12 Constructive Logic

Constructive logic codifies the principles of mathematical reasoning as it is actually practiced. In mathematics a proposition is judged true exactly when it has a proof, and is judged false exactly when it has a refutation. Because there are, and always will be, unsolved problems, we cannot expect in general that a proposition is either true or false, for in most cases, we have neither a proof nor a refutation of it. Constructive logic can be described as *logic as if people matter*, as distinct from classical logic, which can be described as *the logic of the mind of god*.

From a constructive viewpoint, a proposition is true when it has a proof. What is a proof is a social construct, an agreement among people as to what is a valid argument. The rules of logic codify a set of principles of reasoning that may be used in a valid proof. The valid forms of proof are determined by the outermost structure of the proposition whose truth is asserted. For example, a proof of a conjunction consists of a proof of each conjunct, and a proof of an implication transforms a proof of its antecedent to a proof of its consequent. When spelled out in full, the forms of proof are seen to correspond exactly to the forms of expression of a programming language. To each proposition is associated the type of its proofs; a proof is then an expression of the associated type. This association between programs and proofs induces a dynamics on proofs. In this way, proofs in constructive logic have *computational content*, which is to say that they are interpreted as executable programs of the associated type. Conversely, programs have *mathematical content* as proofs of the proposition associated to their type.

The unification of logic and programming is called the *propositions as types* principle. It is a central organizing principle of the theory of programming languages. Propositions are identified with types, and proofs are identified with programs. A programming technique corresponds to a method of proof; a proof technique corresponds to a method of programming. Viewing types as behavioral specifications of programs, propositions are problem statements whose proofs are solutions that implement the specification.

12.1 Constructive Semantics

Constructive logic is concerned with two judgments, namely ϕ prop, stating that ϕ expresses a proposition, and ϕ true, stating that ϕ is a true proposition. What distinguishes constructive from non-constructive logic is that a proposition is not conceived of as merely a truth value, but instead as a *problem statement* whose solution, if it has one, is given by a proof. A

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

proposition is *true* exactly when it has a proof, in keeping with ordinary mathematical practice. In practice, there is no other criterion of truth than the existence of a proof.

Identifying truth with proof has important, and possibly surprising, consequences. The most important consequence is that we cannot say, in general, that a proposition is either true or false. If for a proposition to be true means to have a proof of it, what does it mean for a proposition to be false? It means that we have a *refutation* of it, showing that it cannot be proved. That is, a proposition is false if we can show that the assumption that it is true (has a proof) contradicts known facts. In this sense, constructive logic is a logic of *positive*, or *affirmative*, *information*—we must have explicit evidence in the form of a proof to affirm the truth or falsity of a proposition.

In light of this, it is clear that not every proposition is either true or false. For if ϕ expresses an unsolved problem, such as the famous $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ problem, then we have neither a proof nor a refutation of it (the mere absence of a proof not being a refutation). Such a problem is *undecided*, precisely because it has not been solved. Because there will always be unsolved problems (there being infinitely many propositions, but only finitely many proofs at a given point in time), we cannot say that every proposition is *decidable*, that is, either true or false.

Of course, some propositions are decidable, and hence are either true or false. For example, if ϕ expresses an inequality between natural numbers, then ϕ is decidable, because we can always work out, for given natural numbers m and n, whether $m \leq n$ or $m \not\leq n$—we can either prove or refute the given inequality. This argument does not extend to the real numbers. To get an idea of why not, consider the representation of a real number by its decimal expansion. At any finite time, we will have explored only a finite initial segment of the expansion, which is not enough to decide if it is, say, less than 1. For if we have calculated the expansion to be $0.99\ldots9$, we cannot decide at any time, short of infinity, whether or not the number is 1.

The constructive attitude is simply to accept the situation as inevitable, and make our peace with that. When faced with a problem, we have no choice but to roll up our sleeves and try to prove it or refute it. There is no guarantee of success! Life is hard, but we muddle through somehow.

12.2 Constructive Logic

The judgments ϕ prop and ϕ true of constructive logic are rarely of interest by themselves, but rather in the context of a hypothetical judgment of the form

$$\phi_1 \text{ true}, \ldots, \phi_n \text{ true} \vdash \phi \text{ true}.$$

This judgment says that the proposition ϕ is true (has a proof), under the assumptions that each of $\phi_1, \ldots, \phi_n$ are also true (have proofs). Of course, when $n = 0$ this is just the same as the judgment ϕ true.

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

The structural properties of the hypothetical judgment, when specialized to constructive logic, define what we mean by reasoning under hypotheses:

$$\frac{}{\Gamma, \phi \text{ true} \vdash \phi \text{ true}} \tag{12.1a}$$

$$\frac{\Gamma \vdash \phi_1 \text{ true} \quad \Gamma, \phi_1 \text{ true} \vdash \phi_2 \text{ true}}{\Gamma \vdash \phi_2 \text{ true}} \tag{12.1b}$$

$$\frac{\Gamma \vdash \phi_2 \text{ true}}{\Gamma, \phi_1 \text{ true} \vdash \phi_2 \text{ true}} \tag{12.1c}$$

$$\frac{\Gamma, \phi_1 \text{ true}, \phi_1 \text{ true} \vdash \phi_2 \text{ true}}{\Gamma, \phi_1 \text{ true} \vdash \phi_2 \text{ true}} \tag{12.1d}$$

$$\frac{\Gamma_1, \phi_2 \text{ true}, \phi_1 \text{ true}, \Gamma_2 \vdash \phi \text{ true}}{\Gamma_1, \phi_1 \text{ true}, \phi_2 \text{ true}, \Gamma_2 \vdash \phi \text{ true}} \tag{12.1e}$$

The last two rules are implicit in that we regard Γ as a set of hypotheses, so that two "copies" are as good as one, and the order of hypotheses does not matter.

12.2.1 Provability

The syntax of propositional logic is given by the following grammar:

Prop	ϕ	::=	$\top$	$\top$	truth
			$\bot$	$\bot$	falsity
			$\wedge(\phi_1;\phi_2)$	$\phi_1 \wedge \phi_2$	conjunction
			$\vee(\phi_1;\phi_2)$	$\phi_1 \vee \phi_2$	disjunction
			$\supset(\phi_1;\phi_2)$	$\phi_1 \supset \phi_2$	implication

The connectives of propositional logic are given meaning by rules that define (a) what constitutes a "direct" proof of a proposition formed from that connective, and (b) how to exploit the existence of such a proof in an "indirect" proof of another proposition. These are called the *introduction* and *elimination* rules for the connective. The principle of *conservation of proof* states that these rules are inverse to one another—the elimination rule cannot extract more information (in the form of a proof) than was put into it by the introduction rule, and the introduction rules can reconstruct a proof from the information extracted by the elimination rules.

Truth Our first proposition is trivially true. No information goes into proving it, and so no information can be obtained from it.

$$\frac{}{\Gamma \vdash \top \text{ true}} \tag{12.2a}$$

(no elimination rule)

(12.2b)

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

Conjunction Conjunction expresses the truth of both of its conjuncts.

$$\frac{\Gamma \vdash \phi_1 \ \mathsf{true} \quad \Gamma \vdash \phi_2 \ \mathsf{true}}{\Gamma \vdash \phi_1 \wedge \phi_2 \ \mathsf{true}} \tag{12.3a}$$

$$\frac{\Gamma \vdash \phi_1 \wedge \phi_2 \ \mathsf{true}}{\Gamma \vdash \phi_1 \ \mathsf{true}} \tag{12.3b}$$

$$\frac{\Gamma \vdash \phi_1 \wedge \phi_2 \ \mathsf{true}}{\Gamma \vdash \phi_2 \ \mathsf{true}} \tag{12.3c}$$

Implication Implication expresses the truth of a proposition under an assumption.

$$\frac{\Gamma, \phi_1 \ \mathsf{true} \vdash \phi_2 \ \mathsf{true}}{\Gamma \vdash \phi_1 \supset \phi_2 \ \mathsf{true}} \tag{12.4a}$$

$$\frac{\Gamma \vdash \phi_1 \supset \phi_2 \ \mathsf{true} \quad \Gamma \vdash \phi_1 \ \mathsf{true}}{\Gamma \vdash \phi_2 \ \mathsf{true}} \tag{12.4b}$$

Falsehood Falsehood expresses the trivially false (refutable) proposition.

(no introduction rule)

(12.5a)

$$\frac{\Gamma \vdash \bot \ \mathsf{true}}{\Gamma \vdash \phi \ \mathsf{true}} \tag{12.5b}$$

Disjunction Disjunction expresses the truth of either (or both) of two propositions.

$$\frac{\Gamma \vdash \phi_1 \ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2 \ \mathsf{true}} \tag{12.6a}$$

$$\frac{\Gamma \vdash \phi_2 \ \mathsf{true}}{\Gamma \vdash \phi_1 \vee \phi_2 \ \mathsf{true}} \tag{12.6b}$$

$$\frac{\Gamma \vdash \phi_1 \vee \phi_2 \ \mathsf{true} \quad \Gamma, \phi_1 \ \mathsf{true} \vdash \phi \ \mathsf{true} \quad \Gamma, \phi_2 \ \mathsf{true} \vdash \phi \ \mathsf{true}}{\Gamma \vdash \phi \ \mathsf{true}} \tag{12.6c}$$

Negation The negation, $\neg\phi$, of a proposition ϕ is defined as the implication $\phi \supset \bot$. As a result, $\neg\phi \ \mathsf{true}$ if $\phi \ \mathsf{true} \vdash \bot \ \mathsf{true}$, which is to say that the truth of ϕ is *refutable* in that we may derive a proof of falsehood from any purported proof of ϕ. Because constructive truth is defined to be the existence of a proof, the implied semantics of negation is rather strong. In particular, a problem ϕ is *open* exactly when we can neither affirm nor refute it. In contrast, the classical conception of truth assigns a fixed truth value to each proposition so that every proposition is either true or false.

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

12.2.2 Proof Terms

The key to the propositions-as-types principle is to make explicit the forms of proof. The basic judgment ϕ true, which states that ϕ has a proof, is replaced by the judgment $p : \phi$, stating that p is a proof of ϕ. (Sometimes p is called a "proof term," but we will simply call p a "proof.") The hypothetical judgment is modified correspondingly, with variables standing for the presumed, but unknown, proofs:

$$x_1 : \phi_1, \ldots, x_n : \phi_n \vdash p : \phi.$$

We again let Γ range over such hypothesis lists, subject to the restriction that no variable occurs more than once.

The syntax of proof terms is given by the following grammar:

Prf	p	::=	$\texttt{true-I}$	$\langle\rangle$	truth intro
			$\texttt{and-I}(p_1; p_2)$	$\langle p_1, p_2\rangle$	conj. intro
			$\texttt{and-E[l]}(p)$	$p \cdot \texttt{l}$	conj. elim
			$\texttt{and-E[r]}(p)$	$p \cdot \texttt{r}$	conj. elim
			$\texttt{imp-I}(x.p)$	$\lambda\,(x)\,p$	impl. intro
			$\texttt{imp-E}(p_1; p_2)$	$p_1(p_2)$	impl. elim
			$\texttt{false-E}(p)$	$\texttt{abort}(p)$	false elim
			$\texttt{or-I[l]}(p)$	$\texttt{l} \cdot p$	disj. intro
			$\texttt{or-I[r]}(p)$	$\texttt{r} \cdot p$	disj. intro
			$\texttt{or-E}(p; x_1.p_1; x_2.p_2)$	$\texttt{case}\ p\ \{\texttt{l} \cdot x_1 \hookrightarrow p_1 \mid \texttt{r} \cdot x_2 \hookrightarrow p_2\}$	disj. elim

The concrete syntax of proof terms is chosen to stress the correspondence between propositions and types discussed in Section 12.4 below.

The rules of constructive propositional logic can be restated using proof terms as follows.

$$\frac{}{\Gamma \vdash \langle\rangle : \top} \tag{12.7a}$$

$$\frac{\Gamma \vdash p_1 : \phi_1 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash \langle p_1, p_2\rangle : \phi_1 \wedge \phi_2} \tag{12.7b}$$

$$\frac{\Gamma \vdash p_1 : \phi_1 \wedge \phi_2}{\Gamma \vdash p_1 \cdot \texttt{l} : \phi_1} \tag{12.7c}$$

$$\frac{\Gamma \vdash p_1 : \phi_1 \wedge \phi_2}{\Gamma \vdash p_1 \cdot \texttt{r} : \phi_2} \tag{12.7d}$$

$$\frac{\Gamma, x : \phi_1 \vdash p_2 : \phi_2}{\Gamma \vdash \lambda\,(x)\,p_2 : \phi_1 \supset \phi_2} \tag{12.7e}$$

$$\frac{\Gamma \vdash p : \phi_1 \supset \phi_2 \quad \Gamma \vdash p_1 : \phi_1}{\Gamma \vdash p(p_1) : \phi_2} \tag{12.7f}$$

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

$$\frac{\Gamma \vdash p : \bot}{\Gamma \vdash \mathtt{abort}(p) : \phi} \tag{12.7g}$$

$$\frac{\Gamma \vdash p_1 : \phi_1}{\Gamma \vdash \mathtt{l} \cdot p_1 : \phi_1 \vee \phi_2} \tag{12.7h}$$

$$\frac{\Gamma \vdash p_2 : \phi_2}{\Gamma \vdash \mathtt{r} \cdot p_2 : \phi_1 \vee \phi_2} \tag{12.7i}$$

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2 \quad \Gamma, x_1 : \phi_1 \vdash p_1 : \phi \quad \Gamma, x_2 : \phi_2 \vdash p_2 : \phi}{\Gamma \vdash \mathtt{case}\, p\, \{\mathtt{l} \cdot x_1 \hookrightarrow p_1 \mid \mathtt{r} \cdot x_2 \hookrightarrow p_2\} : \phi} \tag{12.7j}$$

12.3 Proof Dynamics

Proof terms in constructive logic are given a dynamics by *Gentzen's Principle*. It states that the elimination forms are inverse to the introduction forms. One aspect of Gentzen's Principle is the principle of *conservation of proof*, which states that the information introduced into a proof of a proposition can be extracted without loss by elimination. For example, we may state that conjunction elimination is post-inverse to conjunction introduction by the definitional equations:

$$\frac{\Gamma \vdash p_1 : \phi_1 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash \langle p_1, p_2 \rangle \cdot \mathtt{l} \equiv p_1 : \phi_1} \tag{12.8a}$$

$$\frac{\Gamma \vdash p_1 : \phi_1 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash \langle p_1, p_2 \rangle \cdot \mathtt{r} \equiv p_2 : \phi_2} \tag{12.8b}$$

Another aspect of Gentzen's Principle is that principle of *reversibility of proof*, which states that every proof can be reconstructed from the information that can be extracted from it by elimination. In the case of conjunction this can be stated by the definitional equation

$$\frac{\Gamma \vdash p_1 : \phi_1 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash \langle p \cdot \mathtt{l}, p \cdot \mathtt{r} \rangle \equiv p : \phi_1 \wedge \phi_2} \tag{12.9}$$

Similar equivalences can be stated for the other connectives. For example, the conservation and reversibility principles for implication are given by these rules:

$$\frac{\Gamma, x : \phi_1 \vdash p_2 : \phi_2 \quad \Gamma \vdash p_2 : \phi_2}{\Gamma \vdash (\lambda\,(x)\, p_2)(p_1) \equiv [p_1/x]p_2 : \phi_2} \tag{12.10a}$$

$$\frac{\Gamma \vdash p : \phi_1 \supset \phi_2}{\Gamma \vdash \lambda\,(x)\,(p(x)) \equiv p : \phi_1 \supset \phi_2} \tag{12.10b}$$

The corresponding rules for disjunction and falsehood are given as follows:

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2 \quad \Gamma, x_1 : \phi_1 \vdash p_1 : \psi \quad \Gamma, x_2 : \phi_2 \vdash p_2 : \psi}{\Gamma \vdash \mathtt{case}\, \mathtt{l} \cdot p\, \{\mathtt{l} \cdot x_1 \hookrightarrow p_1 \mid \mathtt{r} \cdot x_2 \hookrightarrow p_2\} \equiv [p/x_1]p_1 : \psi} \tag{12.11a}$$

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2 \quad \Gamma, x_1 : \phi_1 \vdash p_1 : \psi \quad \Gamma, x_2 : \phi_2 \vdash p_2 : \psi}{\Gamma \vdash \mathtt{case}\, \mathtt{r} \cdot p\, \{\mathtt{l} \cdot x_1 \hookrightarrow p_1 \mid \mathtt{r} \cdot x_2 \hookrightarrow p_2\} \equiv [p/x_2]p_2 : \psi} \tag{12.11b}$$

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

$$\frac{\Gamma \vdash p : \phi_1 \vee \phi_2 \quad \Gamma, x : \phi_1 \vee \phi_2 \vdash q : \psi}{\Gamma \vdash [p/x]q \equiv \mathtt{case}\ p\ \{\mathtt{l} \cdot x_1 \hookrightarrow [\mathtt{l} \cdot x_1/x]q \mid \mathtt{r} \cdot x_2 \hookrightarrow [\mathtt{r} \cdot x_2/x]q\} : \psi} \tag{12.11c}$$

$$\frac{\Gamma \vdash p : \bot \quad \Gamma, x : \bot \vdash q : \psi}{\Gamma \vdash [p/x]q \equiv \mathtt{abort}(p) : \psi} \tag{12.11d}$$

12.4 Propositions as Types

Reviewing the statics and dynamics of proofs in constructive logic reveals a striking similarity to the statics and dynamics of expressions of various types. For example, the introduction rule for conjunction specifies that a proof of a conjunction consists of a pair of proofs, one for each conjunct, and the elimination rule inverts this, allowing us to extract a proof of each conjunct from any proof of a conjunction. There is an obvious analogy with the static semantics of product types, whose introduction form is a pair and whose elimination forms are projections. Gentzen's Principle extends the analogy to the dynamics as well, so that the elimination forms for conjunction amount to projections that extract the appropriate components from an ordered pair.

The following chart summarizes the correspondence between propositions and types and between proofs and programs:

Prop	Type
$\top$	`unit`
$\bot$	`void`
$\phi_1 \wedge \phi_2$	$\tau_1 \times \tau_2$
$\phi_1 \supset \phi_2$	$\tau_1 \to \tau_2$
$\phi_1 \vee \phi_2$	$\tau_1 + \tau_2$

The correspondence between propositions and types is a cornerstone of the theory of programming languages. It exposes a deep connection between computation and deduction, and serves as a framework for the analysis of language constructs and reasoning principles by relating them to one another.

12.5 Notes

The propositions as types principle has its origins in the semantics of intuitionistic logic developed by Brouwer, according to which the truth of a proposition is witnessed by a construction providing computable evidence for it. The forms of evidence are determined by the form of the proposition, so that evidence for an implication is a computable function transforming evidence for the hypothesis into evidence for the conclusion. An explicit formulation of this semantics was introduced by Heyting, and further developed by several people, including de Bruijn, Curry, Gentzen, Girard, Howard, Kolmogorov, Martin-Löf, and Tait. The propositions-as-types correspondence is sometimes called the *Curry-Howard*

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

Isomorphism, but this terminology neglects the crucial contributions of the others just mentioned. Moreover, the correspondence is not, in general, an isomorphism; rather, it expresses Brouwer's Dictum that the concept of proof is best explained by the more general concept of construction (program).

Exercises

12.1. The *law of the excluded middle (LEM)* is the statement that every proposition ϕ is *decidable* in the sense that $\phi \vee \neg\phi$ true. Constructively, the law of the excluded middle states that, for every proposition ϕ, we either have a proof of ϕ or a refutation of ϕ (proof of its negation). Because this is manifestly not the case in general, one may suspect that the law of the excluded middle is not constructively valid. This is so, but not in the sense that the law is *refuted*, but rather in the sense that it is *not affirmed*. First, any proposition ϕ for which we have a proof or a refutation is already decided, and so is decidable. Second, there are broad classes of propositions for which we can, on demand, produce a proof or a refutation. For example, it is decidable whether or not two integers are equal. Third, and most important, there are, and always will be, propositions ϕ whose status is unresolved: it may turn out that ϕ is true, or it may turn out that ϕ is false. For all these reasons, constructive logic *does not refute* the decidability propositions: $\neg\neg(\phi \vee \neg\phi)$ true for any proposition ϕ. Prove it using the rules given in this chapter.

12.2. The proposition $\neg\neg\phi$ is no stronger than ϕ: prove $\phi \supset \neg\neg\phi$ true. The *law of double-negation elimination (DNE)* states that $(\neg\neg\phi) \supset \phi$ true for every proposition ϕ. It follows immediately from Exercise **12.1** that DNE entails LEM; prove the converse.

12.3. Define the relation $\phi \leq \psi$ to mean that ϕ true $\vdash \psi$ true according to the rules of constructive logic given above. With respect to this relation, show the following facts:

(a) It is a *pre-order*, which is say that it is reflexive and transitive.

(b) $\phi \wedge \psi$ is the *meet*, or *greatest lower bound*, of ϕ and ψ, and $\top$ is the *top*, or *greatest*, element.

(c) Show that $\phi \vee \psi$ is the *join*, or *least upper bound*, of ϕ and ψ, and that $\bot$ is the *bottom*, or *least*, element.

(d) Show that $\phi \supset \psi$ is an *exponential*, or *pseudo-complement*, in the sense that it is the *largest* ρ such that $\phi \wedge \rho \leq \psi$. (The exponential $\phi \supset \psi$ is sometimes written ψ^{ϕ}.)

Altogether these facts state that entailment in constructive propositional logic forms a *Heyting algebra*. Show that a general Heyting algebra (that is, an ordering with the above structure) is *distributive* in the sense that

$$\phi \wedge (\psi_1 \vee \psi_2) \equiv (\phi \wedge \psi_1) \vee (\phi \wedge \psi_2)$$
$$\phi \vee (\psi_1 \wedge \psi_2) \equiv (\phi \vee \psi_1) \wedge (\phi \vee \psi_2),$$

where $\phi \equiv \psi$ means $\phi \leq \psi$ and $\psi \leq \phi$.

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press

12.4. In any Heyting algebra, we have $\phi \wedge \neg\phi \leq \bot$, which is to say that the negation is inconsistent with the negated. But $\neg\phi$ is not necessarily the *complement* of ϕ in the sense that $\phi \vee \neg\phi \leq \top$. A *Boolean algebra* is a Heyting algebra in which negation is always the complement of the negated: $\top \leq \phi \vee \neg\phi$ for every ϕ. Check that the two-element Boolean algebra for which meets, joins, and exponentials are given by the classical truth tables (defining $\phi \supset \psi$ as $(\neg\phi) \vee \psi$ is Boolean algebra. Conclude that it is consistent to adjoin LEM to constructive logic, which is to say that classical logic is a special case of constructive logic in which we assume that every proposition is decidable. Being a Heyting algebra, every Boolean algebra is clearly distributive. Show that every Boolean algebra also satisfies the *de Morgan duality laws*:

$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$
$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi.$$

The first of these is valid in any Heyting algebra; the second only in a Boolean algebra.

https://doi.org/10.1017/CBO9781316576892.014 Published online by Cambridge University Press