From Implicit via Inductive to Explicit Definitions*

Gerard R. Renardel de Lavalette
Department of Computing Science, University of Groningen
P.O. Box 800, 9700 AV Groningen, The Netherlands
E-mail: grl@cs.rug.nl

Abstract
This paper reports on a method to provide general implicit descriptions with a sound logical semantics. This method has been applied in the specification languages COLD and VVSL.

1 Introduction
Besides precision through formality, another desired feature of specification and design languages is generality. It should be possible for the user of such a language to create abstract and high-level descriptions, e.g. when the overall structure of the system or model at hand is to be formulated. During the subsequent development process, ranging from detailed specifications based on the overall specification to the implementation (possibly via transformations), abstraction and generality will give way to concrete and explicit formulations.

1.1 Definitions
Here we consider definitions in the light of generality. In high-level and general descriptions, many definitions are implicit: an object or a collection of objects is often not explicitly defined, but only described in terms of itself. A standard example here is the characterisation of the collection $N$ of natural numbers in terms of 0 and $S$ (successor) by

$$0 \in N \land \forall x(x \in N \rightarrow Sx \in N). \tag{1}$$

Another more telling example is the description of the collection $F(C)$ of finite sets of objects of a collection $C$:

$$0 \in F(C) \land \forall x \in C(\{x\} \in F(C)) \land \forall XY \in F(C)(X \cup Y \in F(C)) \tag{2}$$

Here we assume the usual explicit definitions $\emptyset$, $\{\}$ and $\cup$ in terms of $\in$ to be given beforehand, together with the extensionality property $\forall XY(\forall a(a \in X \iff$

---

\(\alpha \in Y \leftrightarrow X = Y\). When we compare (2) with the more common definition (using the insertion function \(i : C \times \mathcal{F}(C) \rightarrow \mathcal{F}(C)\)):

\[ \emptyset \in \mathcal{F}(C) \land \forall x \in C \forall X \in \mathcal{F}(C)(i(x, X) \in \mathcal{F}(C)) \]

which is usual in algebraic specifications and declarative programming, we have the opinion that (3) is more biased to implementation (e.g. by term rewriting) and hence less abstract than (2). In section 4, we shall come back to this example when we give an inductive definition of the cardinality function which runs parallel to (2).

In (1) and (2), the intention is to characterise \(N\) resp. \(\mathcal{F}(C)\) as the smallest collection satisfying the description. This intended meaning is obtained when (1) and (2) are considered as inductive definitions. In this paper, we study the mechanisms involved in this interpretation of implicit definitions.

### 1.2 COLD and VVSL

The method of making implicit definitions explicit by considering them as inductive definitions has been used in the logical semantics of the wide spectrum languages COLD and VVSL. COLD (Common Object-oriented Language for Design) is developed at Philips Research Eindhoven, mainly by Hans Jonkers. We refer to [3] for the formal definition of the kernel language COLD-K and its semantics and to [2] for a textbook on formal specification and design based on COLD. VVSL (for VIP VDM Specification Language) is a specification language designed by Kees Middelburg in the ESPRIT project VIP (VDM Interfaces for the PCTE). VVSL is based on VDM, temporal logic and the modularisation and parametrisation mechanisms of COLD-K. See [8], [9].

The definition of the logical semantics of COLD and VVSL uses MPL\(_w\) (Many-sorted Partial Infinitary Logic). This is an extension of classical first-order logic with sorts, definedness predicates \(\downarrow_S\) for every sort \(S\) (\(t \downarrow_S\) means: \(t\) is a defined term of sort \(S\)), partial functions (i.e. \(f(x) \downarrow\) does not hold automatically), descriptions \(\exists x : S(A)\) (meaning: the unique \(x\) of sort \(S\) satisfying \(A\) if such an \(x\) exists, otherwise undefined) and countably infinite conjunctions \(\bigwedge_n A_n\) and disjunctions \(\bigvee_n A_n\). MPL\(_w\) is a definitional extension of \(L_{w,w}\) (first-order logic with \(\bigwedge_n A_n\) and \(\bigvee_n A_n\); see [5], [7]) and shares with it a Completeness Theorem and the Interpolation Property. See [6] for more information on MPL\(_w\). In this paper, we work with logics like MPL\(_w\): as we shall see further on, infinite disjunctions are used to make inductive definitions.

### 1.3 Inductive and explicit definitions

In section 1.1 we mentioned that implicit definitions like (1) can be considered as inductive definitions, viz.

\[N\) the smallest predicate satisfying \(N(0) \land \forall x(N(x) \rightarrow N(x + 1)). \]
But this is only one way to present inductive definitions, another is to define $N$ as the least fixpoint $\mu \Phi$ of a monotonic operator $\Phi$ on predicates: the definition then reads

$$N \eqdef \mu(x \mid x = 0 \lor \exists y(x = y + 1 \land P(y))).$$

Definition (5) is, in some sense, more explicit than (4), and it is straightforward to define its semantics in logical terms if the logic used allows the definition of least fixpoints of predicate operators. This is e.g. the case for the logic MPL$\mu$, a forerunner of MPL$\omega$ introduced in [4] for the definition of the semantics of a fragment of COLD. (N.B. MPL$\mu$ is called MPL in [4], but the latter name is used to denote the finitary fragment of MPL$\omega$ in subsequent publications, e.g. [11].) On the other hand, definitions of type (5) have the disadvantage that they are harder to read and more difficult to devise. This makes it interesting to consider the following question: how to define the logical semantics of definitions like (4)?

In second-order logics with enough comprehension, this is no problem: predicate definitions of the form

$$P$$

is the smallest predicate satisfying $A(P)$

can be made explicit by $P = \exists X (X \land \forall Y (A(Y) \rightarrow \forall Y(X \land Y))).$ However, second-order logic was thought to be far too heavy and not well enough understood to provide the logical semantics of specification languages. As to COLD-K, this led to considering weaker extensions of first-order logic such as MPL$\mu$ and MPL$\omega$, of which the second one has been chosen, for the following reasons:

- the Interpolation Property, desirable for an adequate semantics of modularisation and parametrisation, holds for MPL$\omega$ and not for MPL$\mu$ (see [10]);
- fixpoints $\mu \Phi$ of continuous predicate functions $\Phi$ can be expressed explicitly in MPL$\omega$ as follows: $\mu \Phi = \bigvee_n A_n$ with $A_0 = \bot, A_{n+1} = \Phi(A_n)$.

So the problem is solved if we can find a method to transform formulae as in (4) into predicate operators as in (5). The rest of this paper is devoted to finding and elaborating such a method.

2 Preliminaries

In the rest of this paper, we consider the situation from two perspectives: set theory and logic. The logic we have in mind is MPL$\omega$, but for simplicity we restrict it most of the time to $L_{\omega \cdot 1}$ (first-order logic with equality and countably infinite conjunctions and disjunctions; no sorts, no partiality, no descriptions). We shall use a definitional extension $L$ in which the following are present:

- term substitution $[x := t]A$;
• predicate variables $X, Y, \ldots$, all assumed to be unary (for the sake of simplicity), with the equivalent notations $xt$ and $t \epsilon X$;

• defined predicates $\{x|A\}$, with the meaning given by $t \epsilon \{x|A\} \equiv \{x|A\}(t) \equiv [x := t]A$;

• predicate substitution of the defined predicate $D$ for the predicate variable $X$ in the formula $A$ denoted by $[X := D]A$;

• predicate substitution $[X^+ := D]$ and $[X^- := D]$, which only act on the positive resp. negative occurrences of $X$;

• predicate functions $\Lambda X.A$ and predicate operators $\Lambda X.\{x|A\}$, with the meaning given by $(\Lambda X.A)D = [X := D]A$, $(\Lambda X.\{x|A\})D = \{x[X := D]A\}$.

$L$ will sometimes be used to denote the collection of formulae of the language of $\mathcal{L}$; $\mathcal{L}(X)$ is the subset of $\mathcal{L}$ of formulae containing no predicate variables besides $X$, and $\mathcal{L}(X, x)$ the subset of $\mathcal{L}(X)$ of formulae containing no individual variables besides $x$.

For reasoning in the realm of sets we fix a universe $U$, so elements are elements of $U$ and sets are subsets of $U$. $\gamma$ is the first cardinal larger than the cardinality of $U$. We define $x^c$, the complement of $x$, by $x^c \equiv_{def} \{y \epsilon U | y \neq x\}$ for elements $x$. Operators $\Phi$ between sets are called monotonic if $\forall XY(X \subseteq Y \Rightarrow \Phi X \subseteq \Phi Y)$ and continuous if $X_1 \subseteq X_2 \subseteq X_3 \ldots$ implies $\Phi(\bigcup\{X_n|n \epsilon \omega\}) = \bigcup\{\Phi(X_n)|n \epsilon \omega\}$; $\Phi$ is a closure operator if it is monotonic and projective, i.e. $\Phi \circ \Phi = \Phi$. A condition $C$ is a subset of $\psi(U)$; the notations $X \epsilon C$ and $C(X)$ have the same meaning. $C$ is called $\bigcap$-closed if, for all conditions $C' \subseteq \psi(U)$, we have $\forall X \epsilon C'(C(X)) \Rightarrow C(\bigcap C')$.

The link between the set-theoretic and the logical perspective is provided by interpretation of expressions of the language $\mathcal{L}$ in a model with $U$ as its universe. The details of this are fairly standard and will not be given here. We only observe that defined predicates are interpreted by sets, predicate functions by conditions and predicate operators by operators between sets. For the rest of this paper, we assume some interpretation $I$ of $\mathcal{L}$ into $U$ to be given; it is clear that $I$ is completely determined by its effect on the nonlogical constants (function and predicate symbols) of $\mathcal{L}$. A set $X$ is called $\mathcal{L}$-definable or elementary (modulo $I$) if there is a formula $A$ with $X = I(\{x|A\})$; similar for conditions and operators. So the interpretation by $I$ of the nonlogical constants act as a parameter in this definability notion.

3 Implicit definition of predicates

3.1 What are inductive definitions?

In [1], Aczel defines the notion of inductive definition of a set $X$ by

$$X \text{ is the smallest set closed under a collection of rules } R;$$  \hspace{1cm} (6)
rules are of the form \( Y \rightarrow y \), where \( Y \) is a set and \( y \) an element; a set \( Z \) is closed under rule \( Y \rightarrow y \) iff \( Y \subseteq Z \Rightarrow y \in Z \) holds. (6) is a good definition, for
\[
X = \bigcap \{ Z | \forall r \in R(Z \text{ is closed under } r) \};
\]
this follows from the fact that the condition \( C_R \) with \( C_R(Z) = \forall r \in R(Z \text{ is closed under } r) \) is \( \cap \)-closed. This observation suggests an alternative and more abstract notion of inductive definition:
\[
X \text{ is the smallest set satisfying the } \cap \text{-closed condition } C;
\]
this is also a good definition, for
\[
X = \bigcap \{ Z | C(Z) \};
\]
(8)

We observe that (7) and (8) are examples of impredicative definitions: \( X \) is defined in terms of the collection of subsets of \( U \), to which \( X \) itself belongs. For the case of (7), there is a well-known method (named after its inventor S.C. Kleene) to obtain a more ‘constructive’ definition of \( X \) in stages: it is the construction of the least fixpoint \( \mu \Phi \) of the monotonic set operator \( \Phi = \Phi_R \), defined by \( \Phi_R(Z) = Z \cup \{ y \mid \text{there is a rule } Y \rightarrow y \text{ in } R \text{ with } Y \subseteq Z \} \). The definition reads \((\alpha, \beta \text{ range over the ordinals}): \)
\[
\mu \Phi = \bigcup \{ \Phi^\alpha | \alpha < \gamma \}, \text{ where } \Phi^\alpha = \text{def} \bigcup \{ \Phi^\beta | \beta < \alpha \};
\]
then \( \mu(\Phi_R) = \bigcap \{ Z | C(Z) \} \) (for a proof and more information we refer to [1], section 1.3, and [6], section 4.1). If \( \Phi \) is continuous, then we can stop the definition of \( \mu \Phi \) at \( \omega \), and \( X = \bigcup \{ \Phi^n | n \in \omega \} \); then we have a genuinely predicative definition of \( X \) which can be finitely approximated. If \( \Phi \) happens to be a closure operator then the construction becomes trivial, for \( \mu \Phi = \Phi^1 = \Phi(0) \), so only one step suffices.

In the next subsection, we try to answer the following question: is it possible to generalize the transition from \( C_R \) to \( \Phi_R \) to arbitrary \( \cap \)-closed conditions?

3.2 A syntactical trick

Let us have a closer look at the two inductive definitions (4) and (5) of \( N \) given in 1.3, but now in a slightly different formulation,
\[
N \text{ is the smallest predicate } X \text{ satisfying } C_N(X),
\]
where \( C_N(X) = \text{def} \ X(0) \land \forall x(X(x) \rightarrow X(x+1)) \);
\[
N = \text{def} \mu(\lambda X.\{ x | B_N(X, x) \}),
\]
where \( B_N(X, x) = \text{def} \ (x = 0 \lor \exists y(x = y+1 \land X(y))) \).

To find out how \( B_N \) can be obtained from \( C_N \), we rewrite \( C_N \) to
\[
-(-X(0) \lor \exists y(X(y) \land \neg X(y+1))),
\]
and now it is not hard to see that \( B_N \) can be obtained from \( C_N \) by replacing the positive occurrences of \( X \) by \( x^C(= \{ z | z \neq x \}) \) and dropping the leftmost negation sign:

\[-x^C(0) \lor \exists y(X(y) \land -x^C(y + 1)),\]

and this is equivalent to \( B_N \! \). We capture this transformation of \( C_N \) into \( \Lambda X.\{ x | B_N(X, x) \} \) in the following definition:

\[ \Delta X.C =_{\text{def}} \Lambda X.\{ x | -[X^+ := x^C]C \}. \]

Now, \( N, \Delta X.C.N = \Phi_N \) and \( \mu(\Delta X.C.N) = N \), but can this be generalized? Let us therefore consider the transition from the condition \( C \) to the predicate operator \( \Delta X.C \) in a set-theoretic setting.

### 3.3 From conditions to operations

We start with the following diagram summarizing the situation described in section 3.1:

\[
\begin{array}{c}
\text{φ}(U) \\
\downarrow \mu \\
\text{F}
\end{array}
\quad \downarrow \eta \quad
\begin{array}{c}
\text{C} \\
\phi \\
\text{F}
\end{array}
\]

with the definitions

\[
\text{F} =_{\text{def}} \{ \Phi : \text{φ}(U) \rightarrow \text{φ}(U) | \Phi \text{ monotonic} \}
\]

\[
\text{C} =_{\text{def}} \{ C \subseteq \text{φ}(U) | C \text{ is } \bigcap \text{ closed} \}
\]

\[\eta : \text{C} \rightarrow \text{φ}(U) \text{ is defined by } \eta(C) = \bigcap \{ X | \text{C}(X) \} \]

\[\mu : \text{F} \rightarrow \text{φ}(U) \text{ is defined by } \mu(\Phi) = \bigcap \{ X | \Phi(X) \subseteq X \} \]

\[\phi : \text{F} \rightarrow \text{C} \text{ is defined by } \phi(\Phi) = \{ X | \Phi(X) \subseteq X \} \]

One easily verifies that \( \phi \) is well-defined and that \( \phi(\Phi_R) = C_R \) (with \( \Phi_R, C_R \) as in 3.1). We have the following facts:

\[
\begin{align*}
\mu(\Phi) &= \bigcup \{ \Phi^\alpha | \alpha < \gamma \} \\
\eta \circ \phi &= \mu
\end{align*}
\]

What we are looking for is a right inverse \( \psi : \text{C} \rightarrow \text{F} \) of \( \phi \), for then the diagram commutes (since \( \mu \circ \psi = \eta \circ \phi \circ \psi = \eta \)). A possible definition is

\[\psi(C)(X) = \bigcap \{ Y \supseteq X | \text{C}(Y) \}.\]
Then \( \psi \) is well-defined and \( \phi \circ \psi = id_C \), for \( \phi(\psi(C)) = \{X | \psi(C)(X) \subseteq X \} = \{X | \bigcap \{Y : X \subseteq X \} \subseteq X \} = \{X | \bigcap \{Y : C(Y) \subseteq X \} \subseteq X \} = \{X \subseteq X \} = C \);
for the fourth equality we used that \( C(\bigcap \{Y : X \subseteq X \}) \) holds, as a consequence of the \( \bigcap \)-closedness of \( C \).

At first sight it seems that this \( \psi \) works; there are, however, two objections. Firstly, \( \psi(C) \) has no elementary definition which has no counterpart in \( L \). Secondly, \( \psi(C) \) appears to be a closure operator, since \( X \subseteq \bigcap \{Y : X \subseteq X \} \subseteq X \) = \( \psi(C)(X) \) and \( \psi(C)(\psi(C)(X)) = \psi(\psi(C)(X)) = \bigcap \{Z : \bigcap \subseteq \bigcap \} \subseteq \bigcap \) = \( \psi(C)(X) \), where the \( \bigcap \)-closedness of \( C \) is used for the third equality. As a consequence, \( \eta(C) = \psi(C)(\emptyset) \) so all induction steps are absorbed in the definition of \( \psi(C) \).

So this direct transition from \( C \) to \( F \) does not work, and we try it indirectly. The syntactical trick of 3.2 suggests to consider conditions \( C \) as the diagonal of binary relations \( D = D(X, Y), \) i.e. \( C(X) = D(X, X) \). We therefore define \( D \) as the collection of binary relations on sets which are antimono- tonic in their first argument and both monotonic and \( \bigcap \)-closed in their second argument, so

\[
D = \{ D \subseteq \forall X \forall Y \forall X' \forall Y' (D(X, Y) \land X' \subseteq X \land Y \subseteq Y' \rightarrow D(X', Y')) \land \forall X \forall C \subseteq \forall X \forall Y \forall X' \forall Y' (D(X, Y) \rightarrow D(X, \bigcap C)) \}.
\]

The diagonalizing mapping \( \delta : D \rightarrow C \) defined by \( \delta(D)(X) \leftrightarrow \bigcap D(X, X) \) maps \( D \) into \( C \), for \( D(X, X) \) is \( \bigcap \)-closed if \( \delta(D)(X) \); to see this, assume \( \forall X \forall X' \exists D(X, X) \) (for \( D(X, Y) \) is antimono- tonic in \( X \), hence \( D(X, \bigcap C) \) (for \( D(X, X) \) is \( \bigcap \)-closed in \( Y \).)

A mapping \( \alpha : F \rightarrow D \) which commutes with \( \phi \) and \( \delta \) is straightforward: put \( \alpha(\Phi)(X, Y) \leftrightarrow \Phi(X) \subseteq Y \). For the other way round, we find inspiration in the definition of \( \Delta X \) presented in the previous section, and we put

\[
\beta : D \rightarrow F \text{ is defined by } \beta(D)(X) = \{ x | \neg D(X, x^c) \}.
\]

This turns out to work, for we have

**Lemma.**

i) \( \alpha \circ \beta = id_F \);
ii) \( \beta \circ \alpha = id_D \);
iii) \( \phi \circ \beta = \delta \).

**Proof.**

i) \( \alpha(\beta(D))(X, Y) \leftrightarrow \beta(D)(X) \subseteq Y \leftrightarrow \{ x | \neg D(X, x^c) \} \subseteq Y \leftrightarrow \forall x(D(-X, x^c) \rightarrow x \in Y) \leftrightarrow \forall x \notin Y \forall x D(X, x^c) \leftrightarrow D(X, \bigcap \{ x^c \subseteq Y \}) \leftrightarrow D(X, Y) \).
ii) \( \beta(\alpha(\Phi))(X) = \{ x | \neg \alpha(\Phi)(X, x^c) \} = \{ x | \neg \Phi(X) \subseteq x^c \} \).
iii) \( \phi \circ \beta = \delta \circ \alpha \circ \beta = \delta \circ id_D = \delta \). \( \square \)

We observe that \( \alpha \) and \( \beta \) have elementary definitions. What is still missing is a mapping from \( C \) to \( D \) which commutes with \( \delta \) and has an elementary definition. We do not intend to give such a mapping here; presumably it does not exist in this set-theoretic context. However, we can get quite close to this by defining a subset \( S \) of \( L \)-formulae which can be seen as representations of
elementary conditions in $D$. This is done as follows: define $S$ and the mappings $\rho$ and $\sigma$ by

$$ S = \{ A \in L(X) | Y \text{ not in } A, I(\Delta XY. [X^+ := Y]A) \in D \} $$

$$ \rho : S \to C \text{ is defined by } \rho(A) = I(\Delta X.A) $$

$$ \sigma : S \to D \text{ is defined by } \sigma(A) = I(\Delta XY. [X^+ := Y]A) $$

It is evident that $\delta \circ \sigma = \rho$, so $\rho$ is well-defined and we have the following commuting diagram (where $*$ indicates the non-elementary mappings):

```
```

We summarize the upshot of this section in a theorem:

**Theorem.**

For every condition $C = \rho(A)$ in the range of $\rho$, an elementary monotonic operator $\Phi = \beta(\sigma(A)) = I(\Delta X.A)$ can be found which satisfies $\mu \Phi = \eta C$.

### 3.4 Syntactical conditions

In this section we return to the logical perspective and establish some syntactically defined subsets of $S$. Since $[X^+ := Y]A$ ($Y$ not in $A$) is always antimonic in $X$ and monotonic in $Y$, we have

$$ S = \{ A \in L(X) | I(\Delta XY. [X^+ := Y]A) \text{ is } \cap \text{-closed in its second argument} \} $$

**Lemma.** $S$ contains all formulae of the form

$$ \forall (A \to X(t)) \quad (X \text{ not negatively in } A), $$

i.e. all (finite or infinite) conjunctions of universally quantified formulae of the form $A \to X(t)$ where $X$ does not occur negatively in $A$.

**Proof.** This is a consequence of

$$ X(t) \in S $$

$$ A_1, A_2, \ldots \in S \Rightarrow \bigwedge_n A_n S $$

$$ A \in S, X \text{ not negatively in } B \Rightarrow (B \to A) \in S $$

$$ A \in S \Rightarrow \forall x A \in S. $$
The previous lemma describes a collection of formulae for which the first part of the method works: going from a formula \( A \) defining a \( \bigcap \)-closed condition \( I(\Delta X.A) \in C \) to an operator \( I(\Delta X.A) \) whose fixpoint equals the smallest set satisfying \( I(\Delta X.A) \). The second step is the restriction to \( L \)-definable fixpoints, which is accomplished if the operator \( I(\Delta X.A) \) is continuous, for then the fixpoint is reached in a countable number of steps (viz. \( \omega \)) and hence definable by an infinite disjunction in \( L \). So we define

\[
\delta X.A = \bigvee_n B_n, \text{ where } B_0 = \bot, B_{n+1} = (\Delta X.A)(B_n)
\]

\[
\text{Cts} = \{ A \in L(X,x)| I(\Delta X.\{x\}|A) \text{ is continuous} \}
\]

\[
\text{Adm} = \{ A \in S | [X^+ := x^e]A \in \text{Cts} \}
\]

and we have: if \( A \in \text{Adm} \), then

\[
\delta X.A \text{ is the smallest predicate satisfying } A,
\]

i.e. \( [X := \delta X.A]A \) and \( (A \rightarrow \delta X.A) \subseteq X \).  \( \tag{9} \)

**Lemma.** \( \text{Cts} \) contains all formulae of the form

\[
\bigvee \exists (A \land X(t_1) \land \ldots \land X(t_n)) \quad (X \text{ not in } A),
\]

i.e. all (finite or infinite) disjunctions of existentially quantified formulae \( A_1 \land \ldots \land A_n \), where \( A_i \) is of the form \( X(t) \) or does not contain \( X \).

**Proof.** This is a consequence of the following closure properties:

\[
X(t) \in \text{Cts}
\]

\[
A \in \text{Cts} \text{ if } X \text{ not in } A
\]

\[
A_1, A_2, \ldots \in \text{Cts} \Rightarrow \bigvee_n A_n \in \text{Cts}
\]

\[
A, B \in \text{Cts} \Rightarrow A \land B \in \text{Cts}
\]

\[
A \in \text{Cts} \Rightarrow \exists y A \in \text{Cts}
\]

Now we define the class of formulae for which the full method works:

**Definition.** A **Horn formula** in \( X \) is a formula of the form

\[
\bigwedge \forall (A \land X(s_1) \land \ldots \land X(s_m) \rightarrow X(t)) \quad (X \text{ not in } A),
\]

i.e. a (finite or infinite) conjunction of universally quantified formulae of the form \( A_1 \land \ldots \land A_m \rightarrow X(t) \) \( (m \geq 0) \), where \( A_i \) is of the form \( X(s) \) or does not contain \( X \). The collection of all Horn formulae in \( X \) is denoted \( \text{Horn}(X) \).

**Lemma.** \( \text{Horn}(X) \subseteq \text{Adm} \).

**Proof.** \( \text{Horn}(X) \subseteq S \) is evident; if \( A \in \text{Horn} \) then \( \neg[X^+ := x^e]A \in \text{Cts} \) by previous lemma. So \( A \in \text{Adm} \).

**Theorem.** If \( A \) is a Horn formula in \( X \), then \( \delta X.A \) is the predicate inductively defined by \( A \), hence satisfies (9).
4 Implicit definitions of functions

For implicitly defined functions the story is somewhat more complicated. We need a logical language with definedness predicate and with descriptions (viz. the description of MPL\textsubscript{w} in section 1.2). The method now proceeds by first going from functions \( f \) (which, for simplicity, we assume to be unary) to functional predicates \( F \) with \( f(x) = y \leftrightarrow F(x, y) \), then making the definition of \( F \) explicit by \( \delta F \), followed by going back from the predicate \( \delta F \) to the function \( \delta f.A \) using descriptions: \( \delta f.A = \lambda x,y.\langle \delta f.A \rangle \). There is complication, however: the condition needed for the expected property

\[
\delta f. A \text{ is the smallest partial function satisfying } A,
\]

i.e. \([f := \delta f.A] A \) and \((A \rightarrow \delta f.A \subseteq f)\) to hold, has two parts. The first part is as for predicates and can be approximated with a syntactical criterion \textbf{Horn}(\(f\)), requiring \(A\) to be a conjunction of universally quantified formulae of the form \(A_1 \land \ldots \land A_m \land s \downarrow \rightarrow (f(s) \leftrightarrow t)\)

where \(A_i\) is atomic or does not contain \(f\) and with \(f\) not in \(s\). Here \(\leftrightarrow\) is a directed weakening of equality:

\[
(a \leftrightarrow b) \text{ is defined by } \forall x(x = b \rightarrow x = a) \quad \text{(}b \downarrow \rightarrow a = b).\]

The intended meaning of \(a \leftrightarrow b\) can be paraphrased as: \(a\) is defined as \(b\), or: \(a\) rewrites/reduces to \(b\). (In COLD-K and VVSL, \(=\) is written for \(\leftrightarrow\).) The second part of the condition, however, has to do with the functionality of the predicate used to define \(\delta f.A\) and this functionality requirement eludes syntactical characterisation. See [11] for more details.

We give two examples. For the first, we assume that \(m,n\) range over the natural numbers. Now

\[
\forall n(0 + n \leftrightarrow n) \land \forall m n(S(m) + n \leftrightarrow S(m + n)) \quad (10)
\]

inductively defines + (addition) in terms of 0 and \(S\) (although (10) does not belong to \textbf{Horn}(\(+\)) properly, but it can easily be brought into that form). The functionality requirement involved here follows from the properties of \(0\) and \(S\).

The second example is related to \(\mathcal{F}(C)\), defined in (2). Then the cardinality function \(|.|: \mathcal{F}(C) \rightarrow \mathbb{N}\), can be defined implicitly by

\[
\emptyset \leftrightarrow 0 \land \forall c \epsilon C([c] \leftrightarrow 1) \land \forall X Y ([X \cup Y] \leftrightarrow |X| + |Y| - |X \cap Y|) \quad (11)
\]

and this yields a cardinality function with all the desired properties.

5 Final remarks

5.1 Closure properties of \textbf{Adm}

\textbf{Adm} is not closed under \(\equiv\), so formulae equivalent to an element of \textbf{Horn} are not always in \textbf{Adm}; however, if \(A \equiv B\) are both element of \textbf{Adm} then
\[ \delta X.A \equiv \delta X.B. \] On the other hand, \( \text{Adm} \) is closed under a stronger equivalence \( \equiv_X \), defined by

\[ A \equiv_X B \quad =_{\text{def}} \quad [X^+ := Y]A \equiv [X^+ := Y]B \quad (Y \text{ not in } A,B). \]

### 5.2 Conclusion

We described and provided the foundations for a method which turns a large class of implicit descriptions of predicates and functions into explicit definitions in a logical language with infinite conjunctions and disjunctions. This method is applied in the definition of the logical semantics of COLD-K ([3]) and VVSL ([9]); as a consequence, these languages support the use of fairly easy readable implicit descriptions, appropriate for abstract and high-level specifications and designs. This was illustrated with the example of finite sets and the cardinality function (see (2) and (11)).

### 5.3 Acknowledgement

Karst Koymans (University of Utrecht) is gratefully acknowledged for pointing out clearly the relevance of the set-theoretic perspective in these matters.

### References


