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 $f: Fr^n \to Fr$ 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:

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

where $\bar{\alpha}$ 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A special case of this theorem was proved by Robert Hill (unpublished, personal communication).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.

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: $Fr^n \to Fr$) there is a finite set C_f of Horn clauses over Fr (that is all the function symbols occuring in C_f also occur in Fr), and a relation symbol F_f such that

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

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

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

Proof. Let ω denote the set of natural numbers. The idea of the proof is the following:

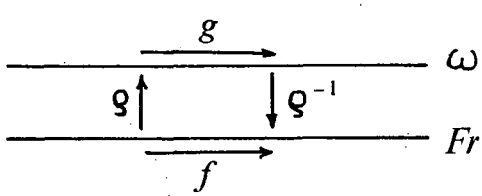

Fig. 1

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

Let the type t be denoted as: $t \overset{d}{=} \{\langle f_j^i, i\rangle : i \leqq k, j \leqq m_i\}$. In other words: there are numbers k and m_i (for every $i \leqq k$) such that f_j^i is the j-th i-ary function symbol for $k \geqq i$, $j \leqq m_i$. Note that $\{f_j^0 : j \leqq m_0\}$ is the set of constant symbols.

Now we define the function ϱ. 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 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_i. In this scheme the text: "for $i \leqq k$, $j \leqq m_i$, $0 < p \leqq i$: $f \ldots$" is written in a metalanguage and can be translated by copying "$f \ldots$" as many times as i's, j's and p's are possible.

$$F(0) \stackrel{\mathrm{d}}{=} f_0^0$$

$$F(n+1) \stackrel{\mathrm{d}}{=}$$

$$\begin{cases} \text{for } i \leqq k, j \leqq m_i \text{ and } 0 < p \leqq i: & \\ f_j^i(F(n_1), \ldots, F(n_p+1), F(0), \ldots, F(0)) & \text{if } F(n) = f_j^i(F(n_1), \ldots, F(n_i)) \text{ and } (\forall p < z \leqq i)\, Sz(n_z+1) \geqq Sz(n) \text{ and } Sz(n_p+1) < Sz(n) \\ \text{for } i \leqq k, j < m_k: & \\ f_{j+1}^i(F(0), \ldots, F(0)) & \text{if } F(n) = f_j^i(F(n_1), \ldots, F(n_i)) \text{ and } (\forall 0 < z \leqq i)\, Sz(n_z+1) \geqq Sz(n) \\ \text{for } i < k: & \\ f_0^{i+1}(F(0), \ldots, F(0)) & \text{if } F(n) = f_{m_i}^i(F(n_1), \ldots, F(n_i)) \text{ and } (\forall 0 < z \leqq i)\, Sz(n_z+1) \geqq Sz(n) \\ f_0^0 & \text{if } F(n) = f_{m_k}^k(F(n_1), \ldots, F(n_k)) \text{ and } (\forall 0 < z \leqq k)\, Sz(n_z+1) \geqq Sz(n) \end{cases} \stackrel{\mathrm{d}}{=}$$

$$Sz(0) \stackrel{\mathrm{d}}{=} 0,$$

$$Sz(n+1) \stackrel{\mathrm{d}}{=} \begin{cases} Sz(n)+1, & \text{if } F(n) = f_{m_k}^k(F(n_1), \ldots, F(n_k)) \text{ and } (\forall 1 \leqq z < k)\, Sz(n_z+1) \geqq Sz(n), \\ Sz(n), & \text{otherwise.} \end{cases}$$

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 μ-operator. Remember that $\mu x(R(x))$ is the smallest number x for which $R(x)$ is true.

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

$$S(\tau) \stackrel{\mathrm{d}}{=} F(\mu n(\mu k(F(k) = \tau) < n \;\&\; E(n))),$$

$$\xi(0) \stackrel{\mathrm{d}}{=} F(0),$$

$$\xi(n+1) \stackrel{\mathrm{d}}{=} S(\xi(n)),$$

$$\varrho(\tau) \stackrel{\mathrm{d}}{=} \mu n(\xi(n) = \tau).$$

Lemma. *a)* ϱ: $Fr \to \omega$ is one-one and onto,
b) ϱ and ϱ^{-1} are Turing computable.

Proof. ad *a)* Note, that any total function f with domain ω can be considered as a "listing" or an enumeration of the range of f.

Now, we define a system of subsets H_i $(i \in \omega)$.

$$H_{-1} \stackrel{d}{=} \{f_0^0, \ldots, f_{m_0}^0\},$$

$$H_{l+1} \stackrel{d}{=} \{f_j^i(\tau_1, \ldots, \tau_i) : \tau_1, \ldots, \tau_i \in H_l,\ i \leqq k,\ j \leqq m_i\}.$$

For each $i \in \omega$, on the set H_i a linear ordering can be defined in a natural way: For H_{-1}: $f_i^0 < f_j^0$ iff $i<j$. To define the ordering on H_{l+1}, suppose, that the ordering on H_l has been defined.

Now for any two elements of H_{l+1}:

$$f_j^i(\tau_1, \ldots, \tau_i) < f_{j'}^{i'}(\tau_1', \ldots, \tau_{i'}') \quad \text{iff} \quad \langle i, j, \tau_1, \ldots, \tau_i \rangle < \langle i', j', \tau_1', \ldots, \tau_{i'}' \rangle$$

according to the lexicographic ordering obtained from the ordering on natural numbers and the ordering $<$ defined on H_l.

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 $\bigcup_{i=1}^{\infty} H_i = Fr$, 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 ω where an element occurs (is listed) first. The function ξ picks out only those occurrences (of elements of Fr) which are marked by E. Thus ξ is already one-one, while since F is onto, ξ is also onto.

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

Lemma. To every partial recursive function g over ω g: $\omega^n \to \omega$, 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

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

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

zero function $Z(x) \stackrel{d}{=} 0$;

the successor function $S(x) \stackrel{d}{=} x+1$;

the projection functions $I_m^n(x_1, \ldots, x_n) \stackrel{d}{=} x_m$,

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 $\quad f(x) \overset{d}{=} h(g_1(x), \ldots, g_n(x))$

recursion $\quad f(x, 0) \overset{d}{=} g(x)$

$$f(x, n+1) \overset{d}{=} h(x, n, f(x, n))$$

the μ-operator $\quad f(x) \overset{d}{=} \mu y(g(x, y) = 0)$.

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

a) *the zero function:*

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

$$C_z \overset{d}{=} \{F_z(x, f_0^0) \leftarrow\}$$

It is easy to see that C_z defines exactly the function $\varrho^{-1} \circ Z \circ \varrho$. 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 $C_z \not\models F_z(\tau, \sigma)$, 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 $R \overset{d}{=} \{\langle \tau, f_0^0 \rangle : \tau \in Fr\}$. In the model obtained this way C_z is valid while $F_z(\tau, \sigma)$ is clearly false. Thus, $C_z \not\models F_z(\tau, \sigma)$.

b) *the successor function:*

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

This is the only more laborious step: Here we need an explicit and constructive description of the function ϱ. We shall, not do anything but translate the definition of ϱ 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:

$$\hat{0} \overset{d}{=} f_0^0, \text{ and } \widehat{n+1} \overset{d}{=} f_0^1(\hat{n}).$$

Remark. If $m_1 = -1$ then let i be the smallest number such that $m_i \geqq 0$. Now $\widehat{n+1} \overset{d}{=} f_0^j(\hat{n}, f_0^0, \ldots, f_0^0)$.

$$C_s \overset{d}{=} \{\leqq(x, x) \leftarrow,$$
$$\leqq(x, y) \leftarrow \leqq(f_0^1 x, y),$$
$$<(x, y) \leftarrow \leqq(f_0^1 x, y)\} \cup$$

$\{F(f_0^0, f_0^0) \leftarrow\} \cup$

$$\{F(f_0^1 y, f_j^i(x_1, \ldots, x_{p-1}, v, f_0^0, \ldots, f_0^0)) \leftarrow \quad F(y, f_j^i(x_1, \ldots, x_i)) \wedge \bigwedge_{z=1}^{i} F(y_z, x_z) \wedge$$

$$\bigwedge_{z=1}^{i} Sz(f_0^1 y_z, w_z) \wedge Sz(y, w) \wedge \bigwedge_{z=p+1}^{i} \leqq (w, w_z) \wedge$$

$$\prec (w, w_p) \wedge F(f_0^1 y, v)$$

$$: i \leqq k, j \leqq m_i, 0 \prec p \leqq i\} \cup$$

$$\{F(f_0^1 y, f_{j+1}^i(f_0^0, \ldots, f_0^0)) \leftarrow \quad F(y, f_j^i(x_1, \ldots, x_i)) \wedge \bigwedge_{z=1}^{i} F(y_z, x_z) \wedge$$

$$\bigwedge_{z=1}^{i} Sz(f_0^1 y_z, w_z) \wedge Sz(y, w) \wedge \bigwedge_{z=1}^{i} \leqq (w, w_z)$$

$$: i \leqq k, j \prec m_i\} \cup$$

$$\{F(f_0^1 y, f_0^{i+1}(f_0^0, \ldots, f_0^0)) \leftarrow \quad F(y, f_{m_i}^i(x_1, \ldots, x_i)) \wedge \bigwedge_{z=1}^{i} F(y_z, x_z) \wedge$$

$$\bigwedge_{z=1}^{i} Sz(f_0^1 y_z, w_z) \wedge Sz(y, w) \wedge \bigwedge_{z=1}^{i} \leqq (w, w_z)$$

$$: i \prec k\} \cup$$

$$\{F(f_0^1 y, f_0^0) \leftarrow \quad F(y, f_{m_k}^k(x_1, \ldots, x_k)) \wedge \bigwedge_{z=1}^{k} F(y_z, x_z) \wedge$$

$$\bigwedge_{z=1}^{k} Sz(f_0^1 y_z, w_z) \wedge Sz(y, w) \wedge \bigwedge_{z=1}^{k} \leqq (w, w_z)\}$$

$$\cup$$

$$\{Sz(f_0^0, f_0^0) \leftarrow,$$

$$Sz(f_0^1 y, f_0^1 w) \leftarrow F(y, f_{m_k}^k(x_1, \ldots, x_k)) \wedge \bigwedge_{z=1}^{k} F(y_z, x_z) \wedge \bigwedge_{z=1}^{k} Sz(f_0^1 y_z, w_z) \wedge Sz(y, w) \wedge$$

$$\bigwedge_{z=1}^{k} \prec (w, w_z)\}$$

$$\cup$$

$$\{\neq (f_i^0, f_j^0) \leftarrow \quad : i \neq j\} \cup$$

$$\neq (f_j^i(x_1, \ldots, x_i), f_{j'}^{i'}(y_1, \ldots, y_{i'})) \leftarrow \quad : \langle i, j \neq \langle i', j' \rangle\} \cup$$

$$\neq (f_j^i(x_1, \ldots, x_i), f_j^i(y_1, \ldots, y_i)) \leftarrow \neq (x_p, y_p) : i \leqq k, j \leqq m_i, 0 \prec p \leqq i\} \cup$$

$$\{NE(y, f_0^0) \leftarrow,$$

$$NE(y, f_0^1 w) \leftarrow NE(y, w) \wedge \neq (x, v) \wedge F(y, x) \wedge F(w, v),$$

$$E(y) \leftarrow NE(y, y)\} \cup$$

$$\{\neg E(y) \leftarrow \prec (z, y) \wedge F(z, w) \wedge F(y, w),$$

$$N(x, f_0^0) \leftarrow,$$

$$N(x, f_0^1 y) \leftarrow N(x, y) \wedge \neq (w, x) \wedge F(y, w),$$

$$M(x, y) \leftarrow N(x, y) \wedge F(y, x),$$

$$N_1(y, f_0^0) \leftarrow$$

$$N_1(y, f_0^1 z) \leftarrow N_1(y, z) \wedge \leqq (z, y),$$

$$M_1(y, f_0^1 z) \leftarrow N_1(y, z) \wedge \neg E(z),$$

$$S(x, w) \leftarrow M(x, z) \wedge M_1(z, y) \wedge F(y, w)\}.$$

c) *the projection function:*
$\varrho^{-1} \circ I_m^n \circ \varrho$ is Horn definable:

$$C_I \overset{d}{=} \{F_I(x_1, \ldots, x_n, x_m) \leftarrow\}.$$

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

d) *substitution:*
$f \overset{d}{=} \varrho^{-1} \circ Su(h, g_1, \ldots, g_n) \circ \varrho$ is Horn definable; where $Su(h, g_1, \ldots, g_n)$ is the function defined by substitution from $h, g_1, \ldots, g_n$.

$$C_f \overset{d}{=} \{F_f(x, y) \leftarrow F_h(y_1, \ldots, y_n, y) \wedge F_{g_1}(x, y_1) \wedge \ldots \wedge F_{g_n}(x, y_n)\} \cup$$

$$C_h \cup C_g \cup \ldots \cup C_{g_n}.$$

To prove that C_f really defines f, note that

$$\varrho^{-1} \circ Su(h, g_1, \ldots, g_n) \circ \varrho(\bar{x}) = \varrho^{-1}\big(h(g_1(\varrho(\bar{x})), \ldots, g_n(\varrho(\bar{x}))\big) =$$

$$= \varrho^{-1} \circ h \circ \varrho\big(\varrho^{-1} \circ g \circ \varrho(\bar{x}), \ldots, \varrho^{-1} \circ g_n \circ \varrho(\bar{x})\big).$$

(Similar remarks will be omitted in the following.)

e) *recursion:*
$f \overset{d}{=} \varrho^{-1} \circ R(g, h) \circ \varrho$ is Horn definable, where $R(g, h)$ is the function defined by recursion from g and h.

$$C_f \overset{d}{=} \{F_f(\bar{x}, f_0^0, y) \leftarrow F_g(\bar{x}, y),$$

$$F_f(\bar{x}, w, y) \leftarrow F_s(z, w) \wedge F_f(x, z, y_1) \wedge F_h(\bar{x}, z, y_1, y)\} \cup$$

$$C_g \cup C_s \cup C_h.$$

Remember that C_s defines the function $\varrho^{-1} \circ S \circ \varrho$, where S is the successor function on ω.

f) *the μ-operator:*
$f \overset{d}{=} \varrho^{-1} \circ Myg \circ \varrho$ is Horn definable, where Myg is the function defined by the μ-operator from g.

$$C_f \overset{d}{=} \{N(f_0^0) \leftarrow, N(w) \leftarrow S(z, w) \wedge N(z) \wedge F_g(\bar{x}, z, y) \wedge S(y_1, y),$$

$$F_f(\bar{x}, y) \leftarrow N(y) \wedge F_g(\bar{x}, y, f_0^0)\} \cup C_g \cup C_s.$$

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.

MATHEMATICAL INSTITUTE OF THE
HUNGARIAN ACADEMY OF SCIENCES
REÁLTANODA U. 13—15.
BUDAPEST, HUNGARY
H—1053

References

[1] WARREN, D. WARPLAN, A system for generating plans, *DCL Memo*, No. 76, 1974.
[2] BATTANI, G., H. MELONI, Interpreteur du language de programmation PROLOG, Université d'Aix Marseille, 1973.
]3] KOWALSKI, R., Predicate logic as programming language *DCL Memo*, No. 70, University of Edinburg, 1973, and Proc. IFIP Congr. 1974.
[4] KOWALSKI, R., Logic for problem solving. *DCL Memo*, No. 75, University of Edinburg, 1974.
[5] VAN EMDEN, M., First-order predicate logic as a high-level program language. School of AI, University of Edinburg, MIP—R—106, 1974.
[6] VAN EMDEN, M., R. KOWALSKI, The semantics of predicate logic as a programming language, *DCL Memo*, No. 73, University of Edinburg, 1974.
[7] VAN EMDEN, M., Recursion equations as predicate-logic programs, to be published.

(Received Jan. 13, 1978)