Electronic Notes in Theoretical Computer Science 204 (2008) 21–34 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Decidability of Innermost Termination and Context-Sensitive Termination for

Semi-Constructor Term Rewriting Systems

Keita Uchiyama [1](#_bookmark0) Masahiko Sakai [2](#_bookmark0) Toshiki Sakabe [3](#_bookmark0)

*Graduate School of Information Science Nagoya University*

*Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan*

**Abstract**

Yi and Sakai [[13](#_bookmark50)] showed that the termination problem is a decidable property for the class of semi- constructor term rewriting systems, which is a superclass of the class of right-ground term rewriting systems. Decidability was shown by the fact that every non-terminating TRS in the class has a loop. In this paper we modify the proof of [[13](#_bookmark50)] to show that both innermost termination and *μ*-termination are decidable properties for the class of semi-constructor TRSs.

*Keywords:* Context-Sensitive Termination, Dependency Pair, Innermost Termination

# Introduction

Termination is one of the central properties of term rewriting systems (TRSs for short), where we say a TRS terminates if it does not admit any infinite reduction sequence. Since termination is undecidable in general, several decidable classes have been studied [[6](#_bookmark43),[8](#_bookmark45),[9](#_bookmark46),[12](#_bookmark49),[13](#_bookmark50)]. The class of semi-constructor TRSs is one of them [[13](#_bookmark50)], where a TRS is in this class if for every right-hand side of rules all its subterms having a defined symbol at root position are ground.

Innermost reduction, the strategy which rewrites innermost redexes, is used for call-by-value computation. Context-sensitive reduction is a strategy in which rewritable positions are indicated by specifying arguments of function symbols. Some non-terminating TRSs are terminating by context-sensitive reduction without loss of computational ability. The termination property with respect to innermost

1 Email: [uchiyama@sakabe.i.is.nagoya-u.ac.jp](mailto:uchiyama@sakabe.i.is.nagoya-u.ac.jp)

2 Email: [sakai@is.nagoya-u.ac.jp](mailto:sakai@is.nagoya-u.ac.jp)

3 Email: [sakabe@is.nagoya-u.ac.jp](mailto:sakabe@is.nagoya-u.ac.jp)

1571-0661 © 2008 Elsevier B.V. Open access under [CC BY-NC-ND license](http://creativecommons.org/licenses/by-nc-nd/3.0/).

doi:10.1016/j.entcs.2008.03.051

(resp. context-sensitive) reduction is called innermost (resp. context-sensitive) ter- mination. Since innermost termination and context-sensitive termination are also undecidable in general, methods for proving these terminations have been studied [[2](#_bookmark39),[4](#_bookmark41)].

In this paper, we prove that innermost termination and context-sensitive termi- nation for semi-constructor TRSs are decidable properties. We show that context- sensitive termination for *μ*-semi-constructor TRSs having no infinite variable depen- dency chain is a decidable property. We also extend the classes by using dependency graphs.

# Preliminaries

We assume the reader is familiar with the standard definitions of term rewriting systems [[5](#_bookmark42)], dependency pairs [[4](#_bookmark41)], and context-sensitive rewriting [[2](#_bookmark39)]. Here we just review the main notations used in this paper.

A *signature F* is a set of function symbols, where every *f ∈F* is associated with a non-negative integer by an arity function: arity: *F →* N. The set of all *terms* built from a signature *F* and a countably infinite set *V* of *variables* such that *F ∩ V* = *∅*, is represented by *T* (*F, V*). The set of *ground terms* is *T* (*F, ∅*). The set of variables

occurring in a term *t* is denoted by Var(*t*).

The set of all *positions* in a term *t* is denoted by *P*os(*t*) and *ε* represents the root position. *P*os(*t*) is: *P*os(*t*) = *{ε}* if *t ∈ V*, and *P*os(*t*) = *{ε}∪ {iu |* 1 *≤ i ≤ n, u ∈ P*os(*ti*)*}* if *t* = *f* (*t*1*,..., tn*). Let *C* be a *context* with a hole . We write *C*[*t*]*p* for the term obtained from *C* by replacing at position *p* with a term *t*. We sometimes write *C*[*t*] for *C*[*t*]*p* by omitting the position *p*. We say *t* is a subterm of *s* if *s* = *C*[*t*] for some context *C*. We denote the *subterm relation* by *Œ*, that is, *t Œ s* if *t* is a subterm of *s*, and *t s* if *t Œ s* and *t /*= *s*. The *root symbol* of a term *t* is denoted by root(*t*).

A *substitution θ* is a mapping from *V* to *T* (*F, V*) such that the set Dom(*θ*) =

*{x ∈ V | θ*(*x*) */*= *x}* is finite. We usually identify a substitution *θ* with the set

*{x '→ θ*(*x*) *| x ∈* Dom(*θ*)*}* of variable bindings. In the following, we write *tθ* instead of *θ*(*t*).

A *rewrite rule l → r* is a directed equation which satisfies *l /∈ V* and Var(*r*) *⊆*

Var(*l*). A *term rewriting system* TRS is a finite set of rewrite rules. A *redex* is a term *lθ* for a rule *l → r* and a substitution *θ*. A term containing no redex is called a *normal form*. A substitution *θ* is *normal* if *xθ* is in normal forms for every *x*. The

*reduction relation −→*

*R*

*⊆ T* (*F, V*) *×T* (*F, V*) associated with a TRS *R* is defined

as follows: *s −→ t* if there exist a rewrite rule *l → r ∈ R*, a substitution *θ*, and a

*R*

context *C*[ ]*p* such that *s* = *C*[*lθ*]*p* and *t* = *C*[*rθ*]*p*, we say that *s* is reduced to *t* by

contracting redex *lθ*. We sometimes write

*−→p*

*R*

for *−→*

*R*

by displaying the position *p*.

A redex is *innermost* if all its proper subterms are in normal forms. If *s* is

reduced to *t* by contracting an innermost redex, then *s →R t* is said to be an

*innermost reduction* denoted by *s −−−→ t*.

*in,R*

**Proposition 2.1** *Fora TRS R, if there is a reduction s −−−→ t, then C*[*s*] *−−−→ C*[*t*]

*for any context C.*

*in,R*

*in,R*

A mapping *μ* : *F → P*(N) is a *replacement map* (or *F*-map) if *μ*(*f* ) *⊆*

*{*1*,...,* arity(*f* )*}*. The set of *μ-replacing positions P*os*μ*(*t*) of a term *t* is: *P*os*μ*(*t*) =

*{ε}*, if *t ∈V* and *P*os*μ*(*t*) = *{ε}∪ {iu | i ∈ μ*(*f* )*,u ∈ P*os*μ*(*ti*)*}*, if *t* = *f* (*t*1*,..., tn*). A context *C*[ ]*p* is *μ-replacing* denoted by *Cμ*[ ]*p* if *p ∈ P*os*μ*(*C*). The set of all *μ-replacing variables* of *t* is Var*μ*(*t*) = *{x ∈* Var(*t*) *| ∃C, Cμ*[*x*]*p* = *t}*. The *μ- replacing subterm relation Œμ* is given by *s Œμ t* if there is *p ∈ P*os*μ*(*t*) such that

*t* = *C*[*s*]*p*. A *context-sensitive rewriting system* is a TRS with an *F*-map. If *s −→p t*

and *p ∈ P*os*μ*(*s*), then *s −→p*

*t* is said to be a *μ-reduction* denoted by *s −−→ t*.

*μ,R*

Let *→* be a binary relation on terms, the transitive closure of *→* is denoted by

*→*+. The transitive and reflexive closure of *→* is denoted by *→∗*. If *s →∗ t*, then we say that there is a *→-sequence* starting from *s* to *t* or *t* is *→-reachable from s*. We write *s →k t* if *t* is *→*-reachable from *s* with *k* steps. A term *t terminates* with respect to *→* if there exists no infinite *→*-sequence starting from *t*.

**Example 2.2** Let *R*1 = *{g*(*x*) *→ h*(*x*)*, h*(*d*) *→ g*(*c*)*, c → d}* and *μ*1(*g*) = *μ*1(*h*) =

*∅*. A *μ*1-reduction sequence starting from *g*(*d*) is *g*(*d*) *−−−−→ h*(*d*) *−−−−→ g*(*c*). We

*μ*1*,R*1 *μ*1*,R*1

can not reduce *g*(*c*) to *g*(*d*) because *c* is not a *μ*1-replacing subterm of *g*(*c*).

**Proposition 2.3** *For a TRS R and F-map μ, if there is a reduction s −−→ t, then*

*μ,R*

*C* [*s*] *−−→ C* [*t*] *for any μ-replacing context C .*

*μ μ μ*

*μ,R*

For a TRS *R* (and *F*-map *μ*), we say that *R* terminates (resp. innermost ter- minates, *μ*-terminates) if every term terminates with respect to *→* (resp. *−−−→* ,

*R*

*in,R*

*−−→* ).

*μ,R*

For a TRS *R*, a function symbol *f ∈ F* is *deﬁned* if *f* = root(*l*) for some rule *l → r ∈ R*. The set of all defined symbols of *R* is denoted by *DR* = *{*root(*l*) *| l → r ∈ R}*. A term *t* has a *deﬁned root symbol* if root(*t*) *∈ DR*.

Let *R* be a TRS over a signature *F*. The signature *F* denotes the union of

*F* and *D*

*R*

= *{f | f ∈ DR}* where *F ∩ D*

= *∅* and *f* has the same arity as *f* .

We call these fresh symbols *dependency pair symbols*. We define a notation *t* by

*R*

*t* = *f*(*t*1*,..., tn*) if *t* = *f* (*t*1*,..., tn*) and *f ∈ DR*, *t* = *t* if *t ∈ V*. If *l → r ∈ R*

and *u* is a subterm of *r* with a defined root symbol and *u / l*, then the rewrite rule

*l → u* is called a *dependency pair* of *R*. The set of all dependency pairs of *R* is

denoted by DP(*R*).

**Example 2.4** Let *R*2 = *{a → g*(*f* (*a*))*, f* (*f* (*x*)) *→ h*(*f* (*a*)*,f* (*x*))*}*. We have

DP(*R*2) = *{a → a, a → f*(*a*)*, f*(*g*(*x*)) *→ a, f*(*g*(*x*)) *→ f*(*a*)*}*.

A rule *l → r* is said to be right ground if *r* is ground. Right-ground TRSs are TRSs that consist of right-ground rules.

**Definition 2.5** [Semi-Constructor TRS] A TRS *R* is a *semi-constructor* system if every rule in DP(*R*) is right ground.

**Remark 2.6** The class of semi-constructor TRSs in this paper is a larger class of semi-constructor TRSs by the original definition because a rule *l → u* is not dependency pair if *u l*. The original definition of semi-constructor TRS is as follows [[11](#_bookmark48)]. A term *t ∈ T* (*F, V*) is a *semi-constructor* term if every term *s* such that *s Œ t* and root(*s*) *∈ DR* is ground. A TRS *R* is a semi-constructor system if *r* is a semi-constructor term for every rule *l → r ∈ R*.

**Example 2.7** The TRS *R*2 (in Example [2.4](#_bookmark3)) is a semi-constructor TRS but not in the original definition.

# Decidability of Innermost Termination for Semi- Constructor TRSs

Decidability of termination for semi-constructor TRSs is proved based on the ob- servation that there exists an infinite reduction sequence having a loop if it is not terminating [[13](#_bookmark50)]. In this section, we prove the decidability of innermost termination in a similar way.

**Definition 3.1** [loop] Let *→* be a relation on terms. A reduction sequence *loops* if it contains *t →*+ *C*[*t*] for some context *C*, and *head-loops* if containing *t →*+ *t*.

**Proposition 3.2** *If there exists an innermost sequence that loops, then there exists an inﬁnite innermost sequence.*

**Definition 3.3** [Innermost DP-chain] For a TRS *R*, a sequence of the elements

of DP(*R*) *s → t , s → t ,...* is an *innermost dependency chain* if there exist

1 1 2 2

substitutions *τ*1*, τ*2*,...* such that *sτi* is in normal forms and *tτi −−−→∗ s*

*τi*+1

*i*

holds for every *i*.

*i in,R*

*i*+1

**Theorem 3.4 ([**[**4**](#_bookmark41)**])** *For a TRS R, R does not innermost terminate if and only if there exists an inﬁnite innermost dependency chain.*

Let *M→* denote the set of all *minimal non-terminating terms* for a relation on terms *→* and an order on terms *≥*.

*≥*

**Definition 3.5** [*C*-min] For a TRS *R*, let *C ⊆* DP(*R*). An infinite reduction se-

*−−−→*

quence in *R∪C* in the form *t −−−−→ t −−−−→ t −−−−→ ···* with *ti ∈M in,R* for

1 *in,R∪C* 2 *in,R∪C* 3 *in,R∪C Ḇ*

all *i ≥* 1 is called a *C-min innermost reduction sequence*. We use *Cin*

*min*

the set of all *C*-min innermost reduction sequences starting from *t*.

(*t*) to denote

**Proposition 3.6 ([**[**4**](#_bookmark41)**])** *Given a TRS R, the following statements hold:*

1. *If there exists an inﬁnite innermost dependency chain, then Cin* (*t*) */*= *∅ for*

*−−−→*

*some C ⊆* DP(*R*) *and t ∈M in,R .*

*Ḇ*

*min*

1. *For any sequence in Cin* (*t*)*, reduction by rules of R takes place below the root*

*min*

*while reduction by rules of C takes place at the root.*

1. *For any sequence in Cin* (*t*)*, there is at least one rule in C which is applied*

*min*

*inﬁnitely often.*

**Lemma 3.7 ([**[**4**](#_bookmark41)**])** *For two terms s and s', s −−−−−→∗ s' implies s −−−→∗ C*[*s'*] *for*

*some context C.*

*in,R∪C*

*in,R*

**Proof.** We use induction on the number *n* of reduction steps in *s −−−−→n s'* . In

*in,R∪C*

the case that *n* = 0, *s −−−→∗ C*[*s'*] holds where *C* = . Let *n ≥* 1. Then we have

*in,R*

*s −−−−→n−*1 *s'' −−−−→ s'* for some *s''* . By the induction hypothesis, *s −−−→∗ C*[*s''*].

*in,R∪C*

*in,R∪C*

*in,R*

* Consider the case that *s'' −−−→ s'* . Since *s'' −−−→ s'*, we have *C*[*s''*] *−−−→ C*[*s'*] by

*in,R*

Proposition [2.1](#_bookmark1). Hence *s −−−→∗ C*[*s'*].

*in,R*

*in,R*

*in,R*

* Consider the case that *s'' −−→ s'* . Since *s''* is a normal form with respect to *→* ,

*R*

*in,C*

we have *s'' −−−→ C'*[*s'*] by the definition of dependency pairs. *C*[*s''*] *−−−→ C*[*C'*[*s'*]],

*in,R in,R*

by Proposition [2.1](#_bookmark1). Hence *s −−−→∗ C*[*C'*[*s'*]].

*in,R*

**Lemma 3.8** *For a semi-constructor TRS R, the following statements are equiva- lent:*

1. *R does not innermost terminate.*
2. *There exists l → u ∈* DP(*R*) *such that sq head-loops for some C ⊆* DP(*R*)

*and sq ∈ Cin* (*u*)*.*

*min*

**Proof.** ((ii) *⇒* (i)) : It is obvious from Lemma [3.7](#_bookmark10), and Proposition [3.2](#_bookmark5). ((i) *⇒* (ii))

: By Theorem [3.4](#_bookmark6) there exists an infinite innermost dependency chain. By Propo-

sition [3.6](#_bookmark7)([i](#_bookmark8)), there exists a sequence *sq ∈ Cin* (*t*). By Proposition [3.6](#_bookmark7)([ii](#_bookmark9)),([iii](#_bookmark11)),

*min*

there exists some rule *l → u ∈ C*, which is applied at root position in *sq* in- finitely often. By Definition [2.5](#_bookmark4), *u* is ground. Thus *sq* contains a subsequence

*u −−−−−−−−→∗ · → u*, which head-loops.

*{l →u }*

*in,R∪*DP(*R*)

**Theorem 3.9** *Innermost termination of semi-constructor TRSs is decidable.*

**Proof.** The decision procedure for the innermost termination of a semi-constructor TRS *R* is as follows: consider all terms *u*1*, u*2*,..., un* corresponding to the right-

hand sides of DP(*R*) = *{l → u |* 1 *≤ i ≤ n}*, and simultaneously generate all

*i*

*i*

innermost reduction sequences with respect to *R* starting from *u*1*, u*2*,..., un*. The

procedure halts if it enumerates all reachable terms exhaustively or it detects a looping reduction sequence *u −−−→*+ *C*[*u* ] for some *i*.

*i* *i*

*in,R*

Suppose *R* does not innermost-terminate. By Lemma [3.8](#_bookmark12) and [3.7](#_bookmark10), we have a looping reduction sequence *u −−−→*+ *C*[*u* ] for some *i* and *C*, which we eventually

*i* *i*

*in,R*

detect. If *R* innermost terminates, then the execution of the reduction sequence

generation eventually stops since the reduction relation is finitely branching. In the latter case, the procedure does not detect a looping sequence, otherwise it contradicts Proposition [3.2](#_bookmark5). Thus the procedure decides innermost termination of *R* in finitely many steps.

# Decidability of Context-Sensitive Termination for Semi-Constructor TRSs

The proof of decidability for innermost termination is straightforward. However, the proof for context-sensitive termination is not so straightforward because of the existence of a dependency pair whose right-hand side is variable.

**Definition 4.1** [*μ*-Loop] Let *→* be a relation on terms and *μ* be an *F*-map. A

*reduction sequence μ-loops* if it contains *t →*+ *Cμ*[*t*] for some context *Cμ*.

**Example 4.2** Let *R*3 = *{a → g*(*f* (*a*))*, f* (*g*(*x*)) *→ h*(*f* (*a*)*, x*)*}*, *μ*2(*f* ) = *{*1*}*,

*μ*2(*g*) = *∅* and *μ*2(*h*) = *{*1*,* 2*}*. The *μ*2-reduction sequence with respect to *R*3

*f* (*a*) *−−−−→ f* (*g*(*f* (*a*))) *−−−−→ h*(*f* (*a*)*,f* (*a*)) *−−−−→ ···* is *μ*2-looping.

*μ*2*,R*3

*μ*2*,R*3

*μ*2*,R*3

**Proposition 4.3** *If there exists a μ-looping μ-reduction sequence, then there exists an inﬁnite μ-reduction sequence.*

**Definition 4.4** [Context-Sensitive Dependency Pairs [[2](#_bookmark39)]] Let *R* be a TRS and *μ* be an *F*-map. We define DP(*R, μ*) = DP*F* (*R, μ*) *∪* DP*V* (*R, μ*) to be the set of *context-sensitive* dependency pairs where:

DP*F* (*R, μ*)= *{l → u | l → r ∈ R, u Œμ r,* root(*u*) *∈ DR,u /*

DP*V* (*R, μ*)= *{l → x | l → r ∈ R, x ∈* Var*μ*(*r*) *\* Var*μ*(*l*)*}*

*μl}*

**Example 4.5** Consider TRS *R*3 and *F*-map *μ*2 (in Example [4.2](#_bookmark14)). DP*F* (*R*3*, μ*2) =

*{f*(*g*(*x*)) *→ f*(*a*)*}* and DP*V* (*R*3*, μ*2) = *{f*(*g*(*x*)) *→ x}*.

For a given TRS *R* and an *F*-map *μ*, we define *μ* by *μ*(*f* ) = *μ*(*f* ) for *f ∈ F*,

and *μ*(*f*) = *μ*(*f* ) for *f ∈ DR*. We write *s o t* for *s oμ t*.

*μ*

**Definition 4.6** [Context-Sensitive Dependency Chain] For a TRS *R* and *F*-map *μ*, a sequence of the elements of DP(*R, μ*) *s → t , s → t , ...* is a *context-sensitive*

1 1 2 2

*dependency chain* if there exist substitutions *τ*1*, τ*2*,...* satisfying both:

* *tτi −−−→∗ s τi*+1, if *t /∈V*

*i μ,R*

*i*+1 *i*

* *xτ*

*o u −−−→∗ s τ*

for some term *u* , if *t* = *x*.

*i μ i*

*μ,R*

*i*+1

*i*+1 *i* *i*

**Example 4.7** Consider TRS *R*3 and *F*-map *μ*2 (in Example [4.2](#_bookmark14)).

*−−−−→*

*f* (*a*)*,f* (*g*(*f* (*a*))) *∈M μ*2*,R*3

*Ḇ*

*μ*

*−−−−→*

and *f* (*f* (*a*))*, h*(*f* (*a*)*,f* (*a*)) */∈M μ*2*,R*3 .

*Ḇ*

*μ*

**Theorem 4.8 ([**[**2**](#_bookmark39)**])** *For a TRS R and an F-map μ, there exists an inﬁnite context- sensitive dependency chain if and only if R does not μ-terminate.*

Let *R* be a TRS, *μ* be an *F*-map and *C ⊆* DP(*R, μ*). We define *‹−−−→* as

*μ,R,C*

( *−−−−→ ∪*( *−−−→ ·o* )*∪ −−−→* ) where *C*

= *C∩*DP

(*R, μ*) and *C*

= *C∩*DP

(*R, μ*).

*μ,CF*

*μ,CV*

*μ μ ,R F F V* *V*

**Definition 4.9** [*μ*-*C*-min] Let *R* be a TRS, *μ* be an *F*-map. An infinite sequence of terms in the form *t ‹−−−→ t ‹−−−→ t ‹−−−→ ···* is called a *C-min μ-sequence* if

1 *μ,R,C* 2 *μ,R,C* 3 *μ,R,C*

*−−→*

*ti ∈M μ,R*

*Ḇμ*

for all *i ≥* 1. We use *Cμ*

*min*

(*t*) to denote the set of all *C*-min *μ*-sequences

starting from *t*.

Note that *Cμ*

*min*

*−−→*

(*t*) = *∅* if *t /∈M μ,R* .

*Ḇ*

*μ*

**Example 4.10** Let *C* = DP(*R*3*, μ*2), the sequence *f*(*a*) *‹−−−−−→ f*(*g*(*f* (*a*)))

*μ*2*,R*3*,C*

*‹−−−−−→*

*μ*2*,R*3*,C*

*f*(*a*) *‹−−−−−→ ···* is a *C*-min *μ*-sequence.

*μ*2*,R*3*,C*

**Proposition 4.11 ([**[**2**](#_bookmark39)**])** *Given a TRS R and an F-map μ, the following statements hold:*

1. *If there exists an inﬁnite context-sensitive dependency chain, then Cμ* (*t*) */*= *∅*

*−−→*

*for some C ⊆* DP(*R, μ*) *and t ∈M μ,R .*

*Ḇ*

*μ*

*min*

1. *For any sequence in Cμ*

*min*

(*t*)*, a reduction with −−−→*

*μ,R*

*takes place below the root*

*while reductions with −−−−→*

*μ,CF*

*and −−−→*

*μ,CV*

*take place at the root.*

1. *For any sequence in Cμ* (*t*)*, there is at least one rule in C which is applied*

*min*

*inﬁnitely often.*

**Lemma 4.12** *For two terms s and t, s ‹−−−→∗ t implies s −−→∗ Cμ*[*t*] *for some*

*context Cμ.*

*μ,R,C*

*μ,R*

**Proof.** We use induction on the length *n* of the sequence. In the case that *n* = 0, it holds trivially. Let *n ≥* 1. Then we have *s ‹−−−→∗ u ‹−−−→ t* for some *u*.

*μ,R,C μ,R,C*

* + In the case that *u −−−−→ t*, we have *u −−→ C'* [*t*] by the definition of dependency

pairs.

*μ,CF*

*μ,R μ*

* + In the case that *u −−−→ v o t*, we have *u −−→ C''*[*v*] by the definition of depen-

*μ,CV μ*

*μ,R μ*

dency pairs and *v* = *C'''*[*t*]. Thus *u −−→ C''*[*C'''*[*t*]] = *C'* [*t*].

*μ μ,R μ μ μ*

* + In the case that *u −−−→ t*, we have *u −−→ C'* [*t*] for *C'* [ ] = .

*μ,R*

*μ,R μ μ*

Therefore *s −−→∗ Cμ*[*u*] *−−→ Cμ*[*C'* [*t*]] by the induction hypothesis and Proposi-

*μ,R*

*μ,R μ*

tion [2.3](#_bookmark2).

* 1. *Context-Sensitive Semi-Constructor TRS*

In this subsection, we discuss the decidability of *μ*-termination for context-sensitive semi-constructor TRSs.

**Definition 4.13** [Context-Sensitive Semi-Constructor TRS] For an *F*-map *μ*, a TRS *R* is a *context-sensitive semi-constructor (μ-semi-constructor) TRS* if all rules in DP*F* (*R, μ*) are right ground.

For an *F*-map *μ*, the class of *μ*-semi-constructor TRSs is a superclass of the class of semi-constructor TRSs from Definition [2.5](#_bookmark4) and [4.13](#_bookmark22).

For a TRS *R* and *F*-map *μ*, we say *R* is free from the infinite variable dependency chain (FFIVDC) if and only if there exists no infinite context-sensitive dependency

chain consisting of only elements in DP*V* (*R, μ*). If *R* is FFIVDC, then *Cμ*

*min*

(*t*) = *∅*

for any *C ⊆* DP*V* (*R, μ*) and any term *t*.

**Lemma 4.14** *Let μ be an F-map. If a μ-semi-constructor TRS R is FFIVDC, then the following statements are equivalent:*

1. *R does not μ-terminate.*
2. *There exists l → u ∈* DP*F* (*R, μ*) *such that sq head-loops for C ⊆* DP(*R, μ*)

*and some sq ∈ Cμ* (*u*)*.*

*min*

**Proof.** ((ii) *⇒* (i)) : It is obvious from Lemma [4.12](#_bookmark21), and Proposition [4.3](#_bookmark15). ((i)

*⇒* (ii)) : By Theorem [4.8](#_bookmark16) there exists an infinite context-sensitive dependency

chain. By Proposition [4.11](#_bookmark17)([i](#_bookmark18)), there exists a sequence *sq ∈ Cμ* (*t*). By Proposi-

*min*

tion [4.11](#_bookmark17)([ii](#_bookmark19)),([iii](#_bookmark20)) and the fact that *R* is FFIVDC, there is some rule in *l → u ∈ CF*

which is applied at the root position in *sq* infinitely often.

By Definition [4.13](#_bookmark22), *u* is ground. Thus *sq* contains a subsequence *u ‹−−−→*+ *u*,

*μ,R,C*

which head-loops and is in *Cμ* (*u*).

*min*

**Theorem 4.15** *Let μ be an F-map. If a μ-semi-constructor TRS R is FFIVDC, then μ-termination of R is decidable.*

**Proof.** The decision procedure for *μ*-termination of a *μ*-semi-constructor TRS *R* is as follows: consider all terms *u*1*, u*2*,..., un* corresponding to the right-hand sides of DP*F* (*R, μ*) = *{l → u |* 1 *≤ i ≤ n}*, and simultaneously generate all *μ*-reduction

*i* *i*

sequences with respect to *R* starting from *u*1*, u*2*,..., un*. The procedure halts if

it enumerates all reachable terms exhaustively or it detects a *μ*-looping reduction sequence *u −−→*+ *C* [*u* ] for some *i*.

*i μ i*

*μ,R*

Suppose *R* does not *μ*-terminate. By Lemma [4.14](#_bookmark23) and [4.12](#_bookmark21), we have a *μ*-looping reduction sequence *ui −−→*+ *Cμ*[*ui*] for some *i* and *Cμ*, which we eventually detect.

*μ,R*

If *R μ*-terminates, then the execution of the reduction sequence generation even- tually stops since the reduction relation is finitely branching. In the latter case, the procedure does not detect a *μ*-looping sequence, otherwise it contradicts to Proposition [4.3](#_bookmark15). Thus the procedure decides *μ*-termination of *R* in finitely many steps.

We have to check the FFIVDC property in order to use Theorem [4.15](#_bookmark24). How- ever, The FFIVDC property is not necessarily decidable. The following proposition provides a sufficient condition. The set DP1 (*R, μ*) is a subset of DP*V* (*R, μ*) defined

*V*

as follows:

DP1 (*R, μ*) = *{f*(*u*1*,..., uk*) *→ x ∈* DP*V* (*R, μ*) *| ∃i,* 1 *≤ i ≤ k, i /∈ μ*(*f* )*,x ∈ V ar*(*ui*)*}*

*V*

**Proposition 4.16 ([**[**2**](#_bookmark39)**])** *Let R be a TRS, μ be an F-map and C ⊆* DP1 (*R, μ*)*.*

*V*

*μ*

*C*

*min*

(*t*) = *∅ for any term t.*

If DP1 (*R, μ*) = DP*V* (*R, μ*) then *R* is FFIVDC by Proposition [4.16](#_bookmark25). Hence the following corollary directly follows from Theorem [4.15](#_bookmark24) and the fact that DP1 (*R, μ*) = DP*V* (*R, μ*) is decidable.

*V*

*V*

**Corollary 4.17** *For an F-map μ and a μ-semi-constructor TRS R, μ-termination of R is decidable if* DP*V* (*R, μ*) = DP1 (*R, μ*)*.*

*V*

* 1. *Semi-Constructor TRS*

In this subsection, we try to remove FFIVDC condition from the results of the previ- ous subsection. As a result, it appears that *μ*-termination of semi-constructor TRSs (not *μ*-semi-constructor) is decidable. The arguments of following Lemma [4.18](#_bookmark27) and

[4.19](#_bookmark28) are similar to those of Lemma 3.5 and Proposition 3.6 in [[3](#_bookmark40)].

**Lemma 4.18** *Consider a reduction s* = *Cμ* [*lθ*]*p −−−→ t* = *Cμ* [*rθ*]*p* = *C'*[*u*]*q*

*−−→*

*where s, u ∈M μ,R*

*Ḇ*

*μ*

*μ,R*

*and q ∈ P*os(*t*)*\P*os*μ*(*t*)*. Then one of the following statements*

*holds*

1. *s D u*
2. *vθ* = *u and r* = *C''*[*v*]*q' for some θ, v /∈ V, C'', and q' ∈ P*os(*r*) *\ P*os*μ*(*r*)

**Proof.** Since *q ∈ P*os(*t*) *\ P*os*μ*(*t*), *p* is not below or equal to *q*. In the case that *p* and *q* are in parallel positions, *s D u* trivially holds. In the case that *p* is above *q*, it is obvious that *s D u* holds or, *vθ* = *u* and *r* = *C''*[*v*]*q'* for some *θ*, *v /∈ V*, *C''*. Here the fact that *q' ∈ P*os(*r*) *\ P*os*μ*(*r*) follows from *p ∈ P*os*μ*(*t*) and *q /∈ P*os*μ*(*t*).

**Lemma 4.19** *Let R be a semi-constructor TRS, μ be an F-map. For a C-min*

*μ-sequence s −−−→∗ t −−−→*

*u o s −−−→∗ t −−−→*

*u o* *with no reduction*

1 *μ,R* 1 *μ,CV* 1 *μ* 2 *μ,R* 2 *μ,CV* 2 *μ*

*by rules in CF , one of the following statements holds for each i:*

1. *si D si*+1
2. *There exists l → s ∈* DP(*R*) *for some l*

*i*+1

**Proof.** Since *t −−−→ u*

*o s*

, we have *t* = *C*[*s* ] for some *q ∈ P*os(*t* ) *\*

*i μ,CV*

*i μ i*+1

*i i*+1 *q* *i*

*P*os*μ*(*ti*). We show (i) or the following (ii’) by induction on the number *n* of steps

of *s −−−→n t* = *C*[*si*+1].

*i μ,R i*

(ii’) There exists a reduction by *l → r* in *s −−−→∗ t* and *l → s ∈* DP(*R*)

*i μ,R i*

*i*+1

* In the case that *n* = 0, trivially *si* = *ti D si*+1.
* In the case that *n >* 0, let *s −−−→ s' −−−→n−*1 *t* = *C*[*si*+1]*q*. By the induction

*i μ,R μ,R* *i*

hypothesis, *s' D si*+1 or the condition (ii’) follows. In the former case, we have *si D si*+1, or, we have *vθ* = *si*+1 and *r* = *C'*[*v*]*q'* for some *l → r ∈ R*, *θ*, *v /∈ V*, *C'* and *q' ∈ P*os(*r*) *\ P*os*μ*(*r*) by Lemma [4.18](#_bookmark27). Hence *vθ* = *v* due to root(*si*+1) *∈ DR*

and Definition [2.5](#_bookmark4). Therefore (ii’) follows.

One may think that the Lemma [4.19](#_bookmark28) would hold even if DP(*R*) were replaced with DP(*R, μ*). However, it does not hold as shown by the following counter exam- ple.

**Example 4.20** Consider the semi-constructor TRS *R*4 = *{f* (*g*(*x*)) *→ x, g*(*b*) *→*

*g*(*f* (*g*(*b*)))*}*, *μ*3(*f* ) = *{*1*}* and *μ*3(*g*) = *∅*. There exists a *C*-min *μ*3-sequence

*f*(*g*(*b*)) *−−−−→*

*f*(*g*(*f* (*g*(*b*))) *−−−→ f* (*g*(*b*)) *o*

*f*(*g*(*b*)) where *C*

= DP (*R ,μ* ).

*μ ,R*4

3

*μ*3

*μ ,CV*

3

*V V* 4 3

However there exists no dependency pair having *f*(*g*(*b*)) in the right-hand side in DP(*R, μ*).

**Lemma 4.21** *For a semi-constructor TRS R and an F-map μ, the following state- ments are equivalent:*

1. *R does not μ-terminate.*
2. *There exists l → u ∈* DP(*R*) *such that sq head-loops for C ⊆* DP(*R, μ*) *and*

*some sq ∈ Cμ* (*u*)*.*

*min*

**Proof.** ((ii) *⇒* (i)) : It is obvious from Lemma [4.12](#_bookmark21), and Proposition [4.3](#_bookmark15). ((i) *⇒*

(ii)) : By Theorem [4.8](#_bookmark16) there exists a context-sensitive dependency chain. By Propo-

sition [4.11](#_bookmark17)([i](#_bookmark18)), there exists a sequence *sq ∈ Cμ* (*t*). By Proposition [4.11](#_bookmark17)([ii](#_bookmark19)),([iii](#_bookmark20)),

*min*

there exists a rule in *C* applied at root position in *sq* infinitely often.

* Consider the case that there exists a rule *l → r ∈ CF* with infinite use in *sq*. Since *u* is ground by Proposition [4.11](#_bookmark17)([ii](#_bookmark19)) and *CF ⊆* DP(*R*), *sq* has a subsequence *u ‹−−−→*+ *u*.

*μ,R,C*

* Otherwise, *sq* has an infinite subsequence without the use of the rules in *CF* . The

subsequence is in *Cμ* (*s*) for some *s*. Then the condition (ii) of Lemma [4.19](#_bookmark28) holds

*min*

for infinitely many *i*’s; otherwise, we have an infinite sequence *sk D sk*+1 *D ···* for

some *k*, which is a contradiction. Hence there exists a *l → u ∈* DP(*R*) such that

*u* occurs more than once in *sq*. Thus the sequence *u ‹−−−→*+ *u* appears in *sq*.

*μ,R,C*

**Theorem 4.22** *The property μ-termination of semi-constructor TRSs is decidable.*

**Proof.** The decision procedure for *μ*-termination of a semi-constructor TRS *R* is as follows: consider all terms *u*1*, u*2*,..., un* corresponding to the right-hand sides of DP(*R*) = *{l → u |* 1 *≤ i ≤ n}*, and simultaneously generate all *μ*-reduction

*i* *i*

sequences with respect to *R* starting from *u*1*, u*2*,..., un*. The procedure halts if

it enumerates all reachable terms exhaustively or it detects a *μ*-looping reduction sequence *u −−→*+ *C* [*u* ] for some *i*.

*i μ i*

*μ,R*

Suppose *R* does not *μ*-terminate. By Lemma [4.21](#_bookmark29) and [4.12](#_bookmark21), we have a *μ*-looping reduction sequence *ui −−→*+ *Cμ*[*ui*] for some *i* and *Cμ*, which we eventually detect.

*μ,R*

If *R μ*-terminates, then the execution of the reduction sequence generation even- tually stops since the reduction relation is finitely branching. In the latter case, the procedure does not detect a *μ*-looping sequence, otherwise it contradicts to Proposition [4.3](#_bookmark15). Thus the procedure decides *μ*-termination of *R* in finitely many steps.

# Extending the Classes by DP-graphs

* 1. *Innermost Termination*

In this subsection, we extend the class for which innermost termination is decidable by using the dependency graph.

**Lemma 5.1** *Let R be a TRS whose innermost termination is equivalent to the non-existence of an innermost dependency chain that contains inﬁnite use of right- ground dependency pairs. Then innermost termination of R is decidable.*

**Proof.** We apply the procedure used in the proof of Lemma [3.9](#_bookmark13) starting with terms *u*1*, u*2*,..., un*, where *u*’s are all ground right-hand sides of dependency pairs. Suppose *R* is innermost non-terminating, then we have an innermost dependency chain with infinite use of a right-ground dependency pair. Similarly to the semi- constructor case, we have a looping sequence *u −−−→*+ *C*[*u* ], which can be detected

*i*

*i* *i*

*in,R*

by the procedure.

**Definition 5.2** [Innermost DP-Graph [[4](#_bookmark41)]] The *innermost dependency graph* (in- nermost DP-graph for short) of a TRS *R* is a directed graph whose nodes are the dependency pairs and there is an arc from *s → t* to *u → v* if there exist normal substitutions *σ* and *τ* such that *tσ −−−→∗ uτ* and *uτ* is a normal form with respect

*in,R*

to *R*.

An approximated innermost DP-graph is a graph that contains the innermost DP-graph as a subgraph. Such computable graphs are proposed in [[4](#_bookmark41)], for example.

**Theorem 5.3** *Let R be a TRS and G be an approximated innermost DP-graph of R. If at least one node in the cycle is right-ground for every cycle of G, then innermost termination of R is decidable.*

**Proof.** From Lemma [5.1](#_bookmark31).

**Example 5.4** Let *R*5 = *{f* (*s*(*x*)) *→ g*(*x*)*, g*(*s*(*x*)) *→ f* (*s*(0))*}*. Then DP(*R*5) =

*{f*(*s*(*x*)) *→ g*(*x*)*, g*(*s*(*x*)) *→ f*(*s*(0))*}*. The innermost DP-graph of *R*5 has one cycle, which contains a right-ground node [Fig. [1](#_bookmark33)]. The innermost termination of *R*5 is decidable by Theorem [5.3](#_bookmark32). Actually we know *R*5 is innermost terminating from the procedure in the proof of Theorem [3.9](#_bookmark13) since all innermost reduction sequences from *f* (*s*(0)) terminate.

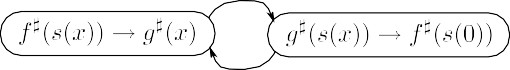


Fig. 1. The innermost DP-graph of *R*5

**Example 5.5** Let *R*6 = *{a → b, f* (*a, x*) *→ x, f* (*x, b*) *→ g*(*x, x*)*, g*(*b, x*) *→ h*(*f* (*a, a*)*, x*)*}*. Then DP(*R*6) = *{f*(*x, b*) *→ g*(*x, x*)*, g*(*b, x*) *→ f*(*a, a*)*, g*(*b, x*) *→ a}*. The innermost DP-graph of *R*6 has one cycle, which contains a right- ground node [Fig. [2](#_bookmark34)]. The innermost termination of *R*6 is decidable by Theorem [5.3](#_bookmark32). Actually we know *R*6 is not innermost terminating from the procedure in the proof of Theorem [3.9](#_bookmark13) by detecting the looping sequence *f* (*a, a*) *−−−→ f* (*b, b*) *−−−→ g*(*b, b*) *−−−→ h*(*f* (*a, a*)*, b*).

*in,R*6

*in,R*6

*in,R*6

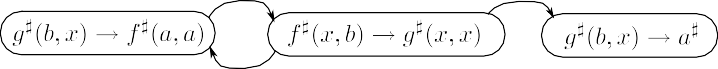


Fig. 2. The innermost DP-Graph of *R*6

* 1. *Context-Sensitive Termination*

We extend the class for which *μ*-termination is decidable by using the dependency graph. The class extended in this subsection is the class that satisfies the condition of Corollary [4.17](#_bookmark26).

**Lemma 5.6** *Let R be a TRS and μ be an F-map. If μ-termination of R is equiv- alent to the non-existence of a context-sensitive dependency chain that contains inﬁnite use of right-ground rules in* DP*F* (*R, μ*)*, then μ-termination of R is decid- able.*

**Proof.** We apply the procedure used in the proof of Lemma [4.22](#_bookmark30) starting with terms *u*1*, u*2*,..., un*, where *u*’s are all ground right-hand sides of rules in DP*F* (*R, μ*). Suppose *R* is non-*μ*-terminating, then we have a context-sensitive dependency chain with infinite use of right-ground rules in DP*F* (*R, μ*). Similar to the *μ*-semi- constructor case, we have a looping sequence *u −−→*+ *C* [*u* ], which can be detected

*i*

*i μ i*

*μ,R*

by the procedure.

**Definition 5.7** [Context-Sensitive DP-Graph [[2](#_bookmark39)]] The *context-sensitive dependency graph* (context-sensitive DP-graph for short) of a TRS *R* and an *F*-map *μ* is a directed graph whose nodes are elements of DP(*R, μ*):

* + 1. There is an arc from *s → t ∈* DP*F* (*R, μ*) to *u → v ∈* DP(*R, μ*) if there exist substitutions *σ* and *τ* such that *tσ −−−→∗ uτ* .

*μ,R*

* + 1. There is an arc from *s → t ∈* DP*V* (*R, μ*) to each dependency pair *u → v ∈*

DP(*R, μ*).

Similar to the innermost case, a computable approximated context-sensitive DP- graph is proposed [[2](#_bookmark39),[3](#_bookmark40)].

**Theorem 5.8** *Let R be a TRS, μ be an F-map and G be an approximated context- sensitive DP-graph of R. The property μ-termination of R is decidable if one of following holds for every cycle in G.*

1. *The cycle contains at least one node that is right-ground.*
2. *All nodes in the cycle are elements in* DP1 (*R, μ*)*.*

*V*

**Proof.** From Lemma [5.6](#_bookmark35) and Theorem [4.16](#_bookmark25).

**Example 5.9** Let *R*7 = *{h*(*x*) *→ g*(*x, x*)*, g*(*a, x*) *→ f* (*b, x*)*, f* (*x, x*) *→ h*(*a*)*, a →*

*b}* and *μ*4(*f* ) = *μ*4(*g*) = *μ*4(*h*) = *{*1*}* [[10](#_bookmark47)]. Then DP(*R*7*, μ*4) = *{h*(*x*) *→*

*g*(*x, x*)*, g*(*a, x*) *→ f*(*b, x*)*, f*(*x, x*) *→ h*(*a*)*, f*(*x, x*) *→ a}*. The context- sensitive DP-graph of *R*7 and *μ*4 has one cycle, which contains a right-ground node [Fig.[3](#_bookmark37)]. The *μ*4-termination of *R*7 is decidable by Theorem [5.8](#_bookmark36). Actually we know

*R*7 is *μ*4-terminating from the procedure in the proof of Theorem [4.15](#_bookmark24) since all

*μ*4-reduction sequences from *h*(*a*) terminate.

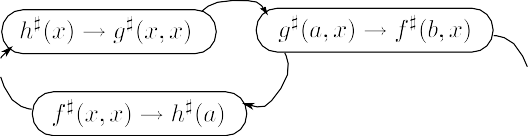


Fig. 3. The context-sensitive DP-Graph of *R*7 and *μ*4

**Example 5.10** Let *μ*5(*g*) = *{*2*}* and *μ*5(*f* ) = *μ*5(*h*) = *{*1*}*. Consider the *μ*5-termination of *R*7. The context-sensitive DP-graph for *R*7 and *μ*5 is the same as the one for *R*7 and *μ*4 [Fig.[3](#_bookmark37)]. The *μ*5-termination of *R*7 is decidable by Theorem [5.8](#_bookmark36). By the decision procedure, we can detect the *μ*5-looping se- quence *h*(*a*) *−−−−→ g*(*a, a*) *−−−−→ g*(*a, b*) *−−−−→ f* (*b, b*) *−−−−→ h*(*a*). Thus *R*7 is non-

*μ*5*,R*7

*μ*5-terminating.

*μ*5*,R*7

*μ*5*,R*7

*μ*5*,R*7

The class of TRSs that satisfy the conditions of Theorem [5.8](#_bookmark36) is a superclass of the class of TRS that satisfy the conditions of Corollary [4.17](#_bookmark26). The class of semi- constructor TRSs and the class of TRSs that satisfy the conditions of Theorem [5.8](#_bookmark36) are not included in each other.

**Example 5.11** The TRS *R*7 with an *F*-map *μ*4 satisfies the condition of Theo- rem [5.8](#_bookmark36), but is not semi-constructor TRS. On the other hand, the TRS *R*3 with an *F*-map *μ*2 is a semi-constructor TRS, but does not satisfy the second condition of Theorem [5.8](#_bookmark36).

# Conclusion

We have shown that innermost termination for semi-constructor TRSs is a decid- able property and *μ*-termination for semi-constructor TRSs and *μ*-semi-constructor TRSs are decidable properties.

It is not difficult to implement the procedures in proofs of Theorem [3.9](#_bookmark13), Theo- rem [4.15](#_bookmark24) and Theorem [4.22](#_bookmark30). The class of semi-constructor TRSs are a rather small class: approximately 3 % of the TRSs in the termination problem data base 4.0 [[1](#_bookmark38)] are in this class. We can extend the decidable classes if we succeed in developing a method for good approximated DP-graphs.

In the future we will study the decidability of innermost termination and *μ*- termination by applying known techniques for termination results [[7](#_bookmark44),[13](#_bookmark50)]. Currently, innermost termination for shallow TRSs is known to be decidable [[7](#_bookmark44)]. There are several future works, studying whether the condition FFIVDC is removed from Theorem [4.15](#_bookmark24) or not, and extending the class of semi-constructor TRSs by using notions of context-sensitive DP-graph.

# Acknowledgement

We would like to thank the anonymous referees for their helpful comments and remarks. This work is partly supported by MEXT.KAKENHI #18500011 and #16300005.

# References

1. The termination problems data base. [http://www.lri.fr/˜marc](http://www.lri.fr/)he/tpdb/.
2. B. Alarc´on, R. Guti´errez, and S. Lucas. Context-sensitive dependency pairs. In *the 26th Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 4337 of *Lecture Notes in Computer Science*, pages 298–309, 2006.
3. B. Alarc´on, R. Guti´errez, and S. Lucas. Improving the context-sensitive dependency graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.
4. T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236:133–178, 2000.
5. F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.
6. N. Dershowitz. Termination of linear rewriting systems. In *the 8th International Colloquium on Automata, Languages and Programming*, volume 115 of *Lecture Notes in Computer Science*, pages 448–458, 1981.
7. G. Godoy, E. Huntingford, and A. Tiwari. Termination of rewriting with right-flat rules. In *the 18th* *International Conference on Rewriting Techniques and Applications*, volume 4533 of *Lecture Notes in Computer Science*, pages 200 – 213, 2007.
8. G. Godoy and A. Tiwari. Termination of rewrite systems with shallow right-linear, collapsing, and right-ground rules. In *the 20th International Conference on Automated Deduction*, volume 3632 of *Lecture Notes in Computer Science*, pages 164–176, 2005.
9. G. Huet and D. Lankford. On the uniform halting problem for term rewriting systems. Technical report, INRIA, 1978.
10. S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204:1782–1846, 2006.
11. I. Mitsuhashi, M. Oyamaguchi, Y. Ohta, and T. Yamada. The joinability and unification problems for confluent semi-constructor trss. In *the 15th International Conference on Rewriting Techniques and Applications*, volume 3091 of *Lecture Notes in Computer Science*, pages 285 – 300, 2004.
12. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Information and Computation*, 178:499–514, 2002.
13. Y. Wang and M. Sakai. Decidability of termination for semi-constructor trss, left-linear shallow trss and related systems. In *the 17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 343–356, 2006.