

Specifications of Computations in Intuitionistic and Linear Logic

Alwen Tiu

Australian National University

ANU Logic Summer School
Lecture 1, December 12, 2007

Motivations

- Constructive logics, in particular intuitionistic and linear logics, play an important role in theoretical computer science. The emphasis is on *proofs* (or proof constructions), rather than validity.
- The study of *proposition-as-type* and proof normalisation form the basis of many of the functional programming languages used today.
- The study of *proof search* serves as a foundation for logic programming and leads to the discovery of new logic programming languages. Proof search techniques also play an important role in mechanised reasoning, which is the basis of formal verification.

Outline

- 1 Proof theory of intuitionistic logic.
- 2 Proof theory of linear logic.
- 3 Programming with intuitionistic logic: abstract data type and modules.
- 4 Programming with intuitionistic linear logic: resource and states.
- 5 Programming with full linear logic: concurrency.

Constructivism: a brief history

- *Constructivism* is a point of view concerning the concepts and methods used in mathematical proofs, with preference towards constructive concepts and methods. It emerged in the late 19th century, as a response to the increasing use of abstracts concepts and methods in proofs in mathematics.
- There are several branches in constructivism, each with a varying degree of preference towards constructive concepts and methods.
- The (perhaps) most well-known view is the idea of *intuitionism*, pioneered by Brouwer in early 20th century. It essentially rejects the principle of “proof by contradiction” or the “excluded middle”.

Example: a non-constructive proof

Theorem

There are irrational numbers a and b such that a^b is rational.

Proof.

Consider $\sqrt{2}^{\sqrt{2}}$. If it is rational, then we are done: let $a = b = \sqrt{2}$. Otherwise, it is irrational. Then we have

$$(\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \times \sqrt{2}} = \sqrt{2}^2 = 2$$

which is certainly rational. So in this case, let $a = \sqrt{2}^{\sqrt{2}}$ and let $b = \sqrt{2}$. □

Example: constructive or non-constructive?

Theorem

There are infinitely many prime numbers. More precisely, given any number M , there is a prime number bigger than M .

Proof.

Suppose there are only finitely many primes, that is, for some M , all the prime numbers are less than or equal to M . Then we can list them explicitly, say,

p_1, \dots, p_k .

Let $N = p_1 \cdot p_2 \cdot \dots \cdot p_k + 1$. Note that none of p_i divide N . Now, either N is prime or it can be factored into primes. So there is at least one prime q that divides N , but q can't be any of the p_i 's, so q is a prime number not in the list, which contradicts our assumption. □

Brouwer's programme

- In 1912 Brouwer started a programme on reconstructing mathematics on intuitionistic principles. He noted that his approach to mathematics also required a revision of the principles of classical logic, although this was not made precise.
- Brouwer's implicit interpretation of his intuitionistic principles was later made precise by Heyting (1934), a student of Brouwer, and independently by Kolmogorov (1932). This is later known as the Brouwer-Heyting-Kolmogorov (BHK) interpretation of intuitionistic logic.
- Subsequent formalisations of intuitionistic logic, in terms of symbolic systems, were made by Glivenko (1928), and later, by Gentzen (1935) and Prawitz (1960's) into what are most well known today in natural deduction and sequent calculus.

BHK interpretation

In BHK interpretation, the meaning of a statement A is given by explaining what constitutes a *proof of A* .

- 1 For atomic sentences, we assume we know intrinsically what a proof is, e.g., pencil and paper calculation serves as a proof of “ $2 + 3 = 5$ ”.
- 2 A proof of $A \wedge B$ is a pair (p, q) consisting of a proof p of A and a proof q of B .
- 2 A proof of $A \vee B$ is a pair (i, p) with
 - $i = 0$, and p is a proof of A , or
 - $i = 1$, and q is a proof of B .
- 3 A proof of $A \supset B$ is a function f which maps for each proof p of A to a proof $f(p)$ of B .
- 4 Absurdity \perp (‘the contradiction’) has no proof; a proof of $\neg A$ is a construction which transforms any supposed proof of A into a proof of \perp .

Disjunction property

Contrast the BHK interpretation with the Tarskian interpretation of logical operators. In particular, the excluded middle:

$$A \vee \neg A$$

which is classically valid, is not intuitionistically valid. It is not always the case that we can either prove A or its negation. Consider open problem like $P = NP$. Classically, the statement like

P is equal to NP or P is not equal to NP

is valid. But intuitionistically, to show this statement is valid, we have to come up with either a proof of $P = NP$ or a proof of $P \neq NP$.

Existential property

The BHK interpretation of quantifiers:

- 5 A proof of $\forall x.A$ is a function f that maps each point a in the domain of x to a proof of $f(a)$ of $A[a/x]$.
- 6 A proof of $\exists x.A$ is a pair (a, p) where a is a point in the domain of x and p is a proof of $A[a/x]$.

The latter means that whenever one claims that a statement $\exists x.A$ holds, he/she must exhibit an object a such that A is true of a . This is very different from classical logic, where existence proof sometimes means “freedom from contradiction”.

Natural deduction

BHK interpretation of logical formula is formalised by Prawitz into a calculus called *natural deduction*. It is a collection of rules for forming a deduction tree. A single formula

A

constitutes a deduction of A , given the assumption A . The interpretation of \wedge and \supset are as follows:

$$\frac{\begin{array}{c} \vdots \\ A \end{array} \quad \begin{array}{c} \vdots \\ B \end{array}}{A \wedge B} \qquad \frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \supset B}$$

These are called the *introduction rules* for the corresponding logical operators. In the introduction rule for \supset , the assumption A is *discharged*.

Elimination rules

Introduction rules alone are not sufficient to capture intuitionistic provability. We also need *elimination rules*, which are dual to the introduction rules:

$$\frac{\vdots}{A \wedge B} \quad \frac{\vdots}{A \wedge B} \quad \frac{\vdots \quad \vdots}{A \supset B}$$
$$\frac{A \wedge B}{A} \quad \frac{A \wedge B}{B} \quad \frac{A \quad A \supset B}{B}$$

Normalisation

The interaction between the introduction and the elimination rules captures the idea behind the BHK interpretation of implication. This is most apparent in the *normalisation* of proofs:

$$\frac{\begin{array}{c} \vdots \\ \vdots \\ \vdots \\ A \end{array} \quad \frac{\begin{array}{c} [A] \\ \vdots \\ \vdots \\ B \end{array}}{A \supset B}}{B} \quad \Longrightarrow \quad \begin{array}{c} \vdots \\ A \\ \vdots \\ \vdots \\ B \end{array}$$

The meaning of $A \rightarrow B$ as a function is captured by the “plugging-in” of the proof of A into the assumption (or parameter) in the proof of $A \supset B$. Normalisation of proofs corresponds closely to a model of computation called λ -calculus.

λ -calculus

- A universal notation for writing (recursive) functions.
- The basic constructs: *abstractions* and *applications*.

$$s, t ::= x \mid (s \ t) \mid \lambda x. t$$

- Computation is modelled via *reductions*

$$\beta\text{-rule} \quad (\lambda x. s) \ t \rightarrow s[t/x]$$

$$\eta\text{-rule} \quad (\lambda x. s \ x) \rightarrow s, x \text{ not free in } s$$

- Restricted to *simple type* fragment, it is confluent (for both β and $\beta\eta$) and terminating. Simple types are given by:

$$\alpha, \beta ::= a \mid \alpha \rightarrow \beta.$$

Proposition as types, a.k.a. Curry-Howard correspondence

- Proofs in natural deduction formulation of intuitionistic logic have precise correspondence with simply typed λ -calculus.
- Types = Propositions (replace \supset with \rightarrow).
- Well-typed terms = Proofs.
- Introduction rule for \rightarrow corresponds to typing rule for λ -abstraction:

$$\frac{\begin{array}{c} [x : A] \\ \vdots \\ t : B \end{array}}{\lambda x. t : A \rightarrow B}$$

Elimination of \rightarrow corresponds to function application:

$$\frac{\begin{array}{c} \vdots \\ t : A \end{array} \quad \begin{array}{c} \vdots \\ s : A \rightarrow B \end{array}}{(s t) : B}$$

Typing judgments

Another way of writing typing rules for simply typed λ -calculus makes use of judgments of the form $\Gamma \vdash t : \alpha$, where Γ is a *typing environment*, i.e., pairs of variables and types (propositions).

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \quad \frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x_{\alpha}. t : \alpha \rightarrow \beta} \quad x \text{ not free in } \Gamma$$
$$\frac{\Gamma \vdash s : \alpha \rightarrow \beta \quad \Gamma \vdash t : \beta}{\Gamma \vdash (s t) : \beta}$$

Reduction and normalisation

β -reduction in λ -calculus, i.e.,

$$(\lambda x.s) t \longrightarrow s[t/x]$$

corresponds to normalisation in natural deduction.

$$\frac{\begin{array}{c} \vdots \\ t : A \end{array} \quad \frac{\begin{array}{c} [x : A] \\ \vdots \\ s : B \end{array}}{\lambda x.s : A \rightarrow B}}{(st) : B} \quad \Longrightarrow \quad \begin{array}{c} \vdots \\ t : A \\ \vdots \\ s[t/x] : B \end{array}$$

“Defects” of natural deduction

The rules for disjunction:

$$\frac{\vdots}{A} \quad \frac{\vdots}{B} \quad \frac{A \vee B \quad \begin{array}{c} [A] \\ \vdots \\ C \end{array} \quad \begin{array}{c} [B] \\ \vdots \\ C \end{array}}{C}$$

The elimination rule is bad for proof search, because C here has no structural link with the formula being eliminated. The same problem appears in the elimination rule for \exists .

Sequent calculus

Sequent calculus, invented by Gentzen (1935), fixes some defects of natural deduction. In particular, for some systems, e.g., classical logic, it is easier to do metamathematical analysis in sequent calculus rather than natural deduction. A sequent is an expression of the form:

$$\Gamma \longrightarrow \Delta$$

where Γ and Δ are sets or multisets of formulas. As we shall see, different restrictions on the form of sequents can produce different logics.

The top-down symmetry of natural deduction is replaced by the left-right symmetry. Instead of introduction/elimination rules we have left/right introduction rules.

Some rules of sequent calculus

Some rules for classical logic (here we assume sequents with multisets):

$$\frac{A, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \quad \frac{B, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \quad \frac{\Gamma \longrightarrow A, \Delta \quad \Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \wedge B, \Delta}$$

$$\frac{A, \Gamma \longrightarrow \Delta \quad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta} \quad \frac{\Gamma \longrightarrow A, \Delta}{\Gamma \longrightarrow A \vee B, \Delta} \quad \frac{\Gamma \longrightarrow B, \Delta}{\Gamma \longrightarrow A \vee B, \Delta}$$

$$\frac{\Gamma \longrightarrow A, \Delta \quad B, \Gamma \longrightarrow \Delta}{A \supset B, \Gamma \longrightarrow \Delta} \quad \frac{\Gamma, A \longrightarrow \Delta}{\Gamma \longrightarrow A \supset B, \Delta}$$

Notice the nice symmetry in all rules, in particular, the rules for \vee , which does not have the “defect” of natural deduction.

Structural rules

To capture completely classical logic, we need some extra rules that do not deal with logical connectives, the so-called *structural rules*

$$\frac{\Gamma, A, A \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta} \quad \frac{\Gamma \longrightarrow A, A, \Delta}{\Gamma \longrightarrow A, \Delta} \quad \frac{\Gamma \longrightarrow \Delta}{\Gamma, A \longrightarrow \Delta} \quad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow A, \Delta}$$

These are called contraction and weakening rules.

The cut rule

The most important rule in sequent calculus is the so-called *cut rule*:

$$\frac{\Gamma \longrightarrow \Delta, A \quad A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta} \textit{cut}$$

It may vary from one system to another, but it is present in *all* logical systems formalised in sequent calculus. It essentially embodies the principle of *modus ponens*, the core of any formal logical system.

However, notice that the formula A has no structural link with Γ and Δ . That is, this is precisely the “defect” we were talking about in natural deduction. This defect is fixed by showing that the cut rule is *admissible*. This is known as the *cut-elimination* theorem.

Cut-elimination is the central result of sequent calculus; many other properties follow from this result.

Cut elimination

Cut elimination can be seen as an analog of proof normalisation in natural deduction. One typically proves cut-elimination by analysis on the deduction above cuts, e.g.,

$$\frac{\frac{\frac{\vdots}{\Gamma \longrightarrow \Delta, A} \quad \frac{\vdots}{\Gamma \longrightarrow \Delta, B}}{\Gamma \longrightarrow \Delta, A \wedge B}}{\Gamma \longrightarrow \Delta} \quad \frac{A, \Gamma \xrightarrow{\vdots} \Delta}{A \wedge B, \Gamma \longrightarrow \Delta}}{\Gamma \longrightarrow \Delta} \quad \Longrightarrow \quad \frac{\frac{\vdots}{\Gamma \longrightarrow \Delta, A} \quad A, \Gamma \xrightarrow{\vdots} \Delta}{\Gamma \longrightarrow \Delta}$$

Some consequences of cut elimination

- Syntactic consistency: It is not the case that both $\vdash A$ and $A \vdash \perp$.
- *Subformula property*.
- *Separation property*: If a formula P is provable, then it is provable using only introduction rules for the connectives in P .
- A *fragment* of a logic is defined by the set of allowed connectives. Separation property means that cut-elimination holds for any fragment of the logic.

Sequent calculus for intuitionistic logic

Sequent calculus for intuitionistic logic is obtained from the classical one by restricting to sequents with single conclusion, i.e.,

$$\Gamma \longrightarrow C.$$

As a consequence of cut-elimination and this structural restriction we have

- *Disjunction property*: If $\cdot \longrightarrow A \vee B$ is provable then $\cdot \longrightarrow A$ is provable or $\cdot \longrightarrow B$ is provable.
- *Existential property*: If $\cdot \longrightarrow \exists x.B$ is provable then there exists t such that $\cdot \longrightarrow B[t/x]$ is provable.

Double-negation embedding of classical logic

Gödel and Gentzen (1933) independently discovered that one can interpret classical validity in intuitionistic validity, via a *double-negation translation*. A formula A is translated into another formula A^N , by replacing the atomic propositions p with $\neg\neg p$, and replacing $\dots \vee \dots$ with $\neg(\neg\dots \wedge \neg\dots)$, $\exists x\dots$ with $\neg\neg\forall x.\neg\dots$

For example, $A \vee B$ is translated to $\neg(\neg A \wedge \neg B)$, if A and B are atomic propositions.

So while a classicists would say something like:

$P = NP$ or $P \neq NP$ is true

an intuitionist would say

It is not the case that both $P = NP$ and $P \neq NP$ are false.

The classicist would of course say that both statements are equivalent.

Curry-Howard correspondence for sequent calculus

Curry-Howard correspondence for sequent calculus is less obvious than natural deduction, although it is still possible to obtain. Proofs in sequent calculus have more “bureaucracy” than natural deduction, e.g., the two proofs

$$\frac{\frac{\frac{\vdots}{\Gamma, C \vdash A}}{\Gamma, C \vdash A \vee B}}{\Gamma, C \wedge D \vdash A \vee B} \qquad \frac{\frac{\frac{\vdots}{\Gamma, C \vdash A}}{\Gamma, C \wedge D \vdash A}}{\Gamma, C \wedge D \vdash A \vee B}$$

are distinguished in sequent calculus, although they map to the same proof in natural deduction:

$$\Gamma \frac{\frac{C \wedge D}{C}}{\vdots} A \vee B$$

It is possible to restrict sequent calculus into a certain form that allows precise correspondence with variants of λ -calculus.

Proof normalisation as computation

- As we have seen, normalisation of proofs correspond closely to models of computation.
- For intuitionistic logic, it is the simply typed λ -calculus.
- It is possible to extend this correspondence to a richer logic, for example, second-order intuitionistic logic corresponds to polymorphic λ -calculus.
- Curry-Howard correspondence has been further developed into type theory, e.g., Martin-Löf's *intuitionistic type theory*. The first-order version is also known as *dependent type theory*, where one can express a type which is parametric on an object of another type.
- Advances in type theory have trickled down to the design of programming languages, such as ML, and proof assistants, such as Coq.

Proof search as computation

- A different interpretation of the computational content of constructive logics arises from viewing sequents as encoding states of computation.
- This interpretation considers sequents of the form $\Gamma \longrightarrow G$ where Γ is a multiset of *program clauses* (a particular class of formulas) and G as the goal of computation. A successful computation yields the answer substitutions for the variables in G .
- The Prolog language can be seen in light of this interpretation. Here Γ is just a collection of Horn-clauses, and G is a Prolog goal (i.e., formulas constructed from conjunction and disjunction).

Encoding computation states as sequents

- In simple languages like Prolog, the states are encoded in the variables of the goal.
- By moving to a richer subset of (intuitionistic) logic, we can capture more refined notions of computation. For example by allowing implication in goal formula, we can encode the notion of modules:

$$\frac{\Gamma, D \longrightarrow G}{\Gamma \longrightarrow D \supset G} \supset R$$

Here the premise of the rule contains a “new” program D , which was not there before.

Sequents with explicit signatures

- In using proof search in sequent calculus to encode computation, we need a rich term structure to encode various syntactical constructs in programming languages.
- We shall use the simply typed λ -calculus as our term language.
- We consider an extended form of sequents

$$\Sigma : \Gamma \longrightarrow \Delta.$$

Here Σ is the *signature* for the sequent. It serves as a typing context for the constants and free variables in Γ and Δ .

- Formulas are simply typed terms. Logical connectives are typed constants, e.g.,

$$\begin{array}{ll} \perp : o & \top : o \\ \vee : o \rightarrow o \rightarrow o & \wedge : o \rightarrow o \rightarrow o \\ \exists_{\tau} : (\tau \rightarrow o) \rightarrow o & \forall_{\tau} : (\tau \rightarrow o) \rightarrow o \end{array}$$

Some sequent rules

$$\frac{\Sigma \vdash t : \tau \quad \Sigma : \Gamma, B[t/x] \longrightarrow C}{\Sigma : \Gamma, \forall_{\tau} x. B \longrightarrow C} \forall_L \quad \frac{y : \tau, \Sigma : \Gamma \longrightarrow B[y/x]}{\Sigma : \Gamma \longrightarrow \forall_{\tau} x. B} \forall_{R, y \notin \Sigma}$$
$$\frac{y : \tau, \Sigma : \Gamma, B[y/x] \longrightarrow C}{\Sigma : \Gamma, \exists_{\tau} x. B \longrightarrow C} \exists_{L, y \notin \Sigma} \quad \frac{\Sigma \vdash t : \tau \quad \Sigma : \Gamma \longrightarrow B[t/x]}{\Sigma : \Gamma \longrightarrow \exists_{\tau} x. B} \exists_R$$

Note that we do not assume all types are inhabited. In particular,

$$\Sigma : . \longrightarrow \exists_{\tau} x. \top$$

is not provable if we cannot construct a simply typed term of type τ , from the context Σ .

Summary

- We briefly reviewed the history of constructive logics in mathematics, and the motivations for studying them from a computer science perspective.
- We have concentrated on the proof theory aspects of constructive logics (as opposed to model theory), with focus on proofs and proof constructions, as opposed to validity.
- We have looked at the proof theory of intuitionistic logic and discussed briefly its computational interpretation.
- We will look at the proof theory of linear logic and the proof-search-as-computation paradigm in subsequent lectures.