Logic-Programs for Primitive-Recursive Predicates

Urs-Martin Künzi

Institut für Informatik und angewandte Mathematik
Universität Bern, Länggassstr. 52,
3012 Bern, Switzerland
Tel: +41 31 65 39 84,
e-mail: kuenzi@iam.unibe.ch

Abstract

Meyer and Ritchie gave a description of primitive recursive functions by loop-programs ([5], [1]). In this paper a class of logic-programs is described which computes the primitive-recursive sets on Herbrand-universes. Furthermore, an internal description of primitive-recursive functions and sets on Herbrand-universes is given.

Keywords

Computing Reviews Classification: F.4.1 Logic Programming,
Recursive Function Theory.
Further Keyword: Primitive-Recursive Sets.

1 Introduction

Let L be a finite set of constants and function-symbols, containing at least one constant and one function-symbol, and let $\mathfrak{H}$ be the *Herbrand-model* of L (the L-structure with the set of closed L-terms as underlying set H; this set is the *Herbrand-universe* of L). A subset of H^n is *primitive-recursive* iff its characteristic function is primitive-recursive with respect to a fixed Gödelization of H. In the next section, an intrinsic description of primitive-recursive predicates is given (that does not depend on a Gödelization). An *L-program* is a definite program T ([3], chap. 2), containing only constants and function-symbols from L, together with a distinguished predicate $\mathtt{T}$. The *set computed by* T is the set $\{(t_1, \ldots, t_n) \in H^n \mid \mathsf{T} \vdash \mathtt{T}(t_1, \ldots, t_n)\}$. In this article we describe syntactically a class of L-programs, computing precisely the primitive-recursive subsets of H^n.

A program-clause is called *closed* iff its body contains only variables also contained in its head. For an L-term t with variables $x_1, \ldots, x_n$ we define $\ell(t)$ to be the linear polynomial with coefficients in $\mathbb{N}$ and with variables $|x_1|, \ldots, |x_n|$, recursively defined by $\ell(c) := 1$ for all constants, $\ell(x_i) := |x_i|$ and $|f(t_1, \ldots, t_n)| :=$

$1+|t_1|+\cdots+|t_n|$. For $\ell(t)$ we simply write $|t|$. If t is a variable-free term, then $|t|$ is the number of constants and function-symbols contained in t. We order the polynomials by means of

$$f(\bar{x}) \leq g(\bar{x}) \;:\Longleftrightarrow\; \forall \bar{m} \in (\mathbb{N} \setminus \{0\})^n \; f(\bar{m}) \leq g(\bar{m}) \tag{1}$$

and

$$f(\bar{x}) < g(\bar{x}) \;:\Longleftrightarrow\; \forall \bar{m} \in (\mathbb{N} \setminus \{0\})^n \; f(\bar{m}) < g(\bar{m}) \tag{2}$$

So given $f(\bar{x}) := a + b_1|x_1| + \cdots + b_n|x_n|$ and $g(\bar{x}) := c + d_1|x_1| + \cdots + d_n|x_n|$, we have that

$$f(\bar{x}) \leq g(\bar{x}) \iff \forall i(b_i \leq d_i) \;\wedge\; a + b_1 + \cdots + b_n \leq c + d_1 + \cdots + d_n \tag{3}$$

and

$$f(\bar{x}) < g(\bar{x}) \iff \forall i(b_i \leq d_i) \;\wedge\; a + b_1 + \cdots + b_n < c + d_1 + \cdots + d_n. \tag{4}$$

Given terms $s(x_1, \ldots, x_n)$ and $t(x_1, \ldots, x_n)$ we have $|s| \leq |t|$ (respectively $|s| < |t|$) iff for all tupels of variable-free terms $(u_1, \ldots, u_n)$ it holds $|s(u_1, \ldots, u_n)| \leq |t(u_1, \ldots, u_n)|$ (respectively $|s(u_1, \ldots, u_n)| < |t(u_1, \ldots, u_n)|$). n-tupels of terms are ordered (lexicographically) by

$$(t_1, \ldots, t_n) \prec (s_1, \ldots, s_n) \;:\Longleftrightarrow\; \exists i \leq n \, (\forall j < i(|t_j| = |s_j|) \wedge |t_i| < |s_i|). \tag{5}$$

A L-program T is called *tame* iff it satisfies the following conditions.

(i) All clauses of T are closed.

(ii) The predicates of T can be so linearly ordered, that $\mathtt{T}$ is the greatest predicate, and for every clause of T, the predicate of its head is greater than or equal to every predicate of its body.

(iii) If a clause of T with head $\mathtt{R}(t_1, \ldots, t_n)$ contains in its body the formula $\mathtt{R}(s_1, \ldots, s_n)$, then $(s_1, \ldots, s_n) \prec (t_1, \ldots, t_n)$.

We now choose a fixed order satiesfying (ii). Then we can order atomic formulas by

$$\mathtt{Q}(t_1, \ldots, t_n) \prec \mathtt{R}(s_1, \ldots, s_m) \;:\Longleftrightarrow\; \begin{cases} \mathtt{Q} < \mathtt{R} & \text{or} \\ \mathtt{Q} = \mathtt{R} \;\&\; (t_1, \ldots, t_n) \prec (s_1, \ldots, s_n). \end{cases} \tag{6}$$

This is a well-founded ordering of the variable-free atomic formulas. If we resolve a variable-free atomic formula Φ with a clause of a tame program, the resolvent consists of variable-free atoms smaller than Φ. It follows that tame programs, applied to variable-free atomic formulas, always stop.

We shall show that the tame programs compute primitive-recursive sets and that every primitive-recursive set is computed by a tame program.

2 Primitive Recursion on a Herbrand-Universe

Before we continue our investigations on tame programs, we want to give an internal description of primitive-recursion on Herbrand-universes. This approach follows [2]. With respect to the fixed language L, let PR_L be the smallest set of functions $H^n \to H$ (for all $n \in \mathbb{N}$) satisfying the following conditions.

(iv) Every function $\tilde{t} : H^n \to H : (a_1, \ldots, a_n) \mapsto t(a_1, \ldots, a_n)$, with $t(x_1, \ldots, x_n)$ an arbitrary L-term, is in PR_L.

(v) PR_L is closed under compositions of functions, i.e. if $\varphi(x_1, \ldots, x_n)$ and $\psi_i(\bar{y})$ are in PR_L, then $\varphi(\psi_1(\bar{y}), \ldots, \psi_n(\bar{y}))$ is also in PR_L.

(vi) For every constant c and for every n-ary function-symbol f, let $\varphi_c(\bar{y})$ and $\varphi_f(x_1, \ldots, x_n, z_1, \ldots, z_n, \bar{y})$ be functions of PR_L. Then the function φ, defined by

$$\varphi(c, \bar{y}) := \varphi_c(\bar{y}) \tag{7}$$

$$\varphi(f(x_1, \ldots, x_n), \bar{y}) := \varphi_f(x_1, \ldots, x_n, \varphi(x_1, \bar{y}), \ldots, \varphi(x_n, \bar{y}), \bar{y}), \tag{8}$$

is in PR_L.

By (iv), PR_L contains all projection-functions, so the recursion-schema (vi) holds for all arguments (and not only for the first one).

Let $\gamma : H \to \mathbb{N}$ be a primitive-recursive Gödelization of H, i.e. an injection with the following properties.

(vii) γ is increasing, i.e. if s is a subterm of t then $\gamma(s) < \gamma(t)$.

(viii) If f is a function-symbol with arity n_f then there is a primitive-recursive function $\gamma_f : \mathbb{N}^{n_f} \to \mathbb{N}$ such that $\gamma(f(t_1, \ldots, t_{n_f})) = \gamma_f(\gamma(t_1), \ldots, \gamma(t_{n_f}))$ for all t_i.

(ix) Let $f_1, \ldots, f_r$ be an enumeration of the function-symbols and constants of L (where the constants thought of as 0-ary function-symbols), and let n_i be the arity of f_i. Then the map $\pi_0 : \mathbb{N} \to \mathbb{N}$, defined by $\pi_0(\gamma(f_i(t_1, \ldots, t_{n_i}))) := i$ and $\pi_0(x) := 0$ if $x \notin \mathrm{Im}(\gamma)$, is primitive-recursive.

(x) For any natural number $i > 0$ which does not exceed all arities of function-symbols of L, the map $\pi_i : \mathbb{N} \to \mathbb{N}$, defined by $\pi_i(\gamma(f_j(t_1, \ldots, t_{n_j}))) := \gamma(t_i)$ if $i \leq n_j$, else $\pi_i(x) := 0$, is primitive recursive.

We remark that (ix) and (x) are consequences of the other conditions. Let $\mathbb{G} \subset \mathbb{N}$ be the image of γ, and let $\gamma^{\times n} : H^n \to \mathbb{G}^n$ be the map with $(t_1, \ldots, t_n) \mapsto (\gamma(t_1), \ldots, \gamma(t_n))$. Then to any map $\psi : H^n \to H$ corresponds a unique map $\psi^* : \mathbb{G}^n \to \mathbb{G}$ for which the following diagram commutes:

$$\begin{tikzcd} H^n \arrow[r, "\psi"] \arrow[d, "\gamma^{\times n}"] & H \arrow[d, "\gamma"] \\ \mathbb{G}^n \arrow[r, "\psi^*"] & \mathbb{G} \end{tikzcd}$$

The function $\varphi : H^n \to H$ is called *primitive-recursive* iff φ^* is the restriction of a primitive-recursive function; a subset $U \subset H^n$ is called *primitive-recursive* iff $\gamma^{\times n}(U)$ is a primitive-recursive set. Exploiting (viii) it is easy to see by course-of-values-recursion that the functions of PR_L are primitive-recursive. We show that the contrary is also true.

Fix a constant 0 and a 1-ary function-symbol $'$. (If there is no 1-ary function symbol in L, then we take x' to be an abbreviation for $g(x, 0, \ldots, 0)$, where g is a fixed function-symbol). Then $\iota : \mathbb{N} \to H$, defined by $\iota(0) := 0$ and $\iota(n+1) := \iota(n)'$, is an injection. Let N be the image of ι. Again given any function $\varphi : \mathbb{N} \to \mathbb{N}$, there is a unique function $\varphi^\circ : N^n \to N$, for which the following diagram commutes:

$$\begin{tikzcd} \mathbb{N}^n \arrow[r, "\varphi"] \arrow[d, "\iota^{\times n}"] & \mathbb{N} \arrow[d, "\iota"] \\ N^n \arrow[r, "\varphi^\circ"] & N \end{tikzcd}$$

It follows immediately from the definitions that if φ is primitive recursive, then φ° is the restriction of a function of PR_L. Let $f(t_1, \ldots, t_n)$ be a term. Then $\iota\gamma(f(t_1, \ldots, t_n)) = \iota\gamma_f(\gamma(t_1), \ldots, \gamma(t_n)) = \iota\gamma_f \circ (\iota^{\times n})^{-1}(\iota\gamma(t_1), \ldots, \iota\gamma(t_n)) = \gamma_f^\circ(\iota\gamma(t_1), \ldots, \iota\gamma(t_n))$, so that the bijection $\iota\gamma : H \to N$ is in PR_L. In order to show that the inverse of this bijection is the restriction of a map from PR_L we need a Lemma:

Lemma 1 (Definition by cases): *Let $t_1, \ldots, t_{m-1}$ be distinct terms and let $\varphi_1(\bar{y}), \ldots, \varphi_{m-1}(\bar{y})$ and $\varphi_m(x, \bar{y})$ be functions of PR_L. Then the function ψ, defined by*

$$\psi(x, \bar{y}) := \begin{cases} \varphi_i(\bar{y}) & \text{if } x = t_i \\ \varphi_m(x, \bar{y}) & \text{else} \end{cases} \tag{9}$$

is also in PR_L.

Proof: We may assume that $m = 2$. (The general case follows by an obvious induction). It follows by (vi), that in PR_L there is a function $\chi(z, x, \bar{y})$ with $\chi(0, x, \bar{y}) = \varphi_1(\bar{y})$ and $\chi(0', x, \bar{y}) = \varphi_2(x, \bar{y})$. Now let $\rho : N \to N$ be the restriction of a function of PR_L with $\rho(\iota\gamma(t_1)) := 0$ and $\rho(x) := 0'$ for $x \in N$ and $x \neq \iota\gamma(t_1)$. We get $\psi(x, \bar{y}) = \chi(\rho\iota\gamma(x), x, \bar{y})$, so that ψ is in PR_L. □

Lemma 2: *There is a function $\eta : H \to H$ in PR_L such that $\eta\restriction N$ is the inverse of $\iota\gamma$.*

Proof: First we assume that L contains a function-symbol q with arity greater than 1. Then we may assume that there is a 2-ary function-symbol p; (otherwise we consider $p(x, y) := q(x, y, 0, \ldots, 0)$ as an abbreviation). We can use p to encode lists: $p(s, t)$ encodes the list with head s and tail t. There is a 2-ary function $[x]_i$ in PR_L with $[p(x, y)]_0 := x$ and $[p(x, y)]_{i'} := [y]_i$. Let f_i, π_i and n_i be as in (ix) and (x). Define $s \dot{-} t$ to be the difference between s and t when they are both in N and $s \geq t$, and $s \dot{-} t := 0$ otherwise. Then $\dot{-}$ is in PR_L. By Lemma 1, the function $\lambda(i, x_1, \ldots, x_n, a, l) := f_i([l]_{a \dot{-} x_1}, \ldots, [l]_{a \dot{-} x_{n_i}})$ is in PR_L. Now there is a function μ in PR_L with $\mu(0) := 0$ and $\mu(x') := p\Big(\lambda\Big(\pi_0(x), \pi_1(x), \ldots \pi_n(x), x, \mu(x)\Big), \mu(x)\Big)$. If x is in N, then $\mu(x)$ can be considered to be the list $((\iota\gamma)^{-1}(x \dot{-} 1), \ldots, (\iota\gamma)^{-1}(0))$, so we may set $\eta(x) := [\mu(x')]_0$ and the Lemma is proved for this case.

We now assume that every function-symbols of L is 1-ary. There is a primitive-recursive function $\rho : \mathbb{N}^2 \to \mathbb{N}$ with $\rho\Big(i, \gamma\Big(f_{m_j} \ldots f_{m_1}(f_{m_0})\Big)\Big) := m_i$ if $i \leq m_j$, and $\rho(i, x) := 0$ otherwise. Now use Lemma 1 to define $\xi : H^2 \to H$ by $\xi(0, r) := f_{\rho(0,r)}$ and $\xi(i', r) := f_{\rho(i',r)}(\xi(i, r))$ if $\rho(i', r) \neq 0$ and $\xi(i', r) := \xi(i, r)$ otherwise. We set $\eta(r) := \xi(r, r)$. □

We now obtain the desired internal description of primitive recursive functions and sets:

Theorem 3: *A function $\varphi : H^n \to H$ is primitive-recursive iff it is in PR_L. A subset $U \subset H^n$ is a primitive-recursive set iff there is a function $\xi : H^n \to H$ in PR_L such that $U = \{\bar{x} \in H \mid \xi(\bar{x}) = 0\}$.*

Proof: We have already mentioned that every function of PR_L is primitive-recursive. We now assume that φ^* is the restriction of a primitive-recursive function. Then, by Lemma 2, we get that $\varphi = \gamma^{-1}\varphi^*\gamma = \gamma^{-1}\iota^{-1}\varphi^{*\circ}\iota\gamma = \eta\varphi^{*\circ}(\iota\gamma)$ is in PR_L. The second part of the theorem follows immediately from the first part. □

3 Ackermann-Predicates

In this section L consists of the constant 0 and the 1-ary function-symbol $'$. We identify N and $\mathbb{N}$ with ι, so that $H = N = \mathbb{N}$. The programs A_m consists of the following clauses.

$$
\begin{aligned}
\mathtt{A}_m(x_m, \dots, x_2, x_1', z) &\leftarrow \mathtt{A}_m(x_m, \dots, x_2, x_1, z') && (10)\\
\mathtt{A}_m(x_m, \dots, x_3, x_2', 0, z) &\leftarrow \mathtt{A}_m(x_m, \dots, x_3, x_2, z', z') && (11)\\
&\vdots\\
\mathtt{A}_m(x_m, \dots, x_{i+1}, x_i', 0, \dots, 0, z) &\leftarrow \mathtt{A}_m(x_m, \dots, x_i, z', 0, \dots, 0, z') && (12)\\
&\vdots\\
\mathtt{A}_m(x_m', 0, \dots, 0, z) &\leftarrow \mathtt{A}_m(x_m, z', 0, \dots, 0, z'). && (13)
\end{aligned}
$$

These programs are tame. For every tuple $(t_m, \dots, t_0)$ there is precisely one term u with $\mathsf{A}_m \vdash \mathtt{A}_m(t_m, \dots, t_0) \leftarrow \mathtt{A}_m(0, \dots, 0, u)$. We define $\alpha_m(t_m, \dots, t_0) := u$. The functions α_m satisfy the following equations.

$$
\begin{aligned}
\alpha_m(0, \dots, 0, x) &= x && (14)\\
\alpha_m(x_m, \dots, x_2, x_1 + 1, z) &= \alpha_m(x_m, \dots, x_2, x_1, z + 1) && (15)\\
\alpha_m(x_m, \dots, x_3, x_2 + 1, 0, z) &= \alpha_m(x_m, \dots, x_3, x_2, z + 1, z + 1) && (16)\\
&\vdots\\
\alpha_m(x_m, \dots, x_{i+1}, x_i + 1, 0, \dots, 0, z) &= \alpha_m(x_m, \dots, x_i, z + 1, 0, \dots, 0, z + 1) && (17)\\
&\vdots\\
\alpha_m(x_m + 1, 0, \dots, 0, z) &= \alpha_m(x_m, z + 1, 0, \dots, 0, z + 1). && (18)
\end{aligned}
$$

The functions α_m are uniquely defined by these equations. It follows that they are strictly increasing in all arguments. The following equations also hold:

$$
\begin{aligned}
\alpha_1(x, y) &= x + y && (19)\\
\alpha_2(x, y, z) &= 2^x(y + z + 2) - 2 && (20)\\
\alpha_m(0, x_{m-1}, \dots, x_0) &= \alpha_{m-1}(x_{m-1}, \dots, x_0) && (21)\\
\alpha_m(x_m, \dots, x_0) &= \alpha_m(x_m, \dots, x_{i+1}, 0, \dots, 0, \alpha_i(x_i, \dots, x_0)). && (22)
\end{aligned}
$$

Let $\beta_m(x, y) := \alpha_m(x, 0, \dots, 0, y) = \alpha_m(x, 0, \dots, 0, y, 0)$. Then

$$
\begin{aligned}
\beta_1(x, y) &= x + y && (23)\\
\beta_m(0, y) &= y && (24)\\
\beta_m(x + 1, y) &= \beta_m(x, \beta_{m-1}(y + 1, y + 1)) && (25)
\end{aligned}
$$

If we set $\delta(y) := \beta_{m-1}(y + 1, y + 1)$, then (25) becomes $\beta_m(x + 1, y) = \beta_m(x, \delta(y))$, hence $\beta_m(x, y) = \delta^x(y)$. It follows that the functions β_m are primitive-recursive.

It follows from $\alpha_m(x_m, \ldots, x_0) = \beta_m(x_m, \alpha_{m-1}(x_{m-1}, \ldots, x_0))$ that the α_m are primitive-recursive too.

$\beta_m(x, y)$ is a variant of the Ackermann-function. In fact it mayorizes the Ackermann-function. So given any primitive-recursive function $\varphi : \mathbb{N} \to \mathbb{N}$ there is a m with $\beta_m(1, x) > \varphi(x)$ for every $x \in \mathbb{N}$. Also, for any primitive-recursive function $\varphi : \mathbb{N}^n \to \mathbb{N}$ there is an m with $\alpha_m(1, 0, \ldots, 0, x_1, \ldots, x_n) > \varphi(x_1, \ldots, x_n)$.

4 "Tame" implies "Primitive-Recursive"

We first show that it suffices to consider programs with only one predicates. This is in fact nothing else than a variant of the transitivity theorem for inductive definitions ([4], Theorem 1C.3). Let T be a L-program with predicates $\mathtt{P}_1, \ldots, \mathtt{P}_s = \mathtt{T}$, where the predicates are ordered by $\mathtt{P}_i < \mathtt{P}_j \;:\Longleftrightarrow\; i < j$. We choose a constant $0 \in L$ and variable-free terms $l_1, \ldots, l_s$ with $|l_1| < \cdots < |l_s|$. Let $\mathtt{Q}$ be a new predicate whose arity is greater than all the arities of the predicates of T. Given an atomic formula $\Pi = \mathtt{P}_i(t_1, \ldots, t_{n_j})$ we construct a formula $\tilde{\Pi} := \mathtt{Q}(l_i; t_1, \ldots, t_{n_i}, 0, \ldots, 0)$. We attach to T a program $\tilde{\mathsf{T}}$ by replacing all atomic formulas Π contained in T by $\tilde{\Pi}$. Then a tupel $(t_1, \ldots, t_n) \in H^n$ is in the set computed by T iff $(l_s; t_1, \ldots, t_n, 0, \ldots, 0)$ is in the set computed by $\tilde{\mathsf{T}}$. If T is tame, $\tilde{\mathsf{T}}$ is tame also. So we get:

Lemma 4: *Every subset of H^n computed by a tame program is a section of a set computed by a tame program containing only one predicate.* □

We really do need sections: If $L = \{0, '\}$, then the subsets of $H = N$ computable by a tame program with only one 1-ary predicate are the sets definable in the Pressburger arithmetic. Let Φ be an atomic formula. A *proof-tree for* Φ (with respect to T) is a finite tree $\mathcal{F}$ with the following properties.

(xi) The nodes of $\mathcal{F}$ are atomic formulas.

(xii) The root of $\mathcal{F}$ is Φ.

(xiii) If Ψ is a node of $\mathcal{F}$ and if $\Xi_1, \ldots, \Xi_r$ are its successors, then $\Psi \leftarrow \Xi_1 \wedge \cdots \wedge \Xi_r$ is a substitution-instance of a clause of T.

Let $\mathbb{A}$ be the set of all variable-free atomic formulas and $\mathbb{T}$ the finite set of all proof-trees with respect to T. The map $\Theta_{\mathsf{T}} : \mathbb{A} \times \mathbb{N} \to \mathcal{P}_\omega(\mathbb{T})$, moving a pair (Ψ, m) to the set of all proof-trees for Ψ with a depth not greater than m, is primitive-recursive.

Lemma 5: *Tame programs compute primitive-recursive sets.*

Proof: Let T be a tame program. By Lemma 4, we may assume that T contains only one predicate $\mathtt{T}$, because sections of primitive-recursive sets are primitive-recursive themselves. Choose $p \in \mathbb{N}$ such that for any term t occuring in the body of some clause of T, every coefficient of $|t|$ is smaller than p. Let n be the arity of $\mathtt{T}$. Now for $(t_1,\dots,t_n) \in H^n$, we set $\|t_1,\dots,t_n\| := \alpha_{2n+1}(|t_1|,\dots,|t_n|,|t_1|,\dots,|t_n|,p,0) \in \mathbb{N}$. Let $\mathtt{T}(t_1,\dots,t_n)$ be a node of a proof-tree and $\mathtt{T}(s_1,\dots,s_n)$ a successor of this node. For each i, we have $p + p\sum_{j=1}^n |t_j| > |s_i|$; as T is tame there is a $m \le n$ such that $|t_i| \ge |s_i|$ for $i < m$ and $|t_m| > |s_m|$. It follows that

$$\begin{aligned}
\|t_1,\dots,t_n\| &= \alpha_{2n+1}(|t_1|,\dots,|t_n|,|t_1|,\dots,|t_n|,p,0)\\
&\ge \alpha_{2n+1}(|t_1|,\dots,|t_n|,0,\dots,0,\sum_{j=1}^n |t_j|,p,0)\\
&\ge \alpha_{2n+1}(|t_1|,\dots,|t_n|,0,\dots,0,p+p\sum_{j=1}^n |t_j|)\\
&= \alpha_{2n+1}(|t_1|,\dots,|t_{m-1}|,|t_m|-1, \\
&\qquad 1+p+p\sum_{j=1}^n |t_j|,0,\dots,0,1+p+p\sum_{j=1}^n |t_j|) \qquad (26)\\
&> \alpha_{2n+1}(|t_1|,\dots,|t_{m-1}|,|t_m|-1,p+p\sum_{j=1}^n |t_j|,\dots,p+p\sum_{j=1}^n |t_j|)\\
&> \alpha_{2n+1}(|s_1|,\dots,|s_n|,|s_1|,\dots,|s_n|,p,0)\\
&= \|s_1,\dots,s_n\|.
\end{aligned}$$

So the depth of a proof-tree for $\mathtt{T}(t_1,\dots,t_n)$ is bounded by $\|t_1,\dots,t_n\|$, and hence

$$\mathsf{T} \vdash \mathtt{T}(t_1,\dots,t_n) \iff \exists m \le \|t_1,\dots,t_n\| \, (\Theta_{\mathsf{T}}(\mathtt{T}(t_1,\dots,t_n),m) \neq \emptyset). \qquad (27)$$

The existential quantor in the formula above is bounded, so the set on the right-hand side is primitive-recursive. □

5 Computation-Power of Tame Programs

In this section we show that all primitive predicates can be computed by tame predicates. We again embed the natural numbers in the Herbrand-universe by means of $\iota : \mathbb{N} \to N \subset H$ (as in the second section). N can be computed by the following tame program:

$$\mathtt{N}(0). \qquad (28)$$
$$\mathtt{N}(x') \leftarrow \mathtt{N}(x). \qquad (29)$$

We show in the next two Lemmas, that the graphs of some functions can be computed by tame programs.

Lemma 6: *For every primitive-recursive function $\varphi : \mathbb{N}^n \to \mathbb{N}$ there is a tame program, computing the graph of φ°.*

Proof: The computability of the graph of a projection, of a constant function or of the successor-function is trivial.

Before we treat the composition of function and the schema of primitive-recursion, we consider the Ackermann-predicates again. Let A_m^r be the program we obtain from A_m by replacing $\mathtt{A}_m(t_m, \ldots, t_0)$ by $\mathtt{A}_m^r(y_1, \ldots, y_r; t_m, \ldots, t_0)$. The new parameters have no influence on a computation.

Let $\varphi : \mathbb{N} \to \mathbb{N}$ and $\psi : \mathbb{N} \to \mathbb{N}$ be primitive recursive functions, and let F and P be tame programs, computing the graphs of φ° and ψ°. There is an $m \in \mathbb{N}$ such that $\psi(x) < \beta_m(1, x)$ for every $x \in \mathbb{N}$. Let K be the program containing $\mathsf{F}, \mathsf{P}, \mathsf{A}_m^2$ and the following clauses.

$$\mathtt{K}(x, z) \leftarrow \mathtt{A}_m^2(x, z; 1, 0, \ldots, 0, x, 0). \tag{30}$$

$$\mathtt{A}_m^2(x, z; y_m, \ldots, y_0) \leftarrow \mathtt{P}(x, y_0) \wedge \mathtt{F}(y_0, z). \tag{31}$$

We may assume w.l.o.g. that the programs F and P neither have a predicate in common nor does one of them contain $\mathtt{A}_m^2$ or $\mathtt{K}$ (otherwise we must rename these predicates). Then K is a tame program; we claim that K computes the graph of $\varphi \circ \psi$. We apply this program to $\mathtt{K}(x, z)$; this resolves first to $\mathtt{A}_m^2(x, z; 1, 0, \ldots, 0, x, 0)$; then, "running A_m^2", it resolves to $\mathtt{A}_m^2(x, z; t_m, \ldots, t_1, \psi(x))$. So $\mathsf{K} \vdash \mathtt{K}(x, z)$ iff $z = \varphi\psi(x)$. A similar proof works for functions with several variables.

It remains to prove that the graph of functions defined with primitive-recursion can be represented. Let $\varphi : \mathbb{N}^3 \to \mathbb{N}$ and $\psi : \mathbb{N} \to \mathbb{N}$ again primitive-recursive functions, and let F and P be programs computing the graphs of φ° and ψ°. κ is the function defined by $\kappa(x, 0) := \psi(x)$ and $\kappa(x, i+1) := \varphi(x, \kappa(x, i), i)$. Now choose $m \in \mathbb{N}$ with $\kappa(x, i) < \beta_m(1, x+i)$ for every $x, i \in \mathbb{N}$. Let K be the program containing the programs F, P and $\mathtt{A}_m^3$ and the following clauses.

$$\mathtt{K}(x, i, z) \leftarrow \mathtt{A}_m^3(x, i, z; 1, 0, \ldots, 0, x, i, 0). \tag{32}$$

$$\mathtt{A}_m^3(x, 0, z; y_m, \ldots, y_0) \leftarrow \mathtt{P}(x, z). \tag{33}$$

$$\mathtt{A}_m^3(x, i', z; y_m, \ldots, y_0) \leftarrow \mathtt{A}_m^3(x, i, y_0; 1, 0, \ldots, 0, x, i, 0) \wedge \mathtt{F}(x, y_0, i, z). \tag{34}$$

It is easy to see that K computes the graph of κ. The recursion-schema with parameters can be handled in the same way. $\square$

Lemma 7: *The graph of a primitive-recursive function $\varphi : H^n \to N \subset H$ is computable by a tame program.*

Proof: We first show that the graph of the map $\iota\gamma : H \to N$ is computable by a tame program. For the function-symbol $f \in L$, the map $\gamma_f : \mathbb{N}^{n_f} \to \mathbb{N}$ in (viii) is primitive-recursive and hence by Lemma 6, there are tame programs G_f computing the graphs of the functions γ_f°. Let n be the maximum of all the arities of function-symbols of L. Let G be the program containing the following clauses.

$$\mathtt{G}(x, y) \leftarrow \mathtt{Q}(x, y; \underbrace{y, \ldots, y}_{n \text{ times}}). \tag{35}$$

$$\mathtt{Q}(x,y;y_1',y_2,\ldots,y_n) \leftarrow \mathtt{Q}(x,y;y_1,y_2,\ldots,y_n). \tag{36}$$

$$\vdots$$

$$\mathtt{Q}(x,y;y_1,\ldots,y_{n-1},y_n') \leftarrow \mathtt{Q}(x,y;y_1,\ldots,y_{n-1},y_n). \tag{37}$$

Furthermore for any constant $c \in L$ there is a clause

$$\mathtt{Q}(c,\iota\gamma(c);y_1,\ldots,y_n). \tag{38}$$

and for every function-symbol f there is a clause

$$\mathtt{Q}(f(x_1,\ldots,x_{n_f}),y;y_1,\ldots,y_n) \leftarrow \mathtt{G}_f(y_1,\ldots,y_{n_f};y) \wedge \bigwedge_{i=1}^{n_f} \mathtt{Q}(x_i,y_i;y_i,\ldots,y_i) \tag{39}$$

By (vii), γ is increasing, so it is easy to see that G is a tame program, computing the graph of $\iota\gamma$.

Now let $\varphi : H \to N$ be a primitive-recursive function. Then, by Lemma 2, the function $\varphi^\star := \varphi\eta : N \to N$ is primitive-recursive, and so it follows by Lemma 6 that its graph is computable by a tame program F. Because of $\varphi = \varphi^\star\iota\gamma$ we must construct a program K for the composition of F and G; the construction is similar to the construction in the proof of Lemma 6.

Let $t \in H$ be a term. Then the *depth* $\delta(t)$ of t is defined recursively by $\delta(c) := 1$ for constants c and $\delta(f(t_1,\ldots,t_n)) := 1 + \max(\delta(t_1),\ldots,\delta(t_n))$. As $\iota\gamma$ is primitive-recursive, there exists an m such that $\iota\gamma(t) < \beta_m(1,\delta(t))$ for all $t \in H$. K is the program, containing the programs F, G, the program A^4_m defined in the proof of Lemma 6 and the following clauses.

$$\mathtt{K}(x,z) \leftarrow \mathtt{A}^4_m(x,z,x,0';0,\ldots,0). \tag{40}$$

$$\mathtt{A}^4_m(x,z,f(u_1,\ldots,u_{n_f}),v;0,\ldots,0) \leftarrow \mathtt{A}^4_m(x,z,u_i,v';0,\ldots,0). \tag{41}$$

$$\mathtt{A}^4_m(x,z,c,v';0,\ldots,0) \leftarrow \mathtt{A}^4_m(x,z,0,0;1,0,\ldots,0,v',0). \tag{42}$$

$$\mathtt{A}^4_m(x,z,0,0;t_m,\ldots,t_0) \leftarrow \mathtt{G}(x,t_0) \wedge \mathtt{F}(t_0,z). \tag{43}$$

(For each function-symbol f and for each $i \leq n_f$, there is a clause of the form (41), and for every constant there is a clause of the form (42).) Again we may assume that $\mathtt{K}$ and $\mathtt{A}^4_m$ are not contained in the programs F and G and we may assume that these two programs have no common predicate. Then K is tame. If we apply this program to $\mathtt{K}(x,z)$ then it resolves first to $\mathtt{A}^4_m(x,z,c,\delta(x);0,\ldots,0)$ if resolving with the clause (41) the right i is always chosen. Furthermore this resolve to $\mathtt{A}^4_m(x,z,0,0;t_m,\ldots,t_0)$, where t_0 can be any element from m with $t_0 \leq \beta_m(1,\delta(x))$; so it resolves to $\mathtt{A}^4_m(x,z,0,0;t_m,\ldots,t_1,\iota\gamma(x))$, and so K resolves to true iff $\iota\gamma(x) = z$. For $\varphi : H^n \to N$, the proof is similar. □

Now we can prove the theorem announced:

Theorem 8: *Tame programs compute primitive-recursive sets and every primitive-recursive set is computed by a tame program.*

Proof: Lemma 5 says that sets computed by tame programs are primitive-recursive. So let $U \subset H^n$ be a primitive-recursive set. Let $\xi : H^n \to H$ be a primitive-recursive function with $U = \{x \in H \mid \xi(x) = 0\}$. Then the graph of ξ is computable by a tame program, and so U is computable by a tame program to. $\square$

6 Complements

The class of primitive-recursive sets is closed under Boolean operations. So the class of sets computable by tame programs is also closed under Boolean operations. For union and intersection this is clear, but it is not so obvious for the complement without using Theorem 8. Given a tame program P, we shall construct a program $\bar{\mathsf{P}}$ computing the complement of the set computed by P.

We start with the following program $\mathtt{Eq}$ for equality containing a clause

$$\mathtt{Eq}(c, c). \qquad \text{for every constant } c \text{ and} \tag{44}$$

$$\mathtt{Eq}(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n)) \leftarrow \mathtt{Eq}(x_1, y_1) \wedge \cdots \wedge \mathtt{Eq}(x_n, y_n). \tag{45}$$

for every function symbol f. A tame program P is called *free* if the following hold:

(xiv) The variables in the head of a clause of P are all distinct.

(xv) If $\mathtt{Q}(t_1, \ldots, t_n)$ and $\mathtt{Q}(s_1, \ldots, s_n)$ are the heads of two clauses of P, then either there is an $i \leq n$ such that t_i and s_i are not unifiable, or the two heads are equal.

(xvi) If $\mathtt{Q}$ is a predicate occuring in P and if $\mathtt{Q}(t_1, \ldots, t_n)$ is a variable-free atomic formula, then there is a clause in P whose head is unifiable with $\mathtt{Q}(t_1, \ldots, t_n)$.

We remark that the program $\mathtt{eq}$ is free (in contrast to the usual way of defining equality by $\mathtt{Equal}(x, x)$.). We call two programs P, Q *equivalent* if they compute the same sets. We shall show that an equivalent free tame program can be constructed for every tame program. We begin with two lemmas:

Lemma 9: *For every tame program* P*, an equivalent tame program satisfying* (xiv) *of the definition above can be constructed effectively.*

Proof: Let

$$\mathtt{R}(t_1, \ldots, t_n) \leftarrow \bigwedge_{i=1}^{r} \mathtt{R}(t_1^i, \ldots, t_n^i) \wedge \Psi \tag{46}$$

be a clause, where Ψ is a formula not containing $\mathtt{R}$, and assume that the variable x occurs in t_u and t_v (with $u \neq v$). Let y be a new variable. If in P we replace the clause above by $R(t_1, \ldots, t_{v-1}, t_v\left[\frac{y}{x}\right], t_{v+1}, \ldots, t_n) \leftarrow \mathtt{Eq}(x, y) \wedge$

$\bigwedge_{i=1}^{r} R(t_1^i, \ldots, t_{v-1}^i, t_v \left[\frac{y}{x}\right], t_{v+1}, \ldots, t_n^i) \wedge \Psi$, then we obtain a tame program computing the same set as P. ($t \left[\frac{y}{x}\right]$ is the term we get from t by substituting x by y.)

By iterated application of the above argument, we may assume that in no clauses (46) of P do the terms t_i and t_j have any variables in common ($i \neq j$). We now assume that the variable x occurs v times in the term t_k of (46). Then we replace these v occurrences in t_k by the new variables $x_1, \ldots, x_v$. If $|t_k| \geq |t_k^j|$, then x occurs not more than v times in t_k^j; in this case we replace all the occurrences of x in $|t_k^j|$ by different x_i. All occurrences of x in (46) not contained in such a t_k^i we replace by an arbitrary x_i. Finally we add $\mathtt{Eq}(x_1, x_2) \wedge \mathtt{Eq}(x_1, x_3) \wedge \cdots \wedge \mathtt{Eq}(x_1, x_v)$ to the body of the modified clause (46). If we do this for every k and every clause, then we get a tame program that computes the same set as P and satisfies (xiv). $\square$

Lemma 10: *Let $t_1, \ldots, t_v$ be terms. Assume that for every i the variables contained in t_i are distinct. Then there is a finite set of terms $\{s_1, \ldots, s_w\}$ with the following properties.*

(xvii) *The variables in a s_i are all distinct.*

(xviii) *Every closed term is unifiable with exactly one s_i.*

(xix) *Each s_i is a substitution instance of a t_j.*

(xx) *If s_i and t_j are unifiable, then s_i is substitution instance of t_j.*

Proof: Let $\mathcal{L}_0 := \{x_i \mid i \in \mathbb{N}\}$ be a set of variables. We define $\mathcal{L}_{m+1}$ recursively to be the sets of all constants $c \in L$ and all terms of the form $f(t_1, \ldots, t_n)$ where $f \in L$ and $t_i \in \mathcal{L}$. Let $\hat{\mathcal{L}}_m$ be the subset of $\mathcal{L}_m$ consisting of the terms t with the following properties.

(xxi) The variables in t are all distinct.

(xxii) If x_i occurs in t and $j < i$ then x_j occurs in t too. Furthermore, t_j stands on the left side of x_i.

Then the sets $\hat{\mathcal{L}}_j$ satisfy (xvii) and (xviii). If j is large enough for all t_i to be contained in $\mathcal{L}_j$, then $\{s_1, \ldots, s_w\} := \hat{\mathcal{L}}_j$ also satisfies (xix) and (xx). $\square$

We take a tame program P satisfying (xiv). Let Ξ be a clause of P and t a term whose variables are all distinct and not contained in Ξ. If we substitute an arbitrary variable of Ξ by t, we again obtain a tame program satisfying (xiv). We now are ready to proof the announced proposition:

Proposition 11: *An equivalent free tame program can be effectively constructed for any tame program* P.

Proof: By Lemma 9, we may assume that P satisfies (xiv). For every predicate $\mathtt{Q}$ occuring in P we add the clause $\mathtt{Q}(x_1, \ldots, x_n) \leftarrow \perp$ to P (where $\perp$ stands for the falsum). Let $\mathtt{Q}(t_1^i, \ldots, t_n^i) \leftarrow \Xi^i$ be clauses of P, where $i = 1, \ldots, v$. We fix $k \leq n$

and choose $s_1, \ldots, s_w$ satisfying (xvii) to (xx) with respect to $t_k^1, \ldots, t_k^v$. We now replace a clause $\mathsf{Q}(t_1^i, \ldots, t_n^i) \leftarrow \Xi^i$ by all of its substitution instances transforming t_k^i in a s_l and not changing the terms t_j^i for $j \neq k$. We repeat this for every k. Then these replacements leed to a free tame program equivalent to the original one. $\square$

With this preparation, the construction of a program $\bar{\mathsf{P}}$ computing the complement of the set computed by P is easy:

Proposition 12: *If P is a tame program, then a tame program $\bar{\mathsf{P}}$ computing the complement of the set computed by P can be constructed effectively.*

Proof: By Proposition 11, we may assume that P is a free tame program. Let

$$\Phi \leftarrow \Psi_{i,1} \wedge \cdots \wedge \Psi_{i,m_i} \qquad i = i, \ldots, k \tag{47}$$

be all the clauses of P with head Φ. Then we add to $\bar{\mathsf{P}}$ the clauses

$$\bar{\Phi} \leftarrow \bar{\Psi}_{1,\sigma(1)} \wedge \cdots \wedge \bar{\Psi}_{k,\sigma(k)} \tag{48}$$

where σ runs over all maps $\{1, \ldots, k\} \to \mathbb{N} \setminus \{0\}$ with $\sigma(i) \leq m_i$. By induction on the ordering (6) defined in the first section and by de Morgan's laws it follows, that for a variable-free formula $\mathsf{Q}(t_1, \ldots, t_n)$ the following equivalence holds: $\bar{\mathsf{P}} \vdash \bar{\mathsf{Q}}(t_1, \ldots, t_n) \iff \mathsf{P} \not\vdash \mathsf{Q}(t_1, \ldots, t_n)$. $\square$

References

[1] E. Börger, *Berechenbarkeit, Komplexität, Logik,* Vieweg, Braunschweig, 1986.

[2] S. Feferman, *Finitary inductively presented logics,* in Logic Colloquium '88, North-Holland, Amsterdam, 1989, 191–220.

[3] J. W. Lloyd, *Foundations of Logic Programming,* Springer-Verlag, Heidelberg, 1987.

[4] Y. N. Moschovakis, *Elementary Induction on Abstract Structures,* North-Holland, Amsterdam, 1974.

[5] A. R. Meyer and D. M. Ritchie, *The complexity of loop programs,* in Proc. of the ACM 22nd National conference, Thomson Book Co., Washington D. C., 1967, 465–469.