CHAPTER 1

Whereas denotation was modelled at a very early stage, sense has been pushed towards subjectivism, with the result that the present mathematical treatment of sense is more or less reduced to syntactic manipulation. This is not a priori in the essence of the subject, and we can expect in the next decades to find a treatment of computation that would combine the advantages of denotational semantics (mathematical clarity) with those of syntax (finite dynamics). This book clearly rests on a tradition that is based on this unfortunate current state of affairs: in the dichotomy between infinite, static denotation and finite, dynamic sense, the denotational side is much more developed than the other.

page 2

So he sees the manipulation of sense as a finite comp process.
And sense now is reduced to syntax.

Sense Denotation & Semantics:

$27 \times 37 = 999$

The two sides have the same referent but different senses. See Frege.

Emphasising the dichotomy

• Sense		• Denotation
• Syntax	vs	• Truth
• proof		• Semantics
		• Algebraic operations

If the sense is a facts manner of presentation we can see why it is related to syntax (valid ways of presenting) & proof (manipulations from one presentation to another).

Denotational / Algebraic Tradition:
- Disregard all sense
- Leads to model theory

why?
My guess is that model theory allows you to take several different "Senses" (models) and as long as they satisfy properties you can just go ahead & pretend its the same.
→ Same type of denotation.

Syntactic Tradition:
The tangible aspect of sense is the syntax. How something is written on the page.

Girard claims the symetries of the senses of logic are obscured by this limitation.

"We are always in search of an operational distinction between sense & syntax".

the famous theorem of Gentzen of 1934 shows that logic possesses some profound symmetries at the syntactical level (expressed by cut-elimination) ?

Semantic Traditions:

Tarski: Only interested in the denotation of a sentence as true or false (1, 0).

Once again, this definition is ludicrous from the point of view of logic, but entirely adequate for its purpose. The development of Model Theory shows this. ?
pg 5

Heyting: Model the proofs not just the denotation. Asks not when is A true but what is a proof of A.

Girard views proofs not as a transcript (syntax) but as an "inherent object", something already a process.

The Heyting semantics interprets logical operations by their constructions.

Daniells interpretation:
He's talking about how despite syntactic differences two proofs can be "essentially" the same

Tarski semantics

Nor has it foundational prospects, since for foundations, one has to give an explanation in terms of something more primitive, which moreover itself needs its own foundation. The tradition of Heyting is original, but fundamentally has the same problems — Gödel's incompleteness theorem assures us, by the way, that it could not be otherwise. If we wish to explain A by the act of proving A, we come up against the fact that the definition of a proof uses quantifiers twice (for ⇒ and ∀). Moreover in the ⇒ case, one cannot say that the domain of definition of ⇒ is particularly well understood!
Since the ⇒ and ∀ cases were problematic (from this absurd foundational point of view), it has been proposed to add to clauses 4 and 6 the codicil "together with a proof that f has this property". Of course that settles nothing, and the Byzantine discussions about the meaning which would have to be given to this codicil — discussions without the least mathematical content — only serve to discredit an idea which, we repeat, is one of the cornerstones of Logic

These constructions are typed programs! (Think Curry-Howard proof as program).

CHAPTER 2

Natural deduction is a slightly paradoxical system: it is limited to the intuitionistic case (in the classical case it has no particularly good properties) but it is only satisfactory for the (∧,⇒, ∀) fragment of the language: we shall defer consideration of ∨ and ∃ until chapter 10. Yet disjunction and existence are the two most typically intuitionistic connectors!

Natural Deduction:

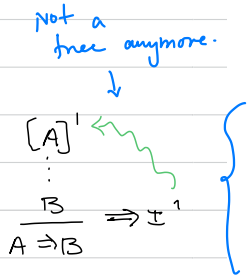
Natural deduction is a proof system that presents proofs as trees.

The tree structure has none or more hypothesis and a single conclusion. This allows us to uniquely determine what the last rule used was (always).

Setup: We denote a deduction of A by $\begin{array}{c}\vdots\\ A\end{array}$

A proof / deduction will be a finite tree with leaves being sentences.
Sentences can be dead (discharged) or alive.
Alive sentences at the leaves will be termed hypotheses.
We discharge hypotheses by using the rules of the calculus, once discharged they no longer play a part in the deduction.

The basic idea of natural deduction is an asymmetry: a proof is a vaguely tree-like structure (this view is more a graphical illusion than a mathematical reality, but it is a pleasant illusion) with one or more hypotheses (possibly none) but a single conclusion.



Its critical to know when a hypothesis was discharged, and so we link the use of rules that discharge with the hypotheses, this however destroys the tree structure in all but graphical representation.

Natural deduction has the following rules

- Hypothesis: A — i.e. from hypothesis A we have a deduction of A.
- $\dfrac{\begin{array}{c}\vdots\\ A\end{array}\quad\begin{array}{c}\vdots\\ B\end{array}}{A \wedge B}\ \wedge I$ — From a deduction of A and a deduction of B we have a deduction of $A \wedge B$.

What kind of objects does natural deduction operate on? What is A..

I guess its the usual set-up of $\mathcal{V}$ infinite var & $\Lambda = \mathcal{V} \cup \{\wedge, \vee, \Rightarrow, \forall, .\}$

Then

$B, A ::= x \mid A \mid A \Rightarrow B \mid A \wedge B \mid \ldots$

Is it just a formalism for expressing propositional logic proofs.

Natural deduction is just propositional logic (intuitionistic)

- $\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Rightarrow B} \Rightarrow I$
- $\dfrac{\begin{array}{c}\vdots \\ A\end{array}}{\forall \xi . A} \forall I$ — We require that ξ is not a free variable in the hypotheses
- $\dfrac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{A} \wedge 1E$
- $\dfrac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{B} \wedge 2E$
- $\dfrac{\begin{array}{cc}\vdots & \vdots \\ A & A \Rightarrow B\end{array}}{B} \Rightarrow E$
- $\dfrac{\begin{array}{c}\vdots \\ \forall \xi . A\end{array}}{A[a/\xi]} \forall E$

Computational Interpretation:

This is just a conceptual framework or actually a specific semantics for a specific system?
-> If so which

Applying Heytings semantics after, fixing an interpretation of the atoms & the quantifiers, we can think of A as a set, whose elements are all possible deductions of A.

Then a deduction of A from $B_1, \ldots, B_n$ can be considered a function $t(x_1, \ldots, x_n)$ such that $t(b_1, \ldots, b_n) \in A$

We work with parcels of hypotheses; B_i may appear several times amongst the hypotheses & two occurences of B_i in the same parcel will correspond to the same variable.

In contrast with a set? I think this kind of talking about tuples of sets

$(\{B \ldots\}, \{B \ldots\}, \{B, B\}, \ldots)$

can simplify $\{B\}$

Notes on Nat' Deduction:

1. When given a proof with multiple of the same hypothesis we can choose to to discharge one or more of them. There is some logical content here:

https://youtu.be/fpIXJ_X4XDM

A talk by Dan on this topic

2. If we use $\wedge I$ to get a deduction from $u(x_1, \ldots, x_n)$ & $v(x_1, \ldots, x_n)$ then we can label this deduction the pair $\langle u(x_1, \ldots, x_n), v(x_1, \ldots, x_n) \rangle$

π_1 & π_2 are the projections

3. $\wedge 1E$ takes $\pi_1(t(x_1, \ldots, x_n))$ where $t(x_1, \ldots, x_n)$ was the sub deduction $\wedge E$ was applied to. $\wedge 2E$ is $\pi_2(t(x_1, \ldots, x_n))$

Although this is not very formal, it will be necessary to consider the fundamental equations:
π1hu, vi = u π2hu, vi = v hπ1t, π2ti = t
These equations (and the similar ones we shall have occasion to write down) are the essence of the correspondence between logic and computer science

pg 11

4. We associate to a discharge (from $\Rightarrow$I) a binding in the λ-cale.

The rules of Natural deduction lead to a notion that two proofs are the same

$$\dfrac{\dfrac{\vdots\ A \qquad \vdots\ B}{A \wedge B}}{A}$$ Detour

Girard claims, is somehow the same as

$$\vdots\ A$$

CHAPTER 3

Look up: Scotts Semantics

The Curry-Howard Isomorphism:

Girard now establishes the λ-calc as a formal system to capture:
- "The functional objects behind the proof"
- The equations written above

We can interpret typed λ terms in two ways
- Sense: Rewrite Rules
- Referent: Equations defining equality

Types: Formulas (propositions) become types (λ-calc)

$$U, V ::= T_1 \mid \cdots \mid T_n \mid U \to V \mid U \times V$$

($T_1 \mid \cdots \mid T_n$: Atomic types; $U \to V$: $\Rightarrow$; $U \times V$: $\wedge$)

Terms: Proofs become λ terms with type the proposition that they prove.

Axioms of typing:

selecting variable names, not capturing, what else.

Issues of binding variables & substitution can be solved to ensure a well defined system so we assume we have fixed such a system. (The details of which are tedious & unhelpful).

- Variables $x_1^T, \ldots, x_n^T, \ldots$ are type T
- u type U & v type V gives that $\langle u, v \rangle$ is type $U \times V$.
- t type $U \times V$ gives $\pi_1(t)$ type U & $\pi_2(t)$ type V.
- v type V & x_n^U type U then $\lambda x_n^U . v$ type $U \to V$
- t type $U \to V$, u type U then tu type V.

Interpretation of Referent:

This is an external interpretation that will make the CH isomorphism more intuitive & motivate the rules of typing.

Types give the KIND of object under consideration. If x has type:
- $U \to V$ x is a function from $U \to V$
- $U \times V$ x is an ordered pair

The meaning of atomic variables is not important.

We can interpret the terms of the λ-calc as
- $\langle u, v \rangle$ an ordered pair
- $\pi_i t$ the i^{th} projection
- $\lambda x_i^U . v$ is the function which associates to any u of type U the formula $v[u/x]$
- tu is applying function t to u.

Check with other text. Are these axioms or what?

This interpretation can be summarised in the following equations

- $\pi_1(u,v) = u$
- $\pi_2(u,v) = v$
- $(\lambda x^U . v)u = v[u/x]$
- $(\pi_1(t), \pi_2(t)) = t$
- $\lambda x^U . tx = t$

Decidable:

T. These equations give a consistent & decidable system

[Proof: Not Given]

Interpretation of Sense: Terms are programs; They calculate the denotation - The type of a program can be seen as what the program abstractly does.
We can think of the equations above as rewrite rules.

Conversion: A term t converts to t', if one of the following holds:

- $t = \pi_1(u,v)$, $t' = u$
- $t = \pi_2(u,v)$, $t' = v$
- $t = (\lambda x^U . v)u$, $t' = v[u/x]$

When t converts to t' t is the redex & t' the contractum-

t is normal if it has no subterms of the following forms:

- $\pi_1(u,v)$
- $\pi_2(u,v)$
- $(\lambda x^U . v)u$

$u \twoheadrightarrow v$ when there is a sequence $u = t_0, \ldots, t_n = v$ such that for each i t_i is a redex & t_{i+1} the contractum- $\twoheadrightarrow$ pronounced "reduces". This is a reflexive & transitive relation.

A normal form n of t is a term such that $t \twoheadrightarrow n$ & n is normal.

Are there any conditions on y?

T. A term t is normal iff it is in head normal form

$\lambda x_1 \cdots x_n \cdot y\, u_1 \cdots u_m$

Where the u_i are in normal form.

PROOF:

($\Leftarrow$) is immediate

($\Rightarrow$) t in normal form. Induct on # of subterms in t. Therefore it must be a

- Variable : Trivially in HNF
- An abstraction : Trivially in HNF
- An application : $t = uv$ where u is a normal term (not an abstraction by t's normality) & v is a variable.

Then applying induction hypothesis u is in HNF $\Rightarrow$ $t = uv$ is in HNF. ∎

I think v is a variable but why can't it be another normal λ term. ie. Why can we always break a normal term into Nterm - variable.

I don't follow

It follows from this that if the free variables in t have strictly simpler type than t then t is an abstraction.

A λ term is closed if it has no free variables. So in particular all closed terms are abstractions.

The isomorphism: We give an explicit translation of natural deduction to λ terms.

1. To the deduction A (A in parcel i) corresponds the variable x_i^A.

2. To the deduction $\dfrac{\begin{array}{cc}\vdots & \vdots \\ A & B\end{array}}{A \wedge B}\wedge\mathcal{I}$ corresponds $\langle u, v\rangle$ where u and v correspond to the deductions of A and B.

3. To the deduction $\dfrac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{A}\wedge 1\mathcal{E}$ (respectively $\dfrac{\begin{array}{c}\vdots \\ A \wedge B\end{array}}{B}\wedge 2\mathcal{E}$) corresponds $\pi^1 t$ (respectively $\pi^2 t$), where t corresponds to the deduction of $A \wedge B$.

4. To the deduction $\dfrac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Rightarrow B}\Rightarrow\mathcal{I}$ corresponds $\lambda x_i^A.v$, if the deleted hypotheses form parcel i, and v corresponds to the deduction of B.

5. To the deduction $\dfrac{\begin{array}{cc}\vdots & \vdots \\ A & A \Rightarrow B\end{array}}{B}\Rightarrow\mathcal{E}$ corresponds the term $t\,u$, where t and u correspond to the deductions of $A \Rightarrow B$ and B.

This translation defines only a bijection. We have talked about conversion & normal forms for λ-calc terms; There is an analogous notion for natural deduction & it is this structure that is preserved under the bijection, giving it the further title of an isomorphism.

A justification to the philosophical outlook.

This isomorphism is relevant to Girard because:

- Tells us that all constructive logic must have an operational side.
- One cannot work with typed calculi without considering the symetries of logic.

If one attemps to "improve" typing rules without considering the logical implication they will fail.

I get the idea but is there something concrete he's talking about.

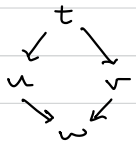
- What improvement
- What broke
  - → Logically
  - → Operationally.

CHAPTER 4

The Normalisation Theorems:

The two results here will guarantee that a unique normal form always exists for simply typed λ terms.
This means it is computationally very well behaved.

Church-Rosser: If $t \twoheadrightarrow u, v$ then there exists a w $u, v \twoheadrightarrow w$



This implies normal forms are unique immediately. (because they reduce only to themselves)

Note this is a statement independent of existence of normal forms & can be made sense of even in untyped calculi.

Proof: Not Given

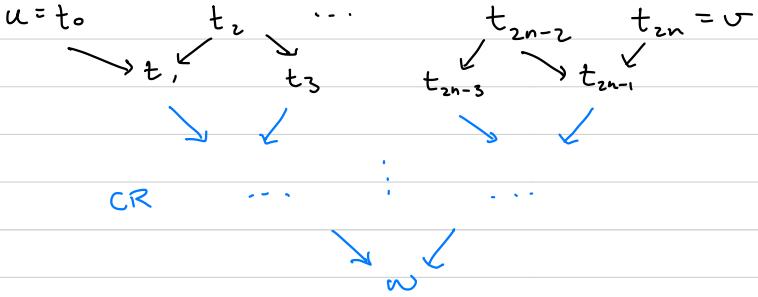
Consistency: CR shows consistency of λ-calc.
Consistency in logic means that the system implies no contradiction. Here we take it to mean that it is not the case that $u = v$ is deducible for all u & v of the same type.
Not deducible in particular from our set of equations (3.2).

A lemma before the proof

Lemma: $u = v$ is derivable $\Rightarrow \exists w$ $u, v \twoheadrightarrow w$

Proof
$u = v$ derivable $\Rightarrow \exists u = t_1, \ldots, t_{2n} = v$ (an even length sequence of reductions) such that for $i = 0, \ldots, n-1$
$t_{2i}, t_{2i+2} \twoheadrightarrow t_{2i+1}$
Then apply CR repeatedly to get the w

In particular
$u = v \iff u \twoheadrightarrow v$ or $v \twoheadrightarrow u$
so $u = v \Rightarrow \exists u = j_0, \cdots, j_m = v$
(assume WLOG $u \twoheadrightarrow v$)
a sequence of reductions
Then the sequence
$j_0, j_1, j_1, j_2, j_2, \cdots, j_m, j_m$
satisfies the conditions.



Proof (of consistency):
Using the contrapositive of the lemma; if u & v are distinct normal forms no w can exist such that $u, v \twoheadrightarrow w$ and thus $u \neq v$. ⌟

If our calculus is built from more than one variable. ←

(so we have consistency if we have distinct normal forms, which we explicitly do).

Weak Normalisation:

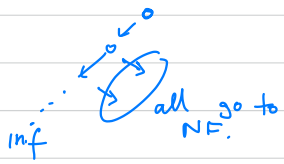
T. There exists a normal form for every term.

Note the normal form is unique by CR. WN gives the decidability of the denotational equivilence ($u, v \twoheadrightarrow w$) because we can compute u & v's respective unique normal forms & then compare them.

Does SN follow immediately from CR + WN?

↘ NO.

WN tells us some path would exist → But not every path



Strong Normalisation:

A term is strongly normalising if no sequence of reductions is infinite.

T. Every λ-term is strongly normalisable.

proof: Delayed

To prove them we need some machinery.

Degree:

The degree of a type T, denoted $\partial(T)$ is

- $\partial(T_i) = 1$ $\quad T_i$ atomic
- $\partial(U \to V) = \partial(U \times V) = \max(\partial(U), \partial(V)) + 1$

The degree of a redex (term before conversion) is

- $\partial(\pi_i \langle u, v \rangle) = \partial(U \times V)$ for $\langle u, v \rangle$ type $U \times V$
- $\partial((\lambda x.v)(u)) = \partial(U \to V)$ for $\lambda x.v$ type $U \to V$

The degree of a term is the sup over its redex degrees:

$$d(t) = \sup \{ \partial(r) : r \text{ a redex in } t \}$$

We define $\partial(n)$ of a normal term to be 0.

A redex is a term too so it makes sense to look at $\partial(r)$ & $d(r)$. Its clear that $\partial(r) \leq d(r)$

Lemma: x type $U \Rightarrow d(t[u/x]) \leq \max(d(t), d(u), \partial(U))$

Proof: Consider the possible terms in $t[u/x]$ (because d takes the sup we need to show that i.e. all the elements are less)

- redexes of t (represented by $d(t)$)
- redexes of u ($\longrightarrow$ $d(u)$)
- New redexes

We can have new redexes by

- $\pi_i x$ & $u = \langle u_1, u_2 \rangle$
- $x\, v$ & $u = \lambda y . u_1$

But all these new redexes have degree $\partial(U)$. ∎

pg 26. Isnt it equal by def? → r a redex type T then $\partial(r) > \partial(T)$

mma: $t \rightsquigarrow u \Rightarrow d(t) \geq d(u)$

Proof: Consider only one step conversions. $t \rightsquigarrow u$ by replacing r in t by c. Then only the following can happen

- Redexes in t but not in r are reduced by replacing r with c.

↳ not clear come back.

maximal degree? → Just assuming $d(r) \leq n$

Lemma: r a redex of maximal degree n in t, further assume all redexes in r have a degree less than n.

If u is obtained from t by converting r to c, u has strictly fewer redex of degree n.

Proof

Actually is it clear: The degree of a term is not well defined: if t is a redex is "t a redex in t"? is its own degree suped over? Does Sup introduce any a priori issues showing we don't need a $+1$?

↳ i.e. need W.N. to show sup is given by max: (either sup is ∞ or its equal to max.

CHAPTER 5

Sequent Calculus:

SC is not limited to the intuitionistic case.

A sequent is something of form $\underline{A} \vdash \underline{B}$ where $\underline{A}$ is a finite sequence of formulae $A_1, \dots, A_n$ (same for $\underline{B}$).

From an algorithmic viewpoint, the sequent calculus has no Curry-Howard isomorphism, because of the multitude of ways of writing the same proof.

Is Natural deduction also has multiple ways to write the same proof.

Is the point that Curry-Howard is just not as straight forward to present.

An interpretation:

This is the interpretation it was created to have by Gentzen (to talk about Nat D).

One CAN think of this system as representing $A_1 \wedge \cdots \wedge A_n \Rightarrow B_1 \vee \cdots \vee B_m$. The $\underline{A}$ (conjunction) implies $\underline{B}$ (disjunction) One could then interpret

- $\vdash \underline{B}$ as just asserting $B_1 \vee \cdots \vee B_m$
- $\underline{A} \vdash$ as asserting $\neg(A_1 \wedge \cdots \wedge A_n)$
- $\emptyset \vdash \emptyset$ asserts a contradiction

The Rules:

- Exchange (conjunction/disjunction are commutative)

$$\frac{\underline{A}, B, C, \underline{D} \vdash \underline{E}}{\underline{A}, C, B, \underline{D} \vdash \underline{E}} \; LX \qquad \frac{\underline{A} \vdash \underline{B}, C, D, \underline{E}}{\underline{A} \vdash \underline{B}, D, C, \underline{E}} \; RX$$

Girard claims that:
- These rules determine the behavior of the logical operations
- What do these rules operationally
- They can be restricted to give linear logic. Assuming that there is an iso to Nat Ded is it these rules that make it so?
Or that does NatDed have an iso the UNTYPED λ if we remove its structural rules.

- Weakening

$$\frac{\underline{A} \vdash \underline{B}}{\underline{A}, C \vdash \underline{B}} \; LW \qquad \frac{\underline{A} \vdash \underline{B}}{\underline{A} \vdash \underline{B}, C} \; RW$$

- Contraction

$$\frac{\underline{A}, C, C \vdash \underline{B}}{\underline{A}, C \vdash \underline{B}} \; LC \qquad \frac{\underline{A} \vdash C, C, \underline{B}}{\underline{A} \vdash C, \underline{B}} \; RC$$

Intuitionistic SC:

We acheive this by restricting to the case of sequents of the form $\underline{A} \vdash \underline{B}$ where $\underline{B}$ is at most one formula.

It will be noticed that in this case the only right hand structural rule is RW.

It is also possible to get intitionistic SC by restricting to intuitinistic sequents.

The intuitionistic restriction is in fact a modification to the management of the formulae — the particular place distinguished by the symbol ` is a place where contraction is forbidden — and from that, numerous properties follow

Is this correct? *

Logical Rules:

- For every formula we have the axiom $C \vdash C$
- Somehow dual to this is the cut axiom

Read Gentzen's paper. Where does SC end & SC as a tool for specifically prop' logic begin? Does he distinguish?
↳ I mean you can use it as a proof system for other things so...

Note that $\underline{A} = A_1, \ldots, A_n$ but C is a single word.

↳ kind of a response from Girard:

one can amuse oneself by inventing one's own logical operations, but they have to respect the left/right symmetry, otherwise one creates a logical atrocity without interest.

$$\frac{\underline{A} \vdash C, \underline{B} \qquad \underline{A'}, C \vdash \underline{B}}{\underline{A}, \underline{A'} \vdash \underline{B}}\ \text{cut}$$

Note that it can be shown that the cut rules can be totally removed.

1. *Negation*: the rules for negation allow us to pass from the right hand side of "$\vdash$" to the left, and conversely:

$$\frac{\underline{A} \vdash C, \underline{B}}{\underline{A}, \neg C \vdash \underline{B}}\,\mathcal{L}\neg \qquad \frac{\underline{A}, C \vdash \underline{B}}{\underline{A} \vdash \neg C, \underline{B}}\,\mathcal{R}\neg$$

2. *Conjunction*: on the left, two unary rules; on the right, one binary rule:

$$\frac{\underline{A}, C \vdash \underline{B}}{\underline{A}, C \wedge D \vdash \underline{B}}\,\mathcal{L}1\wedge \qquad \frac{\underline{A}, D \vdash \underline{B}}{\underline{A}, C \wedge D \vdash \underline{B}}\,\mathcal{L}2\wedge$$

$$\frac{\underline{A} \vdash C, \underline{B} \quad \underline{A'} \vdash D, \underline{B'}}{\underline{A}, \underline{A'} \vdash C \wedge D, \underline{B}, \underline{B'}}\,\mathcal{R}\wedge$$

can't this be eliminated by $\neg$ & $\wedge$.

3. *Disjunction* obtained from conjunction by interchanging right and left:

$$\frac{\underline{A}, C \vdash \underline{B} \quad \underline{A'}, D \vdash \underline{B'}}{\underline{A}, \underline{A'}, C \vee D \vdash \underline{B}, \underline{B'}}\,\mathcal{L}\vee$$

$$\frac{\underline{A} \vdash C, \underline{B}}{\underline{A} \vdash C \vee D, \underline{B}}\,\mathcal{R}1\vee \qquad \frac{\underline{A} \vdash D, \underline{B}}{\underline{A} \vdash C \vee D, \underline{B}}\,\mathcal{R}2\vee$$

Special case: The intuitionistic rule $\mathcal{L}\vee$ is written:

$$\frac{\underline{A}, C \vdash \underline{B} \quad \underline{A'}, D \vdash \underline{B}}{\underline{A}, \underline{A'}, C \vee D \vdash \underline{B}}\,\mathcal{L}\vee$$

where $\underline{B}$ contains zero or one formula. This rule is not a special case of its classical analogue, since a classical $\mathcal{L}\vee$ leads to $\underline{B}, \underline{B}$ on the right. This is the only case where the intuitionistic rule is not simply a restriction of the classical one.

4. *Implication*: here we have on the left a rule with two premises and on the right a rule with one premise. They match again, but in a different way from the case of conjunction: the rule with one premise uses *two* occurrences in the premise:

$$\frac{\underline{A} \vdash C, \underline{B} \quad \underline{A'}, D \vdash \underline{B'}}{\underline{A}, \underline{A'}, C \Rightarrow D \vdash \underline{B}, \underline{B'}}\,\mathcal{L}\Rightarrow \qquad \frac{\underline{A}, C \vdash D, \underline{B}}{\underline{A} \vdash C \Rightarrow D, \underline{B}}\,\mathcal{R}\Rightarrow$$

5. *Universal quantification*: two unary rules which match in the sense that one uses a *variable* and the other a *term*:

$$\frac{\underline{A}, C[a/\xi] \vdash \underline{B}}{\underline{A}, \forall\xi.\, C \vdash \underline{B}}\,\mathcal{L}\forall \qquad \frac{\underline{A} \vdash C, \underline{B}}{\underline{A} \vdash \forall\xi.\, C, \underline{B}}\,\mathcal{R}\forall$$

$\mathcal{R}\forall$ is subject to a restriction: ξ must not be free in $\underline{A}, \underline{B}$.

6. *Existential quantification*: the mirror image of 5:

$$\frac{\underline{A}, C \vdash \underline{B}}{\underline{A}, \exists\xi.\, C \vdash \underline{B}}\,\mathcal{L}\exists \qquad \frac{\underline{A} \vdash C[a/\xi], \underline{B}}{\underline{A} \vdash \exists\xi.\, C, \underline{B}}\,\mathcal{R}\exists$$

$\mathcal{L}\exists$ is subject to the same restriction as $\mathcal{R}\forall$: ξ must not be free in $\underline{A}, \underline{B}$.

Spectacular consequences of cut elimination:

T. If we can prove A in predicate calculus we can show $\vdash A$ without cut.

Girard notes that in the case analysis of the last rule used we get a justification for a restriction on sequents (leading to linear logic) to make the last rules more manageble / symetric.

Next consider looking at a conclusion of a proof. What premises were used?
Well if cut is allowed we cannot say because an arbitrary formula is removed. All other rules have the property that the premises of a conclusion must be subformulas.

Conclusion	Subformulas
$A \wedge B$, $A \vee B$, $A \Rightarrow B$	A, B
$\neg A$	A
$\forall \xi \cdot A$, $\exists \xi \cdot A$	$A[a/\xi]$

a an arbitrary term ←

Without cut we dont have the symetry of $A \vdash A$. So A on the left is stronger than A on the right. Girard claims this is interesting.

why? ←

SC & ND: We can surjectively map from SC to natural deduction.

why "noble" ← Girard restricts to ND without $\forall, \exists, \neg$

To $\underline{A} \vdash B$ corrosponds a deduction of B on hypotheses $\underline{A}$.
The reverse translation is not unique.
Girard writes out an explicit translation.

The hypotheses are "parcels"? whats a parcell.

I will transcribe only the properties here:

- The map is not injective. In particular

for a given ND deduction there may be more than one SC proof.

- Sequent calculus is perceived by Girard (& Gentzen) as 'authentically' a proof than natural deduction.
  It is a tool to reason about ND.

- Logical rules on the right correspond to introduction rules
- Logical rules on the left correspond to elimination rules
- Contraction & weakening correspond to creating (mock) parcels
- Exchange corresponds to nothing.

sequent calculus can complicate things "unnecissarily". Its very bearocratic.

- Translating a cut free proof always gives a normal form.

CHAPTER 6

For simple typed λ-calculus, there is proof theoretic techniques which make it possible to express the argument of the proof in arithmetic, and even in a very weak system. However our method extends straightforwardly to Gödel's system T, which includes a type of integers and hence codes Peano Arithmetic. As a result, strong normalisation implies the consistency of PA, which means that it cannot itself be proved in PA (Second Incompleteness Theorem)

Strong Normalisation Theorem

The goal is to proove this theorem using a technique we can later generalise to system F.

Reducibility:

So things in RED_T will have types consisting of $T, \times, \to$
eg. $T \times T \to (T \to T)$

Define the set of reducible terms of type T (in simply typed λ-calc) RED_T as

- t of atomic type T is reducible (in RED_T) if it is strongly normalisable
- t type $U \times V$ is reducible iff $\pi_1 t$ & $\pi_2 t$ are redueible
- t type $U \to V$ is reducible iff for all reducible u of type U, tu is reducible type V.

The deep reason why reducibility works where combinatorial intuition fails is its logical complexity. Indeed, we have:
$t \in REDU \to V$ iff $\forall u\ (u \in REDU \Rightarrow t\,u \in REDV)$

We see that in passing to $U \to V$, REDU has been negated, and a universal quantifier has been added. In particular the normalisation argument cannot be directly formalised in arithmetic because $t \in REDT$ is not expressed as an arithmetic formula in t and T.

A neutral term is one of form

- x
- $\pi_i t$
- tu

Namely not of form $\langle u, v \rangle$ or $\lambda x.t$

Where x, t, u are? Variables of terms?

How are we reasoning about this? I mean this is outside of any formal system (presumably why we don't use $\Leftrightarrow$) so is this as "meta" as we get & just start going with our intuition.

T. A term t is strongly normalisable iff there is a number $\nu(t)$ which bounds the length of every normalisation sequence

Proof: ($\Rightarrow$) Trivial ($\Leftarrow$) Konigs Lemma

The proof is spelled out by James Clifts notes.

CHAPTER 7

System T:

System T is a λ-calc extension to make it more expressive than simply typed.

For example, can we use it to represent the integers or the booleans, and if so can we represent sufficiently many functions on them? The answer is clearly no.

↳ Church numerals
Extended polynomials.

System T has two drawbacks
- Its terms / types no longer correspond to proofs (in any 'extended logics' either)
- It begs the question of what 'improvements' & extensions to make.

(supposedly system F resolves these satisfactorily).

Types: T adds types int & bool as well as structural rules and constants to moderate their behaviour

Constants & Rules:

- int introduction
  - 0 is a constant of type int
  - t type int $\Rightarrow$ St type int
- int elimination
  u, v, t having respective types U, $U \to (\text{int} \to U)$, int
  Then $Ruvt$ is type U
- Bool introduction
  T & F are constants of type bool
- Bool elimination
  u, v, t respective types U, U, bool
  then $Duvt$ is type U.

The redexes imply we are working in λ-calc? in part as above? Same for right?

Along with the new redexes
- $Ruv0 \rightsquigarrow u$
- $Ruv(St) \rightsquigarrow v(Ruvt)t$
- $DuvT \rightsquigarrow u$
- $DuvF \rightsquigarrow v$

An interpretation:
- 0 is zero
- S is the successor function
- T & F are truth values
- R is a recursion operator. In particular $Ruv(n+1) = v(Ruvn)n$
- D is the operator "if ... then ... else".

strong normalisation & Church-Rosser

T. System T is strongly normalising. ie. Each term is strongly normalisable to a unique normal form.
[Proof: Essentially the same]

Comments on the expressive power of T

ALMOST
We can construct the usual boolean operations of negation, disjunction & conjunction.

shown in 9.3.1
Absolutely wild. ←
"semantic methods"?
Could we not already construct these in simply typed?

No term in T will give a symetric disjunction operator.

We can represent the integers in T by $\bar{n} = S^n(0)$.
One can define addition
$+[x,y] = R_x(\lambda z^{int}.\lambda z'^{int}.Sz)y$
This has the properties we need; Namely
• $+[x,0] \rightsquigarrow x$ • $+[x,Sy] \rightsquigarrow S(+[x,y])$

One can also define predicates on integers (truth valued functions)

So T gives us the bare minimum.
But how far can we go with it?

We have the iterator it as a term
type $(int \to int) \to (int \to int)$
given by $\lambda x^{int \to int}.\, it(x)$
Which gives $it(f)\bar{n} \rightsquigarrow f^n(\bar{1})$

is that right? ←

Look into classifications of functions.
& their relation to Real ⁂'s -

This unlocks a huge class of functions.
We can easily build Ackermans function
$n \mapsto it^n(f_0)$ (for given f_0) which for relatively simple f_0 will grow faster than any primitive recursive functions.

can it or not?

The one-step predecessor satisfying the equations pred(O) = O, pred(S x) = x cannot be constructed using the iterator: R is essential. In fact, if one has only the iterator one can define the same functions but a certain number of equations with variables disappear. So the predecessor will still be definable, but will satisfy pred(S t) t only when t is of the form n. In other words by values. This is a little annoying (in particular for F, where we shall no longer have anything but the iterator), for it shows that to calculate pred(n), the program makes n steps, which is manifestly excessive. We do not know how to type the predecessor, except in systems like T, where the solution is visibly ad hoc.

Time but it is definable still

Fake

we show these are the only normal forms anyway

Next if we have a closed term of type $int \to int$, called t, it induces a function $|t| : \mathbb{N} \to \mathbb{N}$ by

$$|t|(n) = m \iff t\bar{n} \rightsquigarrow \bar{m}$$

The algorithm to normalise the term t tells us $|t|$ is calculable and therefore recursive.

We can think of $|t|$ as looking for a normal form $(t\bar{n})$ and so by strong normalisation of T we can guarantee that $|t|$ as a program will always terminate.

what guarantee do we have that Int represents the integers, Bool the booleans, etc.? It is not because we have represented the integers in the type Int that this type can immediately claim to represent the integers.

I think I get what he's saying but he's saying it horrendously

Can we make sense of this in a mathematical framework (instead of T).

Yes it is possible to show $|t|$ terminates in Peano Arithmetic. $|t|$ is "Provably Total".

In fact Provably total functions are exactly the terms of type $int \to int$ in T.

f provably Total $\iff \exists t \in T^{int \to int}$ $|t| = f$.

In other words, the expressive power of the system T is enormous, and much more than what is feasible1 on a computer! T

↳ what class of functions is feasable?

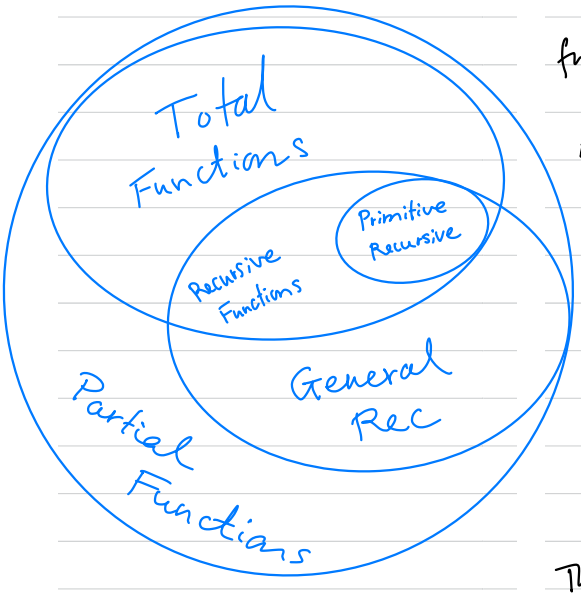
what else do I know about function classes: • cts • Diffreantiable • Compact Open

Appendix: Function classes

Primitive Recursive:

Intuitively these are functions that can be computed by a program using only for loops.
ie. The # of iterations is bounded above before the loop is entered.

They are the subset of total general recursive functions.



provably total / computable?

enumerable

A primitive recursive function is a map $f: \mathbb{N}^m \longrightarrow \mathbb{N}$ satisfying:

- Constant map: $\exists n \in \mathbb{N}$ $f(x) = n$ is primitive recursive
- Successor: $S(x) = x+1$ is primitive recursive
- Projection: $P_i(x_1, \dots, x_m) = x_i$ is primitive recursive
- Composition: $g_1, \dots, g_m : \mathbb{N}^k \longrightarrow \mathbb{N}$ primitive recursive & $h: \mathbb{N}^m \longrightarrow \mathbb{N}$ primitive recursive
  $\Rightarrow f = h(g_1, \dots, g_m)$ is primitive recursive.
- Iterate: $g: \mathbb{N}^m \longrightarrow \mathbb{N}$, $h: \mathbb{N}^{m+2} \longrightarrow \mathbb{N}$ primitive recursive

Then $\rho(g,h): \mathbb{N}^{m+1} \longrightarrow \mathbb{N}$ defined by

$$\rho(g,h)(0, x_1, \dots, x_m) = g(x_1, \dots, x_m)$$

$$\rho(g,h)(S(y), x_1, \dots, x_m) = h(y, \rho(g,h)(y, x_1, \dots, x_m), x_1, \dots, x_m)$$

- $S(y)$: iterations
- $x_1, \dots, x_m$: immutable constants / inputs
- y: counter
- $\rho(g,h)(y, x_1, \dots, x_m)$: passed results of g & h applied to lower values of y (i.e. previous iterations)

These functions correspond to Turing machines that always halt.

General Recursive:

Also refered to as computable functions

We add to the primitive recursive conditions

- minimisation: given general recursive function $f: \mathbb{N}^{m+1} \longrightarrow \mathbb{N}$ then $\mu(f): \mathbb{N}^m \longrightarrow \mathbb{N}$ is general recursive & defined by

$$\mu(f)(x_1, \dots, x_m) = z$$
$$\Leftrightarrow f(i, x_1, \dots, x_m) \neq 0 \quad i = 1, \dots, z-1$$
$$f(z, x_1, \dots, x_m) = 0$$

This clause finds the smallest value z such that $f(-, x_1, \dots, x_m)(z) = 0$.

This is what can cause general recursive functions to be partial; if this minimisation "search" never terminates i.e. f is always non zero.

This class of functions corresponds exactly to functions computable by a turing machine or

defined by λ-calculus.

Total Function: Familiar set theoretic function.

Partial Function: A normal set theoretic function that is however only defined on a subset of the domain.
i.e. it is allowed that for $f: X \to Y$ $\nexists\, S \subseteq X$ $f(S) = \emptyset$.

Details of Peano A.: 1st & 2nd

↑
why doesn't anyone go to 3rd?

CHAPTER 11

System F:

Adds an operation of abstraction for types. All typical data types are conveniently definable in F.

Type variable vs type? The variable is the symbol only the type is the "object" whatever?

Types: $U, V ::= X \mid Y \mid \cdots \mid U \to V \mid \Pi X.V$

Terms:

- Variables: $x^T, y^T, \ldots$ of type T.
- Applications: For $t^{U \to V}$ & u^U, tu is a term of type V
- λ-Abstraction: x^U & v^V then $\lambda x^U.v$ is of type $U \to V$
- Universal Abstraction: v^V then $\Lambda X.$ is a term of type $\Pi X.V$ (where X is not free in any of the types of the free variables in v)

Corresponds under Curry-Howard to $\forall$ rules.

- Universal Application: t has type $\Pi X.V$ & U is some type then tU is a term of type $V(U/X)$

Corresponding under CH to similar restrictions.

This adds the following conversion

$$(\Lambda X.v)U \rightsquigarrow v[U/X]$$

Interpretation: An object of type $\Pi X.V$ can be naively viewed as a function which replaces the argument type in the given λ term by X.

eg. $(\Lambda X.\lambda x^{X \to X \to V}.xy)U \rightsquigarrow \lambda x^{U \to U \to V}.xy$

Representing Structures: We can represent common structures in F.

- Booleans
- Product
- Sum
- Empty type

He gives a representation of free structures in F.

In general we can represent free structures. This includes integers, lists & trees.

Curry-Howard: Types in F are propositions with second order quantifiers.

$\Lambda X.t$: $t : A$

$$\frac{\vdots\ A}{\forall X.A}\ \forall^2 I \qquad \frac{\vdots\ \forall X.A}{A[B/X]}\ \forall^2 E$$

Correspond to universal abstraction & application in F.

$(\Lambda X.t)\,U \rightsquigarrow t[U/X]$ corresponds to equating the two deductions

$$\frac{\vdots\ A}{\dfrac{\forall X.A}{A[B/X]}\ \forall E}\ \forall I \qquad \& \quad \vdots\ A[B/X]$$

CHAPTER 13

Cut Elimination

T. The cut rule is redundant in SC.

Discussion:

- We capture the complexity (# of cuts) by using the concept of degree; the complexity of the formula eliminated in the cut
- We show given a proof with a cut we can construct one with a lower degree (may have more cuts)
- Induct to see that we can reduce the degree to 0 & therefore have no cuts.
- Cut elimination in the worst case increases the height of the proof hyperexponentially A proof of height h & degree d at worst becomes $\mathcal{H}(d,h)$ where:

$$\mathcal{H}(0,h) = h, \quad \mathcal{H}(d+1,h) = 4^{\mathcal{H}(d,h)}$$

- If we have nontrivial axioms we only need to use cut on sequents with proper (non-trivial) axioms.

proof is largely "It follows from structural rules"

$\mathcal{H}(d,h) = 4^{4^{\cdot^{\cdot^{\cdot^{4}}}}}$ (h 4's, d-ish)

Nontrivial would be $A \vdash B$ (contra $A \vdash A$)

As a consequence, if we confine ourselves to atomic sequents (built from atomic formulae) as proper axioms, and as the conclusion, there is no need for the logical rules.

CHAPTER 14

Strong Normalisation

T. All terms in system F are strongly normalisable.

Proved in PA_2?

Existence is much more delicate; in fact, we shall see in chapter 15 that the normalisation theorem for F implies the consistency of second order arithmetic PA2. The classic result of logic, if anything deserves that name, is Gödel's second incompleteness theorem, which says (assuming that it is not contradictory) that the consistency of PA2 cannot be proved within PA2. Consequently, since consistency can be deduced from normalisation within PA2, the normalisation theorem cannot be proved within PA2