3. Languages for Algebras, Induction

The intended model of our theory is a many-sorted structure, with one sort for every type. We assume that the model consists of *domains*, in the sense of domain theory (cf. [1]). The reason for the setting is that we want to deal with *computable functionals*. Since their (mathematically correct) domains are the Scott-Ershov partial continuous functionals, this is the intended range of the quantifiers.

3.1. **Languages for Algebras.** We now define the specific logical language we use for our algebras. It should be thought of as a form of arithmetical language, since it is supposed to describe our particular intended model.

A *variable* of a given type is interpreted by a continuous functional (object) of that type. We use the word "variable" and not "program variable", since continuous functionals are not necessarily computable. So for each type ρ we have *general variables* $\hat{x}^\rho, \hat{y}^\rho, \dots$ of type ρ.

In most cases we need to argue about existing (i.e. total) objects only. For the notion of totality we have to refer to [1, Chapter 8.3]; particularly relevant here is exercise 8.5.7. To make formal arguments with quantifiers relativized to total objects more managable, we use a special sort of variables intended to range over such objects only. So for each type ρ we have *total variables* $x^\rho, y^\rho, \dots$ of type ρ.

We reserve certain strings as names for variables of a given type, e.g. n, m for variables of type $\mathbf{N}$. Then also $n_0, n_1, n_2, \dots, m_0, \dots$ can be used for the same purpose. For example, $n_0, n_1, n_2, \dots, m_0, \dots$ range over total natural numbers, and $\hat{n}_0, \hat{n}_1, \hat{n}_2, \dots$ are general variables. We say that the *degree of totality* for the former is 1, and for the latter 0.

A *predicate variable* $\hat{P}$ of arity $\rho_1, \dots, \rho_n$ is a placeholder for a formula A with distinguished (different) variables $\hat{x}_1, \dots, \hat{x}_n$ of types $\rho_1, \dots, \rho_n$. Such an entity is called a *comprehension term*, written $\{\, \hat{x}_1, \dots, \hat{x}_n \mid A \,\}$. By default we have the predicate variable $\bot$ (of empty arity), called (logical) *falsity*. It is viewed as a predicate variable rather than a predicate constant, since (when translating a classical proof into a constructive one) we want to substitute for $\bot$.

Often we will argue about *Harrop formulas* only, i.e. formulas without computational content. For convenience we use a special sort of predicate variables intended to range over comprehension terms with Harrop formulas only. For example, $P_0, P_1, P_2, \dots, Q_0, \dots$ range over comprehension terms with Harrop formulas, and $\hat{P}_0, \hat{P}_1, \hat{P}_2, \dots$ are general predicate variables. We say that *Harrop degree* for the former is 1, and for the latter 0.

We also allow predicate constants with a fixed intended meaning. Predicate variables and constants are both called predicate symbols. The need for predicate constants comes up when e.g. an inductively defined set is expressed via a formula stating the existence of a generation tree; the kernel of this formula is to be axiomatized, using the tree constructors. Prime formulas built from predicate constants do not give rise to extracted terms, and cannot be substituted for.

Specific predicate constants are

- atom of arity $(\mathbf{B})$,

- for every type ρ *equality* $\approx_\rho$ of arity (ρ, ρ), and
- for every type ρ *totality* Total_ρ of arity (ρ).

Notice that for finitary algebras, e.g. $\mathbf{N}$, we have continuous boolean valued functions equality $=_{nat} \colon \mathbf{N} \to \mathbf{N} \to \mathbf{B}$ and existence (definedness, totality) $e_{nat} \colon \mathbf{N} \to \mathbf{B}$. Then we can express equality $r = s$ by $\mathsf{atom}(=(r, s))$ and existence $E(r)$ by $\mathsf{atom}(e(r))$.

A prime formula has the form $P(r_1, \dots, r_n)$ with a predicate variable or constant P and terms $r_1, \dots, r_n$. Write

- T, F for $\mathsf{atom}(\mathsf{tt})$, $\mathsf{atom}(\mathsf{ff})$,
- $r = s$ for $\mathsf{atom}(=(r, s))$,
- $E(r)$ for $\mathsf{atom}(e(r))$, and
- $r \approx s$ for $\approx (r, s)$.

Formulas are built from prime formulas by

- implication $A \to B$,
- conjunction $A \wedge B$,
- tensor $A \otimes B$,
- all quantification $\forall \hat{x}^\rho A$ and
- existential quantification $\exists \hat{x}^\rho A$.

Moreover we have classical existential quantification in an arithmetical and a logical form:

$$\exists^{\mathsf{ca}} \hat{x}_1 \dots \hat{x}_n . A_1 \otimes \dots \otimes A_m \quad \text{arithmetical version}$$

$$\exists^{\mathsf{cl}} \hat{x}_1 \dots \hat{x}_n . A_1 \otimes \dots \otimes A_m \quad \text{logical version.}$$

For all quantifiers we allow that the quantified variable is formed without ˆ, i.e. ranges over total objects only.

Remark. When dealing with the classical existential quantifier, it is – for obvious reasons – useful to be able to unfold and fold it as it seems appropriate. However, notice that there are some slight difficulties in this context. To see the problems, consider

$$\exists^{\mathsf{cl}} x, y A$$

If we read this as $\exists^{\mathsf{cl}} x \exists^{\mathsf{cl}} y A$, then the unfolded form would be

$$\neg \forall x \neg \neg \forall y \neg A$$

However, it would be simpler if we could unfold this formula into the equivalent

$$\neg \forall x \forall y \neg A$$

To achive this effect, we allow lists of variables after a classical existential quantifier, and unfold $\exists^{\mathsf{cl}} x, y A$ into the latter (shorter) formula, but $\exists^{\mathsf{cl}} x \exists^{\mathsf{cl}} y A$ into the former.

Another (small) problem arises when we want to fold

$$\neg \forall x . A \to B \to \bot \tag{6}$$

The result should be $\exists^{\mathsf{cl}} x . A \wedge B$, but this is not quite correct, since the latter formula unfolds into $\neg \forall x . A \wedge B \to \bot$. Therefore we make use of a connective called *tensor* (written $\otimes$) with the property that (6) folds into

$$\exists^{\mathsf{cl}} x . A \otimes B$$

and unfolds again into (6).

Formulas can be *unfolded* in the sense that all classical existential quantifiers are replaced according to their definiton

$$\exists^{\mathsf{ca}}\hat{x}_1 \dots \hat{x}_n.A_1 \otimes \dots \otimes A_m := (\forall \hat{x}_1 \dots \hat{x}_n.A_1 \to \dots \to A_m \to F) \to F$$

$$\exists^{\mathsf{cl}}\hat{x}_1 \dots \hat{x}_n.A_1 \otimes \dots \otimes A_m := (\forall \hat{x}_1 \dots \hat{x}_n.A_1 \to \dots \to A_m \to \bot) \to \bot$$

Inversely a formula can be *folded* in the sense that classical existential quantifiers are introduced wherever possible.

3.2. **Induction.** We now spell out what precisely we mean by induction over simultaneous free algebras $\vec{\mu} = \mu\vec{\alpha}\,\vec{\kappa}$, with goal formulas $\forall x_j^{\mu_j}\,\hat{P}_j(x_j)$. For the constructor type

$$\kappa_i = \vec{\rho} \to (\vec{\sigma}_1 \to \alpha_{j_1}) \to \dots \to (\vec{\sigma}_n \to \alpha_{j_n}) \to \alpha_j \in \mathsf{KT}(\vec{\alpha})$$

we have the *step formula*

$$\begin{aligned} D_i := \forall y_1^{\rho_1}, \dots, y_m^{\rho_m}, y_{m+1}^{\vec{\sigma}_1 \to \mu_{j_1}}, \dots, y_{m+n}^{\vec{\sigma}_n \to \mu_{j_n}}.\forall \vec{x}^{\vec{\sigma}_1}\,\hat{P}_{j_1}(y_{m+1}\vec{x}) \to \dots \to&\\ \forall \vec{x}^{\vec{\sigma}_n}\,\hat{P}_{j_n}(y_{m+n}\vec{x}) \to&\\ \hat{P}_j(\mathsf{constr}_i^{\vec{\mu}}(\vec{y})).& \end{aligned}$$

Here $\vec{y} = y_1^{\rho_1}, \dots, y_m^{\rho_m}, y_{m+1}^{\vec{\sigma}_1 \to \mu_{j_1}}, \dots, y_{m+n}^{\vec{\sigma}_n \to \mu_{j_n}}$ are the *components* of the object $\mathsf{constr}_i^{\vec{\mu}}(\vec{y})$ of type μ_j under consideration, and

$$\forall \vec{x}^{\vec{\sigma}_1}\,\hat{P}_{j_1}(y_{m+1}\vec{x}), \dots, \forall \vec{x}^{\vec{\sigma}_n}\,\hat{P}_{j_n}(y_{m+n}\vec{x})$$

are the hypotheses available when proving the induction step. The induction axiom $\mathsf{Ind}_{\mu_j}^{\vec{x},\vec{A}}$ with $\vec{x} = (x_j^{\mu_j})_{j=1,\dots,N}$ and $\vec{A} = (A_j)_{j=1,\dots,N} = (\hat{P}_j(x_j^{\mu_j}))_{j=1,\dots,N}$ then proves the formula

$$D_1 \to \dots \to D_k \to \forall x_j^{\mu_j}\,\hat{P}_j(x_j).$$

We will often write $\mathsf{Ind}_j^{\vec{x},\vec{A}}$ for $\mathsf{Ind}_{\mu_j}^{\vec{x},\vec{A}}$, and omit the upper indices $\vec{x}, \vec{A}$ when they are clear from the context. In case of a non-simultaneous free algebra, i.e. of type $\mu\alpha\,\kappa$, for $\mathsf{Ind}_{\mu}^{x,A}$ we normally write $\mathsf{Ind}_{x,A}$.

3.3. **Examples of Induction.**

$$\mathsf{Ind}_{p,A}\colon A[p := \mathsf{tt}] \to A[p := \mathsf{ff}] \to \forall p^{\mathbf{B}} A,$$

$$\mathsf{Ind}_{n,A}\colon A[n := 0] \to (\forall n.A \to A[n := \mathsf{S}n]) \to \forall n^{\mathbf{N}} A,$$

$$\mathsf{Ind}_{l,A}\colon A[l := \mathsf{Nil}] \to (\forall x, l.A \to A[l := \mathsf{Cons}(x, l)]) \to \forall l^{\mathbf{L}(\alpha)} A$$

$$\mathsf{Ind}_{x,A}\colon \forall y_1 A[x := \mathsf{Inl}(y_1)] \to \forall y_2 A[x := \mathsf{Inr}(y_2)] \to \forall x^{\rho_1 + \rho_2} A.$$

For the simultaneously defined algebras tree and tlist the induction axiom $\mathsf{Ind}_{\mathsf{tree}}^{b, bs, \hat{P}_1(b), \hat{P}_2(bs)}$ is

$$D_1 \to D_2 \to D_3 \to D_4 \to \forall b^{\mathsf{tree}}\hat{P}_1(b)$$

with

$$D_1 := \forall n \hat{P}_1(\mathsf{Leaf}(n)),$$

$$D_2 := \forall bs^{\mathsf{tlist}}.\hat{P}_2(bs) \to \hat{P}_1(\mathsf{Branch}(bs)),$$

$$D_3 := \hat{P}_2(\mathsf{Empty}),$$
$$D_4 := \forall b^{\mathsf{tree}}, bs^{\mathsf{tlist}}.\hat{P}_1(b) \to \hat{P}_2(bs) \to \hat{P}_2(\mathsf{Tcons}(b, bs)).$$

Simplified Simultaneous Induction. It may happen that in an induction on simultaneously defined algebras one only needs to induct on some of those algebras. Then we can simplify the induction formula accordingly, by

- omitting all step formulas D_i corresponding to constructor types with irrelevant value type τ_j, and
- simplifying the remaining step formulas by omitting from the induction hypotheses $\forall \vec{x}^{\vec{\sigma}_1} \hat{P}_{j_1}(y_{m+1}\vec{x}), \dots, \forall \vec{x}^{\vec{\sigma}_n} \hat{P}_{j_n}(y_{m+n}\vec{x})$ all those corresponding to constructor types with irrelevant value type τ_{j_ν}.

In the tree, tlist-example, if we only want to induct on tlist, then the step formulas are

$$D_3 := \hat{P}_2(\mathsf{Empty}),$$
$$D_4 := \forall b^{\mathsf{tree}}, bs^{\mathsf{tlist}}.\hat{P}_2(bs) \to \hat{P}_2(\mathsf{Tcons}(b, bs)).$$

Hence the simplified induction axiom is

$$\mathsf{Ind}_{bs,\hat{P}_2(bs)} \colon D_3 \to D_4 \to \forall bs^{\mathsf{tlist}} \hat{P}_2(bs).$$

3.4. **Cases.** There is an important variant of the induction axiom, where no induction hypotheses are used, i.e. all step formulas have the form

$$D_i := \forall y_1^{\rho_1}, \dots, y_m^{\rho_m}, y_{m+1}^{\vec{\sigma}_1 \to \mu_{j_1}}, \dots, y_{m+n}^{\vec{\sigma}_n \to \mu_{j_n}}.\hat{P}_j(\mathsf{constr}_i^{\vec{\mu}}(\vec{y})).$$

This variant is called the *cases axiom*; it distinguishes cases according to the outer constructor form. The formula of the cases axiom is

$$\mathsf{Cases}_{x_j,\hat{P}_j(x_j)} \colon D_{i_1}, \dots, D_{i_q} \to \forall x_j^{\mu_j} \hat{P}_j(x_j),$$

where $D_{i_1}, \dots, D_{i_q}$ consists of all D_i concerning constructors for μ_j.

Examples are

$$\mathsf{Cases}_{n,A} \colon A[n := 0] \to \forall n A[n := \mathsf{S}n] \to \ \forall n^{\mathbf{N}} A,$$
$$\mathsf{Cases}_{l,A} \colon A[l := \mathsf{Nil}] \to \forall x, l A[l := \mathsf{Cons}(x, l)] \to \forall l^{\mathbf{L}(\alpha)} A.$$

In the tree, tlist-example, if we want to distinguish cases on tlist, then the step formulas are

$$D_3 := \hat{P}(\mathsf{Empty}),$$
$$D_4 := \forall b^{\mathsf{tree}} \forall bs^{\mathsf{tlist}} \hat{P}(\mathsf{Tcons}(b, bs)).$$

Hence the cases axiom is

$$\mathsf{Cases}_{bs,\hat{P}(bs)} \colon \hat{P}(\mathsf{Empty}) \to \forall b^{\mathsf{tree}} \forall bs^{\mathsf{tlist}} \hat{P}(\mathsf{Tcons}(b, bs)) \to \forall bs^{\mathsf{tlist}} \hat{P}(bs).$$