The generalised completeness of Horn predicate-logic as a programming language*

By H. ANDRÉKA and I. NÉMETI

To the memory of Professor László Kalmár

Here we prove the “generalised” completeness of “Prolog-like” languages [1], [2] or “Horn-predicate logic as a programming language” [3], [4], [5], [6].

More precisely we prove the following. Let $Fr$ be an arbitrary Herbrand-universe (in other words, $Fr$ is a word algebra of an arbitrary finite type generated by the constant symbols). For any $f: Fr^n \rightarrow Fr$ Turing-computable partial function over $Fr$, there is a finite set $C_f$ of Horn clauses over $Fr$ (that is there are no other function or constant symbols in $C_f$ but only those which occur in $Fr$) and a relation symbol $F_f$ such that $C_f$ defines $f$ over $Fr$, more precisely:

$$(\forall \bar{a}, \beta \in Fr)[f(\bar{a}) = \beta \iff C_f \models F_f(\bar{a}, \beta)]$$

where $\bar{a}$ is a vector of elements from $Fr$.

This means, that if we are given an arbitrary Herbrand-universe $Fr$ and an arbitrary computable task over $Fr$, then we can write a Prolog program which solves this task and which does not contain other function or constant symbols but only those which occur in $Fr$. This is somehow a statement about the adequateness of Horn logic as a programming language: Any computable problem can be formulated in Horn logic without using auxiliary function symbols. That is without “coding” the data to be processed.

A special case of this theorem was proved by Robert Hill (unpublished, personal communication). He proved the above statement for the case when $Fr$ is the set of natural numbers together with the successor function and constant 0. The proof stated here is a generalisation of his one. In generalising any proof from the natural numbers to arbitrary Herbrand universes $Fr$ the difficulty originates from the unfortunate fact, that — as far as we know — there is very little work done on the “nice” characterisation of the computable functions over $Fr$.

Another related result has recently been proved by Tärnlund, c.f. “Sten Ake Tärnlund: Logic Information Processing”, University of Stockholm, report

*Part of the research for this paper was done during the author’s stay in the Department of Artificial Intelligence, University of Edinburgh.
H. Andréka and I. Németi

TRITA—IBADB—1034, 1975—II—24. He proves that if we are given an arbitrary Herbrand-universe $Fr$ together with a computable function $f$ over it, then there exists a set of binary Horn-clauses $C_f$ defining $f$. However in Tärnlund's paper $C_f$ is defined over a Herbrand-universe which is definitely larger than $Fr$. (In defining $C_f$ he makes extensive use of auxiliary function symbols.)

The main result proved in this paper is that $C_f$ can be defined over $Fr$ itself; in other words, that we can dispense with the auxiliary function symbols.

Remark. We believe that an alternative (perhaps more natural) proof can be given by starting from Emden's work [7] and investigating the generalisation of Kleene's recursion equations to arbitrary word algebras. To this end first it should be proved that any Turing-computable partial function $f$ over an arbitrary word-algebra $Fr$ can be defined by such a finite system of Emden's modified recursion equations (see [7]), in which system all the constant functions belong to $Fr$.

**Theorem.** Let $Fr$ be an arbitrary Herbrand-universe (that is a word-algebra of arbitrary finite type generated by the empty set, in other words: generated by the constant symbols of the type).

Now, for any finitary Turing-computable partial function $f$ over $Fr$ ($f: Fr^n \rightarrow Fr$) there is a finite set $C_f$ of Horn clauses over $Fr$ (that is all the function symbols occurring in $C_f$ also occur in $Fr$), and a relation symbol $F_f$ such that

$$(\forall \bar{a}, \beta \in Fr)[f(\bar{a}) = \beta \iff C_f \models F_f(i,f,\bar{a})]$$

where $\bar{a}$ is a vector of elements of $Fr$.

Moreover, $C_f$ can be effectively computed from the Turing-definition of $f$.

**Proof.** Let $\omega$ denote the set of natural numbers. The idea of the proof is the following:

First we define a one-one function $g$ from $Fr$ onto $\omega$, such that $g$ as well as $g^{-1}$ are Turing-computable. Now if $f: Fr^n \rightarrow Fr$ is Turing-computable, then $g = \circ \circ g^{-1}$ is a Turing-computable function on $\omega$, and $f = g^{-1} \circ g \circ g$. But every Turing-computable function on $\omega$ is recursive. Thus every Turing-computable function $f$ over $Fr$ is the image by $g$ of some recursive function $g$ over $\omega$ ($f = g^{-1} \circ g \circ g$). By this it is enough to prove for any recursive function $g$ over $\omega$, that the function $g^{-1} \circ g \circ g$ is Horn-definable over $Fr$ (see figure).

Let the type $t$ be denoted as: $t = \{(f^i_j; i): i \leq k, j \leq m_i\}$. In other words: there are numbers $k$ and $m_i$ (for every $i \leq k$) such that $f^i_j$ is the $j$-th $i$-ary function symbol for $k \leq i, j \leq m_i$. Note that $\{f^i_j; j \leq m_i\}$ is the set of constant symbols.

Now we define the function $g$. To this end we first define the auxiliary functions $F$ and $Sz$ by a simultaneous recursion.

* Thus Tärnlund's result is different from Hill's one in two respects:

1. Tärnlund says more than Hill by allowing arbitrary Herbrand-universes and using only binary Horn-clauses.

2. On the other hand Tärnlund says less than Hill, since he says nothing about the number of auxiliary functions symbols.
The generalised completeness of Horn predicate-logic as a programming language

The intuition behind the following definitions of $F$ and $Sz$ is explained later in the proof of the first lemma.

The only important property of $F$ is that $F$ enumerates the word algebra $Fr$. Any other recursively defined function with this property could be substituted for $F$ without changing the rest of the proof. The function $Sz$ is only an auxiliary function in the definition of $F$. That is, we use $Sz$ only to define $F$.

We define $F$ by a definition scheme which can be translated into a definition for any given type, that is for any fixed numbers $k$ and $m$. In this scheme the text: "for $i \leq k, j \leq m, 0 \leq p \leq i$..." is written in a metalanguage and can be translated by copying "..." as many times as $i$, $j$’s and $p$’s are possible.

$$
F(0) = f_0^0 \\
F(n + 1) = 
\begin{cases}
  f_j^i(F(n_1), ..., F(n_p + 1), F(0), ..., F(0)) & \text{if } F(n) = f_j^i(F(n_1), ..., F(n_i)) \text{ and } (\forall p < z \leq i) Sz(n_z + 1) \equiv Sz(n) \\
  f_j^{i+1}(F(0), ..., F(0)) & \text{if } F(n) = f_j^{i+1}(F(0), ..., F(0)) \text{ and } (\forall 0 < z \leq i) Sz(n_z + 1) \equiv Sz(n) \\
  f_0^0 & \text{if } F(n) = f_0^0(F(0), ..., F(0)) \text{ and } (\forall 0 < z \leq k) Sz(n_z + 1) \equiv Sz(n)
\end{cases}
$$

$$
Sz(0) = 0, \\
Sz(n + 1) = 
\begin{cases}
  Sz(n) + 1, & \text{if } F(n) = f_0^k(F(n_1), ..., F(n_k)) \text{ and } (\forall 1 \leq z < k) Sz(n_z + 1) \equiv Sz(n), \\
  Sz(n), & \text{otherwise}.
\end{cases}
$$

It is easy to see that the above simultaneously recursive definition is correct, that is it really defines the functions $F$ and $Sz$.

In the following definitions we use the recursion theoretic $\mu$-operator. Remember that $\mu x(R(x))$ is the smallest number $x$ for which $R(x)$ is true.

$$
E(n) = \begin{cases}
  \text{True, if } (\forall j < n) F(j) \neq F(n), \\
  \text{False, if otherwise}
\end{cases}
$$

$$
S(\tau) = F(\mu n(\mu k(F(k) = \tau) < n \& E(n))), \\
\xi(0) = F(0), \\
\xi(n + 1) = S(\xi(n)), \\
\rho(\tau) = \mu n(\xi(n) = \tau).
$$
Lemma. a) \( \varrho: Fr \rightarrow \omega \) is one-one and onto, 
b) \( \varrho \) and \( \varrho^{-1} \) are Turing computable.

Proof. ad a) Note, that any total function \( f \) with domain \( \omega \) can be considered as a "listing" or an enumeration of the range of \( f \). Now, we define a system of subsets \( H_i \) (\( i \in \omega \)).

\[
H_{-1} = \{ f^0, \ldots, f^m \}, \\
H_{i+1} = \{ f_j(\tau_1, \ldots, \tau_k): \tau_1, \ldots, \tau_k \in H_i, i \equiv k, j \equiv m_i \}.
\]

For each \( i \in \omega \), on the set \( H_i \) a linear ordering can be defined in a natural way:

For \( H_{-1} \): \( f^0_i < f^0_j \) iff \( i < j \). To define the ordering on \( H_{i+1} \), suppose, that the ordering on \( H_i \) has been defined.

Now for any two elements of \( H_{i+1} \):

\[
f_j^i(\tau_1, \ldots, \tau_k) < f_j^i(\tau_1', \ldots, \tau_k') \iff \langle i, j, \tau_1, \ldots, \tau_k \rangle < \langle i', j', \tau_1', \ldots, \tau_k' \rangle
\]

according to the lexicographic ordering obtained from the ordering on natural numbers and the ordering \( \prec \) defined on \( H_i \).

It is easy to check, by the definition of \( F \), that the function \( F \) first enumerates \( H_0 \) in accordance with the above defined ordering on \( H_0 \), then enumerates similarly \( H_1 \), then \( H_2 \) etc. Since \( \bigcup_{i=1}^{\omega} H_i = Fr \), the function \( F \) enumerates the whole \( Fr \).

However, unfortunately, \( F \) might enumerate an element of \( Fr \) more than once, in other words, the function \( F \) is not one-one. To deal with this, the relation \( E \) marks those places in \( \omega \) where an element occurs (is listed) first. The function \( \xi \) picks out only those occurrences (of elements of \( Fr \)) which are marked by \( E \). Thus \( \xi \) is already one-one, while since \( F \) is onto, \( \xi \) is also onto.

ad b) From the fact that \( \xi \) is one-one it follows that \( \varrho = \xi^{-1} \), and from their definition it is easy to see that both \( \varrho \) and \( \xi \) are Turing-computable. (For, from their definition it is easy to construct a computer program which computes \( \varrho \) and \( \xi \).) And by this the lemma is proved.

Lemma. To every partial recursive function \( g \) over \( \omega \), \( g: \omega^n \rightarrow \omega \), the function \( f = \varrho^{-1} \circ g \circ \varrho \) is Horn-definable over \( Fr \), that is there is a set of Horn-clauses \( C_f \) and a relation symbol \( F_f \) such that

\[
(\forall \bar{a}, \beta \in Fr)[g^{-1} \circ g \circ g \ (\bar{a}) = \beta \ \iff \ C_f \models F_f(\bar{a}, \beta)].
\]

Proof. By the definition of recursive functions, it suffices to prove the above statement for the:

- zero function \( Z(x) \equiv 0 \);
- the successor function \( S(x) \equiv x + 1 \);
- the projection functions \( \pi^n_m(x_1, \ldots, x_n) \equiv x_m \),
and to prove that if the above statement holds for the functions $h, g, g_1, \ldots, g_n$ then it also holds for the functions obtained from these by

- **substitution** \[ f(x) = h(g_1(x), \ldots, g_n(x)) \]
- **recursion** \[ f(x, 0) = g(x) \] 
  \[ f(x, n+1) = h(x, n, f(x, n)) \]

- the $\mu$-operator \[ f(x) = \mu y (g(x, y) = 0). \]

Note, that $\mu$ has already been defined in the definition of the function $g$. In writing Horn-clauses we use the notation of Kowalski [3].

a) **the zero function:**

$g^{-1} \circ Z \circ \varnothing$ is Horn definable:

\[ C_z = \{ F_z(x, f_0^0) \rightarrow \} \]

It is easy to see that $C_z$ defines exactly the function $g^{-1} \circ Z \circ \varnothing$. Here we give the detailed proof of this statement, but we shall omit the proofs of the following statements about the successor function, etc. because they are mechanical analogues of the present one.

Now we prove that $C_z \models F_z(\tau, \sigma)$ iff $\sigma = f_0^0$.

1. For all $\tau \in Fr$, we immediately have $C_z \models F_z(\tau, f_0^0)$.
2. To prove the implication in the other direction.

Let $\sigma \neq f_0^0$, and $\tau, \sigma \in Fr$.

In this case $C_z \nvdash F_z(\tau, \sigma)$, because we can construct a model of $C_z$ in which $F_z(\tau, \sigma)$ fails. Let on the Herbrand-universe $Fr$ the interpretation of the relation symbol $F_z$ be the relation $R = \{ (\tau, f_0^0) : \tau \in Fr \}$. In the model obtained this way $C_z$ is valid while $F_z(\tau, \sigma)$ is clearly false. Thus, $C_z \nvdash F_z(\tau, \sigma)$.

b) **the successor function:**

$g^{-1} \circ S \circ \varnothing$ is Horn definable:

This is the only more laborious step: Here we need an explicit and constructive description of the function $g$. We shall, not do anything but translate the definition of $g$ into Horn-clausal form. To this end however we first have to "code" the natural numbers by elements of $Fr$. For any number $n \in \omega$, the symbol $\hat{n}$ stands for the code of $n$ in $Fr$. We define the code recursively:

\[ \hat{0} = f_0^0, \text{ and } \hat{n+1} = f_0^1(\hat{n}). \]

Remark. If $m_i = -1$ then let $i$ be the smallest number such that $m_i \equiv 0$.

Now $\hat{n+1} = f_0^1(\hat{n}, f_0^0, \ldots, f_0^0)$.

\[ C_s = \{ \leq (x, y) \rightarrow, \]
\[ \equiv (x, y) \rightarrow \equiv (f_0^1 x, y), \]
\[ < (x, y) \rightarrow \equiv (f_0^1 x, y) \} \cup \]

\[ \{ F(f_0^0, f_0^0) \rightarrow \} \cup \]
\[
\{F(f_0^i y, f_j^i(x_1, \ldots, x_{p-1}, v, f_0^i, \ldots, f_0^q)) - F(y, f_j^i(x_1, \ldots, x_i)) \land \bigwedge_{z=1}^i F(y_z, x_z) \land \\
\bigwedge_{z=p+1}^i \equiv (w, w_z) \land \\
: i \equiv k, j \equiv m_i, 0 < p \equiv i\} \cup \\
\{F(f_0^i y, f_{j+1}^i(f_0^i, \ldots, f_0^q)) - F(y, f_j^i(x_1, \ldots, x_i)) \land \bigwedge_{z=1}^i F(y_z, x_z) \land \\
\bigwedge_{z=1}^i Sz(f_0^i y_z, w_z) \land Sz(y, w) \land \bigwedge_{z=1}^i \equiv (w, w_z) \\
: i \equiv k, j < m_i\} \cup \\
\{F(f_0^i y, f_0^k) \rightarrow \\
F(y, f_m^k(x_1, \ldots, x_i)) \land \bigwedge_{z=1}^i F(y_z, x_z) \land \\
\bigwedge_{z=1}^i Sz(f_0^i y_z, w_z) \land Sz(y, w) \land \bigwedge_{z=1}^i \equiv (w, w_z) \} \cup \\
\{Sz(f_0^i y, f_0^k w) \rightarrow F(y, f_m^k(x_1, \ldots, x_i)) \land \bigwedge_{z=1}^i F(y_z, x_z) \land \\
\bigwedge_{z=1}^i Sz(f_0^i y_z, w_z) \land Sz(y, w) \land \bigwedge_{z=1}^i \equiv (w, w_z) \} \cup \\
\{x \neq f_0^i f_0^j \rightarrow : i \neq j\} \cup \\
\neq (f_j^i(x_1, \ldots, x_i), f_j^i(y_1, \ldots, y_i)) \rightarrow : i, j \neq i', j'\} \cup \\
\neq (f_j^i(x_1, \ldots, x_i), f_j^i(y_1, \ldots, y_i)) \rightarrow \neq (x_p, y_p): i \equiv k, j \equiv m_i, 0 < p \equiv i\} \cup \\
\{NE(y, f_0^i) \rightarrow \\
NE(y, f_0^w) \rightarrow NE(y, w) \land \neq (x, v) \land F(y, x) \land F(w, v), \\
E(y) \rightarrow NE(y, y)\} \cup \\
\{\neg E(y) \rightarrow \neq (z, y) \land F(z, w) \land F(y, w), \\
N(x, f_0^i) \rightarrow, \\
N(x, f_0^i y) \rightarrow N(x, y) \land \neq (w, x) \land F(y, w), \\
M(x, y) \rightarrow N(x, y) \land F(y, x)\}.
\]
The generalised completeness of Horn predicate-logic as a programming language

\[ N_1(y, f_0^0) \leftarrow \]
\[ N_1(y, f_0^0 z) \leftarrow N_1(y, z) \land \equiv (z, y), \]
\[ M_1(y, f_0^0 z) \leftarrow N_1(y, z) \land \lnot E(z), \]
\[ S(x, w) \leftarrow M(x, z) \land M_1(z, y) \land F(y, w) \}.
\]

c) the projection function:
\( q^{-1} \circ I_0^0 \circ q \) is Horn definable:
\[ C_f \overset{d}{=} \{ F_1(x_1, \ldots, x_n, x_m) \leftarrow \}.
\]

Now for the following steps suppose that \( C_h, C_g, C_{g_1}, \ldots, C_{g_n} \) define
\( q^{-1} \circ h \circ q, q^{-1} \circ g \circ q, \ldots \) respectively.

d) substitution:
\( f = q^{-1} \circ Su(h, g_1, \ldots, g_n) \circ q \) is Horn definable; where \( Su(h, g_1, \ldots, g_n) \) is the function defined by substitution from \( h, g_1, \ldots, g_n \).
\[ C_f \overset{d}{=} \{ F_f(x, y) \leftarrow F_h(y_1, \ldots, y_n, y) \land F_{g_1}(x, y_1) \land \ldots \land F_{g_n}(x, y_n) \} \cup \\
C_h \cup C_g \cup \ldots \cup C_{g_n}.
\]
To prove that \( C_f \) really defines \( f \), note that
\[ q^{-1} \circ Su(h, g_1, \ldots, g_n) \circ q(x) = q^{-1}(h(g_1(q(x))), \ldots, g_n(q(x))) = \\
q^{-1} \circ h \circ q(q^{-1} \circ g \circ q(x), \ldots, q^{-1} \circ g_n \circ q(x)).
\]
(Similar remarks will be omitted in the following.)

e) recursion:
\( f = q^{-1} \circ R(g, h) \circ q \) is Horn definable, where \( R(g, h) \) is the function defined by recursion from \( g \) and \( h \).
\[ C_f \overset{d}{=} \{ F_f(x, f_0^0, y) \leftarrow F_g(x, y), \\
F_f(x, w, y) \leftarrow F_s(z, w) \land F_f(x, z, y_1) \land F_h(x, z, y_1, y) \} \cup \\
C_g \cup C_s \cup C_h.
\]
Remember that \( C_s \) defines the function \( q^{-1} \circ S \circ q \), where \( S \) is the successor function on \( \omega \).

f) the \( \mu \)-operator:
\( f = q^{-1} \circ Myg \circ q \) is Horn definable, where \( Myg \) is the function defined by the \( \mu \)-operator from \( g \).
\[ C_f \overset{d}{=} \{ N(f_0^0) \leftarrow, N(w) \leftarrow S(z, w) \land N(z) \land F_g(x, z, y) \land S(y_1, y), \\
F_f(x, y) \leftarrow N(y) \land F_g(x, y, f_0^0) \} \cup C_g \cup C_s.
\]
Abstract

The adequacy of Horn clauses as a programming language is demonstrated by proving that any computable problem can be formulated in Horn logic without using auxiliary function symbols.

References


(Received Jan. 13, 1978)