Chapter 10

Related subjects

10.1 Impredicative type theories

Second-order, or impredicative, or polymorphic, type theories like the Calculus of Constructions [21] and second-order typed lambda calculus allow the formation of types in the lowest universe, which we call **Data**: **Type** here, by quantification over types from a higher universe:

$$\frac{A\colon \mathbf{Type} \qquad D\colon \mathbf{Data}^A}{\Pi(A;D)\colon \mathbf{Data}}$$

Thus, **Data** is very much like our **Prop**, except that **Prop** has additional equality rules stating that equivalent propositions are equal, and that all proofs of the same proposition are equal. Furthermore, their use is different, for objects in **Data** are used for actual computation, while objects in **Prop** are used for stating properties only. The impredicative quantification allows one to define all kinds of weakly initial and final algebras, without using further primitive notions.

Example 10.1 The type of booleans can be defined by

$$\begin{aligned}
\mathsf{Bool} &:= \Pi(X\colon \mathbf{Data};\ x\colon X;\ y\colon X :: X)\\
\mathsf{true} &:= (X;x;y :: x)\\
\mathsf{false} &:= (X;x;y :: y)\\
b\colon \mathsf{Bool};\ E\colon \mathbf{Data};\ e_0,e_1\colon E \vdash \quad \underline{\mathsf{if}}\ b\ \underline{\mathsf{then}}\ e_0\ \underline{\mathsf{else}}\ e_1 &:= bEe_0e_1
\end{aligned}$$

(End of example)

A drawback of such impredicative encodings is that dependent types like $T\colon \Pi(x\colon \mathsf{Bool} :: \mathbf{Data})$ cannot use an elimination on the impredicative object x, because expression '$x\mathbf{Data}T_0T_1$' would be wrongly typed.

Luo's ECC [48] extends the Calculus of Constructions with generalized sums and a hierarchy of universes as in *ADAM*. Ore [66] extended ECC further with disjoint sums (sums over a finite type) and inductive types at the predicative level. We may call this system ECCI. It is equivalent to *ADAM* without equality types and strong proof elimination; data types are to be built at the predicative level rather than in the impredicative universe **Prop**.

10.1.1 Weak initial algebras

The weak initial algebra $(T;\theta)$ that has a sequence of constructors $\theta_j\colon F_j.T \to T$ can be impredicatively defined by: ($\Delta\colon \mathcal{C} \to \mathcal{C}^N$ is the diagonal functor $X \mapsto (i :: X)$)

$$\begin{array}{rcl} T\colon \mathbf{Data} & := & \Pi(X\colon \mathbf{Data};\ \phi\colon F.X \to \Delta.X :: X) \\ \theta_j\colon F_j.T \to T & := & y \mapsto (X;\phi :: \phi_j.(F_j.(x \mapsto xX\phi).y)\,) \end{array}$$

In particular, a weak initial F-algebra for $F\colon \mathbf{Data} \to \mathbf{Data}$ is given by:

$$\begin{array}{rcl} \mu_{\mathsf{w}}F\colon \mathbf{Data} & := & \Pi(X\colon \mathbf{Data};\ \phi\colon F.X \to X :: X) \\ \tau\colon F.\mu_{\mathsf{w}}F \to \mu_{\mathsf{w}}F & := & y \mapsto (X;\phi :: \phi.(F.(x \mapsto xX\phi).y)\,) \end{array}$$

Given another F-algebra $(U;\psi)$, there is a homomorphism

$$(\!|U;\psi|\!)\colon \mu_{\mathsf{w}}F \to U \ := \ x \mapsto xU\psi$$

for which we have the reduction rule $(\!|U;\psi|\!)\circ\tau \ \texttt{=>}\ \psi\circ F.(\!|U;\psi|\!)$. One cannot prove that this homomorphism is unique. For instance, given (weak) binary products, we cannot construct a weak paramorphism $[\![\psi]\!]$ such that $[\![\psi]\!]\circ\tau \texttt{=>} \psi\circ F.\langle \mathsf{Id}, [\![\psi]\!]\rangle$, nor even a true $\tau^{\cup}$ such that $\tau^{\cup}.(\tau.y) \texttt{=>} y$, nor a (transfinite) induction property like (6.5).

In section D.7 we give an impredicative definition of $\mathbb{N}$ in typed lambda calculus, and prove that induction holds by naturality for all terms of type $\mathbb{N}$. This generalizes easily to type $\mu_{\mathsf{w}}F$, and one may expect that the naturality property holds for generalized calculi like CC too. Yet this would not yield an induction theorem *within* the calculus.

One might restrict, as suggested in [73], all quantifications over $\mu_{\mathsf{w}}F$ as to use only its *standard* elements, being those elements that satisfy transfinite induction:

$$\mathsf{St}(x\colon \mu_{\mathsf{w}}F) \ := \ \Pi(U\colon \mathbf{Data}^{\mu_{\mathsf{w}}F};\ h\colon \Pi(y\colon F.\mu_{\mathsf{w}}F :: (F'.U)(y) \to U(\tau.y)) :: Ux)$$

where $(F'.U)(y)$ stands, as in paragraph 6.2.3, for the product of Uz for all immediate predecessors z of $\tau.y$. In particular, if $F.X = \Sigma(a\colon A :: X^{Ba})$, then

$$(F'.U)(a;t) \ = \ \Pi(y\colon Ba :: U(t_y)) \ .$$

A declaration '$x\colon \mu F$' may then be replaced by '$x\colon \mu_{\mathsf{w}}F;\ \mathsf{St}\,x$ ' Now, if one's calculus has subtypes, one can use the subtype $\{\,x\colon \mu_{\mathsf{w}}F \mid\colon \mathsf{St}\,x\}$ for μF. If it does not have subtypes, as the Calculus of Constructions, this restriction of quantifications to standard elements does not give a satisfactory solution, for then a quantification over all types cannot be applied to the class of all standard elements of $\mu_{\mathsf{w}}F$.

So it will be more satisfactory to extend the calculus with inductive types as a primitive notion (at the impredicative level). This is done by Coquand and Paulin in [22], using type definitions as described in subsection 5.3.1 and a recursor as we described in paragraph 6.2.3.

Ore [66] discusses extending CC with inductive types either at the impredicative or predicative level.

10.1.2 Weak final algebras

An analogous treatment as in 10.1.1 is possible for co-inductive types. The dual impredicative definition of weak final coalgebras utilizes Σ, but this Σ can be translated into a double use of Π:

$$\begin{aligned}
\nu_{\mathsf{w}}F\colon \mathbf{Data} &:= \Sigma_{\mathsf{w}}(X\colon \mathbf{Data};\ \phi\colon X \to F.X :: X) \\
&:= \Pi(Y\colon \mathbf{Data};\ \Pi(X\colon \mathbf{Data};\ \phi\colon X \to F.X;\ x\colon X :: Y) :: Y) \\
\delta\colon \nu_{\mathsf{w}}F \to F.\nu_{\mathsf{w}}F &:= (X;\phi;x) \mapsto F.(z \mapsto (X;\phi;z)).(\phi.x) \\
&:= u \mapsto u(F.\nu_{\mathsf{w}}F)(X;\phi;x :: F.(z \mapsto (X;\phi;z)).(\phi.x)\,)
\end{aligned}$$

Given another F-coalgebra $(U;\phi)$, the mediating morphism (anamorphism) is

$$[\!(U;\phi)\!]\colon U \to \nu_{\mathsf{w}}F \;:=\; u \mapsto (U;\phi;u)\ .$$

10.2 Using type-free values

The notion of type as mainly used in this thesis comprises that types are introduced together with their values; there are no values without types. An alternative is to define types as sets of basically type-free values. A suitable universe of values is the set of untyped lambda terms, to be taken modulo conversion.

When types may be built by unrestricted comprehension, i.e. $\{\,x \mid\colon \Psi(x)\}$ where $\Psi(x)$ is a formula from second-order logic, then one gets inductive types by taking simply

$$\mu F \;:=\; \bigcap(\,X \mid\colon F.X \subseteq X)\ .$$

Such a system with unrestricted second-order comprehension cannot admit types as values themselves, because this would lead to inconsistency.

10.2.1 Henson's calculus TK

Martin Henson [39] introduced a calculus with kinds organized into a hierarchy of levels, in order to avoid inconsistency. Unlike the hierarchy of universes in type theory, kinds of a higher level do not collect kinds of previous levels, nor do they admit greater cardinalities, for the kind that contains everything, $\{\,x \mid\colon \mathsf{True}\}$, is already of level zero. Rather, kinds of a higher level admit a greater definitional complexity.

- Terms are built from constants c, application $(t\,t)$, lambda abstraction $\lambda x.t$ and $\lambda X.t$ where x is a term variable and X a kind variable of some specific level, and may furthermore contain kind expressions and logical formulae as primitive values
- Atomic formulae include $t \in T$, $t = t'$, and $t\!\downarrow$ (meaning "t is defined"), where t, t' are terms and T is a kind
- Formulae are built from atomic formulae by the ordinary propositional connectives and by quantification over either all terms or all kinds of some specific level
- Types are kinds of level 0

- Kinds of level n are either (1) kind variables of level at most n, (2) comprehensions $\{\, x \mid: \Psi(x)\}$ over lambda terms where formula $\Psi(x)$ may contain kind-quantifications over levels below n only, or (3) inductive kinds $\Xi(\Phi, K)$ of level n

Inductive kinds of level n have the form $\Xi(\Phi, K)$, where $\Phi(z, x)$ is a formula that contains kind quantifications below level n only, and K is a kind of level n. Kind $\Xi(\Phi, K)$ is the smallest kind X such that

$$K \subseteq X \quad \text{and} \quad \{\, z \mid: \forall x :: \Phi(z, x) \Rightarrow x \in X\} \subseteq X \ . \tag{10.1}$$

So, when there would be no level restriction on comprehension, Ξ would be definable by

$$\Xi(\Phi, K) \ := \ \bigcap(\, X \mid: (10.1)\,) \ .$$

Put otherwise, $\Xi(\Phi, K)$ is the well-ordered type generated by the relation

$$x \prec z \ := \ z \notin K \wedge \Phi(z, x) \ .$$

An an initial F-algebra, it is

$$\mu(X \mapsto K \cup \{\, z \mid: \forall x :: \Phi(z, x) \Rightarrow x \in X\}) \ .$$

Note that the second argument of Ξ is really superfluous, as by taking $\Phi'(z, x) := x \prec z$ we have

$$\Xi(\Phi, K) \ = \ \Xi(\Phi', \emptyset) \ .$$

Therefore we find this type unnecessarily complicated. The comprehension and induction rules given in [39] and some other publications are actually erroneous — and the claimed consistency proof flawed. For example, the induction rule draws a conclusion $\forall z\!: \in \Xi(\Phi, K) :: \psi z$ without a premise $\forall z\!: \in K :: \psi z$. By taking $\Phi(z, x) := \mathsf{False}$, one would obtain $\forall z :: \psi z$ for every formula ψz. Later publications, like [40], had correct rules.

A more fundamental objection is that the usefulness of kinds and formulae as values is negligible, because terms may not be used in place of kinds or formulae. Henson seems to miss this point. We note that the higher level abstraction facilities are quite limited: one can abstract over kinds, but not over functions on kinds. Finally, we remark that the intuitive basis for this hierarchy of kinds is rather weak.

10.3 Inductive universe formation

Predicative universes of types or sets, such as our $\mathbf{Type}_i$, are described by listing the rules for constructing their elements. Then one might add a principle that the universe is actually the least (or initial) one that is closed under these rules, by giving a universe elimination (or recursion) principle. This would make the universe an initial algebra in a category of families of types and extensions between them. N.P. Mendler discusses the categorical semantics of such recursion rules in [61].

One might strengthen type theory by adding a rule for introducing new inductive universes *inside the system* by listing their introduction rules. The difference between universes and ordinary inductive types is that introduction rules for universes, like Π-formation (B.9), when they have a premise that A be a type in the universe, may in subsequent premises quantify over the type (associated with) A itself.

For this purpose it is best to treat universes *à la* Tarski, namely as a pair $(U;T)$ where U is a type and T assigns to each "code" $A{:}\,U$ a type TA. So a universe is a family of types.

For any type S, we define a category $\mathsf{FAM}\,S$. Its object are families in $\mathsf{Fam}\,S$, and its morphisms are given by:

$$(D;s) \to (D';s') \ \underline{\text{in}}\ \mathsf{FAM}\,S \ := \ \{\, f{:}\,D \to D' \mid{:}\ \forall d{:}\,D :: sd = s'(f.d)\}\ . \tag{10.2}$$

A universe formation principle might read: For any endofunctor F on $\mathsf{FAM}\,\textbf{Type}$ that satisfies some constraints, there is an initial F-algebra. The constraints that are required here are much more difficult to express than for ordinary inductive types, and we will not try to do so.

Example 10.2 The type constructor Π gives rise to an endofunctor P on $\mathsf{FAM}\,\textbf{Type}$, such that there is a morphism $p{:}\,P.(U;T) \to (U;T)$ just when the universe is closed under Π, i.e. when for $a{:}\,U$ and $b{:}\,U^{Ta}$ there is some $c{:}\,U$ with $Tc \cong \Pi(x{:}\,Ta :: T(bx))$. This P is given by:

$$\begin{array}{rcl} P.(U;T) & := & (\,\Sigma(a{:}\,U :: U^{Ta});\\ & & \quad((a;b) :: \Pi(x{:}\,Ta :: T(bx)))\ ) \end{array}$$

and for $f{:}\,(U;T) \to (U';T')$ $\underline{\text{in}}$ $\mathsf{FAM}\,\textbf{Type}$ the definition of $P.f$ is obtained from (10.2):

$$\begin{array}{ll} & P.f \in P.(U;T) \to P.(U';T') \\ \Leftrightarrow & \forall a{:}\,U;\ b{:}\,U^{Ta};\ (a';b') := P.f.(a;b) :: \\ & \quad \Pi(x{:}\,Ta :: T(bx)) = \Pi(x{:}\,T'a' :: T'(b'x)) \qquad \{(10.2) \text{ for } P.f\} \\ \Leftarrow & \forall a{:}\,U;\ b{:}\,U^{Ta};\ (a';b') := P.f.(a;b) :: \\ & \qquad f.a = a' \ \wedge\ \forall x{:}\,Ta :: f.bx = b'x \qquad \{\forall a{:}\,U :: Ta = T'(f.a)\} \\ \Leftrightarrow & \ \forall a{:}\,U;\ b{:}\,U^{Ta} :: P.f.(a;b) = (f.a;\ f^{Ta}.b) \end{array}$$

We can do the same for other type constructors, and define an endofunctor F such that the carrier of an initial F-algebra may serve as the definition of the universe $\textbf{Type}_0$. The object part of F is as follows:

$$\begin{array}{rcll} F.(U;T){:}\,\mathsf{FAM}\,\textbf{Type} & := & (\{\ \mathsf{N} \mid \mathsf{Fin}(n{:}\,\mathbb{N}) \mid \mathsf{Prop} \mid \mathsf{Holds}(P{:}\,\textbf{Prop}) & \\ & & \ \mid \mathsf{Pi}(a{:}\,U;\ b{:}\,U^{Ta}) \mid \mathsf{Sigma}(a{:}\,U;\ b{:}\,U^{Ta})\ \}; & \\ & & \ (\mathsf{N} :: & \mathbb{N} \\ & & \ \mid \mathsf{Fin}(n) :: & n \\ & & \ \mid \mathsf{Prop} :: & \textbf{Prop} \\ & & \ \mid \mathsf{Holds}(P) :: & P \\ & & \ \mid \mathsf{Pi}(a;b) :: & \Pi(x{:}\,Ta :: T(bx)) \\ & & \ \mid \mathsf{Sigma}(a;b) :: & \Sigma(x{:}\,Ta :: T(bx)) \\ & & \ )\ ) & \end{array}$$

10.4 Bar recursion

Though the scheme of bar recursion introduced by Spector [80] has little to do with inductive types, we include it here because it is so remarkably different from our other recursion schemes. It defines a function $f\colon A^* \to B$ on finite sequences by recursive application to *longer* sequences, until a special termination condition holds.

Well-definedness of such a function depends on a property that a computable function $c\colon A^\omega \to \mathbb{N}$ is *continuous* in the sense that its value on an infinite sequence t depends only on a finite prefix $t|_n$ of t:

$$\forall t\colon A^\omega :: \exists n\colon \mathbb{N} :: \forall u\colon A^\omega :: (t|_n = u|_n \Rightarrow c.t = c.u)\ . \tag{10.3}$$

Let a type A with some default value $a\colon A$ be given. We define an embedding of finite into infinite sequences. (Alternatively, we may restrict the definition to nonempty sequences and replace 'a' by s_0.)

$$s\colon A^* \vdash \quad [s]\colon A^\omega \quad := \quad (i :: \underline{\mathsf{if}}\ i < \#s\ \underline{\mathsf{then}}\ s_i\ \underline{\mathsf{else}}\ a\,)$$

The termination condition of f mentioned above is $c.[s] < \#s$. The point is that, as s grows in length, $c.[s]$ must become constant and the condition will be satisfied when s is long enough.

Theorem 10.1 (Bar recursion) *Classically, one can derive:*

$$\frac{\begin{array}{l} c\colon A^\omega \to \mathbb{N} \\ c \text{ is continuous} \\ b\colon A^* \to B \\ e(\_\colon B^A)\colon A^* \to B \end{array}}{\begin{array}{l} \exists! f\colon A^* \to B :: \\ \quad \forall s\colon A^* :: f.s = \underline{\mathsf{if}}\ c.[s] < \#s\ \underline{\mathsf{then}}\ b.s\ \underline{\mathsf{else}}\ e(x :: f.(s +\!\!+ \langle x\rangle)\,).s \end{array}}$$

Proof. The equation for f obviously has a least solution $f\colon A^* \to {\uparrow}B$ in the domain of partial functions. Then for any $s\colon A^*$ such that $f.s = \bot$ one has

$$c.[s] \geq \#s\ \wedge\ \exists(x\colon A :: f.(s +\!\!+ \langle x\rangle) = \bot)\ .$$

Let $g\colon \Pi(s\colon A^*;\ f.s = \bot :: \{x\colon A\ |\colon f.(s +\!\!+ \langle x\rangle) = \bot\})$ be a corresponding choice operator.

To prove totality of f, suppose $f.s = \bot$ for some $s\colon A^*$. Define $t\colon A^\omega$ using total induction by

$$t_i\ :=\ \underline{\mathsf{if}}\ i < \#s\ \underline{\mathsf{then}}\ s_i\ \underline{\mathsf{else}}\ g(t|_{i-1})$$

and see that, for $i\colon\geq \#s$, one has $f.(t|_i) = \bot$, hence $c.[t|_i] \geq \#t|_i = i$.
Let n be according to (10.3) for t, then for $i\colon\geq n$ we have $c.t = c.[t|_i]$ as $t|_n = [t|_i]|_n$.
Taking $i := \mathsf{max}\langle \#s, n, c.t + 1\rangle$, it follows that $c.t = c.[t|_i] \geq i \geq c.t + 1$, contradiction. Thus $f.s = \bot$ cannot be, and f is total. ■

Function c actually defines a well-founded relation by

$$|{\prec}|\ :=\ \{s\colon A^*;\ c.[s] \geq \#s;\ x\colon A :: (s +\!\!+ \langle x\rangle, s)\,\}\ .$$

In a constructive calculus, the continuity condition for c is automatically satisfied and may be omitted from the premises. In that case, the principle of bar recursion is essentially stronger than algebraic recursion in typed polymorphic lambda calculus, as Barendsen and Bezem prove [10].