Exercise 3.5.22 *(1) Show that the logical approximation theorem still holds, replacing the type system $\mathcal{D}\Omega$ by the type system of figure 3.2. (Warning: this involves revisiting a number of syntactic lemmas, typically lemma 3.5.9.) (2) Show the approximation theorem for the D_∞ model based on $D_0 = \{\bot, \top\}$ and the standard pair (i_0, j_0), i.e., $[\![M]\!] = \bigvee\{[\![A]\!] \mid A \leq BT(M)\}$ (cf. exercise 3.3.14 and theorem 3.4.8).*

4

Interpretation of λ-calculi in CCC's

In first approximation, typed λ-calculi are *natural deduction* presentations of certain fragments of minimal logic (a subsystem of intuitionistic logic). These calculi have a natural computational interpretation as core of typed functional languages where computation, intended as $\beta\eta$-reduction, corresponds to proof normalization. In this perspective, we reconsider in section 4.1 the simply typed λ-calculus studied in chapter 2. We exhibit a precise correspondence between the simply typed λ-calculus and a natural deduction formalization of the implicative fragment of propositional implicative logic.

Next, we address the problem of *modelling* the notions of $\beta\eta$-reduction and equivalence. It turns out that simple models can be found by interpreting types as sets and terms as functions between these sets. But, in general, which are the structural properties that characterize such models? The main problem considered in this chapter is that of understanding what is the *model theory* of simply typed and untyped λ-calculi. In order to answer this question, we introduce in section 4.2 the notion of cartesian closed category (CCC). We present CCC's as a natural categorical generalization of certain adjunctions found in Heyting algebras. As a main example, we show that the category of directed complete partial orders and continuous functions is a CCC.

The description of the models of a calculus by means of category theoretical notions will be a central and recurring topic of this book. We will not always fully develop the theory but in this chapter we can take advantage of the simplicity of the calculus to go into a complete analysis. In section 4.3, we describe the interpretation of the simply typed λ-calculus into an arbitrary CCC, and we present some basic properties such as the substitution theorem. The interpretation into a categorical language can be seen as a way of implementing α-renaming and substitution. This eventually leads to the definition of a *categorical abstract machine*.

In section 4.4, we address the problem of understanding which equivalence is induced on terms by the interpretation in a CCC. To this end, we introduce the notion of λ-theory. Roughly speaking, a λ-theory is a congruence over λ-terms (i.e., an equivalence relation closed under λ-abstraction and application) which includes $\beta\eta$-equivalence. It turns out that every CCC induces a λ-theory. Vice versa, one may ask: does any λ-theory come from the interpretation in a CCC? We answer this question positively by showing how to build a suitable CCC from

any λ-theory. This concludes our development of a *model theory* for the simply typed λ-calculus. Related results will be presented in chapter 6 for PCF, a simply typed λ-calculus extended with arithmetical operators and fixpoint combinators.

In section 4.5 we introduce *logical relations* which are a useful tool to establish links between syntax and semantics. In particular, we apply them to the problem of characterizing equality in the set theoretical model of the simply typed λ-calculus, and to the problem of understanding which elements of a model are definable by a λ-term.

In section 4.6 we regard the untyped λ-calculus as a typed λ-calculus with a *reflexive type*. We show that every CCC with a reflexive object gives rise to an untyped λ-theory. We present a general method to build a category of retractions out of a reflexive object in a CCC. We give two applications of this construction. First, we hint at the fact that every untyped λ-theory is induced by a reflexive object in a CCC (this is similar to the result presented in section 4.4 for the simply typed λ-calculus). Second, following Engeler, we adopt the category of retractions as a frame for embedding algebraic structures in λ-models.

This chapter is mainly based on [LS86, Sco80, Cur86] to which the reader seeking more advanced results is directed.

4.1 Simply typed λ-calculus

In chapter 2, we have presented a simply typed λ-calculus in which every subterm is labelled by a type. This was well-suited to our purposes but it is probably not the most illuminating treatment. So far, we have (mainly) discussed the λ-calculus as a core formalism to represent functions-as-algorithms. The simply typed λ-calculus receives an additional interpretation: it is a language of proofs for minimal logic. Let us revisit simple types first, by considering basic types as atomic propositions and the function space symbol as implication:

$$\begin{array}{lll} At & ::= & \kappa \mid \kappa' \mid \cdots \\ \sigma & ::= & At \mid (\sigma \to \sigma)\ . \end{array}$$

Forgetting the terms for a while, we briefly describe the provability of formulas for this rudimentary logic. We use a deduction style called *natural deduction* [Pra65]. A formula σ is proved relative to a list $\sigma_1, \ldots, \sigma_n$ of assumptions. The formal system described in figure 4.1 allows us to derive judgments of the form $\sigma_1, \ldots, \sigma_n \vdash \sigma$, which are called sequents.

An important remark with a wide range of possible applications [How80] is that proofs in natural deduction can be encoded precisely as λ-terms. To this aim hypotheses are named by variables. Raw terms are defined by the following syntax (in the following, we feel free to be sparing with parentheses):

$$\begin{array}{lll} v & ::= & x \mid y \mid \ldots \\ M & ::= & v \mid \lambda v : \sigma.M \mid MM\ . \end{array}$$

A *context* Γ is a list of pairs, $x : \sigma$, where x is a variable and σ is a type, and where all variables are distinct. We write $x : \sigma \in \Gamma$ to express that the pair $x : \sigma$ occurs in Γ. A judgment has the shape $\Gamma \vdash M : \sigma$. Whenever we write $\Gamma \vdash M : \sigma$ it is intended that the judgment is provable. We also write $M : \sigma$ to say that there exists a context Γ such that $\Gamma \vdash M : \sigma$. A term M with this property is called well-typed. Provable judgments are inductively defined in figure 4.2. We may omit the labels on the λ-abstractions when the types are obvious from the context. It is easily seen that any derivable judgment admits a unique derivation, thus yielding a one-to-one correspondence between proofs and terms.

Yet another presentation of the typing rules omits all type information in the λ-terms. The corresponding typing system is obtained from the one in figure 4.2 by removing the type σ in $\lambda x : \sigma.M$. In this case a term in a given context can be given several types. For instance the term $\lambda x.x$ can be assigned in the empty context any type $\sigma \to \sigma$, for any σ. To summarize, we have considered three styles of typing:

(1) A totally explicit typing where every subterm is labelled by a type (see section 2.2).

(2) A more economic typing, where only the variables bound in abstractions are labelled by a type. This style is known as 'typing à la Church'.

(3) A type assignment system, where an untyped term receives a type. This is known as 'typing à la Curry'.

In the first system, the term itself carries all the typing information. We note that once we have labelled free variables and λ-abstractions, the label of each subterm can be reconstructed in a unique way. In the two other systems, à la Church and à la Curry, a separate context carries type information for the free variables. In the system à la Church, the context together with the types of bound

$$\frac{1 \le i \le n}{\sigma_1, \ldots, \sigma_n \vdash \sigma_i} \qquad \frac{\sigma_1, \ldots, \sigma_n, \sigma \vdash \tau}{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau} \qquad \frac{\sigma_1, \ldots, \sigma_n \vdash \sigma \to \tau \quad \sigma_1, \ldots, \sigma_n \vdash \sigma}{\sigma_1, \ldots, \sigma_n \vdash \tau}$$

Figure 4.1: Natural deduction for minimal implicative logic

$$\text{(Asmp)}\ \frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma} \qquad (\to_I)\ \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau} \qquad (\to_E)\ \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

Figure 4.2: Typing rules for the simply typed λ-calculus

variables carry all the necessary information to reconstruct uniquely the type of the term. In the system à la Curry, a term, even in a given context, may have many types. In general, the problem of deciding if an untyped λ-term has a type in a given context is a non-trivial one. This is referred to as the *type inference* or *type reconstruction* problem.

Type reconstruction algorithms are quite relevant in practice as they relieve the programmer from the burden of explicitly writing all type information and allow for some form of polymorphism. For the simply typed discipline presented here, it can be shown that the problem is decidable and that it is possible to represent by a type schema (a type with type variables) all derivable solutions to a given type reconstruction problem [Hin69]. On the other hand, the type inference problem turns out to be undecidable in certain relevant type disciplines (e.g., second order [Wel94]).

In this chapter, we concentrate on the interpretation of λ-terms with *explicit* type information. We regard these calculi à la Church as central, by virtue of their strong ties with category theory and proof theory. The interpretation of type assignment systems has already been mentioned in chapter 3, and it will be further developed in section 15.3.

Exercise 4.1.1 *Let M^σ be a totally explicitly typed term. Let $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ be its free variables. Let erase be the function that erases all type information in a λ-term. Show that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash erase(M) : \sigma$ is à la Curry derivable. Define a function semi-erase such that $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash semi\text{-}erase(M) : \sigma$ is à la Church derivable. Conversely, from a derivation à la Curry of $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma$, construct a totally explicitly typed term N^σ, whose free variables are $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$, and such that $erase(N^\sigma) = M$. Design a similar transformation from a derivation à la Church. Investigate how these transformations compose.*

Exercise 4.1.2 *Show that the structural rules of exchange, weakening, and contraction are derived in the system above, in the sense that, if the premises are provable, then the conclusion is provable.*

$$\begin{array}{lll} (exch) & \Gamma, x:\sigma, y:\tau, \Gamma' \vdash M:\rho & \Rightarrow \quad \Gamma, y:\tau, x:\sigma, \Gamma' \vdash M:\rho \\ (weak) & \Gamma \vdash M:\tau \text{ and } x:\sigma \notin \Gamma & \Rightarrow \quad \Gamma, x:\sigma \vdash M:\tau \\ (contr) & \Gamma, x:\sigma, y:\sigma \vdash M:\tau & \Rightarrow \quad \Gamma, z:\sigma \vdash M[z/x, z/y]:\tau \quad (z \text{ fresh}) \,. \end{array}$$

We consider two basic axioms for the reduction of terms (cf. section 2.1):

$$\begin{array}{ll} (\beta) & (\lambda x:\sigma.M)N \to M[N/x] \\ (\eta) & \lambda x:\sigma.(Mx) \to M \quad \text{if } x \notin FV(M) \,. \end{array}$$

We denote with $\to_{\beta\eta}$ their compatible (or contextual) closure (cf. figure 2.4), and with $\to^*_{\beta\eta}$ the reflexive and transitive closure of $\to_{\beta\eta}$.

Exercise 4.1.3 (subject reduction) *Show that well-typed terms are closed under reduction, formally:*

$$(\Gamma \vdash M:\sigma \text{ and } M \to_{\beta\eta} N) \quad \Rightarrow \quad \Gamma \vdash N:\sigma \,.$$

and that if M^σ and N^σ are the totally explicitly typed terms associated to M and N (cf. exercise 4.1.1) then M^σ reduces to N^σ (cf. definition 2.2.7).

Theorem 4.1.4 (confluence and normalization) (1) *The reduction relation $\to^*_{\beta\eta}$ is confluent (both on typed and untyped λ-terms).*

(2) *The reduction system $\to_{\beta\eta}$ is strongly normalizing on well-typed terms, that is, if $M : \sigma$ then all reduction sequences starting from M lead to a $\beta\eta$-normal form.*

We have already proved these properties in chapter 2 for β-reduction and the totally explicit typed variant. The results are easily adapted to the present à la Church setting (cf. exercise 4.1.3). The following exercise provides enough guidelines to extend the results to $\beta\eta$-reduction.

Exercise 4.1.5 *In the following $\to^{\leq 1}$ means reduction in 0 or 1 step. Show the following properties.*

(1) *If $M \to_\eta M_1$ and $M \to_\eta M_2$, then there exists an N such that $M_1 \to_\eta^{\leq 1} N$ and $M_2 \to_\eta^{\leq 1} N$.*

(2) *If $M \to_\eta M_1$ and $M \to_\beta M_2$, then there exists an N such that $M_1 \to_\beta^{\leq 1} N$ and $M_2 \to_\eta^* N$.*

(3) *If $M \to_\eta \cdot \to_\beta N$, then $M \to_\beta \cdot \to_\beta N$ or $M \to_\beta \cdot \to_\eta^* N$, where $M \to_{R_1} \cdot \to_{R_2} N$ stands for $\exists P \ (M \to_{R_1} P \text{ and } P \to_{R_2} N)$.*

4.2 Cartesian closed categories

The reader will find in appendix 2 some basic notions of category theory. Next, we motivate the introduction of CCC's as the combination of two more elementary concepts.

Example 4.2.1 (conjunction and binary products) *Let us consider a simple calculus in which we can pair two values or project a pair to one of its components.*

$$\begin{array}{lll} Types & At & ::= \kappa \mid \kappa' \mid \cdots \\ & \sigma & ::= At \mid (\sigma \times \sigma) \\ Terms & v & ::= x \mid y \mid \cdots \\ & M & ::= v \mid \langle M, M \rangle \mid \pi_1 M \mid \pi_2 M \,. \end{array}$$

This calculus corresponds to the conjunctive fragment of minimal logic. Its typing rules are shown in figure 4.3.

It is intuitive that a cartesian category (i.e., a category with a terminal object and binary products) has something to do with this calculus. Let us make this intuition more precise:

(1) We interpret a type σ as an object $[\![\sigma]\!]$ of a cartesian category $\mathbf{C}$. The interpretation of the type $\sigma \times \tau$ is the cartesian product $[\![\sigma]\!] \times [\![\tau]\!]$.

(2) If types are objects, it seems natural to associate terms to morphisms. If M is a closed term of type σ, we may expect that its interpretation is a morphism $f : 1 \to [\![\sigma]\!]$, where 1 is the terminal object. But what about a term M such that

$$\text{(Asmp)}\ \frac{x:\sigma\in\Gamma}{\Gamma\vdash x:\sigma} \qquad (\times_I)\ \frac{\Gamma\vdash M:\sigma\ \ \Gamma\vdash N:\tau}{\Gamma\vdash\langle M,N\rangle:\sigma\times\tau}$$

$$(\times_{E1})\ \frac{\Gamma\vdash M:\sigma\times\tau}{\Gamma\vdash\pi_1 M:\sigma} \qquad (\times_{E2})\ \frac{\Gamma\vdash M:\sigma\times\tau}{\Gamma\vdash\pi_2 M:\tau}$$

Figure 4.3: Typing rules for a calculus of conjunction

$x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash M:\sigma$? The idea is to interpret this term as a morphism $f:(\cdots(1\times[\![\sigma_1]\!])\times\cdots\times[\![\sigma_n]\!])\to[\![\sigma]\!]$.

This example suggests that types can be seen as objects and terms as morphisms. We do not wish to be more precise at the moment (but see section 3) and leave the following as an exercise.

Exercise 4.2.2 *Define an interpretation of the typed terms of the calculus of conjunction into a cartesian category.*

There is a well-known correspondence between classical propositional logic and boolean algebras: a formula is provable iff it is valid in every boolean algebra interpretation. *Heyting algebras* play a similar role for intuitionistic (or minimal) logic.

Definition 4.2.3 (Heyting algebra) *A Heyting algebra H is a lattice with lub operation $\vee$, glb operation $\wedge$, greatest element 1, least element 0, and with a binary operation $\to$ that satisfies the condition*

$$(x\wedge y)\le z \ \ \textit{iff}\ \ x\le(y\to z)\ .$$

Exercise 4.2.4 *Heyting algebras abound in nature. Show that the collection Ω of open sets of a topological space (X,Ω) ordered by inclusion can be seen as a Heyting algebra by taking:*

$$U\to V=\bigcup\{W\in\Omega\mid W\subseteq(X\backslash U)\cup V\}\ .$$

For our purposes the important point in the definition of Heyting algebra is that the implication is characterized by an adjoint situation (in a poset case, see section A2.4), as for any $y\in H$ the function $\_\wedge y$ is left adjoint to the function $y\to\_$:

$$\forall y\in H\ \ (\_\wedge y)\dashv(y\to\_)\ .$$

In poset categories the interpretation of proofs is trivial. For this reason Heyting algebras cannot be directly applied to the problem of interpreting the simply typed λ-calculus. However, combined with our previous example they suggest a natural generalization: consider a cartesian category in which each functor $\_\times A$ has a right adjoint $(\_)^A$. In this way we arrive at the notion of CCC. The adjunction condition can be reformulated in a more explicit way, as shown in the following definition.

Definition 4.2.5 (CCC) *A category* $\mathbf{C}$ *is called* cartesian closed *if it has:*

(1) *A terminal object* 1.

(2) *For each $A,B\in\mathbf{C}$ a product given by an object $A\times B$ with projections $\pi_A:A\times B\to A$ and $\pi_B:A\times B\to B$ such that:*

$$\forall C\in\mathbf{C}\,\forall f:C\to A\,\forall g:C\to B\,\exists!h:C\to A\times B\ (\pi_A\circ h=f\ \textit{and}\ \pi_B\circ h=g)\ .$$

The morphism h is often denoted by $\langle f,g\rangle$, where $\langle\_,\_\rangle$ is called the pairing operator. Other (most frequently used) notations for π_A and π_B are π_1 and π_2.

(3) *For each $A,B\in\mathbf{C}$ an* exponent *given by an object B^A with $ev:B^A\times A\to B$ such that:*

$$\forall C\in\mathbf{C}\,\forall f:C\times A\to B\,\exists!h:C\to B^A\ (ev\circ(h\times id)=f)\ .$$

The morphism h is often denoted by $\Lambda(f)$, Λ is called the currying operator, and ev the evaluation morphism.

In the following B^A and $A\Rightarrow B$ are interchangeable notations for the exponent object in a category.

Exercise 4.2.6 *Given a CCC* $\mathbf{C}$*, extend the functions $Prod(A,B)=A\times B$ and $Exp(A,B)=B^A$ to functors $Prod:\mathbf{C}\times\mathbf{C}\to\mathbf{C}$ and $Exp:\mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$.*

Exercise 4.2.7 *Show that a CCC can be characterized as a category* $\mathbf{C}$ *such that the following functors have a right adjoint: (i) the unique functor $!:\mathbf{C}\to\mathbf{1}$, (ii) the diagonal functor $\Delta:\mathbf{C}\to\mathbf{C}\times\mathbf{C}$ defined by $\Delta(c)=(c,c)$ and $\Delta(f)=(f,f)$, (iii) the functors $\_\times A:\mathbf{C}\to\mathbf{C}$, for any object A.*

It is possible to *skolemize* the definition of CCC, that is, to eliminate the existential quantifications, using the type operators 1, $(\_\times\_)$, $(\_)^{(\_)}$ and the term operators $*$, $\langle\_,\_\rangle$, $\Lambda(\_)$. In this way, the theory of CCC's can be expressed as a typed equational theory.

Exercise 4.2.8 *Show that a CCC can be characterized as a category* $\mathbf{C}$ *such that the following equations hold.*

- *There are $1\in\mathbf{C}$ and $*_A:A\to 1$, such that for all $f:A\to 1$,*

$$(!)\quad f=*_A\ .$$

- *There are $\pi_1:A\times B\to A$ and $\pi_2:A\times B\to B$, for any $A,B\in\mathbf{C}$, and $\langle f,g\rangle:C\to A\times B$ for any $f:C\to A$, $g:C\to B$, such that for all $f:C\to A$, $g:C\to B$, $h:C\to A\times B$,*

$$\begin{array}{lll}(Fst) & \pi_1\circ\langle f,g\rangle & = f\\ (Snd) & \pi_2\circ\langle f,g\rangle & = g\\ (SP) & \langle\pi_1\circ h,\pi_2\circ h\rangle & = h\ .\end{array}$$

- *There are $ev:B^A\times A\to B$ for any $A,B\in\mathbf{C}$, and $\Lambda(f)$ for any $f:C\times A\to B$, such that for all $f:C\times A\to B$, $h:C\to B^A$,*

$$\begin{array}{lll}(\beta_{cat}) & ev\circ(\Lambda(f)\times id) & = f\\ (\eta_{cat}) & \Lambda(ev\circ(h\times id)) & = h\ ,\end{array}$$

where $f\times g=\langle f\circ\pi_1,g\circ\pi_2\rangle$.

Exercise 4.2.9 *Referring to exercise 4.2.8 prove that (SP) is equivalent to:*

$$\begin{array}{lll} (DPair) & \langle f,g\rangle \circ h & = \langle f\circ h, g\circ h\rangle \\ (FSI) & \langle \pi_1,\pi_2\rangle & = id\,, \end{array}$$

and that (β_{cat}) and (η_{cat}) are equivalent to:

$$\begin{array}{lll} (Beta) & ev\circ\langle\Lambda(f),g\rangle & = f\circ\langle id,g\rangle \\ (D\Lambda) & \Lambda(f)\circ h & = \Lambda(f\circ(h\times id)) \\ (AI) & \Lambda(ev) & = id\,. \end{array}$$

Exercise 4.2.10 *Show that the following categories are cartesian closed: (1) (finite) sets, (2) (finite) posets and monotonic functions. On the other hand prove that the category* **pSet** *of sets and partial functions is not cartesian closed. Hint: Consider the existence of an isomorphism between* $\mathbf{pSet}[2\times 2,1]$ *and* $\mathbf{pSet}[2,4]$.

One can now formally prove that the category of directed complete partial orders (dcpo's) and functions preserving lub's of directed sets is cartesian closed using propositions 1.4.1 and 1.4.4. Exercise 1.4.6 does not say directly that the product construction in **Dcpo** yields a categorical product. This follows from the following general (meta-)property.

Exercise 4.2.11 *Let* $\mathbf{C},\mathbf{C}'$ *be categories, and* $F:\mathbf{C}\to\mathbf{C}'$ *be a faithful functor. Suppose that* $\mathbf{C}'$ *has products, and that for any pair of objects* A *and* B *of* $\mathbf{C}$ *there exists an object* C *and two morphisms* $\alpha:C\to A$ *and* $\beta:C\to B$ *in* $\mathbf{C}$ *such that:*

$$F(C)=F(A)\times F(B),\quad F(\alpha)=\pi_1,\quad F(\beta)=\pi_2\,,$$

and for any object D *and morphisms* $f:D\to A$, $g:D\to B$, *there exists a morphism* $h:D\to C$ *such that* $F(h)=\langle F(f),F(g)\rangle$. *Show that* $\mathbf{C}$ *has products. Explain why this general technique applies to* **Dcpo**.

In a similar way one can verify that the function space construction in **Dcpo** yields a categorical exponent. The check is slightly more complicated than for the product, due to the fact that the underlying set of the function space in **Dcpo** is a proper subset of the function space in **Set**.

Exercise 4.2.12 *Let* $\mathbf{C},\mathbf{C}'$ *be categories, and* $F:\mathbf{C}\to\mathbf{C}'$ *be a faithful functor. Suppose that the assumptions of exercise 4.2.11 hold, and use* $\times$ *to denote the cartesian product in* $\mathbf{C}$. *Suppose that* $\mathbf{C}'$ *has exponents, and that for any pair of objects* A *and* B *of* $\mathbf{C}$ *there exists an object* C *of* $\mathbf{C}$, *a mono* $m:FC\to FB^{FA}$ *and a morphism* $\gamma:C\times A\to B$ *such that: (1)* $F(\gamma)=ev\circ(m\times id)$, *and (2) for any object* D *and arrow* $f:D\times A\to B$, *there exists a morphism* $k:D\to C$ *such that* $m\circ F(k)=\Lambda(F(f))$. *Show that* $\mathbf{C}$ *has exponents. Apply this to* **Dcpo**.

Theorem 4.2.13 (Dcpo CCC) **Dcpo** *is a cartesian closed category. The order for products is componentwise, and the order for exponents is pointwise.* **Cpo** *is cartesian closed too.*

PROOF. We can apply the exercises 1.4.6 and 4.2.12. A direct proof of cartesian closure is also possible and easy. □

$$\begin{array}{lll} (\text{Asmp}) & [\![x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash x_i:\sigma_i]\!] & = \pi_{n,i} \\ (\to_I) & [\![\Gamma\vdash\lambda x:\sigma.M:\sigma\to\tau]\!] & = \Lambda([\![\Gamma,x:\sigma\vdash M:\tau]\!]) \\ (\to_E) & [\![\Gamma\vdash MN:\tau]\!] & = ev\circ\langle[\![\Gamma\vdash M:\sigma\to\tau]\!],[\![\Gamma\vdash N:\sigma]\!]\rangle \end{array}$$

Figure 4.4: Interpretation of the simply typed λ-calculus in a CCC

4.3 Interpretation of λ-calculi

We explain how to interpret the simply typed λ-calculus in an arbitrary CCC. Suppose that $\mathbf{C}$ is a CCC. Let us choose a terminal object 1, a product functor $\times:\mathbf{C}\times\mathbf{C}\to\mathbf{C}$ and an exponentiation functor $\Rightarrow:\mathbf{C}^{op}\times\mathbf{C}\to\mathbf{C}$. Then there is an obvious interpretation for types as objects of the category, which is determined by the interpretation of the atomic types. The arrow is interpreted as exponentiation in $\mathbf{C}$. Hence given an interpretation $[\![\kappa]\!]$ for the atomic types, we have:

$$[\![\sigma\to\tau]\!]=[\![\sigma]\!]\Rightarrow[\![\tau]\!]\,.$$

Consider a provable judgment of the shape $x_1:\sigma_1,\ldots,x_n:\sigma_n\vdash M:\sigma$. Its interpretation will be defined *by induction on the length of the proof* as a morphism from $[\![\Gamma]\!]$ to $[\![\sigma]\!]$, where we set $\Gamma=x_1:\sigma_1,\ldots,x_n:\sigma_n$ and $[\![\Gamma]\!]=1\times[\![\sigma_1]\!]\times\ldots\times[\![\sigma_n]\!]$. We will take the convention that $\times$ associates to the left. We denote with $\pi_{n,i}:[\![\Gamma]\!]\to[\![\sigma_i]\!]$ $(i=1,\ldots,n)$ the morphism: $\pi_2\circ\pi_1\circ\cdots\circ\pi_1$, where π_1 is iterated $(n-i)$ times.

The interpretation is defined in figure 4.4. The last two rules need some explanation. Suppose $C=[\![\Gamma]\!]$, $A=[\![\sigma]\!]$, and $B=[\![\tau]\!]$.

$(\to_I)$ If there is a morphism $f:C\times A\to B$ then there is a uniquely determined morphism $\Lambda(f):C\to B^A$.

$(\to_E)$ If there are two morphisms $f:C\to B^A$ and $g:C\to A$, then one can build the morphism $\langle f,g\rangle:C\to B^A\times A$ and composing with ev one gets $ev\circ\langle f,g\rangle:C\to A$.

Sometimes, we write $[\![M]\!]$ as an abbreviation for $[\![\Gamma\vdash M:\sigma]\!]$. We shall mostly use this abbreviation when M is closed. When composing the interpretation of the judgment $\Gamma\vdash M:\tau$ with an environment, that is, a morphism in $\mathbf{C}[1,[\![\Gamma]\!]]$, we will freely use the notation $[\![M]\!]\circ\langle d_1,\ldots,d_n\rangle$ which relies on an n-ary product.

In section 4.5 we will work with a simply typed λ-calculus enriched with a set of constants C. We suppose that each constant is labelled with its type, say c^σ. The typing system is then enriched with the rule:

$$\frac{}{\Gamma\vdash c^\sigma:\sigma}\,. \tag{4.1}$$

We denote with $\Lambda(C)$ the collection of well-typed terms. The interpretation is fixed by providing for each constant c^σ a morphism $f_c:1\to[\![\sigma]\!]$. The judgment

$\Gamma \vdash c^\sigma : \sigma$ is then interpreted by composing with the terminal morphism:

$$[\![\Gamma \vdash c^\sigma : \sigma]\!] = f_c \circ ! \ . \tag{4.2}$$

The interpretation in figure 4.4 is defined by induction on the structure of a proof of a judgment $\Gamma \vdash M : \sigma$. In the simple system we presented here, a judgment has a *unique* proof. However, in general, there can be several ways of deriving the same judgment, therefore a problem of *coherence* of the interpretation arises, namely one has to show that different proofs of the same judgment receive the same interpretation. Note that in the simply typed calculus the coherence problem is avoided by getting rid of the structural rules. This trick does not suffice in more sophisticated type theories like *LF* (see chapter 11) where the derivation is not completely determined by the structure of the judgment. In this case term judgments and type judgments are inter-dependent.

Exercise 4.3.1 *Show that* $[\![\Gamma, x : \sigma \vdash M : \tau]\!] = [\![\Gamma \vdash M : \tau]\!] \circ \pi_1$ *if* $\Gamma \vdash M : \tau$ *and* $x : \sigma \notin \Gamma$ *(cf. exercise 4.1.2).*

Exercise 4.3.2 *Given two contexts* $\Gamma, x : \sigma, y : \tau, \Gamma'$ *and* $\Gamma, y : \tau, x : \sigma, \Gamma'$ *define an isomorphism* ϕ *between the corresponding objects. Hint: If* $\Gamma \equiv z : \rho$ *and* Γ' *is empty then* $\phi \equiv \langle\langle \pi_1 \circ \pi_1, \pi_2 \rangle, \pi_2 \circ \pi_1 \rangle : (C \times A) \times B \to (C \times B) \times A$. *Show that (cf. exercise 4.1.2):*

$$[\![\Gamma, x : \sigma, y : \tau, \Gamma' \vdash M : \rho]\!] = [\![\Gamma, y : \tau, x : \sigma, \Gamma' \vdash M : \rho]\!] \circ \phi \ .$$

The next step is to analyse the interpretation of substitution in a category.

Theorem 4.3.3 (substitution) *Let* $\Gamma, x : \sigma \vdash M : \tau$, *and* $\Gamma \vdash N : \sigma$. *The following properties hold.*

(1) $\Gamma \vdash M[N/x] : \tau$.

(2) $[\![\Gamma \vdash M[N/x] : \tau]\!] = [\![\Gamma, x : \sigma \vdash M : \tau]\!] \circ \langle id, [\![\Gamma \vdash N : \sigma]\!] \rangle$.

PROOF. (1) By induction on the length of the proof of $\Gamma, x : \sigma \vdash M : \tau$. The interesting case arises when the last deduction is by $(\to_I)$:

$$\frac{\Gamma, x : \sigma, y : \tau \vdash M : \tau'}{\Gamma, x : \sigma \vdash \lambda y : \tau . M : \tau \to \tau'} \ .$$

We observe $(\lambda y : \tau . M)[N/x] \equiv \lambda y : \tau . M[N/x]$. We can apply the inductive hypothesis on $\Gamma, y : \tau, x : \sigma \vdash M : \tau'$ (note the exchange on the assumptions) to get $\Gamma, y : \tau \vdash M[N/x] : \tau'$ from which $\Gamma \vdash (\lambda y : \tau . M)[N/x] : \tau \to \tau'$ follows by $(\to_I)$.

(2) We will use the exercises 4.3.1 and 4.3.2 on the interpretation of weakening and exchange. Again we proceed by induction on the length of the proof of $\Gamma, x : \sigma \vdash M : \tau$ and we just consider the case $(\to_I)$. We set $B = [\![\tau]\!]$, $B' = [\![\tau']\!]$, $C = [\![\Gamma]\!]$, and:

$$\begin{array}{ll}
f_1 = [\![\Gamma \vdash \lambda y : \tau . M[N/x] : \tau \to \tau']\!] & : C \to B'^B \\
g_1 = [\![\Gamma, y : \tau \vdash M[N/x] : \tau']\!] & : C \times B \to B' \\
f_2 = [\![\Gamma, x : \sigma \vdash \lambda y : \tau . M : \tau \to \tau']\!] & : C \times A \to B'^B \\
g_2 = [\![\Gamma, y : \tau, x : \sigma \vdash M : \tau']\!] & : (C \times B) \times A \to B' \\
f_3 = [\![\Gamma \vdash N : \sigma]\!] & : C \to A \\
g_3 = [\![\Gamma, y : \tau \vdash N : \sigma]\!] & : C \times B \to A \\
g_2' = [\![\Gamma, x : \sigma, y : \tau \vdash M : \tau']\!] & : (C \times A) \times B \to B' \ .
\end{array}$$

We have to show $f_1 = f_2 \circ \langle id, f_3 \rangle$, knowing by induction hypothesis that $g_1 = g_2 \circ \langle id, g_3 \rangle$. We observe that $f_1 = \Lambda(g_1)$, $f_2 = \Lambda(g_2')$, and $g_2' = g_2 \circ \phi$, where ϕ is the iso given by exercise 4.3.2. Moreover $g_3 = f_3 \circ \pi_1$ (cf. exercise 4.3.1). We then compute (cf. exercise 4.2.8):

$$\begin{array}{rcl}
f_2 \circ \langle id, f_3 \rangle & = & \Lambda(g_2') \circ \langle id, f_3 \rangle \\
& = & \Lambda(g_2' \circ (\langle id, f_3 \rangle \times id)) \ .
\end{array}$$

So it is enough to show $g_1 = g_2' \circ (\langle id, f_3 \rangle \times id)$. We compute on the right hand side:

$$\begin{array}{rcl}
g_2' \circ (\langle id, f_3 \rangle \times id) & = & g_2 \circ \phi \circ \langle \langle \pi_1, f_3 \circ \pi_1 \rangle, \pi_2 \rangle \\
& = & g_2 \circ \phi \circ \langle \langle \pi_1, g_3 \rangle, \pi_2 \rangle \\
& = & g_2 \circ \langle \langle \pi_1, \pi_2 \rangle, g_3 \rangle \\
& = & g_2 \circ \langle id, g_3 \rangle \ .
\end{array}$$

□

The categorical interpretation can be seen as a way of compiling a language with variables into a language without variables. The slogan is that *variables are replaced by projections*, for instance $[\![\emptyset \vdash \lambda x : \sigma . x : \sigma \to \sigma]\!] = \Lambda(\pi_2)$. In other words, rather than giving a symbolic reference in the form of a variable, one provides a path for accessing a certain information in the context.[1] As a matter of fact the *compilation* of the λ-calculus into the categorical language has been taken as a starting point for the definition of an abstract machine (the *Categorical Abstract Machine* (CAM), see [CCM87]) in the style of Landin's classical *SECD* machine [Lan64] (see [Cur86] for a comparison). The purpose of these machines is to provide a high-level description of data structures and algorithms used to efficiently reduce λ-terms. In the CAM approach, a fundamental problem is that of orienting the equations that characterize CCC's as defined in exercise 4.2.8. In the following we drop all type information and we restrict our attention to the simulation of β-reduction (the treatment of the extensional rules raises additional problems). Hardin [Har89] has studied the term rewriting system $\mathcal{E} + Beta$ described in figure 4.5. The most important results are:

- $\mathcal{E}$ is confluent and strongly normalizing.
- $\mathcal{E} + Beta$ is confluent (on a subset of categorical terms which is large enough to contain all the compilations of λ-terms).

The proof of strong normalization of $\mathcal{E}$ is surprisingly difficult [CHR96]. The proof of confluence for $\mathcal{E} + Beta$ uses the strong normalization property of $\mathcal{E}$ and the confluence of β in the λ-calculus. The key connection is given by the following fact: if $M \to_\beta N$, if f and g are the compilations of M and N, then there is an h such that $f \to_{Beta} h$ and g is the $\mathcal{E}$ normal form of h. The system $\mathcal{E}$ takes care of explicitly carrying the substitution involved in the β-reduction.

Simpler results have been obtained with a related calculus called λσ-calculus [ACCL92, CHL96]. More results on abstract machines which are related to the CAM are described in section 8.3.

[1] de Bruijn conventions for the representation of variables as distances from the respective binders (see, e.g., [ACCL92]), as well as standard implementations of environments in abstract machines (see section 8.3) follow related ideas.

$$(Beta)\quad ev \circ \langle \Lambda(f), g\rangle \to f \circ \langle id, g\rangle$$

$$(\mathcal{E})\quad \begin{cases} (f\circ g)\circ h & \to & f\circ(g\circ h) \\ id\circ f & \to & f \\ \pi_1 \circ id & \to & \pi_1 \\ \pi_2 \circ id & \to & \pi_2 \\ \pi_1 \circ \langle f, g\rangle & \to & f \\ \pi_2 \circ \langle f, g\rangle & \to & g \\ \langle f, g\rangle \circ h & \to & \langle f\circ h, g\circ h\rangle \\ \Lambda(f)\circ h & \to & \Lambda(f\circ\langle h\circ\pi_1, \pi_2\rangle) \end{cases}$$

Figure 4.5: A rewriting system for the β-categorical equations

Exercise 4.3.4 *Show that two λ-terms are compiled into the same categorical term if and only if they are α-convertible (cf. section 2.1).*

4.4 From CCC's to λ-theories and back

We study the equivalence induced by the interpretation of the simply typed λ-calculus in a CCC. It turns out that the equivalence is closed under $\beta\eta$-conversion and forms a congruence.

Definition 4.4.1 (λ-theory) *Let T be a collection of judgments of the shape $\Gamma \vdash M = N : \sigma$ such that $\Gamma \vdash M : \sigma$ and $\Gamma \vdash N : \sigma$. T is called a λ-theory if it is closed under the rules in figure 4.6.*

We note that the congruence generated by the axioms β and η is the smallest λ-theory, we call it the pure $\lambda\beta\eta$ theory. To every CCC we can associate a λ-theory.

Theorem 4.4.2 *Let $\mathbf{C}$ be a CCC and let $[\![\ ]\!]$ be an interpretation in the sense of figure 4.4 of the simply typed λ-calculus defined over $\mathbf{C}$. Then the following collection is a λ-theory.*

$$Th(\mathbf{C}) = \{\Gamma \vdash M = N : \sigma \mid \Gamma \vdash M : \sigma, \Gamma \vdash N : \sigma \text{ and } [\![\Gamma \vdash M : \sigma]\!] = [\![\Gamma \vdash N : \sigma]\!]\}\ .$$

PROOF. We have to check that $Th(\mathbf{C})$ is closed under the rules presented in figure 4.6. For (α) we observe that $[\![\ ]\!]$ is invariant with respect to the names of bound variables (cf. exercise 4.3.4).

(β) Let $[\![\Gamma \vdash (\lambda x : \sigma.M)N : \tau]\!] = ev \circ \langle \Lambda(f), g\rangle$, where $f = [\![\Gamma, x : \sigma \vdash M : \tau]\!]$ and $g = [\![\Gamma \vdash N : \sigma]\!]$. By the substitution theorem, $[\![\Gamma \vdash M[N/x] : \tau]\!] = f \circ \langle id, g\rangle$, and (cf. exercise 4.2.9) $f \circ \langle id, g\rangle = ev \circ \langle \Lambda(f), g\rangle$.

$$(\alpha)\quad \frac{\Gamma \vdash \lambda x : \sigma.N : \sigma \to \tau \quad y \notin FV(N)}{\Gamma \vdash \lambda y : \sigma.N[y/x] = \lambda x : \sigma.N : \sigma \to \tau}$$

$$(\beta)\quad \frac{\Gamma \vdash (\lambda x : \sigma.M)N : \tau}{\Gamma \vdash (\lambda x : \sigma.M)N = M[N/x] : \tau}$$

$$(\eta)\quad \frac{\Gamma \vdash \lambda x : \sigma.(Mx) : \sigma \to \tau \quad x \notin FV(M)}{\Gamma \vdash \lambda x : \sigma.(Mx) = M : \sigma \to \tau}$$

$$(\xi)\quad \frac{\Gamma, x : \sigma \vdash M = N : \tau}{\Gamma \vdash \lambda x : \sigma.M = \lambda x : \sigma.N : \sigma \to \tau}$$

$$(appl)\quad \frac{\Gamma \vdash M = N : \sigma \to \tau \quad \Gamma \vdash M' = N' : \sigma}{\Gamma \vdash MM' = NN' : \tau}$$

$$(Asmp)\quad \frac{\Gamma \vdash M = N : \sigma \in T}{\Gamma \vdash M = N : \sigma} \qquad (weak)\quad \frac{\Gamma \vdash M = N : \sigma \quad x : \tau \notin \Gamma}{\Gamma, x : \tau \vdash M = N : \sigma}$$

$$(refl)\quad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma} \qquad (sym)\quad \frac{\Gamma \vdash M = N : \sigma}{\Gamma \vdash N = M : \sigma}$$

$$(trans)\quad \frac{\Gamma \vdash M = N : \sigma \quad \Gamma \vdash N = P : \sigma}{\Gamma \vdash M = P : \sigma}$$

Figure 4.6: Closure rules for a typed λ-theory

(η) We have:

$$\begin{aligned} [\![\Gamma \vdash \lambda x : \sigma.Mx : \sigma \to \tau]\!] &= \Lambda(ev \circ \langle [\![\Gamma, x : \sigma \vdash M : \sigma \to \tau]\!], [\![\Gamma, x : \sigma \vdash x : \sigma]\!]\rangle) \\ &= \Lambda(ev \circ \langle [\![\Gamma \vdash M : \sigma \to \tau]\!] \circ \pi_1, \pi_2\rangle) \\ &= \Lambda(ev \circ ([\![\Gamma \vdash M : \sigma \to \tau]\!] \times id)) \\ &= [\![\Gamma \vdash M : \sigma \to \tau]\!]\ . \end{aligned}$$

For $(weak)$ we use the exercise 4.3.1. The rules $(refl)$, (sym), $(trans)$ hold since $Th(\mathbf{C})$ is an equivalence. Finally, (ξ), (apl) follow by the definition of the interpretation of abstraction and application. □

Exercise 4.4.3 *Show that there are infinitely many λ-theories. Hints: Interpret the atomic types as finite sets and consider the resulting λ-theory. Then analyse the $\beta\eta$-normal forms of type $(\kappa \to \kappa) \to (\kappa \to \kappa)$.*

Next, we show how to generate a CCC starting from a λ-theory. The construction consists essentially in taking types as objects of the category and (open) terms quotiented by the λ-theory as morphisms (cf. Henkin's term model [Hen50]). It

is convenient to work in an extended setting combining λ-calculus with product types. We take the following steps:

(1) We extend the language with constructors for terminal object and product, as well as the relative equations:

$$\begin{array}{lll} \text{Types} & At & ::= \kappa \mid \kappa' \mid \ldots \\ & \sigma & ::= At \mid 1 \mid \sigma \times \sigma \mid \sigma \to \sigma \\ \text{Terms} & v & ::= x \mid y \mid \ldots \\ & M & ::= v \mid * \mid \langle M, M \rangle \mid \pi_1 M \mid \pi_2 M \mid \lambda v : \sigma . M \mid MM \ . \end{array}$$

1. Typing rules. The rules of the simply typed calculus (figure 4.2), plus the rules for conjunction (figure 4.3), plus:

$$(*) \quad \frac{}{\Gamma \vdash * : 1} \ .$$

2. Equations. A theory is now a set of equality judgments closed under the rules of the pure $\lambda\beta\eta$-theory (figure 4.6) plus:

$$(*) \quad \frac{\Gamma \vdash M : 1}{\Gamma \vdash M = *} \qquad (SP) \quad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash \langle \pi_1 M, \pi_2 M \rangle = M : \sigma \times \tau}$$
$$(\pi_1) \quad \frac{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}{\Gamma \vdash \pi_1 \langle M, N \rangle = M : \sigma} \qquad (\pi_2) \quad \frac{\Gamma \vdash \langle M, N \rangle : \sigma \times \tau}{\Gamma \vdash \pi_2 \langle M, N \rangle = N : \tau} \ .$$

(2) We associate to a theory T a CCC $\mathbf{C}(T)$ as follows.

1. The objects are the types of the extended language.

2. The morphisms are equivalences classes of *open* terms according to the equivalence induced by T. More precisely:

$$\begin{array}{lcl} \mathbf{C}(T)[\sigma, \tau] & = & \{[M] \mid \exists \Gamma \ \ \Gamma \vdash M : \tau\} \\ [M] & = & \{N \mid \exists \Gamma, \tau \ \ \Gamma \vdash M = N : \tau \in T\} \ . \end{array}$$

3. The structure associated to every CCC is defined as follows (we omit the type labels):

$$\begin{array}{ll} (id) & [\lambda x.x] \\ (comp) & [M] \circ [N] = [\lambda x.M(Nx)] \quad (x \text{ fresh}) \\ (term) & !_\sigma = [\lambda x.*] \\ (proj) & \pi_1 = [\lambda x.\pi_1 x] \ \ \pi_2 = [\lambda x.\pi_2 x] \\ (pair) & \langle [M], [N] \rangle = [\lambda x.\langle Mx, Nx \rangle] \quad (x \text{ fresh}) \\ (eval) & ev = [\lambda x.(\pi_1 x)(\pi_2 x)] \\ (curry) & \Lambda([M]) = [\lambda y.\lambda z.M\langle y, z \rangle] \quad (y, z \text{ fresh}) \ . \end{array}$$

4. We leave to the reader the verification of the equations associated to a CCC.

(3) Finally we have to verify that the λ-theory associated to $\mathbf{C}(T)$ is exactly T. To this end one checks that:

$$[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \sigma]\!] = [\lambda x : \tau . M[\pi_{n,1} x / x_1, \ldots \pi_{n,n} x / x_n]_\beta],$$

where $\tau \equiv (\cdots (1 \times \sigma_1) \times \cdots \times \sigma_n)$.

We can summarize our constructions as follows.

Theorem 4.4.4 (from λ-theories to CCC's) *Given any λ-theory T over the simply typed calculus with products and terminal object we can build a CCC $\mathbf{C}(T)$ such that the λ-theory associated to $\mathbf{C}(T)$ coincides with T.*

Remark 4.4.5 (1) *It is possible to see the constructions described here as representing an equivalence between a* category of CCC's *and a* category of λ-theories *[LS86].*

(2) *It is possible to strengthen the previous theorem by considering a theory T over the simply typed λ-calculus (*without *products and terminal object). Then one needs to show that it is possible to add* conservatively *to T the equations $(*)$, (π_1), (π_2), and (SP), i.e., that adding these new equations does not allow us to prove new equalities between two simply typed λ-terms. We refer to [Cur86, chapter 1] for a description of suitable proof techniques.*

4.5 Logical relations

Logical relations are a quite ubiquitous tool in semantics and in logic. They are useful to establish links between syntax and semantics. In this section, logical relations are defined and applied to the proof of three results of this sort: Friedman's completeness theorem [Fri73], which characterizes $\beta\eta$-equality, and Jung-Tiuryn's and Sieber's theorems [JT93, Sie92] on the characterization of λ-definability.

Logical relations are predicates relating models of a given λ-calculus with constants $\Lambda(C)$, defined by induction over types. To simplify matters, throughout the rest of this section, we make the assumption that there is only one basic type κ. We define next (binary) logical relations, to this end we fix some terminology. Recall that an interpretation of simply typed λ-calculus in a CCC $\mathbf{C}$ is given as soon as the basic type κ is interpreted by an object D^κ of $\mathbf{C}$. We shall summarize this by calling the pair $\mathcal{M} = (\mathbf{C}, D^\kappa)$ a model. We write $[\![\sigma]\!] = D^\sigma$, hence $D^{\sigma \to \tau} = D^\sigma \Rightarrow D^\tau$.

If there are constants, then the constants must also be interpreted, but we leave this implicit to keep notation compact. We shall make repeated use of the hom-sets of the form $\mathbf{C}[1, D]$. It is thus convenient to use a shorter notation. We shall write, for any object D of $\mathbf{C}$:

$$\mathbf{C}[1, D] = \underline{D} \ .$$