



HAL
open science

The index and period of a logic program

Christian Antic

► **To cite this version:**

| Christian Antic. The index and period of a logic program. 2023. hal-04219244

HAL Id: hal-04219244

<https://hal.science/hal-04219244>

Preprint submitted on 27 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THE INDEX AND PERIOD OF A LOGIC PROGRAM

CHRISTIAN ANTIĆ

christian.antic@icloud.com
Vienna University of Technology
Vienna, Austria

ABSTRACT. This paper introduces the index and period of a logic program as an algebraic measure of its cyclicity.

1. INTRODUCTION AND PRELIMINARIES

We introduce the index and period of a logic program via the recently introduced sequential composition¹ operation on programs (Antić, 2023d, 2023c, 2023a) in analogy to the same concepts in semigroup theory (see, for example, Howie, 2003, p.10). Numerous examples demonstrate that these notions provide a suitable algebraization of the cyclicity of a program.

For simplicity, we demonstrate the concepts in this paper for propositional programs of a very simple (Krom) form only — the index and period of non-Krom and first-order answer set programs can be defined and studied in essentially the same way where the sequential composition of first-order programs is defined as in (Antić, 2023a, §3.1) and of arbitrary answer set programs as in Antić (2023c). We choose here to restrict ourselves to this simple fragment since it allows us to focus on the novel concepts without introducing too much formalism as the sequential composition of non-Krom answer set programs is quite involved.

We refer readers not familiar with answer set programming to Baral (2003), Brewka et al. (2011), Eiter et al. (2009), and Lifschitz (2019).

Preliminaries. For every $n \geq 1$, let $[n] := \{1, \dots, n\}$.

An (*answer set*) *program* (Gelfond & Lifschitz, 1991) over $[n]$ is a finite set of *rules* of the form

$$(1) \quad a_0 \leftarrow a_1, \dots, a_\ell, \text{not } a_{\ell+1}, \dots, \text{not } a_k, \quad 0 \leq \ell \leq k \leq n,$$

where $a_0, \dots, a_k \in [n]$ are atoms and *not* denotes *negation as failure* (Clark, 1978). We call a program *positive* (or *Horn*) iff it contains no negation, and we call it *negative* iff every body atom is negated.

For our demonstration in this paper it will suffice to consider only *Krom*² programs (cf. Krom, 1967) consisting of at most one body literal of the following forms:

$$a, \quad a \leftarrow b, \quad a \leftarrow \text{not } b.$$

Notation 1. In what follows, the implication relation “ \leftarrow ” binds weaker than any other operation.

¹For a discussion of the sequential composition operation used in this paper in relation to other notions of composition from the literature (e.g. Brogi et al., 1999; Bugliesi et al., 1994; O’Keefe, 1985) see Antić (2023d, §Related work).

²This class of programs is sometimes called *binary* in the literature.

We define the (*sequential*) *composition* (Antić, 2023d, 2023c) of two Krom answer set programs P and R by

$$P \circ R := \left\{ \begin{array}{l|l} a & a \in P \\ b & b \leftarrow b' \in P \text{ and } b' \in R \\ c \leftarrow d & c \leftarrow c' \in P \text{ and } c' \leftarrow d \in R \\ e \leftarrow f & e \leftarrow \text{not } e' \in P \text{ and } e' \leftarrow \text{not } f \in R \\ g \leftarrow \text{not } h & g \leftarrow g' \in P \text{ and } g' \leftarrow \text{not } h \in R \\ i \leftarrow \text{not } j & i \leftarrow \text{not } i' \in P \text{ and } i' \leftarrow j \in R \end{array} \right\}.$$

We will often abbreviate $P \circ R$ as PR .

Fact 2. *The composition of Krom programs is associative and distributes over union.*

Proof. Antić (2023d, Theorem 12) has shown this for positive Krom programs and it is straightforward to generalize the proof to include negative Krom programs as well. \square

We call a program P *idempotent* iff $P^2 = P$. For every interpretation I , we have

$$(2) \quad IP = I,$$

which shows that interpretations are idempotent.

2. THE INDEX AND PERIOD OF A LOGIC PROGRAM

This is the main section of the paper. Here we shall introduce — in analogy to the same concepts in semigroup theory (see, for example, Howie, 2003, p.10) — the index and period of a logic program as an algebraic measure of its cyclicity.

Let P be a logic program and consider

$$[P] := \{P, P^2, P^3, \dots\}$$

generated by P . If there are no repetitions in the list P^2, P^3, \dots , that is, if

$$P^m = P^n \quad \Rightarrow \quad m = n,$$

then $([P], \circ)$ is isomorphic to $(\mathbb{N}, +)$. In this case, we say that P has *infinite order*. Notice that this case can only occur for *infinite* programs, i.e., infinite propositional programs (which are excluded here) or first-order (or higher-order) programs with an infinite grounding (which are not considered in this paper).³

Otherwise, if there are repetitions among the powers of P , then the set

$$\{m \in \mathbb{N} \mid P^m = P^n, \text{ for some } m \neq n\}$$

³A typical example of this form would be the first-order program

$$Nat := \left\{ \begin{array}{l} nat(0) \\ nat(s(x)) \leftarrow nat(x) \end{array} \right\}$$

defining the natural numbers as we have

$$Nat \neq Nat^2 = \left\{ \begin{array}{l} nat(0) \\ nat(s(0)) \\ nat(s(s(x))) \leftarrow nat(x) \end{array} \right\} \neq \dots$$

is non-empty and thus has a least element which we will call the *index* of P denoted by $\text{index}(P)$. Then the set

$$\{r \in \mathbb{N} \mid P^{\text{index}(P)+r} = P^{\text{index}(P)}\}$$

is also non-empty and has a least element which we will call the *period* of P denoted by $\text{period}(P)$.

Intuitively, the index and period of a program contains information about the cyclicity of a program as we shall demonstrate below.

Fact 3. *Every idempotent program has period 1 and index 1. This holds in particular for every interpretation.*

Proof. The first statement is obvious and the second statement is a direct consequence of (2). \square

2.1. Elevators. In this section, we shall construct programs with index m and period 1 (cf. Proposition 4).

For every $m \geq 1$, define the *m-elevator* by the Krom program

$$E_m := \{1\} \cup \{a + 1 \leftarrow a \mid 1 \leq a < m\} = \left\{ \begin{array}{c} 1 \\ 2 \leftarrow 1 \\ \vdots \\ m \leftarrow m - 1 \end{array} \right\}.$$

For example, we have

$$E_1 = \{1\}, \quad E_2 = \left\{ \begin{array}{c} 1 \\ 2 \leftarrow 1 \end{array} \right\}, \quad E_3 = \left\{ \begin{array}{c} 1 \\ 2 \leftarrow 1 \\ 3 \leftarrow 2 \end{array} \right\}, \quad \dots$$

Notice that

$$E_m \subseteq E_n, \quad \text{for all } m \leq n.$$

For every $1 \leq k \leq m$, we have

$$E_m^k = \{1, \dots, k\} \cup \{a + 1 \leftarrow a + 1 - k \mid k \leq a \leq m - 1\}$$

which shows that for all $k, \ell \leq m - 1$,

$$E_m^k = E_m^\ell \quad \Rightarrow \quad k = \ell$$

and

$$E_m^m = [m].$$

For example, for

$$E_4 = \left\{ \begin{array}{c} 1 \\ 2 \leftarrow 1 \\ 3 \leftarrow 2 \\ 4 \leftarrow 3 \end{array} \right\}$$

we obtain the powers

$$E_4, \quad E_4^2 = \left\{ \begin{array}{c} 1 \\ 2 \\ 3 \leftarrow 1 \\ 4 \leftarrow 2 \end{array} \right\}, \quad E_4^3 = \left\{ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \leftarrow 1 \end{array} \right\}, \quad E_4^4 = \left\{ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \right\}.$$

Hence, we have shown:

Proposition 4. *The m -elevator E_m has index m and period 1.*

2.2. **Permutations.** With every permutation $\pi : [n] \rightarrow [n]$, $n \geq 1$, we associate the *permutation (Krom program)*

$$K_\pi := \{\pi(a) \leftarrow a \mid a \in [n]\}.$$

In what follows, we will not distinguish between π and K_π . We use the well-known cycle notation so that

$$\pi_{(1 \dots n)} = \left\{ \begin{array}{l} 1 \leftarrow n \\ 2 \leftarrow 1 \\ 3 \leftarrow 2 \\ \vdots \\ n \leftarrow n-1 \end{array} \right\}.$$

In what follows, we shall write π_n instead of $\pi_{(1 \dots n)}$ and we call π_n the *n -permutation (Krom program)*. The *identity n -permutation* is denoted by

$$\mathbb{1}_n = \{a \leftarrow a \mid a \in [n]\}.$$

Of course,

$$\text{index}(\mathbb{1}_n) = \text{period}(\mathbb{1}_n) = 1.$$

We always have

$$\pi_n^n = \mathbb{1}_n.$$

Notice that we have

$$E_m = (\pi_m - \{1 \leftarrow m\}) \cup \{1\}.$$

That is, E_m and π_m differ only in the rule with head atom 1.

Fact 5. *The Krom program π_n has index 1 and period n . Moreover, all powers π_n^k , $k \geq 1$, have index 1 and period n .*

Example 6. The permutation program $\pi_{(1 \ 2)}$ has index 1 and period 2 since

$$\pi_{(1 \ 2)}^2 = \mathbb{1}_{\{1,2\}} \quad \text{and} \quad \pi_{(1 \ 2)}^3 = \pi_{(1 \ 2)}.$$

2.3. $K_{m,n}$. We now wish to construct a Krom program $K_{m,n}$ with index m and period n . For this, let

$$K_{m,n} := E_m \cup \pi_{(m+1 \dots m+n)}.$$

It is important to notice that we have

$$(3) \quad E_m \cap \pi_{(m+1 \dots m+n)} = \emptyset.$$

Theorem 7. *The Krom program $K_{m,n}$ has index m and period n .*

Proof. Since $K_{m,n}$ is Krom, we have (we write π instead of $\pi_{(m+1 \dots m+n)}$)

$$K_{m,n}^2 = (E_m \cup \pi)(E_m \cup \pi) = E_m^2 \cup \pi E_m \cup E_m \pi \cup \pi^2 = E_m^2 \cup \pi^2,$$

where the last identity follows from

$$E_m \pi \stackrel{(3)}{=} \{1\} \subseteq E_m^2 \quad \text{and} \quad \pi E_m \stackrel{(3)}{=} \emptyset.$$

Notice that we have

$$(4) \quad E_m \pi^k \stackrel{(3)}{=} \{1\} \subseteq E_m^\ell \quad \text{and} \quad \pi^k E_m \stackrel{(3)}{=} \emptyset, \quad \text{for all } 1 \leq \ell \leq k.$$

Another iteration yields

$$K_{m,n}^3 = (E_m \cup \pi)(E_m^2 \cup \pi^2) = E_m^3 \cup E_m \pi^2 \cup \pi E_m^2 \cup \pi^3 \stackrel{(4)}{=} E_m^3 \cup \pi^3.$$

This leads us to the following general formula which is shown by a straightforward induction using the identities in (4):

$$K_{m,n}^k = E_m^k \cup \pi_{(m+1 \dots m+n)}^k.$$

For any $1 \leq k, \ell \leq m-1$, we have

$$E_m^k \neq E_m^\ell$$

and since

$$E_m^k \cap \pi = E_m^\ell \cap \pi = \emptyset,$$

this shows

$$K_{m,n}^k \neq K_{m,n}^\ell.$$

Moreover, we have

$$K_{m,n}^m = [m] \cup \pi^m.$$

and thus

$$K_{m,n}^{m+i} = [m] \cup \pi^{m+i}$$

which by Fact 5 implies that n is the least positive integer such that

$$K_{m,n}^{m+n} = [m] \cup \pi^{m+n} = [m] \cup \pi^m = K_{m,n}^m.$$

Hence,

$$\text{period}(K_{m,n}^m) = \text{period}(\pi^m) = \text{period}(\pi) = n.$$

In total we have thus shown

$$\text{index}(K_{m,n}) = m \quad \text{and} \quad \text{period}(K_{m,n}) = n.$$

□

3. NEGATIVE PROGRAMS

In this section, we will see that the cyclicity of negative programs is quite different from positive ones which is due to the alternating nature of negation vs. double negation.

3.1. **Negative elevators.** The *negative m-elevator* is given by

$$\text{not } E_m = \{2, \dots, m\} \cup \{a + 1 \leftarrow \text{not } a \mid 1 \leq a \leq m - 1\} = \left\{ \begin{array}{c} 2 \\ \vdots \\ m \\ 2 \leftarrow \text{not } 1 \\ \vdots \\ m \leftarrow \text{not } (m - 1) \end{array} \right\}.$$

For every $1 \leq k \leq m$, we have

$$(\text{not } E_m)^k = \begin{cases} \{2, \dots, m\} \cup \{a + 1 \leftarrow a - k + 1 \mid k \leq a \leq m - 1\} & \text{if } k \text{ is even,} \\ \{2, \dots, m\} \cup \{a + 1 \leftarrow \text{not } (a - k + 1) \mid k \leq a \leq m - 1\} & \text{if } k \text{ is odd.} \end{cases}$$

For example, we have

$$(\text{not } E_m)^2 = \left\{ \begin{array}{c} 2 \\ \vdots \\ m \\ 3 \leftarrow 1 \\ 4 \leftarrow 2 \\ \vdots \\ m \leftarrow m - 2 \end{array} \right\}, \quad (\text{not } E_m)^3 = \left\{ \begin{array}{c} 2 \\ \vdots \\ m \\ 4 \leftarrow \text{not } 1 \\ \vdots \\ m \leftarrow \text{not } (m - 3) \end{array} \right\}, \quad (\text{not } E_m)^4 = \left\{ \begin{array}{c} 2 \\ \vdots \\ m \\ 5 \leftarrow 1 \\ \vdots \\ m \leftarrow m - 4 \end{array} \right\}.$$

In particular, we have

$$(\text{not } E_m)^m = [m] = E_m^m,$$

regardless of whether m is even or odd.

We have thus shown:

Proposition 8. *The negative m-elevator not E_m has index m and period 1, that is,*

$$\text{index}(\text{not } E_m) = \text{index}(E_m) = m \quad \text{and} \quad \text{period}(\text{not } E_m) = \text{period}(E_m) = 1.$$

3.2. **Negative permutations.** We now turn our attention to negative programs and we shall see that such programs have a different kind of cyclicity.

The *negative n-permutation (Krom program)* is given by

$$\text{not } \pi_n = \left\{ \begin{array}{c} 2 \leftarrow \text{not } 1 \\ \vdots \\ n \leftarrow \text{not } (n - 1) \\ 1 \leftarrow \text{not } n \end{array} \right\}.$$

We have

$$(5) \quad (\text{not } \pi_n)^k = \begin{cases} \pi_n^k & \text{if } k \text{ is even,} \\ \text{not } \pi_n^k & \text{if } k \text{ is odd.} \end{cases}$$

In particular, we have

$$(6) \quad (\text{not } \pi_n)^n = \begin{cases} \mathbb{1}_n & \text{if } n \text{ is even,} \\ \text{not } \mathbb{1}_n & \text{if } n \text{ is odd.} \end{cases}$$

We thus have

$$n \text{ is even} \Rightarrow \text{period}(\text{not } \pi_n) = n.$$

What if n is odd? We claim:

$$n \text{ is odd} \Rightarrow \text{period}(\text{not } \pi_n) = 2n.$$

We have

$$(\text{not } \mathbb{1}_n)^2 = \mathbb{1}_n^2 = \mathbb{1}_n,$$

which by (6) implies

$$(\text{not } \pi_n)^{2n} = \mathbb{1}_n$$

and thus

$$(\text{not } \pi_n)^{1+2n} = \text{not } \pi_n.$$

It remains to show

$$(7) \quad \text{not } \pi_n \neq (\text{not } \pi_n)^2 \neq \dots \neq (\text{not } \pi_n)^{2n}.$$

Since n is odd by assumption, by (5) this sequence of powers equals

$$(8) \quad \text{not } \pi_n, \pi_n^2, \dots, \text{not } \pi_n^{n-2}, \pi_n^{n-1}, \text{not } \pi_n^n, \pi_n^{n+1}, \text{not } \pi_n^{n+2}, \dots, \text{not } \pi_n^{2n-1}, \pi_n^{2n}.$$

Notice that since n is odd, we have, for all $1 \leq k \leq n$,

$$(\text{not } \pi_n)^{n+k} = \begin{cases} \pi_n^k & \text{if } k \text{ is odd,} \\ \text{not } \pi_n^k & \text{if } k \text{ is even.} \end{cases}$$

In particular, we have

$$\text{not } \pi_n^{2n} = \pi_n^n.$$

Hence, the sequence in (8) can be written as

$$\text{not } \pi_n, \pi_n^2, \dots, \text{not } \pi_n^{n-2}, \pi_n^{n-1}, \text{not } \pi_n^n, \pi_n, \text{not } \pi_n^2, \dots, \text{not } \pi_n^{n-1}, \pi_n^n.$$

This immediately shows (7).

For example, for $n = 3$, the $2n = 6$ distinct powers of $\text{not } \pi_3$ are given by

$$\begin{aligned} (\text{not } \pi_3)^1 &= \begin{pmatrix} 2 \leftarrow \text{not } 1 \\ 3 \leftarrow \text{not } 2 \\ 1 \leftarrow \text{not } 3 \end{pmatrix}, & (\text{not } \pi_3)^2 &= \pi_3^2 = \begin{pmatrix} 2 \leftarrow 3 \\ 3 \leftarrow 1 \\ 1 \leftarrow 2 \end{pmatrix}, & (\text{not } \pi_3)^3 &= \text{not } \pi_3^3 = \begin{pmatrix} 2 \leftarrow \text{not } 2 \\ 3 \leftarrow \text{not } 3 \\ 1 \leftarrow \text{not } 1 \end{pmatrix} \\ (\text{not } \pi_3)^4 &= \pi_3^4 = \begin{pmatrix} 2 \leftarrow 1 \\ 3 \leftarrow 2 \\ 1 \leftarrow 3 \end{pmatrix}, & (\text{not } \pi_3)^5 &= \text{not } \pi_3^5 = \begin{pmatrix} 2 \leftarrow \text{not } 3 \\ 3 \leftarrow \text{not } 1 \\ 1 \leftarrow \text{not } 2 \end{pmatrix}, & (\text{not } \pi_3)^6 &= \pi_3^6 = \begin{pmatrix} 2 \leftarrow 2 \\ 3 \leftarrow 3 \\ 1 \leftarrow 1 \end{pmatrix}. \end{aligned}$$

In total, we have thus shown:

Proposition 9. *The negative n -permutation $\text{not } \pi_n$ has index 1 and period*

$$\text{period}(\text{not } \pi_n) = \begin{cases} n & \text{if } n \text{ is even,} \\ 2n & \text{if } n \text{ is odd.} \end{cases}$$

That is,

$$\text{period}(\text{not } \pi_n) = \begin{cases} \text{period}(\pi_n) & \text{if } n \text{ is even,} \\ 2 \times \text{period}(\pi_n) & \text{if } n \text{ is odd.} \end{cases}$$

Remark 10. Notice that the period of $\text{not } \pi_n$ is always even!

This means that the cyclicity of $\text{not } \pi_n$ is identical to that of π_n iff n is even, and it differs by a factor of 2 iff n is odd; this shows that positive and negative programs are different when it comes to cycles which coincides with our intuition.

4. PROBLEMS

This section lists problems which remained unsolved in this paper and appear to be interesting lines of future research.

Problem 1. Study the relationship between the index and period of a program and its decompositions into factors (cf. Antić, 2023e, 2023b).

Problem 2. For simplicity, in this paper we have dealt only with Krom programs of a very simplistic form. For the future, it is therefore mandatory to study the index and period of non-Krom programs which is more complicated already in the Horn fragment since the sequential composition of propositional Horn programs is not associative and does not distribute over union (cf. Antić, 2023d, Example 8). For non-Krom answer set programs the situation is even more complex since the sequential composition of answer set programs is quite involved (cf. Antić, 2023c).

Problem 3. In this paper, we have restricted ourselves to propositional programs. The main line of future research is to study the index and period of a program for first-order and higher-order logic programs (cf. Miller & Nadathur, 2012; Chen, Kifer, & Warren, 1993).

5. CONCLUSION

The paper introduced the index and period of a logic program as an algebraic measure of the cyclicity of a program. We showed that there are propositional Krom programs, composed of elevators and permutations, of arbitrary index and period.

CONFLICT OF INTEREST

The authors declare that they have no conflict of interest.

DATA AVAILABILITY STATEMENT

The manuscript has no data associated.

REFERENCES

- Antić, C. (2023a). Logic program proportions. *Annals of Mathematics and Artificial Intelligence*, accepted. <https://arxiv.org/pdf/1809.09938.pdf>.
- Antić, C. (2023b). On syntactically similar logic programs and sequential decompositions. submitted to *Information and Computation*, <https://arxiv.org/pdf/2109.05300.pdf>.
- Antić, C. (2023c). Sequential composition of answer set programs. submitted to *Information and Computation*, <https://arxiv.org/pdf/2104.12156v2.pdf>.
- Antić, C. (2023d). Sequential composition of propositional logic programs. submitted to *Annals of Mathematics and Artificial Intelligence*, <https://arxiv.org/pdf/2009.05774v4.pdf>.
- Antić, C. (2023e). Sequential decomposition of propositional logic programs. <https://arxiv.org/pdf/2304.13522.pdf>.
- Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, Cambridge.

- Brewka, G., Eiter, T., & Truszczynski, M. (2011). Answer set programming at a glance. *Communications of the ACM*, 54(12), 92–103.
- Brogi, A., Mancarella, P., Pedreschi, D., & Turini, F. (1999). Modular logic programming. *ACM Transactions on Programming Languages and Systems*, 16(4), 1361–1398.
- Bugliesi, M., Lamma, E., & Mello, P. (1994). Modularity in logic programming. *The Journal of Logic Programming*, 19-20(1), 443–502.
- Chen, W., Kifer, M., & Warren, D. S. (1993). HiLog: A foundation for higher-order logic programming. *The Journal of Logic Programming*, 15(3), 187–230.
- Clark, K. L. (1978). Negation as failure. In Gallaire, H., & Minker, J. (Eds.), *Logic and Data Bases*, pp. 293–322. Plenum Press, New York.
- Eiter, T., Ianni, G., & Krennwallner, T. (2009). Answer set programming: a primer. In *Reasoning Web. Semantic Technologies for Information Systems*, volume 5689 of *Lecture Notes in Computer Science*, pp. 40–110. Springer, Heidelberg.
- Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4), 365–385.
- Howie, J. M. (2003). *Fundamentals of Semigroup Theory*. London Mathematical Society Monographs New Series. Oxford University Press, Oxford.
- Krom, M. R. (1967). The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2), 15–20.
- Lifschitz, V. (2019). *Answer Set Programming*. Springer Nature Switzerland AG, Cham, Switzerland.
- Miller, D., & Nadathur, G. (2012). *Programming with Higher-Order Logic*. Cambridge University Press.
- O’Keefe, R. A. (1985). Towards an algebra for constructing logic programs. In *SLP 1985*, pp. 152–160.