An Isabelle/HOL Formalization of Semi-Thue and Conditional Semi-Thue Systems

Dohan Kim
Department of Computer Science, University of Innsbruck, Austria

Abstract

We present a formalized framework for semi-Thue and conditional semi-Thue systems for studying monoids and their word problem using the Isabelle/HOL proof assistant. We provide a formalized decision procedure for the word problem of monoids if they are finitely presented by complete semi-Thue systems. In particular, we present a new formalized method for checking confluence using (conditional) critical pairs for certain conditional semi-Thue systems. We propose and formalize an inference system for generating conditional equational theories and Thue congruences using conditional semi-Thue systems. Then we provide a new formalized decision procedure for the word problem of monoids which have finite complete (reductive) conditional presentations.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Equational logic and rewriting

Keywords and phrases semi-Thue systems, conditional semi-Thue systems, conditional string rewriting, monoids, word problem

Digital Object Identifier 10.4230/LIPIcs.ITP.2025.10

Supplementary Material *Software (Source Code)*:
`http://cl-informatik.uibk.ac.at/experiments/ITP2025/material.zip`

Funding This research was funded by the Austrian Science Fund (FWF) project I 5943-N.

Acknowledgements The author would like to thank René Thiemann for his valuable comments on this paper. The author also would like to thank the anonymous reviewers for their valuable comments to improve the presentation of this paper.

1 Introduction

It is generally accepted that semi-Thue systems, also known as string rewriting systems, were first introduced by Axel Thue in 1910's in order to solve the word problem of semigroups and monoids [39]. They can also be viewed as presentations of monoids, where monoids are fundamental algebraic structures widely used in mathematics and computer science. In particular, they serve as a well-known framework for solving the word problem for monoids and groups. The word problem of monoids (or groups) is undecidable in general but it is decidable if they are presented by finite complete (i.e., terminating and confluent) semi-Thue systems over finite alphabets. If a finite semi-Thue system is terminating, then the critical pair lemma provides a decision procedure for confluence. If a monoid is presented by a finite terminating semi-Thue system S but it is not confluent, then one may attempt to construct a finite complete semi-Thue system S' equivalent to S using a completion procedure.

Semi-Thue systems can also be considered as a special case of term rewriting systems (TRSs), where term rewriting systems play an important role in programming languages (e.g., functional programming) [26], software engineering (e.g., equationally specified abstract data types) [37], computer algebra [29], and automated theorem proving [20]. For example, a semi-Thue system S on Σ^* can be associated with a term rewriting system (TRS) $\mathcal{R}_S$ in

such a way that $\mathcal{R}_S := \{\ell(x) \to r(x) \mid \ell \to r \in S\}$ [7], where each letter from an alphabet Σ is interpreted as a unary function symbol. Here, variables in $\mathcal{R}_S$ are renamed whenever necessary.

The formalization[1] of TRSs has been done extensively via Isabelle/HOL (e.g., IsaFoR [44]) and Coq (e.g., [8, 9]), and word problems of algebraic structures using TRSs were also discussed in early systems, such as the RRL [25] and the REVE system [31]. However, they are not directly suited for monoid and group presentations.

Our Isabelle/HOL formalization of semi-Thue systems and their related results is based on the simple string matching methods instead of using the more complex term structures. It also uses the simple shortlex order (i.e. length-lexicographic order) [7] on strings, which is simpler than the usual term orderings, such as the lexicographic path order (LPO) [22], the Knuth–Bendix order (KBO) [29] and the weighted path order (WPO) [45]. Some results of semi-Thue systems were also formalized, for example, in Lean [15] and Coq [17].

Meanwhile, conditional semi-Thue systems are extensions of semi-Thue systems, where each of their rules has the form $\ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n$ for strings $\ell, r, s_1, t_1, \ldots, s_n, t_n$. (Here, $\ell \to r$ can also be denoted as an ordered pair (ℓ, r).) A finitely generated monoid with decidable word problem may not admit a finite complete (unconditional) presentation, but it may admit a finite complete conditional presentation [11]. For example, the monoid presented by $\mathcal{R} = \{aba \to ba\}$ on $\Sigma = \{a, b\}$ does not admit a finite complete (unconditional) presentation by a semi-Thue system but admits a finite complete conditional presentation by a conditional semi-Thue system.

The equality symbol $\approx$ in conditional string rewriting rules of conditional semi-Thue systems can be interpreted in different ways, yielding different types of conditional semi-Thue systems. Note that some conditional semi-Thue systems, such as right conditional semi-Thue systems, can be associated with conditional term rewriting systems (CTRSs) [23], but this is not the case for all types of conditional semi-Thue systems, for example, left-right conditional semi-Thue systems. We classify and formalize different types of conditional semi-Thue systems, which extend the classification of conditional semi-Thue systems discussed in [11].

We also formalize conditional equational theories and Thue congruences using conditional semi-Thue systems for the associated monoids and their word problem. In particular, we provide a formalized method for checking confluence using (conditional) critical pairs for certain types of conditional semi-Thue systems, which provides a decision procedure for confluence if they are reductive w.r.t. the shortlex order.

Our Isabelle/HOL formalization of semi-Thue and conditional semi-Thue systems along with their related results is built on IsaFoR (Isabelle/HOL *Formalization of Rewriting*) [44], where Isabelle/HOL [36] is a generic proof assistant and a theorem prover allowing the users to express concepts in mathematics and computer science and to prove properties about them. It also relies on the "A Case Study in Basic Algebra" AFP (archive of formal proofs) entry [4] for a formalization of monoids and groups, and the "Abstract Rewriting" AFP entry [42] for a formalization of the abstract rewriting systems (ARSs) [1] and their related properties, such as the Church-Rosser (CR) and the strong normalization (SN) property.

To the best of our knowledge, conditional semi-Thue systems and their related results have not been formalized in any proof assistant. Our formalization may provide a formalized framework for string rewriting and conditional string rewriting along with their wide variety of applications. The relevant Isabelle/HOL theory files[2] inside IsaFoR under the directory thys/Conditional_Semi_Thue_Systems are as follows:

[1] In this paper, by "formalization", we mean a computer-assisted formalization using a proof assistant.
[2] `http://cl-informatik.uibk.ac.at/experiments/ITP2025/material.zip`.

Semi_Thue_Systems.thy	Conditional_Semi_Thue_Systems.thy
String_Rewriting	Conditional_String_Rewriting.thy
ShortLex.thy	STS_Completion.thy
Conditional_Equational_Theories.thy	STS_Critical_Pairs.thy
CSTS_P_Critical_Pairs.thy	CSTS_R_Critical_Pairs.thy.thy

In the remainder of this paper, we use hyperlinks marked by ☑ for providing an HTML rendering for our formalized proofs in Isabelle/HOL.

2 Preliminaries

The definitions and results in this section can be found, for example, in [7, 13, 14, 19, 40].

Let Σ be a finite alphabet, $>$ be a strict order on Σ (i.e. a precedence on Σ), and Σ^* be the set of all finite (possibly empty) strings from Σ.

The *shortlex order* (or *length-lexicographic order*) $\succ_{sl}$ on Σ^* induced by $>$ is defined as follows: $s = s_1 \cdots s_i \succ_{sl} t_1 \cdots t_j = t$ if $i > j$, or they have the same length (i.e., $i = j$) and $s_1 \cdots s_i$ comes before $t_1 \cdots t_i$ lexicographically using a precedence $>$ on Σ. Here, $s_1 \cdots s_i$ comes before $t_1 \cdots t_i$ lexicographically using a precedence $>$ on Σ if there is a k such that $s_1 \cdots s_{k-1} = t_1 \cdots t_{k-1}$ and $s_k > t_k$. If $\ell \succ_{sl} r$, then $u\ell v \succ_{sl} urv$ for all $u, v \in \Sigma^*$.

A *monoid* is a set equipped with an associative binary operation and a (two-sided) identity element. Note that a monoid cannot be empty because of an identity element. For example, Σ^* is a (free) monoid generated by Σ under the operation of concatenation as an associative binary operation with the empty string ε as an identity element. A *homomorphism* between two monoids $(M, \cdot)$ and $(M', \cdot')$ is a function $\phi : M \to M'$ such that $\phi(x \cdot y) = \phi(x) \cdot' \phi(y)$ for all $x, y \in M$ and $\phi(1) = 1'$, where 1 and $1'$ are the identities of M and M', respectively. A bijective monoid homomorphism is called a *monoid isomorphism*. Two monoids are said to be *isomorphic* if there is a monoid isomorphism between them. An element a of a monoid $(M, \cdot)$ is *invertible* if there is an element $b \in M$ such that $a \cdot b = b \cdot a = 1$, where 1 is an identity element of M. A *group* is a monoid in which every element is invertible.

The reflexive and transitive closure of a binary relation $\to$ is denoted by $\xrightarrow{*}$, and the reflexive, symmetric and transitive closure of $\to$ is denoted by $\xleftrightarrow{*}$ called *conversion*.

A *derivation* for a binary relation $\to$ is a sequence of form $t_0 \to t_1 \to t_2 \cdots$. A binary relation $\to$ is *terminating* (or *strongly normalizing*) if there are no infinite derivations $t_0 \to t_1 \to t_2 \cdots$. An element s is *reducible* (w.r.t. a given binary relation $\to$) if there is an element u such that $s \to u$; otherwise, s is *irreducible*. We say that u is an *$\to$-normal form* of s if $s \xrightarrow{*} u$ and u is irreducible via $\to$. A binary relation $\to$ is *confluent* if there is an element w such that $u \xrightarrow{*} w$ and $v \xrightarrow{*} w$ whenever $s \xrightarrow{*} u$ and $s \xrightarrow{*} v$. A binary relation $\to$ is *locally confluent* if there is an element w such that $u \xrightarrow{*} w$ and $v \xrightarrow{*} w$ whenever $s \to u$ and $s \to v$. A binary relation $\to$ is *Church-Rosser* if there is an element w such that $u \xrightarrow{*} w$ and $v \xrightarrow{*} w$ whenever $u \xleftrightarrow{*} v$. Here, $\to$ is Church-Rosser iff it is confluent. Two elements u and v are *joinable* w.r.t. a binary relation $\to$, denoted by $u \downarrow v$, if there is an element w such that $u \xrightarrow{*} w \xleftarrow{*} v$ (i.e., $u \xrightarrow{*} w$ and $v \xrightarrow{*} w$). A binary relation $\to$ is *complete* if it is both terminating and confluent. If $\to$ is terminating, then it is confluent iff it is locally confluent (see Newman's lemma [1, 35]).

3 Semi-Thue Systems

A *semi-Thue system* (STS) $\mathcal{R}$ [7] over a finite alphabet Σ is a subset of $\Sigma^* \times \Sigma^*$. If $\mathcal{R}$ is an STS over a finite alphabet Σ, then the *single-step string rewriting relation* on Σ^* induced by $\mathcal{R}$ is defined as follows: for any $u, v \in \Sigma^*$, $u \to_{\mathcal{R}} v$ iff there exists $(\ell, r) \in \mathcal{R}$ such that for some $x, y \in \Sigma^*$, $u = x\ell y$ and $v = xry$. Also, $\to_{\mathcal{R}}^*$ denotes the reflexive, transitive closure of $\to_{\mathcal{R}}$. By extension, the terms "terminating", "confluent", etc., used in Section 2 are also applied to an STS $\mathcal{R}$ whose single-step string rewriting relation $\to_{\mathcal{R}}$ has those properties. In the remainder of this paper, unless otherwise stated, we denote by Σ a finite alphabet and assume that a total precedence $>$ on Σ is always given so that the shortlext order $\succ_{sl}$ is a well-founded total order on Σ^*.

An STS $\mathcal{R}$ with the property that $(\ell, r) \in \mathcal{R}$ implies $(r, \ell) \in \mathcal{R}$ is called a *Thue system*. The *Thue congruence* induced by $\mathcal{R}$ is the relation $\overset{*}{\leftrightarrow}_{\mathcal{R}}$. Two strings $u, v \in \Sigma^*$ are *congruent* (modulo $\mathcal{R}$) if $u \overset{*}{\leftrightarrow}_{\mathcal{R}} v$. The congruence class $[w]_{\mathcal{R}}$ of a word $w \in \Sigma^*$ is defined as $[w]_{\mathcal{R}} := \{v \in \Sigma^* \mid w \overset{*}{\leftrightarrow}_{\mathcal{R}} v\}$. The set $\{[w]_{\mathcal{R}} \mid w \in \Sigma^*\}$ of congruence classes (modulo $\mathcal{R}$) is denoted by $\mathcal{M}_{\mathcal{R}}$. In this section, we refer to [7, 18, 21] for the definitions and results related to STSs, monoids, and completion.

▶ **Lemma 1.** *The set $\mathcal{M}_{\mathcal{R}}$ is a monoid under the operation $[u]_{\mathcal{R}} \cdot [v]_{\mathcal{R}} := [uv]_{\mathcal{R}}$ with identity $[\varepsilon]_{\mathcal{R}}$.* ☑

In fact, $\mathcal{M}_{\mathcal{R}}$ is the *factor monoid* of the free monoid Σ^* modulo the congruence $\overset{*}{\leftrightarrow}_{\mathcal{R}}$, i.e., $\mathcal{M}_{\mathcal{R}} = \Sigma^* / \overset{*}{\leftrightarrow}_{\mathcal{R}}$.

▶ **Definition 2.**

(i) Let $\mathcal{R}$ be an STS over Σ. If a monoid $\mathcal{M}$ is isomorphic to the factor monoid $\mathcal{M}_{\mathcal{R}}$ ($\mathcal{M} \cong \mathcal{M}_{\mathcal{R}}$), then the ordered pair $(\Sigma; \mathcal{R})$ is a *presentation* of $\mathcal{M}$, and simply write $\mathcal{M} = (\Sigma; \mathcal{R})$. If $\to_{\mathcal{R}}$ is complete, then $(\Sigma; \mathcal{R})$ is a *complete presentation* of $\mathcal{M}$.

(ii) The monoid $\mathcal{M} = (\Sigma; \mathcal{R})$ is *finitely presented* if both Σ and $\mathcal{R}$ are finite.

(iii) The *word problem* of the monoid $\mathcal{M} = (\Sigma; \mathcal{R})$ is the following decision problem: Given two words $u, v \in \Sigma^*$, decide if $u = v$ in $\mathcal{M}$.

Our formalization of STSs and their associated monoids is based on the following Isabelle's locale, where a locale [3] is a named context for fixed parameters with assumptions.

```
locale semi_Thue =
 fixes R :: sts  and T :: sts  and S :: "char set"
 assumes  "T = (ststep R)^↔* "
   and    "finite S"
   and    "S ≠ {}"
   and    "finite R"
   and    "(ststep R)* ⊆ S* × S* "
begin
 ...
```

Above, R denotes a finite STS over a finite non-empty alphabet S, while T denotes the Thue congruence induced by R. The sts type for R and T is declared as follows:

```
type_synonym sts = "srule set"
```

where the type srule is declared simply as string×string. (Alternatively, instead of using the Isabelle's dedicated string type (i.e., char list type), one can use the 'a list type, which is suitable for more general STSs.) Also, $(ststep\ R)^* \subseteq S^\star \times S^\star$ represents the assumption

$\to_{\mathcal{R}}^* \subseteq \Sigma^* \times \Sigma^*$, which also implies $\to_{\mathcal{R}} \subseteq \Sigma^* \times \Sigma^*$. Our formalization of Lemma 1 also uses the locale monoid in the AFP entry [4]. It is instantiated as a sublocale using the following parameters, where $S^\star/T$ represents the factor monoid $\mathcal{M}_{\mathcal{R}}$ with identity $[\varepsilon]_{\mathcal{R}}$.

sublocale *monoid "$S^\star/T$" "$([\cdot])$" "eq.Class ε"*

We also use the locale group in the same AFP entry, which extends the locale monoid by adding the assumption that every element of a monoid is invertible. This can be further refined as in the following lemma.

▶ **Lemma 3.** *The monoid $\mathcal{M}_{\mathcal{R}}$ is a group if for all $u \in \Sigma$, there is some $v \in \Sigma^*$ such that $uv \overset{*}{\leftrightarrow}_{\mathcal{R}} \varepsilon$ and $vu \overset{*}{\leftrightarrow}_{\mathcal{R}} \varepsilon$.* ☑

If an STS $\mathcal{R}$ is complete, then each congruence class contains a unique irreducible word only, which can be chosen as a normal form for this congruence class. Therefore, if $\mathcal{R}$ is finite and complete, then the word problem of the monoid $M = (\Sigma; \mathcal{R})$ can be solved using string rewriting, that is, given two strings $u, v \in \Sigma^*$, reduce u and v by $\mathcal{R}$ to their respective $\mathcal{R}$-normal forms $\bar{u}$ and $\bar{v}$ and simply compare them. Now, if one can show that $(\Sigma; \mathcal{R})$ is a finite complete presentation, then one can decide the word problem of the monoid $(\Sigma; \mathcal{R})$. First, in order to show that $\mathcal{R}$ is terminating, there is an easy sufficient condition for it.

▶ **Lemma 4.** *If $\ell \succ_{sl} r$ for each $(\ell, r) \in \mathcal{R}$, then $\to_{\mathcal{R}}$ is terminating.* ☑

▶ **Lemma 5.** *If $\to_{\mathcal{R}}$ is terminating, then $\to_{\mathcal{R}}$ is confluent iff $\to_{\mathcal{R}}$ is locally confluent.* ☑

In Lemma 4, the shortlex order $\succ_{sl}$ is formalized as follows:

```
fun shortlex:: "string ⇒ string ⇒ bool" where
  "shortlex str1 str2 = (if lenorder str1 str2 then True else if lenorder str2 str1 then False
    else lexorder str1 str2)
```

where the lenorder function is used for comparing the lengths of two strings and the lexorder function is used for comparing two strings of the same length via the lexicographic order (see below).

```
fun lenorder:: "string ⇒ string ⇒ bool" where
  "lenorder s t = (length s > length t)"

fun lexorder:: "string ⇒ string ⇒ bool" where
  "lexorder s t = (∃u1 u2 u3 ch1 ch2 . s = u1 @ [ch1] @ u2 ∧ t = u1 @ [ch2] @ u3 ∧ ch1 ≻p ch2)"
```

Above, "@" denotes the list append function in Isabelle/HOL. Here, the list append function "@" serves as the string concatenation function. Accordingly, both $ch1$ and $ch2$ are characters, where $ch1 \succ_p ch2$ denotes that $ch1$ has a higher precedence than $ch2$. It is also the case that s is greater than t w.r.t. the lexicographic order if t is a strict prefix of s (see [14]). We omit this case because this case is not necessary for the shortlex order. The shortlex order using the shortlex function is abbreviated as $\succ_{sl}$.

```
abbreviation shortlex_s (infix "≻sl" 50) where "s ≻sl t ≡ shortlex s t"
abbreviation shortlex_ns (infix "≽sl" 50) where "s ≽sl t ≡ shortlex s t ∨ s = t"
```

▶ **Definition 6.**

(i) For each pair of not necessarily distinct string rewriting rules from $\mathcal{R}$, say (u_0, v_0) and (u_0', v_0'), the set of *critical pairs*, denoted by $CP(\mathcal{R})$, corresponding to this pair is $\{(v_0 y, x v_0') \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 y = x u_0' \text{ and } |x| < |u_0|\} \cup \{(v_0, x v_0' y) \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 = x u_0' y\}$.

(ii) A critical pair (z_1, z_2) is *joinable* if $z_1 \downarrow_{\mathcal{R}} z_2$.

▶ **Lemma 7.** *If* $t \,{}_{\mathcal{R}}\!\leftarrow s \rightarrow_{\mathcal{R}} u$, *then* $t \downarrow_{\mathcal{R}} u$ *or* $t \leftrightarrow_{CP(\mathcal{R})} u$. ☑

▶ **Lemma 8.** $\rightarrow_{\mathcal{R}}$ *is locally confluent iff all its critical pairs are joinable.* ☑

By using Definition 6(i), the set of critical pairs for an STS $\mathcal{R}$ is formalized as follows:

```
definition sts_critical_pairs where
  "sts_critical_pairs R = {(v @ y, x @ v') | x y u v u' v'. (u, v) ∈ R ∧ (u', v') ∈ R ∧ u @ y = x @ u' ∧
  length x < length u} ∪ {(v, x @ v' @ y) | x y u v u' v'. (u, v) ∈ R ∧ (u', v') ∈ R ∧ u = x @ u' @ y}"

lemma sts_critical_pair_lemma:
  "WCR (ststep R) ⟷ (∀(s, t) ∈ sts_critical_pairs R . (s, t) ∈ (ststep R)↓)"
```

Our formalization of Lemma 8 is as above. Here, WCR [42] is the existing formalized definition of the Weak Church-Rosser property, where $WCR\,(ststep\ R)$ denotes that $\rightarrow_{\mathcal{R}}$ is locally confluent.

▶ **Theorem 9.** *If* $\rightarrow_{\mathcal{R}}$ *is terminating, then* $\rightarrow_{\mathcal{R}}$ *is confluent iff all its critical pairs are joinable.* ☑

The following theorem and its proof provide a decision procedure for the word problem of a monoid if it is presented by a finite complete STS.

▶ **Theorem 10.** *Let* $\mathcal{R}$ *be a finite STS on* Σ^*. *If* $\rightarrow_{\mathcal{R}}$ *is complete, then we can decide whether* s *and* t *in* Σ^* *are the same element in the monoid* $\mathcal{M}_{\mathcal{R}}$. ☑

Proof. Observe that if $s \overset{*}{\leftrightarrow}_{\mathcal{R}} t$, then s and t are the same element in $\mathcal{M}_{\mathcal{R}} := \Sigma^*/\overset{*}{\leftrightarrow}_{\mathcal{R}}$; otherwise, they are the different elements in $\mathcal{M}_{\mathcal{R}}$. Since $\rightarrow_{\mathcal{R}}$ is finite and complete, we can decide whether $s \overset{*}{\leftrightarrow}_{\mathcal{R}} t$ by comparing the normal forms of s and t w.r.t. $\rightarrow_{\mathcal{R}}$, that is, if the normal forms of s and t w.r.t. $\rightarrow_{\mathcal{R}}$ are the same, then they are the same element in $\mathcal{M}_{\mathcal{R}}$ (i.e., $[s]_{\mathcal{R}} = [t]_{\mathcal{R}}$); otherwise, they are the different elements in $\mathcal{M}_{\mathcal{R}}$ (i.e., $[s]_{\mathcal{R}} \neq [t]_{\mathcal{R}}$). ◀

▶ Remark 11. Thus far, this section has been concerned with the word problem of monoids and its decision procedure if they are presented by finite complete STSs. Our formalization can also be used for the word problem for finitely presented groups. Group presentations can be converted into monoid presentations (see [19, 40]). (Recall that groups are also monoids.) Let S be a set of generators of a group G, R be a set of relations of G, and $S_\alpha = \{s^\alpha \mid s \in S, \alpha \in \{1, -1\}\}$. Then the group defined by the presentation $< S \mid R >$ is equivalent to the monoid defined by the presentation $(\Sigma; \mathcal{R})$, where $\Sigma = S_\alpha$ and $\mathcal{R}$ is an STS orienting the equations in $\{ss^{-1} \approx 1 \mid s \in S_\alpha\} \cup R$ using the total shortlex order on S_α^*. (Here, if $u = v^{-1} \in S_\alpha$, then u^{-1} is defined to be equal to v.) Therefore, Theorem 10 can also be used for the word problem of a group if its group presentation yields the monoid presentation $(\Sigma; \mathcal{R})$ such that $\mathcal{R}$ is a finite complete STS on Σ^*.

Completion [1, 18, 29] attempts to transform a set of equations (or string rewriting rules in equational form) on Σ^* into an equivalent complete string rewriting system. It basically converts non-joinable critical pairs into string rewriting rules so that they are joinable. It uses a reduction order [19] (e.g. the shortlex order) to orient equations on Σ^*, and also uses string rewriting for simplification. (Here, an equation is an unordered pair (s, t) of strings on Σ^*, written as $s \approx t$.) The inference rules, which are given in Figure 1, operate on pairs $(\mathcal{E}, \mathcal{R})$, where $\mathcal{E}$ is a set of equations and $\mathcal{R}$ is a set of string rewriting rules on Σ^*. Intuitively speaking, $\mathcal{E}$ is a set of input equations or critical pairs that have not yet been transformed (i.e., oriented) into string rewriting rules, while $\mathcal{R}$ is a terminating set of string rewriting rules.

Deduce: $\dfrac{(\mathcal{E},\mathcal{R})}{(\mathcal{E}\cup\{su_3\approx u_1t\},\mathcal{R})}$ if $(u_1u_2,s)\in\mathcal{R}$, $(u_2u_3,t)\in\mathcal{R}$, and $u_2\neq\varepsilon$.

Simplify: $\dfrac{(\mathcal{E}\cup\{u_1u_2u_3\approx s\},\mathcal{R})}{(\mathcal{E}\cup\{u_1tu_3\approx s\},\mathcal{R})}$ if $(u_2,t)\in\mathcal{R}$.

Orient: $\dfrac{(\mathcal{E}\cup\{s\approx t\},\mathcal{R})}{(\mathcal{E},\mathcal{R}\cup\{(s,t)\})}$ if $s\succ_{sl}t$.

Collapse: $\dfrac{(\mathcal{E},\mathcal{R}\cup\{(u_1u_2u_3,s)\})}{(\mathcal{E}\cup\{u_1tu_3\approx s\},\mathcal{R})}$ if $(u_2,t)\in\mathcal{R}$.

Compose: $\dfrac{(\mathcal{E},\mathcal{R}\cup\{(s,u_1u_2u_3)\})}{(\mathcal{E},\mathcal{R}\cup\{(s,u_1tu_3)\})}$ if $(u_2,t)\in\mathcal{R}$.

Delete: $\dfrac{(\mathcal{E}\cup\{s\approx s\},\mathcal{R})}{(\mathcal{E},\mathcal{R})}$

Above, $\mathcal{E}$ is a set of equations on Σ^* and $\mathcal{R}$ is a set of string rewriting rules on Σ^* such that for each $(\ell,r)\in R$, $\ell\succ_{sl}r$.

Figure 1 Inference rules for completion of a semi-Thue system $\mathcal{R}$ on Σ^*.

▶ **Definition 12.** We write $(\mathcal{E},\mathcal{R})\vdash_{SR}(\mathcal{E}',\mathcal{R}')$ if $(\mathcal{E}',\mathcal{R}')$ can be obtained from $(\mathcal{E},\mathcal{R})$ by applying one of the inference rules in Figure 1.

▶ **Definition 13.**

(i) A (finite) *run* for an initial set of equations $\mathcal{E}$ is a finite sequence $(\mathcal{E}_0,\mathcal{R}_0)\vdash_{SR}(\mathcal{E}_1,\mathcal{R}_1)\vdash_{SR}\cdots\vdash_{SR}(\mathcal{E}_n,\mathcal{R}_n)$, where $\mathcal{E}_0=\mathcal{E}$ and $\mathcal{R}_0=\emptyset$.

(ii) A run $(\mathcal{E}_0,\mathcal{R}_0)\vdash_{SR}(\mathcal{E}_1,\mathcal{R}_1)\vdash_{SR}\cdots\vdash_{SR}(\mathcal{E}_n,\mathcal{R}_n)$ *fails* if $\mathcal{E}_n\neq\emptyset$.

(iii) A run $(\mathcal{E}_0,\mathcal{R}_0)\vdash_{SR}(\mathcal{E}_1,\mathcal{R}_1)\vdash_{SR}\cdots\vdash_{SR}(\mathcal{E}_n,\mathcal{R}_n)$ is *fair* if $CP(\mathcal{R}_n)\subseteq\bigcup_{i=0}^{n}\leftrightarrow_{\mathcal{E}_i}$.

▶ **Lemma 14.** *For every non-failing run* $(\mathcal{E}_0,\mathcal{R}_0)\vdash_{SR}(\mathcal{E}_1,\mathcal{R}_1)\vdash_{SR}\cdots\vdash_{SR}(\mathcal{E}_n,\mathcal{R}_n)$, $\overset{*}{\leftrightarrow}_{\mathcal{E}_0}=\overset{*}{\leftrightarrow}_{\mathcal{R}_n}$. ☑

▶ **Definition 15** (cf. [18])**.** Let $\mathcal{R}$ be an STS on Σ^* and M be a finite multiset of strings on Σ^*. We write (the labeled string rewriting) $s\xrightarrow[\mathcal{R}]{M}t$ if $s\to_{\mathcal{R}}t$ and there are some $s',t'\in M$ such that $s'\succeq_{sl}s$ and $t'\succeq_{sl}t$. Here, $\succeq_{sl}$ denotes the reflexive closure of $\succ_{sl}$.

▶ **Lemma 16.** *Let* $(\mathcal{E},\mathcal{R})\vdash_{SR}(\mathcal{E}',\mathcal{R}')$. *If* $t\xleftrightarrow[\mathcal{E}\cup\mathcal{R}]{M}{}^{*}u$ *and* $\mathcal{R}'\subseteq\succ_{sl}$, *then* $t\xleftrightarrow[\mathcal{E}'\cup\mathcal{R}']{M}{}^{*}u$. ☑

The following definition is due to [18, 43].

▶ **Definition 17.**

(i) An *abstract rewrite system* (ARS) $\mathcal{A}$ is a set A equipped with a binary relation $\to$. We also write an ARS $\mathcal{A}$ as $\langle A,\{\to_\alpha\}_{\alpha\in I}\rangle$ where we denote by $\to_\alpha$ the part of the binary relation $\to$ with label α indexed by I so that $\to=\bigcup\{\to_\alpha|\ \alpha\in I\}$.

(ii) An ARS $\mathcal{A} = \langle A, \{\to_\alpha\}_{\alpha\in I}\rangle$ is *peak decreasing* if there exists a well-founded order $>$ on I such that for all $\alpha, \beta \in I$ the inclusion

$$\alpha\!\leftarrow \cdot \to_\beta \;\subseteq\; \xleftarrow[\vee\alpha\beta]{*}$$

holds. Here, $\vee\alpha\beta$ denotes the set $\{\gamma \in I \mid \alpha > \gamma \text{ or } \beta > \gamma\}$ and if $J \subseteq I$ then $\xleftrightarrow[J]{*}$ denotes a conversion consisting of $\to_J = \bigcup\{\to_\gamma \mid \gamma \in J\}$ steps.

See [18] for the proof of the following lemma and its Isabelle/HOL formalization.

▶ **Lemma 18.** *Every peak decreasing ARS is confluent.*

▶ **Theorem 19** (cf. [18])**.** *For every fair non-failing run* $(\mathcal{E}_0, \mathcal{R}_0) \vdash_{SR} (\mathcal{E}_1, \mathcal{R}_1) \vdash_{SR} \cdots \vdash_{SR} (\mathcal{E}_n, \mathcal{R}_n)$*,* $\to_{\mathcal{R}_n}$ *is a complete STS for an initial finite set of equations* $\mathcal{E} = \mathcal{E}_0$ *on* Σ^**.* ☑

Sketch of the proof. We first show that $\to_{\mathcal{R}_n}$ is terminating. If $(\mathcal{E}, \mathcal{R}) \vdash_{SR} (\mathcal{E}', \mathcal{R}')$ and $\mathcal{R} \subseteq \succ_{sl}$, then $\mathcal{R}' \subseteq \succ_{sl}$ by the case analysis of each inference rule in Figure 1. We leave the straightforward induction proof to our formalization for showing $\mathcal{R}_n \subseteq \succ_{sl}$, which in turn shows that $\mathcal{R}_n$ is terminating. Now, we let $t \xleftarrow[\mathcal{R}_n]{M_1} s \xrightarrow[\mathcal{R}_n]{M_2} u$ be a labeled local peak in $\mathcal{R}_n$ and show that $\mathcal{R}_n$ is peak decreasing. By Lemma 7, we have $t \downarrow_{\mathcal{R}_n} u$ or $t \leftrightarrow_{CP(\mathcal{R}_n)} u$, and thus $t \downarrow_{\mathcal{R}_n} u$ or $(t, u) \in \bigcup_{i=0}^n \leftrightarrow_{\mathcal{E}_i}$ by fairness and Definition 13(iii). Here, we only consider the case where $(t, u) \in \bigcup_{i=0}^n \leftrightarrow_{\mathcal{E}_i}$ and leave the proof of the other case to our formalization. Since $(t, u) \in \bigcup_{i=0}^n \leftrightarrow_{\mathcal{E}_i}$, there is some k such that $(t, u) \in \leftrightarrow_{\mathcal{E}_k}$. Now, let $M := \{t, u\}$. Then, we have $(t, u) \in \xleftrightarrow[\mathcal{E}_k]{M}$, and thus $(t, u) \in \xleftrightarrow[\mathcal{E}_k \cup \mathcal{R}_k]{M}{}^*$. By repeated applications of Lemma 16, we have $(t, u) \in \xleftrightarrow[\mathcal{R}_n]{M}{}^*$. This means that there is a conversion between t and u, where each step in the conversion is labeled by M. Let $\succ_{sl}^{mul}$ denote the multiset extension of $\succ_{sl}$. Then, $\succ_{sl}^{mul}$ is well-founded because $\succ_{sl}$ is well-founded. Since $M_1 \succ_{sl}^{mul} M$ and $M_2 \succ_{sl}^{mul} M$, $\to_{\mathcal{R}_n}$ is peak decreasing, and thus confluent by Lemma 18. ◀

Our formalization of the inference rules in Figure 1 for a completion step uses the "inductively defined predicates" in Isabelle/HOL.

```
inductive sts_compl_step:: "sts × sts ⇒ sts × sts ⇒ bool" (infix "⊢SR" 55) where
  deduce: "(E, R) ⊢SR (E ∪ {(s @ u3, u1 @ t)}, R)"
    if "(u1 @ u2, s) ∈ R" and "(u2 @ u3, t) ∈ R" and "u2 ≠ []"
| simplifyl: "(E ∪ {(u1 @ u2 @ u3, s)}, R) ⊢SR (E ∪ {(u1 @ t @ u3, s)}, R)"
    if "(u2, t) ∈ R"
| simplifyr: "(E ∪ {(s, u1 @ u2 @ u3)}, R) ⊢SR (E ∪ {(s, u1 @ t @ u3)}, R)"
    if "(u2, t) ∈ R"
| orientl: "(E ∪ {(s, t)}, R) ⊢SR (E, {R ∪ {(s, t)})"   if "s ≻sl t"
| orientr: "(E ∪ {(s, t)}, R) ⊢SR (E, {R ∪ {(t, s)})"   if "t ≻sl s"
| collapse: "(E, R ∪ {(u1 @ u2 @ u3, s)}) ⊢SR (E ∪ {(u1 @ t @ u3, s)}, R)"   if "(u2, t) ∈ R"
| compose: "(E, R ∪ {(s, u1 @ u2 @ u3)}) ⊢SR (E, R ∪ {(s, u1 @ t @ u3)})"   if "(u2, t) ∈ R"
| delete: "(E ∪ {(s, s)}, R) ⊢SR (E, R)"
```

Note that in Figure 1, $\mathcal{E}$ consists of unordered pairs (i.e., equations). In our formalization above, it consists of ordered pairs for technical convenience, so we use *simplifyl* and *simplifyr* for the simplify rule and *orientl* and *orieintr* for the orient rule in Figure 1.

Now, Theorem 19 is formalized as follows, where the assumptions describe a fair non-failing run $(\mathcal{E}_0, \mathcal{R}_0) \vdash_{SR} (\mathcal{E}_1, \mathcal{R}_1) \vdash_{SR} \cdots \vdash_{SR} (\mathcal{E}_n, \mathcal{R}_n)$ in Theorem 19.

```
theorem sts_finite_fair_run_complete: assumes R 0 = {} and E n = {}
  and   "∀i < n . (E i, R i) ⊢SR (E (Suc i), R (Suc i))"
  and   "sts_critical_pairs (R n) ⊆ (⋃i ≤ n . (ststep (E i))↔)"
  shows   "complete (ststep (R n))"
```

▶ Remark 20. Using an inference system for a completion procedure is known to be simple and efficient, and it is easy to apply simplification rules [1]. An inference system-based completion procedure for TRSs was already discussed in the literature [1,2,18]. Our inference system in Figure 1 is adapted from the existing Knuth-Bendix completion procedures for STSs [7, 24] that do not use an inference system. (There are also inference systems for equational reasoning over strings, such as [27,28], but they are not directly applicable to a completion procedure for STSs.) Regarding a completion procedure for finitely presented monoids and groups, one may attempt to use an inference system discussed in abstract completion [2,18] by converting STSs into TRSs as discussed in Section 1. This is not tailored toward a completion procedure for finitely presented monoids and groups. In particular, we use the simple string matching method and the simple shortlex order on strings in Figure 1 rather than using more complex term structures for unification/matching and ordering. Note that given a fixed signature with its precedence, it takes only linear time to compare two finite strings w.r.t. the shortlex order. It takes also linear time to match two finite strings using, for example, the Knuth–Morris–Pratt (KMP) algorithm [30].

4 Conditional Semi-Thue Systems

Conditional semi-Thue systems (CSTSs) [11] are extensions of STSs in the sense that their rules may admit non-empty conditions. A CSTS on Σ^* [11] is defined to be a set of conditional string rewriting rules, where each rule has the form $(\ell, r) \Leftarrow \phi$ (also denoted as $\ell \to r \Leftarrow \phi$) consisting of strings $\ell, r \in \Sigma^*$ and a sequence ϕ of unordered pairs of strings on Σ^*. Here, ϕ can be thought of as a conjunction of conditions for the rule $(\ell, r) \Leftarrow \phi$. In the following, let $\mathcal{R}$ be a CSTS on Σ^*.

▶ **Definition 21.** We write $\mathcal{R} \vdash_{lr} s \approx t$ if the equation $s \approx t$ is derivable from the inference rules in Figure 2, and say that $s \approx t$ belongs to the *left-right conditional equational theory* induced by $\mathcal{R}$.

We may also consider other types of conditional equational theories depending on how contexts are used for the conditions in each conditional string rewriting rule (cf. [11]).

Replacement (R): $$\frac{\forall (s \approx t) \in \phi : su \approx tu}{\ell u \approx r u} \qquad \text{for all } (\ell, r) \Leftarrow \phi \in \mathcal{R}.$$

Above, each condition is considered with a *right context* $u \in \Sigma^*$ and this context is also used for the conclusion.

▶ **Definition 22.** We write $\mathcal{R} \vdash_r s \approx t$ if the equation $s \approx t$ is derivable from the inference rules in Figure 2 substituting the above "Replacement (R)" rule for the "Replacement (LR)" rule. If $\mathcal{R} \vdash_r s \approx t$, then we say that $s \approx t$ belongs to the *right conditional equational theory* induced by $\mathcal{R}$.

Replacement (P): $$\frac{\forall (s \approx t) \in \phi : s \approx t}{\ell \approx r} \qquad \text{for all } (\ell, r) \Leftarrow \phi \in \mathcal{R}.$$

Above, no additional context is used for each condition and the conclusion.

Reflexivity: $\dfrac{}{s \approx s}$

Symmetry: $\dfrac{s \approx t}{t \approx s}$

Transitivity: $\dfrac{s \approx t \qquad t \approx u}{s \approx u}$

Congruence: $\dfrac{s \approx t}{usv \approx utv}$

Replacement (LR): $\dfrac{\forall (s \approx t) \in \phi : usv \approx utv}{u\ell v \approx urv}$ for all $(\ell, r) \Leftarrow \phi \in \mathcal{R}$.

Above, $s, t, u, v \in \Sigma^*$ and $\mathcal{R}$ is a CSTS on Σ^* .

Figure 2 Inference rules for yielding a left-right conditional equational theory induced by a conditional semi-Thue system $\mathcal{R}$ on Σ^*.

▶ **Definition 23.** We write $\mathcal{R} \vdash_p s \approx t$ if the equation $s \approx t$ is derivable from the inference rules in Figure 2 substituting the above "Replacement (P)" rule for the "Replacement (LR)" rule. If $\mathcal{R} \vdash_p s \approx t$, then we say that $s \approx t$ belongs to the *pure conditional equational theory* induced by $\mathcal{R}$.

Our formalization of the inference rules in Figure 2 for the left-right conditional equational theory induced by $\mathcal{R}$ also uses the "inductively defined predicates" in Isabelle/HOL.

```
inductive cond_eq_lr_theory :: "csts ⇒ string × string ⇒ bool" (infix "⊢celr" 55) where
  refl: "R ⊢celr (s, s)"
| symmetry: "R ⊢celr (s, t)" if "R ⊢celr (t, s)"
| transitive: "R ⊢celr (s, u)" if "R ⊢celr (s, t) ∧ R ⊢celr (t, u)"
| congruence: "R ⊢celr (u @ s @ v, u @ t @ v)" if "R ⊢celr (s, t)"
| replacement: "R ⊢celr (u @ l @ w, u @ r @ w)"
    if "((l, r), cs) ∈ R ∧ (∀(si, ti) ∈ set cs. R ⊢celr (u @ si @ w, u @ ti @ w))"
```

Above, $R \vdash_{celr} (s, t)$ denotes that $s \approx t$ belongs to the left-right conditional equational theory induced by $\mathcal{R}$. The right (resp. pure) conditional equational theory induced by $\mathcal{R}$ can be formalized in a similar way, where $\vdash_{cer}$ (resp. $\vdash_{cep}$) is used for inductively defined predicate for the right (resp. pure) conditional equational theory induced by $\mathcal{R}$. They are only different from the *replacement* part (see below):

```
replacement: "R ⊢cer (l @ w, r @ w)"
    if "((l, r), cs) ∈ R ∧ (∀(si, ti) ∈ set cs. R ⊢celr (si @ w, ti @ w))"

replacement: "R ⊢cep (l, r)"
    if "((l, r), cs) ∈ R ∧ (∀(si, ti) ∈ set cs. R ⊢celr (si, ti))"
```

Meanwhile, the types of CSTSs depend on the string rewriting relations induced by CSTSs. More specifically, the types of CSTSs depend on how conditions are evaluated in the conditional parts of their conditional string rewriting rules (cf. [5, 46]). The induced string rewriting relations from CSTSs are structured into *levels* (cf. [41, 46]).

▶ **Definition 24.** The string rewriting relation $\to_{\mathcal{R},lr,s}$ for a *left-right-semi-equational* CSTS $\mathcal{R}$ on Σ^* is defined as follows: $t_1 \to_{\mathcal{R},lr,s} t_2$ iff $t_1 \to_{\mathcal{R}_n} t_2$ for some $n \geq 0$. Here, the unconditional STS $\mathcal{R}_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}_0 := \emptyset \\ \mathcal{R}_{n+1} := \{(u\ell v, urv) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } usv \overset{*}{\leftrightarrow}_{\mathcal{R}_n} utv \text{ for all } s \approx t \in \phi \text{ and } u, v \in \Sigma^*\} \end{cases}$$

The string rewriting relation $\to_{\mathcal{R},lr,j}$ for a *left-right-join* CSTS $\mathcal{R}$ on Σ^* is defined as: $t_1 \to_{\mathcal{R},lr,j} t_2$ iff $t_1 \to_{\mathcal{R}'_n} t_2$ for some $n \geq 0$, where $\mathcal{R}'_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}'_0 := \emptyset \\ \mathcal{R}'_{n+1} := \{(u\ell v, urv) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } usv \downarrow_{\mathcal{R}'_n} utv \text{ for all } s \approx t \in \phi \text{ and } u, v \in \Sigma^*\} \end{cases}$$

▶ **Definition 25.** The string rewriting relation $\to_{\mathcal{R},r,s}$ for a *right-semi-equational* CSTS $\mathcal{R}$ on Σ^* is defined as follows: $t_1 \to_{\mathcal{R},r,s} t_2$ iff $t_1 \to_{\mathcal{R}_n} t_2$ for some $n \geq 0$. Here, the unconditional STS $\mathcal{R}_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}_0 := \emptyset \\ \mathcal{R}_{n+1} := \{(\ell v, rv) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } sv \overset{*}{\leftrightarrow}_{\mathcal{R}_n} tv \text{ for all } s \approx t \in \phi \text{ and } v \in \Sigma^*\} \end{cases}$$

The string rewriting relation $\to_{\mathcal{R},r,j}$ for a *right-join* CSTS $\mathcal{R}$ on Σ^* is defined as: $t_1 \to_{\mathcal{R},r,j} t_2$ iff $t_1 \to_{\mathcal{R}'_n} t_2$ for some $n \geq 0$, where $\mathcal{R}'_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}'_0 := \emptyset \\ \mathcal{R}'_{n+1} := \{(\ell v, rv) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } sv \downarrow_{\mathcal{R}'_n} tv \text{ for all } s \approx t \in \phi \text{ and } v \in \Sigma^*\} \end{cases}$$

▶ **Definition 26.** The string rewriting relation $\to_{\mathcal{R},p,s}$ for a *pure-semi-equational* CSTS $\mathcal{R}$ on Σ^* is defined as follows: $t_1 \to_{\mathcal{R},p,s} t_2$ iff $t_1 \to_{\mathcal{R}_n} t_2$ for some $n \geq 0$. Here, the unconditional STS $\mathcal{R}_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}_0 := \emptyset \\ \mathcal{R}_{n+1} := \{(\ell, r) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } s \overset{*}{\leftrightarrow}_{\mathcal{R}_n} t \text{ for all } s \approx t \in \phi\} \end{cases}$$

The string rewriting relation $\to_{\mathcal{R},p,j}$ for a *pure-join* CSTS $\mathcal{R}$ on Σ^* is defined as: $t_1 \to_{\mathcal{R},p,j} t_2$ iff $t_1 \to_{\mathcal{R}'_n} t_2$ for some $n \geq 0$, where $\mathcal{R}'_n$ are inductively defined as follows:

$$\begin{cases} \mathcal{R}'_0 := \emptyset \\ \mathcal{R}'_{n+1} := \{(\ell, r) \,|\, (\ell, r) \Leftarrow \phi \in \mathcal{R}, \text{ and } s \downarrow_{\mathcal{R}'_n} t \text{ for all } s \approx t \in \phi\} \end{cases}$$

The following lemma says that our inference system for yielding the left-right, right, and pure conditional equational theories is sound w.r.t. the Thue congruences induced by the string rewriting relations associated with the left-right-semi-equational, right-semi-equational, and pure-semi-equational CSTSs, respectively.

▶ **Lemma 27.** *Let $\mathcal{R}$ be a CSTS on Σ^* and $t_1, t_2 \in \Sigma^*$.*

(i) $t_1 \overset{*}{\leftrightarrow}_{\mathcal{R},lr,s} t_2$ *iff* $\mathcal{R} \vdash_{lr} t_1 \approx t_2$. ☑

(ii) $t_1 \overset{*}{\leftrightarrow}_{\mathcal{R},r,s} t_2$ *iff* $\mathcal{R} \vdash_{r} t_1 \approx t_2$. ☑

(iii) $t_1 \overset{*}{\leftrightarrow}_{\mathcal{R},p,s} t_2$ *iff* $\mathcal{R} \vdash_{p} t_1 \approx t_2$. ☑

The bidirectional use of the string rewriting relations for evaluating the conditions in semi-equational CSTSs is both inefficient and unnatural (cf. [46]). We can use the join CSTSs instead of the semi-equational CSTSs if they have the confluence property.

▶ **Lemma 28.** *Let $\mathcal{R}$ be a CSTS on Σ^*.*

(i) *If $\to_{\mathcal{R},lr,j}$ is confluent, then $\to_{\mathcal{R},lr,s} = \to_{\mathcal{R},lr,j}$.* ☑

(ii) *If $\to_{\mathcal{R},r,j}$ is confluent, then $\to_{\mathcal{R},r,s} = \to_{\mathcal{R},r,j}$.* ☑

(iii) *If $\to_{\mathcal{R},p,j}$ is confluent, then $\to_{\mathcal{R},p,s} = \to_{\mathcal{R},p,j}$.* ☑

▶ Remark 29. Recall that an STS S on Σ^* can be associated with a TRS $\mathcal{R}_S$, where each letter from Σ is interpreted as a unary function symbol. Similarly, a right-join CSTS S' on Σ^* can be associated with a join CTRS $\mathcal{R}_{S'}$ in such a way that $\mathcal{R}_{S'} := \{\ell(x) \to r(x) \Leftarrow s_1(x) \approx t_1(x), \ldots, s_n(x) \approx t_n(x) \mid \ell \to r \Leftarrow s_1 \approx t_1, \ldots, s_n \approx t_n \in S'\}$ [11]. (Here, variables in $\mathcal{R}_{S'}$ are renamed whenever necessary.) However, this is not the case for left-right-join CSTSs.

▶ **Example 30.** Consider $\mathcal{R} = \{a\ell \to b\ell, \ell am \to \ell bm, c \to d \Leftarrow a \approx b\}$, where $\Sigma = \{a, b, c, d, \ell, m\}$ with $a > b > c > d > \ell > m$. If $\mathcal{R}$ is a pure-join CSTS, then neither $c\ell \to_{\mathcal{R},p,j} d\ell$ nor $\ell cm \to_{\mathcal{R},p,j} \ell dm$ holds. If $\mathcal{R}$ is a right-join CSTS, then $c\ell \to_{\mathcal{R},r,j} d\ell$ holds, but $\ell cm \to_{\mathcal{R},r,j} \ell dm$ does not hold. If $\mathcal{R}$ is a left-right-join CSTS, then both $c\ell \to_{\mathcal{R},rl,j} d\ell$ and $\ell cm \to_{\mathcal{R},rl,j} \ell dm$ hold.

Now, we discuss our formalization of right-join and pure-join CSTSs. (Our formalization of left-right-join CSTSs is similar and is omitted.) First, the string rewriting relation $\to_{\mathcal{R},r,j}$ associated with a right-join CSTS is formalized as follows:

```
definition "csr_r_join_step R = (⋃n. csr_r_join_step_n R n)"

fun csr_r_join_step_n:: "csts ⇒ nat ⇒ string rel" where
  "csr_r_join_step_n R 0 = {}" |
  "csr_r_join_step_n R (Suc n) =
    {(C⟪ℓ @ w⟫, C⟪r @ w⟫⟫) | C ℓ r cs w. ((ℓ, r), cs) ∈ R ∧
    (∀(s_i, t_i) ∈ set cs. (s_i @ w, t_i @ w) ∈ (csr_r_join_step_n R n)↓)}"
```

In the csr_r_join_step_n function, C is a (string) context and $C\langle\!\langle \ell@w \rangle\!\rangle$ (resp. $C\langle\!\langle r@w \rangle\!\rangle$) denotes the application of the context C to the string ℓw (resp. rw). We formalize a context for strings based on the existing formalization of contexts for terms in IsaFoR.

```
datatype sctxt = Hole "(◯)" | More "string" "sctxt" "string"

fun sctxt_apply_string :: "sctxt ⇒ string ⇒ string" ("_⟪_⟫" [900, 0] 900) where
  "◯⟪s⟫ = s" |
  "(More ss1 C ss2)⟪s⟫ = (ss1 @ (C⟪s⟫) @ ss2)"
```

Using the above function, $C\langle\!\langle s \rangle\!\rangle$ denotes usv for some (possibly empty) strings u and v. In the csr_r_join_step_n function, we consider all possible contexts C because $\to_{\mathcal{R},r,j}$ is a string rewriting relation, i.e., $\ell \to_{\mathcal{R},r,j} r$ implies $u\ell v \to_{\mathcal{R},r,j} urv$ for all (possibly empty) strings u and v. Next, the string rewriting relation $\to_{\mathcal{R},p,j}$ associated with a pure-join CSTS is formalized as follows. Here, no additional string is appended for evaluating each condition.

```
definition "csr_p_join_step R = (⋃n. csr_p_join_step_n R n)"

fun csr_p_join_step_n:: "csts ⇒ nat ⇒ string rel" where
  "csr_p_join_step_n R 0 = {}" |
  "csr_p_join_step_n R (Suc n) =
    {(C⟪ℓ⟫, C⟪r⟫) | C ℓ r cs. ((ℓ, r), cs) ∈ R ∧ (∀(s_i, t_i) ∈
    set cs. (s_i, t_i) ∈ (csr_p_join_step_n R n)↓)}"
```

Our formalization of CSTSs and their associated monoids relies on the following locale.

```
locale conditional_semi_Thue =
 fixes R :: csts  and S :: "char set"
 assumes   "finite S"
  and      "S ≠ {}"
  and      "finite R"
```

For example, the locale for right-join CSTSs extends the above locale and is described as:

```
locale conditional_r_join_semi_Thue = reductive_r_join + conditional_semi_Thue R S
 for R :: csts and S :: "char set" +
 fixes Thue_R_Congruence :: sts
 assumes   "Thue_R_Congruence = (csr_r_join_step R)^↔* "
  and      "Thue_R_Congruence ⊆ S* × S* "
begin
...
```

Since the locale conditional_r_join_semi_Thue extends the locale conditional_semi_Thue, the alphabet denoted by S is assumed to be finite and non-empty. The assumptions in the locale conditional_r_join_semi_Thue also represent the assumption $\overset{*}{\leftrightarrow}_{\mathcal{R},r,j} \subseteq \Sigma^* \times \Sigma^*$, which also implies the assumption $\rightarrow_{\mathcal{R},r,j} \subseteq \Sigma^* \times \Sigma^*$.

▶ **Definition 31.**

(i) We call a CSTS $\mathcal{R}$ *reductive* if for all $(\ell, r) \Leftarrow \phi \in \mathcal{R}$, $\ell \succ_{sl} r$, $\ell \succ_{sl} s_i$, and $\ell \succ_{sl} t_i$ for all $(s_i, t_i) \in \phi$ (cf. *decreasingness* [12] in CTRSs).

(ii) Two STSs $\mathcal{R}_1$ and $\mathcal{R}_2$ on Σ^* are *equivalent* if $\overset{*}{\leftrightarrow}_{\mathcal{R}_1} = \overset{*}{\leftrightarrow}_{\mathcal{R}_2}$, that is, they present the same monoid.

(iii) An STS $\mathcal{R}_1$ and a reductive right-join (resp. a pure-join) CSTS $\mathcal{R}_2$ on Σ^* are *equivalent* if $\overset{*}{\leftrightarrow}_{\mathcal{R}_1} = \overset{*}{\leftrightarrow}_{\mathcal{R}_2,r,j}$ (resp. $\overset{*}{\leftrightarrow}_{\mathcal{R}_1} = \overset{*}{\leftrightarrow}_{\mathcal{R}_2,p,j}$).

▶ **Example 32** (see [11]). The monoid $\mathcal{M} = (\Sigma; \mathcal{R})$, where $\Sigma = \{a, b\}$ and $\mathcal{R} = \{aba \rightarrow ba\}$, is finitely presented and have decidable word problem, but does not admit an equivalent monoid presentation with $(\Sigma; \mathcal{R}')$, where $\mathcal{R}'$ is a finite complete STS. More specifically, a completion procedure for $\mathcal{R}$ may only yield an infinite complete semi-Thue system $\{ab^n a \rightarrow b^n a \mid n \geq 1\}$. However, $\mathcal{R}$ admits an equivalent finite complete (reductive) right-join CSTS $\mathcal{R}'' = \{aba \rightarrow ba, abb \rightarrow bb \Leftarrow ab \approx b\}$. (Here, $(\Sigma; \mathcal{R}'')$ is a finite complete (reductive) conditional presentation of $\mathcal{M}$, which will be discussed later in this section.)

Unlike the string rewriting relations induced by semi-Thue systems, $\rightarrow_{\mathcal{R},r,j}$ is not necessarily decidable even if it is terminating (cf. [37]).

▶ **Lemma 33.**

(i) *If $\mathcal{R}$ is a finite reductive right-join CSTS, then $\rightarrow_{\mathcal{R},r,j}$ is terminating and the set $\Delta^*_{R,r,j,s} = \{t \mid s \rightarrow^*_{\mathcal{R},r,j} t\}$ is finite for all $s \in \Sigma^*$.* ☑

(ii) *If $\mathcal{R}$ is a finite reductive pure-join CSTS, then $\rightarrow_{\mathcal{R},p,j}$ is terminating and the set $\Delta^*_{R,p,j,s} = \{t \mid s \rightarrow^*_{\mathcal{R},p,j} t\}$ is finite for all $s \in \Sigma^*$.* ☑

In the above lemma, if $\mathcal{R}$ is finite and reductive and $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is terminating, then we may simulate the decidability of $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) using the finiteness of the set of descendants $\Delta^*_{R,r,j,s}$ (resp. $\Delta^*_{R,p,j,s}$) for all $s \in \Sigma^*$. Here, the set $\Delta^*_{R,r,j,s}$ (resp. $\Delta^*_{R,p,j,s}$) can be shown to be computable for all $s \in \Sigma^*$ using the well-founded induction on $\succ_{sl}$ if $\mathcal{R}$ is finite and reductive (cf. Theorem 7.2.9 in [37]).

▶ **Definition 34.**

(i) Let $\mathcal{R}$ be a right-join CSTS. For each pair of not necessarily distinct conditional string rewriting rules from $\mathcal{R}$, say $(u_0, v_0) \Leftarrow \forall_{i=1}^{m} u_i \approx v_i$ and $(u_0', v_0') \Leftarrow \forall_{i=1}^{n} u_i' \approx v_i'$, the set of *critical pairs* w.r.t. $\to_{\mathcal{R},r,j}$ corresponding to this pair is $\{(v_0 y, x v_0') \Leftarrow \forall_{i=1}^{m} u_i y \approx v_i y \wedge \forall_{i=1}^{n} u_i' \approx v_i' \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 y = x u_0' \text{ and } |x| < |u_0|\} \cup \{(v_0, x v_0' y) \Leftarrow \forall_{i=1}^{m} u_i \approx v_i \wedge \forall_{i=1}^{n} u_i' y \approx v_i' y \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 = x u_0' y\}$.

(ii) A critical pair $(s_0, t_0) \Leftarrow \forall_{i=1}^{n} s_i \approx t_i$ is *joinable* w.r.t. $\to_{\mathcal{R},r,j}$ if for any $y \in \Sigma^*$, $s_i y \downarrow_{\mathcal{R},r,j} t_i y$ for all $1 \leq i \leq n$ implies $s_0 y \downarrow_{\mathcal{R},r,j} t_0 y$.

(iii) Let $\mathcal{R}$ be a pure-join CSTS. For each pair of not necessarily distinct conditional string rewriting rules from $\mathcal{R}$, say $(u_0, v_0) \Leftarrow \forall_{i=1}^{m} u_i \approx v_i$ and $(u_0', v_0') \Leftarrow \forall_{i=1}^{n} u_i' \approx v_i'$, the set of *critical pairs* w.r.t. $\to_{\mathcal{R},p,j}$ corresponding to this pair is $\{(v_0 y, x v_0') \Leftarrow \forall_{i=1}^{m} u_i \approx v_i \wedge \forall_{i=1}^{n} u_i' \approx v_i' \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 y = x u_0' \text{ and } |x| < |u_0|\} \cup \{(v_0, x v_0' y) \Leftarrow \forall_{i=1}^{m} u_i \approx v_i \wedge \forall_{i=1}^{n} u_i' \approx v_i' \mid \text{there are } x, y \in \Sigma^* \text{ such that } u_0 = x u_0' y\}$.

(iv) A critical pair $(s_0, t_0) \Leftarrow \forall_{i=1}^{n} s_i \approx t_i$ is *joinable* w.r.t. $\to_{\mathcal{R},p,j}$ if $s_i \downarrow_{\mathcal{R},p,j} t_i$ for all $1 \leq i \leq n$ implies $s_0 \downarrow_{\mathcal{R},p,j} t_0$.

Now, we discuss our formalization of Definition 34. First, the set of critical pairs for a right-join CSTS $\mathcal{R}$ is formalized as follows:

```
definition csts_r_critical_pairs where
  "csts_r_critical_pairs R = {((v @ y, x @ v'), cs' @ map(λ(lhs, rhs) . (lhs @ y, rhs @ y)) cs) |
  x y u v u' v' cs cs' . ((u, v), cs) ∈ R ∧ ((u', v'), cs') ∈ R ∧ u @ y = x @ u' ∧ length x < length u} ∪
  {((v, x @ v' @ y), cs @ map(λ(lhs, rhs) . (lhs @ y, rhs @ y)) cs') | x y u v u' v' cs cs' . ((u, v), cs)
  ∈ R ∧ ((u', v'), cs') ∈ R ∧ u = x @ u' @ y}"
```

Above, if cs is of the form $\forall_{i=1}^{m} u_i \approx v_i$, then $map(\lambda(lhs, rhs) . (lhs @ y, rhs @ y))\ cs$ is $\forall_{i=1}^{m} u_i y \approx v_i y$. Also, $((\ell, r), cs)$ denotes the conditional string rewriting rule $(\ell, r) \Leftarrow cs$, where cs is the list type for technical convenience. Based on Definition 34(ii), the statement that all critical pairs of $\mathcal{R}$ are joinable w.r.t. $\to_{\mathcal{R},r,j}$ is formalized as follows:

```
definition all_ccps_r_joinable where
  "all_ccps_r_joinable R = (∀s t cs . ((s, t), cs) ∈ csts_r_critical_pairs R →
  (∀y . csts_conds_r_sat R cs y → (s @ y, t @ y) ∈ (csr_r_join_step R)↓))"
```

where the definition of csts_conds_r_sat is formalized as follows:

```
definition csts_conds_r_sat where
  "csts_conds_r_sat R cs y ⟷ (∀(s_i, t_i) ∈ set cs . (s_i @ y, t_i @ y) ∈ (csr_r_join_step R)↓)"
```

Meanwhile, the set of critical pairs for a pure-join CSTS $\mathcal{R}$ is formalized as follows:

```
definition csts_p_critical_pairs where
  "csts_p_critical_pairs R = {((v @ y, x @ v'), cs' @ cs) | x y u v u' v' cs cs' . ((u, v), cs) ∈ R ∧
  ((u', v'), cs') ∈ R ∧ u @ y = x @ u' ∧ length x < length u} ∪ {((v, x @ v' @ y), cs @ cs' |
  x y u v u' v' cs cs' . ((u, v), cs) ∈ R ∧ ((u', v'), cs') ∈ R ∧ u = x @ u' @ y}"
```

Note that conditions are evaluated without using any context in the above formalization. The statement that all critical pairs of $\mathcal{R}$ are joinable w.r.t. $\to_{\mathcal{R},p,j}$ is formalized as follows:

```
definition all_ccps_p_joinable where
  "all_ccps_p_joinable R = (∀s t cs . ((s, t), cs) ∈ csts_p_critical_pairs R →
  csts_conds_p_sat R cs → (s, t) ∈ csr_p_join_step R)↓)"
```

where the definition of csts_conds_p_sat is simply formalized as follows:

definition *csts_conds_p_sat where*
"csts_conds_p_sat R cs ⟷ (∀(s_i, t_i) ∈ set cs . (s_i, t_i) ∈ (csr_p_join_step R)^↓)"

▶ **Lemma 35** (see [11]). *Let $\mathcal{R}$ be a finite reductive right-join CSTS. Then, $\to_{\mathcal{R},r,j}$ is confluent if and only if all critical pairs of $\mathcal{R}$ are joinable w.r.t. $\to_{\mathcal{R},r,j}$.* ☑

▶ **Lemma 36.** *Let $\mathcal{R}$ be a finite reductive pure-join CSTS. Then, $\to_{\mathcal{R},p,j}$ is confluent if and only if all critical pairs of $\mathcal{R}$ are joinable w.r.t. $\to_{\mathcal{R},p,j}$.* ☑

Proof. Since the "only if" direction is straightforward, we only show the "if" direction. Suppose that all critical pairs of $\mathcal{R}$ are joinable w.r.t. $\to_{\mathcal{R},p,j}$. Since $\mathcal{R}$ is finite and reductive, $\to_{\mathcal{R},p,j}$ is terminating and decidable by Lemma 33(ii), so by Newman's Lemma, it suffices to show that $\to_{\mathcal{R},p,j}$ is locally confluent. Let $s \to_{\mathcal{R},p,j} t$ and $s \to_{\mathcal{R},p,j} u$, and let $(\ell, r) \Leftarrow \forall_{i=1}^m s_i \approx t_i$ and $(\ell', r') \Leftarrow \forall_{i=1}^n s'_i \approx t'_i$ be two rules to reduce s to t and u, respectively. We show that $t \downarrow_{\mathcal{R},p,j} u$ by considering the following three cases according to the positions of ℓ and ℓ' in s.

1. ℓ and ℓ' do not overlap, i.e., $s = x\ell y\ell' z$ and $t = xry\ell' z \;{}_{\mathcal{R},p,j}\!\leftarrow x\ell y\ell' z \to_{\mathcal{R},p,j} x\ell yr' z = u$ using the rules $(\ell, r) \Leftarrow \forall_{i=1}^m s_i \approx t_i$ and $(\ell', r') \Leftarrow \forall_{i=1}^n s'_i \approx t'_i$, where $x, y, z \in \Sigma^*$. Since $\forall_{i=1}^m s_i \downarrow_{\mathcal{R},p,j} t_i$ and $\forall_{i=1}^n s'_i \downarrow_{\mathcal{R},p,j} t'_i$, we have $t = xry\ell' z \to_{\mathcal{R},p,j} xryr' z \;{}_{\mathcal{R},p,j}\!\leftarrow x\ell yr' z = u$. Therefore, t and u are joinable w.r.t. $\to_{\mathcal{R},p,j}$.
2. ℓ' is a substring of ℓ, i.e., $s = x\ell y = xu\ell' vy$ and $t = xry \;{}_{\mathcal{R},p,j}\!\leftarrow x\ell y = xu\ell' vy \to_{\mathcal{R},p,j} xur' vy = u$ using the rules $(\ell, r) \Leftarrow \forall_{i=1}^m s_i \approx t_i$ and $(\ell', r') \Leftarrow \forall_{i=1}^n s'_i \approx t'_i$, where $x, y, u, v \in \Sigma^*$. Since $\ell = u\ell' v$, we have a critical pair $(r, ur' v) \Leftarrow \forall_{i=1}^m s_i \approx t_i \wedge \forall_{i=1}^n s'_i \approx t'_i$. This critical pair is joinable w.r.t. $\to_{\mathcal{R},p,j}$ by hypothesis with $\forall_{i=1}^m s_i \downarrow_{\mathcal{R},p,j} t_i$ and $\forall_{i=1}^n s'_i \downarrow_{\mathcal{R},p,j} t'_i$, so there is some w such that $r \to^*_{\mathcal{R},p,j} w$ and $ur' v \to^*_{\mathcal{R},p,j} w$. Since $xry \to^*_{\mathcal{R},p,j} xwy$ and $xur' vy \to^*_{\mathcal{R},p,j} xwy$, we have $xry \downarrow_{\mathcal{R},p,j} xur' vy$, and thus t and u are joinable w.r.t. $\to_{\mathcal{R},p,j}$.
3. ℓ and ℓ' have an overlap in such a way that $s = x\ell uy = xv\ell' y$ and $t = xruy \;{}_{\mathcal{R},p,j}\!\leftarrow x\ell uy = xv\ell' y \to_{\mathcal{R},p,j} xvr' y = u$ using the rules $(\ell, r) \Leftarrow \forall_{i=1}^m s_i \approx t_i$ and $(\ell', r') \Leftarrow \forall_{i=1}^n s'_i \approx t'_i$, where $|v| < |\ell|$ and $x, y, u, v \in \Sigma^*$. Since $\ell u = v\ell'$ with $|v| < |\ell|$, we have a critical pair $(ru, vr') \Leftarrow \forall_{i=1}^m s_i \approx t_i \wedge \forall_{i=1}^n s'_i \approx t'_i$. This critical pair is also joinable w.r.t. $\to_{\mathcal{R},p,j}$ by hypothesis with $\forall_{i=1}^m s_i \downarrow_{\mathcal{R},p,j} t_i$ and $\forall_{i=1}^n s'_i \downarrow_{\mathcal{R},p,j} t'_i$, so there is some w such that $ru \to^*_{\mathcal{R},p,j} w$ and $vr' \to^*_{\mathcal{R},p,j} w$. Since $xruy \to^*_{\mathcal{R},p,j} xwy$ and $xvr' y \to^*_{\mathcal{R},p,j} xwy$, we have $xruy \downarrow_{\mathcal{R},p,j} xvr' y$, and thus t and u are joinable w.r.t. $\to_{\mathcal{R},p,j}$.

This shows that $t \downarrow_{\mathcal{R},p,j} u$, and thus $\to_{\mathcal{R},p,j}$ is locally confluent. ◀

Lemmas 35 and 36 are simply formalized respectively as follows:

lemma *csts_r_critical_pair_lemma: assumes "reductive R"*
"CR (csr_r_join_step R) ⟷ all_ccps_r_joinable R"

lemma *csts_p_critical_pair_lemma: assumes "reductive R"*
"CR (csr_p_join_step R) ⟷ all_ccps_p_joinable R"

For reductive right-join (or pure-join) CSTSs, joinability of critical pairs suffices to show confluence by Lemma 35 (resp. Lemma 36). However, this is not the case for reductive left-right-join CSTSs. In particular, unlike reductive right-join or pure-join CSTSs, non-overlap may not be joinable for reductive left-right-join CSTSs as shown in the following example.

▶ **Example 37.** Consider the left-right-join CSTS $\mathcal{R} = \{b \to u \Leftarrow i \approx j, c \to v \Leftarrow l \approx m, aic \to ajc, bld \to bmd\}$ over $\Sigma = \{a, b, c, d, i, j, l, m, u, v\}$. Using the shortlex ordering $\succ_{sl}$ induced by the precedence $a > b > c > d > i > j > l > m > u > v$, we see

that $\mathcal{R}$ is reductive. Now, consider a non-overlap $aucd \;{}_{\mathcal{R},rl,j}\!\leftarrow abcd \rightarrow_{\mathcal{R},rl,j} abvd$. Here, the step $abcd \rightarrow_{\mathcal{R},rl,j} aucd$ uses the rules $b \rightarrow u \Leftarrow i \approx j$ and $aic \rightarrow ajc$, and the step $abcd \rightarrow_{\mathcal{R},rl,j} abvd$ uses the rules $c \rightarrow v \Leftarrow l \approx m$ and $bld \rightarrow bmd$. However, it is not the case that $aucd \downarrow_{\mathcal{R},rl,j} abvd$.

Now, we discuss the monoids defined by certain congruence relations induced by the string rewriting relations $\rightarrow_{\mathcal{R},r,j}$ and $\rightarrow_{\mathcal{R},p,j}$.

▶ **Definition 38.** Let $\mathcal{R}$ be a reductive right-join (resp. a pure-join) CSTS on Σ^*. The *Thue congruence* induced by $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is the relation $\overset{*}{\leftrightarrow}_{\mathcal{R},r,j}$ (resp. $\overset{*}{\leftrightarrow}_{\mathcal{R},p,j}$). Two strings $u, v \in \Sigma^*$ are *congruent* w.r.t. $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) if $u \overset{*}{\leftrightarrow}_{\mathcal{R},r,j} v$ (resp. $u \overset{*}{\leftrightarrow}_{\mathcal{R},p,j} v$). The congruence class $[w]_{\mathcal{R},r,j}$ (resp. $[w]_{\mathcal{R},p,j}$) of a word $w \in \Sigma^*$ is defined as $[w]_{\mathcal{R},r,j} := \{v \in \Sigma^* \mid w \overset{*}{\leftrightarrow}_{\mathcal{R},r,j} v\}$ (resp. $[w]_{\mathcal{R},p,j} := \{v \in \Sigma^* \mid w \overset{*}{\leftrightarrow}_{\mathcal{R},p,j} v\}$). The set $\{[w]_{\mathcal{R},r,j} \mid w \in \Sigma^*\}$ (resp. $\{[w]_{\mathcal{R},p,j} \mid w \in \Sigma^*\}$) of congruence classes w.r.t. $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is denoted by $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$).

▶ Remark 39. In Definition 38, one can alternatively define $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) only if $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is confluent so that $\overset{*}{\leftrightarrow}_{\mathcal{R},r,j}$ ($\overset{*}{\leftrightarrow}_{\mathcal{R},p,j}$) is the same as the right (resp. pure) conditional equational theory induced by $\mathcal{R}$ (see Lemmas 27 and 28). In this case, $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) can be directly associated with the right (resp. pure) conditional equational theory induced by $\mathcal{R}$. We use the simpler definition of $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) as in Definition 38 instead of adding the restriction that $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is confluent (cf. Theorem 42).

▶ **Lemma 40.**

(i) *The set $\mathcal{M}_{\mathcal{R},r,j}$ is a monoid under the operation $[u]_{\mathcal{R},r,j} \cdot [v]_{\mathcal{R},r,j} := [uv]_{\mathcal{R},r,j}$ with the identity element $[\varepsilon]_{\mathcal{R},r,j}$.* ☑

(ii) *The set $\mathcal{M}_{\mathcal{R},p,j}$ is a monoid under the operation $[u]_{\mathcal{R},p,j} \cdot [v]_{\mathcal{R},p,j} := [uv]_{\mathcal{R},p,j}$ with the identity element $[\varepsilon]_{\mathcal{R},p,j}$.* ☑

▶ **Definition 41.** Let $\mathcal{R}$ be a reductive right-join (resp. a pure-join) CSTS over Σ. The ordered pair $(\Sigma; \mathcal{R})$ is a (*reductive*) *conditional presentation* of the monoid $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$). The monoid $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) is *finitely presented* if both Σ and $\mathcal{R}$ are finite in $(\Sigma; \mathcal{R})$. The ordered pair $(\Sigma; \mathcal{R})$ is a *complete* (*reductive*) *conditional presentation* of the monoid $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) if $\rightarrow_{\mathcal{R},r,j}$ (resp. $\rightarrow_{\mathcal{R},p,j}$) is complete.

The following theorem and its proof provide a decision procedure for the word problem of monoids with finite complete (reductive) conditional presentations.

▶ **Theorem 42.**

(i) *Let $\mathcal{R}$ be a finite reductive right-join CSTS on Σ^*. If $\rightarrow_{\mathcal{R},r,j}$ is confluent, then we can decide whether s and t on Σ^* are the same element in the monoid $\mathcal{M}_{\mathcal{R},r,j} := \Sigma^* / \overset{*}{\leftrightarrow}_{\mathcal{R},r,j}$.* ☑

(ii) *Let $\mathcal{R}$ be a finite reductive pure-join CSTS on Σ^*. If $\rightarrow_{\mathcal{R},p,j}$ is confluent, then we can decide whether s and t on Σ^* are the same element in the monoid $\mathcal{M}_{\mathcal{R},p,j} := \Sigma^* / \overset{*}{\leftrightarrow}_{\mathcal{R},p,j}$.* ☑

Proof. For the proof of (i), if $s \overset{*}{\leftrightarrow}_{\mathcal{R},r,j} t$, then s and t are the same element in $\mathcal{M}_{\mathcal{R},r,j}$. Otherwise, s and t are the different elements in $\mathcal{M}_{\mathcal{R},r,j}$. Since $\mathcal{R}$ is finite and reductive, $\rightarrow_{\mathcal{R},r,j}$ is terminating and decidable. Furthermore, $\rightarrow_{\mathcal{R},r,j}$ is confluent by hypothesis, so we can decide whether $s \overset{*}{\leftrightarrow}_{\mathcal{R},r,j} t$ by comparing the normal forms of s and t w.r.t. $\rightarrow_{\mathcal{R},r,j}$. In other words, if the normal forms of s and t w.r.t. $\rightarrow_{\mathcal{R},r,j}$ are the same, then they are the same element in $\mathcal{M}_{\mathcal{R},r,j}$, and they are the different elements in $\mathcal{M}_{\mathcal{R},r,j}$, otherwise. We omit the proof of (ii) because it is similar to the proof of (i). ◀

Our formalization of Theorem 42 is an extension of the formalization of Theorem 10 discussed in Section 3 in conditional setting, where the locale monoid is instantiated using the different parameters for $\mathcal{M}_{\mathcal{R},r,j}$ and $\mathcal{M}_{\mathcal{R},p,j}$, respectively:

monoid "S/Thue_R_Congruence" "([·]$_r$)" "equiv_r.Class ε"*

monoid "S/Thue_P_Congruence" "([·]$_p$)" "equiv_p.Class ε"*

Above, *Thue_R_Congruence* and *Thue_P_Congruence* are (csr_r_join_step R)$^{\leftrightarrow^*}$ and (csr_p_join_step R)$^{\leftrightarrow^*}$, respectively. They correspond to the Thue congruence relations induced by $\rightarrow_{\mathcal{R},r,j}$ and $\rightarrow_{\mathcal{R},p,j}$, respectively. Also, $[\cdot]_r$ (resp. $[\cdot]_p$) represents an associative binary operator for the monoid $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$) in such a way that $[u]_{\mathcal{R},r,j}\,[\cdot]_r\,[v]_{\mathcal{R},r,j} = [uv]_{\mathcal{R},r,j}$ (resp. $[u]_{\mathcal{R},p,j}\,[\cdot]_p\,[v]_{\mathcal{R},p,j} = [uv]_{\mathcal{R},p,j}$) for all $u, v \in \Sigma^*$, where Σ is represented by S in the above. Finally, *equiv_r.Class* ε (resp. *equiv_p.Class* ε) represents the congruence class $[\varepsilon]_{\mathcal{R},r,j}$ (resp. $[\varepsilon]_{\mathcal{R},p,j}$) corresponding to the identity element in $\mathcal{M}_{\mathcal{R},r,j}$ (resp. $\mathcal{M}_{\mathcal{R},p,j}$).

5 Conclusion

Semi-Thue and certain types of conditional semi-Thue systems can be viewed as presentations of monoids because they define a quotient of the free monoid modulo the Thue congruence that they induce. We have presented an Isabelle/HOL formalization of semi-Thue and conditional semi-Thue systems along with a decision procedure for the word problem of monoids with finite complete unconditional or reductive conditional presentations. Also, our formalized framework for semi-Thue and Thue systems can provide formalized building blocks for different kinds applications because there is a wide variety of applications of semi-Thue and Thue systems, such as formal language theory [7, 38], group theory [10, 19, 33, 40], and algebraic protocols [6, 7].

The main contributions of this paper are as follows: (i) We have presented the first (computer-aided) formalization of conditional semi-Thue systems (using a proof assistant). In particular, we have provided a new formalized method for checking confluence using (conditional) critical pairs for finite reductive right-join and pure-join conditional semi-Thue systems. (ii) The existing classification of conditional semi-Thue systems basically consists only of left-right-join and right-join conditional semi-Thue systems (see [11]). We have extended this classification depending on how conditions are evaluated in the conditional parts of their rules, and provided a formalized framework for the different types of conditional semi-Thue systems. (For the classification of conditional term rewriting systems, see [46].) Furthermore, we have proposed and formalized a new inference system for generating different conditional equational theories and Thue congruences for the associated monoids along with their word problem. (iii) We have presented and formalized an inference system for a completion procedure of semi-Thue systems, which is adapted from the existing Knuth-completion procedure [7,24] of semi-Thue systems and an abstract completion procedure [2,18] of term rewriting systems. When attempting to construct a finite complete presentation of a monoid or a group, our inference-system based completion procedure is straightforward and easy to use (in comparison to algorithm-based completion procedures [7, 24]) by means of simple string matching. We have also provided the formalized proof of its correctness.

Yet, much work still remains ahead. Unlike semi-Thue systems, conditional semi-Thue systems have not been well researched. For example, effective/operation termination [32, 37], confluence, and the reachability analysis [16] of the different types of conditional semi-Thue systems along with their formalization are possible future research directions. Another possible future research direction is to classify and formalize the different types of conditional string rewriting in terms of the expressive power [34] of string rewriting.

References

1 Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, Cambridge, UK, 1998. doi:10.1017/CBO9781139172752.

2 Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for Equational Proofs. In *Proceedings of the Symposium on Logic in Computer Science (LICS '86), Cambridge, Massachusetts, USA, June 16-18, 1986*, pages 346–357. IEEE Computer Society, 1986.

3 Clemens Ballarin. Tutorial to Locales and Locale Interpretation, 2010. URL: http://isabelle.in.tum.de/doc/locales.pdf.

4 Clemens Ballarin. A Case Study in Basic Algebra. *Arch. Formal Proofs*, 2019. URL: https://www.isa-afp.org/entries/Jacobson_Basic_Algebra.html.

5 Jan A. Bergstra and Jan Willem Klop. Conditional Rewrite Rules: Confluence and Termination. *J. Comput. Syst. Sci.*, 32(3):323–362, 1986. doi:10.1016/0022-0000(86)90033-4.

6 Ronald V. Book and Friedrich Otto. On the Verifiability of Two-Party Algebraic Protocols. *Theor. Comput. Sci.*, 40:101–130, 1985. doi:10.1016/0304-3975(85)90161-6.

7 Ronald V. Book and Friedrich Otto. *String-Rewriting Systems*. Springer, New York, NY, 1993. doi:10.1007/978-1-4613-9771-7.

8 Thomas Braibant and Damien Pous. Tactics for Reasoning Modulo AC in Coq. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs - First International Conference, CPP 2011, Kenting, Taiwan, December 7-9, 2011. Proceedings*, volume 7086 of *Lecture Notes in Computer Science*, pages 167–182. Springer, 2011. doi:10.1007/978-3-642-25379-9_14.

9 Jacek Chrzaszcz and Daria Walukiewicz-Chrzaszcz. Towards Rewriting in Coq. In Hubert Comon-Lundh, Claude Kirchner, and Hélène Kirchner, editors, *Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, volume 4600 of *Lecture Notes in Computer Science*, pages 113–131. Springer, 2007. doi:10.1007/978-3-540-73147-4_6.

10 Daniel E. Cohen. String Rewriting – A Survey for Group Theorists. In Graham A. Niblo and Martin A. Roller, editors, *Geometric Group Theory*, London Mathematical Society Lecture Note Series, pages 37–47. Cambridge University Press, 1993.

11 Thomas Deiß. Conditional Semi-Thue systems for Presenting Monoids. Technical report, Universität Kaiserslautern, 1992. SEKI Report SR-92-12 available at https://agent.informatik.uni-kl.de/seki/1992/Deiss.SR-92-12.english.ps.Z. A shorter version of the paper was presented at Annual Symposium on Theoretical Aspects of Computer Science in 1992 (STACS 92) with the same title.

12 Nachum Dershowitz, Mitsuhiro Okada, and G. Sivakumar. Confluence of Conditional Rewrite Systems. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Conditional Term Rewriting Systems, 1st International Workshop, Orsay, France, July 8-10, 1987, Proceedings*, volume 308 of *LNCS*, pages 31–44. Springer, 1987. doi:10.1007/3-540-19242-5_3.

13 Nachum Dershowitz and David A. Plaisted. Rewriting. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 535–610. Elsevier and MIT Press, 2001. doi:10.1016/B978-044450813-3/50011-4.

14 Patrick Baxter Dragon, Oscar I. Hernandez, Joe Sawada, Aaron Williams, and Dennis Wong. Constructing de Bruijn sequences with co-lexicographic order: The k-ary Grandmama sequence. *Eur. J. Comb.*, 72:1–11, 2018. doi:10.1016/J.EJC.2018.03.006.

15 Hannah Fechtner. Braids in Lean. Master's thesis, Carnegie Mellon University, PA, USA, 2024.

16 Guillaume Feuillade, Thomas Genet, and Valérie Viet Triem Tong. Reachability Analysis over Term Rewriting Systems. *J. Autom. Reason.*, 33(3-4):341–383, 2004. doi:10.1007/S10817-004-6246-0.

17 Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq Library of Undecidable Problems. In *CoqPL 2020 The Sixth International Workshop*

on Coq for Programming Languages, New Orleans, USA, 2020. URL: https://hal.science/hal-02944217.

18 Nao Hirokawa, Aart Middeldorp, Christian Sternagel, and Sarah Winkler. Abstract Completion, Formalized. *Log. Methods Comput. Sci.*, 15(3), 2019. doi:10.23638/LMCS-15(3:19)2019.

19 D. F. Holt, B. Eick, and E. A. O'Brien. *Handbook of computational group theory*. CRC Press, Boca Raton, FL, 2005.

20 Jieh Hsiang, Hélène Kirchner, Pierre Lescanne, and Michaël Rusinowitch. The Term Rewriting Approach to Automated Theorem Proving. *J. Log. Program.*, 14(1&2):71–99, 1992. doi:10.1016/0743-1066(92)90047-7.

21 Nathan Jacobson. *Basic algebra I*. Dover Publications, Inc., Mineola, NY, 2nd edition, 2009.

22 Sam Kamin and Jean-Jacques Levy. Two generalizations of the recursive path ordering. Technical report, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1980.

23 Stéphane Kaplan. Conditional Rewrite Rules. *Theor. Comput. Sci.*, 33:175–193, 1984. doi:10.1016/0304-3975(84)90087-2.

24 Deepak Kapur and Paliath Narendran. The Knuth-Bendix Completion Procedure and Thue Systems. *SIAM J. Comput.*, 14(4):1052–1072, 1985. doi:10.1137/0214073.

25 Deepak Kapur and G. Sivakumar. RRL: theorem proving environment based on rewriting techniques. *SIGSOFT Softw. Eng. Notes*, 10(4):67–68, 1985. doi:10.1145/1012497.1012522.

26 Yutaka Kikuchi and Takuya Katayama. An Application of Term Rewriting Systems for Functional Programming. In Yutaka Ohno and Toshiko Matsuda, editors, *Distributed Environments: Software Paradigms and Workstations*, pages 79–106. Springer Japan, 1991. doi:10.1007/978-4-431-68144-1_7.

27 Dohan Kim. Equational theorem proving for clauses over strings. *Mathematical Structures in Computer Science*, 34(10):1055–1078, 2024. doi:10.1017/S0960129524000112.

28 Dohan Kim. Congruence Closure Modulo Groups. *Logical Methods in Computer Science*, Volume 21, Issue 1, 2025. doi:10.46298/lmcs-21(1:20)2025.

29 D. E. Knuth and P. B. Bendix. Simple Word Problems in Universal Algebras. In Jörg H. Siekmann and Graham Wrightson, editors, *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pages 342–376. Springer Berlin Heidelberg, 1983. doi:10.1007/978-3-642-81955-1.

30 Donald E. Knuth, James H. Morris Jr., and Vaughan R. Pratt. Fast Pattern Matching in Strings. *SIAM J. Comput.*, 6(2):323–350, 1977. doi:10.1137/0206024.

31 Pierre Lescanne. Computer experiments with the REVE term rewriting system generator. In *Proceedings of the 10th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '83, pages 99–108, 1983. doi:10.1145/567067.567078.

32 Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Inf. Process. Lett.*, 95(4):446–453, 2005. doi:10.1016/J.IPL.2005.05.002.

33 Klaus Madlener and Friedrich Otto. Using String-Rewriting for Solving the Word Problem for Finitely Presented Groups. *Inf. Process. Lett.*, 24(5):281–284, 1987. doi:10.1016/0020-0190(87)90149-9.

34 Massimo Marchiori. On the Expressive Power of Rewriting. In S. Ramesh and G. Sivakumar, editors, *Foundations of Software Technology and Theoretical Computer Science, 17th Conference, Kharagpur, India, December 18-20, 1997, Proceedings*, volume 1346 of *Lecture Notes in Computer Science*, pages 88–102. Springer, 1997. doi:10.1007/BFB0058025.

35 M. H. A. Newman. On Theories with a Combinatorial Definition of "Equivalence". *Annals of Mathematics*, 43(2):223–243, 1942. doi:10.2307/1968867.

36 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. doi:10.1007/3-540-45949-9.

37 Enno Ohlebusch. *Advanced topics in term rewriting*. Springer Science & Business Media, Berlin/Heidelberg, Germany, 2002.

38 Friedrich Otto. On the Connections between Rewriting and Formal Language Theory. In Paliath Narendran and Michaël Rusinowitch, editors, *Rewriting Techniques and Applications, 10th International Conference, RTA-99, Trento, Italy, July 2-4, 1999, Proceedings*, volume 1631 of *LNCS*, pages 332–355. Springer, 1999. doi:10.1007/3-540-48685-2_27.

39 James F Power. Thue's 1914 paper: a translation. *arXiv preprint arXiv:1308.5858*, 2013. doi:10.48550/arXiv.1308.5858.

40 C. C. Sims. *Computation with finitely presented groups*. Cambridge University Press, New York, NY, 1994.

41 Christian Sternagel and Thomas Sternagel. Certifying Confluence of Quasi-Decreasing Strongly Deterministic Conditional Term Rewrite Systems. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 413–431. Springer, 2017. doi:10.1007/978-3-319-63046-5_26.

42 Christian Sternagel and René Thiemann. Abstract Rewriting. *Arch. Formal Proofs*, 2010, 2010. URL: https://www.isa-afp.org/entries/Abstract-Rewriting.shtml.

43 Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

44 René Thiemann and Christian Sternagel. Certification of Termination Proofs Using CeTA. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Theorem Proving in Higher Order Logics, 22nd International Conference, TPHOLs 2009, Munich, Germany, August 17-20, 2009. Proceedings*, volume 5674 of *Lecture Notes in Computer Science*, pages 452–468. Springer, 2009. doi:10.1007/978-3-642-03359-9_31.

45 Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe. A unified ordering for termination proving. *Sci. Comput. Program.*, 111:110–134, 2015. doi:10.1016/J.SCICO.2014.07.009.

46 Toshiyuki Yamada, Jürgen Avenhaus, Carlos Loría-Sáenz, and Aart Middeldorp. Logicality of conditional rewrite systems. *Theor. Comput. Sci.*, 236(1-2):209–232, 2000. doi:10.1016/S0304-3975(99)00210-8.