ГОДИШНИК НА СОФИЙСКИЯ УНИВЕРСИТЕТ „СВ. КЛИМЕНТ ОХРИДСКИ"

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА
Книга 1 — Математика
Том 87, 1993

ANNUAIRE DE L'UNIVERSITE DE SOFIA „ST. KLIMENT OHRIDSKI"

FACULTE DE MATHEMATIQUES ET INFORMATIQUE
Livre 1 — Mathématiques
Tome 87, 1993

ON THE DETECTION OF SOME LOOPS IN RECURSIVE COMPUTATIONS*

DIMITER SKORDEV

Димитр Скордев. ОБ ОБНАРУЖЕНИИ НЕКОТОРЫХ ЗАЦИКЛИВАНИЙ В РЕКУРСИВНЫХ ВЫЧИСЛЕНИЯХ

В настоящей работе рекурсивные вычисления трактуются как некоторая детерминированная форма переработки термов. Термы строятся обычным способом из атомов при помощи функциональных символов. Интуитивно атомы трактуются как константы. Простой неатомарный терм — это терм, получающийся, когда некоторый функциональный символ снабжается аргументами, являющимися атомами. Предполагается, что дано некоторое правило рекурсии в форме отображения множества простых неатомарных термов в множество всех термов. Рекурсивное вычисление — это процес конструирования термов, начиная с некоторого данного терма и заменяя, пока возможно, самое левое вхождение простого неатомарного терма в текущий терм на терм, соответствующий согласно данному правилу рекурсии. Предлагается метод в стиле Брента — Ван Гельдера для обнаружения случая, когда некоторый простой неатомарный терм воспроизводит себя снова в самом левом положении после положительного числа шагов в рекурсивном вычислении.

Dimiter Skordev. ON THE DETECTION OF SOME LOOPS IN RECURSIVE COMPUTATIONS

In the present paper recursive computations are treated as a certain deterministic kind of term processing. The terms are built up in the usual way from atoms by means of function symbols. The atoms are intuitively viewed as constants. Simple non-atomic terms are those ones

* Research partially supported by the Ministry of Science and Higher Education, Contract MM 43/91.

constructed by supplying some function symbol with arguments which are atoms. A recursion rule is supposed to be given, considered as a mapping of the set of the simple non-atomic terms into the set of all terms. A recursive computation is a process of constructing terms by starting with some given term and, while possible, replacing the leftmost occurrence of a simple non-atomic term in the current term by the corresponding term according to the given recursion rule. A Brent—Van Gelder's style method is proposed for detection of the case when some simple non-atomic term reproduces itself again in a leftmost position after a positive number of steps in a recursive computation.

1. RECURSIVE COMPUTATIONS AND CYCLIC LOOPS IN THEM

We consider terms built up in the usual way from atoms by means of function symbols. The atoms are intuitively viewed as constants. The set of the atoms will be denoted by $\mathcal{A}$, and the set of the function symbols will be denoted by $\mathcal{F}$. Let $\mathcal{U}$ (the computational universe) be the set of all terms. Let $\mathcal{U}_1$ be the set of all terms of the form $f(\alpha_1, \ldots, \alpha_n)$, where f is some n-ary symbol from $\mathcal{F}$ and $\alpha_1, \ldots, \alpha_n$ are atoms. The elements of $\mathcal{U}_1$ will be called *simple non-atomic terms*. Each mapping of $\mathcal{U}_1$ into $\mathcal{U}$ will be called a *recursion rule*. An example follows which illustrates the intuition behind this convention.

Example 1. Let $\mathbb{N}$ be the set of all non-negative integers (to be called further *natural numbers*). Consider the least defined two-argument partial function φ in $\mathbb{N}$ satisfying the following conditions for all y and z in $\mathbb{N}$:

$$\varphi(2z, y) = z + 1, \quad \varphi(2z + 1, y) = \varphi(\varphi(z, y), \varphi(y, z)).$$

We shall relate to this function definition a recursion rule in the introduced sense. Let $\mathcal{A}$ consist of the usual decimal denotations of the individual natural numbers, and $\mathcal{F}$ consist of a two-argument function symbol f. Identifying the decimal denotations of the natural numbers with the numbers themselves, we define a mapping $\mathcal{D}$ of $\mathcal{U}_1$ into $\mathcal{U}$ in the following way[1]:

$$\mathcal{D}(\mathrm{f}(2z, y)) = z + 1, \quad \mathcal{D}(\mathrm{f}(2z + 1, y)) = \mathrm{f}(\mathrm{f}(z, y), \mathrm{f}(y, z)).$$

For example, we shall have

$$\mathcal{D}(\mathrm{f}(6, 5)) = 4, \quad \mathcal{D}(\mathrm{f}(13, 11)) = \mathrm{f}(\mathrm{f}(6, 11), \mathrm{f}(11, 6)).$$

Turning back to the general situation, we note that each element u of $\mathcal{U} \setminus \mathcal{A}$ has at least one subterm belonging to $\mathcal{U}_1$, and there is a uniquely determined leftmost occurrence of a term from $\mathcal{U}_1$ in u. Replacing this occurrence by any other term will produce again an element of $\mathcal{U}$. From now on, a recursion rule $\mathcal{D}$ will be supposed to be given. We extend $\mathcal{D}$ to $\mathcal{U} \setminus \mathcal{A}$ in the following way: if u is an arbitrary non-atomic term, then we set $\mathcal{D}(u)$ to be the result of replacing the leftmost occurrence of a term from $\mathcal{U}_1$ in u by the corresponding image under the original mapping $\mathcal{D}$. If u is an arbitrary term, then there is a maximal sequence (finite or infinite)

[1] The equality sign here and in all further occasions, where terms are concerned, denotes graphic equality of terms.

of the form $\mathfrak{u}$, $\mathcal{D}(\mathfrak{u})$, $\mathcal{D}^2(\mathfrak{u})$, $\mathcal{D}^3(\mathfrak{u})$, ... This sequence will be called *the recursive computation of* $\mathfrak{u}$.

Example 2. In the situation from Example 1 the recursive computation of the simple non-atomic term $f(13,11)$ looks as follows:

$$f(13,11),\quad f(f(6,11),f(11,6)),\quad f(4,f(11,6)),\quad f(4,f(f(5,6),f(6,5))),$$
$$f(4,f(f(f(2,6),f(6,2)),f(6,5))),\quad f(4,f(f(2,f(6,2)),f(6,5))),$$
$$f(4,f(f(2,4),f(6,5))),\quad f(4,f(2,f(6,5))),\quad f(4,f(2,4)),\quad f(4,2),\quad 3.$$

Again in this situation the first five members of the recursive computation of $f(1,0)$ are the following ones:

$$f(1,0),\quad f(f(0,0),f(0,0)),\quad f(1,f(0,0)),\quad f(1,1),\quad f(f(0,1),f(1,0)).$$

This computation is infinite, since its member $f(1,0)$ turns out to be a subterm of a further one.

Remark 1. It can be proved that an ordered pair (x,y) belongs to the domain of the function φ from Example 1 iff the recursive computation of the simple non-atomic term $f(x,y)$ is finite, and in such a case the computation terminates with the value of $\varphi(x,y)$ (hence $\varphi(13,11)$ is equal to 3, and $\varphi(1,0)$ is not defined).

It is natural to be interested in the problem how to decide whether the recursive computation of a given term is finite. Unfortunately, this problem is algorithmically unsolvable even in some relatively simple concrete cases. Therefore we shall consider a certain special sort of infinite recursive computations, such that the infinity of the computation can be effectively detected after making an appropriate finite number of computational steps. We are going now to introduce some denotations and notions which are useful for describing the mentioned sort of infinite recursive computations.

Whenever $\mathfrak{u}$ is a non-atomic term, we shall denote by $\mathcal{H}(\mathfrak{u})$ the leftmost occurring term from $\mathfrak{U}_1$ in $\mathfrak{u}$; this term will be called *the head of* $\mathfrak{u}$.

Example 3. In the situation from Example 1 the equality

$$\mathcal{H}(f(4,f(f(5,6),f(6,5)))) = f(5,6)$$

holds.

The following fact seems to be intuitively obvious and we shall postpone its proof (cf. Appendix 2):

Fact 1. For any term $\mathfrak{u}$ and any natural number t, if $\mathcal{H}(\mathcal{D}^t(\mathcal{H}(\mathfrak{u})))$ makes sense, then $\mathcal{H}(\mathcal{D}^t(\mathfrak{u}))$ also makes sense and the equality

$$\mathcal{H}(\mathcal{D}^t(\mathcal{H}(\mathfrak{u}))) = \mathcal{H}(\mathcal{D}^t(\mathfrak{u}))$$

holds.

If $\mathfrak{u}$ is a term, $\mathfrak{v}$ is a simple non-atomic term, and t is a natural number, then $\mathfrak{u}$ will be said to *activate* $\mathfrak{v}$ *after* t *steps* if $\mathfrak{u}$ belongs to the domain of $\mathcal{D}^t$ and $\mathfrak{v}$ is the head of $\mathcal{D}^t(\mathfrak{u})$.

Example 4. In the situation from Example 1 $f(1,0)$ activates $f(0,1)$ after 4 steps (cf. Example 2).

Remark 2. Another related notion also deserves attention. This is the following interrelation between a term u, a simple non-atomic term v and a natural number t: the term u belongs to the domain of $\mathcal{D}^t$, and v is a subterm of $\mathcal{D}^t(u)$. This requirement is weaker than the requirement u to activate v after t steps (as seen from Example 2, the above requirement is satisfied in the situation from Example 1 for $u = v = f(1,0)$, $t = 4$, but $f(1,0)$ does not activate $f(1,0)$ after 4 steps). The described other notion can be used more or less instead of the notion of activation. However, we prefer to use the notion of activation for reasons of technical convenience.

Fact 1 can be reformulated as follows: for any term u and any natural number t if $\mathcal{H}(u)$ makes sense and activates a certain simple non-atomic term w after t steps, then u also activates w after t steps.

Lemma 1. *Let u be a term, v, w be simple non-atomic terms, s and t be natural numbers, let u activate v after s steps, and v activate w after t steps. Then u activates w after $s + t$ steps.*

Proof. We have the equalities $v = \mathcal{H}(\mathcal{D}^s(u))$, $w = \mathcal{H}(\mathcal{D}^t(v))$. Applying Fact 1 with $\mathcal{D}^s(u)$ in the role of u, we conclude that $\mathcal{D}^s(u)$ belongs to the domain of $\mathcal{D}^t$, and the equality $w = \mathcal{H}(\mathcal{D}^t(\mathcal{D}^s(u)))$ holds, i.e. u belongs to the domain of $\mathcal{D}^{s+t}$ and the equality $w = \mathcal{H}(\mathcal{D}^{s+t}(u))$ holds. ■

A simple non-atomic term v will be said to be *self-reactivating* iff there is a positive integer r such that v activates v after r steps. The following lemma shows a possibility to apply this notion for establishing the infinity of certain recursive computations.

Lemma 2. *Let a term u activate a self-reactivating simple non-atomic term after some number of steps. Then the recursive computation of u is infinite.*

Proof. Let u activate the self-reactivating simple non-atomic term v after s steps, and let v activate v after r steps, where $r > 0$. Then, by Lemma 1, u activates v after $s + kr$ steps for all k in $\mathbb{N}$. Hence u belongs to the domain of $\mathcal{D}^{s+kr}$ for all k in $\mathbb{N}$, i.e. u belongs to the domain of $\mathcal{D}^r$ for arbitrarily large values of r. ■

Example 5. In the situation from Example 1 the simple non-atomic term $f(1,0)$ is self-reactivating. Namely, it activates itself after 5 steps. Indeed, making use of Example 2 we see that

$$\mathcal{D}^5(f(1,0)) = \mathcal{D}(f(f(0,1),f(1,0))) = f(1,f(1,0)).$$

This fact, together with Lemma 2, yields another proof of the statement that the recursive computation of $f(1,0)$ is infinite.

Example 6. Again in the situation from Example 1 the first four members of the recursive computation of $f(1,3)$ are:

$$f(1,3), \quad f(f(0,3),f(3,0)), \quad f(1,f(3,0)), \quad f(1,f(f(1,0),f(0,1))).$$

Hence the term $f(1,3)$ activates the term $f(1,0)$ after 3 steps. Therefore, by Example 5 and Lemma 2, the recursive computation of $f(1,3)$ is infinite. However, the

term f(1, 3) is not self-reactivating. Namely, one can easily show by induction that no member of the recursive computation of f(1, 3) except the first three ones can contain atoms different from 0 and 1.

Example 7. Let us modify the definition of φ and the corresponding recursion rule from Example 1 by writing "$z+9$" instead of "$z+1$". Then the recursion rule will be

$$\mathcal{D}(\mathrm{f}(2z, y)) = z + 9, \quad \mathcal{D}(\mathrm{f}(2z+1, y)) = \mathrm{f}(\mathrm{f}(z, y), \mathrm{f}(y, z)).$$

Under this recursion rule the term f(9, 17) is self-reactivating, but it activates itself only after a relatively large number of steps. Namely, f(9, 17) activates itself after 126 steps and does not activate itself after a smaller positive number of steps. This can be seen by application of a suitable computer program. For example, making use of Turbo Pascal, Version 4.0 or later, we could apply the program from Figure 1[2].

```
{$S+}
var k,l,m,t:word;

function f(x,y:word):word;
var z:word;
begin
 if (x=k) and (y=l) and (t>0) then
  begin
   writeln('f(',k,',',l,') activates itself after ',t,' steps!');
   halt
  end;
 if t<65535 then t:=t+1
            else begin writeln('Too many steps!');halt end;
 z:=x div 2;if odd(x) then f:=f(f(z,y),f(y,z)) else f:=z+9
end;

begin
 t:=0;
 readln(k,l);m:=f(k,l);writeln('f(',k,',',l,')=',m)
end.
```

Figure 1. A program detecting self-reactivation (Example 7)

[2] The program is obtained by adding appropriate side effects to an ordinary recursive Pascal program for computing the value of φ. The side effects in question are carried out through introducing the step counting variable t and through the operators containing it (the fact is used that the compiler implements the recursion present in the program by means of steps corresponding to applications of the mapping $\mathcal{D}$). A more straight-forward approach would lead to explicitly using dynamic data structures for modelling the recursive computation of the considered term. The way of using side effects in the programs given in this paper is similar to a technique used in 1992 by Igor Đurđanović in Paderborn (he implemented then in Prolog certain loop detection methods designed by the present author and concerning Prolog programs).

Remark 3. Let us call a simple non-atomic term ν *self-reproducing* if there is a positive integer r such that ν belongs to the domain of $\mathcal{D}^r$, and ν is a subterm of $\mathcal{D}^r(\nu)$. In general, self-reproduction is a weaker property than self-reactivation, but can be used in a similar way. For example, the assumption of Lemma 2 means that the recursive computation of $\mathfrak{u}$ contains some member whose head is a self-reactivating simple non-atomic term, and this assumption can be weakened in the following way: some member of the recursive computation of $\mathfrak{u}$ has a subterm which is a self-reproducing simple non-atomic term. We shall not make use of such a strengthening of the lemma, since, for reasons of technical convenience, we prefer to use the notion of self-reactivation.

Let $\mathfrak{u}$ be a term. We shall say that *a cyclic loop is present in the recursive computation of* $\mathfrak{u}$ iff $\mathfrak{u}$ satisfies the assumption of Lemma 2, i.e. iff $\mathfrak{u}$ activates some self-reactivating simple non-atomic term after some number of steps.

Example 8. In the situation from Example 1 a cyclic loop is present in the recursive computation of $f(1,0)$, as well as in the recursive computation of $f(1,3)$ (cf. Examples 5 and 6).

Example 9. In the situation from Example 7 a cyclic loop is present in the recursive computation of $f(9,17)$. A cyclic loop is present also in the recursive computation of $f(1,3)$, since $f(1,3)$ activates $f(9,17)$ after 48 steps (this can be shown by using a suitable modification of the program from Figure 1).

By Lemma 2 the presence of a cyclic loop in the recursive computation of a given term implies the infinity of that computation. It is easy to see that the converse is not true in the general case, i.e. the recursive computation of a term could be infinite without the presence of a cyclic loop in that computation. Nevertheless, the presence of a cyclic loop is a frequently encountered cause for infinity of the recursive computation of a term (cf. for example Appendix 1 in this connection). Therefore it could be useful to have some efficient method for the detection of this kind of loops. Such a method will be presented in the next section. The method will combine some features of a loop detection method of R. P. Brent, described in [1, p. 7, Exercise 7] and generalized in [2][3], and of another one proposed by A. Van Gelder in [3, 4][4] (both mentioned methods are intended for the examination of other kinds of computational processes).

2. THE LOOP DETECTION METHOD

We shall describe the method under the same assumptions as in the previous section. In fact, there will be infinitely many variants of the method. Any concrete variant is determined by the choice of a concrete infinite strictly increasing sequence $\tau_0, \tau_1, \tau_2, \ldots$ of natural numbers such that there is no upper bound for the set of

[3] No citation of related work of other authors is given in [2], due to the lack of information in this respect at the moment of writing that paper.

[4] The presentation in [3] is not correct, as shown in [5], but [4] indicates a way for correcting that presentation.

the differences $\tau_{n+1} - \tau_n$, $n = 0, 1, 2, \ldots$ (such sequences are considered in [2]; concordance with the description of Brent's method in [1] can be achieved by setting $\tau_n = 2^n - 1$). To simplify the presentation of the method, we shall restrict ourselves to the case when the equality $\tau_0 = 0$ holds. From now on, a sequence $\tau_0, \tau_1, \tau_2, \ldots$ with the formulated properties is supposed to be given.

To be able to detect the presence of a cyclic loop in the recursive computation of a given term, we add some loop detection activities to the process of constructing consecutive members of the recursive computation of the term. We shall firstly give a somewhat intuitive description of these additional activities and after that we shall describe them in a more formal way.

Roughly speaking, the main additional activity consists in making a snapshot of the head of the current term at certain moments of the computation, the moment τ_0 being the first of them, and comparing the heads of some of the further arising terms with this snapshot. A detailed formulation (although not yet a thoroughly formal one) concerning the comparison will look as follows.

Suppose we study the recursive computation of a given term u. Let a member $w = \mathcal{D}^t(\mathrm{u})$ of the computation be obtained from u by t applications of $\mathcal{D}$ and a snapshot of $\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u}))$ made after $\bar{t}$ applications of $\mathcal{D}$ to u be available, where $\bar{t} < t$. We suppose that no snapshot is made at moments between $\bar{t}$ and t, and $\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u}))$ belongs to the domain of $\mathcal{D}^{t-\bar{t}}$. After w is obtained, one firstly checks whether w belongs to the domain of $\mathcal{D}$. If w does not belong to this domain (i.e. if w is an atom), then the recursive computation of u is completed and nothing more has to be done (of course, there is no cyclic loop in the computation in this case). If w belongs to the domain of $\mathcal{D}$, then certain loop detection activities must be carried out. First of all, one checks whether $\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u})))$ is an atom or not. If it is an atom, then we shall say that *the available snapshot becomes obsolete at the moment* t. In this case one replaces the snapshot of $\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u}))$ by a snapshot of $\mathcal{H}(\mathcal{D}^t(\mathrm{u}))$ and then constructs the next member $\mathcal{D}^{t+1}(\mathrm{u})$ of the computation. Otherwise $\mathcal{H}(w)$ and $\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u}))$ are compared. If they turn out to be equal, then a loop is detected in the computation at the moment t and nothing more has to be done. Otherwise again the next member $\mathcal{D}^{t+1}(\mathrm{u})$ of the computation must be constructed. But before doing this one must check whether t is a member of the sequence $\tau_0, \tau_1, \tau_2, \ldots$, and if t is such a member, to replace again the snapshot of $\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathrm{u}))$ by a snapshot of $\mathcal{H}(\mathcal{D}^t(\mathrm{u}))$.

We hope the next two examples will make the above presentation of the method more intelligible.

Example 10. Suppose we carry out the computation of f(13, 11) in the situation from Example 1 (cf. Example 2). Let the sequence $\tau_0, \tau_1, \tau_2, \ldots$ be determined by the equality $\tau_n = 2^n - 1$, $n = 0, 1, 2, \ldots$ Then the computation process accompanied with loop detection activities can be represented by Figure 2, where making a snapshot is indicated by an asterisk, the corresponding head is printed bold-faced, and the vertical and broken lines indicate the results of successive applications of $\mathcal{D}$ to such a head (do not pay attention to the rightmost two columns of numbers — they will not be used yet!). Note that snapshots at moments 2, 7, 8 and 9

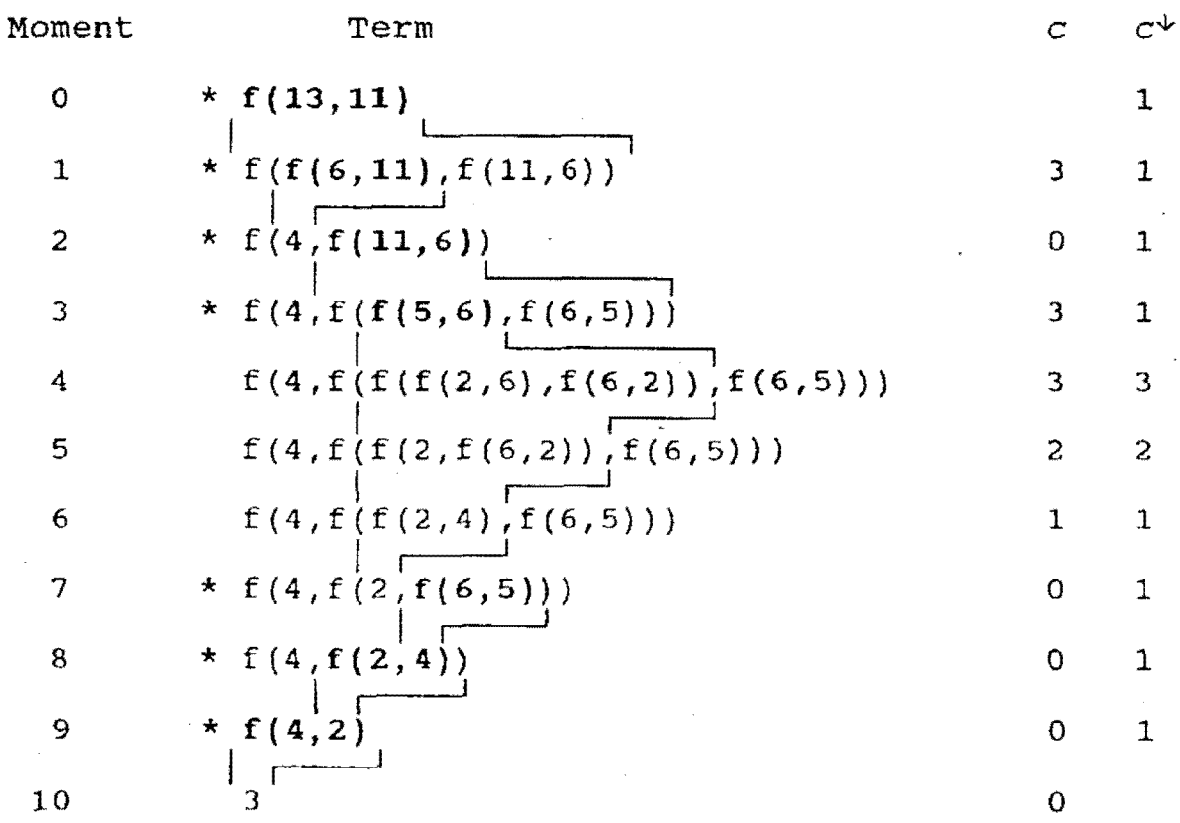

Figure 2. Computation with loop detection activities (Example 10)

are made because the available snapshot becomes obsolete at these moments, and snapshots at moments 0, 1 and 3 are made because these moments are members of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ At the moment 10 the computation is completed, thus its finiteness is seen. Of course, this finiteness has been established much easier in Example 2 without using the loop detection method. In the present example we only observed that the application of the loop detection method to the same computation did not change the final conclusion.

Example 11. Again in the situation from Example 1 let us examine the recursive computation of the term f(1,3) using the same sequence $\tau_0, \tau_1, \tau_2, \ldots$ as in the above example. Figure 3 represents what happens (the same conventions hold as above). At the moments 2 and 8 snapshots are made because the available snapshots become obsolete at these moments, and at the moments 0, 1, 3 and 7 snapshots are made because these moments are members of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ At the moment 13 a cyclic loop is detected in the computation. Note that the loop would be detected earlier (namely, at the moment 8) if we had $\tau_3 \geqq 8$ instead of $\tau_3 = 7$.

The most troublesome thing in the method seems to be the necessity not only to construct successive members of the examined recursive computation, but also to consider results of successive applications of $\mathcal{D}$ to the snapped term heads (in order to check whether the last of these results is an atom or not). In the above two examples the results in question have been visualized by means of some lines. A formal variant of this technique is surely possible, but fortunately there is an way to perform the mentioned check without explicitly indicating the concerned results. We shall explain now that easier way.

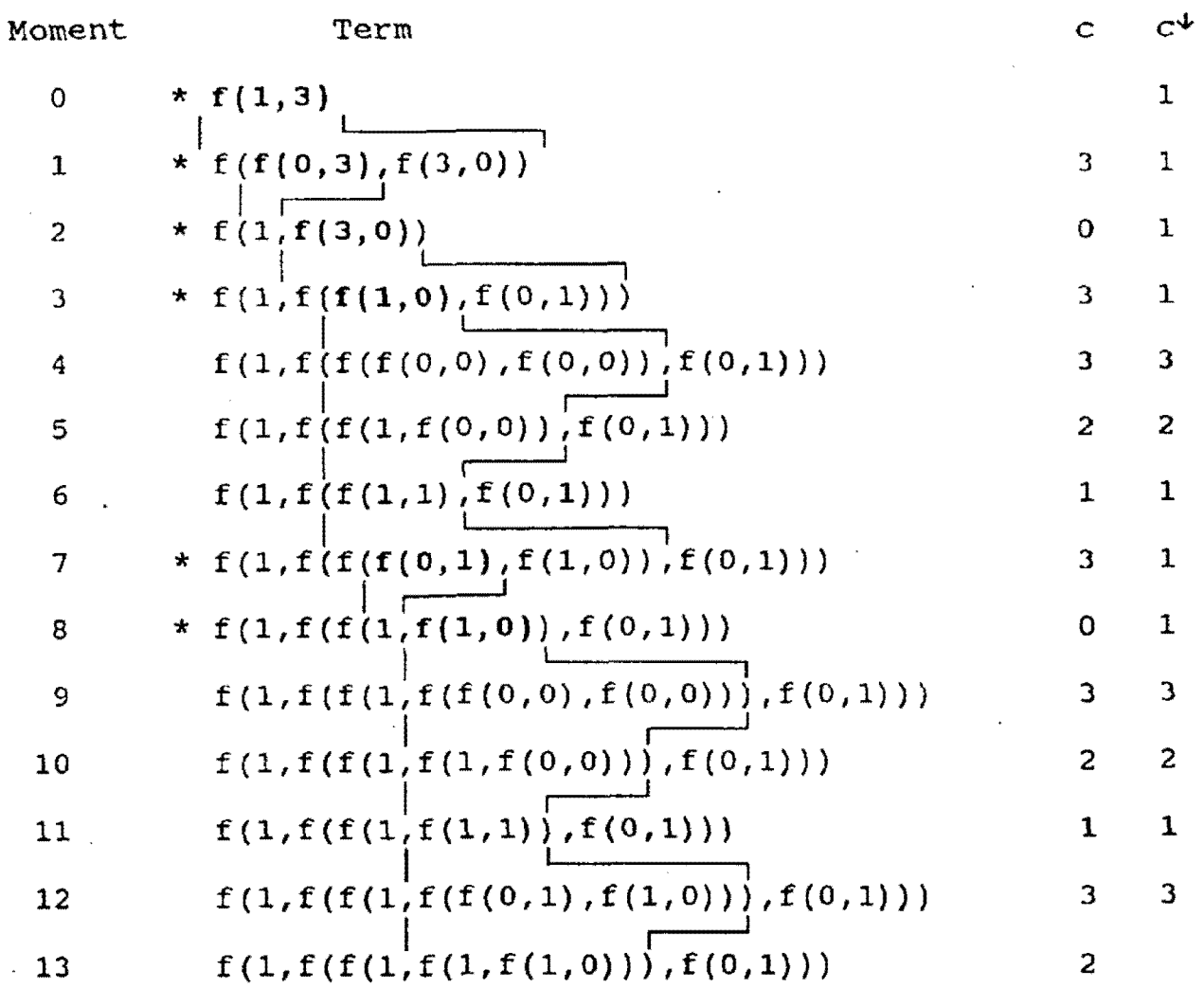

Figure 3. Computation with loop detection activities (Example 11)

For any term u let $|u|$ denote the number of occurrences of symbols from $\mathcal{F}$ in u; this number will be called *the complexity of* u. Clearly, the complexity of a term is equal to 0 iff the term is an atom.

We shall make use of the following intuitively clear statement (cf. Appendix 2 for a proof):

Fact 2. For any term u and any natural number t, if $\mathcal{D}^t(\mathcal{H}(u))$ makes sense, then $\mathcal{D}^t(u)$ also makes sense and the equality

$$|\mathcal{D}^t(u)| - |\mathcal{D}^t(\mathcal{H}(u))| = |u| - 1$$

holds.

The following statement is an immediate corollary of the particular case of Fact 2 when $t = 1$:

Lemma 3. *For any non-atomic term* u *the equality*

$$|\mathcal{D}(u)| - |u| = |\mathcal{D}(\mathcal{H}(u))| - 1$$

holds (*consequently* $|\mathcal{D}(u)| - |u| \geqq -1$).

We shall prove now a lemma which will give us the main tool for incorporating the introduced complexity notion into the loop detection method.

Lemma 4. *Let* u *be a term,* $\bar{t}$ *and* t *be natural numbers such that* $\bar{t} < t$. *Suppose the left-hand side of the equality*

$$|\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = |\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| + |\mathcal{D}(\mathcal{H}(\mathcal{D}^{t}(u)))| - 1$$

makes sense. Then the right-hand one also makes sense and the equality is true.

Proof. The first addend in the right-hand side of the equality obviously makes sense. Let us apply Fact 2 twice with $\mathcal{D}^{\bar{t}}(u)$ in the role of u taking in the role of t the number $t - \bar{t}$ the first time and the number $t - \bar{t} + 1$ the second time. We conclude that $\mathcal{D}^{t}(u)$ and $\mathcal{D}^{t+1}(u)$ make sense and the following equalities hold:

$$|\mathcal{D}^{t}(u)| - |\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = |\mathcal{D}^{\bar{t}}(u)| - 1,$$

$$|\mathcal{D}^{t+1}(u)| - |\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = |\mathcal{D}^{\bar{t}}(u)| - 1.$$

These two equalities imply that

$$|\mathcal{D}^{t}(u)| - |\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = |\mathcal{D}^{t+1}(u)| - |\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))|$$

and therefore

$$|\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = |\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| + |\mathcal{D}^{t+1}(u)| - |\mathcal{D}^{t}(u)|.$$

On the other hand, by Lemma 3 we have also the equality

$$|\mathcal{D}^{t+1}(u)| - |\mathcal{D}^{t}(u)| = |\mathcal{D}(\mathcal{H}(\mathcal{D}^{t}(u))| - 1. \blacksquare$$

Suppose now we examine the recursive computation of a given term u by the already described method. Suppose also the other things assumed in the description of the method, namely: a member $w = \mathcal{D}^{t}(u)$ of the computation is obtained from u by t applications of $\mathcal{D}$, and a snapshot of $\mathcal{H}(\mathcal{D}^{\bar{t}}(u))$ made after $\bar{t}$ applications of $\mathcal{D}$ to u is available, where $\bar{t} < t$, no snapshot is made at moments between $\bar{t}$ and t, and $\mathcal{H}(\mathcal{D}^{\bar{t}}(u))$ belongs to the domain of $\mathcal{D}^{t-\bar{t}}$. In such a situation let us say that $\mathcal{H}(\mathcal{D}^{\bar{t}}(u))$ *is the available snapped head at the moment* t and $\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))$ *is the available descendent term at the moment* t. For the case when the next member $\mathcal{D}^{t+1}(u)$ of the computation must be constructed, it is convenient to introduce also the notion of *descendent term inherited from the moment* t. If no change of the snapshot is made at the moment t, then, by definition, this is the available descendent term at the moment t. Otherwise the descendent term inherited from the moment t is $\mathcal{H}(\mathcal{D}^{t}(u))$, i.e. the new snapped head. We adopt also the convention that the descendent term inherited from the moment 0 is $\mathcal{H}(u)$.

Let c be the complexity of the available descendent term at the moment t. Consider again what has to be done if w belongs to the domain of $\mathcal{D}$. If $c = 0$, then the available snapshot becomes obsolete at the moment t and must be replaced by a snapshot of $\mathcal{H}(\mathcal{D}^{t}(u))$, and then the next member $\mathcal{D}^{t+1}(u)$ of the computation must be constructed. If $c > 0$, then one must compare $\mathcal{H}(w)$ and $\mathcal{H}(\mathcal{D}^{\bar{t}}(u))$. If they turn out to be equal, then a loop is detected in the computation at the moment t. Otherwise, one has again to construct the next member $\mathcal{D}^{t+1}(u)$ of the computation, possibly replacing the snapshot of $\mathcal{H}(\mathcal{D}^{\bar{t}}(u))$ by a snapshot of $\mathcal{H}(\mathcal{D}^{t}(u))$ (if t is a member of the sequence $\tau_0, \tau_1, \tau_2, \ldots$). In any case when a

construction of $\mathcal{D}^{t+1}(u)$ has to be done, it makes sense to consider the available snapped head at the moment $t+1$ and the descendent term at the moment $t+1$. If no change of the snapshot is done at the moment t, then they are $\mathcal{H}(\mathcal{D}^t(u))$ and $\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))$, respectively. Otherwise, they are $\mathcal{H}(\mathcal{D}^t(u))$ and $\mathcal{D}(\mathcal{H}(\mathcal{D}^t(u)))$. Let c' be the complexity of the available descendent term at the moment $t+1$. If no change of the snapshot is done at the moment $t+1$, then, applying Lemma 4, we conclude that

$$c' = c + |\mathcal{D}(\mathcal{H}(\mathcal{D}^t(u))| - 1.$$

In the case when a change of the snapshot is done at the moment $t+1$, we, of course, have

$$c' = |\mathcal{D}(\mathcal{H}(\mathcal{D}^t(u))|.$$

Both equalities may be unified as follows:

$$c' = c^1 + |\mathcal{D}(\mathcal{H}(\mathcal{D}^t(u))| - 1,$$

where c^1 is the complexity of the descendent term inherited from the moment t (*the inherited complexity from the moment* t, for short). The last equality obviously holds also in the case of $t = 0$.

The above considerations suggest the following modification of the detection method: instead of maintaining all the time the available descendent term, one maintains only its complexity. Such a modification works thanks to the above expression for the complexity of the available descendent term at the moment $t+1$. In fact, the expression shows how to calculate the complexity in question if we know the head of the current term at the moment t and the inherited complexity from the moment t. On the other hand, the last complexity is equal to 1 if $t = 0$, otherwise it depends in a very simple way on t and on the complexity of the available descendent term at the moment t (both complexities are equal except for the cases when the complexity of the available descendant term at the moment t is 0 or the moment t is a member of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ — in these cases the inherited complexity from the moment t is equal to 1). Figures 2 and 3 illustrate also the application of this modification of the method. Namely, one must pay no attention to the vertical and broken lines used before, and must look at the two rightmost columns instead.

Example 12. The modified form of the loop detection method is convenient for program implementation. Figure 4 shows a Pascal program which implements the application of the method in the situation from Example 7 to recursive computations with initial member in $\mathcal{U}_1$ (the sequence $\tau_0, \tau_1, \tau_2, \ldots$ with $\tau_n = 2^n - 1$ is used). The program writes the arguments of the snapped head to the variables `a` and `b`. Complexity of descendent terms is written to the variable `c`. Figure 5 displays the output from the application of the same program to the recursive computation of f(1, 3) mentioned in Example 9 (for the sake of saving space the output is displayed in two columns).

The description of the proposed loop detection method can be given in a more formal way, and this is especially advisable for a correctness and completeness proof. Until this moment our presentation used some intuitive ideas without complete definition of the terminology, and no correctness and completeness proof for the

```
{$S+}
var k,l,m,t,tau,a,b,c:word;

procedure snap(x,y:word);
begin
 a:=x;b:=y;c:=1;
 writeln('Snapshot at moment ',t,': f(',x,',',y,')')
end;

function f(x,y:word):word;
var z:word;
begin
 if c=0 then snap(x,y)
 else if (x=a) and (y=b) then
  begin writeln('Loop detected at moment ',t);halt end
 else if t=tau then snap(x,y);
 if t<65535 then begin if t=tau then tau:=2*tau+1;t:=t+1 end
 else begin writeln('Too many steps!');halt end;
 z:=x div 2;
 if odd(x) then begin c:=c+2;f:=f(f(z,y),f(y,z)) end
 else begin c:=c-1;f:=z+9 end
end;

begin
 t:=0;tau:=0;c:=0;
 readln(k,l);m:=f(k,l);writeln('f(',k,',',l,')=',m)
end.
```

Figure 4. A loop detecting program (Example 12)

method has been given. Now we shall give a description of the method by means of standard mathematical terminology. As to the proof, it will be given in the next section.

The process of application of the method (in its modified form) will be presented in the form of constructing some elements of the Cartesian product $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ (i.e. of some quadruples (t,w,v,c), where w is a term, v is a simple non-atomic term, t and c are natural numbers). An element (t,w,v,c) of $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ will be said to *detect a loop* iff the inequality $c>0$ and the equality $\mathcal{H}(w)=v$ hold (clearly, w cannot be an atom in this case). A quadruple from $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ will be called *terminal* iff this quadruple detects a loop or the second component of the quadruple is an atom. A non-terminal element (t,w,v,c) of $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ will be said to *invoke a snapshot* iff the equality $c=0$ holds or $t=\tau_n$ for some n in $\mathbb{N}$. *The snapshot information inherited from* a non-terminal element (t,w,v,c) of $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ is, by definition, the pair $(\mathcal{H}(w),1)$ if (t,w,v,c) invokes a snapshot, and it is the pair (v,c) otherwise. To each non-terminal element (t,w,v,c) of $\mathbb{N}\times\mathcal{U}\times\mathcal{U}_1\times\mathbb{N}$ we make to correspond another quadruple called its *successor*. By definition, this is the quadruple

$$(t+1,\mathcal{D}(w),v^l,c^l+|\mathcal{D}(\mathcal{H}(w))|-1),$$

```
Snapshot at moment 0: f(1,3)      Snapshot at moment 48: f(9,17)
Snapshot at moment 1: f(0,3)      Snapshot at moment 63: f(11,15)
Snapshot at moment 2: f(3,0)      Snapshot at moment 127: f(4,16)
Snapshot at moment 3: f(1,0)      Snapshot at moment 128: f(16,4)
Snapshot at moment 7: f(4,9)      Snapshot at moment 129: f(11,17)
Snapshot at moment 8: f(9,4)      Snapshot at moment 151: f(14,1)
Snapshot at moment 15: f(5,2)     Snapshot at moment 152: f(16,16)
Snapshot at moment 19: f(2,5)     Snapshot at moment 153: f(14,3)
Snapshot at moment 20: f(14,10)   Snapshot at moment 154: f(17,16)
Snapshot at moment 21: f(10,16)   Snapshot at moment 167: f(14,7)
Snapshot at moment 22: f(11,5)    Snapshot at moment 168: f(15,16)
Snapshot at moment 31: f(2,5)     Snapshot at moment 255: f(11,17)
Snapshot at moment 32: f(5,2)     Snapshot at moment 277: f(14,1)
Snapshot at moment 36: f(10,14)   Snapshot at moment 278: f(16,16)
Snapshot at moment 37: f(14,14)   Snapshot at moment 279: f(14,3)
Snapshot at moment 38: f(14,16)   Snapshot at moment 280: f(17,16)
Snapshot at moment 39: f(11,16)   Snapshot at moment 293: f(14,7)
Snapshot at moment 46: f(0,1)     Snapshot at moment 294: f(15,16)
Snapshot at moment 47: f(16,9)    Loop detected at moment 420
```

Figure 5. Output from the loop detecting program (Example 12)

where (ν^1, c^1) is the snapshot information inherited from the quadruple (t, w, ν, c).

Suppose now a term u is given. The process of construction of consecutive members of the recursive computation of u with the addition of loop detection activities looks as follows. We start with the quadruple $(0, u, \nu_0, 0)$, where ν_0 is an arbitrarily chosen simple non-atomic term (the concrete choice of ν_0 will be unessential). If this quadruple is non-terminal, then we go to its successor, and if this successor is also non-terminal, we do the same with it, and so on until eventually a terminal quadruple is obtained. If such a quadruple is obtained in the course of the process and this quadruple detects a loop, then a cyclic loop is present in the recursive computation of u. Conversely, if a cyclic loop is present in the recursive computation of u, then some quadruple detecting a loop will be obtained in the course of the process. The first of these two statements (with a supplement concerning the other kind of terminal quadruples) is the correctness theorem for the presented method, and the second of them (with a similar supplement) is the completeness theorem for this method. Clearly, any of these two theorems needs a proof. Such proofs will be given in the next section.

3. CORRECTNESS AND COMPLETENESS THEOREMS

The correctness of the method means that in all cases of terminating application of the method the conclusion given by it corresponds to the actual state of affairs. To formulate this more precisely, we define a mapping $\mathcal{D}_\tau$ of the set of the non-terminal elements of $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ into the whole $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ by the convention that $\mathcal{D}_\tau$ transforms all element of its domain into their successors. We shall

write simply $\mathcal{D}_\tau^t$ instead of $(\mathcal{D}_\tau)^t$. The images of a given element of $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ under the mappings $\mathcal{D}_\tau^t$, $t = 0, 1, 2, \ldots$, will be called *accessible from* this element.

The following statement can be established by means of a trivial induction:

Lemma 5. *Let* $\mathfrak{u}$ *be an element of* $\mathcal{U}$, *and* ν_0 *be an element of* $\mathcal{U}_1$. *For any natural number* t *such that* $(0, \mathfrak{u}, \nu_0, , 0)$ *belongs to the domain of* $\mathcal{D}_\tau^t$, *the first and the second component of the quadruple* $\mathcal{D}_\tau^t(0, \mathfrak{u}, \nu_0, 0)$ *are* t *and* $\mathcal{D}^t(\mathfrak{u})$, *respectively.*

The correctness theorem reads as follows.

Theorem 1. *Let* $\mathfrak{u}$ *be an element of* $\mathcal{U}$, ν_0 *be an element of* $\mathcal{U}_1$, *and a terminal element of* $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ *be accessible from the quadruple* $(0, \mathfrak{u}, \nu_0, 0)$. *If this terminal element detects a loop, then a cyclic loop is present in the recursive computation of* $\mathfrak{u}$, *otherwise the computation is finite.*

Proof. If the mentioned terminal element does not detect a loop, i.e. if the second component of this element is an atom, then, by Lemma 5, $\mathcal{D}^t(\mathfrak{u})$ will be an atom for some t and consequently the recursive computation of $\mathfrak{u}$ will be finite.

Consider now the case when the terminal element in question detects a loop. We must show that a cyclic loop is present in the recursive computation of $\mathfrak{u}$ in this case.

We shall firstly prove that any quadruple (t, w, ν, c) accessible from $(0, \mathfrak{u}, \nu_0, 0)$ satisfies the following condition: $t = 0$ or there is some natural number $\bar{t}$ such that $\bar{t} < t$, the term ν belongs to the domain of $\mathcal{D}^{t-\bar{t}}$, and the equalities

$$\nu = \mathcal{H}(\mathcal{D}^{\bar{t}}(\mathfrak{u})), \quad c = |\mathcal{D}^{t-\bar{t}}(\nu)| \tag{1}$$

hold. The proof will be done by induction on t. If $t = 0$, then the condition is satisfied. Assume now that the statement is true for a certain natural number t, and suppose that a quadruple of the form $(t+1, w', \nu', c')$ is accessible from $(0, \mathfrak{u}, \nu_0, 0)$. This quadruple is the successor of some quadruple (t, w, ν, c), also accessible from $(0, \mathfrak{u}, \nu_0, 0)$. By the induction hypothesis and the definition of successor the above condition is satisfied for (t, w, ν, c), and we have the equalities

$$\nu' = \nu^{\mathfrak{l}}, \quad c' = c^{\mathfrak{l}} + |\mathcal{D}(\mathcal{H}(w))| - 1,$$

where $(\nu^{\mathfrak{l}}, c^{\mathfrak{l}})$ is the snapshot information inherited from the quadruple (t, w, ν, c). By Lemma 5, also the equality $w = \mathcal{D}^t(\mathfrak{u})$ holds. Let us consider firstly the case when (t, w, ν, c) invokes a snapshot. Then $\nu^{\mathfrak{l}} = \mathcal{H}(w)$, $c^{\mathfrak{l}} = 1$, hence $\nu' = \mathcal{H}(w)$, $c' = |\mathcal{D}(\mathcal{H}(w))|$, and therefore the equalities

$$\nu' = \mathcal{H}(\mathcal{D}^t(\mathfrak{u})), \quad c' = |\mathcal{D}(\nu')|$$

hold. Thus the condition in question will be satisfied if we take ν', c', $t+1$, t in the roles of ν, c, t, $\bar{t}$, respectively. Suppose now that (t, w, ν, c) does not invoke a snapshot. Then $\nu^{\mathfrak{l}} = \nu$, $c^{\mathfrak{l}} = c$, hence $\nu' = \nu$, $c' = c + |\mathcal{D}(\mathcal{H}(w))| - 1$. Since certainly $t > 0$ in this case, we may find a natural number $\bar{t}$ such that $\bar{t} < t$, the term ν belongs to the domain of $\mathcal{D}^{t-\bar{t}}$, and the equalities (1) hold. Then we shall have the inequality $\bar{t} < t + 1$ and the equalities

$$\nu' = \mathcal{H}(\mathcal{D}^{\bar{t}}(\mathfrak{u})), \quad c' = |\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(\mathfrak{u})))| + |\mathcal{D}(\mathcal{H}(\mathcal{D}^t(\mathfrak{u})))| - 1.$$

By Lemma 4 the second of these equalities can be represented in the form $c' = |\mathcal{D}^{t-\bar{t}+1}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))|$ (the application of the lemma is allowed, since $|\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = c > 0$). Hence

$$c' = |\mathcal{D}^{t+1-\bar{t}}(v')|,$$

therefore the condition in question will be satisfied with the same $\bar{t}$ if we take v', c', $t+1$ in the roles of v, c, t, respectively. This completes the induction step of the proof.

Let us take now the mentioned terminal element of $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ as (t, w, v, c). Since this element detects a loop, the inequality $c > 0$ and the equality $\mathcal{H}(w) = v$ hold. By the above considerations we may find a natural number $\bar{t}$ such that $\bar{t} < t$, the term v belongs to the domain of $\mathcal{D}^{t-\bar{t}}$, and the equalities (1) hold. The first of these equalities shows that u activates v after $\bar{t}$ steps. On the other hand, an application of Fact 1 with $\mathcal{D}^{\bar{t}}(u)$ and $t - \bar{t}$ in the roles of u and t, respectively, together with the same equality and the equality $w = \mathcal{D}^t(u)$, shows that $\mathcal{H}(\mathcal{D}^{t-\bar{t}}(v)) = \mathcal{H}(w)$ (the application of Fact 1 is allowed, since $|\mathcal{D}^{t-\bar{t}}(\mathcal{H}(\mathcal{D}^{\bar{t}}(u)))| = c > 0$). Taking into account also the equality $\mathcal{H}(w) = v$, we see that v activates itself after $t - \bar{t}$ steps, and hence v is a self-reactivating term. Thus u activates a self-reactivating term after $\bar{t}$ steps. ■

For the proof of the completeness theorem two lemmas more will be needed.

Lemma 6. *Let u be a term, and let u activate a self-reactivating simple non-atomic term v after s steps. Let v activate itself after r steps, where $r > 0$, and let t be an arbitrary integer satisfying the inequality $t \geqq s$. Then*

$$\mathcal{H}(\mathcal{D}^{t+r}(u)) = \mathcal{H}(\mathcal{D}^t(u)), \quad |\mathcal{D}^{t+r}(u)| \geqq |\mathcal{D}^t(u)|. \tag{2}$$

Proof. The above statements make sense, since, by Lemma 2, the recursive computation of u is infinite. By Lemma 1 u activates v also after $s + r$ steps, i.e.

$$\mathcal{H}(\mathcal{D}^s(u)) = \mathcal{H}(\mathcal{D}^{s+r}(u)) = v.$$

Set $d = t - s$. Then $t = s + d$, $t + r = s + r + d$, hence

$$\mathcal{H}(\mathcal{D}^t(u)) = \mathcal{H}(\mathcal{D}^d(\mathcal{D}^s(u))), \quad \mathcal{H}(\mathcal{D}^{t+r}(u)) = \mathcal{H}(\mathcal{D}^d(\mathcal{D}^{s+r}(u))).$$

From here, making use of Fact 1 with d in the role of t and $\mathcal{D}^s(u)$, $\mathcal{D}^{s+r}(u)$ in the role of u, we get

$$\mathcal{H}(\mathcal{D}^t(u)) = \mathcal{H}(\mathcal{D}^d(v)), \quad \mathcal{H}(\mathcal{D}^{t+r}(u)) = \mathcal{H}(\mathcal{D}^d(v)),$$

thus the equality in (2) is established. For the proof of the inequality we apply Fact 2 in the same way. The result of its application looks as follows:

$$|\mathcal{D}^t(u)| - |\mathcal{D}^d(v)| = |\mathcal{D}^s(u)| - 1,$$
$$|\mathcal{D}^{t+r}(u)| - |\mathcal{D}^d(v)| = |\mathcal{D}^{s+r}(u)| - 1.$$

One more application of Fact 2, this time with r in the role of t and $\mathcal{D}^s(u)$ in the role of u, shows that

$$|\mathcal{D}^{s+r}(u)| - |\mathcal{D}^r(v)| = |\mathcal{D}^s(u)| - 1.$$

From the last three equalities we get

$$|\mathcal{D}^{t+r}(u)| - |\mathcal{D}^t(u)| = |\mathcal{D}^{s+r}(u)| - |\mathcal{D}^s(u)| = |\mathcal{D}^r(v)| - 1,$$

and the validity of the inequality is clear, since $|\mathcal{D}^r(v)| \geqq 1$.

Lemma 7. *Let u be a term, v_0 be an element of $\mathcal{U}_1$. Let $\bar{t}$ and $\tilde{t}$ be natural numbers such that $\bar{t} < \tilde{t}$, no member of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ is strictly between $\bar{t}$ and $\tilde{t}$, the quadruple $(0, u, v_0, 0)$ belongs to the domain of $\mathcal{D}_\tau^{\tilde{t}}$, the quadruple $\mathcal{D}_\tau^{\bar{t}}(0, u, v_0, 0)$ invokes a snapshot, and $|\mathcal{D}^t(u)| \geqq |\mathcal{D}^{\bar{t}}(u)|$ holds, whenever $\bar{t} < t < \tilde{t}$. Then*

$$\mathcal{D}_\tau^{\tilde{t}}(0, u, v_0, 0) = (\tilde{t}, \mathcal{D}^{\tilde{t}}(u), \mathcal{H}(\mathcal{D}^{\bar{t}}(u)), 1 + |\mathcal{D}^{\tilde{t}}(u)| - |\mathcal{D}^{\bar{t}}(u)|).$$

Proof. Induction on $\tilde{t}$ is used with the case of $\tilde{t} = \bar{t} + 1$ as induction basis. To settle that case and to do the induction step from $\tilde{t}$ to $\tilde{t} + 1$, one applies Lemma 3 with $\mathcal{D}^{\tilde{t}}(u)$ and with $\mathcal{D}^{\bar{t}}(u)$ in the role of u, respectively. ■

Now we shall formulate and prove the completeness theorem.

Theorem 2. *For any term u and any v_0 from $\mathcal{U}_1$ the following two statements hold:*

(i) *If a cyclic loop is present in the recursive computation of u, then some element of $\mathbb{N} \times \mathcal{U} \times \mathcal{U}_1 \times \mathbb{N}$ accessible from $(0, u, v_0, 0)$ detects a loop.*

(ii) *If the recursive computation of u is finite, then some element of $\mathbb{N} \times \mathcal{A} \times \mathcal{U}_1 \times \mathbb{N}$ is accessible from $(0, u, v_0, 0)$.*

Proof. The proof of (ii) is quite easy. In fact, suppose that the recursive computation of u is finite and consider a natural number s such that $\mathcal{D}^s(u)$ is an atom. Making use of Lemma 5 we conclude that $(0, u, v_0, 0)$ does not belong to the domain of $\mathcal{D}_\tau^{s+1}$. Therefore some terminal quadruple (with first component not greater than s) is accessible from $(0, u, v_0, 0)$. It is not possible that this terminal quadruple detects a loop, since, by Theorem 1, then a cyclic loop would be present in the recursive computation of u. Hence the terminal quadruple in question has an atomic second component (the first component of the quadruple will be s, as it is easy to see).

Consider now the case when a cyclic loop is present in the recursive computation of u (hence this computation is infinite). Let u activate a self-reactivating simple non-atomic term v after s steps, and let v activate itself after r steps, where $r > 0$. Making use of the properties of the sequence $\tau_0, \tau_1, \tau_2, \ldots$ we find a natural number n such that

$$\tau_n \geqq s, \quad \tau_{n+1} - \tau_n \geqq 2r - 1.$$

We shall prove that some quadruple accessible from $(0, u, v_0, 0)$ and having first component less than $\tau_n + 2r$ detects a loop.

Let $w = \mathcal{D}^{\tau_n}(u)$. Of course, w belongs to the domain of $\mathcal{D}^i$ for any natural number i. Let e be the minimal one among the complexities of the terms $\mathcal{D}^i(w)$, $i = 0, 1, 2, \ldots, r-1$. It is easily seen that a finite sequence of natural numbers $i_0 < i_1 < \cdots < i_p$ can be found with the following three properties: (a) $i_0 = 0$; (b) for any natural number j less than p, i_{j+1} is the least one among the natural numbers

i satisfying the inequalities $i_j < i \leqq r-1$, $|\mathcal{D}^i(w)| < |\mathcal{D}^{i_j}(w)|$; (c) $|\mathcal{D}^{i_p}(w)| = e$ (in case of $|w| = e$ we have $p = 0$ and the properties (b) and (c) become trivial). Clearly, $i_p \leqq r-1$.

We set $T = \tau_n + i_p + r$. Then $\tau_n < T \leqq \tau_n + 2r - 1 \leqq \tau_{n+1}$. If the quadruple $(0, u, v_0, 0)$ does not belong to the domain of $\mathcal{D}_\tau^T$, then some terminal quadruple with first component smaller than T will be accessible from $(0, u, v_0, 0)$. This quadruple must detect a loop, since otherwise the recursive computation of u would be finite according to Theorem 1. Thus it remains to study only the case when $(0, u, v_0, 0)$ belongs to the domain of $\mathcal{D}_\tau^T$. Then we consider firstly the quadruple $\mathcal{D}_\tau^{\tau_n}(0, u, v_0, 0)$. It is non-terminal and its first component is τ_n. Therefore this quadruple invokes a snapshot.

We shall show by induction that for $j = 0, 1, \ldots, p$ the quadruple $\mathcal{D}_\tau^{\tau_n + i_j}(0, u, v_0, 0)$ invokes a snapshot. For $j = 0$ the statement has been already established. Suppose now this statement is true for a certain natural number j less than p and apply Lemma 7 with $\tau_n + i_j$ and $\tau_n + i_{j+1}$ in the roles of $\bar{t}$ and $\tilde{t}$, respectively. We get the equality

$$\mathcal{D}_\tau^{\tau_n + i_{j+1}}(0, u, v_0, 0) = (\tau_n + i_{j+1}, \mathcal{D}^{i_{j+1}}(w), \mathcal{H}(\mathcal{D}^{i_j}(w)), 1 + |\mathcal{D}^{i_{j+1}}(w)| - |\mathcal{D}^{i_j}(w)|).$$

Thus the non-terminal quadruple $\mathcal{D}_\tau^{\tau_n + i_{j+1}}(0, u, v_0, 0)$ has a last component less than 1, hence equal to 0, and therefore the quadruple invokes a snapshot.

By applying the proved statement with $j = p$ we conclude that the quadruple $\mathcal{D}_\tau^{\tau_n + i_p}(0, u, v_0, 0)$ invokes a snapshot. Let w' be the second component of this quadruple, i.e. $w' = \mathcal{D}^{i_p}(w)$. We shall show that $|\mathcal{D}^h(w')| \geqq |w'|$ for $h = 1, 2, \ldots, r$. In fact, $\mathcal{D}^h(w') = \mathcal{D}^{i_p + h}(w)$ and $0 < i_p + h \leqq 2r - 1$ for the specified values of h. Since $|w'| = e$, the inequality in question is obviously satisfied when $i_p + h \leqq r - 1$. On the other hand, if $i_p + h > r - 1$, then $i_p + h = i + r$ for some i among $0, 1, \ldots, r-1$, hence $|\mathcal{D}^h(w')| = |\mathcal{D}^{i+r}(w)| \geqq |\mathcal{D}^i(w)|$ by the inequality in (2) with $\tau_n + i$ in the role of t. Thus $|\mathcal{D}^h(w')| \geqq |w'|$ holds again.

Now let us apply Lemma 7 with $\tau_n + i_p$ and T in the roles of $\bar{t}$ and $\tilde{t}$, respectively. We get the equality

$$\mathcal{D}_\tau^T(0, u, v_0, 0) = (T, \mathcal{D}^r(w'), \mathcal{H}(w'), 1 + |\mathcal{D}^r(w')| - |w'|).$$

We note that $\mathcal{H}(\mathcal{D}^r(w')) = \mathcal{H}(w')$, as seen from the equality in (2) with $\tau_n + i_p$ in the role of t. Taking into account also the inequality $|\mathcal{D}^r(w')| \geqq |w'|$, we conclude that $\mathcal{D}_\tau^T(0, u, v_0, 0)$ detects a loop. ■

APPENDIX 1. ON INFINITE RECURSIVE COMPUTATIONS WITH FINITELY MANY ATOMS AND FUNCTION SYMBOLS

We add one lemma and one theorem more to the results proved in the preceding sections.

Lemma 8. *Let w be a term such that the recursive computation of w is infinite and no member of this computation has a smaller complexity than w. Then the recursive computation of the head of w is also infinite.*

Proof. We shall show by induction that $\mathcal{H}(w)$ belongs to the domain of $\mathcal{D}^t$ for any natural number t. The statement is trivial for $t = 0$. Suppose $\mathcal{D}^t(\mathcal{H}(w))$ makes sense for a certain natural number t. Then $|\mathcal{D}^t(\mathcal{H}(w))| = |\mathcal{D}^t(w)| - |w| + 1 \geqq 1$ by Fact 2 and the assumption of the lemma. Hence $\mathcal{D}^{t+1}(\mathcal{H}(w))$ also makes sense. ∎

Theorem 3. *Let $\mathfrak{u}$ be a term such that the recursive computation of $\mathfrak{u}$ is infinite, but there are only finitely many atoms and function symbols which occur in the members of this computation. Then a cyclic loop is present in the computation.*

Proof. Let us call a natural number t *remarkable* if for any greater natural number s the inequality $|\mathcal{D}^s(\mathfrak{u})| \geqq |\mathcal{D}^t(\mathfrak{u})|$ holds. It is easy to prove (by *reductio ad absurdum*) the existence of infinitely many remarkable numbers. On the other hand, the set of the terms of the form $\mathcal{H}(\mathcal{D}^t(\mathfrak{u}))$ is finite due to the made assumption. Hence, there are remarkable numbers t and t' such that $t < t'$ and $\mathcal{H}(\mathcal{D}^t(\mathfrak{u})) = \mathcal{H}(\mathcal{D}^{t'}(\mathfrak{u}))$. By Lemma 8 the recursive computation of the head of $\mathcal{D}^t(\mathfrak{u})$ is infinite. Then, by Fact 1, also the equality $\mathcal{H}(\mathcal{D}^{t'}(\mathfrak{u})) = \mathcal{H}(\mathcal{D}^{t'-t}(\mathcal{H}(\mathcal{D}^t(\mathfrak{u}))))$ holds and therefore $\mathcal{H}(\mathcal{D}^t(\mathfrak{u}))$ is a self-reactivating term. ∎

Theorem 3 shows that there are many cases when the proposed loop detection method completely solves the problem whether the recursive computation of an arbitrarily chosen term is finite or infinite.

Example 13. Let $\mathcal{A}$ and $\mathcal{F}$ be the same as in Example 1, and let d be some given natural number. Consider the recursion rule

$$\mathcal{D}(\mathrm{f}(2z, y)) = z + d, \quad \mathcal{D}(\mathrm{f}(2z + 1, y)) = \mathrm{f}(\mathrm{f}(z, y), \mathrm{f}(y, z))$$

(it covers the rules from Example 1 and Example 7). Suppose some term $\mathfrak{u}$ is given. Denote by h some natural number which is not less than $2d - 1$ and than the value of any of the atoms occurring in $\mathfrak{u}$. An easy induction shows that no atom occurring in some member of the recursive computation of $\mathfrak{u}$ has a value greater than h. Thus only finitely many atoms and only one function symbol may occur in the members of the computation. By Theorem 3, if this computation is infinite, then a cyclic loop will be present in it.

APPENDIX 2. PROOFS OF FACT 1 AND FACT 2

Let us define a mapping $\mathcal{P}$ of $\mathcal{U} \setminus (\mathcal{U}_1 \cup \mathcal{A})$ into $\mathcal{U} \setminus \mathcal{A}$ as follows: whenever $\mathfrak{u} = f(\mathfrak{u}_1, \ldots, \mathfrak{u}_n)$, where f is some n-ary function symbol, $\mathfrak{u}_1, \ldots, \mathfrak{u}_n$ are terms, and at least one among these terms is non-atomic, then $\mathcal{P}(\mathfrak{u})$ is the first non-atomic member of the sequence $\mathfrak{u}_1, \ldots, \mathfrak{u}_n$.

Clearly, $|\mathcal{P}(\mathfrak{u})| < |\mathfrak{u}|$ for any term $\mathfrak{u}$ from the domain of $\mathcal{P}$. For any such term the equality $\mathcal{H}(\mathfrak{u}) = \mathcal{H}(\mathcal{P}(\mathfrak{u}))$ holds. Of course, $\mathcal{H}(\mathfrak{u}) = \mathfrak{u}$ for any term in $\mathcal{U}_1$. Consequently, the head of any non-atomic term $\mathfrak{u}$ is equal to $\mathcal{P}^n(\mathfrak{u})$, where n is the greatest natural number i such that $\mathfrak{u}$ belongs to the domain of $\mathcal{P}^i$.

We note that the domain and the range of $\mathcal{P}$ are contained in the domain of $\mathcal{D}$ and that the following interrelations between the two mappings are easily verifiable:

Fact A. Whenever $\mathfrak{u}$ belongs to the domain of $\mathcal{P}$ and the inequality $|\mathcal{D}(\mathcal{P}(\mathfrak{u}))| > 0$ holds, then $\mathcal{P}(\mathcal{D}(\mathfrak{u}))$ makes sense and the equality $\mathcal{D}(\mathcal{P}(\mathfrak{u})) = \mathcal{P}(\mathcal{D}(\mathfrak{u}))$ holds.

Fact B. For any $\mathfrak{u}$ in the domain of $\mathcal{P}$ the equality

$$|\mathcal{D}(\mathfrak{u})| - |\mathcal{D}(\mathcal{P}(\mathfrak{u}))| = |\mathfrak{u}| - |\mathcal{P}(\mathfrak{u})| \tag{3}$$

holds.

Now we shall prove some generalizations of Fact A and Fact B.

Lemma A. *Whenever n and t are natural numbers, $\mathfrak{u}$ is a term such that $\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))$ makes sense and the inequality $|\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))| > 0$ holds, then $\mathcal{P}^n(\mathcal{D}^t(\mathfrak{u}))$ also makes sense and the equality*

$$\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u})) = \mathcal{P}^n(\mathcal{D}^t(\mathfrak{u})) \tag{4}$$

holds.

Proof. One firstly considers the case of $t = 1$ and proceeds by induction on n in this case. The general case can be obtained from this particular one by induction on t. ■

Lemma B. *Whenever n and t are natural numbers and $\mathfrak{u}$ is a term such that $\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))$ makes sense, then $\mathcal{D}^t(\mathfrak{u})$ also makes sense and the equality*

$$|\mathcal{D}^t(\mathfrak{u})| - |\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))| = |\mathfrak{u}| - |\mathcal{P}^n(\mathfrak{u})|$$

holds.

Proof. We firstly consider the case of $t = 1$ and proceed by the following induction on n in this case. For $n = 0$ the statement is trivial. Suppose the validity of the statement for a certain natural number n and let $\mathfrak{u}$ be such a term that $\mathcal{D}(\mathcal{P}^{n+1}(\mathfrak{u}))$ makes sense. Then an application of the induction hypothesis with $\mathcal{P}(\mathfrak{u})$ in the role of $\mathfrak{u}$ yields the equality

$$|\mathcal{D}(\mathcal{P}(\mathfrak{u}))| - |\mathcal{D}(\mathcal{P}^{n+1}(\mathfrak{u}))| = |\mathcal{P}(\mathfrak{u})| - |\mathcal{P}^{n+1}(\mathfrak{u})|.$$

This equality together with (3) implies the needed equality

$$|\mathcal{D}(\mathfrak{u})| - |\mathcal{D}(\mathcal{P}^{n+1}(\mathfrak{u}))| = |u| - |\mathcal{P}^{n+1}(\mathfrak{u})|.$$

To obtain the general case from the particular one of $t = 1$, we proceed by induction on t. For $t = 0$ the general statement is trivial. Suppose the validity of the general statement for a certain natural number t, and let the term $\mathfrak{u}$ and the natural number n be such that $\mathcal{D}^{t+1}(\mathcal{P}^n(\mathfrak{u}))$ makes sense. Then $\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))$ also makes sense and the inequality $|\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))| > 0$ holds. Making use of Lemma A we conclude that $\mathcal{P}^n(\mathcal{D}^t(\mathfrak{u}))$ also makes sense and the equality (4) holds. From here, applying consecutively the particular case of $n = 1$ with $\mathcal{D}^t(\mathfrak{u})$ in the role of $\mathfrak{u}$ and the induction hypothesis, we get

$$\begin{aligned}|\mathcal{D}^{t+1}(\mathfrak{u})| - |\mathcal{D}^{t+1}(\mathcal{P}^n(\mathfrak{u}))| &= |\mathcal{D}^{t+1}(\mathfrak{u})| - |\mathcal{D}(\mathcal{P}^n(\mathcal{D}^t(\mathfrak{u})))| = |\mathcal{D}^t(\mathfrak{u})| - |\mathcal{P}^n(\mathcal{D}^t(\mathfrak{u}))| \\ &= |\mathcal{D}^t(\mathfrak{u})| - |\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))| = |\mathfrak{u}| - |\mathcal{P}^n(\mathfrak{u})|. \blacksquare\end{aligned}$$

We shall present proofs of Fact 1 and Fact 2 using Lemma A and Lemma B, respectively.

Proof of Fact 1. Suppose that $\mathfrak{u}$ is a term and t is a natural number such that $\mathcal{H}(\mathcal{D}^t(\mathcal{H}(\mathfrak{u})))$ makes sense. Let n be a natural number such that $\mathcal{H}(\mathfrak{u}) = \mathcal{P}^n(\mathfrak{u})$. Then the expression $\mathcal{H}(\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u})))$ will make sense and therefore the premise of Lemma A will be satisfied. Hence, the conclusion of the lemma will be also satisfied. Thus $\mathcal{P}^n(\mathcal{D}^t(\mathfrak{u}))$ will also make sense and the equality

$$\mathcal{D}^t(\mathcal{H}(\mathfrak{u})) = \mathcal{P}^n(\mathcal{D}^t(\mathfrak{u}))$$

will hold. Now it is clear that the terms $\mathcal{D}^t(\mathcal{H}(\mathfrak{u}))$ and $\mathcal{D}^t(\mathfrak{u})$ are non-atomic, and the term obtained by applicating $\mathcal{P}$ the maximal possible number of times will be one and the same if we start from either of these two terms.

Proof of Fact 2. Suppose that $\mathfrak{u}$ is a term and t is a natural number such that $\mathcal{D}^t(\mathcal{H}(\mathfrak{u}))$ makes sense. Let n be a natural number such that $\mathcal{H}(\mathfrak{u}) = \mathcal{P}^n(\mathfrak{u})$. Then the expression $\mathcal{D}^t(\mathcal{P}^n(\mathfrak{u}))$ will make sense and therefore the premise of Lemma B will be satisfied. Hence, the conclusion of the lemma will be also satisfied. Thus $\mathcal{D}^t(\mathfrak{u})$ also makes sense and the equality

$$|\mathcal{D}^t(\mathfrak{u})| - |\mathcal{D}^t(\mathcal{H}(\mathfrak{u}))| = |\mathfrak{u}| - |\mathcal{H}(\mathfrak{u})|$$

holds. It remains only to take into account that $|\mathcal{H}(\mathfrak{u})| = 1$. ■

Added in proof: When writing the present paper, the author did not know about the paper of F. E. Fich "Lower bounds for the cycle detection problem", J. of Computer and System Sciences, **26**, 1983, 392–409. Thanks are due to Professor Donald Knuth who called the author's attention to that paper, essentially containing the results of [2] (as well as other ones).

REFERENCES

1. Knuth, D. E. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms. Second Edition, Reading, Mass., 1981.
2. Skordev, D. An extremal problem concerning the detection of cyclic loops. — C. R. Acad. Bulgare Sci., **40**, no. 10, 1987, 5–8.
3. Van Gelder, A. Efficient loop detection in Prolog using the tortoise-and-hare technique. — J. Logic Programming, **4**, 1987, 23–31.
4. Van Gelder, A. Van Gelder's response. — J. Logic Programming, **14**, 1992, 185.
5. Skordev, D. On Van Gelder's loop detection algorithm. — J. Logic Programming, **14**, 1992, 181–183.

Received on 12.05.1994