# A Second-Order Formulation of Non-Termination

Fred Mesnard, Étienne Payet

*Université de La Réunion, EA2525-LIM, Saint-Denis de La Réunion, F-97490, France*

**Abstract**

We consider the termination/non-termination property of a class of loops. Such loops are commonly used abstractions of real program pieces. Second-order logic is a convenient language to express non-termination. Of course, such property is generally undecidable. However, by restricting the language to known decidable cases, we exhibit new classes of loops, the non-termination of which is decidable. We present a bunch of examples.

*Keywords:* Termination, non-termination, monadic second-order logic.

## 1. Introduction

In this paper, we recall that second-order logic is a convenient language to express non-termination of while loops, modeled as rules. Such rules are commonly used abstractions of real program pieces, see, e.g., [13] for the Java programming language. Our main contribution is the definition of two new classes of rules, the termination of which is decidable, by restricting the language to known decidable cases, namely S1S and S2S. We also show and illustrate how decision procedures for their weak versions WS1S and WS2S can help proving termination/non-termination.

We organize the paper as follows. Section 2 presents the main concepts we need while Section 3 gives the theoretical results of the paper. Section 4 illustrates the results by means of examples and in Section 5 we discuss related works. Finally, Section 6 concludes the paper.

## 2. Preliminaries

We give a quick description of S1S and S2S, see [14] for a more detailed presentation. S1S is the monadic Second-order theory of 1 Successor. Interpretations correspond to finite or infinite words over a given finite alphabet $\Sigma$. Terms are constructed from the constant 0 and first-order variables $x$, $y$, ... by

*Email addresses:* frederic.mesnard@univ-reunion.fr (Fred Mesnard), etienne.payet@univ-reunion.fr (Étienne Payet)

application of the successor function $+1$, which is left-associative. We abbreviate $n$ successive applications of $+1$ starting from 0 (i.e., $0+1+1+\cdots+1$) to $n$. Atomic formulæ are constructed from terms, second-order variables $X$, $Y$, ... and predicates of the form $P_a$ where $a \in \Sigma$. They have the form $t = t'$, $t < t'$, $t \in X$, $P_a(t)$ where $t$ and $t'$ are terms. Formulæ are constructed from atomic formulæ, the usual boolean connectives ($\vee$, $\wedge$, ...) and quantification ($\forall$ and $\exists$) over first and second-order variables. First-order variables are interpreted as elements of $\mathbb{N}$ representing positions in words and second-order variables as subsets of $\mathbb{N}$. Constant 0 is interpreted as the first position in a word and function $+1$ as the next position. The formula $P_a(t)$ is true in a word $w$ if at position $t$ of $w$ there is character $a$. WS1S (Weak S1S) is a restriction of S1S where second-order variables are interpreted as *finite* sets only.

S2S is the monadic Second-order theory of 2 Successors. Interpretations correspond to finite or infinite labelled binary trees over a given finite alphabet $\Sigma$. Terms and formulæ are constructed as in S1S except that constant 0 is replaced with $\varepsilon$ and the successor function $+1$ is replaced with functions .0 and .1, which are left-associative. We abbreviate successive applications of these functions, for instance $x.0110$ stands for $x.0.1.1.0$, which corresponds to $(((x.0).1).1).0$, and 0110 stands for $\varepsilon.0.1.1.0$. First-order variables are interpreted as elements of $\{0,1\}^*$ representing positions in binary trees and second-order variables as subsets of $\{0,1\}^*$. Constant $\varepsilon$ is interpreted as the root position of a binary tree, .0 as the left successor, .1 as the right successor and $<$ as the proper-prefix relation (for instance $01 < 0110$ but $00 \not< 0110$). WS2S (Weak S2S) is a restriction of S2S where second-order variables are interpreted as *finite* sets only.

A *rule* has the form $r : \tilde{x} \to \psi(\tilde{x}, \tilde{y}), \tilde{y}$ where $r$ is the identifier of the rule, $\psi$ is a binary relation and $\tilde{x}$ and $\tilde{y}$ are tuples of distinct first-order variables ranging over a given domain. If $r$ is a monadic rule of the form $x \to \psi(x, y), y$ and $\psi(x, y)$ is a monadic second-order formula of S1S (S2S) with $x$ and $y$ as free variables, we call $r$ a *monadic* S1S (respectively, S2S) rule. Some examples can be found in Section 4. We define an operational semantics as follows. Starting from a concrete tuple $\tilde{x_0}$ of elements of the domain, we first check whether there exists a concrete tuple $\tilde{x_1}$ such that $\psi(\tilde{x_0}, \tilde{x_1})$. If no such tuple exists, the computation stops. Otherwise, we choose any such tuple $\tilde{x_1}$ and reiterate. The rule $r$ *loops* if we can find a concrete tuple $\tilde{x_0}$ starting an infinite computation. If no such tuple exists, $r$ *terminates*.

## 3. A Second-Order Formulation of Non-Termination

We consider the following second-order formulation of non-termination. Let $r : \tilde{x} \to \psi(\tilde{x}, \tilde{y}), \tilde{y}$ be a rule.

**Definition 1 (recurrence set [8]).** *We let $\phi_r$ denote the second-order formula*

$$\exists X \begin{cases} \exists \tilde{x}\ \tilde{x} \in X \wedge & (1) \\ \forall \tilde{x} \exists \tilde{y}\ (\tilde{x} \in X \Rightarrow [\psi(\tilde{x}, \tilde{y}) \wedge \tilde{y} \in X]) & (2) \end{cases}$$

*A* recurrence set *for $r$ is a set $X$ satisfying $\phi_r$.*

Condition (1) of Definition 1 simply states that the recurrence set $X$ is not empty. Condition (2) ensures that for any element $\tilde{x}$ of $X$, there is an element $\tilde{y}$ of $X$ which satisfies the formula $\psi(\tilde{x}, \tilde{y})$ defining the rule $r$. The existence of a recurrence set is equivalent to non-termination.

**Theorem 2 ([8]).** *$\phi_r$ is true if and only if $r$ loops.*

*Proof.* We prove both implications.
($\Rightarrow$). As $\phi_r$ is true, we can start by selecting any arbitrary $\tilde{x_0} \in X$. We know that there exists $\tilde{y_0} \in X$ s.t. $\psi(\tilde{x_0}, \tilde{y_0})$. By iterating this process, we construct an infinite computation. Hence $r$ loops.
($\Leftarrow$). As there exists $\tilde{x_0}$ such that $r$ loops, let us consider an infinite computation starting at $\tilde{x_0}$: $\tilde{x_0}, \tilde{x_1}, \ldots, \tilde{x_n}, \ldots$ Let $X = \{\tilde{x_i} | i \geq 0\}$. $X$ is a non-empty set verifying $\forall \tilde{x} \exists \tilde{y}\ (\tilde{x} \in X \Rightarrow [\psi(\tilde{x}, \tilde{y}) \wedge \tilde{y} \in X])$. Hence $\phi_r$ holds. □

The second-order formula $\phi_r$ is a necessary and sufficient condition for non-termination of at least one of the computations $r$ can generate. Symmetrically, $\neg\phi_r$ is true if and only if for every value $\tilde{x_0}$, any computation starting at $\tilde{x_0}$ halts. As such a problem is in general undecidable (see, e.g., [3]), it follows that $\phi_r$ is not computable. However, when the second-order logic is restricted to decidable cases, we obtain classes of rules for which the termination/non-termination property is decidable.

**Theorem 3.** *Termination of a* monadic *S1S or S2S rule is decidable.*

*Proof.* The monadic second-order logics S1S and S2S are decidable [5, 12] and so is $\phi_r$ for a monadic S1S or S2S rule $r$. If $\phi_r$ is true then $r$ loops else $r$ terminates. □

Weak versions of these logics, where second-order variables range over *finite* sets, are also decidable and decision procedures have been implemented (see, e.g., MONA [9]). Let $r$ be a monadic S1S or S2S rule.

**Corollary 1.** *Decision procedures for WS1S and WS2S provide computable sufficient conditions for proving non-termination of $r$ in the corresponding structure.*

*Proof.* If such a decision procedure states that $\phi_r$ is true, then we know that there exists a non-empty finite set $X$ such that $\phi_r$ holds. Hence $r$ loops. □

Note that if the decision procedure states that $\phi_r$ is false, then there is no finite set $X$ satisfying $\phi_r$ but an infinite set $X$ satisfying $\phi_r$ may exist. Hence we cannot conclude, except in the following case.

**Corollary 2.** *When we know that the set of points which can start a computation from $r$ is finite, decision procedures for WS1S and WS2S also decide termination of $r$ in the corresponding structure.*

*Proof.* If a decision procedure states that $\phi_r$ is true, then by Corollary 1 $r$ loops. Else it states that $\phi_r$ is false. So there does not exist a finite set $X$ satisfying $\phi_r$. As $X$ cannot be infinite by hypothesis, it means that there does not exist a set $X$ such that $\phi_r$ holds. Hence $r$ terminates. □

Note that the condition of Corollary 2 can be decided in WS1S as it can be stated as $\exists m\ \forall x\ (x > m \Rightarrow \neg\ \exists y\ \psi(x, y))$. However Example 6 shows that it does not decide termination.

## 4. Examples

**Example 4 (S1S).** Consider $r:\ x \to \psi(x, y), y$ where

$$\psi(x, y) = (3 < x \wedge x < 10 \wedge y < x) \vee (x < 3 \wedge y = x + 1)$$

The set of points which can start a computation from $r$ is finite: $\{x \in \mathbb{N} | x \neq 3 \wedge x < 10\}$. MONA tells us that $\phi_r$ is false. By Corollary 2, $r$ terminates. □

**Example 5 (S1S).** Consider $r:\ x \to \psi(x, y), y$ where

$$\psi(x, y) = (3 < x \wedge y < x) \vee (x < 4 \wedge y = x + 1)$$

MONA reports that $\phi_r$ is true, with a computed satisfying $X = \{3, 4\}$. Indeed for any $x \in X = \{3, 4\}$, there is a $y$ in $X$ such that $\psi(x, y)$ holds: if $x = 3$, take $y = 4$ and if $x = 4$, $y = 3$. Note that the set $X$ is not unique, as $\phi_r$ is also true for, e.g., $X = \{2, 3, 4, 2014\}$. By Corollary 1, $r$ loops. □

**Example 6 (S1S).** Consider $r:\ x \to \psi(x, y), y$ where

$$\psi(x, y) = (x < y)$$

Although MONA tells us that there is no finite $X$ satisfying $\phi_r$, as the set of points which can start a computation is infinite, we cannot apply Corollary 2. Indeed, taking $X = \mathbb{N}$ shows that $\phi_r$ is true. Hence by Theorem 2, $r$ loops. Note that any decision procedure for S1S will prove that $\phi_r$ is true. □

**Example 7 (S1S).** Consider $r:\ x \to \psi(x, y), y$ where

$$\psi(x, y) = (y < x)$$

MONA reports that there is no finite $X$ satisfying $\phi_r$. As the set of points which can start a computation is infinite, we cannot apply Corollary 2. Assume that $\phi_r$ is true. So there is a non-empty $X \subseteq \mathbb{N}$ satisfying $\phi_r$. Let $e$ be its least element. Condition (2) of Definition 1 states that there exists $d$ in $X$ such that $d < e$, which contradicts that $e$ is the least element of $X$. Hence $\phi_r$ is false, as should be shown by any decision procedure for S1S. By Theorem 2, $r$ terminates. □

**Example 8 (S1S).** Consider $r: \ x \to \psi(x,y), y$ where

$$\psi(x,y) = (\forall X \ (x \in X \wedge \psi'(X)) \Rightarrow y \in X)$$

with

$$\psi'(X) = (\forall z \ z \in X \Rightarrow z+1 \in X)$$

We have $\psi'(X)$ is true if and only if $X$ is closed by application of the successor function $+1$. So, $\psi(x,y)$ is true if and only if $x \leq y$. MONA reports that $\phi_r$ is true, with a computed satisfying $X = \{0\}$. By Corollary 1, $r$ loops. □

**Example 9 (S2S).** Consider $r: \ x \to \psi(x,y), y$ where

$$\psi(x,y) = (y = x.1 \vee x = y.1)$$

The set $X = \{\varepsilon, 1\}$ is not empty and for any $x$ in $X$ there is a $y$ in $X$ such that $\psi(x,y)$ holds. So $\phi_r$ is true (also shown by MONA). By Corollary 1, $r$ loops. □

**Example 10 (S2S).** Consider $r: \ x \to \psi(x,y), y$ where

$$\begin{aligned}\psi(x,y) = \\ &(\exists z \ x < 0^4 \wedge x = z.0 \wedge y = z.1) \vee \\ &(\exists z \ z.01 \leq x \wedge y = z.11) \vee \\ &(x = 1^2 \wedge y = 0^3)\end{aligned}$$

The set $X = \{1^2, 0^3, 0^2 1, 01^2\}$ is not empty and for any $x$ in $X$ there is a $y$ in $X$ such that $\psi(x,y)$ holds. Hence $\phi_r$ is true (also shown by MONA), so $r$ loops.

**Example 11 (S2S).** Consider $r: \ x \to \psi(x,y), y$ where

$$\begin{aligned}\psi(x,y) = \\ &(\exists z \ x = z.0 \wedge y = z.1) \vee \\ &(\exists z \ x = z.1 \wedge y = z.10)\end{aligned}$$

The infinite set $X = \{0, 1, 10, 11, 110, 111, \ldots\} = 1^*(0+1)$ is not empty and for any $x$ in $X$ there is a $y$ in $X$ such that $\psi(x,y)$ holds. Hence $\phi_r$ is true, as should be shown by any decision procedure for S2S. So $r$ loops.

**Example 12 (S2S).** Consider $r: \ x \to \psi(x,y), y$ where

$$\psi(x,y) = (\forall X \ (x \in X \wedge \psi'(X)) \Rightarrow y \in X)$$

with

$$\psi'(X) = (\forall z \ z \in X \Rightarrow (z.0 \in X \wedge z.1 \in X))$$

We have $\psi'(X)$ is true if and only if $X$ is closed by application of the successor functions .0 and .1. So, $\psi(x,y)$ is true if and only if $x \leq y$. MONA reports that $\phi_r$ is true, with a computed satisfying $X = \{\varepsilon\}$. By Corollary 1, $r$ loops. □

## 5. Related Works

Recurrence sets were first introduced in [8] where $\psi$ denotes any binary relation. Two symbolic analyses are presented in this paper for constructing such sets: a bitwise analysis, which assumes that the state space is finite and encoded using Boolean variables, and a linear arithmetic analysis, which assumes that the program transitions can be represented as rational linear constraints. In contrast to our work, no second-order formulation is considered in this paper.

Let us now focus on termination-decidable classes of rules. In [7], the authors present a decision procedure for an arbitrary rule $\tilde{x} \to \psi(\tilde{x}, \tilde{y}), \tilde{y}$ where $\psi(\tilde{x}, \tilde{y})$ is a conjunction of equality constraints over rational trees. In [11, 6, 1], one finds variations of a decision procedure for finite sets of rules $\tilde{x} \to \psi(\tilde{x}, \tilde{y}), \tilde{y}$ where $\psi(\tilde{x}, \tilde{y})$ is a conjunction of constraints $x > y$ or $x \geq y$ over a well-founded domain (such as the natural numbers) or the integers. Generalizing [15], termination of an arbitrary deterministic linear loop is shown decidable in [4] over the integers, the rationals, and the reals. To the best of our knowledge, termination of a non-deterministic linear loop remains an open problem. By restricting the transition relation $\psi$ to an integer octagon or an integer linear affine relation with the finite monoid property, [2] shows that the non-termination precondition is decidable. The application of their techniques to integer data programs with control structure consisting in more than one loop is presented in [10].

In comparison with other analyses, a drawback of our result comes from the restriction to monadic rules, which implies that our approach can only consider abstractions of program pieces which focus on the evolution of only one variable. On the other hand, as illustrated in the examples of Section 4, the specification of the transition relation may benefit from the expressiveness of S1S or S2S.

## 6. Conclusion

We have seen that second-order logic is a convenient language to express non-termination as a necessary and sufficient condition. Such a condition is in general undecidable. By restricting the language to the decidable cases S1S and S2S, we have defined two new classes of rules, the termination of which is decidable. Finally, we have shown that the weak versions of these logics provide sufficient conditions for termination and non-termination of such rules.

## Acknowledgements

We thank the anonymous reviewers for their useful remarks.

[1] A. M. Ben-Amram. Monotonicity constraints for termination in the integer domain. *Logical Methods in Computer Science*, 7(3), 2011.

[2] M. Bozga, R. Iosif, and F. Konečný. Deciding conditional termination. *Logical Methods in Computer Science*, 10(3), 2014.

[3] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination of polynomial programs. In R. Cousot, editor, *Proc. of the 6th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, volume 3385 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2005.

[4] M. Braverman. Termination of integer linear programs. In T. Ball and R. B. Jones, editors, *Proc. of the 18th International Conference on Computer Aided Verification (CAV'06)*, volume 4144 of *Lecture Notes in Computer Science*, pages 372–385. Springer, 2006.

[5] J. R. Büchi. On a decision method in restricted second-order arithmetic. In E. Nagel, P. Suppes, and A. Tarski, editors, *Proc. of the 1960 International Congress on Logic, Methodology and Philosophy of Science (LMPS'60)*, pages 1–11. Stanford University Press, June 1962.

[6] M. Codish, V. Lagoon, and P. Stuckey. Testing for termination with monotonicity constraints. In M. Gabbrielli and G. Gupta, editors, *Proc. of the 21st International Conference on Logic Programming (ICLP'05)*, volume 3668 of *Lecture Notes in Computer Science*, pages 326–340. Springer, 2005.

[7] D. De Schreye, M. Bruynooghe, and K. Verschaetse. On the existence of nonterminating queries for a restricted class of Prolog-clauses. *Artificial Intelligence*, 41:237–248, 1989.

[8] A. Gupta, Thomas A. Henzinger, R. Majumdar, A. Rybalchenko, and R.-G. Xu. Proving non-termination. In G. C. Necula and P. Wadler, editors, *Proc. of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, pages 147–158. ACM Press, 2008.

[9] N. Klarlund and A. Møller. *MONA Version 1.4 User Manual*. BRICS, Department of Computer Science, Aarhus University, January 2001. Notes Series NS-01-1. Available from `http://www.brics.dk/mona/`. Revision of BRICS NS-98-3.

[10] F. Konečný. *Relational Verification of Programs with Integer Data*. PhD thesis, Brno University of Technology and Université de Grenoble, 2012.

[11] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. In C. Hankin and D. Schmidt, editors, *Proc. of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*, pages 81–92. ACM Press, 2001.

[12] M. O. Rabin. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969.

[13] F. Spoto, F. Mesnard, and É. Payet. A termination analyzer for Java bytecode based on path-length. *ACM Transactions on Programming Languages and Systems*, 32(3), 2010.

[14] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 133–191. Elsevier and MIT Press, 1990.

[15] A. Tiwari. Termination of linear programs. In R. Alur and D. Peled, editors, *Proc. of the 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *Lecture Notes in Computer Science*, pages 70–82. Springer, 2004.