}(M) := \begin{cases} \{x\} & \text{if } M \text{ is a variable } x, \\ \mathrm{FV}(M_1) \cup \mathrm{FV}(M_2) & \text{if } M \text{ is } M_1\, M_2, \\ \mathrm{FV}(M_1) \setminus \{x\} & \text{if } M \text{ is } \lambda x.\, M_1. \end{cases}$$

EXAMPLE

In $(\lambda x.\, x\, y)\, x$ the first two xs are bound; the last, like the y, is free.

A term with no free variables is closed: a whole program. A bound variable may be renamed at will: $\lambda x.\, x$ is $\lambda y.\, y$.

3.4 · REMOVING DETOURS

Definition 3.18: substitution

$M[N/x]$, "M with N for x": the free occurrences of x in M replaced by N, and nothing else. With binders about, that takes care: it is the rule Curry found far more complicated than modus ponens.

$$M[N/x] := \begin{cases} N & \text{if } M \text{ is } x, \\ y & \text{if } M \text{ is a variable } y \neq x, \\ M_1[N/x]\ M_2[N/x] & \text{if } M \text{ is } M_1\ M_2, \\ \lambda x.\, M_1 & \text{if } M \text{ is } \lambda x.\, M_1, \\ \lambda y.\, M_1[N/x] & \text{if } M \text{ is } \lambda y.\, M_1,\ y \neq x, \text{ provided } y \notin \mathrm{FV}(N). \end{cases}$$

Line four stops at a λx: every x inside is bound. The side condition is the one subtlety: pushing N under a λy with y free in N would capture it. Rename first.

3.4 · REMOVING DETOURS

Example: capture

Compute $(\lambda y.\ x\,y)[y/x]$: the function that applies x to its argument, with y put for x. The answer ought to be the function that applies y to its argument.

BLINDLY

$\lambda y.\ y\,y$

The free y has been caught by the λy: a function that applies its argument to itself. Wrong.

RENAME FIRST

$(\lambda z.\ x\,z)[y/x] = \lambda z.\ y\,z$

The function that applies y to its argument. Right.

3.4 · REMOVING DETOURS

β-reduction

There is one rule of computation: to apply a function to an argument, substitute the argument for the parameter, anywhere inside a term. Four rules, read as in Chapter 1:

$$\frac{}{(\lambda x.\, P)\, Q \;\rightsquigarrow\; P[Q/x]}\;\beta \qquad \frac{P \rightsquigarrow P'}{P\,Q \;\rightsquigarrow\; P'\,Q}\;\mathrm{app}_1$$

$$\frac{Q \rightsquigarrow Q'}{P\,Q \;\rightsquigarrow\; P\,Q'}\;\mathrm{app}_2 \qquad \frac{P \rightsquigarrow P'}{\lambda x.\, P \;\rightsquigarrow\; \lambda x.\, P'}\;\mathrm{abs}$$

The first rule is the step, called β; $(\lambda x.\, P)\, Q$ is a redex. The other three: a step in the function, in the argument, or under a λ is a step of the term. The only reduction we use, so $\rightsquigarrow$ has no subscript. No redex: normal form.

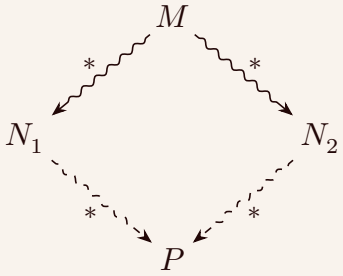
A term may have several redexes. Does it matter which you take first?

3.8 · DROPPING THE TYPES

The Church–Rosser theorem

THEOREM · CHURCH–ROSSER; ON LOAN

If $M \rightsquigarrow^* N_1$ and $M \rightsquigarrow^* N_2$, then there is a term P such that $N_1 \rightsquigarrow^* P$ and $N_2 \rightsquigarrow^* P$.



Two runs of the same term can always be brought back together. So a term has at most one normal form: the order of steps can change whether you reach it, not what it is.

3.8 · DROPPING THE TYPES

Example: running terms

A NUMERAL, APPLIED

$$
\begin{aligned}
&(\lambda f.\, \lambda x.\; f\,(f\,x))\; g\; a \\
\twoheadrightarrow\; &(\lambda x.\; g\,(g\,x))\; a \\
\twoheadrightarrow\; &g\,(g\,a)
\end{aligned}
$$

Two parameters, two β steps. The term is "do f twice".

A TERM THAT NEVER STOPS

With ω for $\lambda x.\; x\,x$:

$$
\begin{aligned}
\Omega \;&= \omega\,\omega \\
&\twoheadrightarrow (x\,x)[\omega/x] \;=\; \omega\,\omega \\
&\twoheadrightarrow \omega\,\omega \;\twoheadrightarrow\; \cdots
\end{aligned}
$$

One redex, and the step gives Ω back, for ever.

Not every term reaches a normal form: nothing in the definition says what may be applied to what.

3.8 · DROPPING THE TYPES

Fixing an order

Church–Rosser leaves the order free: every order that ends, ends at the same normal form. It does not say that every order ends.

$$(\lambda x.\ y)\,\Omega \ \rightsquigarrow\ y \qquad\qquad (\lambda x.\ y)\,\Omega \ \rightsquigarrow\ (\lambda x.\ y)\,\Omega \ \rightsquigarrow\ \cdots$$

LAZY: CALL BY NAME

Keep β and (app_1); delete the rest. An argument is run only where it is used. Here: y in one step.

EAGER: CALL BY VALUE

Keep (app_1), (app_2); β fires only on an argument in normal form. An argument is run before the call. Here: Ω for ever.

A language picks an order by deleting or restricting rules. Neither runs under a λ, so a run can stop at a value, with redexes still inside: about values Church–Rosser says nothing.

3.7 · PROGRAMS AS PROOFS

The combinators are terms

$$\mathsf{K} \;:=\; \lambda x.\, \lambda y.\; x, \qquad\qquad \mathsf{S} \;:=\; \lambda x.\, \lambda y.\, \lambda z.\; x\, z\, (y\, z)$$

K $M\, N \twoheadrightarrow M$ IS TWO β STEPS

$$\begin{aligned} &(\lambda x.\, \lambda y.\; x)\, M\, N \\ \twoheadrightarrow\; &(\lambda y.\; M)\, N \;\twoheadrightarrow\; M \end{aligned}$$

S $M\, N\, P \twoheadrightarrow M\, P\, (N\, P)$ IS THREE

One for each parameter. So every combinator term is a λ-term, and the two constants need no longer be taken as given.

Theorem 3.44: translation into combinators

THEOREM 3.44 · TRANSLATION INTO COMBINATORS

Every λ-term M translates to a term M° built from K, S, variables and application, with no λ in it, such that M° has the same free variables as M and does the same thing to every argument: if M is $\lambda x.\, P$ then $M^\circ\, N \twoheadrightarrow^* P^\circ[N/x]$ for every N.

Conversely every combinator term is a λ-term.

The λ adds no power. It adds convenience, and Section 3.4 will show it adds a rule.

3.7 · PROGRAMS AS PROOFS

Theorem 3.44: the translation

Translate innermost abstractions first. For a term P containing no λ, define $\langle x\rangle P$, "P abstracted over x", by recursion on P:

$$\langle x\rangle P := \begin{cases} \mathsf{K}\,P & \text{if } x \text{ does not occur in } P, \\ \mathsf{S}\,\mathsf{K}\,\mathsf{K} & \text{if } P \text{ is } x, \\ \mathsf{S}\,(\langle x\rangle P_1)\,(\langle x\rangle P_2) & \text{if } P \text{ is } P_1\,P_2 \text{ and } x \text{ occurs in it.} \end{cases}$$

Then $\langle x\rangle P$ contains no λ and no x, and $(\langle x\rangle P)\,N \rightsquigarrow^* P[N/x]$ for every term N.

Then replace an innermost $\lambda x.\,P$ by $\langle x\rangle P$, and repeat until no λ is left.

3.7 · PROGRAMS AS PROOFS

Theorem 3.44: why the translation works

By induction on P, one case for each line, that $(\langle x\rangle P)\,N \rightsquigarrow^* P[N/x]$:

1. x not in P: $\mathsf{K}\,P\,N \rightsquigarrow P$, and $P[N/x]$ is P.
2. P is x: $\mathsf{S}\,\mathsf{K}\,\mathsf{K}\,N \rightsquigarrow \mathsf{K}\,N\,(\mathsf{K}\,N) \rightsquigarrow N$.
3. P is $P_1\,P_2$: $\mathsf{S}\,(\langle x\rangle P_1)\,(\langle x\rangle P_2)\,N \rightsquigarrow (\langle x\rangle P_1)\,N\,((\langle x\rangle P_2)\,N)$, and by induction each half runs to $P_i[N/x]$, giving $(P_1\,P_2)[N/x]$.

Each replacement preserves the free variables and the behaviour on arguments, so the result is M°.

3.7 · PROGRAMS AS PROOFS

Example: compiling $\lambda x.\ f\,x$

$f\,x$ is an application in which x occurs, so the third line applies; then the first line for f, the second for x. Write I for $\mathsf{S}\,\mathsf{K}\,\mathsf{K}$.

$$\langle x\rangle (f\,x) \;=\; \mathsf{S}\,(\langle x\rangle f)\,(\langle x\rangle x) \;=\; \mathsf{S}\,(\mathsf{K}\,f)\,\mathsf{I}$$

CHECK: APPLY IT TO N

$$\mathsf{S}\,(\mathsf{K}\,f)\,\mathsf{I}\,N \;\rightsquigarrow\; \mathsf{K}\,f\,N\,(\mathsf{I}\,N) \;\rightsquigarrow\; f\,(\mathsf{I}\,N) \;\rightsquigarrow^{*}\; f\,N$$

No variable x anywhere, and it does what $\lambda x.\ f\,x$ does.

3.8 · DROPPING THE TYPES

Theorem 3.47: Turing completeness

THEOREM 3.47 · TURING COMPLETENESS

Every function a Turing machine computes is computed by a closed λ-term, and every function a closed λ-term computes is computed by a Turing machine.

3.8 · DROPPING THE TYPES

History went the other way round

- 1936: Church proposed that "computable" should *mean* "definable by a λ-term", and used it to show that no procedure decides whether two terms have the same normal form.
- That September Turing, who had never heard of the calculus, arrived at Princeton to study with Church. In an appendix of 1937 he proved that his machines and Church's terms compute the same functions.
- Curry and his students showed the combinators equivalent to both.

One theorem, proved by three groups of people about three notations. That's a large part of why the history is confusing.

3.8 · DROPPING THE TYPES

Example 3.48: factorial

Let's send something ordinary the whole way round, so you can believe the three notations are one. Factorial, as you would write it on a whiteboard:

```
fact(n):
   if n = 0 then return 1
   else return n * fact(n - 1)
```

Four things in it are not in the combinator language:

a **parameter**, a name that **calls itself**, a **conditional**, and **arithmetic**.

Take them away one at a time.

3.8 · DROPPING THE TYPES

Example 3.48: the parameter and the conditional

The parameter becomes a λ. Write *isZero*, *mult* and *pred* for the test, the product and the predecessor, and use a boolean as its own conditional: "if b then M else N" is $b\,M\,N$.

$$fact \;:=\; \lambda n.\; isZero\, n\, 1\, (mult\, n\, (fact\, (pred\, n)))$$

Nothing has really happened yet: $fact$ still occurs on the right, and a term may not refer to itself.

3.8 · DROPPING THE TYPES

Example 3.48: the recursion

So make the recursive call a parameter too:

$$F \;:=\; \lambda r.\,\lambda n.\; isZero\, n\; 1\; (mult\, n\, (r\, (pred\, n)))$$

F mentions nothing but its own two parameters: give it the function to call and a number, and it does one layer of the work. We want $fact = F\, fact$, a fixed point of F.

Add a constant *fix* with the single step $fix\, F \rightsquigarrow F\, (fix\, F)$. So define $fact := fix\, F$:

$$fix\, F\, \overline{3} \;\rightsquigarrow\; F\, (fix\, F)\, \overline{3}$$

On the right, F's parameter r is $fix\, F$ again: the recursive call, ready to unfold once more when it is needed and no sooner.

3.8 · DROPPING THE TYPES

Example 3.48: the λs compiled away

One thing remains, the two λs, and the translation $\langle x \rangle P$ removes them: abstract n, then r, by the three lines. With I for $\mathsf{S}\,\mathsf{K}\,\mathsf{K}$:

$$F^{\circ} \;=\; \mathsf{S}\,(\mathsf{K}\,(\mathsf{S}\,A))\,(\mathsf{S}\,(\mathsf{K}\,(\mathsf{S}\,C))\,(\mathsf{S}\,(\mathsf{S}\,(\mathsf{K}\,\mathsf{S})\,(\mathsf{S}\,(\mathsf{K}\,\mathsf{K})\,\mathsf{I}))\,(\mathsf{K}\,E)))$$

$$A = \mathsf{S}\,(\mathsf{S}\,(\mathsf{K}\,\mathit{isZero})\,\mathsf{I})\,(\mathsf{K}\,1), \qquad C = \mathsf{S}\,(\mathsf{K}\,\mathit{mult})\,\mathsf{I}, \qquad E = \mathsf{S}\,(\mathsf{K}\,\mathit{pred})\,\mathsf{I}$$

Not a variable, a parameter, a conditional or a recursive call anywhere in it: only K, S, three arithmetic constants and juxtaposition. Yet $\mathit{fix}\,F^{\circ}\,\overline{4}$ runs to 24.

A program you could have written in your first week sits on two constants in exactly this way, and so does every program you'll ever compile.

SECTION 3.5

The simply typed λ-calculus

What the checker checks.

3.8 · DROPPING THE TYPES

The Monster, in λ notation

The function that applies its argument to itself, applied to itself, runs for ever:

$$\Omega \;=\; (\lambda x.\; x\,x)\,(\lambda x.\; x\,x) \;\rightsquigarrow\; (\lambda x.\; x\,x)\,(\lambda x.\; x\,x) \;\rightsquigarrow\; \cdots$$

CONSTANTS CAME FIRST

Schönfinkel had the combinators in 1924. Curry found them again in the 1920s and built a logic on them (1929). Types: 1934.

THEN λ

Church's logic, with variables bound by λ, followed in 1932. Types: 1940.

Self-application made both logics inconsistent (Kleene and Rosser, 1935). The repair was the same for both: types.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Three rules

Types, contexts and judgements are those of the combinator checker: $\Gamma \vdash M : A$ says M has type A when its variables have the types Γ gives them. Two rules are the checker's own. The third is the Deduction Theorem, adopted as a rule with the program written on.

$$\frac{}{\Gamma \vdash x : A}\ (\text{var})\ \text{if } x : A \text{ is in } \Gamma \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\,N : B}\ (\text{app})$$

$$\frac{\Gamma,\ x : A \vdash M : B}{\Gamma \vdash \lambda x.\, M : A \to B}\ (\text{abs})$$

Read (abs) as the checker reads a function definition: add the parameter to the context, check the body, give the function the arrow type.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Example: Baker's rule, typed

Rule (abs) changes the context, so a typing is now a **tree** whose every node is a whole judgement, context included: natural deduction in **sequent style**. Baker's rule, in two steps to fit the page:

STEP 1

$$\dfrac{\dfrac{}{f : t \to r,\ g : r \to d,\ x : t \vdash f : t \to r}\ \text{var} \qquad \dfrac{}{f : t \to r,\ g : r \to d,\ x : t \vdash x : t}\ \text{var}}{f : t \to r,\ g : r \to d,\ x : t \vdash f\,x : r}\ \text{app}$$

STEP 2, USING STEP 1

$$\dfrac{\dfrac{\dfrac{}{f : t \to r,\ g : r \to d,\ x : t \vdash g : r \to d}\ \text{var} \qquad (1)\ f : t \to r,\ g : r \to d,\ x : t \vdash f\,x : r}{f : t \to r,\ g : r \to d,\ x : t \vdash g\,(f\,x) : d}\ \text{app}}{f : t \to r,\ g : r \to d \vdash \lambda x.\, g\,(f\,x) : t \to d}\ \text{abs}$$

Rub out the programs and it is Baker's tree. The (abs) bar deletes $x : t$ from the context, where the tree put a bracket round t.

Function composition, the body

The typing of a program you've written in one form or another, `\f -> \g -> \x -> f (g x)`. The body is checked in the context $f : B \to C,\ g : A \to B,\ x : A$:

STEP 1

$$\dfrac{\dfrac{}{f : B \to C,\ g : A \to B,\ x : A \vdash g : A \to B}\ \text{var} \qquad \dfrac{}{f : B \to C,\ g : A \to B,\ x : A \vdash x : A}\ \text{var}}{f : B \to C,\ g : A \to B,\ x : A \vdash g\,x : B}\ \text{app}$$

STEP 2, USING STEP 1

$$\dfrac{\dfrac{}{f : B \to C,\ g : A \to B,\ x : A \vdash f : B \to C}\ \text{var} \qquad (1)\ f : B \to C,\ g : A \to B,\ x : A \vdash g\,x : B}{f : B \to C,\ g : A \to B,\ x : A \vdash f\,(g\,x) : C}\ \text{app}$$

Three look-ups and two uses of (app), each matching an argument's type against the type a function expects.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Function composition, the three abstractions

Three uses of (abs) finish it, each taking the last variable out of the context and putting a λ on the front of the program.

$$\dfrac{\dfrac{\dfrac{f : B \to C,\ g : A \to B,\ x : A \vdash f\,(g\,x) : C}{f : B \to C,\ g : A \to B \vdash \lambda x.\, f\,(g\,x) : A \to C}\ \text{abs}}{f : B \to C \vdash \lambda g.\, \lambda x.\, f\,(g\,x) : (A \to B) \to A \to C}\ \text{abs}}{\vdash \lambda f.\, \lambda g.\, \lambda x.\, f\,(g\,x) : (B \to C) \to (A \to B) \to A \to C}\ \text{abs}$$

The context shrinks one declaration at a time, until at the root it is empty and the program is closed.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Function composition, rubbed out

- The top is a proof of C from $B \to C$, $A \to B$ and A, by two uses of modus ponens.
- The three (abs) bars discharge A, then $A \to B$, then $B \to C$.
- The first discharge alone gives $A \to C$ from $B \to C$ and $A \to B$: hypothetical syllogism, the rule that chains two implications.

$$\vdash (B \to C) \to (A \to B) \to A \to C$$

Hypothetical syllogism as a theorem, with nothing assumed. The compiler read your composition function as that proof.

3.3 · FOLDING UP PROOFS

Why pairs and unions?

CASE: TWENTY PAST TEN, AS A PROGRAM

Baker's evidence for $a \wedge p$: an authorisation code and a capture id, held together. A *pair*; to use one, take it back out.

CASE: HALF PAST TEN, AS A PROGRAM

"Declined or approved": a value with a tag on it, *Left* or *Right*. To use it, branch on the tag.

TYPE	IN THE LANGUAGES YOU USE
pairs	tuples such as `(min, max)`; records and structs
tagged unions	enums; Haskell's `Either`; Rust's `Result<T, E>`: a value or an error, and you must handle both
the empty type	Haskell's `Void`, Rust's `!`: a computation that never returns

3.3 · FOLDING UP PROOFS

The other connectives

The rules for $\wedge$, $\vee$ and $\bot$ get programs in the same way, and they are programs you know: a **pair**, a **tagged union**, the **empty type**.

INTRODUCTION

builds a value of the type: a constructor.

ELIMINATION

takes a value apart: a pattern match.

One thing changes on the page. While we're typing, we follow the type theorists: the pair type is $A \times B$, the union $A + B$. The empty type $\bot$ and $\to$ are shared.

3.3 · FOLDING UP PROOFS

Using a tagged union

A value M of a tagged union is *Left* or *Right* of something, and code that uses it can't know which in advance, so it has a branch for each:

```
case M:
   Left x  => P
   Right y  => Q
```

If M is *Left* of some value, call that value x and run P; if it is *Right* of some value, call it y and run Q.

Like a function's parameter, x and y name whatever M turns out to carry.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Typing rules: pairs and the empty type

$A \times B$ is the pair type, `(A, B)`; $\bot$ is the type with no constructor, `Void`, of which there are no values; $\neg A$ is $A \to \bot$.

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A \times B}\ (\times\mathrm{I}) \qquad \frac{\Gamma \vdash M : \bot}{\Gamma \vdash absurd\, M : C}\ (\bot\mathrm{E})$$

$$\frac{\Gamma \vdash M : A \times B}{\Gamma \vdash fst\, M : A}\ (\times\mathrm{E}_1) \qquad \frac{\Gamma \vdash M : A \times B}{\Gamma \vdash snd\, M : B}\ (\times\mathrm{E}_2)$$

To build an $A \times B$ you build both halves; to use one you take whichever half you need. Nothing builds a $\bot$; if you ever have one, conclude anything: `absurd :: Void -> a`.

Typing rules: sums

$A + B$ is the sum type, `Either A B`.

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathit{Left}\, M : A + B}\ (+\mathrm{I}_1) \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \mathit{Right}\, M : A + B}\ (+\mathrm{I}_2)$$

$$\frac{\Gamma \vdash M : A + B \qquad \Gamma, x : A \vdash P : C \qquad \Gamma, y : B \vdash Q : C}{\Gamma \vdash \mathbf{case}\ M\ \mathbf{of}\ \{\, \mathit{Left}\, x \Rightarrow P \mid \mathit{Right}\, y \Rightarrow Q \,\} \ :\ C}\ (+\mathrm{E})$$

To build an $A + B$ you build one side and say which; to use one you argue by cases.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Typing rules: reading (+E)

On the page the pseudocode goes on one line, its two branches between braces:

$$\textbf{case}\ M\ \textbf{of}\ \{\, \mathit{Left}\, x \Rightarrow P \mid \mathit{Right}\, y \Rightarrow Q \,\}\ :\ C$$

- $M : A + B$: a tagged value to look at.
- $x : A$ in P only, $y : B$ in Q only: each branch has its own variable.
- $P : C$ and $Q : C$: only one runs, and what follows can't know which.

3.3 · FOLDING UP PROOFS

The central table

CONNECTIVE	TYPE	INTRODUCTION (CONSTRUCTOR)	ELIMINATION (PATTERN MATCH)
$A \to B$	$A \to B$, `A -> B`	$\lambda x.\, M$	application $M\, N$
$A \wedge B$	$A \times B$, `(A, B)`	(M, N)	fst, snd
$A \vee B$	$A + B$, `Either A B`	$Left\, M$, $Right\, M$	`case` with two branches
$\bot$	$\bot$, `Void`	—	$absurd$
$\neg A$	$A \to \bot$, `A -> Void`	$\lambda x.\, M$	application

The connectives of logic are the data types of a programming language; the introduction rules are their constructors, and the elimination rules are the pattern matches that take them apart.

3.7 · PROGRAMS AS PROOFS

Example 3.40: swapping a pair

$A \wedge B \to B \wedge A$ is a theorem, and here is the program that proves it: $\lambda p.\ (snd\, p,\ fst\, p) : A \times B \to B \times A$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p : A \times B \vdash p : A \times B}\,\text{var}}{p : A \times B \vdash snd\, p : B}\,\times\text{E}_2 \qquad \dfrac{\dfrac{}{p : A \times B \vdash p : A \times B}\,\text{var}}{p : A \times B \vdash fst\, p : A}\,\times\text{E}_1}{p : A \times B \vdash (snd\, p,\ fst\, p) : B \times A}\,\times\text{I}}{\vdash \lambda p.\ (snd\, p,\ fst\, p) : A \times B \to B \times A}\,\text{abs}$$

The tree we drew when learning to discharge. The two bracketed leaves are the two occurrences of p in the body.

Example 3.41: contraposition

$(A \to B) \to (\neg B \to \neg A)$: the program is $\lambda f.\, \lambda k.\, \lambda a.\; k\,(f\,a)$. Its body, typed in the context $f : A \to B,\; k : B \to \bot,\; a : A$:

STEP 1

$$\dfrac{\dfrac{}{f : A \to B,\; k : B \to \bot,\; a : A \vdash f : A \to B}\,\text{var} \qquad \dfrac{}{f : A \to B,\; k : B \to \bot,\; a : A \vdash a : A}\,\text{var}}{f : A \to B,\; k : B \to \bot,\; a : A \vdash f\,a : B}\,\text{app}$$

STEP 2, USING STEP 1

$$\dfrac{\dfrac{}{f : A \to B,\; k : B \to \bot,\; a : A \vdash k : B \to \bot}\,\text{var} \qquad (1)\; f : A \to B,\; k : B \to \bot,\; a : A \vdash f\,a : B}{f : A \to B,\; k : B \to \bot,\; a : A \vdash k\,(f\,a) : \bot}\,\text{app}$$

Three uses of (abs) give $(A \to B) \to (B \to \bot) \to A \to \bot$. Baker's modus tollens is this function applied to $b \to g$ and the ledger's $\neg g$.

Example 3.42: swapping a union

$A \vee B \to B \vee A$:

$$\lambda s.\ \mathbf{case}\ s\ \mathbf{of}\ \{\, \mathit{Left}\, a \Rightarrow \mathit{Right}\, a \mid \mathit{Right}\, b \Rightarrow \mathit{Left}\, b \,\}$$

$$: A + B \to B + A$$

The two branch variables a and b are the two leaves $[p]^2$ and $[q]^3$ of the tree we drew for this formula, with A, B for its p, q, each in scope only in its own branch.

In each of the three examples the program is one you could have written without knowing what it proved: a function that swaps a pair; one that composes a test with a function; one that swaps the tags on a union.

The types were there all along.

3.3 · FOLDING UP PROOFS

Theorem 3.15: Curry–Howard

THEOREM 3.15 · CURRY–HOWARD

(i) (*Erasure.*) Take a typing of $\Gamma \vdash M : A$ and delete every term, keeping the names on the hypotheses. Then (var) becomes a hypothesis, (app) an $\to$E bar, (abs) an $\to$I bar, and so on. The result is a derivation of A with open hypotheses among $|\Gamma|$, the types declared in Γ.

(ii) (*Decoration.*) Take a derivation of A with a name on each hypothesis, and write terms from the leaves down: a variable at each leaf, an application at each $\to$E, a λ at each $\to$I, a constructor or destructor at each other bar. The result is a typing of $\Gamma \vdash M : A$, with Γ declaring the open hypotheses.

Folding is a one-to-one correspondence: each undoes the other. In one line: $\Gamma \vdash M : A$ for some term M if and only if $|\Gamma| \vdash A$.

3.3 · FOLDING UP PROOFS

Theorem 3.15: why

Put the typing rules beside the clauses that define derivations. Rule for clause, each derivation clause is the typing rule with the terms removed: premisses, conclusions, and what each takes out of the context or out of OH are otherwise identical.

$$\mathrm{FV}(\lambda x.\, M) = \mathrm{FV}(M) \setminus \{x\} \quad \text{(a } \lambda \text{ binds its variable)}$$

$$\mathrm{OH} = \mathrm{OH}(\mathcal{D}) \setminus \{A\} \quad \text{(an } \to\mathrm{I} \text{ bar discharges its formula)}$$

3.3 · FOLDING UP PROOFS

What the theorem says

"Proofs resemble programs" says far less than the theorem does. It says that a derivation in Gentzen's system and a typing of a program with parameters, pairs, unions and `case` are the **same tree with a column added**; and that the rule which puts a parameter in scope and takes it out again is, column removed, the rule for supposing and discharging.

TWO NOTATIONS

A type theorist writes $A \times B$ and $A + B$: a pair type has as many values as the product, a union as many as the sum. A logician writes $\wedge$ and $\vee$, because those are the words.

TWO NAMES

Curry saw the Hilbert half in 1934. The half with λ and natural deduction is Howard's, from 1969. That's why the observation carries both names.

3.7 · PROGRAMS AS PROOFS

At work: programming languages

- Milner's ML (1978) is the typed λ-calculus with the types worked out by the compiler, and with the promise "well-typed programs cannot go wrong". OCaml, F# and Haskell descend from it.
- Every closure or lambda in Java, C#, Swift, Rust or TypeScript is typed by rule (abs): add the parameter to the context, check the body, give it the arrow type.
- The Glasgow Haskell compiler translates Haskell into a small typed λ-calculus, its "Core", and can type-check that again after each optimisation, so that a compiler bug shows up as a type error.

Every compiler for these languages contains the three rules of this section.

3.7 · PROGRAMS AS PROOFS

At work: checking proofs

- Church's typed calculus of 1940 is the logic of the HOL family of proof assistants: HOL4, HOL Light, Isabelle/HOL.
- In Coq (now Rocq), Lean and Agda a proof *is* a typed λ-term, in a richer calculus than this one, and a small kernel checks it with rules like (var), (app) and (abs).
- What they have checked: the seL4 operating-system kernel (Isabelle/HOL, 2009); the CompCert C compiler (Coq); the Four Colour Theorem (Coq, 2005); the Kepler conjecture (HOL Light and Isabelle, 2014).

The trusted part is a type checker. Whoever wants the theorem does the hard part, finding the term.

SECTION 3.7

Programs as proofs

Every typed program proves its type.

3.4 · REMOVING DETOURS

What Gentzen was hired for

Hilbert hired Gentzen for something other than making proofs look like reasoning: to prove, by arguments so elementary that nobody could doubt them, that mathematics could never derive a contradiction.

The arguments allowed were about the **shapes of written symbols**: no appeal to infinite collections, none to what the symbols were about. Natural deduction was built to make such arguments possible.

> "Perhaps we may express the essential properties of such a normal proof by saying: it is not roundabout. No concepts enter into the proof other than those contained in its final result."
>
> Gentzen, 1934, on the *Hauptsatz*

3.4 · REMOVING DETOURS

Detours

A detour, in Gentzen's word, is an introduction met at once by its own elimination: prove $A \to B$ by $\to$I, then at once use it on a proof $\mathcal{E}$ of A. The proof is shorter without the pair:

$$\dfrac{\dfrac{\begin{array}{c}[A]\\ \mathcal{D}\\ B\end{array}}{A \to B}\,{\to}\mathrm{I} \qquad \begin{array}{c}\mathcal{E}\\ A\end{array}}{B}\,{\to}\mathrm{E} \qquad \text{becomes} \qquad \begin{array}{c}\mathcal{E}\\ A\\ \mathcal{D}\\ B\end{array}$$

A copy of $\mathcal{E}$ goes in place of every leaf $[A]$ of $\mathcal{D}$, and the two bars go.

3.4 · REMOVING DETOURS

Baker's detour

At ten past ten Baker proved the rule $t \to d$ by supposing t and discharging. At twenty to eleven, say, the gateway's log gives her t, and she applies the rule to it.

The reply she writes on the ticket does not mention the rule. It substitutes the evidence for t straight into the reasoning that followed the supposition, and contains only facts about this charge.

"No concepts other than those contained in its final result." Look hard at that substitution, because you've seen it before.

3.4 · REMOVING DETOURS

Why they wanted normal proofs

- Hilbert's programme asked for a proof, by arguments about the shapes of symbols, that mathematics can never derive a contradiction.
- A normal proof is not roundabout: for $\to$ and $\wedge$, every formula in it is a piece of its conclusion or of one of its open hypotheses.
- So ask what a normal proof of $\bot$ from no hypotheses could look like. Its formulas would all be pieces of $\bot$, and $\bot$ has no introduction rule and no pieces. There is nothing to build it from.
- So **if every proof can be made normal, nothing proves** $\bot$.

Normalisation gives consistency, and consistency was the job Gentzen was hired to do.

3.4 · REMOVING DETOURS

Hard to prove

- Gentzen could prove that every derivation normalises for intuitionistic natural deduction, NJ, but not for the classical version.
- To cover classical logic he invented a third presentation of the rules, the sequent calculus, and proved his *Hauptsatz* there. It covered the intuitionistic case too, so he left the natural-deduction proof out of his thesis.
- Prawitz published the direct proof for natural deduction in 1965. In 2005 von Plato found Gentzen's own, in a handwritten version of the thesis among Bernays's papers.

We take normalisation on trust. What the program view gives us is the road from it to consistency: look at the shape of a program that has stopped.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Definitions 3.19 and 3.26: β-reduction, for the whole language

A step rewrites an introduction met by its elimination, anywhere inside a term. One rule for each introduction form met by its elimination:

$$\frac{}{(\lambda x.\, P)\, Q \;\rightsquigarrow\; P[Q/x]}\;\beta \qquad \frac{}{fst\,(P, Q) \;\rightsquigarrow\; P}\;\text{fst} \qquad \frac{}{snd\,(P, Q) \;\rightsquigarrow\; Q}\;\text{snd}$$

$$\frac{}{\textbf{case}\ (Left\, R)\ \textbf{of}\ \{\, Left\, x \Rightarrow P \mid Right\, y \Rightarrow Q \,\} \;\rightsquigarrow\; P[R/x]}\;Left$$

$$\frac{}{\textbf{case}\ (Right\, R)\ \textbf{of}\ \{\, Left\, x \Rightarrow P \mid Right\, y \Rightarrow Q \,\} \;\rightsquigarrow\; Q[R/y]}\;Right$$

and a rule for each place a subterm can sit, as (app_1), (app_2), (abs). None for *absurd*: $\bot$ has no introduction to meet. No step possible: normal form; by Curry–Howard, no detour: normal derivation.

3.5 · THE SIMPLY TYPED λ-CALCULUS

Example: running a detour

The steps your interpreter takes: call a function, project a pair, branch on a tag. Apply the swap of Example 3.40 to a pair (a, b). As a proof: $A \wedge B \to B \wedge A$, proved by discharging, used at once. A detour.

$$
\begin{aligned}
& (\lambda p.\ (snd\, p,\ fst\, p))\,(a, b) && \\
\rightsquigarrow\ & (snd\,(a, b),\ fst\,(a, b)) && \beta \\
\rightsquigarrow\ & (b,\ fst\,(a, b)) && \text{projection} \\
\rightsquigarrow\ & (b,\ a) && \text{projection}
\end{aligned}
$$

The normal form is the direct proof: from b and a, one $\wedge I$. No implication anywhere, no concept not in the result.

Theorem 3.29: subject reduction

THEOREM 3.29 · SUBJECT REDUCTION

If $\Gamma \vdash M : A$ and $M \twoheadrightarrow N$, then $\Gamma \vdash N : A$.

Proof sketch. One new ingredient, a substitution lemma: if $\Gamma, x : A \vdash M : B$ and $\Gamma \vdash N : A$, then $\Gamma \vdash M[N/x] : B$. Go down the typing of M and replace every (var) line for x by the typing of N. On the proof side it is transitivity.

Then one case per step. For β: $(\lambda x.\, M)\, N : B$ came by (app) from $\lambda x.\, M : A \to B$ and $N : A$; the first came by (abs) from $\Gamma, x : A \vdash M : B$; the lemma gives $M[N/x] : B$. □

3.5 · THE SIMPLY TYPED λ-CALCULUS

Definition 3.22: normal and neutral terms

A typed term is neutral if it is a variable, or an elimination applied to a neutral term:

$$N\,M, \qquad \mathit{fst}\,N, \qquad \mathit{snd}\,N, \qquad \textbf{case}\ N\ \textbf{of}\ \{\ldots\}, \qquad \mathit{absurd}\,N$$

with N neutral and the other parts in normal form.

A term is in normal form exactly when it is an introduction form, $\lambda x.\,M$, (M, N), $\mathit{Left}\,M$ or $\mathit{Right}\,M$, whose parts are in normal form, or is neutral.

EXAMPLE

With $f : A \to B \to C$, $p : A \times B$: $f\,(\mathit{fst}\,p)\,(\mathit{snd}\,p)$ is neutral. Put $\lambda p.$ in front: an introduction form.

A neutral term is a chain of eliminations hanging off a variable, and that variable is **free** in it: nothing in the chain binds it.

Proposition 3.28: the types do the work

- Take a normal-form application $M\,N$. Its M is in normal form and has an arrow type, and it is not a λ, or $M\,N$ would be a redex.
- By induction M is an introduction form or neutral. The only introduction form with an arrow type is a λ. So M is neutral, and then so is $M\,N$.
- The same argument handles a projection, whose argument cannot be a pair, and a case, whose argument cannot be a tag.
- The argument of an *absurd* has type $\bot$, which has no introduction form at all, so it too must be neutral.

Without types, Ω is normal in none of these senses and stops in none of them.

SECTION 3.6

Normalisation

Does every proof straighten out?

3.6 · NORMALISATION

Theorem 3.33: normalisation

THEOREM 3.33 · NORMALISATION

Every typed term runs to a normal form. Equivalently: every derivation can be rewritten, by removing detours, into a normal derivation of the same formula from the same open hypotheses.

The same theorem, proved twice by different methods: Prawitz for derivations in 1965, by counting redexes; Tait for programs in 1967, by "computability".

3.6 · NORMALISATION

Lemma 3.37: shape of closed normal forms

LEMMA 3.37 · SHAPE OF CLOSED NORMAL FORMS

A closed typed term in normal form is an introduction form. Consequently there is no closed term in normal form whose type is a letter p, and none whose type is $\bot$.

Proof. A normal-form term is an introduction form or neutral, and a neutral term has a free variable, its head; so a closed one is an introduction form. An introduction form has type $A \to B$, $A \times B$ or $A + B$, by the rule that introduces it, and none of those is a letter or $\bot$. □

3.6 · NORMALISATION

Theorem 3.38: consistency

THEOREM 3.38 · CONSISTENCY

There is no closed term of type $\bot$: $\nvdash \bot$. More generally, for every letter p, $\nvdash p$.

Proof. By Curry–Howard, it is enough that no closed term has type $\bot$. Suppose $\vdash M : \bot$.

1. By normalisation, M runs to a normal form N.
2. By subject reduction, $\vdash N : \bot$.
3. N is closed, since no step of running creates a free variable.
4. So N is a closed normal-form term of type $\bot$, and the shape lemma says there is none.

The argument for a letter p is the same with p for $\bot$. □

3.6 · NORMALISATION

Corollary 3.39: disjunction property

COROLLARY 3.39 · DISJUNCTION PROPERTY

If $\vdash A \vee B$, then $\vdash A$ or $\vdash B$.

Proof. By Curry–Howard there is a closed term of type $A + B$. By normalisation and subject reduction it runs to a closed normal form of that type, which by the shape lemma is an introduction form: $\mathit{Left}\, M$ with $\vdash M : A$, or $\mathit{Right}\, M$ with $\vdash M : B$. □

A proof of a disjunction is a value with a tag on it: you can't establish "A or B" without saying which.

3.8 · DROPPING THE TYPES

Remark 3.49: general recursion

Every language you use has recursion. Model it by a constant $\mathit{fix} : (A \to A) \to A$, for every A, with $\mathit{fix}\, f \rightsquigarrow f\,(\mathit{fix}\, f)$. Our factorial needed it.

$$\begin{array}{c} \mathit{fix}\,(\lambda x.\ x) \ :\ \bot \qquad\qquad \mathit{fix}\,(\lambda x.\ x) \ \rightsquigarrow \\ (\lambda x.\ x)\,(\mathit{fix}\,(\lambda x.\ x)) \ \rightsquigarrow\ \mathit{fix}\,(\lambda x.\ x) \ \rightsquigarrow\ \cdots \end{array}$$

It has every type there is, $\bot$ and every letter included, so consistency fails; and it runs for ever, so normalisation fails with it. The two theorems go together, as they came together.

The price of factorial is the consistency of the logic. Every language you use pays it, knowingly: the loop is worth more to a programmer than consistency is.

SECTION 3.9 (OPTIONAL)

Fitch proofs

Whose boxes have you been drawing?

3.9 · FITCH PROOFS (OPTIONAL)

Jaśkowski's boxes

Jaśkowski's system of 1934 wrote a proof not as a tree but down the page, one line after another as a Hilbert proof does, with one addition: a supposition opens a **box**, every line inside is reasoned under it, and the box closes at the line that discharges it, which stands outside.

Fitch made the notation standard in 1952, drawing the box as a vertical bar with the supposition underlined at its top. A proof drawn this way is a Fitch proof.

The boxes in which the combinator typings were written are Fitch proofs with the programs on: each a single box, because nothing there was ever discharged.

3.9 · FITCH PROOFS (OPTIONAL)

Two inventions that never met

- Gentzen was in Göttingen and published in the *Mathematische Zeitschrift*. Jaśkowski was in Warsaw, among Łukasiewicz's students, and published in English in the first volume of *Studia Logica*.
- Different countries, different journals, different circles; neither ever heard from the other.
- The war scattered the Warsaw school. Jaśkowski's paper reached the wider world only afterwards, through textbooks: Quine's *Methods of Logic* in 1950, then Fitch's book of 1952, whose preface credits both.
- By then Gentzen had been dead for seven years.

Nobody chose between the tree and the box. Two people answered the same question separately, and Theorem 3.52 says their answers agree.

3.9 · FITCH PROOFS (OPTIONAL)

Definition 3.50: Fitch proof, the boxes

A box has a head above a horizontal line and, below it, a finite sequence of entries: each either a formula, a line, or another box, a subbox.

The head of the outermost box is a set of formulas. The head of a subbox is a single formula, its supposition, and its last entry is a line.

Lines are numbered in one sequence down the page, whichever box they stand in; a subbox is cited by the numbers i–j of its first and last lines.

IN SCOPE

A line or subbox is in scope at line k if it comes before line k and is an entry of a box that contains line k.

So a subbox is in scope after it has closed, and the lines inside it are not.

3.9 · FITCH PROOFS (OPTIONAL)

Definition 3.50: Fitch proof, the lines

A Fitch proof of φ from Γ is a box with head Γ whose last entry is the line φ, in which every line φ_k is justified in one of these ways, "earlier" meaning "in scope at line k":

1. (*hyp*) φ_k is in Γ, or is the supposition of a subbox that contains line k;
2. ($\to$E) there are earlier lines φ_i and φ_j such that φ_i is $\varphi_j \to \varphi_k$;
3. ($\to$I) φ_k is $\psi \to \chi$, and there is an earlier subbox with supposition ψ whose last line is χ;
4. ($\wedge$I) φ_k is $\psi \wedge \chi$, and ψ and χ are earlier lines;
5. ($\wedge\mathrm{E}_1$, $\wedge\mathrm{E}_2$) some earlier line is $\varphi_k \wedge \chi$, or $\psi \wedge \varphi_k$;
6. ($\vee\mathrm{I}_1$, $\vee\mathrm{I}_2$) φ_k is $\psi \vee \chi$, and ψ, or χ, is an earlier line;
7. ($\vee$E) some earlier line is $\psi \vee \chi$, and there are two earlier subboxes, with suppositions ψ and χ, each with last line φ_k;
8. ($\bot$E) $\bot$ is an earlier line.

No other kind of line is allowed.

3.9 · FITCH PROOFS (OPTIONAL)

Beside a Hilbert proof

- The outer box is a Hilbert proof's box: hypotheses above the line, lines below it.
- (hyp) is the hypothesis clause, with the supposition of every box around a line counted among its hypotheses. ($\to$E) is modus ponens, word for word.
- The axiom clause has gone: the rules that open subboxes do its work.
- The one restriction comes with the subbox. Once it has closed, nothing outside may cite a line inside it; only the subbox as a whole may be cited, and only by ($\to$I) and ($\vee$E), the two rules that discharge.

Read a line φ with context Δ as the judgement $\Delta \vdash \varphi$, and each clause is a typing rule with the terms deleted: (hyp) is (var), ($\to$E) is (app), ($\to$I) is (abs).

3.9 · FITCH PROOFS (OPTIONAL)

Example 3.51: Baker's rule, as a box

Drawn earlier as a tree; now down the page.

LINE	FORMULA	JUSTIFICATION
	$t \to r,\ r \to d$	
1.	$\underline{t}$	(hyp)
2.	$t \to r$	(hyp)
3.	r	($\to$E) 2, 1
4.	$r \to d$	(hyp)
5.	d	($\to$E) 4, 3
6.	$t \to d$	($\to$I) 1–5

Her six lines of the opening slide. Line 6 stands outside the subbox and cites it as a whole.

3.9 · FITCH PROOFS (OPTIONAL)

Theorem 3.52: trees and boxes

THEOREM 3.52 · TREES AND BOXES

For every set of formulas Γ and every formula φ, the following are equivalent:

(i) $\Gamma \vdash_{\mathrm{N}} \varphi$: some tree derivation has conclusion φ and all its open hypotheses in Γ;
(ii) φ has a Fitch proof from Γ.

Moreover, the translations both ways go step by step: each bar of a tree becomes a line justified by the same rule, and each line of a Fitch proof not justified by (hyp) becomes a bar of the same rule.

Gentzen's trees and Jaśkowski's boxes are two notations for one system.

Theorem 3.52: why

TREE TO BOX: RULE INDUCTION

Show φ has a Fitch proof from every $\Gamma \supseteq \mathrm{OH}(\mathcal{D})$.

- One node: the single line φ, by (hyp).
- Other rules without discharge: copy the premisses' Fitch proofs, add the conclusion.
- ($\to$I): the premiss has a Fitch proof from $\Gamma \cup \{\varphi\}$; copy it into a subbox with supposition φ, then add $\varphi \to \psi$.
- ($\vee$E): the same, one subbox per branch.

BOX TO TREE: DOWN THE PAGE

Write Δ_k for the context of line k. Build for each line a tree concluding φ_k with open hypotheses inside Δ_k.

- (hyp): the one-node tree φ_k.
- Any other line: put the trees already built for the cited lines above a bar for the matching rule.
- ($\to$I) and ($\vee$E) remove the suppositions, so the open hypotheses stay inside Δ_k.

One line for each bar, or one bar for each line, under the same rule.

3.9 · FITCH PROOFS (OPTIONAL)

Three notations for one object

Programs can be written in the boxes too, and then (abs) is a box inside a box: open an inner box declaring $x : A$, type the body inside it, close it. The line $\lambda x.\, M : A \to B$ is written *outside*, because outside the box x is not in scope.

That's the scope rule of a programming language: a parameter is in scope inside the function and not outside it.

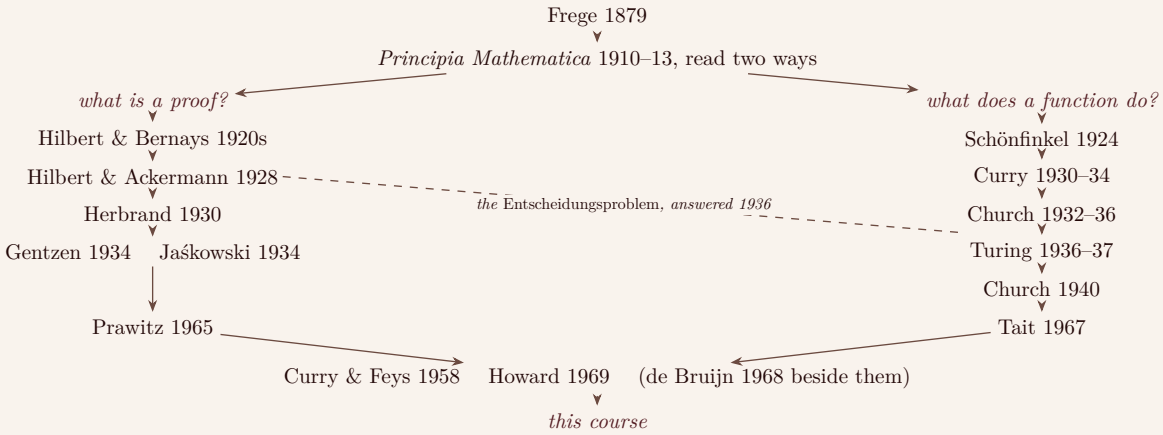
FINALLY

History

Twenty people, one diamond.

HISTORY

The diamond



HISTORY

One book, read two ways

THE LEFT EDGE: WHAT IS A PROOF?

Hilbert read *Principia* and saw that a proof could be written down as a list and checked: so proof theory, the Deduction Theorem, and Gentzen's rebuilding of the list as a tree.

THE RIGHT EDGE: WHAT DOES A FUNCTION DO?

Schönfinkel, Curry and Church saw mechanical instructions for rewriting formulas, above all for handling variables: so the combinators, the λ, and, once Turing had asked in his own way, the theory of computation.

The edges first touch in 1936, when Church and Turing showed there is no procedure deciding provability. They meet at the bottom, in Curry and Howard.

HISTORY

Four pairs of names

CURRY AND CHURCH

Two calculi for one doomed logic, one with constants, one with λ. One paradox broke both; each repaired his with types.

CHURCH AND TURING

Two definitions of “computable” that are one theorem. Their thesis claims the theorem is about everything.

GENTZEN AND JAŚKOWSKI

Invented “suppose” in the same year, unknown to each other: a tree and a box.

CURRY AND HOWARD

Two halves of one correspondence, thirty-five years apart.

HISTORY

Who did what, 1879–1930

WHO	WHEN	WHAT
Frege	1879	proofs as gap-free chains; axioms (I), (II)
Whitehead & Russell	1910–13	the whole of mathematics, by rule
Hilbert & Bernays	1922–	proofs as objects; proof theory
Schönfinkel	1924	K and S suffice
Hilbert & Ackermann	1928	the textbook; the *Entscheidungsproblem*
Herbrand	1930	the Deduction Theorem
Curry	1930–34	combinatory logic; the types of K, S

HISTORY

Who did what, 1932–1969

WHO	WHEN	WHAT
Church	1932–36	the λ-calculus; an unsolvable problem
Gentzen, Jaśkowski	1934	natural deduction; the *Hauptsatz*
Turing	1936–37	the machine; equivalence with λ
Church	1940	the simply typed λ-calculus
Curry & Feys	1958	types of combinators are axioms, in print
Prawitz; Tait	1965; 1967	normalisation, both sides
Howard	1969	the correspondence for λ and NJ

SUMMARY

Chapter 3

- **Natural deduction** is the Hilbert system with every axiom read backwards through the Deduction Theorem: nine rules and no axioms. Its derivations are trees, and a bracket round a leaf is a receipt for a supposition paid off. NJ proves exactly what the Hilbert system proves (Theorem 3.14).
- Decorated with programs, a derivation is a typing in the **simply typed λ-calculus** (Theorem 3.15): a hypothesis is a variable, $\to$I is λ, $\to$E is application, and the other connectives are your data types.
- The untyped calculus compiles to K and S (Theorem 3.44) and computes whatever a Turing machine computes (Theorem 3.47).
- **Running a program is removing detours** from a proof. A stopped closed program is a constructor, so the logic proves no letter and no $\bot$ (Theorem 3.38), and a proof of *or* always says which (Corollary 3.39).

LOOKING AHEAD

Chapter 4

Safe is not correct. Nothing we care about, a sorted list, a transitive order, a discount on every day of the sale, is proved from *no* hypotheses. By the consistency theorem, nothing with a bare claim at the top ever is.

Such things are proved from a specification: sentences about lists, orders or sales, taken as hypotheses.

And a specification says *for all*: every payment, not the one on Baker's ticket; every day of the sale. No formula of this chapter can say that.

CHAPTER 4

For all, and proving from a specification.

READING

Sources

- Gentzen (1934), *Untersuchungen über das logische Schließen.* The original; its introduction, in Szabo's translation, is four pages anyone can read.
- Girard, Lafont and Taylor (1989), *Proofs and Types*, chapters 2–4. Natural deduction, proofs as programs, detours, and normalisation proved in full for $\to$ and $\wedge$. Short, free online, and the right next thing to read.
- Prawitz (1965); Howard (1980); Wadler (2015). Normal proofs as a subject; the correspondence as Howard wrote it in 1969; the story told to programmers.
- Menzler-Trott (2007); von Plato (2008). Gentzen's life; the manuscript in Zürich.

Thank you.