KYBERNETIKA — VOLUME *18* (1982), NUMBER 2

HORN CLAUSE PROGRAMS AND RECURSIVE FUNCTIONS DEFINED BY SYSTEMS OF EQUATIONS

JAN ŠEBELÍK

Every recursive function can be defined by a system of equations. In this paper, we describe an algorithm which transforms every such a system into a Horn clause program computing the same function. On the other hand for a Horn clause program, the corresponding system of equations is constructed. In the end, some syntactic properties of Horn clause programs obtained from a system of equations are discussed.

In his paper [6], Tärnlund introduced the concept of binary Horn clause programs and, in [7], he used binary Horn clause programs to simulate the behavior of a Universal Turing machine as one of the formalisms for algorithms.

In [4], we introduced the concept of stratifiable Horn clause programs. Stratifiable programs were suggested by a natural hierarchy of partial recursive functions with some basic functions at the bottom and generating new computable functions by the operations of composition, primitive recursion and minimization. We constructed a stratifiable Horn clause program for every computable function by induction on recursive functions. It was shown that these programs and binary Horn clause programs are close to each other in the following sense: every stratifiable Horn clause program can be transformed into a binary Horn clause program computing the same function and vice versa.

At Logic Programming Workshop, Debrecen (Hungary), July 80, Robert Kowalski raised the question of interpreting systems of equations and deducibility from them (viz. [2]) by means of Horn clause programs. In this paper, we shall describe an algorithm which transforms every system of equations into a Horn clause program computing the same functions. In this construction, every equation of the system is transformed into a Horn clause. The resulting program is obtained as the collection of these clauses. On the other hand, it will be shown that for every Horn clause program there is a system of equations defining the same functions. Moreover, we shall discuss some syntactic features of Horn clause programs obtained from a system of equations and, in the end, a certain syntactic characterization of Horn clause programs computing primitive recursive functions will be introduced.

0. PRELIMINARIES

Concepts of Horn Logic

Throughout the paper, we shall use the standard concepts and notations of Horn Logic. We refer the reader to [3], [7] for a more detailed exposition. We shall deal with first-order languages containing only two function symbols 0 and S, where 0 is a constant interpreted as zero and S is an unary function symbol interpreted as the successor function $S(x) = x + 1$. The terms of this language we call *arithmetical terms*, variable-free terms 0, S(0), SS(0), ... we call *numerals* and we identify them with natural numbers 0, 1, 2, ...

A *clause* is a disjunction of *literals*, i.e. a disjunction of atomic formulas and of negations of atomic formulas. *Horn clause* is a clause with at most one positive literal, an *assertion* is a Horn clause with exactly one positive literal and no negative literal. We call *Horn clause program* every set of Horn clauses with exactly one positive literal.

The Resolution principle with unification as a pattern matching algorithm is the only Inference Rule of so called Horn Logic.

If $\mathscr{P}$ is a Horn clause program and C_0 is a *goal statement*, i.e. a Horn clause with no positive literal, the sequence $C_0, C_1, \ldots, C_n$ of goal statements is called deduction of C_n from $\mathscr{P}$ and C_0 provided that for every i, C_{i+1} is the resolvent of of C_i and a clause D from $\mathscr{P}$ such that the leftmost atom of C_i and the only positive atom of D are the literals the clauses C_i and D are resolved upon. It follows from the properties of Linear Ordered Resolution (viz. [1]) and from the fact that every Horn clause program is a satisfiable set of clauses that $\mathscr{P} \cup \{C_0\}$ is not satisfiable iff there is a deduction starting with C_0 and refuting $\mathscr{P}$ and C_0, i.e. deduction of an empty clause (denoted $\square$, interpreted as *false*) from $\mathscr{P}$ and C_0. To find such a refutation one has to use a complete search strategy. We shall write $\mathscr{P}$, $C_i \vdash C_j$ to say that the goal statement C_j is deducible from C_i and $\mathscr{P}$. If A is a ground atom (i.e. a variable-free atomic formula), we shall write $\mathscr{P} \vdash A$ instead of $\mathscr{P}, \leftarrow A \vdash \square$ ($\leftarrow A$ is the notation of the goal statement consisting of the only negative literal non-A).

The following easy lemma is used in our proofs.

Lemma 1. Let $\mathscr{P}$ be a Horn clause program, A a ground atom. Let C be a clause from $\mathscr{P}$ of the form $A' \leftarrow B_1, \ldots, B_n$ such that A and A' can be unified by a most general unifier σ.

Then $\mathscr{P} \vdash A$ using C in the first step of the refutation iff there is a substitution η such that $\mathscr{P} \vdash B_i\sigma\eta$ holds for every $i = 1, \ldots, n$, all $B_i\sigma\eta$ being ground atoms.

Kleene's formalism for recursive functions

Kleene, [2], described a formal system for recursive functions. Let us recall some basic concepts.

The formal symbols of the Kleene's system are as follows: = (equals), S (successor), 0 (zero), $x, y, z, \ldots, x_1, x_2, \ldots$ (variables for natural numbers), $f, g, h, \ldots, f_1, f_2, \ldots$ (function letters, i.e. symbols for unspecified functions). Potentially infinite lists of variables and functions letters are supposed to be given.

The symbols $f, g, h, \ldots, f_1, f_2, \ldots$ are called "function letters" rather then "function symbols". We adopt this name to distinguish them from 0 and S.

The *terms* are 0, the variables, and expressions of the form St or $f(t_1, \ldots, t_n)$ where f is a function letter and $t, t_1, \ldots, t_n$ are terms.

An *equation* is defined as a formal expression $r = s$ where r, s are terms. The equations are the only formulas for Kleene's system. A *system of equations* is a finite nonempty set of equations.

In the system there are two inference rules.

R1 is a one-premise rule applicable to an equation $r = s$ containing a variable y. This rule allows to pass from the original equation to an equation resulting from $r = s$ by substituting a numeral b for y.

R2 is a two-premise rule applicable to variable-free equations $r = s$ and $h(a_1, \ldots, a_n) = b$ where h is a function letter and $a_1, \ldots, a_n, b$ are numerals. This rule allows to pass from the original equations to an equation resulting from $r = s$ by replacing all occurrences of $h(a_1, \ldots, a_n)$ in s by b simultaneously.

A *deduction* of an equation e (the *endequation* of the deduction) from a system E of equations is a sequence $e_1, \ldots, e_n$ where e_n is e and for all $i = 1, \ldots, n$, e_i is from E or e_i results from some e_j, $j < i$, using R1, or e_i results from some e_j, e_k, $j, k < i$, using R2.

If there is a deduction of e from E, then e *is deducible* from E (in symbols $E \vdash e$).

Example 1. Let E consist of $h(x, y) = 7$, $f(0) = 4$, $f(Sx) = h(x, f(x))$. We shall demonstrate the deduction of $f(2) = 7$ from E.

(1)	$f(Sx) = h(x, f(x))$	belongs to E
(2)	$f(2) = h(1, f(1))$	R1 to (1)
(3)	$f(1) = h(0, f(0))$	R1 to (1)
(4)	$f(0) = 4$	belongs to E
(5)	$f(1) = h(0, 4)$	R2 to (3), (4)
(6)	$h(x, y) = 7$	belongs to E
(7)	$h(0, y) = 7$	R1 to (6)
(8)	$h(0, 4) = 7$	R1 to (7)
(9)	$f(1) = 7$	R2 to (5), (8)
(10)	$f(2) = h(1, 7)$	R2 to (2), (9)
(11)	$h(1, y) = 7$	R1 to (6)
(12)	$h(1, 7) = 7$	R2 to (11)
(13)	$f(2) = 7$	R2 to (10), (12)

Note. Our aim is to deduce endequations of the form $f(a_1, \ldots, a_n) = b$ where f is a function letter and $a_1, \ldots, a_n, b$ are numerals. It can be seen that if $E \vdash f(a_1, \ldots, a_n) = b$ then the corresponding deduction contains only such equations whose left-hand sides consists of terms containing exactly one function letter; this function letter appears as the leftmost symbol. It follows from the fact that using R2 only term s in $r = s$ can be modified.

Therefore throughout the paper, we shall permit equations with only such terms on the left-hand side, which contain exactly one function letter as the leftmost symbol.

It can be shown that given a partial recursive function φ, there can be found a system E of equations and a function letter f corresponding to φ such that for all numerals $a_1, \ldots, a_n, b$

$$E \vdash f(a_1, \ldots, a_n) = b \quad \text{iff} \quad \varphi(a_1, \ldots, a_n) = b \,.$$

Proof goes by induction on the hierarchy of partial recursive functions. Basic functions, i.e. zero function, the successor function and the projection on the i-th coordinate, are defined by the systems $\{f(x) = 0\}$, $\{f(x) = Sx\}$, and $\{f(x_1, \ldots, x_n) = x_i\}$ respectively.

Given systems of equations for functions $g, g_1, \ldots, g_k, h$, the functions obtained by composition, primitive recursion and minimization are defined by adding the systems

$$\{f(x_1, \ldots, x_n) = h(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))\} \,,$$

$$\{f(0, x_2, \ldots, x_n) = g(x_2, \ldots, x_n),$$

$$f(Sx_1, x_2, \ldots, x_n) = h(x_1, \ldots, x_n, f(x_1, \ldots, x_n))\} \quad \text{and}$$

$$\{k(x, y, 0) = y,\ k(x, y, Sz) = k(x, Sy, g(x, Sy)),\ f(x) = k(x, 0, g(x, 0))\}$$

to the original systems defining the given functions respectively. (In the above construction, f is a function letter corresponding to the defined function, k is a new auxiliary function letter.)

It can be shown by induction on appropriate variables that the above constructions of systems of equations have the announced property.

Remark 1. We say that a system E of equations defines function f iff for all numerals $a_1, \ldots, a_n, b_1, b_2$

$$E \vdash f(a_1, \ldots, a_n) = b_1 \quad \text{and} \quad E \vdash f(a_1, \ldots, a_n) = b_2 \quad \text{implies} \quad b_1 = b_2 \,.$$

A system of equations defines in general rather a recursive predicate then a function. It will be shown later that these predicates strictly correspond to atomic formulas of Horn Logic. Note that it cannot be effectively decided, whether an arbitrary system of equations defines a function or not.

1. SYSTEMS OF EQUATIONS AND HORN CLAUSE PROGRAMS

In this section, it will be shown that

a) every system of equations can be transformed into a Horn clause program computing the same functions and

b) every Horn clause program in the language of arithmetic can be transformed into a system of equations defining the same functions.

These facts are strictly formulated in Theorem 1 and Theorem 2.

Construction 1. Let e be an equation of the form $g(s) = t$ where s is an appropriate tuple of arithmetical terms, t is an arbitrary term. We shall construct a clause called *translation of e*, as follows.

If t contains some term $h_1(r_1)$, where h_1 is a function letter and r_1 is an appropriate tuple of arithmetical terms, then we replace e by the couple of equations

$$g(s) = t_1\,, \quad h_1(r_1) = z_1$$

where z_1 is a new variable not occurring in e and t_1 is obtained from t by replacing all occurrences of $h_1(r_1)$ in t by z_1.

This process is repeated until we have a system of equations

$$g(s) = t_k,\ h_1(r_1) = z_1, \ldots, h_k(r_k) = z_k\,,$$

where t_k is an arithmetical term, h_i are function letters and r_i are appropriate tuples of arithmetical terms, $i = 1, \ldots, k$.

Now, let for $i = 1, \ldots, k$, G, H_i be predicate symbols corresponding to functions letters g, h_i respectively. The translation of e is the clause

$$G(s, t_k) \leftarrow H_1(r_1, z_1), \ldots, H_k(r_k, z_k)\,.$$

If E is a system of equations, we call *HC-translation of E* the Horn clause program consisting of the translations of all equations from E.

Example 2. Let e be $g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}f(x), g(\mathrm{S}x, \mathrm{S}f(x)))$. The right-hand side of e contains the term $f(x)$. We get

$$g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}z_1, g(\mathrm{S}x, z_1))\,, \quad f(x) = z_1$$

$$g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}z_1, z_2)\,, \quad f(x) = z_1\,, \quad g(\mathrm{S}x, z_1) = z_2$$

$$g(\mathrm{S}x, y) = \mathrm{S}z_3\,, \quad f(x) = z_1\,, \quad g(\mathrm{S}x, z_1) = z_2\,, \quad h(\mathrm{S}z_1, z_2) = z_3\,.$$

The translation of e is

$$G(\mathrm{S}x, y, \mathrm{S}z_3) \leftarrow F(x, z_1)\,, \quad G(\mathrm{S}x, z_1, z_2)\,, \quad H(\mathrm{S}z_1, z_2, z_3)\,.$$

Remark 2. We can see that for an n-ary function letter f, the corresponding predicate symbol F is of the arity $n + 1$, hence F is at least unary. Note that if t is

an arithmetical term then $k = 0$ and the translation of e is the assertion $G(s, t) \leftarrow$. Note that for $k > 0$, z_k occurs in t_k.

Lemma 2. Let E be a system of equations, let $\mathscr{P}$ be a HC-translation of E.

Then for every variable-free equation e holds:

if e is deducible from E then the translation of e is deducible from $\mathscr{P}$,

provided that the correspondence between function letters in E and predicate symbols in $\mathscr{P}$ is the same as the correspondence between function letters in e and predicate symbols in the translation of e.

Proof. Let e be a variable-free equation of the form $f(a) = r$, where a is an appropriate tuple of numerals and r is a variable-free term.

Let C be the translation of e of the form

$$F(a, t) \leftarrow H_1(t_1, z_1), \ldots, H_k(t_k, z_k)$$

where $t_1, \ldots, t_k$ are appropriate tuples of arithmetical terms, $z_1, \ldots, z_k$ are variables, t is an arithmetical term. Denote $\bar{C}$ the universal closure of C.

We say that C is deducible from $\mathscr{P}$ if

$$\mathscr{P}, D \vdash \square \tag{1}$$

where D is a Skolem form of the negation of $\bar{C}$.

Since the negation of $\bar{C}$ is

$$(\exists z_1) \ldots (\exists z_k)\,(\neg F(a, t) \,\&\, H_1(t_1, z_1) \,\&\, \ldots \,\&\, H_k(t_k, z_k))\,,$$

we have

$$\leftarrow F(a, t') \,\&\, H_1(t'_1, c_1) \leftarrow \,\&\, \ldots \,\&\, H_k(t'_k, c_k) \leftarrow$$

for D, where c_i are Skolem constants and t', t'_i are obtained by substitution c_i for z_i in t, t_i respectively. (Let us recall that for $k > 0$, z_k occurs in t).

In other words, (1) is equivalent to

$$\mathscr{P} \cup \{H_i(t'_i, c_i) \leftarrow;\ i = 1, \ldots, k\} \vdash F(a, t')\,.$$

We shall prove Lemma 2 by induction on the length of the deduction of e.

1. The case $e \in E$ is trivial because $C \in \mathscr{P}$.

2. a) Let $E \vdash e'$ and e be obtained from e' using R1.

Then the translation C' of e' is deducible from $\mathscr{P}$ according to induction hypothesis and C is an instance of C'. Hence C is deducible from $\mathscr{P}$.

b) Let e be obtained from (variable-free) equations e_1, e_2 using R2, $E \vdash e_1$, $E \vdash e_2$.

Then

$$e_1 \text{ is of the form } f(a) = r'\,,$$

$$e_2 \text{ is of the form } g(b) = d\,,$$

where r' is a variable-free term, b is an appropriate tuple of numerals, d is a numeral,

and $g(b)$ occurs in r', r being obtained by replacing all occurrences if $g(b)$ in r' by d.

The translations C_1, C_2 of e_1, e_2 have the forms

$$F(a, t) \leftarrow G(b, z_0), \quad H_1(t_1'', z_1), \ldots, H_k(t_k'', z_k), \tag{2}$$

$$G(b, d) \leftarrow \quad \text{respectively}, \tag{3}$$

where for $i = 1, \ldots, k$, t_i are obtained by substituting d for z_0 in t_i''. It follows from the induction hypothesis that C_1, C_2 are deducible from $\mathscr{P}$. Hence $\mathscr{P} \cup \{C_1, C_2\}$ is a conservative extension of $\mathscr{P}$. We shall conclude the proof by showing that C is deducible from $\mathscr{P} \cup \{C_1, C_2\}$.

We start the refutation from

$\leftarrow F(a, t')$ and we get

$\leftarrow G(b, z_0),\ H_1(t_1'', z_1), \ldots, H_{k-1}(t_{k-1}'', z_{k-1}),\ H_k(t_k'', c_k)$ using (2),

$\leftarrow H_1(t_1, z_1), \ldots, H_{k-1}(t_{k-1}, z_{k-1}),\ H_k(t_k, c_k)$ using (3),

and

$\square$ using the clauses $H_i(t_i', c_i) \leftarrow$ for $i = 1, \ldots, k$.

This completes the proof of Lemma 2.

Lemma 3. Let E be a system of equations, $\mathscr{P}$ be a HC-translation of E. Let F be the predicate symbol corresponding to a function letter f from E.

Then for all numerals $(a_1, \ldots, a_n) = a$ and b,

$$\text{if} \quad \mathscr{P} \vdash F(a, b) \quad \text{then} \quad E \vdash f(a) = b\,.$$

Proof. We shall prove Lemma 3 by induction on the length of the deduction of $\mathscr{P}, \leftarrow F(a, b) \vdash \square$, which is equivalent to $\mathscr{P} \vdash F(a, b)$.

1. Let $\mathscr{P}, \leftarrow F(a, b) \vdash \square$ in exactly one resolution step. Then for some arithmetical terms t, $(s_1, \ldots, s_n) = s$ the assertion $F(s, t) \leftarrow$ belongs to $\mathscr{P}$ and $F(a, b)$ is unifiable with $F(s, t)$. Hence $f(s) = t$ belongs to E and $E \vdash f(a) = b$ using R1 several times.

2. Let $\mathscr{P}, \leftarrow F(a, b) \vdash D$ in one resolution step using some clause C from $\mathscr{P}$ and an unifier σ.

Let $\mathscr{P}, D \vdash \square$.

Clearly, C is a translation of some $f(s) = r$ of E and C is of the form

$$F(s, t) \leftarrow H_1(t_1, z_1), \ldots, H_k(t_k, z_k)\,,$$

where $t_1, \ldots, t_k$ are appropriate tuples of arithmetical terms, $z_1, \ldots, z_k$ are variables, z_k occurs in the arithmetical term t and $H_1, \ldots, H_k$ are the predicate symbols corresponding to function letters $h_1, \ldots, h_k$ occurring in r, respectively.

Hence D is of the form

$$\leftarrow H_1(t_1, z_1)\,\sigma, \ldots, H_k(t_k, z_k)\,\sigma$$

and using Lemma 1 we have

$$\mathscr{P} \vdash H_i(t_i, z_i)\,\sigma\eta \quad \text{for} \quad i = 1, \ldots, k \quad \text{and some substitution } \eta\,,$$

$H_i(t_i, z_i)\,\sigma\eta$ being ground atoms.

Let $t_i\sigma\eta$ be c_i, $z_i\sigma\eta$ be d_i, $r\sigma\eta$ be r'. It follows from the induction hypothesis that $E \vdash h_i(c_i) = d_i$. Using R1 several times, $f(a) = r'$ can be deduced from $f(s) = r$.

Now, $h_1(c_1)$ occurs in r' and applying R2 to $f(a) = r'$ and $h(c_1) = d_1$ we have $f(a) = r_1$ where r_1 is obtained by replacing $h(c_1)$ by d_1 in r'; $h_2(c_2) = d_2$ occurs in r_1 and applying R2 again to $f(a) = r_1$ and $h_2(c_2) = d_2$, we obtain $f(a) = r_2$.

Repeating this process, we get $f(a) = r_k$, but r_k is $t\sigma\eta$ and $t\sigma\eta$ is b.

We have $E \vdash f(a) = b$.

Now, the first part of the relationship between Kleene's formal system for recursive functions and Horn clause proframs can be stated as follows.

Theorem 1. Let E be a system of equations, let $\mathscr{P}$ be a HC-translation of E, let F be the predicate symbol corresponding to a function letter f from E.

Then for every tuple a of numerals and every numeral b,

$$E \vdash f(a) = b \quad \text{iff} \quad \mathscr{P} \vdash F(a, b)\,. \tag{4}$$

Proof. One implication is formulated in Lemma 3. The other implication is a particular case of Lemma 2. Namely, using Lemma 2 for $f(a) = b$, the corresponding translation is $F(a, b) \leftarrow$ and the right hand side of (4) follows.

Example 3. Let us have the system E of equations for Example 1. The HC-translation of E is a Horn clause program $\mathscr{P}$ consisting of the clauses

$$H(x, y, 7) \leftarrow\,, \qquad F(0, 4) \leftarrow\,, \qquad F(\mathrm{S}x, z) \leftarrow F(x, y)\,,\ H(x, y, z)\,.$$

We shall demonstrate the deduction $\mathscr{P} \vdash F(2, 7)$.

$$\leftarrow F(2, 7)$$
$$\leftarrow F(1, y)\,,\quad H(1, y, 7)$$
$$\leftarrow F(0, y')\,,\quad H(0, y', y)\,,\quad H(1, y, 7)$$
$$\leftarrow H(0, 4, y)\,,\quad H(1, y, 7)$$
$$\leftarrow H(1, 7, 7)$$
$$\square$$

Note that Theorem 1 holds for all systems of equations, not only for the systems defining functions (viz. Remark 1). For every system of equations, Theorem 1 gives a Horn clause program satisfying (4).

Now, for every Horn clause program, we shall construct a system of equations of the same properties as above.

Construction 2. Let $\mathscr{P}$ be a Horn clause program, every predicate symbol occurring in $\mathscr{P}$ being at least unary.

Let E be a system of equations obtained as follows. We call E the *E-translation of* $\mathscr{P}$.

(i) whenever $G(s, t) \leftarrow$ is an assertion of $\mathscr{P}$, then
$g(s) = t$ belongs to E.

(ii) whenever $G(s, t) \leftarrow H_1(s_1, t_1), \ldots, H_k(s_k, t_k)$ is a clause of $\mathscr{P}$, then the couple of equations
$g(s) = g'(s_1, h_1(s_1), \ldots, s_k, h_k(s_k))$,
$g'(s_1, t_1, \ldots, s_k, t_k) = t$ belongs to E,

In (i), (ii), $s, s_1, \ldots, s_k$ are tuples of arithmetical terms,
$t, t_1, \ldots, t_k$ are arithemtical terms,
$g, h_1, \ldots, h_k$ are functions letters corresponding to predicate symbols $G, H_1, \ldots, H_k$ respectively, and g' is an auxiliary function letter.

Theorem 2. Let $\mathscr{P}$ be a Horn clause program, every predicate symbol of $\mathscr{P}$ being at least unary. Let E be the E-translation of $\mathscr{P}$.

Then for every predicate symbol F of $\mathscr{P}$, every tuple a of numerals and numeral b,

$$\mathscr{P} \vdash F(a, b) \quad \text{iff} \quad E \vdash f(a) = b\,,$$

f being the function letter corresponding to F.

Proof. Let $\mathscr{P}'$ be a HC-translation of E. We shall suppose that every predicate symbol of $\mathscr{P}$ occurs in $\mathscr{P}'$ in a natural way, i.e. if a function letter g of E corresponds to some G of $\mathscr{P}$ then G of $\mathscr{P}'$ corresponds to g.
It follows from Theorem 1 that

$$E \vdash f(a) = b \quad \text{iff} \quad \mathscr{P}' \vdash F(a, b)\,.$$

We shall show that

$$\mathscr{P} \vdash F(a, b) \quad \text{iff} \quad \mathscr{P}' \vdash F(a, b) \tag{5}$$

for all F form $\mathscr{P}$.

First shall show that every clause of $\mathscr{P}$ is deducible in $\mathscr{P}'$. Let C be a clause from $\mathscr{P}$ of the form

$$G(s, t) \leftarrow H_1(s_1, t_1), \ldots, H_k(s_k, t_k) \tag{6}$$

where $s, s_1, \ldots, s_k$ are tuples of arithmetical terms,
$t, t_1, \ldots, t_k$ are arithmetical terms.

If $k = 0$ then C is an assertion and $g(s) = t$ belongs to E due to (i) of the Construction 2. It follows from Remark 2 that C belongs to $\mathscr{P}'$.

Let $k > 0$, let e_1, e_2 are the equations obtained from C due to (ii) of the Construction 2.

The translations of e_1, e_2 have then the forms

$$G(s, z) \leftarrow H_1(s_1, z_1), \ldots, H_k(s_k, z_k), \quad G'(s_1, z_1, \ldots, s_k, z_k, z),$$

$$G'(s_1, t_1, \ldots, s_k, t_k, t) \leftarrow ,$$

where $z_1, \ldots, z_k, z$ are variables. These two clauses belong to $\mathscr{P}'$. They can be resoved using the unifier $\{z_i/t_i, z/t\}$ and C is the resolvent of them. Hence C is deducible from $\mathscr{P}'$.

We have: every clause C of $\mathscr{P}$ is deducible from $\mathscr{P}'$. This gives one implication from (5).

To prove the other implication, let $\mathscr{P}' \vdash F(a, b)$, F occurs in $\mathscr{P}$. We shall show $\mathscr{P} \vdash F(a, b)$ by induction on the length of the refutation $\mathscr{P}'$, $\leftarrow F(a, b) \vdash \square$.

1. If $\leftarrow F(a, b)$ can be refuted from $\mathscr{P}'$ in one step then it can be refuted from $\mathscr{P}$ in one step, too, as Horn clause programs $\mathscr{P}$ and $\mathscr{P}'$ have exactly the same assertions, with the exception of the assertions for auxiliary functions letters, which follows from Construction 1 and Construction 2.

2. Let

$$\mathscr{P}', \ \leftarrow F(a, b) \vdash \leftarrow H_1(s_1, z_1)\,\sigma, \ldots, H_k(s_k, z_\kappa)\,\sigma, F'(s_1, z_1, \ldots, s_k, z_k, z)\,\sigma$$

$$\vdash \ldots \vdash \square$$

using a clause

$$F(s, z) \leftarrow H_1(s_1, z_1), \ldots, H_k(s_k, z_k), \quad F'(s_1, z_1, \ldots, s_k, z_k, z)$$

from $\mathscr{P}'$ in the first step of refutation.

It follows from the construction of E and $\mathscr{P}'$ that for some arithmetical terms $t, t_1, \ldots, t_k$,

$$F'(s_1, t_1, \ldots, s_k, t_k, t) \leftarrow \qquad \text{belongs to } \mathscr{P}' \text{ and}$$

$$F(s, t) \leftarrow H_1(s_1, t_1), \ldots, H_k(s_k, t_k) \quad \text{belongs to } \mathscr{P}.$$

It follows from Lemma 1 that there exists a substitution η such that

$$\mathscr{P}' \vdash H_i(s_i, z_i)\,\sigma\eta \quad \text{for} \quad i = 1, \ldots, k \quad \text{and}$$

$$F'(s_1, z_1, \ldots, s_k, z_k, b)\,\sigma\eta \quad \text{is} \quad F'(s_1, t_1, \ldots, s_k, t_k, t)\,\sigma\eta,$$

all $H_i(s_i, z_i)\,\sigma\eta$ are ground instances of $H_i(s_i, t_i)$,

$$b \text{ is } t\sigma\eta \quad \text{and} \quad a \text{ is } s\sigma\eta.$$

It follows from the induction hypothesis that

$$\mathscr{P} \vdash H_i(s_i, z_i)\,\sigma\eta \quad \text{for} \quad i = 1, \ldots, k.$$

Hence $F(a, b)$ can be unified with $F(s, t)$ using the unifier $\sigma\eta$ and $\mathscr{P} \vdash F(a, b)$ using Lemma 1.

This completes the proof of the theorem.

Example 4. Let $\mathscr{P}$ be a Horn clause program consisting of

$$F(x, Sz) \leftarrow G(x, Su),\ H(z, x)$$
$$G(2, 1) \leftarrow$$
$$H(0, y) \leftarrow .$$

The E-translation of $\mathscr{P}$ consists of

(1) $f(x) = f'(x, g(x), z, h(z))$
(2) $f'(x, Su, z, x) = Sz$
(3) $g(2) = 1$
(4) $h(0) = y$.

It is easy to see that $\mathscr{P} \vdash F(2, 1)$. We shall demonstrate the deduction of $E \vdash f(2) = 1$.

(5)	$f(2) = f'(2, g(2), 0, h(0))$	R1 to (1), twice
(6)	$f(2) = f'(2, 1, 0, h(0))$	R2 to (5), (3)
(7)	$h(0) = 2$	R1 to (4)
(8)	$f(2) = f'(2, 1, 0, 2)$	R2 to (6), (7)
(9)	$f'(2, 1, 0, 2) = 1$	R1 to (2), three times
(10)	$f(2) = 1$	R2 to (8), (9)

In Section 1, we have described natural transformations of systems of equations into Horn clause programs and vice versa. This shows a close link between these formalisms for recursive functions or predicates. This link is reflected both between equations and Horn clauses and between the deduction mechanisms.

2. SOME SYNTACTIC FEATURES OF HORN CLAUSE PROGRAMS

We say that a Horn clause C is a *TE-clause* if there is an equation e such that C is the translation of e according to Construction 1.

Lemma 4. Let C be of the form

$$G(s, t) \leftarrow H_1(s_1, t_1), \ldots, H_k(s_k, t_k),$$

where $s, s_1, \ldots, s_k$ are tuples of arithmetical terms,
$t, t_1, \ldots, t_k$ are arithmetical terms, $k > 0$.
Then C is a TE-clause iff

(i) t_i are pairwise different variables not occurring in s
(ii) t_i does not occur in s_j for $j \leqq i$
(iii) for $i \neq k$, t_i occurs in some s_j, $j > i$
(iv) t_k occurs in t

Note that for $k = 0$, C is ever a TE-clause.

Proof. It follows from Construction 1 that if C is TE-clause then (i)–(iv) hold. Let (i)–(iv) hold.

Let e_0 be $f(s) = t$. Let us have $h_1, \ldots, h_k$ as function letters corresponding to $H_1, \ldots, H_k$.

Let for $i = 1, \ldots, k$, e_i be $f(s) = r_i$ where r_i is obtained by substituting $h_{k-i+1}(s_{k-i+1})$ for t_{k-i+1} in r_{i-1}, r_0 being t. Note that the variables t_{k-i} occur in r_i, t_k occurs in r_0.

It can be shown that C is the translation of e_k. We shall demonstrate it in Example 5.

Example 5. Let C be $G(\mathrm{S}x, y, \mathrm{S}z_3) \leftarrow F(x, z_1)$, $G(\mathrm{S}x, \mathrm{S}z_1, z_2)$, $H(\mathrm{S}z_1, z_2, z_3)$. For C, (i)–(iv) hold, $k = 3$. We have

e_0 $\quad g(\mathrm{S}x, y) = \mathrm{S}z_3$

e_1 $\quad g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}z_1, z_2)$

e_2 $\quad g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}z_1, g(\mathrm{S}x, \mathrm{S}z_1))$

e_3 $\quad g(\mathrm{S}x, y) = \mathrm{S}h(\mathrm{S}f(x), g(\mathrm{S}x, \mathrm{S}f(x)))$

It was shown in Example 1 that C is the translation of e_3.

Remark 3. Suppose that $h_1, \ldots, h_k$ are function letters occurring in a system E of equations. Let the functions corresponding to $h_1, \ldots, h_k$ be primitive recursive. Suppose that f is a new function letter, x is a variable, y is a tuple of variables. Let $f(0, y)$ be expressed in terms of y, $h_1, \ldots, h_k$ as a function of y, let $f(\mathrm{S}x, y)$ be expressed in terms of x, y, $h_1, \ldots, h_k$, and $f(x, t)$ as a function of t. It is known (viz [2], § 55) that the function corresponding to f is then primitive recursive.

We shall formulate a similar fact as above for Horn clause programs. To express an analogy for a hierarchical definition of a primitive recursive function, we shall recall the notion of computation tree for a Horn clause program $\mathscr{P}$ and predicate symbol F, introduced in [5].

Let $\mathscr{P}$ be a Horn clause program, F a predicate symbol occurring in $\mathscr{P}$. An AND/OR tree $\mathbb{T}$ is called a *computation tree for $\mathscr{P}$ and F* provided that

(i) the OR-nodes of $\mathbb{T}$ are labelled by predicate symbols from $\mathscr{P}$ or by the empty clause, and AND-nodes of $\mathbb{T}$ are labelled by clauses from $\mathscr{P}$.

(ii) the root of $\mathbb{T}$ is an OR-node labelled by F.

(iii) if n is an OR-node of $\mathbb{T}$ labelled by a predicate symbol Q, the successors of n are AND-nodes labelled by all clauses whose head contains Q.

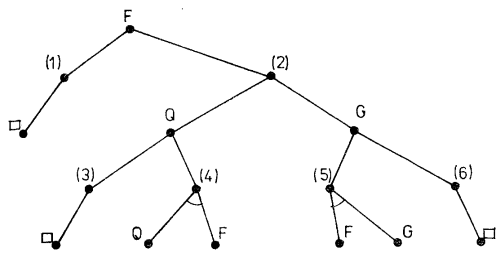
(iv) if n is an AND-node of $\mathbb{T}$ labelled by a clause C, the successors of n are OR-nodes labelled by the predicate symbols occurring in the body of C, if C is an assertion, the only successor of n is an OR-node labelled by the empty clause.

Given a program $\mathscr{P}$ and a predicate symbol F, computation tree $\mathbb{T}$ for $\mathscr{P}$ and F may be infinite. A *reduced computation tree for $\mathscr{P}$ and F* is the maximal subtree $\mathbb{T}'$ of $\mathbb{T}$ such that the root of $\mathbb{T}'$ coincides with the root of $\mathbb{T}$ and no branch of $\mathbb{T}'$ containes a pair of END-nodes labelled by the same clause. Note that the reduced computation tree is finite.

Example 6. Let a Horn clause program $\mathscr{P}$ be of the form

(1) $F() \leftarrow$
(2) $F() \leftarrow Q(), G()$
(3) $Q() \leftarrow$
(4) $Q() \leftarrow Q(), F()$
(5) $G() \leftarrow F(), G()$
(6) $G() \leftarrow$

The reduced computation tree for $\mathscr{P}$ and F is illustrated below.



Now, we shall introduce the concept of level-founded programs. A Horn clause program $\mathscr{P}$ is called *level founded with respect to a predicate symbol* F occurring in $\mathscr{P}$ if on every branch of the reduced computatin tree for $\mathscr{P}$ and F there is no OR-node between two OR-nodes labelled by the same predicate symbol. A Horn clause program is said to be *level-founded* if it is level founded with respect to all predicate symbols occurring in it.

Remark 4. It can be seen that $\mathscr{P}$ is level-founded iff there is a maping s that assigns a natural number to every predicate symbol in $\mathscr{P}$ such that for every clause $G() \leftarrow$ $\leftarrow H_1(), \ldots, H_k()$ from $\mathscr{P}$, $s(H_i) > s(G)$ provided that H_i is not G.

Note that the Horn clause program from Example 6 is not level founded. It is level founded with respect to no predicate symbol occurring in it.

Theorem 3. Let a Horn clause program $\mathscr{P}$ be level-founded with respect to F. Let every clause of $\mathscr{P}$ be a TE-clause of the form

$$G(s, t) \leftarrow H_1(s_1, z_1), \ldots, H_k(s_k, t_k),$$

any variable occurring in tuples $s_1, \ldots, s_k$ of arithmetical terms or in arithmetical term t being either one of $z_1, \ldots, z_k$ or from the tuple s of arithmetical terms.

Let every OR-node of the reduced computation tree for $\mathscr{P}$ and F have
either exactly one AND-successor labelled by a clause of the above form, such that
$s = (x, y)$ and H_i are different from G,

or exactly two AND-successors labelled by clauses C_1, C_2 of the above form, such that
for C_1, $s = (0, y)$ and H_i are different from G,
for C_2, $s = (Sx, y)$ and if H_i is G then $s_i = (x, r_i)$,

where x is a variable, y is a tuple of variables, and r_i are arbitrary tuples of arithmetical terms.

Then there is a primitive recursive function φ corresponding to F, i.e. for every numeral b and every tuple a of numerals,

$$\varphi(a) = b \quad \text{iff} \quad \mathscr{P} \vdash F(a, b) .$$

It is easy to see (viz. [4], [5]) that every primitive recursive function may be specified by a Horn clause program of the above properties.

Sketch of proof. The proof goes by induction on the heigth of the reduced computation tree for $\mathscr{P}$ and F.

Let E be a system of equations such that $\mathscr{P}$ is a HC-translation of E. It can be seen both for the basis of induction and for the induction step, that the equations from E, which define the function corresponding to F from lower level functions, satisfy the conditions of Remark 3. Hence, the function corresponding to F is primitive recursive.

Remark 5. It seems that Theorem 3 can be generalized for k-recursive functions in the way of permitting recursions on more variables. Theorem 3 would be a particular case of such a theorem. As we find such a generalization to be neither interesting nor important, we leave it.

Remark 6. For every Horn clause C, the conditions (i)–(iv) in Lemma 4 are equivalent to C being a TE-clause. Theorem 3, after replacing the assumption that every clause of $\mathscr{P}$ is a TE-clause by (i)–(ii) from Lemma 4, keeps true.

ACKNOWLEDGEMENT

The author is very grateful to Dr. Petr Štěpánek for the valuable questions and comments he has made during the formation of the paper.

(Received May 27, 1981.)

REFERENCES

[1] C. L. Chang, R. T. C. Lee: Symbolic Logic and Mechanical Theorem Proving. Academic Press New York 1971.
[2] S. C. Kleene: Introduction to Metamathematics. North-Holland Publishing Co. — Amsterodam, P. Noordhoff N. V. — Groningen 1967.
[3] R. Kowalski: Logic for Problem Solving. North-Holland, New York 1979.
[4] J. Šebelík, P. Štěpánek: Horn clause programs suggested by recursive functions. In: Proceed-

ings of the Logic Programming Workshop (S. Å. Tärnlund, ed.), Debrecen (Hungary), July 14—16, 1980.
[5] J. Šebelík, P. Štěpánek: Horn clause programs for recursive functions. Preprint, 1980.
[6] S. Å. Tärnlund: Logic Information Processing. TRITA-IBADB-91029, 1975-11-24, Dept. Comp. Sci., Royal Institute of Tech., Stockholm 1975.
[7] S. Å. Tärnlund: Horn clause computability. BIT *17* (1977), 215—226.

Jan Šebelík, Ústav pro využití výpočetní techniky v řízení (Institute for Application of Computing Technique in Control), Revoluční 24, 110 00 Praha 1, Czechoslovakia.