Fundamenta Informaticae 31 (1997) 195–212
IOS Press

An Abstract Approach to Some Loop Detection Problems*

Dimiter Skordev
Faculty of Mathematics and Informatics
University of Sofia
Sofia, Bulgaria
skordev@fmi.uni–sofia.bg

To the memory of Professor Helena Rasiowa

Abstract. The abstract approach proposed here encompasses both the detection of some periodic loops during the execution of Prolog programs and the detection of some periodic loops during recursive computations (an attempt to look at the loop detection problem for Prolog from an abstract point of view has been done previously in a paper by the same author published in 1993).

Keywords: cyclic loop, loop detection, logic program, recursive computation

1. Introduction

The problem we study is how to detect some loops in computational processes in the course of those processes. This problem has both some practical significance and certain theoretical attractiveness. The early methods invented by R. W. Floyd and R. P. Brent (cf. [7], section 3.1) are abstract in their nature. Given any set and any mapping in it, these methods allow the detection of the possible ultimate periodicity of the trajectory obtained by iterated application of the mapping starting from an arbitrarily chosen element of the set. More precisely, they accomplish partial decision procedures computable in the given mapping and the equality predicate in the given set.[1] The same is true also for the generalization of the mentioned methods studied in [5].[2]

Certain important kinds of computational processes lead sometimes to loops with periodic features different from ultimate periodicity of the trajectory. This can happen in cases when

*Research partially supported by the Bulgarian Ministry of Education and Science, Contract I-412/94. A preliminary version of the paper has been presented at the minisemester "Logic, Algebra and Computer Science: Helena Rasiowa in Memoriam" (Stefan Banach International Mathematical Center, Warsaw, Poland, December 2-22, 1996).

[1]Of course, a trivial partial decision procedure computable in the same sense exists, namely comparing each trajectory member with all preceding ones. This procedure, however, is not practically usable in most cases of interest.

[2]We have not included references to [5] in most of our previous papers due to not knowing about that publication during a long period of time.

the complexity of the processed objects may vary in the course of the process. Typical kinds of processes of this sort are the depth-first search at the execution of Prolog programs and the recursive computations (the processed objects are finite sequences of atomic goals in the first case and terms built up from atoms by means of application of functional symbols in the second one). Such processes seldom lead to a complete repetition of the current object under processing. The instances of non–termination of them are more frequently due to a more general sort of self–reproduction characterized by a periodic repetition of some essential part of the object. The setting of the problem and the ways of solving it are more complicated in this case. Nevertheless, the loop detection problem for Prolog (and not only for the depth-first search, but also for the general case), due to its actuality, has been intensively studied by many authors in an impressive stream of publications (as some instances of such ones, cf. e.g. [1], [2], [3], [4], [6], [8], [9], [14], [15], [16]). However most proposed loop detection methods for Prolog essentially use specific features of the sort of program execution in question. An attempt to look at the problem from an abstract point of view has been done in our paper [10], which is also directed to loop detection in Prolog, but uses an axiomatic framework.

The present paper aims at the building of a more general framework encompassing also the case of recursive computations (a slightly more restricted separate study of loop detection in the last case is given in [11]).

2. Goal structures and reducers in them

Our first step will be to give an appropriate axiomatization of the computational universes for the processes of the considered kind.

Definition 2.1. We shall call a *goal structure* any triple (U, C, P), where U is a set (*the universe of goals*), C (*the complexity estimator*) is a mapping of U into the set $\mathbf{N}$ of the non–negative integers, P (*the projection mapping*) is a partial mapping of U into itself, and the following conditions are satisfied:

(gs1) whenever u belongs to $\mathrm{dom}(P)$, the inequalities $0 < C(P(u)) \leq C(u)$ hold;
(gs2) for any u in U, some natural number k exists such that $u \notin \mathrm{dom}(P^k)$.
(gs3) whenever $C(u) > 1$, then u belongs to $\mathrm{dom}(P)$.

If (U, C, P) is a goal structure then, for any u in U, the corresponding integer $C(u)$ will be called *the complexity of* u.

Remark 2.1. In the cases when both inequalities in (gs1) are strict for all u in $\mathrm{dom}(P)$, condition (gs2) becomes obviously superfluous. In such cases $\mathrm{dom}(P)$ consists exactly of the elements of U having complexity greater than 1.

Definition 2.2. Given a goal structure (U, C, P) and an element u of U, we shall call *leading parts of* u the elements $P^k(u)$ for the finitely many k such that $u \in \mathrm{dom}(P^k)$. The leading part $P^k(u)$ of u with the greatest k will be called *the head of* u and will be denoted by $H(u)$.

Remark 2.2. If (U, C, P) is a goal structure then, by the second inequality in (gs1), all leading parts of an element u of U have complexities not greater that $C(u)$.

Remark 2.3. It is easy to see that, in any goal structure (U, C, P), the equality $C(H(u)) = 1$ holds for all u in U such that $C(u) > 0$, and $H(u) = u$ for the elements u of U having complexity 0.

We shall give now some examples of goal structures. The first of them is a trivial one, but it corresponds in some sense to the situation when we are interested in the detection of complete repeating of the objects under processing (i.e. in the detection of the ultimate periodicity of the trajectories).

Example 2.1. Let U be an arbitrary set, let $C(u) = 1$ for any u in U, and let $\mathrm{dom}(P)$ be empty. Then (U, C, P) is a goal structure, and each element of U is its only leading part, hence $H(u) = u$ for any u in U.

The next example covers the computational universe of finite sequences used in the depht-first search at the execution of a logic program, i.e. a Prolog program consisting of Horn clauses (in that example and further, we adopt the denotational convention that applying the composition fg of two mappings f and g means applying first g and then f).

Example 2.2. Let (U, π, R) be a goal space in the sense of [10], i.e. U is a set, π and R are partial operations in U such that $\mathrm{dom}(R) \subseteq \mathrm{dom}(\pi)$, $\mathrm{dom}(R) \cap \mathrm{dom}(\pi^2) \subseteq \mathrm{dom}(R\pi)$, πR is an extension of $R\pi$, and the requirement (gs2) holds with π in the role of P. For any u in U, let $C(u)$ be the length of u in the sense of [10], i.e. the greatest k such that $u \in \mathrm{dom}(\pi^k)$, and let P be the restriction of π to $\mathrm{dom}(\pi^2)$. Then (U, C, P) is a goal structure, such that $C(P(u)) = C(u) - 1$ for any u in $\mathrm{dom}(P)$, and, for any u in $\mathrm{dom}(\pi)$, its leading parts coincide with its beginnings in the sense of [10] that belong to $\mathrm{dom}(\pi)$ (the elements of U not belonging to $\mathrm{dom}(\pi)$ have no leading parts different from themselves). As seen from [10], to cover the computational universe related to a logic program, we have to assume that such a program is given and to denote by U, π and R, respectively, the set of all finite sequences of atomic formulas, the operation of deleting the last member of such a sequence (defined for the non–empty sequences) and the operation of performing resolution upon the first member of the sequence using the first program clause with a head unifiable with that member (this operation is defined if and only if the given sequence from U has a first member unifiable with the head of some program clause; of course the usual procedure of appropriate renaming the program variables is assumed to precede the unification attempts, and this construction of a goal space needs in fact also identification of the sequences from U that coincide up to renaming of the variables). The leading parts of a non–empty sequence from U are the non–empty beginnings of that sequence, and its head is the beginning consisting only of the first member of the sequence. The empty sequence coincides with its head.

Now an example suitable for the case of recursive computations follows.

Example 2.3. Let U consist of all terms built up from some given atoms by means of application of some given function symbols, and let a subset Δ of the set of the function symbols be specified (the function symbols from this subset may be regarded as the definable ones, and it may contain all function symbols). For any u in U, let $C(u)$ be the number of occurrences of function symbols from Δ in u. Let P be the partial mapping of U into itself defined in the following way: $\mathrm{dom}(P)$ consists of the elements u of U having the form $\mathrm{f}(u_1, \ldots, u_n)$, where n is some positive integer, f is some n–ary function symbol, $u_1, \ldots, u_n$ are terms, and at least one u_i has a strictly positive complexity $C(u_i)$; the mapping P transforms any term u of the above form into the leftmost u_i whose complexity is strictly positive. Then (U, C, P) is a goal structure. Let U_1 consist of all terms having the form $\mathrm{f}(u_1, \ldots, u_n)$, where n is some positive integer, f is some n–ary function symbol from Δ, and $u_1, \ldots, u_n$ are terms with complexity 0. Then, whenever u is a term from U with a strictly positive complexity, $H(u)$ is the leftmost occurring in u subterm belonging to U_1, and any element of U with complexity 0 coincides with its head.

Remark 2.4. The situation in the above example becomes simpler in the case when Δ contains all function symbols. Then the complexity of a term is the number of occurrences of function symbols in it, the terms with positive complexities are exactly the non–atomic terms, $\mathrm{dom}(P)$ consists of the terms with complexity greater than 1, and U_1 consists of the terms with complexity 1.

The next step towards the promised abstract treatment of loop detection needs to axiomatize also the computational steps. These will be done by introducing the notion of a reducer in

a goal structure (in its definition and further, when using equalities concerning results of partially defined operations, we interpret the equalities in the sense that if one of the sides is defined then the other one is defined too and has the same value).

Definition 2.3. Let (U, C, P) be a goal structure. A partial mapping R of U into U will be called a *reducer* in the given goal structure if the following conditions are satisfied:

(r1) $C(u) > 0$ for any u in $\mathrm{dom}(R)$;
(r2) $RP(u) = PR(u)$ for any u in $\mathrm{dom}(RP)$ such that $C(RP(u)) > 0$;
(r3) $\mathrm{dom}(RP) = \mathrm{dom}(R) \cap \mathrm{dom}(P)$;
(r4) $C(R(u)) - C(RP(u)) = C(u) - C(P(u))$ for any u in $\mathrm{dom}(RP)$.

Example 2.4. In the goal structure from Example 2.1, any partial mapping of U into U is obviously a reducer.

Example 2.5. In the goal structure from Example 2.2, the mapping R is surely a reducer.

Example 2.6. To get a reducer in the goal structure from Example 2.3, suppose D is a mapping of the set U_1 into U (in the case when Δ contains all function symbols, this is the requirement D to be a recursion rule in the sense of [11]). Let R be the extension of D in the following way generalizing the corresponding construction in [11]: $\mathrm{dom}(R)$ consists of all terms with strictly positive complexities, and, for any such term u, $R(u)$ is the result of replacing of the leftmost occurrence in u of a term from U_1 by the corresponding result of application of D (in [11], the extension in question is denoted in the same way as the initially given recursion rule). Then R is a reducer in (U, C, P).

3. Cyclic elements

We shall generalize the results from [11] to the case of an arbitrary goal structure and an arbitrary reducer in it, thus obtaining also a simpler loop detection method for the case studied in [10] (a variant of this simpler method has been previously described in [12]). The formulation of the results uses the notion of a cyclic element of a goal structure.

From now on, an arbitrary goal structure (U, C, P) and a reducer R in it are supposed to be given (however, we shall not need conditions (gs2), (gs3), (r3) and (r4) in this section, and the first inequality in (gs1) will be not used too). For any u in U, the corresponding complexity $C(u)$ will be denoted by $|u|$. We adopt the following definitions.

Definition 3.1. For any element u of U, the corresponding maximal sequence (finite or infinite)

$$u, R(u), R^2(u), R^3(u), \ldots$$

will be called *the trajectory of* u.

Definition 3.2. An element u of U will be called *strongly cyclic* if u is a leading part of some other member of its trajectory. The element u will be called *strongly cyclic with period* n if n is a positive integer, u belongs to $\mathrm{dom}(R^n)$ and u is a leading part of $R^n(u)$.

Definition 3.3. An element u of U will be called *cyclic* (*with period* n) if u has some strongly cyclic leading part (some leading part strongly cyclic with period n).

Definition 3.4. It will be said that *a cyclic loop* (*with period* n) *is present* in the trajectory of a given element of U if this trajectory has some cyclic member (some member cyclic with period n).

Remark 3.1. Clearly, any strongly cyclic element of U is cyclic. By (r1), the strongly cyclic elements of U have non–zero complexities. By Remark 2.2, this remains true for all cyclic elements.

Now some examples to the last definition follow.

Example 3.1. As we noted, each element of U is the only leading part of itself in the situation from Example 2.1, therefore (for any R) the presence of a cyclic loop in a trajectory is equivalent to the ultimate periodicity of the trajectory in this particular case.

Example 3.2. In the situation described in Examples 2.2 and 2.5, a cyclic loop is present in a trajectory if and only if a periodic loop in the sense of [10] is present in it.

Example 3.3. Let (U, C, P) be the concrete goal structure of the form considered in Example 2.3 with the non–negative integers used as atoms and with $\Delta = \{\mathrm{f}, \mathrm{g}\}$, where f is binary, g is unary and no other functional symbols are used. Let R be the reducer of the form from Example 2.6 corresponding to the following mapping of U_1 into U:

$$\mathrm{f}(2z, y) \mapsto z + 1, \quad \mathrm{f}(2z + 1, y) \mapsto \mathrm{f}(\mathrm{f}(z, y), \mathrm{g}(\mathrm{f}(y, z))), \quad \mathrm{g}(t) \mapsto t + 1.$$

Then a cyclic loop with period 11 is present in the trajectory of the element $\mathrm{f}(1, 7)$ of U, since the member $R^3(\mathrm{f}(1, 7))$ of this trajectory is cyclic with period 11. In fact, $R^3(\mathrm{f}(1, 7))$ is $\mathrm{f}(1, \mathrm{f}(\mathrm{f}(3, 0), \mathrm{g}(\mathrm{f}(0, 3))))$, and the leading part $\mathrm{f}(3, 0)$ of this term is strongly cyclic with period 11, as seen from the fact that $R^{11}(\mathrm{f}(3, 0))$ is $\mathrm{f}(\mathrm{f}(1, \mathrm{g}(\mathrm{f}(3, 0))), \mathrm{g}(\mathrm{f}(0, 1)))$.

We shall give also a modification of the above example in order to make use of the permission not all function symbols to be included in Δ.

Example 3.4. Let (U, C, P) be the concrete goal structure of the form considered in Example 2.3 with 0 as the only atom, with the two function symbols f and g having the same arities as in the above example and with $\Delta = \{\mathrm{f}\}$. Let R be the reducer of the form from Example 2.6 corresponding to the following mapping of U_1 into U, where $\mathbf{t}$ means $\mathrm{g}(\mathrm{g}(\ldots \mathrm{g}(\mathrm{g}(0)) \ldots))$ with t times g:

$$\mathrm{f}(\mathbf{2z}, \mathbf{y}) \mapsto \mathbf{z} + \mathbf{1}, \quad \mathrm{f}(\mathbf{2z} + \mathbf{1}, \mathbf{y}) \mapsto \mathrm{f}(\mathrm{f}(\mathbf{z}, \mathbf{y}), \mathrm{g}(\mathrm{f}(\mathbf{y}, \mathbf{z}))).$$

Then a cyclic loop with period 9 is present in the trajectory of the element $\mathrm{f}(\mathbf{1}, \mathbf{7})$ of U.

Our general study of cyclic loops starts with a generalization of condition (r2).

Proposition 3.1. Let k be any non–negative integer. Then all u in $\mathrm{dom}(RP^k)$ satisfying the inequality $|RP^k(u)| > 0$ belong to $\mathrm{dom}(P^kR)$, and $RP^k(u) = P^kR(u)$ holds for them.

Proof:
We shall proceed by induction on k. The case of $k = 0$ is trivial. Suppose now the statement of the proposition is true for some given non–negative integer k, and let $u \in \mathrm{dom}(RP^{k+1})$, $|RP^{k+1}(u)| > 0$. By applying the induction hypothesis to $P(u)$, we get the conclusion that $P(u) \in \mathrm{dom}(P^kR)$, hence $RP(u) \in \mathrm{dom}(P^k)$, and the equality $RP^{k+1}(u) = P^kRP(u)$ holds. The last equality together with Remark 2.2 shows that $|RP^{k+1}(u)| \leq |RP(u)|$, and hence $|RP(u)| > 0$. By (r2), we conclude that $u \in \mathrm{dom}(PR)$ and the equality $RP(u) = PR(u)$ holds. Therefore $PR(u) \in \mathrm{dom}(P^k)$, hence $u \in \mathrm{dom}(P^{k+1}R)$, and $RP^{k+1}(u) = P^kPR(u) = P^{k+1}R(u)$.

From the above proposition, a useful property of strongly cyclic elements will be derived now.

Proposition 3.2. Let u be an element of U strongly cyclic with period n. Then $u \in \mathrm{dom}(R)$, and $R(u)$ is also strongly cyclic with period n.

Proof:
By definition, n is a positive integer, $u \in \mathrm{dom}(R^n)$ and u is a leading part of $R^n(u)$, i.e. $u = P^k R^n(u)$ for some non–negative integer k. Clearly, $u \in \mathrm{dom}(R)$, and $R(u) = RP^k R^n(u)$. We assert that $|R(u)| > 0$, and therefore $|RP^k R^n(u)| > 0$. If $n > 1$ this is a consequence of (r1), since then $R(u) \in \mathrm{dom}(R)$. On the other hand, by Remark 2.2, we have the inequality $|u| \leq |R^n(u)|$. If $n = 1$ the last inequality is $|u| \leq |R(u)|$, and we get $|R(u)| > 0$ again, since $|u| > 0$ by (r1). By Proposition 3.1, we conclude that $R^n(u) \in \mathrm{dom}(P^k R)$, and the equality $RP^k R^n(u) = P^k R R^n(u)$ holds. Hence $R(u) = P^k R^{n+1}(u) = P^k R^n(R(u))$, i.e $R(u)$ is a leading part of $R^n(R(u))$.

Now we shall see that the above proposition remains valid after replacing "strongly cyclic" by "cyclic".

Proposition 3.3. Let u be an element of U cyclic with period n. Then $u \in \mathrm{dom}(R)$, and $R(u)$ is also cyclic with period n.

Proof:
By definition, there is a non–negative integer k such that $u \in \mathrm{dom}(P^k)$ and the element $P^k(u)$ is strongly cyclic with period n. By Proposition 3.2, $P^k(u) \in \mathrm{dom}(R)$, and $RP^k(u)$ is also strongly cyclic with period n. Again by Proposition 3.2, $RP^k(u) \in \mathrm{dom}(R)$, and hence, by (r1), $|RP^k(u)| > 0$. Therefore, by Proposition 3.1, $u \in \mathrm{dom}(P^k R)$, and the equality $RP^k(u) = P^k R(u)$ holds. Thus $u \in \mathrm{dom}(R)$, and $R(u)$ has a leading part strongly cyclic with period n, namely $RP^k(u)$.

Corollary 3.1. If there is a cyclic loop in the trajectory of a given element of U then this trajectory is infinite.

Of course the converse statement of the above corollary is not generally true - a trajectory may be infinite without having the periodic feature corresponding to the presence of a cyclic member in it.

4. Some technicalities

In this section, certain propositions will be proved that have a more or less technical character. In order to get some intuitive feeling about their content, the reader could check for instance what they do mean in the propositional case of the logic programming situation sketched in Examples 2.2 and 2.5 above.

Proposition 3.1 (generalizing (r2)) will be further generalized as follows.

Proposition 4.1. Let n, k be any non–negative integers. Then all u in $\mathrm{dom}(R^n P^k)$ satisfying the inequality $|R^n P^k(u)| > 0$ belong to $\mathrm{dom}(P^k R^n)$, and $R^n P^k(u) = P^k R^n(u)$ holds for them.

Proof:
We fix k and proceed by induction on n. The case of $n = 0$ is trivial. Let n be any non–negative integer such that the statement of the proposition is true for it, and let u be any element of $\mathrm{dom}(R^{n+1} P^k)$ satisfying the inequality $|R^{n+1} P^k(u)| > 0$. Then $u \in \mathrm{dom}(R^n P^k)$, and, by (r1), $|R^n P^k(u)| > 0$. Hence, by the induction hypothesis, $u \in \mathrm{dom}(P^k R^n)$, and $R^n P^k(u) = P^k R^n(u)$ holds. From here, the equality $R^{n+1} P^k(u) = RP^k R^n(u)$ follows. Thus $R^n(u) \in \mathrm{dom}(RP^k)$, and the inequality $|RP^k(R^n(u))| > 0$ holds. By Proposition 3.1, this implies $R^n(u) \in \mathrm{dom}(P^k R)$, $RP^k(R^n(u)) = P^k R(R^n(u))$. Therefore $u \in \mathrm{dom}(P^k R^{n+1})$, and the equality $R^{n+1} P^k(u) = P^k R^{n+1}(u)$ holds.

Taking into account the definition of head, one can derive the following conclusion from the above proposition:

Corollary 4.1. Let n be any non–negative integer, and u be an element of U such that $H(u) \in \mathrm{dom}(R^n)$ and $|R^n(H(u))| > 0$. Then $H(R^n(H(u))) = H(R^n(u))$.

We continue with generalizations of conditions (r3) and (r4) of the same kind as the generalization of (r2) in Proposition 3.1

Proposition 4.2. For any non–negative integer k, the equality

$$\mathrm{dom}(RP^k) = \mathrm{dom}(R) \cap \mathrm{dom}(P^k)$$

holds.

Proof:
We proceed by induction on k. The case of $k = 0$ is trivial. Suppose now the equality holds for some given non–negative integer k. We have to prove then the equality

$$\mathrm{dom}(RP^{k+1}) = \mathrm{dom}(R) \cap \mathrm{dom}(P^{k+1}).$$

Consider first any u belonging to the left-hand side of this equality. Then clearly u belongs to $\mathrm{dom}(P^{k+1})$. Since $P(u) \in \mathrm{dom}(RP^k)$, the induction hypothesis yields the conclusion that $P(u) \in \mathrm{dom}(R)$, hence $u \in \mathrm{dom}(RP)$. Therefore, by (r3), $u \in \mathrm{dom}(R)$, and, consequently, u belongs to the right-hand side of the equality in question. Now consider any u belonging to the right-hand side of this equality. In order to apply the induction hypothesis, we shall show that $P(u) \in \mathrm{dom}(R) \cap \mathrm{dom}(P^k)$. Since $P(u)$ obviously belongs to $\mathrm{dom}(P^k)$, we have only to show that $P(u) \in \mathrm{dom}(R)$, and this can be done by noticing that $u \in \mathrm{dom}(R) \cap \mathrm{dom}(P)$ and using (r3) to conclude that $u \in \mathrm{dom}(RP)$. The application of the induction hypothesis gives the conclusion that $P(u) \in \mathrm{dom}(RP^k)$, i.e. $u \in \mathrm{dom}(RP^{k+1})$.

Corollary 4.2. For any u in U, if some of the elements u and $H(u)$ belongs to $\mathrm{dom}(R)$ then the other also does.

Proposition 4.3. Let k be any non–negative integer. Then

$$|R(u)| - |RP^k(u)| = |u| - |P^k(u)|$$

holds for all u in $\mathrm{dom}(RP^k)$.

Proof:
We again proceed by induction on k. The case of $k = 0$ is trivial again. Let k be any non–negative integer such that the above equality holds for all u in $\mathrm{dom}(RP^k)$. Consider now any u in $\mathrm{dom}(RP^{k+1})$. Since $P(u) \in \mathrm{dom}(RP^k)$, the application of the induction hypothesis to $P(u)$ yields the equality

$$|RP(u)| - |RP^{k+1}(u)| = |P(u)| - |P^{k+1}(u)|.$$

This equality together with the one from (r4) implies the desired conclusion

$$|R(u)| - |RP^{k+1}(u)| = |u| - |P^{k+1}(u)|.$$

The above proposition together with Remark 2.3 and Corollary 4.2 leads to the following conclusion:

Corollary 4.3. Let $u \in \mathrm{dom}(R)$. Then

$$|R(u)| - |R(H(u))| = |u| - 1, \quad |R(u)| - |u| = |R(H(u))| - 1, \quad |R(u)| - |u| \geq -1.$$

Now the question arises if Propositions 4.2 and 4.3 could be generalized in the way Proposition 4.1 generalizes Proposition 3.1. The first of the mentioned results will be generalized only partially as follows.

Proposition 4.4. For any non–negative integers n, k, the inclusion

$$\mathrm{dom}(R^n P^k) \subseteq \mathrm{dom}(R^n) \cap \mathrm{dom}(P^k)$$

holds.

Proof:
Of course, it is sufficient to prove the inclusion $\mathrm{dom}(R^n P^k) \subseteq \mathrm{dom}(R^n)$. To prove it, we fix k and proceed by induction on n. The case of $n = 0$ is trivial, and the validity of the inclusion for $n = 1$ follows from Proposition 4.2. Suppose now n is a positive integer such that the above inclusion holds, and consider any element u of $\mathrm{dom}(R^{n+1}P^k)$. Then $u \in \mathrm{dom}(RP^k)$ and $RP^k(u) \in \mathrm{dom}(R)$, hence, by (r1), the inequality $|RP^k(u)| > 0$ holds. Therefore, by Proposition 3.1, u belongs to $\mathrm{dom}(P^k R)$ and the equality $RP^k(u) = P^k R(u)$ holds. Hence $R(u) \in \mathrm{dom}(R^n P^k)$, and, by the induction hypothesis, this implies $R(u) \in \mathrm{dom}(R^n)$, i.e. $u \in \mathrm{dom}(R^{n+1})$.

Corollary 4.4. For any u in U and any non–negative integer n, if $H(u)$ belongs to $\mathrm{dom}(R^n)$ then u also does.

Remark 4.1. The inclusion in the above proposition cannot generally be replaced by equality. To see this, consider the goal structure (U, C, P) and the reducer R from Example 3.3, and denote by u the element $\mathrm{f}(\mathrm{f}(0,0),0)$ of U. Then $u \in \mathrm{dom}(R^2) \cap \mathrm{dom}(P)$, but $u \notin \mathrm{dom}(R^2 P)$. Since $H(u) = P(u)$ for this u, we see also that $H(u) \notin \mathrm{dom}(R^2)$, hence the converse implication to the one in the corollary after the proposition also does not hold.

Proposition 4.3 will be, however, completely generalized.

Proposition 4.5. Let n, k be any non–negative integers. Then

$$|R^n(u)| - |R^n P^k(u)| = |u| - |P^k(u)|$$

holds for all u in $\mathrm{dom}(R^n P^k)$.

Proof:
We fix k and proceed by induction on n. The case of $n = 0$ is trivial. Suppose now n is a non–negative integer such that the statement is true for it, and let u be any element of $\mathrm{dom}(R^{n+1}P^k)$. Then $u \in \mathrm{dom}(R^n P^k)$, and the induction hypothesis is applicable to u. On the other hand, $R^n P^k(u) \in \mathrm{dom}(R)$, hence, by (r1), the inequality $|R^n P^k(u)| > 0$ holds. Then, by Proposition 4.1, u belongs to $\mathrm{dom}(P^k R^n)$, and the equality $R^n P^k(u) = P^k R^n(u)$ holds. Therefore $R^n(u) \in \mathrm{dom}(RP^k)$ and, by Proposition 4.3, the equality

$$|R^{n+1}(u)| - |RP^k R^n(u)| = |R^n(u)| - |P^k R^n(u)|$$

holds. From this equality, the previous one and the equality obtained by applying the induction hypothesis to u, we get

$$|R^{n+1}(u)| - |R^{n+1}P^k(u)| = |R^n(u)| - |R^n P^k(u)| = |u| - |P^k(u)|.$$

Corollary 4.5. Let n be any non–negative integer, and u be an element of U such that $H(u) \in \mathrm{dom}(R^n)$. Then

$$|R^n(u)| - |R^n(H(u))| = |u| - 1, \quad |R^n(u)| - |u| = |R^n(H(u))| - 1, \quad |R^n(u)| - |u| \geq -1.$$

As we showed in Remark 4.1, the inclusion $\mathrm{dom}(R^n) \cap \mathrm{dom}(P^k) \subseteq \mathrm{dom}(R^n P^k)$ does not generally hold. However, we shall prove now a statement allowing sometimes to conclude that certain elements of the left-hand side set belong also to the right-hand one.

Proposition 4.6. Let n be a positive integer, and k be a non–negative one. Then, whenever an element u of $\mathrm{dom}(R^n) \cap \mathrm{dom}(P^k)$ satisfies the inequality $|R^i(u)| \geq |u|$ for all positive integers i less than n, this element belongs to $\mathrm{dom}(R^n P^k)$.

Proof:
We fix k and proceed by induction on n. If $n = 1$ then the statement follows from Proposition 4.2. Consider now any positive integer n such that the statement is true for it. Let u be an element of $\mathrm{dom}(R^{n+1}) \cap \mathrm{dom}(P^k)$ satisfying the inequality $|R^i(u)| \geq |u|$ for all positive integers i less than $n+1$. Clearly $u \in \mathrm{dom}(R^n) \cap \mathrm{dom}(P^k)$, and $|R^i(u)| \geq |u|$ for all positive integers i less than n, hence by the induction hypothesis, $u \in \mathrm{dom}(R^n P^k)$. Therefore the equality from Proposition 4.5 holds, hence $|R^n P^k(u)| = |R^n(u)| - |u| + |P^k(u)|$. But $|R^n(u)| - |u| \geq 0$ according to the assumptions about u, and $|P^k(u)| > 0$ by (r1). Consequently, $|R^n P^k(u)| > 0$ and, by Proposition 4.1, $u \in \mathrm{dom}(P^k R^n)$, $R^n P^k(u) = P^k R^n(u)$. Thus $R^n(u) \in \mathrm{dom}(R) \cap \mathrm{dom}(P^k)$ and, therefore, by Proposition 4.2, $R^n(u) \in \mathrm{dom}(RP^k)$. Hence $P^k R^n(u) \in \mathrm{dom}(R)$, and, taking into account the last equality above, we see that u belongs to $\mathrm{dom}(R^{n+1} P^k)$.

Corollary 4.6. Let n be any positive integer. Then, whenever an element u of $\mathrm{dom}(R^n)$ satisfies the inequality $|R^i(u)| \geq |u|$ for all positive integers i less than n, then $H(u)$ also belongs to $\mathrm{dom}(R^n)$.

Now we shall prove a result more explicitly connected with cyclic loops.

Proposition 4.7. Let u be an element of U cyclic with period n. Then $|R^n(u)| \geq |u|$ and $H(R^n(u)) = H(u)$.

Proof:
Let k be a non–negative integer such that $P^k(u)$ is strongly cyclic with period n. Then $P^k(u)$ belongs to $\mathrm{dom}(R^n)$ and is a leading part of $R^n P^k(u)$. By Remark 2.2, this implies the inequality $|P^k(u)| \leq |R^n P^k(u)|$. On the other hand, Proposition 4.5 yields the equality

$$|R^n(u)| - |R^n P^k(u)| = |u| - |P^k(u)|.$$

Hence

$$|R^n(u)| - |u| = |R^n P^k(u)| - |P^k(u)| \geq 0.$$

By Remark 3.1, $|P^k(u)| > 0$ holds, therefore $|R^n P^k(u)| > 0$ too. Thus, by Proposition 4.1, $u \in \mathrm{dom}(P^k R^n)$, and $R^n P^k(u) = P^k R^n(u)$. Consequently, $P^k(u) = P^{k+l} R^n(u)$ for some non–negative integer l, and therefore $H(u) = H(R^n(u))$.

Corollary 4.7. If a cyclic loop with period n is present in a trajectory then the sequence of the heads of its members is ultimately periodic with period n.

We shall show in the next section that the converse statement of the above corollary is also true. However, the proposition preceding the corollary does not allow inversion: even a finite trajectory may contain distinct members with the same complexities and the same heads (consider for example the trajectory corresponding to execution of the propositional Prolog program with clauses

$$\mathrm{p} :- \mathrm{q}, \mathrm{r}. \qquad \mathrm{q}. \qquad \mathrm{r}.$$

on the query $? - \mathrm{q}, \mathrm{p}$).

To make the presentation more complete, we add next proposition showing that, whenever a cyclic loop with period n is present in a trajectory, then the difference $|R^n(u)| - |u|$ is constant for the members u of the trajectory from some place on (the proposition will be not used in the sequel).

Proposition 4.8. Let u be an element of U cyclic with period n. Then

$$|R^{n+1}(u)| - |R(u)| = |R^n(u)| - |u|.$$

Proof:
By Corollary 4.3, we have the equalities

$$|R(u)| - |u| = |R(H(u))| - 1, \quad |R^{n+1}(u)| - |R^n(u)| = |R(H(R^n(u)))| - 1.$$

Hence, by Proposition 4.7, $|R^{n+1}(u)| - |R^n(u)| = |R(u)| - |u|$.

5. Primitive cyclic elements

We shall specify now a certain kind of cyclic elements that are easier to be observed in a trajectory where they occur.

Definition 5.1. An element u of U will be called *primitive cyclic* (*with period* n) if the head of u is cyclic (with period n).

Remark 5.1. Since $H(u)$ is the only leading part of itself, one could replace "cyclic" by "strongly cyclic" in the above definition. Clearly, any primitive cyclic element of U is cyclic (with the same period). In general, a primitive cyclic element is not necessarily strongly cyclic.

Example 5.1. In the situation from Example 3.3, the term $R^3(\mathrm{f}(1,7))$ is in fact primitive cyclic with period 11, since the mentioned there its leading part $\mathrm{f}(3,0)$, strongly cyclic with period 11, is actually its head. By Proposition 3.2, the term $R^2(\mathrm{f}(3,0))$ is also strongly cyclic with period 11, but this term is not primitive cyclic because its head is $\mathrm{f}(0,0)$, and $\mathrm{f}(0,0)$ does not belong to $\mathrm{dom}(R^2)$, hence is not cyclic.

Remark 5.2. The above example shows that $R(u)$ is not always a primitive cyclic element when u is such one. However, if u is primitive cyclic with period n then $R^n(u)$ is also primitive cyclic with period n, since, according to Proposition 4.7, both mentioned elements have one and the same head.

Next proposition states a notable property of primitive cyclic elements.

Proposition 5.1. Let u be a primitive cyclic element of U. Then $|R^i(u)| \geq |u|$ for all positive integers i.

Proof:
Since $H(u)$ is cyclic, $H(u)$ belongs to $\mathrm{dom}(R^i)$ for any positive integer i. Then, by Corollary 4.5 and condition (r1), we have

$$|R^i(u)| - |u| = |R^i(H(u))| - 1 \geq 0.$$

Now a useful complete characterization of the primitive cyclic elements will be given.

Proposition 5.2. Let u be an element of U, and n be a positive integer. The element u is primitive cyclic with period n if and only if u belongs to $\mathrm{dom}(R^n)$ and satisfies the inequalities $|R^i(u)| \geq |u|$, $i = 1, 2, \ldots, n$, as well as the equality $H(R^n(u)) = H(u)$.

Proof:
If u is primitive cyclic with period n then the formulated conditions are fulfiled by the above proposition and Proposition 4.7. For the reasoning in the opposite direction, suppose these conditions are fulfiled. Then, by the inequalities with $i < n$ and Corollary 4.6, $H(u) \in \mathrm{dom}(R^n)$, hence, by Corollary 4.5, the equality

$$|R^n(u)| - |R^n(H(u))| = |u| - 1$$

holds. Making use of the inequality with $i = n$, we get

$$|R^n(H(u))| = |R^n(u)| - |u| + 1 > 0.$$

Thus, by Corollary 4.1 and the assumed equality, we have

$$H(R^n(H(u))) = H(R^n(u)) = H(u),$$

hence $H(u)$ is strongly cyclic with period n.

Making use of the above proposition and of Proposition 4.7, we get

Corollary 5.1. Let u be an element of U cyclic with period n, and let the inequalities $|R^i(u)| \geq |u|$, $i = 1, 2, \ldots, n-1$ hold. Then u is primitive cyclic with period n.

Of course, this corollary implies the converse statement of Proposition 5.1 under the additional assumption that u is cyclic.

The next corollary is a strengthened form of the converse statement of Corollary 4.7.

Corollary 5.2. If a trajectory is infinite, and the sequence of the heads of its members is ultimately periodic with period n then some member of the trajectory is primitive cyclic with period n.

To prove this corollary, one could apply Proposition 5.2 to a member with the least complexity in a tail of the trajectory where the heads of the members already repeat with period n. We shall prove also the following more interesting result.

Proposition 5.3. Let a trajectory be infinite, and let the sequence of the heads of its members contain only finitely many distinct members. Then some member of the trajectory is primitive cyclic.

Proof:
Since all complexities are non–negative integers, there are infinitely many among the members of the trajectory with the property that no subsequent member has a smaller complexity. Then there are two members with this property having distinct subscripts, but one and the same head. To complete the proof, it is sufficient to apply Proposition 5.2 taking u to be the one of the mentioned two members with the smaller subscript and n to be the difference of their subscripts.

Corollary 5.3. Suppose the given goal structure is such that only finitely many elements of U with complexity 1 exist. Then no infinite trajectory may exist without a cyclic loop present in it.

According to the above corollary, if its assumption is satisfied then detecting the infinitity of a given trajectory is equivalent to detecting the presence of a cyclic loop in this trajectory. We note the following two instances when the mentioned assumption is satisfied:

- The goal structure (U, C, P) corresponds (in the way described in Example 2.2) to a logic program without terms other than constants and variables.

- The goal structure (U, C, P) is constructed in the way from Example 2.3 assumed the set of all atoms and all function symbols is finite, and all function symbols are in Δ.

Of course in the most general case the presence of a cyclic loop in a trajectory is only a sufficient condition for the infiniteness of the trajectory. In any case, it would be convenient to have a method that detects the presence of a cyclic loop in a trajectory in a easier way than a direct application of the corresponding definition.

Proposition 5.3 and the corollary preceding it give certain sufficient conditions for the presence of a cyclic loop in a given trajectory, and, by Corollary 4.7, these conditions are also necessary. Unfortunately, they are not directly verifiable (especially unpleasant in this respect is the required infiniteness of the trajectory, since presence of a cyclic loop is interesting mainly as a sufficient condition for such an infiniteness). The existence of a primitive cyclic member of the trajectory is another necessary and sufficient condition that is more useful for the detection of cyclic loops. The necessity of this condition follows from Corollaries 4.7 and 5.2 (instead of the second of them, also Proposition 5.3 could be used). The sufficiency is obvious, and the usefulness will be shown in the next section, where a loop detection method will be presented.

We shall complete the present section by giving a more constructive version of the above formulated statement that existence of a primitive cyclic member of a trajectory is necessary for the presence of a cyclic loop in it. The additional information in this version of the statement will be essential for the details of the loop detection method.

Proposition 5.4. Let u be an element of U cyclic with period n, and let v be one of the elements $u, R(u), R^2(u), \ldots, R^{n-1}(u)$ having the least complexity among them. Then v is primitive cyclic with period n.

Proof:
By Proposition 3.3, v is also cyclic with period n, and, clearly, $v \in \mathrm{dom}(R^n)$. By Proposition 4.7, $|R^n(v)| \geq |v|$ and $H(R^n(v)) = H(v)$. In view of this and Proposition 5.2, it is sufficient to prove the inequalities $|R^i(v)| \geq |v|$, $i = 1, 2, \ldots, n-1$. Let $v = R^m(u)$, where $m < n$, and let i be any of the numbers $1, 2, \ldots, n-1$. Then $R^i(v) = R^{m+i}(u)$. If $m + i < n$ then $|R^i(v)| \geq |v|$, since $R^i(v)$ is some of the n elements mentioned in the proposition. Otherwise $R^i(v) = R^n(R^{m+i-n}(u))$, and $R^{m+i-n}(u)$ is again some of those elements, hence $|R^{m+i-n}(u)| \geq |v|$. Since $R^{m+i-n}(u)$ is also cyclic with period n, applying Proposition 4.7 once more, we get the inequality $|R^i(v)| \geq |R^{m+i-n}(u)|$, and we conclude that the inequality $|R^i(v)| \geq |v|$ holds again.

6. A method for the detection of cyclic loops

The method is based on the fact that in any trajectory with a cyclic loop present in it, primitive cyclic members appear periodically infinitely many times from some place on. The method consists of activities that aim at finding someone of these members. To fix a concrete variant of the method, one must choose an infinite subset $\mathcal{T}$ of $\mathbf{N}$ with arbitrarily long gaps between consecutive numbers in it (for example, the set of all exponents of 2).[3] The following Pascal-style program schematically describes the application of the method to the trajectory of a given element u of U.

[3]Concrete subsets of $\mathbf{N}$ with this property are used in Brent's loop detection method mentioned in the introduction, and arbitrary ones in its generalization in [5].

```
var t:N; h:U \ dom(P); c:N; z:U;
begin
  t:=0; z:=u;
  while not (t in T) do begin t:=t+1; z:=R(z) end;
  repeat
    if (t in T) or (c=0) then begin h:=H(z); c:=1 end;
    t:=t+1; c:=c+|R(H(z))|-1; z:=R(z)
  until (c>0) and (H(z)=h)
end.
```

The application of the method presented by the above program can be characterized as constructing consecutive trajectory members (as values of the variable `z`) together with carrying out certain loop detection activities. More concretely, the head of the last constructed trajectory member has to be saved in the variable `h` at appropriate moments and to be used further for the loop detection during a certain period of time, and this must be repeated until possibly a trajectory member not belonging to $\mathrm{dom}(R)$ is reached or a cyclic loop is detected. The moments for saving are the ones when the value of the variable `t` (i.e. the number of the constructed trajectory members after the initial one) gets into $\mathcal{T}$ and those ones at which, so to say, the saved head becomes obsolete. The last happens in the case of a value 0 of the variable `c` maintaining the complexity of the result that would be obtained from the saved head after the moment of its saving by a parallel application of R the same number of times. Except for the cases when it is obsolete, the saved head is used for the loop detection in the following way: the head of the last constructed trajectory member is compared with the saved head, and if they turn out to be equal, then a cyclic loop is detected in the trajectory of u.

The execution of this program has a normal termination if and only if a cyclic loop is present in the trajectory of u. The value of the variable `z` after such a termination is equal to some of the primitive cyclic trajectory members. The other case of termination of the program execution is the case when the trajectory of u is finite (then some assignment of the form `z:=R(z)` will turn out to be impossible at a certain stage of the execution). The value of `z` after a termination of this kind is equal to the last of all trajectory members. Of course no cyclic loop is present in the trajectory in this case.

To give a mathematical description of the method, we introduce the notion of a *detection state*. By definition, this is an arbitrary quadruple (t, h, c, z), where t and c are non–negative integers, h is an element of $U \setminus \mathrm{dom}(P)$, and z belongs to U. It will be said that (t, h, c, z) *detects a cyclic loop* if $c > 0$ and $H(z) = h$. A detection state will be said to be a *closing* one if its last component does not belong to $\mathrm{dom}(R)$ or a loop is detected by this state.

The application of the detection method starts with consecutively constructing the trajectory members $R^i(u)$ with $i \leq t_0 + 1$, where t_0 is the least number in $\mathcal{T}$. If it turns out that $u \notin \mathrm{dom}(R^{t_0+1})$ then the trajectory of u is finite and no cyclic loop is present in it. Otherwise we form the detection state $(t_0 + 1, H(z_0), |R(H(z_0))|, R(z_0))$, where $z_0 = R^{t_0}(u)$, and, if this detection state is not a closing one, we consecutively construct further detection states in a certain appropriate way, until possibly a closing detection state is reached (after a closing detection state is reached the construction of detection states terminates). Namely, whenever a detection state (t, h, c, z) is constructed, and this state is not a closing one, we form a *next detection state* (t', h', c', z') specified as follows:

(a) $t' = t + 1$ and $z' = R(z)$.

(b) If $t \in \mathcal{T}$ or $c = 0$ then $h' = H(z)$ and $c' = 1 + d$, otherwise $h' = h$ and $c' = c + d$, where $d = |R(H(z))| - 1$ (it will be said that (t, h, c, z) *invokes saving* when $t \in \mathcal{T}$ or $c = 0$).

Remark 6.1. By Corollary 4.3, the equality for d in (b) can be replaced by $d = |z'| - |z|$.

The sequence of detection states defined in the above way will be called *the detection path of* u. Clearly, the members of this sequence have first components greater than t_0, and if two members have equal first components then these members coincide. Note also that, whenever the detection path of u contains a member with first component t, then each integer strictly between t_0 and t is also the first component of some member of the detection path.

Example 6.1. In the situation from Example 3.3, we shall construct the detection path of the term $\mathrm{f}(1,7)$ taking $\mathcal{T}$ to be the set of the Fibonacci numbers with even subscripts: $0, 1, 3, 8, 21, 55, \ldots$ [4] The detection path looks as follows:

(1, $\mathrm{f}(1,7)$, 4, $\mathrm{f}(\mathrm{f}(0,7),\mathrm{g}(\mathrm{f}(7,0)))$),
(2, $\mathrm{f}(0,7)$, 0, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(7,0)))$),
(3, $\mathrm{f}(7,0)$, 4, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(3,0),\mathrm{g}(\mathrm{f}(0,3))))))$),
(4, $\mathrm{f}(3,0)$, 4, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,0),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(5, $\mathrm{f}(3,0)$, 7, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{f}(0,0),\mathrm{g}(\mathrm{f}(0,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(6, $\mathrm{f}(3,0)$, 6, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(0,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(7, $\mathrm{f}(3,0)$, 5, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(1)),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(8, $\mathrm{f}(3,0)$, 4, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,2),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(9, $\mathrm{f}(1,2)$, 4, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{f}(0,2),\mathrm{g}(\mathrm{f}(2,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(10, $\mathrm{f}(1,2)$, 3, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(2,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(11, $\mathrm{f}(1,2)$, 2, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(2)),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(12, $\mathrm{f}(1,2)$, 1, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,3),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(13, $\mathrm{f}(1,2)$, 4, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(\mathrm{f}(0,3),\mathrm{g}(\mathrm{f}(3,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(14, $\mathrm{f}(1,2)$, 3, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(3,0))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(15, $\mathrm{f}(1,2)$, 6, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(1,0),\mathrm{g}(\mathrm{f}(0,1))))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(16, $\mathrm{f}(1,2)$, 9, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(0,0),\mathrm{g}(\mathrm{f}(0,0))),\mathrm{g}(\mathrm{f}(0,1))))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(17, $\mathrm{f}(1,2)$, 8, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(0,0))),\mathrm{g}(\mathrm{f}(0,1))))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(18, $\mathrm{f}(1,2)$, 7, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(1)),\mathrm{g}(\mathrm{f}(0,1))))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$),
(19, $\mathrm{f}(1,2)$, 6, $\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(\mathrm{f}(1,\mathrm{g}(\mathrm{f}(\mathrm{f}(1,2),\mathrm{g}(\mathrm{f}(0,1))))),\mathrm{g}(\mathrm{f}(0,1))),\mathrm{g}(\mathrm{f}(0,3))))))$).

The path members with first components 1, 3 and 8 invoke saving, since these numbers belong to $\mathcal{T}$ and the mentioned members are not closing. The member with first component 2 also invokes saving due to the value 0 of its third component. The member with first component 19 detects a cyclic loop.[5]

We are going now to prove a correctness theorem and a completeness theorem for the described loop detection algorithm. We shall assume that an infinite subset $\mathcal{T}$ of $\mathbf{N}$ is given such that there are arbitrarily long gaps between consecutive members in it, and we shall denote by t_0 the least number in $\mathcal{T}$.

Two lemmas will be proved first.

Lemma 6.1. Let u be an element of $\mathrm{dom}(R^{t_0+1})$, and let (t,h,c,z) be a member of the detection path of u. Then $z = R^t(u)$ and there is some integer s satisfying the inequalities $t_0 \le s < t$, such that $h = H(R^s(u))$, $c = |R^t(u)| - |R^s(u)| + 1$, and $|R^r(u)| \ge |R^s(u)|$, whenever $s < r < t$.

Proof:
We proceed by induction on t. The least possible value of t is t_0+1, and if $t = t_0+1$ then we can take $s = t_0$ (the first equality involving s is true by the definition of the initial detection state, the validity of the second one is clear from Corollary 4.3, and the last condition concerning s is satisfied in a trivial way). Suppose now for some t not less than $t_0 + 1$

[4] A certain kind of optimality of this set is shown in [5] for the generalized Brent's loop detection method.

[5] The detection path of the same term could be shorter at some other choice of the set $\mathcal{T}$. For instance, a cyclic loop would be detected by a path member with first component 14 if we would use a set $\mathcal{T}$ such that $3 \in \mathcal{T}$, but $\mathcal{T}$ contains no number between 3 and 14.

the statement is true, and $(t+1, h', c', z')$ is a detection path member. This member is surely obtained from another one of the form (t, h, c, z) in the way specified by (a) and (b). By the induction hypothesis, $z = R^t(u)$ holds, and an s with the formulated properties can be found for the mentioned other member of the detection path. Then the equality $z = R^{t+1}(u)$ is clear. As to the existence statement, we have to reason by cases. In the case when (t, h, c, z) invokes saving the statement holds with a new value of s, namely t (the reasoning is similar to the one for the case of $t = t_0 + 1$). In the opposite case, we use the available value of s also for $t+1$. The first equality is unproblematic since $h' = h$, the equality $c' = |R^{t+1}(u)| - |R^s(u)| + 1$ immediately follows from the induction hypothesis and Remark 6.1, and the verification of the last condition uses the induction hypothesis and the fact that $c > 0$ in this case.

Remark 6.2. The equality concerning c may be used to justify the claim that the variable c in the presented Pascal-style program maintains the complexity of the result obtainable from the saved head by the appropriate number of applications of R. Namely, this equality together with Corollary 4.5 yields $c = |R^{t-s}(R^s(u))| - |R^s(u)| + 1 = |R^{t-s}(H(R^s(u)))|$.

Lemma 6.2. Let s and t be natural numbers such that $t_0 \leq s < t$, and there is no number from $\mathcal{T}$ strictly between s and t. Let u be an element of $\mathrm{dom}(R^t)$ such that the detection path of u contains a member with first component t, and $|R^r(u)| \geq |R^s(u)|$ holds for any integer r satisfying the inequalities $s < r < t$. In case of $s > t_0$, let also saving be invoked by the detection path member with first component s. Then the detection path member with first component t has the form $(t, H(R^s(u)), |R^t(u)| - |R^s(u)| + 1, R^t(u))$.

Proof:
We shall proceed by induction on t. The least possible value of t is $s+1$. If $t = s+1$ then $(t, H(R^s(u)), |R(H(R^s(u)))|, R^t(u))$ is a member of the detection path of u by the definition of next detection state and the definition of the initial member of the detection path, and we have $|R(H(R^s(u)))| = |R^t(u)| - |R^s(u)| + 1$ by Corollary 4.3. Suppose now the statement is true for some integer t greater than s, the assumptions concerning t being fulfiled with $t+1$ in the role of t (i.e. no number from $\mathcal{T}$ is strictly between s and $t+1$, u belongs to $\mathrm{dom}(R^{t+1})$, the detection path of u contains a member with first component $t+1$, and the inequality $|R^r(u)| \geq |R^s(u)|$ holds for any integer r satisfying the inequalities $s < r < t+1$). Then the induction hypothesis can be applied to conclude that

$$(t, H(R^s(u)), |R^t(u)| - |R^s(u)| + 1, R^t(u))$$

is a member of the detection path of u. This member does not invoke saving because $t \notin \mathcal{T}$ and $|R^t(u)| - |R^s(u)| + 1 > 0$. Therefore the next detection state for it is

$$(t+1, H(R^s(u)), |R^{t+1}(u)| - |R^s(u)| + 1, R^{t+1}(u))$$

(Remark 6.1 is used for representing the third component in the needed form).

Next two theorems are the correctness theorem and the completeness theorem for the detection method.

Theorem 6.1. Let u be an element of $\mathrm{dom}(R^{t_0+1})$, and let some member of the detection path of u detect a cyclic loop. Then a cyclic loop is present in the trajectory of u.

Proof:
Suppose a member (t, h, c, z) of the detection path of u detects a cyclic loop, i.e. $c > 0$ and $H(z) = h$. By Lemma 6.1, $z = R^t(u)$, and there is some integer s satisfying the inequalities $t_0 \leq s < t$, such that $h = H(R^s(u))$, $c = |R^t(u)| - |R^s(u)| + 1$, and $|R^r(u)| \geq |R^s(u)|$ holds, whenever $s < r < t$. The inequality and the equality concerning c show that

$|R^r(u)| \geq |R^s(u)|$ also for $r = t$. Let us set $w = R^s(u)$, $n = t - s$. Then $w \in \mathrm{dom}(R^n)$, $|R^i(w)| \geq |w|$ for $i = 1, 2, \ldots, n$, and $H(R^n(w)) = H(z) = H(w)$. By Proposition 5.2, the element w is primitive cyclic with period n.[6] Thus a cyclic loop with period n is present in the trajectory of u.

Theorem 6.2. Let u be an element of $\mathrm{dom}(R^{t_0+1})$, and let a cyclic loop be present in the trajectory of u. Then some member of the detection path of u detects a cyclic loop.

Proof:
Suppose a cyclic loop with period n is present in the trajectory of u (then clearly $u \in \mathrm{dom}(R^i)$ for any natural number i). Let j be a non–negative integer such that $R^j(u)$ is cyclic with period n, and let a and b be two consecutive elements of $\mathcal{T}$ satisfying the inequalities $a \geq j$, $b - a \geq 2n - 1$. We shall prove that the detection path of u contains some member having first component not greater than $a + 2n - 1$ and detecting a cyclic loop. To prove this, we suppose there is a detection path member with first component $a + 2n$ and we aim at getting a contradiction (the non-existence of such a member would imply the existence of a closing detection path member with first component not greater than $a + 2n - 1$, and due to the infiniteness of the trajectory of u such a member would necessarily detect a cyclic loop). Of course, the existence of a detection path member with first component $a + 2n$ implies the existence also of members with first components equal to arbitrary integers intermediate between a and $a + 2n$.

Let us set $w = R^a(u)$. Since $a \geq j$, the element w is cyclic with period n. We denote by m the first number i among $0, 1, 2, \ldots, n - 1$ such that $|R^i(w)|$ is the least one among the numbers $|w|, |R(w)|, |R^2(w)|, \ldots, |R^{n-1}(w)|$. By Proposition 5.4, the element $R^m(w)$ is primitive cyclic with period n, hence, by Proposition 5.1, all subsequent trajectory members have complexities not less than $|R^m(w)|$. Clearly $a < a + m + n < a + 2n$ holds, hence there is a member with first component $a + m + n$ in the detection path of u. We shall aim now at showing that the member in question has the form

$$(a + m + n, H(R^m(w)), |R^{m+n}(w)| - |R^m(w)| + 1, R^{m+n}(w)).$$

Since $a + m + n < a + 2n$, and such a member would detect a cyclic loop, this will be a contradiction.

The formulated statement about the form of the detection path member with first component $a + m + n$ will be proved by applying Lemma 6.2 with $s = a + m$, $t = a + m + n$ (the considered form is just the one from the conclusion of the lemma if we replace w by what it denotes). Thus things are reduced to a verification of the assumptions of Lemma 6.2 in this case. A look at these assumptions shows that all of them are immediately verifiable, except for the condition that saving is invoked by the detection path member with first component $a + m$ in the case of $a + m > t_0$.

If $m = 0$ then $a + m = a \in \mathcal{T}$, hence if $a + m > t_0$ then the detection state with first component s (being not closing) surely invokes saving. So it remains to consider the case when $m > 0$.

By the definition of m, the inequality $|R^r(w)| > |R^m(w)|$ holds for $r = 0, 1, 2, \ldots, m - 1$. Therefore a finite sequence $r_0, r_1, \ldots, r_p$ of integers can be found with the properties that $0 = r_0 < r_1 < \ldots < r_p = m$, and, for $l = 0, 1, \ldots, p - 1$, r_{l+1} is the least integer r, greater than r_l and such that $|R^r(w)| < |R^{r_l}(w)|$. If for some of the mentioned subscripts l it is known that $r_l = t_0$ or the detection path member with first component r_l invokes saving, then an application of Lemma 6.2 with $s = a + r_l$, $t = a + r_{l+1}$ shows that the detection path member with first component $a + r_{l+1}$ has the form

$$(a + r_{l+1}, H(R^{r_l}(u)), |R^{r_{l+1}}(u)| - |R^{r_l}(u)| + 1, R^{r_{l+1}}(u)),$$

[6] By Remark 5.2, the element z is also primitive cyclic with period n.

hence this member (being non–closing and having third component less than 1, i.e. equal to 0) surely invokes saving. Since $a+r_0 = a \in \mathcal{T}$, the above implication allows inductively to establish that saving is surely invoked by the detection path member with first component $a + r_l$ for $l = 1, 2, \ldots, p$. Applying this for $l = p$, we conclude that the detection path member with first component $a + m$ invokes saving.

7. Concluding remarks

The described loop detection method can be easily refined in order to give not only an indication about the presence of a cyclic loop, but also an information about the period of the loop. This can be done by additionally maintaining the number of applications of R after the last saving (the realization is by including a corresponding additional component in the detection states).

Some programming implementations of a variant of the method and of the method itself have been suggested in [12] and [11] – for Prolog programs and for recursive Pascal programs, respectively. The first of these implementation has been given by Igor Durdanović, and both use his idea to add the appropriate loop detection mechanism to the concrete programs by making accessible the heads and the complexities and leaving the total processing of the corresponding complex objects to the usual mechanisms of some existing compiler for the concerned programming language (in [12], a universal program transformer is given for doing this in the considered case, whereas in [11] the technique of the implementation is illustrated by example).

As we mentioned, the applicability of our loop detection method to Prolog is restricted to detection of some loops during depth-first search. In [13], a similar, but more complicated method has been described that can detect certain periodic loops during Prolog execution with backtracking. The abstract approach presented here can be modified to cover also that more complicated case. To achieve this, one has to assume a goal structure to be given, and to model the consecutive clauses of a Prolog program by means of a finite sequence of reducers in the given goal structure. We hope to be able to present the details in some subsequent paper.

8. Acknowledgments

The present paper is written as an expression of my profound respect for the memory of Professor Helena Rasiowa. Her works set an inspiring example how to apply non-trivial mathematics to essential problems of computer science. Professor Rasiowa's publications and the personal contacts with her had a serious influence on the orientation of my research, and I am deeply indebted to her for this.

I passed a considerable part of my way to the results in this paper during two stays in Germany – in 1990 in Duisburg and in 1992 in Paderborn, the second one sponsored by a DAAD grant. Both became possible by the kindness of Professor Hans Kleine Büning, and the contacts with him and his scientific group have been very useful for my work on loop detection. In particular, Igor Durdanović's idea mentioned above played an essential role for choosing the present direction of generalization.

Two participations in Warsaw Banach Center activities also strongly stimulated my research in the present field, namely the ones in the Semester on Algebraic Methods in Logic and in Computer Science held in 1991 and in the Memorial Symposium in 1996 on Logic, Algebra and Computer Science dedicated to Professor Helena Rasiowa. I heartily thank the organizers of these excellent scientific meetings for the invitations to me.

Thanks are due also to Professor Donald Knuth who called my attention to the paper [5] thus filling a serious gap in my knowledge.

References

[1] Bol, R. N., Apt, K. R. and Klop, J. W.: "An analysis of loop checking mechanisms for logic programs", *Theoretical Computer Science*, **86**, 1991, 35–79.

[2] Covington, M. A.: "Eliminating unwanted loops in logic programming", *SIGPLAN Notices*, **20**, No. 1, 1985, 20–26.

[3] Covington, M. A.: "A further note on looping in Prolog", *SIGPLAN Notices*, **20**, No. 8, 1985, 28–31.

[4] Ferrucci, F., Pacini, G. and Sessa, M. I.: "Redundancy elimination and loop checks for logic programs", *Information and Computation*, **119**, 1995, 137–153.

[5] Fich, F. E.: "Lower bounds for the cycle detection problem", *J. of Computer and System Sciences*, **26**, 1983, 392–409.

[6] Kleine Büning, H., Löwen, U. and Schmitgen, S.: "Loop detection in propositional Prolog programs". E. Börger, H. Kleine Büning and M. M. Richter (eds.), *CSL '88, 2nd Workshop on Computer Science Logic. Lecture Notes in Computer Science, vol. 385*, Springer, 1989, 148–165.

[7] Knuth, D. E.: *The Art of Computer Programming. Vol. 2: Seminumerical Algorithms*, Reading, Mass., Addison-Wesley, 1981.

[8] Nute, D.: "A programming solution to certain problems with loops in Prolog", *SIGPLAN Notices*, **20**, No. 8, 1985, 32–37.

[9] Poole, P. and Goebel, R.: "On eliminating loops in PROLOG", *SIGPLAN Notices*, **20**, No. 8, 1985, 38–40.

[10] Skordev, D.: "On the detection of some periodic loops during the execution of Prolog programs". C. Rauszer (ed.), *Algebraic Methods in Logic and in Computer Science. Banach Center Publications, vol. 28*, Inst. of Math., Polish Acad. of Sciences, Warsaw, 1993, 151–166.

[11] Skordev, D.: "On the detection of some loops in recursive computations.", *Annuaire de l'Univ. de Sofia, Fac. de Math. et Inform.*, **87**, Livre 1 – Math. (in print).

[12] Skordev, D. and Durdanović, I.: "Loop detection in Prolog by searching for primitive cyclic goals", Paderborn, 1992 (unpublished manuscript).

[13] Skordev, D. and Durdanović, I.: "Loop detection in Prolog in the case of possible backtracking", Paderborn, 1992 (unpublished manuscript).

[14] Smith, D. E., Genesereth, M. R. and Ginsberg, M. L.: "Controlling recursive inference", *Artificial Intelligence*, **30**, 1986, 343–389.

[15] Van Gelder, A.: "Efficient loop detection in Prolog using the tortoise-and-hare technique", *J. Logic Programming*, **4**, 1987, 23-31.

[16] Vieille, L.: "Recursive query processing: The power of logic", *Theoretical Computer Science*, **69**, 1989, 1-53.