Information Processing Letters 115 (2015) 882–885

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

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

ARTICLE INFO

Article history:
Received 27 June 2014
Received in revised form 23 February 2015
Accepted 24 May 2015
Available online 28 May 2015
Communicated by J.L. Fiadeiro

Keywords:
Program correctness
Termination
Non-termination
Monadic second-order logic

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.

1. Introduction

In this paper, we recall that second-order logic is a convenient language to express non-termination of while loops, modelled 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 organise 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 Σ. Terms are constructed from the constant 0 and first-order variables x, y, … by 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

[*] Corresponding author.
E-mail addresses: frederic.mesnard@univ-reunion.fr (F. Mesnard), etienne.payet@univ-reunion.fr (É. Payet).

http://dx.doi.org/10.1016/j.ipl.2015.05.012
0020-0190/© 2015 Elsevier B.V. All rights reserved.

binary trees over a given finite alphabet Σ. Terms and formulæ are constructed as in S1S except that constant 0 is replaced with ε 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 ε 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, ψ 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).* (See [8].) We let ϕ_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 ϕ_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. *(See [8].) ϕ_r is true if and only if r loops.*

Proof. We prove both implications.
($\Rightarrow$). As ϕ_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 $\tilde{x_0}$: $\tilde{x_0}, \tilde{x_1}, \dots, \tilde{x_n}, \dots$. 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 ϕ_r holds. □

The second-order formula ϕ_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 ϕ_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 ϕ_r for a monadic S1S or S2S rule r. If ϕ_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 ϕ_r is true, then we know that there exists a non-empty finite set X such that ϕ_r holds. Hence r loops. □

Note that if the decision procedure states that ϕ_r is false, then there is no finite set X satisfying ϕ_r but an infinite set X satisfying ϕ_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 ϕ_r is true, then by Corollary 1 r loops. Else it states that ϕ_r is false. So there does not exist a finite set X satisfying ϕ_r. As X cannot be infinite by hypothesis, it means that there does not exist a set X such that ϕ_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 ϕ_r is false. By Corollary 2, r terminates. □