

CHAPTER 2

Computable functionals

The objects studied in mathematics have types, which in many cases are function types, possibly of a higher type. We are particularly interested in computable such objects. Their domains are partial continuous functionals (Scott, Ersov) which we study first, via Scott's information systems.

2.1. Information systems

The basic idea of information systems is to provide an axiomatic setting to describe approximations of abstract objects (like functions or functionals) by concrete, finite ones. We do not attempt to analyze the notion of “concreteness” or finiteness here, but rather take an arbitrary countable set A of “bits of data” or “tokens” as a basic notion to be explained axiomatically. In order to use such data to build approximations of abstract objects, we need a notion of “consistency”, which determines when the elements of a finite set of tokens are consistent with each other. We also need an “entailment relation” between consistent sets U of data and single tokens a , which intuitively expresses the fact that the information contained in U is sufficient to compute the bit of information a . The axioms below are a minor modification of Scott's (1982), due to Larsen and Winskel (1991).

2.1.1. Ideals.

DEFINITION. An *information system* is a structure $(A, \text{Con}, \vdash)$ where A is an at most countable non-empty set (the *tokens*), Con is a set of finite subsets of A (the *consistent* sets) and $\vdash$ is a subset of $\text{Con} \times A$ (the *entailment* relation), which satisfy

$$\begin{aligned} U \subseteq V \in \text{Con} &\rightarrow U \in \text{Con}, \\ \{a\} &\in \text{Con}, \\ U \vdash a &\rightarrow U \cup \{a\} \in \text{Con}, \\ a \in U \in \text{Con} &\rightarrow U \vdash a, \\ U \in \text{Con} &\rightarrow \forall_{a \in V} (U \vdash a) \rightarrow V \vdash b \rightarrow U \vdash b. \end{aligned}$$

The elements of Con are called *formal neighborhoods*. We use U, V, W to denote *finite* sets, and write

$$\begin{aligned} U \vdash V & \text{ for } U \in \text{Con} \wedge \forall_{a \in V} (U \vdash a), \\ a \uparrow b & \text{ for } \{a, b\} \in \text{Con} \quad (a, b \text{ are consistent}), \\ U \uparrow V & \text{ for } \forall_{a \in U, b \in V} (a \uparrow b). \end{aligned}$$

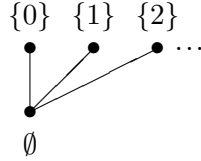
DEFINITION. The *ideals* (also called *objects*) of an information system $\mathbf{A} = (A, \text{Con}, \vdash)$ are defined to be those subsets x of A which satisfy

$$\begin{aligned} U \subseteq x & \rightarrow U \in \text{Con} \quad (x \text{ is consistent}), \\ U \vdash a & \rightarrow U \subseteq x \rightarrow a \in x \quad (x \text{ is deductively closed}). \end{aligned}$$

We write $x \in |\mathbf{A}|$ to mean that x is an ideal of $\mathbf{A}$.

EXAMPLES. The *deductive closure* $\overline{U} := \{a \in A \mid U \vdash a\}$ of $U \in \text{Con}$ is an ideal.

Every countable set A can be turned into a “flat” information system by letting the set of tokens be A , $\text{Con} := \{\emptyset\} \cup \{\{a\} \mid a \in A\}$ and $U \vdash a$ mean $a \in U$. In this case the ideals are just the elements of Con . For $A = \mathbb{N}$ we have the following picture of the Con -sets.



A rather important example is the following, which concerns approximations of functions from a countable set A into a countable set B . The tokens are the pairs (a, b) with $a \in A$ and $b \in B$, and

$$\begin{aligned} \text{Con} &:= \{ \{ (a_i, b_i) \mid i < k \} \mid \forall_{i,j < k} (a_i = a_j \rightarrow b_i = b_j) \}, \\ U \vdash (a, b) &:= (a, b) \in U. \end{aligned}$$

It is easy to verify that this defines an information system whose ideals are (the graphs of) all partial functions from A to B .

REMARK. One can show that for an arbitrary information system $\mathbf{A} = (A, \text{Con}, \vdash)$ the structure $(|\mathbf{A}|, \subseteq, \overline{})$ is a “domain” (also called Scott-Ershov domain, or “bounded complete algebraic cpo”), whose set of “compact elements” can be represented as $|\mathbf{A}|_c = \{ \overline{U} \mid U \in \text{Con} \}$. The converse holds as well: every countable domain can be represented as an information system. We will not need this relation to standard (non-constructive) domain theory (cf. Abramsky and Jung (1994)), and hence not even define these notions here.

2.1.2. Function spaces. We define the “function space” $\mathbf{A} \rightarrow \mathbf{B}$ between two information systems $\mathbf{A}$ and $\mathbf{B}$.

DEFINITION. Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ be information systems. Define $\mathbf{A} \rightarrow \mathbf{B} = (C, \text{Con}, \vdash)$ by

$$C := \text{Con}_A \times B,$$

$$\{(U_i, b_i) \mid i \in I\} \in \text{Con} := \forall J \subseteq I \left(\bigcup_{j \in J} U_j \in \text{Con}_A \rightarrow \{b_j \mid j \in J\} \in \text{Con}_B \right).$$

For the definition of the entailment relation $\vdash$ it is helpful to first define the notion of an *application* of $W := \{(U_i, b_i) \mid i \in I\} \in \text{Con}$ to $U \in \text{Con}_A$:

$$\{(U_i, b_i) \mid i \in I\}U := \{b_i \mid U \vdash_A U_i\}.$$

From the definition of Con we know that this set is in Con_B . Now define $W \vdash (U, b)$ by $WU \vdash_B b$.

REMARK. Clearly application is *monotone in the second argument*, in the sense that $U \vdash_A U'$ implies $(WU' \subseteq WU, \text{ hence also } WU \vdash_B WU')$. In fact, application is also *monotone in the first argument*, i.e.,

$$W \vdash W' \quad \text{implies} \quad WU \vdash_B W'U.$$

To see this let $W = \{(U_i, b_i) \mid i \in I\}$ and $W' = \{(U'_j, b'_j) \mid j \in J\}$. By definition $W'U = \{b'_j \mid U \vdash_A U'_j\}$. Now fix j such that $U \vdash_A U'_j$; we must show $WU \vdash_B b'_j$. By assumption $W \vdash (U'_j, b'_j)$, hence $WU'_j \vdash_B b'_j$. Because of $WU \supseteq WU'_j$ the claim follows.

LEMMA 2.1.1. *If $\mathbf{A}$ and $\mathbf{B}$ are information systems, then so is $\mathbf{A} \rightarrow \mathbf{B}$ defined as above.*

PROOF. Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$. The first, second and fourth property of the definition are clearly satisfied. For the third, suppose

$$\{(U_1, b_1), \dots, (U_n, b_n)\} \vdash (U, b), \quad \text{i.e.,} \quad \{b_j \mid U \vdash_A U_j\} \vdash_B b.$$

We have to show that $\{(U_1, b_1), \dots, (U_n, b_n), (U, b)\} \in \text{Con}$. So let $I \subseteq \{1, \dots, n\}$ and suppose

$$U \cup \bigcup_{i \in I} U_i \in \text{Con}_A.$$

We must show that $\{b\} \cup \{b_i \mid i \in I\} \in \text{Con}_B$. Let $J \subseteq \{1, \dots, n\}$ consist of those j with $U \vdash_A U_j$. Then also

$$U \cup \bigcup_{i \in I} U_i \cup \bigcup_{j \in J} U_j \in \text{Con}_A.$$

Since

$$\bigcup_{i \in I} U_i \cup \bigcup_{j \in J} U_j \in \text{Con}_A,$$

from the consistency of $\{(U_1, b_1), \dots, (U_n, b_n)\}$ we can conclude that

$$\{b_i \mid i \in I\} \cup \{b_j \mid j \in J\} \in \text{Con}_B.$$

But $\{b_j \mid j \in J\} \vdash_B b$ by assumption. Hence

$$\{b_i \mid i \in I\} \cup \{b_j \mid j \in J\} \cup \{b\} \in \text{Con}_B.$$

For the final property, suppose

$$W \vdash W' \quad \text{and} \quad W' \vdash (U, b).$$

We have to show $W \vdash (U, b)$, i.e., $WU \vdash_B b$. We obtain $WU \vdash_B W'U$ by monotonicity in the first argument, and $W'U \vdash_B b$ by definition. $\square$

We shall now give an alternative characterization of the ideals in $\mathbf{A} \rightarrow \mathbf{B}$, as “approximable maps”. The basic idea for approximable maps is the desire to study “information respecting” maps from $\mathbf{A}$ into $\mathbf{B}$. Such a map is given by a relation r between Con_A and B , where $(U, b) \in r$ intuitively means that whenever we are given the information $U \in \text{Con}_A$, then we know that at least the token b appears in the value.

DEFINITION. Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ be information systems. A relation $r \subseteq \text{Con}_A \times B$ is an *approximable map* if it satisfies the following:

- (a) if $(U, b_1), \dots, (U, b_n) \in r$, then $\{b_1, \dots, b_n\} \in \text{Con}_B$;
- (b) if $(U, b_1), \dots, (U, b_n) \in r$ and $\{b_1, \dots, b_n\} \vdash_B b$, then $(U, b) \in r$;
- (c) if $(U', b) \in r$ and $U \vdash_A U'$, then $(U, b) \in r$.

THEOREM 2.1.2. *Let $\mathbf{A}$ and $\mathbf{B}$ be information systems. Then the ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are exactly the approximable maps from $\mathbf{A}$ to $\mathbf{B}$.*

PROOF. Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$. If $r \in |\mathbf{A} \rightarrow \mathbf{B}|$ then $r \subseteq \text{Con}_A \times B$ is consistent and deductively closed. We have to show that r satisfies the axioms for approximable maps.

(a) Let $(U, b_1), \dots, (U, b_n) \in r$. We must show that $\{b_1, \dots, b_n\} \in \text{Con}_B$. But this clearly follows from the consistency of r .

(b) Let $(U, b_1), \dots, (U, b_n) \in r$ and $\{b_1, \dots, b_n\} \vdash_B b$. We must show that $(U, b) \in r$. But

$$\{(U, b_1), \dots, (U, b_n)\} \vdash (U, b)$$

by the definition of the entailment relation $\vdash$ in $\mathbf{A} \rightarrow \mathbf{B}$, hence $(U, b) \in r$ since r is deductively closed.

(c) Let $U \vdash_A U'$ and $(U', b) \in r$. We must show that $(U, b) \in r$. But

$$\{(U', b)\} \vdash (U, b)$$

since $\{(U', b)\}U = \{b\}$ (which follows from $U \vdash_A U'$), hence $(U, b) \in r$, again since r is deductively closed.

For the other direction suppose that $r \subseteq \text{Con}_A \times B$ is an approximable map. We must show that $r \in |\mathbf{A} \rightarrow \mathbf{B}|$.

Consistency of r . Suppose $(U_1, b_1), \dots, (U_n, b_n) \in r$ and $U = \bigcup \{U_i \mid i \in I\} \in \text{Con}_A$ for some $I \subseteq \{1, \dots, n\}$. We must show that $\{b_i \mid i \in I\} \in \text{Con}_B$. Now from $(U_i, b_i) \in r$ and $U \vdash_A U_i$ we obtain $(U, b_i) \in r$ by axiom (c) for all $i \in I$, and hence $\{b_i \mid i \in I\} \in \text{Con}_B$ by axiom (a).

Deductive closure of r . Suppose $(U_1, b_1), \dots, (U_n, b_n) \in r$ and

$$W := \{(U_1, b_1), \dots, (U_n, b_n)\} \vdash (U, b).$$

We must show $(U, b) \in r$. By definition of $\vdash$ for $\mathbf{A} \rightarrow \mathbf{B}$ we have $WU \vdash_B b$, which is $\{b_i \mid U \vdash_A U_i\} \vdash_B b$. Further by our assumption $(U_i, b_i) \in r$ we know $(U, b_i) \in r$ by axiom (c) for all i with $U \vdash_A U_i$. Hence $(U, b) \in r$ by axiom (b). $\square$

2.1.3. Continuous functions. We can also characterize approximable maps in a different way, which is closer to usual characterizations of continuity¹:

LEMMA 2.1.3. *Let $\mathbf{A}$ and $\mathbf{B}$ be information systems and $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ monotone (i.e., $x \subseteq y$ implies $f(x) \subseteq f(y)$). Then the following are equivalent.*

- (a) *f satisfies the “principle of finite support” PFS: If $b \in f(x)$, then $b \in f(\bar{U})$ for some $U \subseteq x$.*
- (b) *f commutes with directed unions: for every directed $D \subseteq |\mathbf{A}|$ (i.e., for any $x, y \in D$ there is a $z \in D$ such that $x, y \subseteq z$)*

$$f\left(\bigcup_{x \in D} x\right) = \bigcup_{x \in D} f(x).$$

Note that in (b) the set $\{f(x) \mid x \in D\}$ is directed by monotonicity of f ; hence its union is indeed an ideal in $|\mathbf{B}|$. Note also that from PFS and monotonicity of f it follows immediately that if $V \subseteq f(x)$, then $V \subseteq f(\bar{U})$ for some $U \subseteq x$.

PROOF. Let f satisfy PFS, and $D \subseteq |\mathbf{A}|$ be directed. $f(\bigcup_{x \in D} x) \supseteq \bigcup_{x \in D} f(x)$ follows from monotonicity. For the reverse inclusion let $b \in f(\bigcup_{x \in D} x)$. Then by PFS $b \in f(\bar{U})$ for some $U \subseteq \bigcup_{x \in D} x$. From the

¹In fact, approximable maps are exactly the continuous functions w.r.t. the so-called Scott topology. However, we will not enter this subject here.

directedness and the fact that U is finite we obtain $U \subseteq z$ for some $z \in D$. From $b \in f(\overline{U})$ and monotonicity infer $b \in f(z)$. Conversely, let f commute with directed unions, and assume $b \in f(x)$. Then

$$b \in f(x) = f\left(\bigcup_{U \subseteq x} \overline{U}\right) = \bigcup_{U \subseteq x} f(\overline{U}),$$

hence $b \in f(\overline{U})$ for some $U \subseteq x$. $\square$

We call a function $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ continuous if it satisfies the conditions in Lemma 2.1.3. Hence continuous maps $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ are those that can be completely described from the point of view of finite approximations of the abstract objects $x \in |\mathbf{A}|$ and $f(x) \in |\mathbf{B}|$: whenever we are given a finite approximation V to the value $f(x)$, then there is a finite approximation U to the argument x such that already $f(\overline{U})$ contains the information in V ; note that by monotonicity $f(\overline{U}) \subseteq f(x)$.

Clearly the identity and constant functions are continuous, and also the composition $g \circ f$ of continuous functions $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ and $g: |\mathbf{B}| \rightarrow |\mathbf{C}|$.

THEOREM 2.1.4. *Let $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$, $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ be information systems. Then the ideals of $\mathbf{A} \rightarrow \mathbf{B}$ are in a natural bijective correspondence with the continuous functions from $|\mathbf{A}|$ to $|\mathbf{B}|$, as follows.*

- (a) *With any approximable map $r \subseteq \text{Con}_A \times B$ we can associate a continuous function $|r|: |\mathbf{A}| \rightarrow |\mathbf{B}|$ by*

$$|r|(z) := \{b \in B \mid (U, b) \in r \text{ for some } U \subseteq z\}.$$

We call $|r|(z)$ the application of r to z .

- (b) *Conversely, with any continuous function $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ we can associate an approximable map $\hat{f} \subseteq \text{Con}_A \times B$ by*

$$\hat{f} := \{(U, b) \mid b \in f(\overline{U})\}.$$

These assignments are inverse to each other, i.e., $f = |\hat{f}|$ and $r = \widehat{|r|}$.

PROOF. Let r be an ideal of $\mathbf{A} \rightarrow \mathbf{B}$; then by Theorem 2.1.2 we know that r is an approximable map. We first show that $|r|$ is well-defined. So let $z \in |\mathbf{A}|$.

$|r|(z)$ is consistent: let $b_1, \dots, b_n \in |r|(z)$. Then there are $U_1, \dots, U_n \subseteq z$ such that $(U_i, b_i) \in r$. Hence $U := U_1 \cup \dots \cup U_n \subseteq z$ and $(U, b_i) \in r$ by axiom (c) of approximable maps. Now from axiom (a) we can conclude that $\{b_1, \dots, b_n\} \in \text{Con}_B$.

$|r|(z)$ is deductively closed: let $b_1, \dots, b_n \in |r|(z)$ and $\{b_1, \dots, b_n\} \vdash_B b$. We must show $b \in |r|(z)$. As before we find $U \subseteq z$ such that $(U, b_i) \in r$. Now from axiom (b) we can conclude $(U, b) \in r$ and hence $b \in |r|(z)$.

Continuity of $|r|$ follows immediately from part (a) of Lemma 2.1.3 above, since by definition $|r|$ is monotone and satisfies PFS.

Now let $f: |\mathbf{A}| \rightarrow |\mathbf{B}|$ be continuous. It is easy to verify that $\hat{f}$ is indeed an approximable map. Furthermore

$$\begin{aligned} b \in |\hat{f}|(z) &\leftrightarrow (U, b) \in \hat{f} \quad \text{for some } U \subseteq z \\ &\leftrightarrow b \in f(\overline{U}) \quad \text{for some } U \subseteq z \\ &\leftrightarrow b \in f(z) \quad \text{by monotonicity and PFS.} \end{aligned}$$

Finally, for any approximable map $r \subseteq \text{Con}_A \times B$ we have

$$\begin{aligned} (U, b) \in r &\leftrightarrow \exists_{V \subseteq \overline{U}} (V, b) \in r \quad \text{by axiom (c) for approximable maps} \\ &\leftrightarrow b \in |r|(\overline{U}) \\ &\leftrightarrow (U, b) \in \widehat{|r|}, \end{aligned}$$

hence $r = \widehat{|r|}$. □

Consequently we can (and will) view approximable maps $r \subseteq \text{Con}_A \times B$ as continuous functions from $|\mathbf{A}|$ to $|\mathbf{B}|$.

Equality of two subsets $r, s \subseteq \text{Con}_A \times B$ means that they consist of the same tokens (U, b) . We can characterize equality $r = s$ by extensional equality of the associated functions $|r|, |s|$. It even suffices that $|r|$ and $|s|$ coincide on all compact elements $\overline{U}$ for $U \in \text{Con}_A$.

LEMMA 2.1.5 (Extensionality). *Assume that $\mathbf{A} = (A, \text{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \text{Con}_B, \vdash_B)$ are information systems and $r, s \subseteq \text{Con}_A \times B$ approximable maps. Then the following are equivalent.*

- (a) $r = s$,
- (b) $|r|(z) = |s|(z)$ for all $z \in |\mathbf{A}|$,
- (c) $|r|(\overline{U}) = |s|(\overline{U})$ for all $U \in \text{Con}_A$.

PROOF. It suffices to prove (c) $\rightarrow$ (a). As above this follows from

$$\begin{aligned} (U, b) \in r &\leftrightarrow \exists_{V \subseteq \overline{U}} (V, b) \in r \quad \text{by axiom (c) for approximable maps} \\ &\leftrightarrow b \in |r|(\overline{U}). \end{aligned} \quad \square$$

Moreover, one can easily check that

$$s \circ r := \{ (U, c) \mid \exists_V ((V, c) \in s \wedge (U, V) \subseteq r) \}$$

is an approximable map (where $(U, V) := \{ (U, b) \mid b \in V \}$), and

$$|s \circ r| = |s| \circ |r|, \quad \widehat{g \circ f} = \hat{g} \circ \hat{f}.$$

We usually write $r(z)$ for $|r|(z)$, and similarly $(U, b) \in f$ for $(U, b) \in \hat{f}$. It should always be clear from the context where the mods and hats should be inserted.

2.2. Partial continuous functionals

We define types, built from base types by the formation of function types, $\tau \rightarrow \sigma$; types may involve type variables. For every closed type τ we define an information system $\mathbf{C}_\tau$ whose ideals will be the partial continuous functionals of type τ .

2.2.1. Types. Types will be built from “algebras” (given by a list of “constructor types”) by the formation of “function types” $\tau \rightarrow \sigma$. Types may involve type variables $\alpha, \beta, \gamma, \xi, \zeta$. Iterated arrows are understood as associated to the right. For example, $\alpha \rightarrow \beta \rightarrow \gamma$ means $\alpha \rightarrow (\beta \rightarrow \gamma)$, not $(\alpha \rightarrow \beta) \rightarrow \gamma$.

DEFINITION. *Constructor types* κ have the form

$$\vec{\alpha} \rightarrow (\xi)_{i < n} \rightarrow \xi$$

with all type variables α_i distinct from each other and from ξ . An argument type of a constructor type is called a *parameter* argument type if it is different from ξ , and a *recursive* argument type otherwise. A constructor type is *recursive* if it has a recursive argument type. We call

$$\iota := \mu_\xi \vec{\kappa}$$

with $\vec{\kappa}$ not empty an *algebra form*. An algebra form is *non-recursive* (or *explicit*) if it does not have recursive argument types.

EXAMPLES. We list some algebra forms without parameters, with standard names for the constructors added to each constructor type.

$\mathbb{U} := \mu_\xi(\text{Dummy}: \xi)$	(unit),
$\mathbb{B} := \mu_\xi(\text{tt}: \xi, \text{ff}: \xi)$	(booleans),
$\mathbb{N} := \mu_\xi(0: \xi, S: \xi \rightarrow \xi)$	(natural numbers, unary),
$\mathbb{P} := \mu_\xi(1: \xi, S_0: \xi \rightarrow \xi, S_1: \xi \rightarrow \xi)$	(positive numbers, binary),
$\mathbb{Y} := \mu_\xi(-: \xi, \text{Branch}: \xi \rightarrow \xi \rightarrow \xi)$	(binary trees, or derivations).

Algebra forms with parameters are

$\mathbb{I}(\alpha)$	$:= \mu_\xi(\text{Id}: \alpha \rightarrow \xi)$	(identity),
$\mathbb{L}(\alpha)$	$:= \mu_\xi(\text{Nil}: \xi, \text{Cons}: \alpha \rightarrow \xi \rightarrow \xi)$	(lists),
$\mathbb{S}(\alpha)$	$:= \mu_\xi(\text{SCons}: \alpha \rightarrow \xi \rightarrow \xi)$	(streams),
$\alpha \times \beta$	$:= \mu_\xi(\text{Pair}: \alpha \rightarrow \beta \rightarrow \xi)$	(product),
$\alpha + \beta$	$:= \mu_\xi(\text{InL}: \alpha \rightarrow \xi, \text{InR}: \beta \rightarrow \xi)$	(sum),
$\text{uysum}(\alpha)$	$:= \mu_\xi(\text{DummyL}: \xi, \text{Inr}: \alpha \rightarrow \xi)$	(for $\mathbb{U} + \alpha$),
$\text{ysumu}(\alpha)$	$:= \mu_\xi(\text{Inl}: \alpha \rightarrow \xi, \text{DummyR}: \xi)$	(for $\alpha + \mathbb{U}$).

The default name for the i -th constructor of an algebra form is C_i .

DEFINITION (Type).

$$\tau, \sigma, \rho ::= \alpha \mid \iota(\vec{\tau}) \mid \tau \rightarrow \sigma,$$

where ι is an algebra form with $\vec{\alpha}$ its parameters, and $\iota(\vec{\tau})$ the result of substituting the (already generated) types $\vec{\tau}$ for $\vec{\alpha}$. A type is *closed* if it has no free type variables. Types of the form $\iota(\vec{\tau})$ are called *algebras*. The *level* of a type is defined by

$$\begin{aligned} \text{lev}(\alpha) &:= 0, \\ \text{lev}(\iota(\vec{\tau})) &:= \max(\text{lev}(\vec{\tau})), \\ \text{lev}(\tau \rightarrow \sigma) &:= \max(\text{lev}(\sigma), 1 + \text{lev}(\tau)). \end{aligned}$$

Base types are types of level 0, and a *higher* type has level at least 1.

- EXAMPLES. 1. $\mathbb{L}(\alpha)$, $\mathbb{L}(\mathbb{L}(\alpha))$, $\alpha \times \beta$ are algebras.
 2. $\mathbb{L}(\mathbb{L}(\mathbb{N}))$, $\mathbb{N} + \mathbb{B}$, $\mathbb{Z} := \mathbb{P} + \mathbb{U} + \mathbb{P}$, $\mathbb{Q} := \mathbb{Z} \times \mathbb{P}$ are closed base types.
 3. $\mathbb{R} := (\mathbb{N} \rightarrow \mathbb{Q}) \times (\mathbb{P} \rightarrow \mathbb{N})$ is a closed algebra of level 1.

There can be many equivalent ways to define a particular type. For instance, we could take $\mathbb{U} + \mathbb{U}$ to be the type of booleans, $\mathbb{L}(\mathbb{U})$ to be the type of natural numbers, and $\mathbb{L}(\mathbb{B})$ to be the type of positive binary numbers.

2.2.2. The information system $\mathbf{C}_\tau$. For every closed type τ we define the information system $\mathbf{C}_\tau = (C_\tau, \text{Con}_\tau, \vdash_\tau)$. The ideals $x \in |\mathbf{C}_\tau|$ are the *partial continuous functionals* of type τ . Since we will have $\mathbf{C}_{\tau \rightarrow \sigma} = \mathbf{C}_\tau \rightarrow \mathbf{C}_\sigma$, the partial continuous functionals of type $\tau \rightarrow \sigma$ will correspond to the continuous functions from $|\mathbf{C}_\tau|$ to $|\mathbf{C}_\sigma|$.

DEFINITION (Information system of a closed type τ). We simultaneously define C_ι , $C_{\tau \rightarrow \sigma}$, Con_ι and $\text{Con}_{\tau \rightarrow \sigma}$.

- (a) The *tokens* $a \in C_\iota$ are the type correct constructor expressions $Ca_1^* \dots a_n^*$ where a_i^* is an *extended token*, i.e., a token or the special symbol $*$ which carries no information.
- (b) The tokens in $C_{\tau \rightarrow \sigma}$ are the pairs (U, b) with $U \in \text{Con}_\tau$ and $b \in C_\sigma$.
- (c) A finite set U of tokens in C_ι is *consistent* (i.e., $U \in \text{Con}_\iota$) if all its elements start with the same constructor C , say of arity $\tau_1 \rightarrow \dots \rightarrow \tau_n \rightarrow \iota$, and all $U_i \in \text{Con}_{\tau_i}$ for $i = 1, \dots, n$, where U_i consists of all (proper) tokens at the i -th argument position of some token in $U = \{Ca_1^*, \dots, Ca_m^*\}$.
- (d) $\{(U_i, b_i) \mid i \in I\} \in \text{Con}_{\tau \rightarrow \sigma}$ is defined to mean

$$\forall J \subseteq I \left(\bigcup_{j \in J} U_j \in \text{Con}_\tau \rightarrow \{b_j \mid j \in J\} \in \text{Con}_\sigma \right).$$

Building on this definition, we define $U \vdash_\tau a$ for $U \in \text{Con}_\tau$ and $a \in C_\tau$.

- (e) $\{Ca_1^*, \dots, Ca_m^*\} \vdash_\iota C'a^*$ is defined to mean $C = C'$, $m \geq 1$ and $U_i \vdash a_i^*$, with U_i as in (c) above (and $U \vdash *$ taken to be true).
- (f) $W \vdash_{\tau \rightarrow \sigma} (U, b)$ is defined to mean $WU \vdash_\sigma b$, where application WU of $W = \{(U_i, b_i) \mid i \in I\} \in \text{Con}_{\tau \rightarrow \sigma}$ to $U \in \text{Con}_\tau$ is defined to be $\{b_i \mid U \vdash_\tau U_i\}$; recall that $U \vdash V$ abbreviates $\forall_{a \in V} (U \vdash a)$.

Note that the present definition is by recursion on the *height* of the syntactic expressions involved:

$$\begin{aligned}
|\alpha| &:= 0, \\
|\iota(\tau_1 \dots \tau_n)| &:= 1 + \max\{|\tau_i| \mid i = 1, \dots, n\}, \\
|\tau \rightarrow \sigma| &:= \max\{1 + |\tau|, 1 + |\sigma|\}, \\
|Ca_1^* \dots a_n^*| &:= 1 + \max\{|a_i^*| \mid i = 1, \dots, n\}, \\
|*| &:= 0, \\
|(U, b)| &:= \max\{1 + |U|, 1 + |b|\}, \\
|\{a_i \mid i \in I\}| &:= \max\{1 + |a_i| \mid i \in I\}, \\
|U \vdash a| &:= \max\{1 + |U|, 1 + |a|\}.
\end{aligned}$$

It is easy to see that $(C_\tau, \text{Con}_\tau, \vdash_\tau)$ is an information system. Observe that all the notions involved are computable: $a \in C_\tau$, $U \in \text{Con}_\tau$ and $U \vdash_\tau a$.

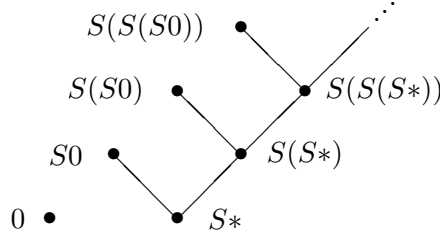
DEFINITION (Partial continuous functionals). For every closed type τ let $\mathbf{C}_\tau$ be the information system $(C_\tau, \text{Con}_\tau, \vdash_\tau)$. The set $|\mathbf{C}_\tau|$ of ideals in $\mathbf{C}_\tau$ is the set of *partial continuous functionals* of type τ . A partial continuous functional $x \in |\mathbf{C}_\tau|$ is *computable* if it is recursively enumerable when viewed as a set of tokens.

Notice that $\mathbf{C}_{\tau \rightarrow \sigma} = \mathbf{C}_\tau \rightarrow \mathbf{C}_\sigma$ as defined generally for information systems.

For example, the tokens for the base type $\mathbb{N}$ are shown in Figure 1. For tokens a, b we have $\{a\} \vdash b$ if and only if there is a path from a (up) to b (down). As another (more typical) example, consider the base type $\mathbb{Y}$ of binary trees with a nullary constructor $-$ and a binary Branch. Then $\{\text{Branch}(-, *), \text{Branch}(*, -)\}$ is consistent, and it entails $\text{Branch}(-, -)$.

Recall that a partial continuous functional is an ideal in $\mathbf{C}_\tau$. We will be interested in “cototal” and “total” ideals. To grasp what this means it helps to look at examples of tokens for some closed base types.

$$\begin{array}{ll}
\mathbb{B} & \mathbf{tt}, \quad \mathbf{ff} \\
\mathbb{N} & S^n 0, \quad S^{n+1} * \\
\mathbb{L}(\mathbb{B}) & \text{Cons}(\mathbf{tt}, \text{Cons}(*, \text{Cons}(\mathbf{ff}, [])))
\end{array}$$

FIGURE 1. Tokens and entailment for $\mathbb{N}$

$$\begin{aligned} \mathbb{S}(\mathbb{B}) & \quad \text{SCons}(\mathbf{tt}, \text{SCons}(*, \text{SCons}(\mathbf{ff}, *))) =: a \\ \mathbb{L}(\mathbb{S}(\mathbb{B})) & \quad \text{Cons}(a, \text{Cons}(a, [])). \end{aligned}$$

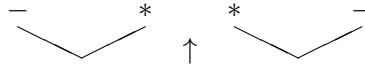
Generally these tokens can be described as terms built from the constructors involved in the used base types and possibly the symbol $*$.

DEFINITION (Cototal and total ideals of a closed base type). Let ι be a closed base type. Its tokens can be seen as constructor terms with some argument positions occupied by $*$. An ideal x in $\mathbf{C}_\iota$ is *cototal* if for each of its tokens $t(*)$ with a distinguished occurrence of $*$ there is another token of the form $t(C\vec{*})$ in x . Here C is a constructor for the base type of $*$'s position in $t(*)$. We call x *total* if it is cototal and finite.

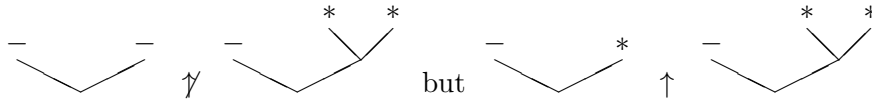
We give some examples, with a more pictorial representation. Tokens in $\mathbf{C}_\mathbb{Y}$:



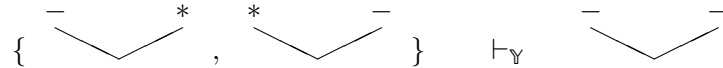
Consistency in $\mathbf{C}_\mathbb{Y}$:



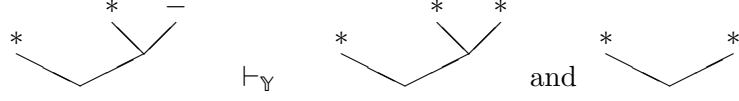
Moreover



Entailment in $\mathbf{C}_\mathbb{Y}$:

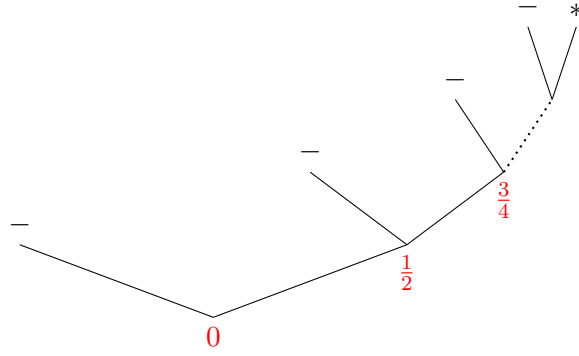


and also



Ideals in C_Y :

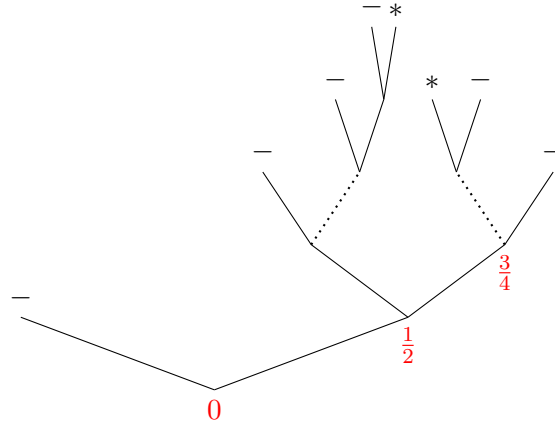
- (1) $1 :=$ closure of all



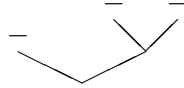
- (2) -1 is defined similarly

- (3) $-1 \cup 1$

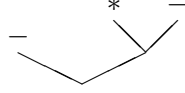
- (4) $\frac{1}{2} :=$ closure of all



- (5) Closure of



(6) Closure of



Among these are

- (1) – (4) cototal ideals,
- (5) a total ideal,
- (6) a neither total nor cototal ideal.

2.2.3. Constructors as continuous functions. Let ι be a closed base type. Every constructor C generates the following ideal in the function space determined by the type of the constructor:

$$r_C := \{ (\vec{U}, C\vec{a}^*) \mid \vec{U} \vdash \vec{a}^* \},$$

where $(\vec{U}, a)$ abbreviates $(U_1, (U_2, \dots (U_n, a) \dots))$.

According to the general definition of a continuous function associated to an ideal in a function space the continuous map $|r_C|$ satisfies

$$|r_C|(\vec{x}) = \{ C\vec{a}^* \mid \exists \vec{U} \subseteq \vec{x} (\vec{U} \vdash \vec{a}^*) \}.$$

(For $\mathbb{N}$ we have $|r_S|(\{0\}) = \{S0, S*\}$ and $|r_S|(\{S0, S*\}) = \{SS0, SS*, S*\}$.) An immediate consequence is that the (continuous maps corresponding to) constructors are injective and their ranges are disjoint.

LEMMA 2.2.1 (Constructors are injective and have disjoint ranges). *Let ι be a closed base type and C be a constructor of ι . Then*

$$|r_C|(\vec{x}) \subseteq |r_C|(\vec{y}) \leftrightarrow \vec{x} \subseteq \vec{y}.$$

If C_1, C_2 are distinct constructors of ι , then $|r_{C_1}|(\vec{x}) \neq |r_{C_2}|(\vec{y})$, since the two ideals are non-empty and disjoint.

PROOF. Immediate from the definitions. □

REMARK. Notice that neither property holds for flat information systems, since for them, by monotonicity, constructors need to be *strict* (i.e., if one argument is the empty ideal, then the value is as well). But then we have

$$\begin{aligned} |r_C|(\bar{\emptyset}, y) &= \bar{\emptyset} = |r_C|(x, \bar{\emptyset}), \\ |r_{C_1}|(\bar{\emptyset}) &= \bar{\emptyset} = |r_{C_2}|(\bar{\emptyset}) \end{aligned}$$

where in the first case we have one binary and, in the second, two unary constructors.

LEMMA (Ideals of a closed base type). *Every non-empty ideal in the information system associated to a closed base type has the form $|r_C|(\vec{x})$ with a constructor C and ideals $\vec{x}$.*

PROOF. Let z be a non-empty ideal and $Ca_0^*b_0^* \in z$, where for simplicity we assume that C is a binary constructor. Let $x := \{a \mid Ca^* \in z\}$ and $y := \{b \mid C^*b \in z\}$; clearly x, y are ideals. We claim that $z = |r_C|(x, y)$. For $\supseteq$ consider Ca^*b^* with $a^* \in x \cup \{*\}$ and $b^* \in y \cup \{*\}$. Then by definition $\{Ca^*, C^*b^*\} \subseteq z$, hence $Ca^*b^* \in z$ by deductive closure. Conversely, notice that an arbitrary element of z must have the form Ca^*b^* , because of consistency. Then $\{Ca^*, C^*b^*\} \subseteq z$ again by deductive closure. Hence $a^* \in x \cup \{*\}$ and $b^* \in y \cup \{*\}$, and therefore $Ca^*b^* \in |r_C|(x, y)$. $\square$

It is in this proof that we need entailment to be a relation between finite sets of tokens and single tokens, not just a binary relation between tokens. Information systems with the latter property are called *atomic*.

The information systems $\mathbf{C}_\tau$ enjoy the pleasant property of *coherence*, which amounts to the possibility to locate inconsistencies in two-element sets of data objects. Generally, an information system $\mathbf{A} = (A, \text{Con}, \vdash)$ is *coherent* if it satisfies: $U \subseteq A$ is consistent if and only if all of its two-element subsets are.

LEMMA 2.2.2. *Let $\mathbf{A}$ and $\mathbf{B}$ be information systems. If $\mathbf{B}$ is coherent, then so is $\mathbf{A} \rightarrow \mathbf{B}$.*

PROOF. Exercise. $\square$

2.3. Terms

To work with functionals in a formal theory we need to have a language to denote them. Again lambda calculus is the appropriate tool, this time extended by constants for functionals given by certain defining equations, or more precisely, “computation rules”. For their formulation we need some knowledge on most general unifiers and unification generally. The resulting term system can be seen as a common extension of Gödel’s T (1958) and Plotkin’s PCF (1977); we call it T^+ .

2.3.1. Unification. We show that for any two constructor terms one can decide whether there exists a “unifier”, and if so, compute a “most general” one. A solution of this problem has been given by Robinson (1965). In the formulation of the algorithm below we follow Martelli and Montanari (1982).

By a constructor term t, s (term for short) we mean a term built from variables x, y, z and constructors C by application. A *substitution* is a finite set $\vartheta = \{t_1/x_1, \dots, t_n/x_n\}$ of pairs of variables and terms, such that $x_i \neq x_j$

for $i \neq j$, and $t_i \neq x_i$ for all i . An element t_i/x_i of ϑ is called a *binding* (of x_i to t_i). By $t\vartheta$ we denote the result of simultaneously replacing each variable x_i in t by t_i , and call $t\vartheta$ the *instance* of t induced by ϑ . We shall use ϑ, η, ζ for substitutions. Let ε be the empty substitution. For given substitutions

$$\begin{aligned}\vartheta &= \{t_1/x_1, \dots, t_n/x_n\} \\ \eta &= \{s_1/y_1, \dots, s_m/y_m\},\end{aligned}$$

the *composition* $\vartheta\eta$ of ϑ and η is the substitution obtained by deleting in the set

$$\{t_1\eta/x_1, \dots, t_n\eta/x_n, s_1/y_1, \dots, s_m/y_m\}$$

all bindings $t_i\eta/x_i$ such that $t_i\eta = x_i$, and also all bindings s_j/y_j such that $y_j \in \{x_1, \dots, x_n\}$. A substitution ϑ is *idempotent* if $\vartheta\vartheta = \vartheta$. A substitution ϑ is called *more general* than η (written $\eta \leq \vartheta$), if there is a substitution ζ such that $\eta = \vartheta\zeta$. ϑ and η are *equivalent*, if $\vartheta \leq \eta \leq \vartheta$.

It is easy to see that $(t\vartheta)\eta = t(\vartheta\eta)$, and that composition is associative.

We now come to the unification problem. By this we mean the question whether for two given terms t, s there is a substitution ϑ “unifying” the two terms, i.e., with the property $t\vartheta = s\vartheta$.

Let E denote finite equation systems, i.e., multisets

$$\{t_1 = s_1, \dots, t_n = s_n\}$$

of equations between terms (more precisely pairs of terms). Consider $\{\perp\}$ as a (contradictory) equation system. A substitution ϑ *unifies* E , if for every equation $t = s$ in E we have $t\vartheta = s\vartheta$; no ϑ unifies $\{\perp\}$. ϑ is a *most general unifier* (mgu) of E , if ϑ is a unifier of E and $\eta \leq \vartheta$ for every unifier η of E .

The following characterization of idempotent mgus will be useful in the proof of the Unification Theorem below.

LEMMA 2.3.1 (Characterization of idempotent mgu’s). *Let ϑ be a unifier of E . Then ϑ is an idempotent mgu of E iff $\eta = \vartheta\eta$ for all unifiers η of E .*

PROOF. Exercise. □

DEFINITION (Unification algorithm). $E \mapsto_{\vartheta} E'$ is defined by

- (a) $\{t = x\} \cup E \mapsto_{\varepsilon} \{x = t\} \cup E$, if t is not a variable.
- (b) $\{x = x\} \cup E \mapsto_{\varepsilon} E$.
- (c) $\{Ct_1 \dots t_n = Cs_1 \dots s_n\} \cup E \mapsto_{\varepsilon} \{t_1 = s_1, \dots, t_n = s_n\} \cup E$.
- (d) $\{Ct_1 \dots t_n = C's_1 \dots s_n\} \cup E \mapsto_{\varepsilon} \{\perp\}$ if $C \neq C'$.
- (e) $\{x = t, t_1(x) = s_1(x), \dots, t_n(x) = s_n(x)\} \mapsto_{\{t/x\}} \{t_1(t) = s_1(t), \dots, t_n(t) = s_n(t)\}$ if $x \notin \text{FV}(t)$.
- (f) $\{x = t\} \cup E \mapsto_{\varepsilon} \{\perp\}$, if $x \in \text{FV}(t)$ and $t \neq x$.

PROPOSITION. Assume $E \mapsto_{\vartheta} E'$.

- (a) If η' is a unifier of E' , then $\vartheta\eta'$ is a unifier of E .
 (b) If η is a unifier of E , then $\eta = \vartheta\eta$ and η is a unifier of E' .

PROOF. By cases according to the definition of $E \mapsto_{\vartheta} E'$. Clearly it suffices to treat case (e).

Let η' be a unifier of E' . Then $\{t/x\}\eta'$ is a unifier of E .

Let η be a unifier of E . Then $x\eta = t\eta$, hence $\eta = \{t/x\}\eta$ (since both substitutions coincide on all variables), and moreover

$$t_i\{t/x\}\eta = t_i\eta = s_i\eta = s_i\{t/x\}\eta.$$

Hence η is a unifier of E' . □

COROLLARY. Assume

$$E_1 \mapsto_{\vartheta_1} E_2 \mapsto_{\vartheta_2} \dots E_n \mapsto_{\vartheta_n} E_{n+1}.$$

- (a) If ϑ is a unifier of E_{n+1} , then $\vartheta_1 \dots \vartheta_n \vartheta$ is a unifier of E_1 .
 (b) If η is a unifier of E_1 , then $\eta = \vartheta_1 \dots \vartheta_n \eta$ and η is a unifier of E_{n+1} .

PROOF. The first part clearly follows from the first part of the Proposition. The second part is proved by induction on n . For $n = 0$ there is nothing to show. In the step we split the assumption into

$$E_1 \mapsto_{\vartheta_1} E_2 \quad \text{and} \quad E_2 \mapsto_{\vartheta_2} \dots E_n \mapsto_{\vartheta_n} E_{n+1}.$$

By the second part of the Proposition we have that $\eta = \vartheta_1\eta$ is a unifier of E_2 . Hence by IH $\eta = \vartheta_2 \dots \vartheta_n \eta$ is a unifier of E_{n+1} . Moreover we have $\eta = \vartheta_1\eta = \vartheta_1\vartheta_2 \dots \vartheta_n \eta$. □

UNIFICATION THEOREM. Let E be a finite equation system. Then every sequence

$$E = E_1 \mapsto_{\vartheta_1} E_2 \mapsto_{\vartheta_2} \dots$$

terminates with $E_{n+1} = \emptyset$ or $E_{n+1} = \{\perp\}$. In the first case E is unifiable, and $\vartheta_1 \dots \vartheta_n$ is an idempotent mgu of E . In the second case E is not unifiable.

PROOF. Exercise. Termination is proved using the lexicographic ordering of $\mathbb{N}^3$. To every $E = \{t_1 = s_1, \dots, t_n = s_n\}$ assign a triple $(n_1, n_2, n_3) \in \mathbb{N}^3$ by

$n_1 :=$ number of variables in E ,

$n_2 :=$ number of occurrences of variables and constructors in E ,

$n_3 :=$ number of equations $t = x$ in E such that t is not a variable. □

2.3.2. A common extension T^+ of Gödel's T and Plotkin's PCF.

DEFINITION (Terms). *Terms* of T^+ are built from (typed) variables and (typed) constants (constructors C or defined constants D ; see the definition below) by application and abstraction:

$$M, N ::= x^\tau \mid C^\tau \mid D^\tau \mid (\lambda_{x^\tau} M^\sigma)^{\tau \rightarrow \sigma} \mid (M^{\tau \rightarrow \sigma} N^\tau)^\sigma.$$

The set $FV(M)$ of free variables of a term M is defined by

$$\begin{aligned} FV(x) &:= \{x\}, & FV(C) &:= FV(D) := \emptyset, \\ FV(\lambda_x M) &:= FV(M) \setminus \{x\}, & FV(MN) &:= FV(M) \cup FV(N). \end{aligned}$$

DEFINITION (Conversion). We define a conversion relation $\mapsto_\beta$ for terms similarly to what we did for derivation terms:

$$(\lambda_x M(x))^{\tau \rightarrow \sigma} N^\tau \mapsto_\beta M(N)^\sigma.$$

In addition we will employ another conversion relation $\mapsto_\eta$ defined by

$$\lambda_x(Mx) \mapsto_\eta M \quad \text{if } x \notin FV(M) \text{ (} M \text{ not an abstraction).}$$

DEFINITION (Computation rule). Every defined constant D comes with a system of *computation rules*, consisting of finitely many equations

$$(13) \quad D\vec{P}_i(\vec{y}_i) = M_i \quad (i = 1, \dots, n \text{ where } n \geq 0)$$

with free variables of $\vec{P}_i(\vec{y}_i)$ and M_i among $\vec{y}_i$, where the arguments on the left hand side must be “constructor patterns”, i.e., lists of applicative terms built from constructors and distinct variables. To ensure consistency of the defining equations, we require that for $i \neq j$ $\vec{P}_i$ and $\vec{P}_j$ have disjoint free variables, and either $\vec{P}_i$ and $\vec{P}_j$ are non-unifiable (i.e., there is no substitution which identifies them), or else for the most general unifier ϑ of $\vec{P}_i$ and $\vec{P}_j$ we have $M_i\vartheta = M_j\vartheta$. Notice that the substitution ϑ assigns to the variables $\vec{y}_i$ in M_i constructor patterns $\vec{R}_k(\vec{z})$ ($k = i, j$). A further requirement on a system of computation rules $D\vec{P}_i(\vec{y}_i) = M_i$ is that the lengths of all $\vec{P}_i(\vec{y}_i)$ are the same; this number is called the *arity* of D , denoted by $\text{ar}(D)$. A substitution instance of a left hand side of (13) is called a *D-redex*.

More formally, constructor patterns are defined inductively by (we write $\vec{P}(\vec{x})$ to indicate all variables in $\vec{P}$):

- (a) x is a constructor pattern.
- (b) The empty list is a constructor pattern.
- (c) If $\vec{P}(\vec{x})$ and $Q(\vec{y})$ are constructor patterns whose variables $\vec{x}$ and $\vec{y}$ are disjoint, then $(\vec{P}, Q)(\vec{x}, \vec{y})$ is a constructor pattern.
- (d) If C is a constructor and $\vec{P}$ a constructor pattern, then so is $C\vec{P}$.

REMARK. The requirement of disjoint variables in constructor patterns $\vec{P}_i$ and $\vec{P}_j$ used in computation rules of a defined constant D is needed to ensure that applying the most general unifier produces constructor patterns again. However, for readability we take this as an implicit convention, and write computation rules with possibly non-disjoint variables.

Examples of constants D defined by computation rules are abundant. In particular, the (structural) recursion and corecursion operators will be defined by computation rules.

The simplest example is the constant $\perp_\iota$ of type ι with no computation rules. The boolean connectives `andb`, `impb` and `orb` are defined by

$$\begin{array}{lll} \mathbf{tt} \text{ andb } y = y, & \mathbf{ff} \text{ impb } y = \mathbf{tt}, & \mathbf{tt} \text{ orb } y = \mathbf{tt}, \\ x \text{ andb } \mathbf{tt} = x, & \mathbf{tt} \text{ impb } y = y, & x \text{ orb } \mathbf{tt} = \mathbf{tt}, \\ \mathbf{ff} \text{ andb } y = \mathbf{ff}, & x \text{ impb } \mathbf{tt} = \mathbf{tt}, & \mathbf{ff} \text{ orb } y = y, \\ x \text{ andb } \mathbf{ff} = \mathbf{ff}, & & x \text{ orb } \mathbf{ff} = x. \end{array}$$

Notice that when two such rules overlap, their right hand sides are equal under any unifier of the left hand sides.

Decidable *equality* $=_\iota: \iota \rightarrow \iota \rightarrow \mathbb{B}$ for a closed base type ι can be defined easily by computation rules. For example,

$$\begin{array}{ll} (0 =_{\mathbb{N}} 0) = \mathbf{tt}, & (Sn =_{\mathbb{N}} 0) = \mathbf{ff}, \\ (0 =_{\mathbb{N}} Sm) = \mathbf{ff}, & (Sn =_{\mathbb{N}} Sm) = (n =_{\mathbb{N}} m) \end{array}$$

and similarly for $\mathbb{Y}$ with constructors `0` (leaf) and `C` (branch)

$$\begin{array}{ll} (0 =_{\mathbb{Y}} 0) = \mathbf{tt}, & (Cts =_{\mathbb{Y}} 0) = \mathbf{ff}, \\ (0 =_{\mathbb{Y}} Cts) = \mathbf{ff}, & (Cts =_{\mathbb{Y}} Ct's') = (t =_{\mathbb{Y}} t' \text{ andb } s =_{\mathbb{Y}} s'). \end{array}$$

For $\mathbb{L}(\iota)$ with ι a closed base type we define $=_{\mathbb{L}(\iota)}: \mathbb{L}(\iota) \rightarrow \mathbb{L}(\iota) \rightarrow \mathbb{B}$ by

$$\begin{array}{ll} ([] =_{\mathbb{L}(\iota)} []) = \mathbf{tt}, & (a::\ell = []) =_{\mathbb{L}(\iota)} \mathbf{ff}, \\ ([] =_{\mathbb{L}(\iota)} a::\ell) = \mathbf{ff}, & (a::\ell =_{\mathbb{L}(\iota)} a'::\ell') = (a =_\iota a' \text{ andb } \ell =_{\mathbb{L}(\iota)} \ell'). \end{array}$$

For the base type $\mathbb{N}$ of natural numbers we have the doubling function

$$\begin{array}{ll} \text{Double}(0) & := 0, \\ \text{Double}(S(n)) & := S(S(\text{Double}(n))). \end{array}$$

Addition (written infix) is defined similarly, this time with a parameter m :

$$\begin{array}{ll} m + 0 & := m, \\ m + S(n) & := S(m + n). \end{array}$$

Multiplication (again written infix) is defined using addition by

$$\begin{aligned} m \cdot 0 &:= 0, \\ m \cdot S(n) &:= (m \cdot n) + m. \end{aligned}$$

Similarly we can define all primitive recursive functions.

Up to now we have only considered examples of total functions, in the sense that total arguments are mapped to total values. But recall that in our setting functions need not be total. To give an example consider the base type of streams defined by

$$\mathbb{S}(\alpha) := \mu_{\xi}(\alpha \rightarrow \xi \rightarrow \xi)$$

with a single constructor $C: \alpha \rightarrow \mathbb{S}(\alpha) \rightarrow \mathbb{S}(\alpha)$. We write $C_a u$ or $a :: u$ for Cau . This base type differs from the ones previously considered by not having a nullary constructor. As a consequence it does not have non-empty total ideals, but clearly cototal ones. An example is $\{C_a^n(*) \mid n \geq 1\}$. We define the function Map of type $(\rho \rightarrow \sigma) \rightarrow \mathbb{S}(\rho) \rightarrow \mathbb{S}(\sigma)$ mapping its function argument $h: \rho \rightarrow \sigma$ over a stream u of type $\mathbb{S}(\rho)$ by the computation rule

$$\text{Map}_h(a :: u) := (ha) :: \text{Map}_h(u).$$

2.3.3. Recursion operators. Important examples of such constants D are the (structural) higher type *recursion operators* $\mathcal{R}_l^{\tau}$ introduced by Hilbert (1925) and Gödel (1958). They are used to construct maps from the algebra ι to the value type τ , by recursion on the structure of ι .

For instance, $\mathcal{R}_{\mathbb{N}}^{\tau}$ has type $\mathbb{N} \rightarrow \tau \rightarrow (\mathbb{N} \rightarrow \tau \rightarrow \tau) \rightarrow \tau$. It is defined by the computation rules

$$\begin{aligned} \mathcal{R}_{\mathbb{N}}^{\tau} 0af &:= a, \\ \mathcal{R}_{\mathbb{N}}^{\tau} (Sn)af &:= fn(\mathcal{R}_{\mathbb{N}}^{\tau} naf). \end{aligned}$$

The first argument is the recursion argument, the second one gives the base value, and the third one gives the step function, mapping the recursion argument and the previous value to the next value. For example, $\mathcal{R}_{\mathbb{N}}^{\mathbb{N}} nm\lambda_{n,l}(Sl)$ defines addition $m+n$ by recursion on n . For $\lambda_{n,l}(Sl)$ we often write $\lambda_{-,l}(Sl)$ since the bound variable n is not used.

It will be convenient to write an algebra form

$$\mu_{\xi}(\vec{\alpha}_i \rightarrow (\xi)_{\nu < n_i} \rightarrow \xi)_{i < k} \quad \text{as} \quad \mu_{\xi}((\rho_{i\nu}(\xi))_{\nu < n_i} \rightarrow \xi)_{i < k}.$$

DEFINITION (Type of $\mathcal{R}_l^{\tau}$). For the algebra $\iota = \mu_{\xi}((\rho_{i\nu}(\xi))_{\nu < n_i} \rightarrow \xi)_{i < k}$ and the result type τ we define the type of the recursion operator $\mathcal{R}_l^{\tau}$ to be

$$\iota \rightarrow ((\rho_{i\nu}(\iota \times \tau))_{\nu < n_i} \rightarrow \tau)_{i < k} \rightarrow \tau.$$

Here ι is the type of the recursion argument, and each $(\rho_{i\nu}(\iota \times \tau))_{\nu < n_i} \rightarrow \tau$ is called a *step type*. Usage of $\iota \times \tau$ rather than τ in the step types can be seen

as a “strengthening”, since then one has more data available to construct the value of type τ . Moreover, for recursive argument types we avoid the product type in $\iota \times \tau$ and take the two argument types ι and τ instead (“duplication”).

DEFINITION (Computation rules for $\mathcal{R}_\iota^\tau$). Let

$$\alpha_0 \rightarrow \dots \rightarrow \alpha_{m-1} \rightarrow (\xi)_{i < n} \rightarrow \xi$$

be the type of the i -th constructor C_i of ι and consider a term $C_i \vec{x}$ of type ι . We write $\vec{x}^P = x_0^P, \dots, x_{m-1}^P$ for the *parameter arguments* $x_0^{\alpha_0}, \dots, x_{m-1}^{\alpha_{m-1}}$ and $\vec{x}^R = x_0^R, \dots, x_{n-1}^R$ for the *recursive arguments* $x_m^\iota, \dots, x_{m+n-1}^\iota$. Writing $\mathcal{R}$ for $\mathcal{R}_\iota^\tau$ we take as its computation rules

$$\mathcal{R}(C_i \vec{x}) \vec{f} := f_i \vec{x} (\mathcal{R} x_0^R \vec{f}) \dots (\mathcal{R} x_{n-1}^R \vec{f}).$$

EXAMPLES.

$$\begin{aligned} \mathcal{R}_\mathbb{B}^\tau &: \mathbb{B} \rightarrow \tau \rightarrow \tau \rightarrow \tau, \\ \mathcal{R}_\mathbb{N}^\tau &: \mathbb{N} \rightarrow \tau \rightarrow (\mathbb{N} \rightarrow \tau \rightarrow \tau) \rightarrow \tau, \\ \mathcal{R}_\mathbb{P}^\tau &: \mathbb{P} \rightarrow \tau \rightarrow (\mathbb{P} \rightarrow \tau \rightarrow \tau) \rightarrow (\mathbb{P} \rightarrow \tau \rightarrow \tau) \rightarrow \tau, \\ \mathcal{R}_\mathbb{Y}^\tau &: \mathbb{Y} \rightarrow \tau \rightarrow (\mathbb{Y} \rightarrow \tau \rightarrow \mathbb{Y} \rightarrow \tau \rightarrow \tau) \rightarrow \tau, \\ \mathcal{R}_{\mathbb{L}(\rho)}^\tau &: \mathbb{L}(\rho) \rightarrow \tau \rightarrow (\rho \rightarrow \mathbb{L}(\rho) \rightarrow \tau \rightarrow \tau) \rightarrow \tau, \\ \mathcal{R}_{\rho+\sigma}^\tau &: \rho + \sigma \rightarrow (\rho \rightarrow \tau) \rightarrow (\sigma \rightarrow \tau) \rightarrow \tau, \\ \mathcal{R}_{\rho \times \sigma}^\tau &: \rho \times \sigma \rightarrow (\rho \rightarrow \sigma \rightarrow \tau) \rightarrow \tau. \end{aligned}$$

It is a helpful exercise to write out the computation rules for these particular recursion operators.

There is an important variant of recursion, where no recursive calls occur. This variant is called the *cases operator*; it distinguishes cases according to the outer constructor form. For the algebra $\iota = \mu_\xi((\rho_{i\nu}(\xi))_{\nu < n_i} \rightarrow \xi)_{i < k}$ and result type τ the type of the cases operator $\mathcal{C}_\iota^\tau$ is

$$\iota \rightarrow ((\rho_{i\nu}(\iota))_{\nu < n_i} \rightarrow \tau)_{i < k} \rightarrow \tau.$$

The simplest example (for type $\mathbb{B}$) is *if-then-else*. Another example is

$$\mathcal{C}_\mathbb{N}^\tau: \mathbb{N} \rightarrow \tau \rightarrow (\mathbb{N} \rightarrow \tau) \rightarrow \tau.$$

It could be used to define the *predecessor* function on $\mathbb{N}$ by the term

$$Pm := \mathcal{C}_\mathbb{N}^\mathbb{N} m 0 (\lambda_n n).$$

However, it is easier to define the predecessor function by the computation rules $P0 := 0$ and $P(Sn) := n$.

REMARK. When computing the value of a cases term, we do not want to (eagerly) evaluate all arguments, but rather compute the test argument first and depending on the result (lazily) evaluate at most one of the other arguments. This phenomenon is well known in functional languages; for instance, in SCHEME the **if**-construct is called a *special form* (as opposed to an operator). Therefore instead of taking the cases operator applied to a full list of arguments, one rather uses a **case**-construct to build this term; it differs from the former only in that it employs lazy evaluation. Hence the predecessor function could be written in the form

$$[\mathbf{case} \ m^{\mathbb{N}} \ \mathbf{of} \ (0 \mapsto 0 \mid Sn \mapsto n)].$$

2.3.4. Corecursion. The computation rules for $\mathcal{R}$ work from the leaves towards the root, and terminate because total ideals are finite. If, however, we deal with cototal ideals, then a similar operator is available to define functions with cototal ideals as values, namely “corecursion”.

To understand the type of a corecursion operator recall the constructor types $\kappa_i(\iota)$ of an algebra $\iota = \mu_{\xi}(\kappa_0, \dots, \kappa_{k-1})$, written as

$$(\rho_{i\nu}(\iota))_{\nu < n_i} \rightarrow \iota \quad (i < k).$$

The product of these k constructor types is isomorphic to

$$\sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota) \rightarrow \iota$$

and the type of the recursion operator $\mathcal{R}_{\iota}^{\tau}$ is isomorphic to

$$\iota \rightarrow \left(\sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota \times \tau) \rightarrow \tau \right) \rightarrow \tau.$$

Dually for the algebra ι the type of its *destructor* D_{ι} (disassembling a constructor-built object into its parts) is

$$\iota \rightarrow \sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota).$$

The corecursion operator ${}^{\text{co}}\mathcal{R}_{\iota}^{\tau}$ is used to construct a map from τ to ι by “corecursion” on the structure of ι . Its type is

$$(14) \quad \tau \rightarrow \left(\tau \rightarrow \sum_{i < k} \prod_{\nu < n_i} \rho_{i\nu}(\iota + \tau) \right) \rightarrow \iota.$$

EXAMPLES (Types and computation rules of corecursion operators).

$$\begin{aligned}
{}^{\text{co}}\mathcal{R}_{\mathbb{B}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + \mathbb{U}) \rightarrow \mathbb{B}, \\
{}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + (\mathbb{N} + \tau)) \rightarrow \mathbb{N}, \\
{}^{\text{co}}\mathcal{R}_{\mathbb{P}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + (\mathbb{P} + \tau) + (\mathbb{P} + \tau)) \rightarrow \mathbb{P}, \\
{}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + (\mathbb{Y} + \tau) \times (\mathbb{Y} + \tau)) \rightarrow \mathbb{Y}, \\
{}^{\text{co}}\mathcal{R}_{\mathbb{L}(\rho)}^{\tau} &: \tau \rightarrow (\tau \rightarrow \mathbb{U} + \rho \times (\mathbb{L}(\rho) + \tau)) \rightarrow \mathbb{L}(\rho), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau} &: \tau \rightarrow (\tau \rightarrow \rho \times (\mathbb{S}(\rho) + \tau)) \rightarrow \mathbb{S}(\rho).
\end{aligned}$$

The computation rule for each of these is defined below. For $f: \rho \rightarrow \tau$ and $g: \sigma \rightarrow \tau$ we denote $\lambda_x(\mathcal{R}_{\rho+\sigma}^{\tau} x f g)$ of type $\rho + \sigma \rightarrow \tau$ by $[f, g]$, and similiary for ternary sumtypes etcetera. The identity functions id below are of type $\iota \rightarrow \iota$ with ι the respective algebra.

$$\begin{aligned}
{}^{\text{co}}\mathcal{R}_{\mathbb{B}}^{\tau} x f &:= [\lambda_{\text{tt}}, \lambda_{\text{ff}}](f x), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau} x f &:= [\lambda_0, \lambda_y(S([\text{id}^{\mathbb{N} \rightarrow \mathbb{N}}, P_{\mathbb{N}}]y))](f x), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{P}}^{\tau} x f &:= [\lambda_1, \lambda_y(S_0([\text{id}, P_{\mathbb{P}}]y)), \lambda_y(S_1([\text{id}, P_{\mathbb{P}}]y))](f x), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau} x f &:= [\lambda_0, \lambda_{y_0, y_1}(C([\text{id}, P_{\mathbb{Y}}]y_0)([\text{id}, P_{\mathbb{Y}}]y_1))](f x), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{L}(\rho)}^{\tau} x f &:= [\lambda_{\text{[]}}, \lambda_{y_0, y_1}(y_0 :: [\text{id}, P_{\mathbb{L}(\rho)}]y_1)](f x), \\
{}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau} x f &:= (f x)_0 :: [\text{id}, P_{\mathbb{S}(\rho)}](f x)_1
\end{aligned}$$

with $(f x)_i$ the i -th component of the pair $f x$, and $P_{\alpha} := \lambda_x({}^{\text{co}}\mathcal{R}_{\alpha}^{\tau} x f)$ for $\alpha \in \{\mathbb{N}, \mathbb{P}, \mathbb{Y}, \mathbb{L}(\rho), \mathbb{S}(\rho)\}$.

DEFINITION. The (single) computation rule for ${}^{\text{co}}\mathcal{R}_{\iota}^{\tau}$ of type (14) is

$${}^{\text{co}}\mathcal{R}_{\iota}^{\tau} x f := [g_0, \dots, g_{k-1}](f x)$$

where g_i of type $\prod_{\nu < n_i} \rho_{i\nu}(\iota + \tau) \rightarrow \iota$ is defined as

$$\begin{aligned}
g_i &:= \lambda_{\vec{x}}(C_i(N_{\nu})_{\nu < n_i}) \quad \text{with } x_{\nu}: \rho_{i\nu}(\iota + \tau), \\
N_{\nu} &:= \begin{cases} x_{\nu} & \text{if } \rho_{i\nu}(\xi) \text{ is a parameter arg. type,} \\ [\text{id}^{\iota \rightarrow \iota}, P^{\tau \rightarrow \iota}]x_{\nu}^{\iota + \tau} & \text{otherwise,} \end{cases}
\end{aligned}$$

and $P := \lambda_x({}^{\text{co}}\mathcal{R}_{\iota}^{\tau} x f)$ contains the corecursive call.

REMARK. It can be difficult to read the computation rules for corecursion operators. However, it helps if we know some properties of the “step”

function f . For instance we have

$$\begin{aligned} {}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau}xf &= \begin{cases} 0 & \text{if } fx = \text{DummyL}^{\mathbb{U}+(\mathbb{N}+\tau)} \\ Sn & \text{if } fx = \text{Inr}(\text{InL}^{\mathbb{N} \rightarrow \mathbb{N}+\tau}n) \\ S({}^{\text{co}}\mathcal{R}_{\mathbb{N}}^{\tau}x'f) & \text{if } fx = \text{Inr}(\text{InR}^{\tau \rightarrow \mathbb{N}+\tau}x') \end{cases} \\ {}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}xf &= \begin{cases} 0 & \text{if } fx = \text{DummyL}^{\mathbb{U}+(\mathbb{Y}+\tau) \times (\mathbb{Y}+\tau)} \\ Cts & \text{if } fx = \text{Inr}\langle \text{InL}^{\mathbb{Y} \rightarrow \mathbb{Y}+\tau}t, \text{InL}^{\mathbb{Y} \rightarrow \mathbb{Y}+\tau}s \rangle \\ Ct({}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}yf) & \text{if } fx = \text{Inr}\langle \text{InL}^{\mathbb{Y} \rightarrow \mathbb{Y}+\tau}t, \text{InR}^{\tau \rightarrow \mathbb{Y}+\tau}y \rangle \\ C({}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}yf)s & \text{if } fx = \text{Inr}\langle \text{InR}^{\tau \rightarrow \mathbb{Y}+\tau}y, \text{InL}^{\mathbb{Y} \rightarrow \mathbb{Y}+\tau}s \rangle \\ C({}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}yf)({}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}zf) & \text{if } fx = \text{Inr}\langle \text{InR}^{\tau \rightarrow \mathbb{Y}+\tau}y, \text{InR}^{\tau \rightarrow \mathbb{Y}+\tau}z \rangle \end{cases} \\ {}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau}xf &= \begin{cases} a :: u & \text{if } fx = \langle a, \text{InL}^{\mathbb{S}(\rho) \rightarrow \mathbb{S}(\rho)+\tau}u \rangle \\ a :: {}^{\text{co}}\mathcal{R}_{\mathbb{S}(\rho)}^{\tau}x'f & \text{if } fx = \langle a, \text{InR}^{\tau \rightarrow \mathbb{S}(\rho)+\tau}x' \rangle. \end{cases} \end{aligned}$$

Recall that Map of type $(\rho \rightarrow \sigma) \rightarrow \mathbb{S}(\rho) \rightarrow \mathbb{S}(\sigma)$ maps its function argument $h: \rho \rightarrow \sigma$ over a stream u of type $\mathbb{S}(\rho)$ (see Section 2.3.2, page 31). It is an easy exercise to give an alternative definition of the function Map by means of the corecursion operator.

REMARK. It is possible to define interesting cototal objects by means of corecursion operators. For instance the rightmost infinite path in the algebra $\mathbb{Y}$ of binary trees is $t_R := {}^{\text{co}}\mathcal{R}_{\mathbb{Y}}^{\tau}x_0f_0$ with τ, x_0 arbitrary (for instance $\tau := \mathbb{U}, x_0 := \text{Dummy}^{\mathbb{U}}$) and

$$f_0x^{\tau} := \text{Inr}\langle \text{InL}^{\mathbb{Y} \rightarrow \mathbb{Y}+\tau}0, \text{InR}^{\tau \rightarrow \mathbb{Y}+\tau}x \rangle.$$

For the leftmost path we similarly have t_L .

Another example is a function converting a real number given as a Cauchy sequence of rationals together a Cauchy modulus into an infinite stream of signed digits $\{-1, 0, 1\}$. Such a function can be defined by a simple corecursion. One can extract it from a proof (using “coinduction”) of the fact that such a conversion exists.

2.4. Denotational semantics

We now set up a connection between the model $(|\mathbf{C}_{\rho}|)_{\rho}$ of partial continuous functionals described in Section 2.2.2 and the term system $\mathbf{T}^+$ from Section 2.3.2. The main point is to clarify how we can use computation rules to define an ideal z in a function space.

2.4.1. Ideals as values of terms. The general idea is to inductively define the set of tokens (U, a) that make up z . It is convenient to define the value $\llbracket \lambda_{\vec{x}} M \rrbracket$, where M is a term with free variables among $\vec{x}$. Since this value is a token set, we can define inductively the relation $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$.

For a constructor pattern $\vec{P}(\vec{x})$ and a list $\vec{V}$ of the same length and types as $\vec{x}$ we define a list $\vec{P}(\vec{V})$ of formal neighborhoods of the same length and types as $\vec{P}(\vec{x})$, by induction on $\vec{P}(\vec{x})$. $x(V)$ is the singleton list V , and for $\langle \rangle$ we take the empty list. $(\vec{P}, Q)(\vec{V}, \vec{w})$ is covered by the induction hypothesis. Finally

$$(C\vec{P})(\vec{V}) := \{ Ca^* \mid a_i^* \in P_i(\vec{V}_i) \text{ if } P_i(\vec{V}_i) \neq \emptyset, \text{ and } a_i^* = * \text{ otherwise } \}.$$

We use the following notation. $(\vec{U}, a)$ means $(U_1, (U_2, \dots (U_n, a)) \dots)$, and $(\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$ means $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ for all (finitely many) $a \in V$.

DEFINITION 2.4.1 (Inductive, of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$).

$$\frac{U_i \vdash a}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket}(V), \quad \frac{(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}(A).$$

For every constructor C and defined constant D we have

$$\frac{\vec{V} \vdash a^*}{(\vec{U}, \vec{V}, Ca^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket}(C), \quad \frac{(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}} M \rrbracket \quad \vec{W} \vdash \vec{P}(\vec{V})}{(\vec{U}, \vec{W}, a) \in \llbracket \lambda_{\vec{x}} D \rrbracket}(D)$$

with one such rule (D) for every computation rule $D\vec{P}(\vec{y}) = M$.

The *height* of a derivation of $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ is defined as usual, by adding 1 at each rule. We define its *D-height* similarly, where only rules (D) count.

We begin with some simple consequences of this definition.

LEMMA 2.4.2. *The following transformations preserve D-height:*

- (15) $\vec{V} \vdash \vec{U} \rightarrow (\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \rightarrow (\vec{V}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$,
- (16) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket \leftrightarrow (\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad \text{if } y \notin \text{FV}(M)$,
- (17) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y}(My) \rrbracket \leftrightarrow (\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad \text{if } y \notin \text{FV}(M)$,
- (18) $(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}}(M(\vec{P}(\vec{y}))) \rrbracket \leftrightarrow (\vec{U}, \vec{P}(\vec{V}), a) \in \llbracket \lambda_{\vec{x}, \vec{z}}(M(\vec{z})) \rrbracket$.

PROOF. (15) and (16) are both proved by easy inductions on the respective derivations.

(17). Assume $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y}(My) \rrbracket$. By (A) we then have W such that $(\vec{U}, V, W) \subseteq \llbracket \lambda_{\vec{x}, y} y \rrbracket$ (i.e., $V \vdash W$) and $(\vec{U}, V, W, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. By (15) from the latter we obtain $(\vec{U}, V, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. Now since $y \notin \text{FV}(M)$, (16) yields $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, as required. Conversely, assume $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$. Since $y \notin \text{FV}(M)$, (16) yields $(\vec{U}, V, V, a) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$. Clearly we have $(\vec{U}, V, V) \subseteq \llbracket \lambda_{\vec{x}, y} y \rrbracket$. Hence by (A) $(\vec{U}, V, a) \in \llbracket \lambda_{\vec{x}, y}(My) \rrbracket$, as required. Notice that the *D-height* did not change in these transformations.

(18). By induction on $\vec{P}$, with a side induction on M . We distinguish cases on M . The cases x_i , C and D follow immediately from (16). In case MN the following are equivalent by the side induction hypothesis:

$$\begin{aligned} (\vec{U}, \vec{V}, a) &\in \llbracket \lambda_{\vec{x}, \vec{y}}((MN)(\vec{P}(\vec{y}))) \rrbracket \\ \exists W((\vec{U}, \vec{V}, W, a) &\in \llbracket \lambda_{\vec{x}, \vec{y}}(M(\vec{P}(\vec{y}))) \rrbracket \wedge (\vec{U}, \vec{V}, W) \subseteq \llbracket \lambda_{\vec{x}, \vec{y}}(N(\vec{P}(\vec{y}))) \rrbracket) \\ \exists W((\vec{U}, \vec{P}(\vec{V}), W, a) &\in \llbracket \lambda_{\vec{x}, \vec{z}}(M(\vec{z})) \rrbracket \wedge (\vec{U}, \vec{P}(\vec{V}), W) \subseteq \llbracket \lambda_{\vec{x}, \vec{z}}(N(\vec{z})) \rrbracket) \\ (\vec{U}, \vec{P}(\vec{V}), a) &\in \llbracket \lambda_{\vec{x}, \vec{z}}((MN)(\vec{z})) \rrbracket. \end{aligned}$$

The final case is where M is z_i . Then we have to show

$$(\vec{U}, \vec{V}, a) \in \llbracket \lambda_{\vec{x}, \vec{y}}(P(\vec{y})) \rrbracket \leftrightarrow P(\vec{V}) \vdash a.$$

We distinguish cases on $P(\vec{y})$. If $P(\vec{y})$ is y_j , then both sides are equivalent to $V_j \vdash a$. In case $P(\vec{y})$ is $(C\vec{Q})(\vec{y})$ the following are equivalent:

$$\begin{aligned} (\vec{U}, \vec{V}, a) &\in \llbracket \lambda_{\vec{x}, \vec{y}}((C\vec{Q})(\vec{y})) \rrbracket \\ (\vec{U}, \vec{V}, a) &\in \llbracket \lambda_{\vec{x}, \vec{y}}(C\vec{Q}(\vec{y})) \rrbracket \\ (\vec{U}, \vec{Q}(\vec{V}), a) &\in \llbracket \lambda_{\vec{x}, \vec{u}}(C\vec{u}) \rrbracket \quad \text{by IH for } \vec{Q}(\vec{y}) \\ (\vec{U}, \vec{Q}(\vec{V}), a) &\in \llbracket \lambda_{\vec{x}} C \rrbracket \quad \text{by (17)} \\ \exists_{a^*} (a = C\vec{a}^* \wedge \vec{Q}(\vec{V}) \vdash a^*) & \\ C\vec{Q}(\vec{V}) \vdash a. & \quad \square \end{aligned}$$

Let $\sim$ denote the equivalence relation on formal neighborhoods generated by entailment, i.e., $U \sim V$ means $(U \vdash V) \wedge (V \vdash U)$.

LEMMA 2.4.3.

(19) *If $\vec{U} \vdash \vec{P}(\vec{V})$, then there are $\vec{W}$ such that $\vec{U} \sim \vec{P}(\vec{W})$ and $\vec{W} \vdash \vec{V}$.*

PROOF. By induction on $\vec{P}$. The cases x and $\langle \rangle$ are clear, and in case $\vec{P}, Q$ we can apply the induction hypothesis. It remains to treat the case $C\vec{P}(\vec{x})$. Since $U \vdash C\vec{P}(\vec{V})$ there is a $\vec{b}_0^*$ such that $C\vec{b}_0^* \in U$. Let

$$U_i := \{ a \mid \exists_{a^*} (C\vec{a}^* \in U \wedge a = a_i^*) \}.$$

For the constructor pattern $C\vec{x}$ consider $C\vec{U}$. By definition

$$C\vec{U} = \{ C\vec{a}^* \mid a_i^* \in U_i \text{ if } U_i \neq \emptyset, \text{ and } a_i^* = * \text{ otherwise} \}.$$

We first show $U \sim C\vec{U}$. Assume $C\vec{a}^* \in C\vec{U}$. For each i , if $U_i \neq \emptyset$, then there is an a_i^* such that $C\vec{a}_i^* \in U$ and $a_{ii}^* = a_i^*$, and if $U_i = \emptyset$ then $a_i^* = *$. Hence

$$U \supseteq \{ C\vec{a}_i^* \mid U_i \neq \emptyset \} \cup \{ C\vec{b}_0^* \} \vdash C\vec{a}^*.$$

Conversely assume $C\vec{a}^* \in U$. We define $C\vec{b}^* \in C\vec{U}$ by $b_i^* = a_i^*$ if $a_i^* \neq *$, $b_i^* = *$ if $U_i = \emptyset$, and otherwise (i.e., if $a_i^* = *$ and $U_i \neq \emptyset$) take an arbitrary $b_i^* \in U_i$. Clearly $\{C\vec{b}^*\} \vdash C\vec{a}^*$.

By definition $\vec{U} \vdash \vec{P}(\vec{V})$. Hence by induction hypothesis there are $\vec{W}$ such that $\vec{U} \sim \vec{P}(\vec{W})$ and $\vec{W} \vdash \vec{V}$. Therefore $U \sim C\vec{U} \sim C\vec{P}(\vec{W})$. $\square$

LEMMA 2.4.4 (Unification). *If $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$, then $\vec{P}_1, \dots, \vec{P}_n$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that*

$$(\vec{P}_1\vartheta)(\vec{W}) = \dots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n).$$

PROOF. Assume $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$. Then $\vec{P}_1(\vec{V}_1), \dots, \vec{P}_n(\vec{V}_n)$ are componentwise consistent and hence $\vec{P}_1, \dots, \vec{P}_n$ are unifiable with a most general unifier ϑ . We now proceed by induction on $\vec{P}_1, \dots, \vec{P}_n$. If they are either all empty or all variables the claim is trivial. In the case $(\vec{P}_1, P_1), \dots, (\vec{P}_n, P_n)$ it follows from the linearity condition on variables that a most general unifier of $(\vec{P}_1, P_1), \dots, (\vec{P}_n, P_n)$ is the union of most general unifiers of $\vec{P}_1, \dots, \vec{P}_n$ and of $P_1, \dots, P_n$. Hence the induction hypothesis applies. In the case $C\vec{P}_1, \dots, C\vec{P}_n$ the assumption $C\vec{P}_1(\vec{V}_1) \sim \dots \sim C\vec{P}_n(\vec{V}_n)$ implies $\vec{P}_1(\vec{V}_1) \sim \dots \sim \vec{P}_n(\vec{V}_n)$ and hence again the induction hypothesis applies. The remaining case is when some are variables and the other ones of the form $C\vec{P}_i$, say $x, C\vec{P}_2, \dots, C\vec{P}_n$. By assumption

$$V_1 \sim C\vec{P}_2(\vec{V}_2) \sim \dots \sim C\vec{P}_n(\vec{V}_n).$$

By induction hypothesis we obtain the required $\vec{W}$ such that

$$(\vec{P}_2\vartheta)(\vec{W}) = \dots = (\vec{P}_n\vartheta)(\vec{W}) \sim \vec{P}_2(\vec{V}_2) \sim \dots \sim \vec{P}_n(\vec{V}_n). \quad \square$$

LEMMA 2.4.5 (Consistency). $\llbracket \lambda_{\vec{x}} M \rrbracket$ is consistent.

PROOF. Let $(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ for $i = 1, 2$. By coherence it suffices to prove that $(\vec{U}_1, a_1)$ and $(\vec{U}_2, a_2)$ are consistent. We shall prove this by induction on the maximum of the D -heights and a side induction on the maximum of the heights.

Case (V). Let $(\vec{U}_1, a_1), (\vec{U}_2, a_2) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket$, and assume that $\vec{U}_1$ and $\vec{U}_2$ are componentwise consistent. Then $U_{1i} \vdash a_1$ and $U_{2i} \vdash a_2$. Since $U_{1i} \cup U_{2i}$ is consistent, a_1 and a_2 must be consistent as well.

Case (C). For $i = 1, 2$ we have

$$\frac{\vec{V}_i \vdash a_i^*}{(\vec{U}_i, \vec{V}_i, C\vec{a}_i^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket}.$$

Assume $\vec{U}_1, \vec{V}_1$ and $\vec{U}_2, \vec{V}_2$ are componentwise consistent. The consistency of $C\vec{a}_1^*$ and $C\vec{a}_2^*$ follows from $\vec{V}_i \vdash a_i^*$ and the consistency of $\vec{V}_1$ and $\vec{V}_2$.

Case (A). For $i = 1, 2$ we have

$$\frac{(\vec{U}_i, V_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}_i, V_i) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}.$$

Assume $\vec{U}_1$ and $\vec{U}_2$ are componentwise consistent. By the side induction hypothesis for the right premises $V_1 \cup V_2$ is consistent. Hence by the side induction hypothesis for the left hand sides a_1 and a_2 are consistent.

Case (D). For $i = 1, 2$ we have

$$\frac{(\vec{U}_i, \vec{V}_i, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket \quad \vec{W}_i \vdash \vec{P}_i(\vec{V}_i)}{(\vec{U}_i, \vec{W}_i, a_i) \in \llbracket \lambda_{\vec{x}} D \rrbracket} (D)$$

for computation rules $D\vec{P}_i(\vec{y}_i) = M_i(\vec{y}_i)$. Assume $\vec{U}_1, \vec{W}_1$ and $\vec{U}_2, \vec{W}_2$ are componentwise consistent; we must show that a_1 and a_2 are consistent. Since $\vec{W}_1 \cup \vec{W}_2 \vdash \vec{P}_i(\vec{V}_i)$ for $i = 1, 2$, by (19) there are $\vec{V}'_1, \vec{V}'_2$ such that $\vec{V}'_i \vdash \vec{V}_i$ and $\vec{W}_1 \cup \vec{W}_2 \sim \vec{P}_i(\vec{V}'_i)$. Then by Lemma 2.4.4 (on unification) there are $\vec{W}$ such that $(\vec{P}_1\vartheta)(\vec{W}) = (\vec{P}_2\vartheta)(\vec{W}) \sim \vec{P}_i(\vec{V}'_i) \vdash \vec{P}_i(\vec{V}_i)$ for $i = 1, 2$, where ϑ is the most general unifier of $\vec{P}_1$ and $\vec{P}_2$. But then also

$$(\vec{y}_i\vartheta)(\vec{W}) \vdash \vec{V}_i,$$

and hence by (15) we have

$$(\vec{U}_i, (\vec{y}_i\vartheta)(\vec{W}), a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket$$

with smaller D -height. Now (18) gives

$$(\vec{U}_i, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{z}} M_i(\vec{y}_i)\vartheta \rrbracket$$

without increasing the D -height. Notice that $M_1(\vec{y}_i)\vartheta = M_2(\vec{y}_i)\vartheta$ by our condition on computation rules. Hence the induction hypothesis applied to $(\vec{U}_1, \vec{W}, a_1), (\vec{U}_2, \vec{W}, a_2) \in \llbracket \lambda_{\vec{x}, \vec{z}} M_1(\vec{y}_1)\vartheta \rrbracket$ implies the consistency of a_1 and a_2 , as required. $\square$

LEMMA 2.4.6 (Deductive closure). $\llbracket \lambda_{\vec{x}} M \rrbracket$ is deductively closed, i.e., if $W \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$ and $W \vdash (\vec{V}, b)$, then $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$.

PROOF. By induction on the maximum of the D -heights and a side induction on the maximum of the heights of $W \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$. We distinguish cases on the last rule of these derivations (which is determined by M).

Case (V). For all $(\vec{U}, a) \in W$ we have

$$\frac{U_i \vdash a}{(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} x_i \rrbracket}.$$

We must show $V_i \vdash b$. By assumption $W \vdash (\vec{V}, b)$, hence $W\vec{V} \vdash b$. It suffices to prove $V_i \vdash W\vec{V}$. Let $c \in W\vec{V}$; we show $V_i \vdash c$. There are $\vec{U}$ such that $\vec{V} \vdash \vec{U}$ and $(\vec{U}, c) \in W$. But then by the above $U_i \vdash c$, hence $V_i \vdash U_i \vdash c$.

Case (A). Let $W = \{(\vec{U}_1, a_1), \dots, (\vec{U}_n, a_n)\}$. For each $(\vec{U}_i, a_i) \in W$ there is U_i such that

$$\frac{(\vec{U}_i, U_i, a_i) \in \llbracket \lambda_{\vec{x}} M \rrbracket \quad (\vec{U}_i, U_i) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket}{(\vec{U}_i, a_i) \in \llbracket \lambda_{\vec{x}}(MN) \rrbracket}.$$

Define $U := \bigcup \{U_i \mid \vec{V} \vdash \vec{U}_i\}$. We first show that U is consistent. Let $a, b \in U$. There are i, j such that $a \in U_i$, $b \in U_j$ and $\vec{V} \vdash \vec{U}_i, \vec{U}_j$. Then $\vec{U}_i$ and $\vec{U}_j$ are consistent; hence since $\llbracket \lambda_{\vec{x}} N \rrbracket$ is consistent by Lemma 2.4.5 a and b are consistent as well.

Next we show $(\vec{V}, U) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket$. Let $a \in U$; we show $(\vec{V}, a) \in \llbracket \lambda_{\vec{x}} N \rrbracket$. Fix i such that $a \in U_i$ and $\vec{V} \vdash \vec{U}_i$, and let $W_i := \{(\vec{U}_i, b) \mid b \in U_i\} \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket$. Since by the side induction hypothesis $\llbracket \lambda_{\vec{x}} N \rrbracket$ is deductively closed it suffices to prove $W_i \vdash (\vec{V}, a)$, i.e., $\{b \mid b \in U_i \wedge \vec{V} \vdash \vec{U}_i\} \vdash a$. But the latter set equals U_i , and $a \in U_i$.

Finally we show $(\vec{V}, U, b) \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket$. Let

$$W' := \{(\vec{U}_1, U_1, a_1), \dots, (\vec{U}_n, U_n, a_n)\} \subseteq \llbracket \lambda_{\vec{x}} M \rrbracket.$$

By side induction hypothesis it suffices to prove that $W' \vdash (\vec{V}, U, b)$, i.e., $\{a_i \mid \vec{V} \vdash \vec{U}_i \wedge U \vdash U_i\} \vdash b$. But by definition of U the latter set equals $\{a_i \mid \vec{V} \vdash \vec{U}_i\}$, which in turn entails b because by assumption $W \vdash (\vec{V}, b)$.

Now we can use (A) to infer $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, as required.

Case (C). Assume $W \subseteq \llbracket \lambda_{\vec{x}} C \rrbracket$. Then W consists of $(\vec{U}, \vec{U}', Ca^*)$ such that $\vec{U}' \vdash a^*$. Assume further $W \vdash (\vec{V}, \vec{V}', b)$. Then

$$\{Ca^* \mid \exists \vec{U}, \vec{U}', ((\vec{U}, \vec{U}', Ca^*) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}')\} \vdash b.$$

By definition of entailment b has the form Cb^* such that

$$W_i := \{a \mid \exists \vec{U}, \vec{U}', a^* (a = a_i^* \wedge (\vec{U}, \vec{U}', Ca^*) \in W \wedge \vec{V} \vdash \vec{U} \wedge \vec{V}' \vdash \vec{U}')\} \vdash b_i^*.$$

We must show $(\vec{V}, \vec{V}', Cb^*) \in \llbracket \lambda_{\vec{x}} C \rrbracket$, i.e., $\vec{V}' \vdash b^*$. It suffices to show $V'_i \vdash W_i$, for every i . Let $a \in W_i$. Then there are $\vec{U}, \vec{U}', a^*$ such that $a = a_i^*$, $(\vec{U}, \vec{U}', Ca^*) \in W$ and $\vec{V}' \vdash \vec{U}'$. Hence $V'_i \vdash U'_i \vdash a_i^* = a$.

Case (D). Let $W = \{(\vec{U}_1, \vec{U}_1'', a_1), \dots, (\vec{U}_n, \vec{U}_n'', a_n)\}$. For every i there is an $\vec{U}'_i$ such that

$$\frac{(\vec{U}_i, \vec{U}'_i, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket \quad \vec{U}_i'' \vdash \vec{P}_i(\vec{U}'_i)}{(\vec{U}_i, \vec{U}_i'', a_i) \in \llbracket \lambda_{\vec{x}} D \rrbracket}$$

for $D\vec{P}_i(\vec{y}_i) = M_i(\vec{y}_i)$ a computation rule. Assume $W \vdash (\vec{V}, \vec{V}'', b)$. We must prove $(\vec{V}, \vec{V}'', b) \in \llbracket \lambda_{\vec{x}} D \rrbracket$. Let

$$I := \{ i \mid 1 \leq i \leq n \wedge \vec{V} \vdash \vec{U}_i \wedge \vec{V}'' \vdash \vec{U}_i'' \}.$$

Then $\{ a_i \mid i \in I \} \vdash b$, hence $I \neq \emptyset$. For $i \in I$ we have $\vec{V}'' \vdash \vec{U}_i'' \vdash \vec{P}_i(\vec{U}_i')$, hence by (19) there are $\vec{V}_i'$ such that $\vec{V}'' \sim \vec{P}_i(\vec{V}_i')$ and $\vec{V}_i' \vdash \vec{U}_i'$. In particular for $i, j \in I$

$$\vec{V}'' \sim \vec{P}_i(\vec{V}_i') \sim \vec{P}_j(\vec{V}_j').$$

To simplify notation assume $I = \{1, \dots, m\}$. Hence by the unification lemma $\vec{P}_1, \dots, \vec{P}_m$ are unifiable with a most general unifier ϑ and there exists $\vec{W}$ such that

$$(\vec{P}_1\vartheta)(\vec{W}) = \dots = (\vec{P}_m\vartheta)(\vec{W}) \sim \vec{P}_1(\vec{V}_1') \sim \dots \sim \vec{P}_m(\vec{V}_m').$$

Let $i, j \in I$. Then by the conditions on computation rules $M_i\vartheta = M_j\vartheta$. Also $(\vec{y}_i\vartheta)(\vec{W}) \vdash \vec{V}_i' \vdash \vec{U}_i'$. Therefore by (15)

$$(\vec{V}, (\vec{y}_i\vartheta)(\vec{W}), a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i) \rrbracket$$

and hence by (18)

$$(\vec{V}, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_i(\vec{y}_i\vartheta) \rrbracket.$$

But $M_i(\vec{y}_i\vartheta) = M_i\vartheta = M_1\vartheta = M_1(\vec{y}_1\vartheta)$ and hence for all $i \in I$

$$(\vec{V}, \vec{W}, a_i) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1\vartheta) \rrbracket.$$

Therefore $X := \{ (\vec{V}, \vec{W}, a_i) \mid i \in I \} \subseteq \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1\vartheta) \rrbracket$. Since $\{ a_i \mid i \in I \} \vdash b$, we have $X \vdash (\vec{V}, \vec{W}, b)$ and hence the induction hypothesis implies $(\vec{V}, \vec{W}, b) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1\vartheta) \rrbracket$. Using (18) again we obtain $(\vec{V}, (\vec{y}_i\vartheta)(\vec{W}), b) \in \llbracket \lambda_{\vec{x}, \vec{y}_i} M_1(\vec{y}_1) \rrbracket$. Since $\vec{V}'' \sim \vec{P}_1(\vec{V}_1') \sim \vec{P}_1((\vec{y}_1\vartheta)(\vec{W}))$ we obtain $(\vec{V}, \vec{V}'', b) \in \llbracket \lambda_{\vec{x}} D \rrbracket$, by (D). $\square$

COROLLARY. $\llbracket \lambda_{\vec{x}} M \rrbracket$ is an ideal.

2.4.2. Preservation of values. We now prove that Definition 2.4.1 of the denotation of a term is reasonable in the sense that this denotation is not changed by an application of the standard (β - and η -) conversions or a computation rule. For the β -conversion part of this proof it is helpful to first introduce a more standard notation, which involves variable environments.

DEFINITION. Assume that all free variables in M are among $\vec{x}$. Let $\llbracket M \rrbracket_{\vec{x}}^{\vec{U}} := \{ b \mid (\vec{U}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket \}$ and $\llbracket M \rrbracket_{\vec{x}, \vec{y}}^{\vec{u}, \vec{V}} := \bigcup_{\vec{U} \subseteq \vec{u}} \llbracket M \rrbracket_{\vec{x}, \vec{y}}^{\vec{U}, \vec{V}}$.

From (16) we obtain $\llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V} = \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$ if $y \notin \text{FV}(M)$, and similarly for ideals $\vec{u}, v$ instead of $\vec{U}, V$. We have a useful monotonicity property, which follows from the deductive closure of $\llbracket \lambda_{\vec{x}} M \rrbracket$.

LEMMA 2.4.7. (a) If $\vec{V} \vdash \vec{U}$, $a \vdash b$ and $a \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$, then $b \in \llbracket M \rrbracket_{\vec{x}}^{\vec{V}}$.
 (b) If $\vec{v} \supseteq \vec{u}$, $a \vdash b$ and $a \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}$, then $b \in \llbracket M \rrbracket_{\vec{x}}^{\vec{v}}$.

PROOF. (a) $\vec{V} \vdash \vec{U}$, $a \vdash b$ and $(\vec{U}, a) \in \llbracket \lambda_{\vec{x}} M \rrbracket$ together imply $(\vec{V}, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket$, by the deductive closure of $\llbracket \lambda_{\vec{x}} M \rrbracket$. (b) follows from (a). $\square$

LEMMA 2.4.8. (a) $\llbracket x_i \rrbracket_{\vec{x}}^{\vec{U}} = \overline{U}_i$ and $\llbracket x_i \rrbracket_{\vec{x}}^{\vec{u}} = u_i$.
 (b) $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{U}} = \{ (V, b) \mid b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V} \}$ and $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}} = \{ (V, b) \mid b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, V} \}$.
 (c) $\llbracket MN \rrbracket_{\vec{x}}^{\vec{U}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}$ and $\llbracket MN \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{u}} \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}$.

PROOF. (b) It suffices to prove the first part. But $(V, b) \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{U}}$ and $b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{U}, V}$ are both equivalent to $(\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}, y} M \rrbracket$.

(c) For the first part we argue as follows.

$$\begin{aligned} c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}} &\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\ &\leftrightarrow \exists_V ((\vec{U}, V) \subseteq \llbracket \lambda_{\vec{x}} N \rrbracket \wedge (\vec{U}, V, c) \in \llbracket \lambda_{\vec{x}} M \rrbracket) \\ &\leftrightarrow (\vec{U}, c) \in \llbracket \lambda_{\vec{x}} (MN) \rrbracket \quad \text{by (A)} \\ &\leftrightarrow c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{U}}. \end{aligned}$$

The second part is an easy consequence:

$$\begin{aligned} c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}} \llbracket N \rrbracket_{\vec{x}}^{\vec{u}} &\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}) \\ &\leftrightarrow \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{u}}} \exists_{\vec{U} \subseteq \vec{u}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\ &\leftrightarrow \exists_{\vec{U}_1 \subseteq \vec{u}} \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_1}} \exists_{\vec{U} \subseteq \vec{u}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\ &\leftrightarrow^{(*)} \exists_{\vec{U} \subseteq \vec{u}} \exists_{V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}} ((V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}) \\ &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}} \llbracket N \rrbracket_{\vec{x}}^{\vec{U}}) \\ &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} (c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{U}}) \quad \text{by the first part} \\ &\leftrightarrow c \in \llbracket MN \rrbracket_{\vec{x}}^{\vec{u}}. \end{aligned}$$

Here is the proof of the equivalence marked (*). The upward direction is obvious. For the downward direction we use monotonicity. Assume $\vec{U}_1 \subseteq \vec{u}$, $V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_1}$, $\vec{U} \subseteq \vec{u}$ and $(V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}}$. Let $\vec{U}_2 := \vec{U}_1 \cup \vec{U} \subseteq \vec{u}$. Then by monotonicity $V \subseteq \llbracket N \rrbracket_{\vec{x}}^{\vec{U}_2}$ and $(V, c) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{U}_2}$. $\square$

COROLLARY. $\llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}, v} = \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, v}$.

PROOF.

$$\begin{aligned}
b \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}} v &\leftrightarrow \exists_{V \subseteq v} ((V, b) \in \llbracket \lambda_y M \rrbracket_{\vec{x}}^{\vec{u}}) \\
&\leftrightarrow \exists_{V \subseteq v} (b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, V}) \quad \text{by the lemma, part (b)} \\
&\leftrightarrow b \in \llbracket M \rrbracket_{\vec{x}, y}^{\vec{u}, v}. \quad \square
\end{aligned}$$

LEMMA 2.4.9 (Substitution). $\llbracket M(z) \rrbracket_{\vec{x}, z}^{\vec{u}, [N]_{\vec{z}}^{\vec{u}}} = \llbracket M(N) \rrbracket_{\vec{x}}^{\vec{u}}.$

PROOF. By induction on M , and cases on the form of M .

Case $\lambda_y M$. For readability we leave out $\vec{x}$ and $\vec{u}$.

$$\begin{aligned}
\llbracket \lambda_y M(z) \rrbracket_z^{[N]} &= \{ (V, b) \mid b \in \llbracket M(z) \rrbracket_{z, y}^{[N], V} \} \\
&= \{ (V, b) \mid b \in \llbracket M(N) \rrbracket_y^V \} \quad \text{by induction hypothesis} \\
&= \llbracket \lambda_y M(N) \rrbracket \quad \text{by the last lemma, part (b)} \\
&= \llbracket (\lambda_y M)(N) \rrbracket.
\end{aligned}$$

The other cases are easy. $\square$

LEMMA 2.4.10 (Preservation of values, β). $\llbracket (\lambda_y M(y))N \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M(N) \rrbracket_{\vec{x}}^{\vec{u}}.$

PROOF. Again we leave out $\vec{x}$, $\vec{u}$. By the last two lemmata and the corollary, $\llbracket (\lambda_y M(y))N \rrbracket = \llbracket \lambda_y M(y) \rrbracket \llbracket N \rrbracket = \llbracket M(y) \rrbracket_y^{[N]} = \llbracket M(N) \rrbracket.$ $\square$

LEMMA 2.4.11 (Preservation of values, η). $\llbracket \lambda_y (My) \rrbracket_{\vec{x}}^{\vec{u}} = \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}$ if $y \notin \text{FV}(M).$

PROOF.

$$\begin{aligned}
(V, b) \in \llbracket \lambda_y (My) \rrbracket_{\vec{x}}^{\vec{u}} &\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} ((\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}, y} (My) \rrbracket) \\
&\leftrightarrow \exists_{\vec{U} \subseteq \vec{u}} ((\vec{U}, V, b) \in \llbracket \lambda_{\vec{x}} M \rrbracket) \quad \text{by (17)} \\
&\leftrightarrow (V, b) \in \llbracket M \rrbracket_{\vec{x}}^{\vec{u}}. \quad \square
\end{aligned}$$

LEMMA 2.4.12 (Preservation of values, computation rules). *For every computation rule $D\vec{P}(\vec{y}) = M$ of a defined constant D*

$$\llbracket \lambda_{\vec{y}} D\vec{P}(\vec{y}) \rrbracket = \llbracket \lambda_{\vec{y}} M \rrbracket.$$

PROOF. The following are equivalent:

$$\begin{aligned}
(\vec{V}, b) &\in \llbracket \lambda_{\vec{y}} D\vec{P}(\vec{y}) \rrbracket \\
(\vec{P}(\vec{V}), b) &\in \llbracket D \rrbracket = \llbracket \lambda_{\vec{z}} D\vec{z} \rrbracket \quad \text{by (18)} \\
(\vec{V}, b) &\in \llbracket \lambda_{\vec{y}} M \rrbracket,
\end{aligned}$$

where the last step is by definition. $\square$