CHAPTER 3

A theory TCF of partial continuous functionals

After getting clear about the domains we intend to reason about, the partial continuous functionals, we now set up a theory to prove their properties. The main concepts are those of inductively and coinductively defined predicates.

3.1. Formulas and their computational content

Formulas will built up from prime formulas $P\vec{t}$ by implication $\to$ and universal quantification $\forall_x$; here the t_i are terms, x is a variable and P is a predicate of a certain arity (a list of types). Types and terms are defined as in Chapter 2. We often write $\vec{t} \in P$ for $P\vec{t}$.

Predicates can be inductively or coinductively defined. An example for the former is $T_{\mathbb{L}(\mathbb{N})}$, which is defined by the clauses (i) $[] \in T_{\mathbb{L}(\mathbb{N})}$ and (ii) $\forall_{n\in\mathbb{N}}(\ell \in T_{\mathbb{L}(\mathbb{N})} \to n :: \ell \in T_{\mathbb{L}(\mathbb{N})})$. An example for the latter is ${}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ defined by a closure axiom saying that every $\ell \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ is of the form $n :: \ell'$ with $n \in \mathbb{N}$ and $\ell' \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ again. According to Kolmogorov (1932) a formula can be seen as a problem, asking for a solution. In the inductive example a solution for $4 :: 2 :: 0 :: [] \in T_{\mathbb{L}(\mathbb{N})}$ would be the generating (finite) sequence $[]$, $0 :: []$, $2 :: 0 :: []$, $4 :: 2 :: 0 :: []$, and in the coinductive example a solution for a prime formula $t \in {}^{\mathrm{co}}T_{\mathbb{L}(\mathbb{N})}$ would be an (infinite) stream of natural numbers. Generally, a solution for an inductive predicate is a finite construction tree, and for a coinductive predicate a finitely branching possibly infinite destruction tree. Such trees can be seen as ideals of the closed base types considered in Section 2.2.2. A solution for a problem posed by the formula $A \to B$ is a computable functional mapping solutions of A into solutions of B.

Sometimes the solution of a problem does not need all available input. We therefore mark the sources of such computationally superfluous input – that is, some (co)inductive predicates – as "non-computational" (n.c.).

Assume an infinite supply of predicate variables, each of its own *arity* (a list of types). We distinguish two sorts of predicate variables, "computationally relevant" ones $X^{\mathrm{c}}, Y^{\mathrm{c}}, Z^{\mathrm{c}}, W^{\mathrm{c}} \dots$ and "non-computational" ones X^{nc}, Y^{nc}, Z^{nc}, W^{nc} ..., and use $X, Y, Z, W \dots$ for both.

Let Z be a predicate variable. By $\bar{Z}$ we denote the result of applying the predicate variable Z to a list of terms of fitting types, and by $\tilde{Z}$ lists of those.

DEFINITION (Clauses and predicate forms). Let X be a predicate variable. An *X-clause* is a formula

$$K := \forall_{\vec{x}}(\tilde{Y}^{\mathrm{c}} \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}_i^{\mathrm{nc}} \to \bar{X}_i))_{i<n} \to \bar{X})$$

with all predicate variables Y_i^{c}, Z_i^{nc}, W_i^{nc} occurring exactly once and distinct from each other and from X, and all $\bar{X}_i$ coming from the fixed X. A premise of a clause is called a *parameter* premise if X does not occur in it, and a *recursive* premise otherwise. A clause K is *non-recursive* if it has no recursive premises.

Let $\vec{K}$ be a list of X-clauses. We call $I^{\mathrm{c}} := \mu_{X^{\mathrm{c}}}\vec{K}$ and $I^{\mathrm{nc}} := \mu_{X^{\mathrm{nc}}}\vec{K}$ (with $\vec{K}$ not empty) *predicate forms* (and use I for both), and similarly with ${}^{\mathrm{co}}I$ for I and ν for μ.

EXAMPLES. Recall that $\mathrm{Nil}^{\mathbb{L}(\alpha)}$ and $\mathrm{Cons}^{\alpha\to\mathbb{L}(\alpha)\to\mathbb{L}(\alpha)}$ are the two constructors of the base type $\mathbb{L}(\alpha)$ of lists, written $[]$ and $::$ (infix).

1. Let Y of arity (α) and X of arity $(\mathbb{L}(\alpha))$ be predicate variables. Then

$$K_0 := ([] \in X),$$
$$K_1 := \forall_x(x \in Y \to \forall_\ell(\ell \in X \to x :: \ell \in X))$$

are clauses and both *relative totality* $T_{\mathbb{L}}(Y)$ (or $T_{\mathbb{L},Y}$) defined by $\mu_X(K_0, K_1)$ and also *relative cototality* ${}^{\mathrm{co}}T_{\mathbb{L}}(Y)$ (or ${}^{\mathrm{co}}T_{\mathbb{L},Y}$) defined by $\nu_X(K_0, K_1)$ are predicate forms with Y a parameter predicate variable. Note that we can omit the type parameter α, since it can be read off from the arity of Y.

2. Alternatively let Y of arity (α, α) and X of arity $(\mathbb{L}(\alpha), \mathbb{L}(\alpha))$ be predicate variables. Then

$$K_0 := X([], []),$$
$$K_1 := \forall_{x,x'}(Y(x, x') \to \forall_{\ell,\ell'}(X(\ell, \ell') \to X(x :: \ell, x' :: \ell')))$$

are clauses and both *similarity* $\sim_{\mathbb{L}}(Y)$ defined by $\mu_X(K_0, K_1)$ and *bisimilarity* $\approx_{\mathbb{L}}(Y)$ defined by $\nu_X(K_0, K_1)$ are predicate forms with Y a parameter predicate variable.

Note that a predicate form I may contain type variables $\vec{\alpha}$ and predicate variables $\vec{Y}$. We write $I(\vec{\rho}, \vec{P})$ for the result of substituting in I the types $\vec{\rho}$ for $\vec{\alpha}$ and the predicates $\vec{P}$ for $\vec{Y}$.

DEFINITION (Constructor types of a predicate form). From every clause K we obtain a constructor type by

- omitting quantifiers,

- dropping all n.c. predicates and from the c.r. predicates their arguments, and
- replacing the remaining predicate variables by type variables.

That is, from the clause

$$\forall_{\vec{x}}(\tilde{Y}^{\mathrm{c}} \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}_i^{\mathrm{nc}} \to \bar{X}_i))_{i<n} \to \bar{X})$$

we obtain the constructor type $\vec{\alpha} \to (\xi)_{i<n} \to \xi$. With every predicate form $I^{\mathrm{c}} := (\mu/\nu)_{X^{\mathrm{c}}}\vec{K}$ we associate the list $\vec{\kappa}$ of constructor types.

Definition (Predicates and formulas).

$$\begin{aligned} P, Q &::= X \mid \{\, \vec{x} \mid A \,\} \mid I(\vec{\rho}, \vec{P}\,) \mid {}^{\mathrm{co}}I(\vec{\rho}, \vec{P}\,) && \text{(predicates)},\\ A, B &::= P\vec{t} \mid A \to B \mid \forall_x A && \text{(formulas)} \end{aligned}$$

with $I/{}^{\mathrm{co}}I$ a predicate form. To take care of the difference between X^{c} and X^{nc} we define the *final predicate* of a predicate or formula by

$$\begin{aligned} \mathrm{fp}(X) &:= X, & \mathrm{fp}(P\vec{t}\,) &:= \mathrm{fp}(P),\\ \mathrm{fp}(\{\, \vec{x} \mid A \,\}) &:= \mathrm{fp}(A), & \mathrm{fp}(A \to B) &:= \mathrm{fp}(B),\\ \mathrm{fp}((I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)) &:= I/{}^{\mathrm{co}}I, & \mathrm{fp}(\forall_x A) &:= \mathrm{fp}(A). \end{aligned}$$

We call a predicate or formula C *non-computational* (n.c., or *Harrop*) if its final predicate $\mathrm{fp}(C)$ is of the form X^{nc} or I^{nc}, else *computationally relevant* (c.r.). We require that all predicate substitutions involved in $(I/{}^{\mathrm{co}}I)(\vec{\rho}, \vec{P}\,)$ substitute c.r. predicates for c.r. predicate variables and n.c. predicates for n.c. predicate variables. Such predicate substitutions are called *sharp*.

Predicates of the form $I(\vec{\rho}, \vec{P}\,)$ are called *inductive*, and predicates of the form ${}^{\mathrm{co}}I(\vec{\rho}, \vec{P}\,)$ *coinductive*.

The terms $\vec{t}$ are those introduced in Section 2.3.1, i.e., typed terms built from typed variables and constants by abstraction and application, and (importantly) those with a common reduct are identified.

A predicate of the form $\{\, \vec{x} \mid C \,\}$ is called a *comprehension term*. We identify $\{\, \vec{x} \mid C(\vec{x}\,) \,\}\vec{t}$ with $C(\vec{t}\,)$. For a predicate C of arity $(\rho, \vec{\sigma}\,)$ we write Ct for $\{\, \vec{y} \mid Ct\vec{y} \,\}$.

It is a natural question to ask what the type of a "realizer" or "witness" of a c.r. predicate or formula C should be.

DEFINITION (Type $\tau(C)$ of a c.r. predicate or formula C). Assume a global injective assignment of type variables ζ to c.r. predicate variables X^{c}.

$$\begin{aligned} \tau(X^{\mathrm{c}}) &:= \zeta, & \tau(P\vec{t}\,) &:= \tau(P), \\ \tau(\{\,\vec{x} \mid A\,\}) &:= \tau(A), & \tau(A \to B) &:= \begin{cases} \tau(A) \to \tau(B) & (A \text{ c.r.}) \\ \tau(B) & (A \text{ n.c.}), \end{cases} \\ \tau(I(\vec{\rho}, \vec{P}\,)) &:= \iota_{\vec{\kappa}(\vec{\rho},\tau(\vec{P}^{\mathrm{c}}))}, & \tau(\forall_x A) &:= \tau(A). \end{aligned}$$

In the I-case we have assumed $I = (\mu/\nu)_X \vec{K}$ with X-clauses $\vec{K}$. Every K_i has an assigned constructor type κ_i. Free in $\vec{\kappa}$ are the type variables $\vec{\alpha}$ from $\vec{K}$ and the type variables $\vec{\zeta}$ globally assigned to the c.r. predicate variables $\vec{Y}^{\mathrm{c}}$ in $\vec{K}$. Now $\vec{\kappa}(\vec{\rho}, \tau(\vec{P}^{\mathrm{c}}))$ is the result of substituting $\vec{\rho}$ for $\vec{\alpha}$ and of the (already generated) types $\tau(P_i^{\mathrm{c}})$ for ζ_i in $\vec{\kappa}$.

3.2. Examples of inductive predicates

A simple example of an inductive predicate is totality $T_{\mathbb{N}}$ of the natural numbers. It is defined as

$$T_{\mathbb{N}} := \mu_X(K_0, K_1)$$

with

$$\begin{aligned} K_0 &:= (0 \in X), \\ K_1 &:= \forall_n(n \in X \to Sn \in X). \end{aligned}$$

Depending on whether the predicate variable X is n.c. or c.r. we have an n.c. or a c.r. totality predicate.

Recall that a variable of type τ ranges over arbitrary objects of type τ, which may be partial. However, in practice we ofter want to argue on total objects only. To make such a restriction easy to read we introduce two sorts of variable names: a general one written $\hat{x}$ ranging over arbitrary (possibly partial) objects, and a special one written x ranging over total objects only. Then we use the abbreviation

$$\forall_x A(x) := \forall_{\hat{x}}(\hat{x} \in T_\tau \to A(\hat{x})).$$

We will follow this convention from now on. Hence the clause K_1 above should now be written

$$K_1 := \forall_{\hat{n}}(\hat{n} \in X \to S\hat{n} \in X).$$

Another particularly important example of an inductive predicate is *Leibniz equality*, defined simply by

$$\mathrm{EqD} := \mu_{X^{\mathrm{nc}}}(\forall_{\hat{x}} X^{\mathrm{nc}} \hat{x}\hat{x}) \qquad (\mathrm{D} \text{ for "inductively } \mathbf{d}\text{efined"}).$$

We will use the abbreviation

$$(t \equiv s) := \mathrm{EqD}(t, s).$$

The missing logical connectives existence, disjunction and conjunction can also be defined inductively. Existence is defined inductively by

$$\begin{aligned}\mathrm{Ex}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\hat{x}}(\hat{x} \in Y^{\mathrm{c}} \to X^{\mathrm{c}})),\\ \mathrm{ExNc}_{Y} &:= \mu_{X^{\mathrm{nc}}}(\forall_{\hat{x}}(\hat{x} \in Y \to X^{\mathrm{nc}})).\end{aligned}$$

Then by definition

$$\tau(\mathrm{Ex}) = \mu_{\xi}(\beta \to \xi) = \mathbb{I}(\beta).$$

We use the abbreviation

$$\begin{aligned}\exists_{\hat{x}} A &:= \mathrm{Ex}_{\{\hat{x}|A\}},\\ \exists^{\mathrm{nc}}_{\hat{x}} A &:= \mathrm{ExNc}_{\{\hat{x}|A\}},\end{aligned}$$

and again since the decoration is determined by the c.r./n.c. status of the parameter predicate we usually leave out the decoration and just write $\exists$.

For a context where only total objects are of interest we have

$$\begin{aligned}\mathrm{ExDT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\hat{x}}(\hat{x} \in T^{\mathrm{c}} \to \hat{x} \in Y^{\mathrm{c}} \to X^{\mathrm{c}})),\\ \mathrm{ExLT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\hat{x}}(\hat{x} \in T^{\mathrm{c}} \to \hat{x} \in Y^{\mathrm{nc}} \to X^{\mathrm{c}})),\\ \mathrm{ExRT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\hat{x}}(\hat{x} \in T^{\mathrm{nc}} \to \hat{x} \in Y^{\mathrm{c}} \to X^{\mathrm{c}})),\\ \mathrm{ExNcT}_{Y} &:= \mu_{X^{\mathrm{nc}}}(\forall_{\hat{x}}(\hat{x} \in T \to \hat{x} \in Y \to X^{\mathrm{nc}})).\end{aligned}$$

Here D is for "double", L for "left" and R for "right". Then by definition

$$\begin{aligned}\tau(\mathrm{ExDT}) &= \mu_{\xi}(\tau \to \beta \to \xi) = \tau \times \beta\\ \tau(\mathrm{ExLT}) &= \mu_{\xi}(\tau \to \xi) \qquad = \mathbb{I}(\tau),\\ \tau(\mathrm{ExRT}) &= \mu_{\xi}(\beta \to \xi) \qquad = \mathbb{I}(\beta).\end{aligned}$$

To make these formulas more readable we can again use our convention concerning the two sorts $\hat{x}$ and x of variable names. Then the inductive predicates above are written as

$$\begin{aligned}\mathrm{ExDT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{x}(x \in Y^{\mathrm{c}} \to X^{\mathrm{c}})),\\ \mathrm{ExLT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{x}(x \in Y^{\mathrm{nc}} \to X^{\mathrm{c}})),\\ \mathrm{ExRT}_{Y^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall^{\mathrm{nc}}_{x}(x \in Y^{\mathrm{c}} \to X^{\mathrm{c}})),\\ \mathrm{ExNcT}_{Y} &:= \mu_{X^{\mathrm{nc}}}(\forall_{x}(x \in Y \to X^{\mathrm{nc}})).\end{aligned}$$

We use the abbreviations

$$\begin{array}{ll}
\exists^{\mathrm{d}}_x A := \mathrm{ExDT}_{\{x|A\}} & \text{if } A \text{ is c.r.,}\\
\exists^{\mathrm{l}}_x A := \mathrm{ExLT}_{\{x|A\}} & \text{if } A \text{ is n.c.,}\\
\exists^{\mathrm{r}}_x A := \mathrm{ExRT}_{\{x|A\}} & \text{if } A \text{ is c.r.,}\\
\exists^{\mathrm{nc}}_x A := \mathrm{ExNcT}_{\{x|A\}} & \text{for arbitrary } A.
\end{array}$$

Disjunction is a special case of union

$$\mathrm{Cup}_{Y,Z} := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y\vec{x} \to X^{\mathrm{c}}\vec{x}),\ \forall_{\vec{x}}(Z\vec{x} \to X^{\mathrm{c}}\vec{x})).$$

Since the parameter predicates Y, Z can be chosen as either c.r. or n.c. we obtain the variants

$$\begin{array}{ll}
\mathrm{CupD}_{Y^{\mathrm{c}},Z^{\mathrm{c}}} & := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x}),\ \forall_{\vec{x}}(Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})),\\
\mathrm{CupL}_{Y^{\mathrm{c}},Z^{\mathrm{nc}}} & := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x}),\ \forall_{\vec{x}}(Z^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x})),\\
\mathrm{CupR}_{Y^{\mathrm{nc}},Z^{\mathrm{c}}} & := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x}),\ \forall_{\vec{x}}(Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})),\\
\mathrm{CupU}_{Y^{\mathrm{nc}},Z^{\mathrm{nc}}} & := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x}),\ \forall_{\vec{x}}(Z^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x})),\\
\mathrm{CupNc}_{Y,Z} & := \mu_{X^{\mathrm{nc}}}(\forall_{\vec{x}}(Y\vec{x} \to X^{\mathrm{nc}}\vec{x}),\ \forall_{\vec{x}}(Z\vec{x} \to X^{\mathrm{nc}}\vec{x})).
\end{array}$$

Here D is for "double", L for "left", R for "right" and U for "uniform". Then by definition

$$\begin{array}{lll}
\tau(\mathrm{CupD}) = \mu_\xi(\beta_0 \to \xi, \beta_1 \to \xi) & = \beta_0 + \beta_1,\\
\tau(\mathrm{CupL}) = \mu_\xi(\beta \to \xi, \xi) & = \beta + \mathbb{U} = \mathtt{ysumu}(\beta),\\
\tau(\mathrm{CupR}) = \mu_\xi(\xi, \beta \to \xi) & = \mathbb{U} + \beta = \mathtt{uysum}(\beta),\\
\tau(\mathrm{CupU}) = \mu_\xi(\xi, \xi) & = \mathbb{B}.
\end{array}$$

We use the abbreviations

$$\begin{array}{ll}
P \cup^{\mathrm{d}} Q & := \mathrm{CupD}_{P,Q},\\
P \cup^{\mathrm{l}} Q & := \mathrm{CupL}_{P,Q},\\
P \cup^{\mathrm{r}} Q & := \mathrm{CupR}_{P,Q},\\
P \cup^{\mathrm{u}} Q & := \mathrm{CupU}_{P,Q},\\
P \cup^{\mathrm{nc}} Q & := \mathrm{CupNc}_{P,Q}.
\end{array}$$

In case of nullary predicates we use

$$\begin{aligned} A \vee^{\mathrm{d}} B &:= \mathrm{CupD}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{l}} B &:= \mathrm{CupL}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{r}} B &:= \mathrm{CupR}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{u}} B &:= \mathrm{CupU}_{\{|A\},\{|B\}}, \\ A \vee^{\mathrm{nc}} B &:= \mathrm{CupNc}_{\{|A\},\{|B\}}. \end{aligned}$$

Since the "decoration" is determined by the c.r./n.c. status of the two parameter predicates we usually leave it out in $\vee^{\mathrm{d}}, \vee^{\mathrm{l}}, \vee^{\mathrm{r}}, \vee^{\mathrm{u}}$ and just write $\vee$. However in the final nc-variant we suppress even the information which clause has been used, and hence must keep the notation $\vee^{\mathrm{nc}}$.

Similarly conjunction is a special case of intersection

$$\mathrm{Cap}_{Y,Z} := \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X^{\mathrm{c}}\vec{x})),$$

and we obtain the variants

$$\begin{aligned} \mathrm{CapD}_{Y^{\mathrm{c}},Z^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapL}_{Y^{\mathrm{c}},Z^{\mathrm{nc}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{c}}\vec{x} \to Z^{\mathrm{nc}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapR}_{Y^{\mathrm{nc}},Z^{\mathrm{c}}} &:= \mu_{X^{\mathrm{c}}}(\forall_{\vec{x}}(Y^{\mathrm{nc}}\vec{x} \to Z^{\mathrm{c}}\vec{x} \to X^{\mathrm{c}}\vec{x})), \\ \mathrm{CapNc}_{Y,Z} &:= \mu_{X^{\mathrm{nc}}}(\forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X^{\mathrm{nc}}\vec{x})). \end{aligned}$$

Then by definition

$$\begin{aligned} \tau(\mathrm{CapD}) = \mu_\xi(\beta_0 \to \beta_1 \to \xi) &= \beta_0 \times \beta_1 \\ \tau(\mathrm{CapL}) = \tau(\mathrm{CapR}) = \mu_\xi(\beta \to \xi) &= \mathbb{I}(\beta). \end{aligned}$$

We use the abbreviations

$$\begin{aligned} P \cap^{\mathrm{d}} Q &:= \mathrm{CapD}_{P,Q}, \\ P \cap^{\mathrm{l}} Q &:= \mathrm{CapL}_{P,Q}, \\ P \cap^{\mathrm{r}} Q &:= \mathrm{CapR}_{P,Q}, \\ P \cap^{\mathrm{nc}} Q &:= \mathrm{CapNc}_{P,Q}. \end{aligned}$$

In case of nullary predicates we use

$$\begin{aligned} A \wedge^{\mathrm{d}} B &:= \mathrm{CapD}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{l}} B &:= \mathrm{CapL}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{r}} B &:= \mathrm{CapR}_{\{|A\},\{|B\}}, \\ A \wedge^{\mathrm{nc}} B &:= \mathrm{CapNc}_{\{|A\},\{|B\}}. \end{aligned}$$

Again since the decoration is determined by the c.r./n.c. status of the two parameter predicates we usually leave out the decoration and just write $\wedge$.

3.3. Axioms of TCF

We define a theory of continuous functionals, called TCF. Formulas are the ones defined above, involving typed variables. Derivations use the rules of minimal logic for $\to$ and $\forall$, and the axioms introduced below. However, because of the distinction between n.c. and c.r. predicates and formulas we have an extra degree of freedom. By an *n.c. part* of a derivation we mean a subderivation with an n.c. end formula. Such n.c. parts will not contribute to the computational content of the whole derivation, and hence we can ignore all decorations in those parts (i.e., use a modified notion of equality of formulas there).

3.3.1. Axioms for inductive predicates. For each inductive predicate there are "clauses" or introduction axioms, together with a "least-fixed-point" or elimination axiom. To grasp the general form of these axioms it is convenient to write a clause

$$\forall_{\vec{x}}(\tilde{Y}^{\mathrm{c}} \to \tilde{Z}^{\mathrm{nc}} \to (\forall_{\vec{y}_i}(\tilde{W}_i^{\mathrm{nc}} \to \bar{X}_i))_{i<n} \to \bar{X}) \quad \text{as} \quad \forall_{\vec{x}}((A_\nu(X))_{\nu<n} \to X\vec{t}\,).$$

Definition (Introduction and elimination axioms for inductive predicates). For an inductive predicate $\mu_X(\forall_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i} \to X\vec{t}_i))_{i<k} =: I$ we have k introduction axioms I_i^+ $(i<k)$ and one elimination axiom I^-:

$$I_i^+ \colon \forall_{\vec{x}_i}((A_{i\nu}(I))_{\nu<n_i} \to I\vec{t}_i), \tag{12}$$

$$I^- \colon (\forall_{\vec{x}_i}((A_{i\nu}(I\cap X))_{\nu<n_i} \to X\vec{t}_i))_{i<k} \to I \subseteq X \tag{13}$$

($I\cap X$ was inductively defined above). (13) expresses that every competitor X satisfying the same clauses contains I. We take all substitution instances of I_i^+, I^- (w.r.t. substitutions for type and predicate variables) as axioms.

Remarks. (i) We use a "strengthened" form of the "step formula", namely $\forall_{\vec{x}_i}(A_{i\nu}(I\cap X))_{\nu<n_i} \to X\vec{t}_i$ rather than $\forall_{\vec{x}_i}(A_{i\nu}(X))_{\nu<n_i} \to X\vec{t}_i$. In applications of the least-fixed-point axiom this simplifies the proof of the "step", since we have an additional I-hypothesis available.

(ii) Notice that there is no circularity here for the inductive predicate $Y\cap Z := \mathrm{Cap}_{Y,Z}$, since there are no recursive calls in this particular inductive definition and hence $\cap$ does not occur in

$$\mathrm{Cap}^-_{Y,Z} \colon \forall_{\vec{x}}(Y\vec{x} \to Z\vec{x} \to X\vec{x}) \to \mathrm{Cap}_{Y,Z} \subseteq X.$$

(iii) The elimination axiom (13) could equivalently be written as

$$I^- \colon \forall_{\vec{x}}(I\vec{x} \to (\forall_{\vec{x}_i}((A_{i\nu}(I\cap X))_{\nu<n_i} \to X\vec{t}_i))_{i<k} \to X\vec{x}\,)$$

In this form it fits better with our (i.e., Gentzen's) way to write the logical elimination rules, where the main premise comes first. More importantly, its type (cf. Section 3.1) will then be the type of the recursion operator $\mathcal{R}^\tau_\iota$

taken as the "computational content" (cf. Section 4.1) of the elimination axiom for I. Therefore in the implementation of TCF this form is used. However, for readability we often prefer the form (13) in the present notes.

In Section 3.2 we considered several basic examples of inductive predicates. Their introduction and elimination axioms have important consequences, which we shall study now.

Totality $T_{\mathbb{N}}$ for the natural numbers has the introduction axioms

$$(T_{\mathbb{N}})_0^+ : 0 \in T_{\mathbb{N}},$$
$$(T_{\mathbb{N}})_1^+ : \forall_{\hat{n}}(\hat{n} \in T_{\mathbb{N}} \to S\hat{n} \in T_{\mathbb{N}}).$$

Its elimination axiom is

$$(T_{\mathbb{N}})^- : 0 \in X \to \forall_{\hat{n}}(\hat{n} \in T_{\mathbb{N}} \to \hat{n} \in X \to S\hat{n} \in X) \to \forall_{\hat{n}}(\hat{n} \in T_{\mathbb{N}} \to \hat{n} \in X)$$

or in abbreviated form (recall our convention on using both $\hat{n}$ and n as variable names)

$$(T_{\mathbb{N}})^- : 0 \in X \to \forall_n(n \in X \to Sn \in X) \to \forall_n(n \in X).$$

This is the usual induction axiom for (total) natural numbers.

The n.c. Leibniz equality has the introduction axiom

$$(\mathrm{EqD})_0^+ : \forall_{\hat{x}}(\hat{x} \equiv \hat{x}).$$

Its elimination axiom is

$$(\mathrm{EqD})^- : \forall_{\hat{x}} X\hat{x}\hat{x} \to \forall_{\hat{x},\hat{y}}(\hat{x} \equiv \hat{y} \to X\hat{x}\hat{y}).$$

From this definition we can deduce the property Leibniz used as a definition.

LEMMA 3.3.1 (Compatibility of EqD). $\forall_{\hat{x},\hat{y}}(\hat{x} \equiv \hat{y} \to A(\hat{x}) \to A(\hat{y}))$.

PROOF. By the elimination axiom with $X := \{\, \hat{x}, \hat{y} \mid A(\hat{x}) \to A(\hat{y}) \,\}$. □

Using compatibility of EqD one easily proves symmetry and transitivity.

An important usage of EqD in TCF is that it allows to introduce *falsity* and hence *negation*. Recall that the language of TCF contains constructors for base types. For the base type $\mathbb{B}$ of booleans we have as constructors Frege's "Wahrheitswerte" $\mathtt{tt}$ and $\mathtt{ff}$. Using these we can define

DEFINITION (Falsity, Negation). (a) Falsity $\mathbf{F}$ is defined by

$$\mathbf{F} := (\mathtt{ff} \equiv \mathtt{tt}).$$

(b) The negation $\neg A$ of a formula A is defined by

$$\neg A := (A \to \mathbf{F}).$$

Now using the fact that we identify terms with a common reduct and that we have recursion operators in our language we can prove "ex-falso-quodlibet" for formulas $I\vec{t}$ with I a predicate form. Easy examples are

LEMMA 3.3.2 (Ex-falso for EqD and $T_{\mathbb{N}}$). TCF *proves*

(a) $\mathbf{F} \to \forall_{\hat{x},\hat{y}}(\hat{x} \equiv \hat{y})$,
(b) $\mathbf{F} \to \forall_{\hat{n}}(\hat{n} \in T_{\mathbb{N}})$.

PROOF. (a) We show $\mathrm{Ef}_{\mathrm{EqD}} \colon \mathbf{F} \to \hat{x}^\tau \equiv \hat{y}^\tau$. To see this, we first obtain $\mathcal{R}^\tau_{\mathbb{B}}\mathtt{ff}\hat{x}\hat{y} \equiv \mathcal{R}^\tau_{\mathbb{B}}\mathtt{ff}\hat{x}\hat{y}$ from the introduction axiom. Then from $\mathtt{ff} \equiv \mathtt{tt}$ we get $\mathcal{R}^\tau_{\mathbb{B}}\mathtt{tt}\hat{x}\hat{y} \equiv \mathcal{R}^\tau_{\mathbb{B}}\mathtt{ff}\hat{x}\hat{y}$ by compatibility. Now $\mathcal{R}^\tau_{\mathbb{B}}\mathtt{tt}\hat{x}\hat{y}$ converts to $\hat{x}$ and $\mathcal{R}^\tau_{\mathbb{B}}\mathtt{ff}\hat{x}\hat{y}$ converts to $\hat{y}$. Hence $\hat{x} \equiv \hat{y}$, since we identify terms with a common reduct.

(b) We show $\mathrm{Ef}_{T_{\mathbb{N}}} \colon \mathbf{F} \to \hat{n} \in T_{\mathbb{N}}$. Assume $\mathbf{F}$. Then $\hat{n} \equiv 0$ by (a), hence $\hat{n} \in T_{\mathbb{N}}$ by $0 \in T_{\mathbb{N}}$ and compatibility. □

A similar result holds for arbitrary predicates and formulas. We postpone it until the axioms for coinductive predicates have been introduced.

An important use of Leibniz equality EqD is that it allows to turn a term t of type $\mathbb{B}$ into a formula $\mathrm{atom}(t)$, defined by

DEFINITION (Boolean terms as formulas).

$$\mathrm{atom}(t) := (t \equiv \mathtt{tt}).$$

This opens up a convenient way to deal with equality on closed base types. The computation rules ensure that, for instance, the boolean term $St =_{\mathbb{N}} Ss$, or more precisely $=_{\mathbb{N}}(St, Ss)$, is identified with $t =_{\mathbb{N}} s$. We can now turn this boolean term into the formula $(St =_{\mathbb{N}} Ss) \equiv \mathtt{tt}$, which again is abbreviated by $St =_{\mathbb{N}} Ss$, but this time with the understanding that it is a formula. Then (importantly) the two formulas $St =_{\mathbb{N}} Ss$ and $t =_{\mathbb{N}} s$ are identified because the latter is a reduct of the first. Consequently there is no need to prove the implication $St =_{\mathbb{N}} Ss \to t =_{\mathbb{N}} s$ explicitly.

Recall the inductive definitions of the logical connectives existence, disjunction and conjunction given above. For nullary predicates $P = \{ \mid A \}$ and $Q = \{ \mid B \}$ we write $A \vee B$ for $P \cup Q$ and $A \wedge B$ for $P \cap Q$. For simplicity we only consider the "double" versions. Then the introduction axioms are

$$\begin{aligned}
&\forall_x(A \to \exists^{\mathrm{d}}_x A),\\
&A \to A \vee^{\mathrm{d}} B, \qquad B \to A \vee^{\mathrm{d}} B,\\
&A \to B \to A \wedge^{\mathrm{d}} B,
\end{aligned}$$

and the elimination axioms are (now written in the equivalent form mentioned above, where the main premise comes first)

$$\begin{aligned}
&\exists^{\mathrm{d}}_x A \to \forall_x(A \to B) \to B \qquad (x \notin \mathrm{FV}(B)),\\
&A \vee^{\mathrm{d}} B \to (A \to C) \to (B \to C) \to C,\\
&A \wedge^{\mathrm{d}} B \to (A \to B \to C) \to C.
\end{aligned}$$

3.3.2. Axioms for coinductive predicates. For each coinductive predicate there is a closure axiom, together with a "greatest-fixed-point" axiom. For example, for the base type $\mathbb{Y}$ of of binary trees

- the cototality predicate ${}^{\mathrm{co}}T_{\mathbb{Y}}$ is defined by the closure axiom (3) (page 18) and the greatest-fixed-point axiom by (4) (page 18), and
- the bisimilarity predicate $\approx_{\mathbb{Y}}$ by the closure axiom (7) (page 19) and the greatest-fixed-point axiom (8) (page 19).

To understand the general axioms for coinductive predicates note that the conjunction of the k clauses (12) of an inductive predicate I is equivalent to

$$\forall_{\vec{x}}(\bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I)\land\vec{x}\equiv\vec{t}_i)\to I\vec{x}\,).$$

Definition (Closure and greatest-fixed-point axioms). For an inductive predicate $\mu_X(\forall_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i}\to X\vec{t}_i))_{i<k} =: I$ we define its dual ${}^{\mathrm{co}}I$ (with ν for μ) by the *closure axiom* ${}^{\mathrm{co}}I^-$ and the *greatest-fixed-point axiom* ${}^{\mathrm{co}}I^+$:

$$(14)\qquad {}^{\mathrm{co}}I^-\colon \forall_{\vec{x}}({}^{\mathrm{co}}I\vec{x}\to\bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I)\land\vec{x}\equiv\vec{t}_i)),$$

$$(15)\qquad {}^{\mathrm{co}}I^+\colon \forall_{\vec{x}}(X\vec{x}\to\bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I\cup X)\land\vec{x}\equiv\vec{t}_i))\to X\subseteq{}^{\mathrm{co}}I.$$

(${}^{\mathrm{co}}I\cup X$ was inductively defined above). The axiom expresses that every "competitor" X satisfying the closure axiom is contained in ${}^{\mathrm{co}}I$. We take all substitution instances of ${}^{\mathrm{co}}I^+$, ${}^{\mathrm{co}}I^-$ (w.r.t. substitutions for type and predicate variables) as axioms.

Again we have used a "strengthened" form of the "step formula", with $A_{i\nu}({}^{\mathrm{co}}I\cup X)$ rather than $A_{i\nu}(X)$. In applications of the greatest-fixed-point axiom this simplifies the proof of the "step", since its conclusion is weaker.

Remark. The greatest-fixed-point axiom (15) could be written as

$$\forall_{\vec{x}}(X\vec{x}\to\forall_{\vec{x}}(X\vec{x}\to\bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I\cup X)\land\vec{x}\equiv\vec{t}_i))\to{}^{\mathrm{co}}I\vec{x}).$$

Then its type will be the type of the corecursion operator ${}^{\mathrm{co}}\mathcal{R}^\tau_\iota$ taken as the "computational content" (cf. Section 4.1) of the greatest-fixed-point axiom for ${}^{\mathrm{co}}I$. Therefore in the implementation of TCF this form is used. However, for readability we prefer the form (15) in the present notes.

Remark. Instead of Leibniz equality $\equiv$ in (14) and (15) we could also use a different equality relation, for instance the n.c. variant $\doteq^{\mathrm{nc}}$ of pointwise equality to be introduced in Section 3.4. This leads to a new variant of ${}^{\mathrm{co}}I$.

EXAMPLES. (1) To show how to construct the dual ${}^{\mathrm{co}}I$ of an inductive predicate I we consider the predicate Even, which is defined by the clauses

$$(\mathrm{Even})^+_0 : 0 \in \mathrm{Even},$$
$$(\mathrm{Even})^+_1 : \forall_n(n \in \mathrm{Even} \to S(Sn) \in \mathrm{Even}).$$

The conjunction of its two clauses is equivalent to

$$\forall_n(n \equiv 0 \vee \exists_{n'}(n' \in \mathrm{Even} \wedge n \equiv S(Sn')) \to n \in \mathrm{Even}).$$

Now the dual ${}^{\mathrm{co}}\mathrm{Even}$ of Even is defined by its closure axiom ${}^{\mathrm{co}}\mathrm{Even}^-$:

$$\forall_n(n \in {}^{\mathrm{co}}\mathrm{Even} \to n \equiv 0 \vee \exists_{n'}(n' \in {}^{\mathrm{co}}\mathrm{Even} \wedge n \equiv S(Sn')))$$

and its greatest-fixed-point axiom ${}^{\mathrm{co}}\mathrm{Even}^+$:

$$\forall_n(Xn \to n \equiv 0 \vee \exists_{n'}(n' \in ({}^{\mathrm{co}}\mathrm{Even} \cup X) \wedge n \equiv S(Sn'))) \to X \subseteq {}^{\mathrm{co}}\mathrm{Even}.$$

(2) Consider the inductive predicate I of arity $(\mathbb{R})$ defined by the clause

$$\forall_{d,x',x}(d \in \mathbb{D} \wedge x' \in \mathbb{R} \wedge |x'| \le_{\mathbb{R}} 1 \wedge x' \in I \wedge x =_{\mathbb{R}} \frac{x'+d}{2} \to x \in I).$$

Here it is assumed that the real numbers $\mathbb{R}$ together with the relations $=_{\mathbb{R}}$ and $\le_{\mathbb{R}}$ are available. $\mathbb{D}$ is the base type of signed digits $\{-1, 0, 1\}$. The dual ${}^{\mathrm{co}}I$ is defined by its closure axiom ${}^{\mathrm{co}}I^-$:

$$\forall_x(x \in {}^{\mathrm{co}}I \to$$
$$\exists^{\mathrm{r}}_{d,x',y}(d \in \mathbb{D} \wedge x' \in \mathbb{R} \wedge |x'| \le_{\mathbb{R}} 1 \wedge x' \in {}^{\mathrm{co}}I \wedge y =_{\mathbb{R}} \frac{x'+d}{2} \wedge x =_{\mathbb{R}} y)$$

and its greatest-fixed-point (or coinduction) axiom ${}^{\mathrm{co}}I^+$:

$$\forall_x(x \in X \to \exists^{\mathrm{r}}_{d,x',y}($$
$$d \in \mathbb{D} \wedge x' \in \mathbb{R} \wedge |x'| \le_{\mathbb{R}} 1 \wedge (x' \in {}^{\mathrm{co}}I \cup X) \wedge y =_{\mathbb{R}} \frac{x'+d}{2} \wedge x =_{\mathbb{R}} y)) \to$$
$$X \subseteq {}^{\mathrm{co}}I).$$

REMARK. For n.c. inductive or coinductive predicates the axioms are formed as in the c.r. case, using $\vee^{\mathrm{nc}}$ for the closure axiom of ${}^{\mathrm{co}}I^{\mathrm{nc}}$. But there is an important restriction: for I^{nc} with more than one clause the elimination axiom $(I^{\mathrm{nc}})^-$ can only be used with a *non-computational* competitor predicate. This is needed in the proof of the soundness theorem. However, this restriction does not apply to I^{nc} defined by one clause only. Important examples of such *one-clause-nc* inductive predicates are Leibniz equality and the non-computational variants of the existential quantifier and of conjunction.

Generally, an inductive predicate is always contained in its dual.

LEMMA 3.3.3. $I \subseteq {}^{\mathrm{co}}I$, $I^{\mathrm{nc}} \subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$.

PROOF. The least-fixed-point axiom (13) for I (i.e., I^-) is equivalent to

$$\forall_{\vec{x}}(\bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I\cap X)\land \vec{x}\equiv\vec{t}_i)\to X\vec{x}\,)\to I\subseteq X.$$

It suffices that its premise holds with ${}^{\mathrm{co}}I$ for X. This follows from the greatest-fixed-point axiom (15) (i.e., ${}^{\mathrm{co}}I^+$), with the competitor predicate

$$X:=\{\,\vec{x}\mid \bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}(I\cap {}^{\mathrm{co}}I)\land \vec{x}\equiv\vec{t}_i)\,\}.$$

This means that we have to show the premise of (15) with this X, i.e.,

$$\forall_{\vec{x}}(X\vec{x}\to \bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I\cup X)\land \vec{x}\equiv\vec{t}_i)).$$

But if we unfold the premise $X\vec{x}$, this follows from $I\cap {}^{\mathrm{co}}I\subseteq {}^{\mathrm{co}}I\cup X$. For I^{nc} the proof is similar. □

REMARK. In case of an inductive predicate with non-recursive clauses only also the reverse inclusions ${}^{\mathrm{co}}I\subseteq I$, ${}^{\mathrm{co}}I^{\mathrm{nc}}\subseteq I^{\mathrm{nc}}$. Hence it is not necessary to consider ${}^{\mathrm{co}}I$. Examples are the inductively defined logical connectives $\exists$, $\vee$, $\wedge$ and Leibniz equality.

LEMMA 3.3.4. $I\subseteq I^{\mathrm{nc}}$, ${}^{\mathrm{co}}I\subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$.

PROOF. Let $I:=\mu_X(\forall_{\vec{x}_i}((A_{i\nu}(X))_{\nu<n_i}\to X\vec{t}_i))_{i<k}$.

For $I\subseteq I^{\mathrm{nc}}$ we use the elimination axiom (13) with I^{nc} as competitor predicate:

$$(\forall_{\vec{x}_i}((A_{i\nu}(I\cap I^{\mathrm{nc}}))_{\nu<n_i}\to I^{\mathrm{nc}}\vec{t}_i))_{i<k}\to I\subseteq I^{\mathrm{nc}}.$$

It suffices to prove the premises. Let $i<k$, fix $\vec{x}_i$ and assume $A_{i\nu}(I\cap I^{\mathrm{nc}})$ for all $\nu<n_i$. Since $A_{i\nu}(X)$ is strictly positive in X we obtain $A_{i\nu}(I^{\mathrm{nc}})$ for all $\nu<n_i$ and hence $I^{\mathrm{nc}}\vec{x}_i$ by $(I^{\mathrm{nc}})^+_i$.

For ${}^{\mathrm{co}}I\subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}$ we use the greatest-fixed-point axiom for ${}^{\mathrm{co}}I^{\mathrm{nc}}$ with ${}^{\mathrm{co}}I$ as competitor predicate:

$$\forall_{\vec{x}}({}^{\mathrm{co}}I\vec{x}\to \bigvee_{i<k}\exists_{\vec{x}_i}(\bigwedge_{\nu<n_i} A_{i\nu}({}^{\mathrm{co}}I^{\mathrm{nc}}\cup {}^{\mathrm{co}}I)\land \vec{x}\equiv\vec{t}_i))\to {}^{\mathrm{co}}I\subseteq {}^{\mathrm{co}}I^{\mathrm{nc}}.$$

It suffices to prove the premise, which again follows from the fact that $A_{i\nu}(X)$ is strictly positive in X. □

Now we are ready to generalize Lemma 3.3.2 on Ex-falso for EqD and $T_{\mathbb{N}}$ to arbitrary predicates and formulas. However, we have to take care of the following issues:

- Predicate variables might occur as non "strictly positive parts".
- (Co)inductive predicates as strictly positive parts might not have a nullary clause.

DEFINITION. The notion of a strictly positive part (s.p.p) of a predicate or formula C is defined inductively.

(a) C is a strictly positive part of C.
(b) If I is a strictly positive part of C and A a premise of a non-recursive clause of I then A is a strictly positive part of C.
(c) If $\{\,\vec{x} \mid A(\vec{x})\,\}$ is a strictly positive part of C then so is $A(\vec{t}\,)$.
(d) If $A \to B$ is a strictly positive part of C then so is B.
(e) If $\forall_{\hat{x}} A(\hat{x})$ is a strictly positive part of C then so is $A(t)$.

THEOREM 3.3.5 (Ex-falso-quodlibet). *Let C be a predicate or formula.* TCF *proves*

$$\begin{cases} \forall_{\vec{\hat{x}}}(\mathbf{F} \to P\vec{\hat{x}}\,) & \text{if } C \text{ is a predicate } P \\ \mathbf{F} \to A & \text{if } C \text{ is a formula } A \end{cases}$$

from assumptions

$$\begin{cases} \forall_{\vec{\hat{x}}}(\mathbf{F} \to Y\vec{\hat{x}}\,) & \text{if } Y \text{ is a predicate variable strictly positive in } C \\ \forall_{\vec{\hat{x}}}(\mathbf{F} \to I\vec{\hat{x}}\,) & \text{if } I \text{ has no non-recursive clause and is a s.p.p. of } C \end{cases}$$

PROOF. By Lemma 3.3.2 we have $\mathrm{Ef}_{\mathrm{EqD}} \colon \mathbf{F} \to \hat{x}^\tau \equiv \hat{y}^\tau$. The claim can now be proved by induction on C.

Case $I\vec{s}$. If I has no non-recursive clause we can use the assumption $\forall_{\vec{\hat{x}}}(\mathbf{F} \to I\vec{\hat{x}}\,)$. Otherwise let K_i be a non-recursive clause, with final conclusion $I\vec{t}$. By induction hypothesis from $\mathbf{F}$ we can derive all parameter premises. Hence $I\vec{t}$. From $\mathbf{F}$ we also obtain $s_i \equiv t_i$, by the remark above. Hence $I\vec{s}$ by compatibility.

Case ${}^{\mathrm{co}}I\vec{s}$. Use Lemma 3.3.3. The cases $Y\vec{s}$, $A \to B$ and $\forall_{\hat{x}} A$ are obvious. □

3.4. Equality and extensionality

Equality at closed base types of level 0 is easy to handle. For simplicity we only consider the type $\mathbb{Y}$ of binary trees. Recall that our theory TCF has an intended model, determined by the ideals of the information systems $\boldsymbol{A}_\tau$. We have seen in the Bisimilarity Lemma 2.2.1 (on page 19) that for closed base types bisimilarity implies equality, which in TCF is formalized by the inductively defined Leibniz equality EqD. Therefore we take as an axiom:

AXIOM (Bisimilarity). *For every closed base type bisimilarity implies Leibniz equality.*

This axiom is justified by the fact that it holds in our intended model. As a consequence we can prove in TCF

PROPOSITION 3.4.1 (Characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$).

(a) $\forall_{x,x'}(x \sim_{\mathbb{Y}} x' \leftrightarrow x, x' \in T_{\mathbb{Y}} \wedge x \equiv x')$.
(b) $\forall_{x,x'}(x \approx_{\mathbb{Y}} x' \leftrightarrow x, x' \in {}^{\mathrm{co}}T_{\mathbb{Y}} \wedge x \equiv x')$.

PROOF. (b). The proof of Proposition 2.2.2 relies on Lemma 2.2.1, which we just added as an axiom. The rest of the proof uses poperties of ${}^{\mathrm{co}}T_{\mathbb{Y}}$ and $\approx_{\mathbb{Y}}$ available in TCF.

(a). Similar to (b), using $T^{\pm}_{\mathbb{Y}}$, $\sim^{\pm}_{\mathbb{Y}}$ instead. For the proof of $x \sim_{\mathbb{Y}} x' \to x \equiv x'$ use (b) and $\sim_{\mathbb{Y}} \subseteq \approx_{\mathbb{Y}}$ (which follows from Lemma 3.3.3). □

This characterization of equality at $T_{\mathbb{Y}}$ and ${}^{\mathrm{co}}T_{\mathbb{Y}}$ is useful because it gives us a tool (induction, coinduction) to prove equalities $t \equiv t'$, which otherwise would be difficult.

COROLLARY 3.4.2.
(a) $\forall_x(x \sim_{\mathbb{Y}} x \leftrightarrow x \in T_{\mathbb{Y}})$.
(b) $\forall_x(x \approx_{\mathbb{Y}} x \leftrightarrow x \in {}^{\mathrm{co}}T_{\mathbb{Y}})$.

PROOF. Immediate from Proposition 3.4.1. □

COROLLARY 3.4.3.
(a) $\sim_{\mathbb{Y}}$ *is an equivalence relation on* $T_{\mathbb{Y}}$.
(b) $\approx_{\mathbb{Y}}$ *is an equivalence relation on* ${}^{\mathrm{co}}T_{\mathbb{Y}}$.

PROOF. Immediate from Proposition 3.4.1. □

REMARK. For closed base types like $\mathbb{Y}$ we can also relate $\sim_{\mathbb{Y}}$ to the binary boolean-valued function $=_{\mathbb{Y}}\colon \mathbb{Y} \to \mathbb{Y} \to \mathbb{B}$ defined in Section 2.3.1. One easily proves that

$$\forall_x(x \in T_{\mathbb{Y}} \to x = x),$$
$$\forall_x(x \in T_{\mathbb{Y}} \to \forall_y(y \in T_{\mathbb{Y}} \to x = y \to x \sim_{\mathbb{Y}} y)).$$

Usage of $=_{\mathbb{Y}}$ has the advantage that proofs may become shorter, since we identify terms with a common reduct. Pointwise equality $\doteq_{\mathbb{Y}}$ is defined to be $\sim_{\mathbb{Y}}$.

Up to now we have mainly dealt with base types. However, our theory TCF allows function types as well. We extend the notions of totality and pointwise equality from base types to function types. For simplicity we only consider parameter-free types.

DEFINITION (Totality and pointwise equality for function types).

$$(f \in T_{\tau\to\sigma}) := \forall_x(x \in T_\tau \to fx \in T_\sigma),$$
$$(f \doteq_{\tau\to\sigma} g) := f, g \in T_{\tau\to\sigma} \wedge \forall_{x,y}(x \doteq_\tau y \to fx \doteq_\sigma gy).$$

Extensionality is defined as diagonalization of pointwise equality:

DEFINITION (Extensionality).

$$(x \in \mathrm{Ext}_\tau) := (x \doteq_\tau x).$$

EXAMPLE (A non-extensional functional). Define f, g of type $\mathbb{N} \to \mathbb{N}$ by the computation rules $fn = 0$ and $g0 = 0$, $g(Sn) = gn$. Then $f\bot_{\mathbb{N}} = 0$ because of the computation rules for f. For $g\bot_{\mathbb{N}}$ no computation rule fits, but because of the inductive definition of $(\vec{U}, a) \in [\![\lambda_{\vec{x}} M]\!]$ in Section 2.4 (page 89) $[\![g\bot_{\mathbb{N}}]\!]$ is the empty ideal $[\![\bot_{\mathbb{N}}]\!]$. Hence $f \doteq g$, i.e., $f, g \in T_{\mathbb{N}\to\mathbb{N}}$ and $\forall_{n,m}(n \doteq_{\mathbb{N}} m \to fn \doteq_{\mathbb{N}} gm)$. The latter holds since $n \doteq_{\mathbb{N}} m$ implies $n \in T_{\mathbb{N}}$ and $n \equiv m$. Therefore the functional F defined by $Fh = h\bot_{\mathbb{N}}$ maps the pointwise equal f, g to different values.

By Corollary 3.4.2 (page 45) we know the equivalence of $\mathrm{Ext}_{\mathbb{Y}}$ and $T_{\mathbb{Y}}$; this also holds for arbitrary closed base types. This equivalence can be extended to closed types of level 1:

LEMMA 3.4.4. *The predicates* Ext_τ *and* T_τ *are equivalent for closed types of level* ≤ 1.

PROOF. For closed base types this has been proved in Corollary 3.4.2 (for the special case of the base type $\mathbb{Y}$). In case of level 1 we use induction on the height of the type, defined by

$$|\tau \to \sigma| := 1 + \max\{|\tau|, |\sigma|\}$$

Let $\tau \to \sigma$ be a closed type of level 1. The following are equivalent.

$f \in \mathrm{Ext}_{\tau\to\sigma}$
$f \doteq_{\tau\to\sigma} f$
$\forall_{x,y}(x \doteq_\tau y \to fx \doteq_\sigma fy)$
$\forall_{x\in T_\tau}(fx \doteq_\sigma fx)$ by Corollary 3.4.2, since $\mathrm{lev}(\tau) = 0$
$\forall_{x\in T_\tau}(fx \in \mathrm{Ext}_\sigma)$.

By induction hypothesis the final formula is equivalent to $f \in T_{\tau\to\sigma}$. □

For arbitrary closed types τ the relation $\doteq_\tau$ is a "partial equivalence relation", which means the following.

LEMMA 3.4.5. *For every closed type* τ *the relation* $\doteq_\tau$ *is an equivalence relation on* Ext_τ.

PROOF. Exercise.

□

LEMMA 3.4.6 (Compatibility of terms). *For every term* $t(\vec{x})$ *with extensional constants and free variables among* $\vec{x}$ *we have*

$$\forall_{\vec{x},\vec{y}}(\vec{x} \doteq_{\vec{\rho}} \vec{y} \to t(\vec{x}) \doteq_\tau t(\vec{y})).$$

PROOF. This is proved by induction on t. *Case* x. Immediate. *Case* c. By assumption $c \doteq_\tau c$. *Case* $\lambda_x t(x, \vec{x})$. Let $\vec{x} \doteq_{\vec{\rho}} \vec{y}$. The goal is $\lambda_x t(x, \vec{x}) \doteq_{\tau\to\sigma} \lambda_x t(x, \vec{y})$, which by definition means

$$\forall_{x,y}(x \doteq_\tau y \to t(x, \vec{x}) \doteq_\sigma t(y, \vec{y})).$$

Assume $x \doteq_\tau y$. With $\vec{x} \doteq_{\vec{\rho}} \vec{y}$ the claim $t(x, \vec{x}) \doteq_\sigma t(y, \vec{y})$ holds by the IH. *Case* $t(\vec{x})s(\vec{x})$. Let $\vec{x} \doteq_{\vec{\rho}} \vec{y}$. By IH we have $t(\vec{x}) \doteq_{\tau\to\sigma} t(\vec{y})$, i.e.,

$$\forall_{x,y}(x \doteq_\tau y \to t(\vec{x})x \doteq_\sigma t(\vec{y})y).$$

Again by IH we have $s(\vec{x}) \doteq_\tau s(\vec{y})$. Hence $t(\vec{x})s(\vec{x}) \doteq_\sigma t(\vec{y})s(\vec{y})$. □

LEMMA 3.4.7 (Extensionality of terms). *For every term* $t(\vec{x})$ *with extensional constants and free variables among* $\vec{x}$ *we have*

$$\forall_{\vec{x}}(\vec{x} \in \mathrm{Ext}_{\vec{\rho}} \to t(\vec{x}) \in \mathrm{Ext}_\tau).$$

PROOF. Let $t(\vec{x})$ with free variables among $\vec{x}$ be given, and assume $\vec{x} \in \mathrm{Ext}_{\vec{\rho}}$. By Lemma 3.4.6 applied to $\vec{x}, \vec{x}$ we obtain $t(\vec{x}) \doteq_\tau t(\vec{x})$, hence $t(\vec{x}) \in \mathrm{Ext}_\tau$. □