Convergent Term Rewriting Systems for Inverse Computation of Injective Functions*

Naoki Nishida, Masahiko Sakai, and Terutoshi Kato

Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
{nishida,sakai}@is.nagoya-u.ac.jp kato@sakabe.i.is.nagoya-u.ac.jp

Abstract. This paper shows a sufficient syntactic condition for constructor TRSs whose inverse-computation CTRSs generated by Nishida, Sakai and Sakabe's inversion compiler are confluent and operationally terminating. By replacing the unraveling at the second phase of the compiler with Serbanuta and Rosu's transformation, we generate convergent TRSs for inverse computation of injective functions satisfying the sufficient condition.

1 Introduction

Given a program written in a functional language, an *inversion compiler* for the language generates another program written in the same language, so-called an *inverse(-computation) program* of the given program, that defines inverses of functions defined in the given one. Several inversion compilers for functional languages have been proposed [8, 2, 4, 5]. Inversion compilers are useful in automatically generating inverse programs that should have high reliability, such as *compression/decompression* tools and *shared key encryption/decryption* functions. Therefore, developing the compilers and theoretically proving their correctness are valuable.

The inversion compiler proposed in [4, 5] is applicable to constructor term rewriting systems. Given a term rewriting system (TRS), it first generates an inverse conditional TRS as an intermediate result, and then transforms the conditional TRS (CTRS) into a TRS that is equivalent with the CTRS with respect to inverse computation. The first phase of the compiler works as an *inversion* by itself. At the second phase, the compiler employs a variant of Ohlebusch's *unraveling* [6] that is a transformation from deterministic 3-CTRSs into TRSs. Unfortunately, inverse computation by the generated TRSs sometimes have several garbage normal forms that mean dead ends by wrong choices at inverse-computation branches. Note that given a normal form, it is decidable whether the normal form is a solution or a garbage. The cause of the occurrence of such normal forms is that unravelings generally produce TRSs approximating the corresponding CTRSs [6]. We face this problem even when we restrict functions

* This work is partly supported by MEXT. KAKENHI #17700009 and #18500011, and Kayamori Foundation of Informational Science Advancement.

This is the full version of the paper submitted to the 9th International Workshops on Termination (WST'07) and has the appendix where the proofs of the theorems are described.

to injective ones. For example, consider the following constructor TRS where $\mathsf{Snoc}(xs, y)$ produces the list obtained from xs by adding y as the last element:

$$R_1 = \left\{ \begin{array}{c} \mathsf{Snoc}(\mathsf{nil}, y) \to \mathsf{cons}(y, \mathsf{nil}) \\ \mathsf{Snoc}(\mathsf{cons}(x, xs), y) \to \mathsf{cons}(x, \mathsf{Snoc}(xs, y)). \end{array} \right.$$

The compiler transforms R_1 into the following TRS:

$$\mathbb{U}(\mathcal{I}nv(R_1)) = \left\{ \begin{array}{c} \mathsf{InvSnoc}(\mathsf{cons}(y, \mathsf{nil})) \to (\mathsf{nil}, y) \\ \mathsf{InvSnoc}(\mathsf{cons}(x, ys)) \to \mathsf{U}_1(\mathsf{InvSnoc}(ys), x) \\ \mathsf{U}_1((xs, y), x) \to (\mathsf{cons}(x, xs), y) \\ \mathsf{InvSnoc}(\mathsf{Snoc}(xs, y)) \to (xs, y). \end{array} \right.$$

Here, we abbreviate the tuple $\mathsf{tp}_n(t_1, \ldots, t_n)$ of n terms $t_1, \ldots, t_n$ to $(t_1, \ldots, t_n)$, and the list $\mathsf{cons}(t_1, \mathsf{cons}(t_2, \cdots, \mathsf{cons}(t_n, \mathsf{nil}) \cdots))$ to $\mathtt{[}t_1, t_2, \ldots, t_n\mathtt{]}$. The unique normal form of $\mathsf{Snoc}(\mathtt{[}\mathsf{A}, \mathsf{B}\mathtt{]}, \mathsf{C})$ is $\mathtt{[}\mathsf{A}, \mathsf{B}, \mathsf{C}\mathtt{]}$ but $\mathsf{InvSnoc}(\mathtt{[}\mathsf{A}, \mathsf{B}, \mathsf{C}\mathtt{]})$ has two normal forms, a solution $(\mathtt{[}\mathsf{A}, \mathsf{B}\mathtt{]}, \mathsf{C})$ and a garbage $\mathsf{U}_1(\mathsf{U}_1(\mathsf{U}_1(\mathsf{InvSnoc}(\mathsf{nil}), \mathsf{C}), \mathsf{B}), \mathsf{A})$.

In this paper, we propose a method to generate convergent inverse TRSs of injective functions. More precisely, we show a sufficient syntactic condition for input constructor TRSs whose inverse CTRSs generated by the compiler are confluent and operationally terminating [3], and then we show that Serbanuta and Rosu's transformation [7] from CTRSs into TRSs generates convergent TRSs from the intermediate CTRSs of the compilers if the input TRSs satisfy the sufficient condition. Finally, we show an example of non-injective functions such that Serbanuta and Rosu's transformation does not preserve confluence of the inverse CTRSs, and another example that their transformation does not preserve operational termination of the inverse CTRSs. The proofs of the theorems in this paper are described in the appendix.

This paper follows the general notions of term rewriting [6].

2 Inversion Compiler for Constructor TRSs

In this section, using an example, we briefly explain the first phase of the inversion compiler for constructor TRSs [5], and some of its properties.

Consider the TRS R_1 again. By introducing a fresh variable for each subterm in the right-hand side of every rule that is rooted with a defined symbol, we obtain from R_1 the following CTRS:

$$R_1' = \left\{ \begin{array}{c} \mathsf{Snoc}(\mathsf{nil}, y) \to \mathsf{cons}(y, \mathsf{nil}) \\ \mathsf{Snoc}(\mathsf{cons}(x, xs), y) \to \mathsf{cons}(x, ys) \Leftarrow \mathsf{Snoc}(xs, y) \to ys. \end{array} \right.$$

The first phase of the compiler, denoted by $\mathcal{I}nv$, exchanges the both sides of rules and conditions, reverses the order of conditional parts, applies inverse symbols, removes $InvF(F(\ ))$, adds some special rules (necessary for partial functions [4, 5]), and then generates the following CTRS as an intermediate result:

$$\mathcal{I}nv(R_1) = \left\{ \begin{array}{ll} \mathsf{InvSnoc}(\mathsf{cons}(y, \mathsf{nil})) \to (\mathsf{nil}, y) & \\ \mathsf{InvSnoc}(\mathsf{cons}(x, ys)) \to (\mathsf{cons}(x, xs), y) \Leftarrow \mathsf{InvSnoc}(ys) \to (xs, y) & \\ \mathsf{InvSnoc}(\mathsf{Snoc}(xs, y)) \to (xs, y). & \text{(special rule)} \end{array} \right.$$

Theorem 1 ([5]). *Let R be a convergent constructor TRS.*

- *$\mathcal{I}nv(R)$ is a non-erasing constructor deterministic CTRS.*
- *If R is non-erasing, then $\mathcal{I}nv(R)$ is a 3-CTRS.*
- *Let F be an n-ary defined symbol of R, and $t_1, \ldots, t_n, s$ be normal forms of R. $F(t_1, \ldots, t_n) \xrightarrow{*}_R s$ if and only if $InvF(s) \to_{\mathcal{I}nv(R)} (t_1, \ldots, t_n)$.*

3 Convergence of Inverse Systems for Injective Functions

In this section, we first give a sufficient syntactic condition for input constructor TRSs whose inverse CTRSs generated by $\mathcal{I}nv$ are confluent and operationally terminating. Then, we show that in this case, Serbanuta and Rosu's transformation [7] generates convergent inverse TRSs.

Definition 2. *Let R be a convergent constructor TRS. A defined symbol F of R is called* injective (with respect to normal forms) *if for all normal forms $s_1, \ldots, s_n$ and $t_1, \ldots, t_n$ of R, $F(s_1, \ldots, s_n) \downarrow_R F(t_1, \ldots, t_n)$ implies $s_i \equiv t_i$ for all i. The TRS R is called* injective (with respect to normal forms) *if all of its defined symbols are injective.*

Proposition 3. *Every injective TRS is non-erasing.*

Theorem 4. *Let R be a non-erasing constructor TRS. Suppose that for every rule $F(u_1, \ldots, u_n) \to r$ in R, if r is not a variable, then the root symbol of r does not depend* [1] *on F (either a constructor or a defined symbol not depending on F). Then, all of the following hold:*

(a) the CTRS $\mathcal{I}nv(R)$ is operationally terminating, and
(b) if R is injective, then the CTRS $\mathcal{I}nv(R)$ is confluent.

Note that the CTRS $\mathcal{I}nv(R)$ is convergent if R is injective, because operational termination implies termination (non-existence of infinite reduction sequences).

Serbanuta and Rosu's transformation [7], denoted by $\mathbb{T}$, can preserve convergence when transforming CTRSs into TRSs. Their transformation introduce the special constant $\bot$ and the unary symbol $\{\}$, and extends the arities of defined symbols of CTRSs S. More precisely, the new arity of an n-ary defined symbol F is $n+m$ where m is the number of conditions in F-rules in S. For a term t in the original signature, $\overline{t}$ denotes the term on the extended signature, that is obtained from t by adding $\bot$ to the extended arguments of defined symbols in t.

Example 5. The CTRS $\mathcal{I}nv(R_1)$ is transformed by $\mathbb{T}$ into $\mathbb{T}(\mathcal{I}nv(R_1))$ in Fig. 1 [7]. The ground term $\mathsf{InvSnoc}(\texttt{[}\mathsf{A},\mathsf{B},\mathsf{C}\texttt{]},\bot)$ $(= \overline{\mathsf{InvSnoc}(\texttt{[}\mathsf{A},\mathsf{B},\mathsf{C}\texttt{]})})$ has the unique ground normal form $\{(\texttt{[}\mathsf{A},\mathsf{B}\texttt{]},\mathsf{C})\}$ of $\mathbb{T}(\mathcal{I}nv(R_1))$.

[1] We say that an n-ary symbol G of R *depends on a symbol F* if (G, F) is in the transitive closure of the relation $\{\, (G', F') \mid G'(\cdots) \to C[F'(\cdots)] \in R \,\}$.

$$\left\{\begin{array}{r}
\mathsf{InvSnoc}(\mathsf{cons}(y,\mathsf{nil}),z) \to \{(\mathsf{nil},y)\} \\
\mathsf{InvSnoc}(\mathsf{cons}(x,ys),\bot) \to \mathsf{InvSnoc}(\mathsf{cons}(x,ys),\{\mathsf{InvSnoc}(ys,\bot)\}) \\
\mathsf{InvSnoc}(\mathsf{cons}(x,ys),\{(xs,y)\}) \to \{(\mathsf{cons}(x,xs),y)\} \\
\mathsf{InvSnoc}(\mathsf{Snoc}(xs,y),z) \to \{(xs,y)\} \\
\{\{x\}\} \to \{x\}, \quad \mathsf{InvSnoc}(\{xs\},z) \to \{\mathsf{InvSnoc}(xs,\bot)\} \\
\mathsf{cons}(\{x\},xs) \to \{\mathsf{cons}(x,xs)\}, \quad (\{x\},y) \to \{(x,y)\}, \quad \mathsf{Snoc}(\{xs\},y) \to \{(xs,y)\} \\
\mathsf{cons}(x,\{xs\}) \to \{\mathsf{cons}(x,xs)\}, \quad (x,\{y\}) \to \{(x,y)\}, \quad \mathsf{Snoc}(xs,\{y\}) \to \{(xs,y)\}.
\end{array}\right.$$

Fig. 1. Rewrite rules in $\mathbb{T}(\mathcal{I}nv(R_1))$.

Theorem 6 ([7]). *Let S be a deterministic 3-CTRS. If S is finite, ground confluent and operationally terminating on t, then $\mathbb{T}(S)$ is computationally equivalent with S, that is,*

- *$\mathbb{T}$ is sound and complete ($s \xrightarrow{*}_S t$ if and only if $\overline{s} \xrightarrow{*}_{\mathbb{T}(S)} \overline{t}$ for any $s,t \in \mathcal{T}(\mathcal{F})$), and*
- *$\mathbb{T}(S)$ is ground confluent and terminating on terms reachable from $\overline{t}$.*

Corollary 7. *Let R be an injective TRS that satisfies the assumption in Theorem 4, F be an n-ary defined symbol of R, and $t_1,\ldots,t_n,t$ be ground normal forms of R such that $F(t_1,\ldots,t_n) \xrightarrow{*}_R t$. Then, $\mathbb{T}(\mathcal{I}nv(R))$ is terminating on $InvF(t,\bot,\ldots,\bot)$ that has the unique ground normal form $\{(t_1,\ldots,t_n)\}$.*

Note that in the above corollary, $\overline{t} \equiv t$ because every functional symbol $F \in \mathcal{F}$ is a constructor of $\mathcal{I}nv(R)$. For every data list ts, R_1 is ground-convergent on $\mathsf{InvSnoc}(ts,\bot)$ and computationally equivalent with $\mathcal{I}nv(R_1)$ because R_1 satisfies the assumption in Theorem 4.

4 Discussion on Non-Injective Cases

Let's consider the following constructor TRS:

$$R_2 = \left\{ \mathsf{D}(x) \to \mathsf{Add}(x,x), \quad \mathsf{Add}(0,y) \to y, \quad \mathsf{Add}(\mathsf{s}(x),y) \to \mathsf{s}(\mathsf{Add}(x,y)) \right\}.$$

The defined symbol D is injective and R_2 is convergent. However, R_2 is not injective because Add is not injective. The term $\mathsf{InvD}(\mathsf{s}^{2n}(0))$ ($= \overline{\mathsf{InvD}(\mathsf{s}^{2n}(0))}$) for some n (> 0) has more than two normal forms of $\mathbb{U}(\mathcal{I}nv(R_2))$ and $\mathbb{T}(\mathcal{I}nv(R_2))$, respectively, although the CTRS $\mathcal{I}nv(R_2)$ is confluent on $\mathsf{InvD}(\mathsf{s}^{2n}(0))$. Thus, similarly to the unraveling $\mathbb{U}$, Serbanuta and Rosu's transformation $\mathbb{T}$ cannot preserve confluence of CTRSs for inverses of non-injective TRSs.

Next, we give an example showing that $\mathbb{T}$ can generates non-terminating inverse TRSs for TRSs with erasing rules. When input TRSs have erasing rules, the generated CTRSs are not 3-CTRSs, that is, the CTRSs have extra variables in the right-hand side not in the conditional part. In such cases, *narrowing* can be used for inverse computation by the unraveled inverse CTRSs [4, 5]. Consider the following constructor TRS computing multiplication:

$$R_3 = R_2 \cup \left\{ \begin{array}{ll} \mathsf{Mul}(\mathsf{0}, y) \to \mathsf{0}, & \mathsf{Mul}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{s}(\mathsf{Add}(\mathsf{Mul}(x, \mathsf{s}(y)), y)) \\ \mathsf{Mul}(x, \mathsf{0}) \to \mathsf{0} & \end{array} \right\}.$$

Narrowing from $\mathsf{InvMul}(\mathsf{s}^n(\mathsf{0}), \bot, \bot)$ does not terminate on $\mathbb{T}(\mathcal{I}nv(R_3))$ while narrowing from $\mathsf{InvMul}(\mathsf{s}^n(\mathsf{0}))$ does on $\mathbb{U}(\mathcal{I}nv(R_3))$ and gives us desired solutions. The cause of non-termination is the added rules $c(x_1, \ldots, \{x_i\}, \ldots, x_n) \to \{c(x_1, \ldots, x_n)\}$ where c is a constructor. For example, we have the infinite narrowing sequence $\overline{\mathsf{InvMul}(\mathsf{0})} \equiv \mathsf{InvMul}(\mathsf{0}, \bot, \bot) \rightsquigarrow_{\mathbb{T}(\mathcal{I}nv(R_3))} \{(\mathsf{0}, z)\} \rightsquigarrow_{\mathbb{T}(\mathcal{I}nv(R_3))} \{\{(\mathsf{0}, z')\}\} \;{}_{\{z \mapsto \{z'\}\}}\!\rightsquigarrow_{\mathbb{T}(\mathcal{I}nv(R_3))} \cdots$ because $(x, \{y\}) \to \{(x, y)\} \in \mathbb{T}(\mathcal{I}nv(R_3))$. Note that $\mathbb{U}(\mathcal{I}nv(R_3))$ is terminating on $\mathsf{InvMul}(\mathsf{s}^n(\mathsf{0}))$ with respect to narrowing. Therefore, it can be said that $\mathbb{T}$ always generates non-terminating inverse-TRSs for TRSs with erasing-rules.

In conclusion, comparing with the unraveling, Serbanuta and Rosu's transformation is more effective for injective TRSs at the second phase of the inversion compiler, incomparable for non-injective and non-erasing TRSs, and less effective for the remaining case. In the last case, their transformation should not be employed at the second phase of the inversion compiler.

The class of injective TRSs satisfying the sufficient condition in this paper is incomparable with that of injective TRSs for which Dershowitz and Mitra's Inversion Algorithm [1] terminates. There is a TRS whose inverse TRS is convergent, and for which termination of the algorithm is not guaranteed. As a related work, Kawabe and Glück proposed a transformation based on LR-parsing [2], in order to generate convergent inverses of injective functions in a functional languages. Comparison of our method with theirs is one of future works.

References

1. Dershowitz, N., Mitra, S.: Jeopardy. In: Proceedings of RTA'99. Volume 1631 of LNCS, Springer (1999) 16–29
2. Kawabe, M., Glück, R.: The program inverter lrinv and its structure. In: Proceedings of PADL'05. Volume 3350 of LNCS, Springer (2005) 219–234
3. Lucas, S., Marché, C., Meseguer, J.: Operational termination of conditional term rewriting systems. Information Processing Letters **95**(4) (2005) 446–453
4. Nishida, N., Sakai, M., Sakabe, T.: Partial inversion of constructor term rewriting systems. In: Proceedings of RTA'05. Volume 3467 of LNCS, Springer (2005) 264–278
5. Nishida, N., Sakai, M., Sakabe, T.: Generation of inverse computation programs of constructor term rewriting systems. The IEICE Transactions on Information and Systems **J88-D-I**(8) (2005) 1171–1183 (in Japanese)
6. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer-Verlag (2002)
7. Serbanuta, T. F., Rosu, G.: Computationally equivalent elimination of conditions. In: Proceedings of RTA'06. Volume 4098 of LNCS, Springer (2006) 19–34
8. Romanenko, A.: Inversion and metacomputation. In: Proceedings of PEPM'91. Volume 26 of SIGPLAN Notices, ACM Press (1991) 12–22

A Proof of Proposition 3

Suppose that R has an erasing rule $F(u_1,\ldots,u_n) \to r$. Let x be an erased variable in the rule such that $x \in \mathcal{V}ar(u_i)\backslash\mathcal{V}ar(r)$. Let σ be a normalized substitution such that $x \notin \mathcal{D}om(\sigma)$, s and t be different terms ($s \not\equiv t$), $\sigma_s = \sigma \cup \{x \mapsto s\}$, and $\sigma_t = \sigma \cup \{x \mapsto t\}$. Then we have $F(u_1\sigma_s,\ldots,u_i\sigma_s,\ldots,u_n\sigma) \to_R r\sigma_s \equiv r\sigma$ and $F(u_1\sigma_t,\ldots,u_i\sigma_t,\ldots,u_n\sigma) \to_R r\sigma_t \equiv r\sigma$, and hence $F(u_1\sigma_s,\ldots,u_i\sigma_s,\ldots,u_n\sigma)$ $\downarrow_R$ $F(u_1\sigma_t,\ldots,u_i\sigma_t,\ldots,u_n\sigma)$. It follows form $s \not\equiv t$ that $u_i\sigma_s \not\equiv u_i\sigma_t$. This contradicts injectivity of R. □

B Proof of Theorem 4 (a)

We show *quasi-simplifyingness* of $\mathcal{I}nv(R)$. Then, operational termination of $\mathcal{I}nv(R)$ follows from quasi-simplifyingness.

Definition 8 (quasi-simplifying [6]). *A deterministic 3-CTRS S over a signature $\mathcal{F}$ is called* quasi-simplifying *if there is an extension $\mathcal{F}'$ of the signature $\mathcal{F}$ (so $\mathcal{F} \subseteq \mathcal{F}'$) and a simplification ordering $\succ$ on $\mathcal{T}(\mathcal{F}',\mathcal{V})$ that satisfies the following conditions for every rule $l \to r \Leftarrow s_1 \to t_1,\ldots,s_k \to t_k \in S$, every substitution $\sigma\colon \mathcal{V} \to \mathcal{T}(\mathcal{F}',\mathcal{V})$, and every $0 \leq i < k$:*

1. *if $s_j\sigma \succeq t_j\sigma$ for every $1 \leq j \leq i$, then $l\sigma \succ s_{i+1}\sigma$,*
2. *if $s_j\sigma \succeq t_j\sigma$ for every $1 \leq j \leq k$, then $l\sigma \succ r\sigma$.*

Lemma 9 ([6, 3]). *Quasi-simplifyingness implies operational termination.*

Proposition 10 ([5]). *Let R be a constructor TRS. Then, every rewrite rule $F(u_1,\ldots,u_n) \to r$ in R is transformed by the inversion $\mathcal{I}nv$ into a deterministic conditional rule*

$$InvF(r') \to (u_1,\ldots,u_n) \Leftarrow \textstyle\bigwedge_{i=1}^{k} InvF_i(y_i) \to (w_{i,1},\ldots,w_{i,m_i})$$

where

- *each F_i is a defined of R,*
- *F_i and F_j ($i \neq j$) appear at different positions of r,*
- *r' is a constructor term of R,*
- *each m_i is the arity of F_i,*
- *each variable y_i is not in r,*
- *y_i and y_j ($i \neq j$) are different, and*
- *each y_i appears exactly once in either r' or $w_{j,j'}$ ($j < i$) and not in $w_{l,l'}$ and $u_{n'}$ ($i \leq l$).*

Moreover, the conditional rule has the following properties:

(a) if the original rule is non-erasing, then each variable in $\mathcal{V}ar(u_1,\ldots,u_n)$ occurs in either r' or $u_{i,j}$,
(b) if r is a constructor term of R, then $k = 0$ and $r' \equiv r$, and

(c) if the root symbol G of r is a defined symbol of R, then $r' \equiv y_1$ and $G = F_1$.

Lemma 11. *Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \setminus \mathcal{V}$, $t \in \mathcal{T}(\mathcal{G}, V)$ for signatures $\mathcal{F}$ and $\mathcal{G}$ $(\subseteq \mathcal{F})$, σ be a substitution, $>_{\mathsf{lpo}}$ be the lexicographic path ordering determined by a precedence $>$ on $\mathcal{F}$. If $\mathrm{root}(s) > G$ for all $G \in \mathcal{G}$ and $s\sigma >_{\mathsf{lpo}} x\sigma$ for all $x \in \mathcal{V}ar(t)$, then $s\sigma >_{\mathsf{lpo}} t\sigma$.*

Proof. We prove this by induction on structure of t.

- Case of $t \equiv x \in \mathcal{V}$. It follows from the assumption that $s\sigma >_{\mathsf{lpo}} x\sigma \equiv t\sigma$.
- Let $t \equiv G(t_1, \ldots, t_m)$ where $G \in \mathcal{G}$. By the induction hypothesis, we have $s\sigma >_{\mathsf{lpo}} t_i\sigma$. Now we have $\mathrm{root}(s) > G$ and $s\sigma >_{\mathsf{lpo}} t_i\sigma$. Thus, it follows from the definition of the LPO that $s\sigma >_{\mathsf{lpo}} G(t_1, \ldots, t_m)\sigma \equiv t\sigma$. □

Lemma 12. *Let R be a non-erasing constructor TRS that satisfies the assumption in Theorem 4. Then, $\mathcal{I}nv(R)$ is quasi-simplifying.*

Proof. Let $\mathcal{D}_R$ be the set of defined symbols of R, $inv\mathcal{F}$ be the set of defined symbols of $\mathcal{I}nv(R)$ such that $\{InvF \mid F \in \mathcal{D}_R\}$. We suppose that tuples symbols that are abbreviated to ( ) are in $\mathcal{F}$.

Let $>_{\mathsf{lpo}}$ be the lexicographic path ordering determined by the precedence $>$ that satisfies all of the following:

- $InvF > G$ for all $InvF \in inv\mathcal{F}$ and $G \in \mathcal{F}$, and
- if $F \in \mathcal{D}_R$ calls $G \in \mathcal{D}_R$ and G does not depend on F, then $InvF > InvG$. Otherwise, $InvF = InvG$.

It is clear that the special rules $InvF(F(x_1, \ldots, x_n)) \to (x_1, \ldots, x_n) \in \mathcal{I}nv(R)$ satisfy $InvF(F(x_1, \ldots, x_n)) >_{\mathsf{lpo}} (x_1, \ldots, x_n)$. We only show the rule obtained from $F(u_1, \ldots, u_n) \to r \in R$ satisfies the conditions of quasi-simplifyingness.

Let the conditional rule obtained from $F(u_1, \ldots, u_n) \to r \in R$ be $InvF(r') \to (u_1, \ldots, u_n) \Leftarrow \bigwedge_{i=1}^{k} InvF_i(y_i) \to (w_{i,1}, \ldots, w_{i,m_i})$. Consider the case that $r \in \mathcal{V}$. Let $r \equiv x$. It follows from Proposition 10 that $k = 0$, $r' \equiv x$ and $\mathcal{V}ar(u_1, \ldots, u_n) = \{x\}$, and hence $InvF(x) >_{\mathsf{lpo}} (u_1, \ldots, u_n)$. Therefore, the conditional rule satisfies the conditions of quasi-simplifyingness.

Consider the remaining case that $r \notin \mathcal{V}$. We first prove the following claim for every i $(1 \leq i \leq k)$ by induction on i:

> if $InvF_j(y_j)\sigma \succeq_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma$ for $1 \leq j < i$, then $InvF(r')\sigma >_{\mathsf{lpo}} InvF_i(y_i)\sigma$.

- *Base case* $(i = 1)$.
  - Case that $\mathrm{root}(r)$ is a constructor of R. It follows from Proposition 10 that $\mathrm{root}(r')$ is a constructor of R and $y_1 \in \mathcal{V}ar(r')$, and hence $r' \rhd y_1$. By the assumption on $>$, we have $InvF \geq InvF_1$. It follows from the definition of LPO that $InvF(r') >_{\mathsf{lpo}} InvF_1(y_1)$. Since $>_{\mathsf{lpo}}$ is closed under substitutions, $InvF(r')\sigma >_{\mathsf{lpo}} InvF_1(y_1)\sigma$.

  - The remaining case that $\mathrm{root}(r)$ is a defined symbol of R. It follows from Proposition 10 that $\mathrm{root}(r) = F_1$ and $r' \equiv y_1$. By assumption, F_1 does not depend on F. Thus, it follows from the construction of $>$ that $\mathit{InvF} > \mathit{InvF}_1$, and hence $\mathit{InvF}(y_1) >_{\mathsf{lpo}} \mathit{InvF}_1(y_1)$. Therefore, we have $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} \mathit{InvF}_1(y_1)\sigma$.

– *Induction case* ($i > 1$). Suppose that $\mathit{InvF}_j(y_j)\sigma \geq_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma$ for $1 \leq j < i$. It follows from Proposition 10 that there exist some j ($< i$) and j' ($1 \leq j' \leq m_j$) such that $y_i \in \mathcal{V}ar(w_{j,j'})$. By the induction hypothesis, we have $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} \mathit{InvF}_j(y_j)\sigma$. It is clear that $\mathit{InvF}_j(y_j)\sigma >_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma >_{\mathsf{lpo}} y_i\sigma$. It follows from the construction of $>$ that $\mathit{InvF} \geq \mathit{InvF}_i$. Therefore, it follows from Lemma 11 that $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} \mathit{InvF}_i(y_i)\sigma$.

Therefore, it follows from the above claim that the conditional rule satisfies the first condition of quasi-simplifyingness.

Next we show that the conditional rule satisfies the second condition of quasi-simplifyingness. Suppose that $\mathit{InvF}_j(y_j)\sigma \geq_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma$ for $1 \leq j \leq k$. Then we have $\mathit{InvF}_j(y_j)\sigma >_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma$ because $\mathit{InvF}_j > (\ )$. It follows from non-erasingness of R and Proposition 10 that $\mathcal{V}ar(u_1, \ldots, u_n) \subseteq \mathcal{V}ar(r', w_{1,1}, \ldots, w_{k,m_k})$. Let $x \in \mathcal{V}ar(u_1, \ldots, u_n)$.

– *Case of* $x \in \mathcal{V}ar(r')$. It is clear that $\mathit{InvF}(r') \rhd x$, and hence $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} x\sigma$.
– *The remaining case.* There exist some j ($1 \leq j \leq k$) and j' ($1 \leq j' \leq m_j$) such that $x \in \mathcal{V}ar(w_{j,j'})$, and hence $(w_{j,1}, \ldots, w_{j,m_j}) \rhd x$. It follows from $\mathit{InvF}_j(y_j)\sigma >_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma$ and the first condition of quasi-simplifyingness that $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} \mathit{InvF}_j(y_j)\sigma$, and hence $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} (w_{j,1}, \ldots, w_{j,m_j})\sigma >_{\mathsf{lpo}} x\sigma$.

Thus we have $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} x\sigma$. It follows from Lemma 11 that $\mathit{InvF}(r')\sigma >_{\mathsf{lpo}} (u_1, \ldots, u_n)\sigma$. Therefore, the conditional rule satisfies the second condition of quasi-simplifyingness. $\square$

Main Proof

Theorem 4 (a) follows from Lemmas 9 and 12. $\square$

C Proof of Theorem 4 (b)

We first give some notions associated with CTRSs.

Definition 13 ([6]). *Let S be a deterministic 3-CTRS.*

– *A term t is called* strongly irreducible with respect to S *if $t\sigma$ is a normal form for every normalized substitution σ.*
– *S is called* strongly deterministic *if for every rule $l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k$ in S, every term t_i is strongly irreducible.*

– *Suppose that $\langle s, t\rangle$ is a critical pair obtained from $l_1 \to r_1$, $l_2 \to r_2$ and the most general unifier σ of $l_1|_p$ and l_2 where $l_1|_p \notin \mathcal{V}$. Then, $\langle s, t\rangle \Leftarrow c_1\sigma \wedge c_2\sigma$ is a* conditional critical pair *of deterministic conditional rules $l_1 \to r_1 \Leftarrow c_1$ and $l_2 \to r_2 \Leftarrow c_2$ of S.*
– *A conditional critical pair $\langle s, t\rangle \Leftarrow \bigwedge_{i=1}^{k} s_i \to t_i$ of S is called* joinable *if $s\theta$ and $t\theta$ are joinable for all substitutions θ such that $s_i\sigma \xrightarrow{*}_S t_i\theta$ for every i.*

Theorem 14 ([6]). *Every quasi-simplifying strongly deterministic 3-CTRS with joinable conditional critical pairs is confluent.*

Proposition 15. *If R is non-erasing, then $\mathcal{I}nv(R)$ is strongly deterministic.*

Proof. Let $l \to r \Leftarrow \bigwedge_{i=1}^{k} s_i \to t_i \in \mathcal{I}nv(R)$ where $i > 0$. It follows from Proposition 10 that each t_i is a constructor term of R, and hence t_i is a constructor term of $\mathcal{I}nv(R)$. Therefore, $t_i\sigma$ is a normal form of $\mathcal{I}nv(R)$ for all normalized substitutions σ. □

Since $\mathcal{I}nv(R)$ is strongly deterministic and quasi-simplifying, it is enough to show joinability of each critical pair of $\mathcal{I}nv(R)$.

Main Proof

Since $\mathcal{I}nv(R)$ is a constructor system, every two overlapped rules overlap only at the root positions. Consider two rules $InvF(s_1) \to (t_1, \ldots, t_n) \Leftarrow Cond_1$ and $InvF(s_2) \to (u_1, \ldots, u_n) \Leftarrow Cond_2$ such that $\mathcal{V}ar(s_1, t_1, \ldots, t_n, Cond_1) \cap \mathcal{V}ar(s_2, u_1, \ldots, u_n, Cond_2) = \emptyset$ and s_1 and s_2 are unifiable. Let σ be a most general unifier of s_1 and s_2. Then, their critical pairs are $\langle (t_1, \ldots, t_n)\sigma, (u_1, \ldots, u_n)\sigma\rangle \Leftarrow Cond_1\sigma \wedge Cond_2\sigma$ and $\langle (u_1, \ldots, u_n)\sigma, (t_1, \ldots, t_n)\sigma\rangle \Leftarrow Cond_1\sigma \wedge Cond_2\sigma$.

Suppose that $\langle (t_1, \ldots, t_n)\sigma, (u_1, \ldots, u_n)\sigma\rangle \Leftarrow Cond_1\sigma \wedge Cond_2\sigma$ is not joinable. Then there exists a substitution δ and some j such that $t_j\sigma\delta$ and $u_j\sigma\delta$ are not joinable, and $Cond_1\sigma\delta$ and $Cond_2\sigma\delta$ are true, that is, $InvF(s_1\sigma\delta) \to_{\mathcal{I}nv(R)} (t_1, \ldots, t_n)\sigma\delta$ and $InvF(s_2\sigma\delta) \to_{\mathcal{I}nv(R)} (u_1, \ldots, u_n)\sigma\delta$. It follows from Theorem 1 that $F(t_1, \ldots, t_n)\sigma\delta \xrightarrow{*}_R s_1\sigma\delta$ and $F(u_1, \ldots, u_n)\sigma\delta \xrightarrow{*}_R s_2\sigma\delta \equiv s_1\sigma\delta$. We can suppose without loss of generality that δ is a normalized substitution of both R and $\mathcal{I}nv(R)$. Then, $t_j\sigma\delta$ and $u_j\sigma\delta$ are normal forms of both R and $\mathcal{I}nv(R)$ because t_j and u_j are constructor terms of R. Since $t_j\sigma\delta$ and $u_j\sigma\delta$ are not joinable, we have $t_j\sigma\delta \not\equiv u_j\sigma\delta$. This contradicts injectivity of F. Thus $\langle (t_1, \ldots, t_n)\sigma, (u_1, \ldots, u_n)\sigma\rangle \Leftarrow Cond_1\sigma \wedge Cond_2\sigma$ is joinable.

From similar discussion, $\langle (u_1, \ldots, u_n)\sigma, (t_1, \ldots, t_n)\sigma\rangle \Leftarrow Cond_1\sigma \wedge Cond_2\sigma$ is joinable. Therefore, the critical pairs of the above two conditional rules are joinable, and hence $\mathcal{I}nv(R)$ is confluent. □