Equational Theorem Proving for Clauses over Strings

Dohan Kim
A. I. Research Lab, Kyungwon Plaza 201, Sujeong-gu, Seongnam-si, Gyeonggi-do, South Korea
dkim@airesearch.kr

Although reasoning about equations over strings has been extensively studied for several decades, little research has been done for equational reasoning on general clauses over strings. This paper introduces a new superposition calculus with strings and present an equational theorem proving framework for clauses over strings. It provides a saturation procedure for clauses over strings and show that the proposed superposition calculus with contraction rules is refutationally complete. This paper also presents a new decision procedure for word problems over strings w.r.t. a set of conditional equations R over strings if R can be finitely saturated under the proposed inference system.

1 Introduction

Strings are fundamental objects in mathematics and many fields of science including computer science and biology. Reasoning about equations over strings has been widely studied in the context of string rewriting systems, formal language theory, word problems in semigroups, monoids and groups [8, 14], etc. Roughly speaking, reasoning about equations over strings replaces equals by equals w.r.t. a given reduction ordering $\succ$. For example, if we have two equations over strings $u_1u_2u_3 \approx s$ and $u_2 \approx t$ with $u_1u_2u_3 \succ s$ and $u_2 \succ t$, where u_2 is not the empty string, then we may infer the equation $u_1tu_3 \approx s$ by replacing u_2 in $u_1u_2u_3 \approx s$ with t. Meanwhile, if we have two equations over strings $u_1u_2 \approx s$ and $u_2u_3 \approx t$ with $u_1u_2 \succ s$ and $u_2u_3 \succ t$, where u_2 is not the empty string, then we should also be able to infer the equation $u_1t \approx su_3$. This can be done by concatenating u_3 to both sides of $u_1u_2 \approx s$ (i.e., $u_1u_2u_3 \approx su_3$) and then replacing u_2u_3 in $u_1u_2u_3 \approx su_3$ with t. Here, the *monotonicity property* of equations over strings is assumed, i.e., $s \approx t$ implies $usv \approx utv$ for strings s, t, u, and v.[1]

This reasoning about equations over strings is the basic ingredient for *completion* [8, 16] of string rewriting systems. A completion procedure [8, 16] attempts to construct a finite convergent string rewriting system, where a finite convergent string rewriting system provides a decision procedure for its corresponding equational theory.

Unlike reasoning about equations over strings, equational reasoning on general clauses over strings has not been well studied, where clauses are often the essential building blocks for logical statements.

This paper proposes a superposition calculus and an equational theorem proving procedure with clauses over strings. The results presented here generalize the results about completion of equations over strings [8, 16]. Throughout this paper, the monotonicity property of equations over strings is assumed and considered in the proposed inference rules. This assumption is natural and common to equations over strings occurring in algebraic structures (e.g., semigroups and monoids), formal language theory, etc. The *cancellation property* of equations over strings is not assumed, i.e., $su \approx tu$ implies $s \approx t$ for strings s, t, and a nonempty string u (cf. *non-cancellative* [8] algebraic structures).

Now, the proposed superposition inference rule is given roughly as follows:

[1]Note that it suffices to assume the right monotonicity property of equations over strings, i.e., $s \approx t$ implies $su \approx tu$ for strings s, t, and u, when finding overlaps between equations over strings under the monotonicity assumption.

P. Fontaine, D. Nantes-Sobrinho (Eds.):
Logical and Semantic Frameworks with Applications 2022 (LSFA'22)
EPTCS 376, 2023, pp. 49–66, doi:10.4204/EPTCS.376.6

Superposition: $\dfrac{C \vee u_1u_2 \approx s \qquad D \vee u_2u_3 \approx t}{C \vee D \vee u_1t \approx su_3}$

if u_2 is not the empty string, and $u_1u_2 \succ s$ and $u_2u_3 \succ t$.

Intuitively speaking, using the monotonicity property, $C \vee u_1u_2u_3 \approx su_3$ can be obtained from the left premise $C \vee u_1u_2 \approx s$. Then the above inference by Superposition can be viewed as an application of a conditional rewrite rule $D \vee u_2u_3 \approx t$ to $C \vee u_1u_2u_3 \approx su_3$, where u_2u_3 in $C \vee u_1u_2u_3 \approx su_3$ is now replaced by t, and D is appended to the conclusion. (Here, D can be viewed as consisting of the positive and negative conditions.) Note that both u_1 and u_3 can be the empty string in the Superposition inference rule. These steps are combined into a single Superposition inference step. For example, suppose that we have three clauses 1: $ab \approx d$, 2: $bc \approx e$, and 3: $ae \not\approx dc$. We use the Superposition inference rule with 1 and 2, and obtain 4: $ae \approx dc$ from which we derive a contradiction with 3. The details of the inference rules in the proposed inference system are discussed in Section 3.

The proposed superposition calculus is based on the simple string matching methods and the efficient length-lexicographic ordering instead of using equational unification and the more complex orderings, such as the lexicographic path ordering (LPO) [13] and Knuth-Bendix ordering (KBO) [2].

This paper shows that a clause over strings can be translated into a clause over first-order terms, which allows one to use the existing notion of redundancy in the literature [3, 22] for clauses over strings. Based on the notion of redundancy, one may delete redundant clauses using the contraction rules (i.e., Simplification, Subsumption, and Tautology) during an equational theorem proving derivation in order to reduce the search space for a refutation.

The *model construction techniques* [3, 22] is adapted for the refutational completeness of the proposed superposition calculus. This paper also uses a Herbrand interpretation by translating clauses over strings into clauses over first-order terms, where each nonground first-order clause represents all its ground instances. Note that this translation is not needed for the proposed inference system itself.

Finally, the proposed equational theorem proving framework with clauses over strings allows one to provide a new decision procedure for word problems over strings w.r.t. a conditional equational theory R if R can be finitely saturated under the proposed inference system.

2 Preliminaries

It is assumed that the reader has some familiarity with equational theorem proving [3, 22] and string rewriting systems [8, 16, 17]. The notion of conditional equations and Horn clauses are discussed in [12].

An *alphabet* Σ is a finite set of symbols (or letters). The set of all strings of symbols over Σ is denoted Σ^* with the empty string λ.

If $s \in \Sigma^*$, then the *length* of s, denoted $|s|$, is defined as follows: $|\lambda| := 0$, $|a| := 1$ for each $a \in \Sigma$, and $|sa| := |s| + 1$ for $s \in \Sigma^*$ and $a \in \Sigma$.

A *multiset* is an unordered collection with possible duplicate elements. We denote by $M(x)$ the number of occurrences of an object x in a multiset M.

An *equation* is an expression $s \approx t$, where s and t are strings, i.e., $s, t \in \Sigma^*$. A *literal* is either a positive equation L, called a *positive literal*, or a negative equation $\neg L$, called a *negative literal*. We also write a negative literal $\neg(s \approx t)$ as $s \not\approx t$. We identify a positive literal $s \approx t$ with the multiset $\{\{s\}, \{t\}\}$ and a negative literal $s \not\approx t$ with the multiset $\{\{s, t\}\}$. A *clause* (over Σ^*) is a finite multiset of literals, written as a disjunction of literals $\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n$ or as an implication $\Gamma \to \Delta$, where

$\Gamma = A_1 \wedge \cdots \wedge A_m$ and $\Delta = B_1 \vee \cdots \vee B_n$. We say that Γ is the *antecedent* and Δ is the *succedent* of clause $\Gamma \to \Delta$. A *Horn clause* is a clause with at most one positive literal. The *empty clause*, denoted $\square$, is the clause containing no literals.

A *conditional equation* is a clause of the form $(s_1 \approx t_1 \wedge \cdots \wedge s_n \approx t_n) \to l \approx r$. If $n = 0$, a conditional equation is simply an equation. A conditional equation is naturally represented by a Horn clause. A *conditional equational theory* is a set of conditional equations.

Any ordering $\succ_S$ on a set S can be extended to an ordering $\succ_S^{mul}$ on finite multisets over S as follows: $M \succ_S^{mul} N$ if (i) $M \neq N$ and (ii) whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some y such that $y \succ_S x$.

Given a multiset M and an ordering $\succ$ on M, we say that x is *maximal* (resp. *strictly maximal*) in M if there is no $y \in M$ (resp. $y \in M \setminus \{x\}$) with $y \succ x$ (resp. $y \succ x$ or $x = y$).

An ordering $>$ on Σ^* is *terminating* if there is no infinite chain of strings $s > s_1 > s_2 > \cdots$ for any $s \in \Sigma^*$. An ordering $>$ on Σ^* is *admissible* if $u > v$ implies $xuy > xvy$ for all $u, v, x, y \in \Sigma^*$. An ordering $>$ on Σ^* is a *reduction ordering* if it is terminating and admissible.

The *lexicographic ordering* $\succ_{lex}$ induced by a total precedence ordering $\succ_{prec}$ on Σ ranks strings of the same length in Σ^* by comparing the letters in the first index position where two strings differ using $\succ_{prec}$. For example, if $a = a_1a_2\cdots a_k$ and $b = b_1b_2\cdots b_k$, and the first index position where a and b are differ is i, then $a \succ_{lex} b$ if and only if $a_i \succ_{prec} b_i$.

The *length-lexicographic ordering* $\succ$ on Σ^* is defined as follows: $s \succ t$ if and only if $|s| > |t|$, or they have the same length and $s \succ_{lex} t$ for $s, t \in \Sigma^*$. If Σ and $\succ_{prec}$ are fixed, then it is easy to see that we can determine whether $s \succ t$ for two (finite) input strings $s \in \Sigma^*$ and $t \in \Sigma^*$ in $O(n)$ time, where $n = |s| + |t|$. The length-lexicographic ordering $\succ$ on Σ^* is a reduction ordering. We also write $\succ$ for a multiset extension of $\succ$ if it is clear from context.

We say that $\approx$ has the *monotonicity property* over Σ^* if $s \approx t$ implies $usv \approx utv$ for all $s, t, u, v \in \Sigma^*$. Throughout this paper, it is assumed that $\approx$ has the monotonicity property over Σ^*.

3 Superposition with Strings

3.1 Inference Rules

The following inference rules for clauses over strings are parameterized by a selection function $\mathscr{S}$ and the length-lexicographic ordering $\succ$, where $\mathscr{S}$ arbitrarily selects exactly one negative literal for each clause containing at least one negative literal (see Section 3.6 in [22] or Section 6 in [5]). In this strategy, an inference involving a clause with a selected literal is performed before an inference from clauses without a selected literal for a theorem proving process. The intuition behind the (eager) selection of negative literals is that, roughly speaking, one may first prove the whole antecedent of each clause from other clauses. Then clauses with no selected literals are involved in the main deduction process. This strategy is particularly useful when we consider Horn completion in Section 6 and a decision procedure for the word problems associated with it. In the following, the symbol $\bowtie$ is used to denote either $\approx$ or $\not\approx$.

Superposition: $\dfrac{C \vee u_1u_2 \approx s \qquad D \vee u_2u_3 \approx t}{C \vee D \vee u_1t \approx su_3}$

if (i) u_2 is not λ, (ii) C contains no selected literal, (iii) D contains no selected literal, (iv) $u_1u_2 \succ s$, and (v) $u_2u_3 \succ t$.[2]

[2]We do not require that $u_1u_2 \approx s$ (resp. $u_2u_3 \approx t$) is strictly maximal in the left premise (resp. the right premise) because of the assumption on the monotonicity property of equations over strings (see also Lemma 1 in Section 3.2).

Rewrite: $$\frac{C \vee u_1u_2u_3 \bowtie s \qquad D \vee u_2 \approx t}{C \vee D \vee u_1tu_3 \bowtie s}$$

if (i) $u_1u_2u_3 \bowtie s$ is selected for the left premise whenever $\bowtie$ is $\not\approx$, (ii) C contains no selected literal whenever $\bowtie$ is $\approx$, (iii) D contains no selected literal, and (iv) $u_2 \succ t$.[3]

Equality Resolution: $$\frac{C \vee s \not\approx s}{C}$$

if $s \not\approx s$ is selected for the premise.

The following Paramodulation and Factoring inference rules are used for non-Horn clauses containing positive literals only (cf. *Equality Factoring* [3,22] and *Merging Paramodulation* rule [3]).

Paramodulation: $$\frac{C \vee s \approx u_1u_2 \qquad D \vee u_2u_3 \approx t}{C \vee D \vee su_3 \approx u_1t}$$

if (i) u_2 is not λ, (ii) C contains no selected literal, (iii) C contains a positive literal, (iv) D contains no selected literal, (v) $s \succ u_1u_2$, and (vi) $u_2u_3 \succ t$.

Factoring: $$\frac{C \vee s \approx t \vee su \approx tu}{C \vee su \approx tu}$$

if C contains no selected literal.

In the proposed inference system, finding whether a string s occurs within a string t can be done in linear time in the size of s and t by using the existing string matching algorithms such as the Knuth-Morris-Pratt (KMP) algorithm [9]. For example, the KMP algorithm can be used for finding u_2 in $u_1u_2u_3$ in the Rewrite rule and finding u_2 in u_1u_2 in the Superposition and Paramodulation rule.

In the remainder of this paper, we denote by $\mathfrak{S}$ the inference system consisting of the Superposition, Rewrite, Equality Resolution, Paramodulation and the Factoring rule, and denote by S a set of clauses over strings. Also, by the *contraction rules* we mean the following inference rules–Simplification, Subsumption and Tautology.

Simplification: $$\frac{S \cup \{C \vee l_1ll_2 \bowtie v,\ l \approx r\}}{S \cup \{C \vee l_1rl_2 \bowtie v,\ l \approx r\}}$$

if (i) $l_1ll_2 \bowtie v$ is selected for $C \vee l_1ll_2 \bowtie v$ whenever $\bowtie$ is $\not\approx$, (ii) l_1 is not λ, and (iii) $l \succ r$.

In the following inference rule, we say that a clause C *subsumes* a clause C' if C is contained in C', where C and C' are viewed as the finite multisets.

Subsumption: $$\frac{S \cup \{C, C'\}}{S \cup \{C\}}$$

if $C \subseteq C'$.

Tautology: $$\frac{S \cup \{C \vee s \approx s\}}{S}$$

Example 1. Let $a \succ b \succ c \succ d \succ e$ and consider the following inconsistent set of clauses 1: $ad \approx b \vee ad \approx c$, 2: $b \approx c$, 3: $ad \approx e$, and 4: $c \not\approx e$. Now, we show how the empty clause is derived:

[3] Note that $u_2 \succ t$ implies that u_2 cannot be the empty string λ.

5: $ad \approx c \vee ad \approx c$ (Paramodulation of 1 with 2)
6: $ad \approx c$ (Factoring of 5)
7: $c \approx e$ (Rewrite of 6 with 3)
8: $e \not\approx e$ ($c \not\approx e$ is selected for 4. Rewrite of 4 with 7)
9: $\Box$ ($e \not\approx e$ is selected for 8. Equality Resolution on 8)

Note that there is no inference with the selected literal in 4 from the initial set of clauses 1, 2, 3, and 4. We produced clauses 5, 6, and 7 without using a selected literal. Once we have clause 7, there is an inference with the selected literal in 4.

Example 2. Let $a \succ b \succ c \succ d$ and consider the following inconsistent set of clauses 1: $aa \approx a \vee bd \not\approx a$, 2: $cd \approx b$, 3: $ad \approx c$, 4: $bd \approx a$, and 5: $dab \not\approx db$. Now, we show how the empty clause is derived:
6: $aa \approx a \vee a \not\approx a$ ($bd \not\approx a$ is selected for 1. Rewrite of 1 with 4)
7: $aa \approx a$ ($a \not\approx a$ is selected for 6. Equality resolution on 6)
8: $ac \approx ad$ (Superposition of 7 with 3)
9: $add \approx ab$ (Superposition of 8 with 2)
10: $ab \approx cd$ (Rewrite of 9 with 3)
11: $dcd \not\approx db$ ($dab \not\approx db$ is selected for 5. Rewrite of 5 with 10)
12: $db \not\approx db$ ($dcd \not\approx db$ is selected for 11. Rewrite of 11 with 2)
13: $\Box$ ($db \not\approx db$ is selected for 12. Equality Resolution on 12)

3.2 Lifting Properties

Recall that Σ^* is the set of all strings over Σ with the empty string λ. We let $T(\Sigma \cup \{\perp\})$ be the set of all first-order ground terms over $\Sigma \cup \{\perp\}$, where each letter from Σ is interpreted as a unary function symbol and $\perp$ is the only constant symbol. (The constant symbol $\perp$ does not have a special meaning (e.g., "false") in this paper.) We remove parentheses for notational convenience for each term in $T(\Sigma \cup \{\perp\})$. Since $\perp$ is the only constant symbol, we see that $\perp$ occurs only once at the end of each term in $T(\Sigma \cup \{\perp\})$. We may view each term in $T(\Sigma \cup \{\perp\})$ as a string ending with $\perp$. Now, the definitions used in Section 2 can be carried over to the case when Σ^* is replaced by $T(\Sigma \cup \{\perp\})$. In the remainder of this paper, we use the string notation for terms in $T(\Sigma \cup \{\perp\})$ unless otherwise stated.

Let $s \approx t$ be an equation over Σ^*. Then we can associate $s \approx t$ with the equation $s(x) \approx t(x)$, where $s(x) \approx t(x)$ represents the set of all its ground instances over $T(\Sigma \cup \{\perp\})$. (Here, $\lambda(x)$ and $\lambda\perp$ correspond to x and $\perp$, respectively.) First, $s \approx t$ over Σ^* corresponds to $s\perp \approx t\perp$ over $T(\Sigma \cup \{\perp\})$. Now, using the monotonicity property, if we concatenate string u to both sides of $s \approx t$ over Σ^*, then we have $su \approx tu$, which corresponds to $su\perp \approx tu\perp$.

There is a similar approach in string rewriting systems. If S is a string rewriting system over Σ^*, then it is known that we can associate term rewriting system R_S with S in such a way that $R_S := \{l(x) \to r(x) \,|\, l \to r \in S\}$ [8], where x is a variable and each letter from Σ is interpreted as a unary function symbol. We may rename variables (by standardizing variables apart) whenever necessary. This approach is particularly useful when we consider critical pairs between the rules in a string rewriting system. For example, if there are two rules $aa \to c$ and $ab \to d$ in S, then we have $cb \leftarrow aab \to ad$, where $<cb, ad>$ (or $<ad, cb>$) is a *critical pair* formed from these two rules. This critical pair can also be found if we associate $aa \to c \in S$ with $a(a(x)) \to c(x) \in R_S$ and $ab \to d \in S$ with $a(b(x)) \to d(x) \in R_S$. First, we rename the rule $a(b(x)) \to d(x) \in R_S$ into $a(b(y)) \to d(y)$. Then by mapping x to $b(z)$ and y to z, we have $c(b(z)) \leftarrow a(a(b(z))) \to a(d(z))$, where $<c(b(z)), a(d(z))>$ is a critical pair formed from these two rules. This critical pair can be associated with the critical pair $<cb, ad>$ formed from $aa \to c$ in S and

$ab \to d$ in S.

However, if $s \not\approx t$ is a negative literal over strings, then we cannot simply associate $s \not\approx t$ with the negative literal $s(x) \not\approx t(x)$ over first-order terms. Suppose to the contrary that we associate $s \not\approx t$ with $s(x) \not\approx t(x)$. Then $s \not\approx t$ implies $su \not\approx tu$ for a nonempty string u because we can substitute $u(y)$ for x in $s(x) \not\approx t(x)$, and $su \not\approx tu$ can also be associated with $s(u(y)) \not\approx t(u(y))$. Using the contrapositive argument, this means that $su \approx tu$ implies $s \approx t$ for the nonempty string u. Recall that we do not assume the cancellation property of equations over strings in this paper.[4] Instead, we simply associate $s \not\approx t$ with $s\bot \not\approx t\bot$. The following lemma is based on the above observations. We denote by $T(\Sigma \cup \{\bot\}, X)$ the set of first-order terms built on $\Sigma \cup \{\bot\}$ and a denumerable set of variables X, where each symbol from Σ is interpreted as a unary function symbol and $\bot$ is the only constant symbol.

Lemma 1. *Let $C := s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ be a clause over Σ^* and P be the set of all clauses that follow from C using the monotonicity property. Let Q be the set of all ground instances of the clause $s_1(x_1) \approx t_1(x_1) \vee \cdots \vee s_m(x_m) \approx t_m(x_m) \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$ over $T(\Sigma \cup \{\bot\}, X)$, where $x_1, \ldots, x_m$ are distinct variables in X and each letter from Σ is interpreted as a unary function symbol. Then there is a one-to-one correspondence between P and Q.*

Proof. For each element D of P, D has the form $D := s_1w_1 \approx t_1w_1 \vee \cdots \vee s_mw_m \approx t_mw_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ for some $w_1, \ldots, w_m \in \Sigma^*$. (If $w_i = \lambda$ for all $1 \le i \le m$, then D is simply C.) Now, we map each element D of P to D' in Q, where $D' := s_1w_1\bot \approx t_1w_1\bot \vee \cdots \vee s_mw_m\bot \approx t_mw_m\bot \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$. Since $\bot$ is the only constant symbol in $\Sigma \cup \{\bot\}$, it is easy to see that this mapping is well-defined and bijective. $\square$

Definition 2. (i) We say that every term in $T(\Sigma \cup \{\bot\})$ is a *g-term*. (Recall that we remove parentheses for notational convenience.)
(ii) Let $s \approx t$ (resp. $s \to t$) be an equation (resp. a rule) over Σ^*. We say that $su\bot \approx tu\bot$ (resp. $su\bot \to tu\bot$) for some string u is a *g-equation* (resp. a *g-rule*) of $s \approx t$ (resp. $s \to t$).
(iii) Let $s \not\approx t$ be a negative literal over Σ^*. We say that $s\bot \not\approx t\bot$ is a (negative) *g-literal* of $s \not\approx t$.
(iv) Let $C := s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ be a clause over Σ^*. We say that $s_1w_1\bot \approx t_1w_1\bot \vee \cdots \vee s_mw_m\bot \approx t_mw_m\bot \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$ for some strings $w_1, \ldots, w_m$ is a *g-clause* of clause C. Here, each $w_k\bot \in T(\Sigma \cup \{\bot\})$ for nonempty string w_k in the g-clause is said to be a *substitution part* of C.
(v) Let π be an inference (w.r.t. $\mathfrak{S}$) with premises $C_1, \ldots, C_k$ and conclusion D. Then a *g-instance* of π is an inference (w.r.t. $\mathfrak{S}$) with premises $C'_1, \ldots, C'_k$ and conclusion D', where $C'_1, \ldots, C'_k$ and D' are g-clauses of $C_1, \ldots, C_k$ and D, respectively.

Since each term in $T(\Sigma \cup \{\bot\})$ is viewed as a string, we may consider inferences between g-clauses using $\mathfrak{S}$. Note that concatenating a (nonempty) string at the end of a g-term is not allowed for any g-term over $T(\Sigma \cup \{\bot\})$. For example, $abc\bot d$ is not a g-term, and $a\bot \not\approx b\bot \vee abc\bot d \approx def\bot d$ is not a g-clause. We emphasize that we are only concerned with inferences between (legitimate) g-clauses here.

We may also use the length-lexicographic ordering $\succ_g$ on g-terms. Given a total precedence ordering on $\Sigma \cup \{\bot\}$ for which $\bot$ is minimal, it can be easily verified that $\succ_g$ is a total reduction ordering on $T(\Sigma \cup \{\bot\})$. We simply denote the multiset extension $\succ_g^{mul}$ of $\succ_g$ as $\succ_g$ for notational convenience. Similarly, we denote ambiguously all orderings on g-terms, g-equations, and g-clauses over $T(\Sigma \cup \{\bot\})$ by $\succ_g$. Now, we consider the lifting of inferences of $\mathfrak{S}$ between g-clauses over $T(\Sigma \cup \{\bot\})$ to inferences of $\mathfrak{S}$ between clauses over Σ^*. Let $C_1, \ldots, C_n$ be clauses over Σ^* and let

[4] One may assume the cancellation property and associate $s \not\approx t$ over strings with $s(x) \not\approx t(x)$ over first-order terms, which is beyond the scope of this paper.

$$\frac{C'_1 \ldots C'_n}{C'}$$

be an inference between their g-clauses, where C'_i is a g-clause of C_i for all $1 \leq i \leq n$. We say that this inference between g-clauses can be *lifted* if there is an inference

$$\frac{C_1 \ldots C_n}{C}$$

such that C' is a g-clause of C. In what follows, we assume that a g-literal L'_i in C'_i is selected in the same way as L_i in C_i, where L_i is a negative literal in C_i and L'_i is a g-literal of L_i.

Lifting of an inference between g-clauses is possible if it does not correspond to a g-instance of an inference (w.r.t. $\mathfrak{S}$) into a substitution part of a clause, which is not necessary (see [4, 22]). Suppose that there is an inference between g-clauses $C'_1 \ldots C'_n$ with conclusion C' and there is also an inference between clauses $C_1 \ldots C_n$ over Σ^* with conclusion C, where C'_i is a g-clause of C_i for all $1 \leq i \leq n$. Then, the inference between g-clauses $C'_1 \ldots C'_n$ over $T(\Sigma \cup \{\bot\})$ can be lifted to the inference between clauses $C_1 \ldots C_n$ over Σ^* in such a way that C' is a g-clause of C. This can be easily verified for each inference rule in $\mathfrak{S}$.

Example 3. Consider the following Superposition inference with g-clauses:

$$\frac{ad\bot \approx cd\bot \vee aabb\bot \approx cbb\bot \qquad abb\bot \approx db\bot}{ad\bot \approx cd\bot \vee adb\bot \approx cbb\bot}$$

where $ad\bot \approx cd\bot \vee aabb\bot \approx cbb\bot$ (resp. $abb\bot \approx db\bot$) is a g-clause of $a \approx c \vee aa \approx c$ (resp. $ab \approx d$) and $aabb\bot \succ_g cbb\bot$ (resp. $abb\bot \succ_g db\bot$). This Superposition inference between g-clauses can be lifted to the following Superposition inference between clauses over Σ^*:

$$\frac{a \approx c \vee aa \approx c \qquad ab \approx d}{a \approx c \vee ad \approx cb}$$

where $aa \succ c$ and $ab \succ d$. We see that conclusion $ad\bot \approx cd\bot \vee adb\bot \approx cbb\bot$ of the Superposition inference between the above g-clauses is a g-clause of conclusion $a \approx c \vee ad \approx cb$ of this inference.

Example 4. Consider the following Rewrite inference with g-clauses:

$$\frac{a\bot \not\approx d\bot \vee aabb\bot \not\approx cd\bot \qquad abb\bot \approx cb\bot}{a\bot \not\approx d\bot \vee acb\bot \not\approx cd\bot}$$

where $aabb\bot \not\approx cd\bot$ is selected and $a\bot \not\approx d\bot \vee aabb\bot \not\approx cd\bot$ (resp. $abb\bot \approx cb\bot$) is a g-clause of $a \not\approx d \vee aabb \not\approx cd$ (resp. $ab \approx c$) with $abb\bot \succ_g cb\bot$. This Rewrite inference between g-clauses can be lifted to the following Rewrite inference between clauses over Σ^*:

$$\frac{a \not\approx d \vee aabb \not\approx cd \qquad ab \approx c}{a \not\approx d \vee acb \not\approx cd}$$

where $aabb \not\approx cd$ is selected and $ab \succ c$. We see that conclusion $a\bot \not\approx d\bot \vee acb\bot \not\approx cd\bot$ of the Rewrite inference between the above g-clauses is a g-clause of conclusion $a \not\approx d \vee acb \not\approx cd$ of this inference.

4 Redundancy and Contraction Techniques

By Lemma 1 and Definition 2, we may translate a clause $C := s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ over Σ^* with all its implied clauses using the monotonicity property into the clause $s_1(x_1) \approx t_1(x_1) \vee \cdots \vee s_m(x_m) \approx t_m(x_m) \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$ over $T(\Sigma \cup \{\bot\}, X)$ with all its ground instances, where $x_1, \ldots, x_m$ are distinct variables in X, each symbol from Σ is interpreted as a unary function symbol, and $\bot$ is the only constant symbol. This allows us to adapt the existing notion of redundancy found in the literature [3, 22].

Definition 3. (i) Let R be a set of g-equations or g-rules. Then the congruence $\leftrightarrow_R^*$ defines an equality *Herbrand Interpretation* I, where the domain of I is $T(\Sigma \cup \{\perp\})$. Each unary function symbol $s \in \Sigma$ is interpreted as the unary function s_I, where $s_I(u\perp)$ is the g-term $su\perp$. (The constant symbol $\perp$ is simply interpreted as the constant $\perp$.) The only predicate $\approx$ is interpreted by $s\perp \approx t\perp$ if $s\perp \leftrightarrow_R^* t\perp$. We denote by R^* the interpretation I defined by R in this way. I *satisfies* (is a *model* of) a g-clause $\Gamma \to \Delta$, denoted by $I \models \Gamma \to \Delta$, if $I \not\supseteq \Gamma$ or $I \cap \Delta \neq \emptyset$. In this case, we say that $\Gamma \to \Delta$ is *true* in I. We say that I *satisfies* a clause C over Σ^* if I satisfies all g-clauses of C. We say that I *satisfies* a set of clauses S over Σ^*, denoted by $I \models S$, if I satisfies every clause in S.
(ii) A g-clause C *follows* from a set of g-clauses $\{C_1, \dots, C_n\}$, denoted by $\{C_1, \dots, C_n\} \models C$, if C is true in every model of $\{C_1, \dots, C_k\}$.

Definition 4. Let S be a set of clauses over Σ^*.
(i) A g-clause C is *redundant* w.r.t. S if there exist g-clauses $C'_1, \dots, C'_k$ of clauses $C_1, \dots, C_k$ in S, such that $\{C'_1, \dots, C'_k\} \models C$ and $C \succ_g C'_i$ for all $1 \leq i \leq k$. A clause in S is *redundant* w.r.t. S if all its g-clauses are redundant w.r.t. S.
(ii) An inference π with conclusion D is *redundant* w.r.t. S if for every g-instance of π with maximal premise C' (w.r.t. $\succ_g$) and conclusion D', there exist g-clauses $C'_1, \dots, C'_k$ of clauses $C_1, \dots, C_k$ in S such that $\{C'_1, \dots, C'_k\} \models D'$ and $C' \succ_g C'_i$ for all $1 \leq i \leq k$, where D' is a g-clause of D.

Lemma 5. *If an equation $l \approx r$ simplifies a clause $C \vee l_1 l l_2 \bowtie v$ into $C \vee l_1 r l_2 \bowtie v$ using the Simplification rule, then $C \vee l_1 l l_2 \bowtie v$ is redundant w.r.t. $\{C \vee l_1 r l_2 \bowtie v, l \approx r\}$.*

Proof. Suppose that $l \approx r$ simplifies $D := C \vee l_1 l l_2 \not\approx v$ into $C \vee l_1 r l_2 \not\approx v$, where $l_1 l l_2 \not\approx v$ is selected for D. Then, every g-clause D' of D has the form $D' := C' \vee l_1 l l_2 \perp \not\approx v\perp$, where C' is a g-clause of C. Now, we may infer that $\{D'', l l_2 \perp \approx r l_2 \perp\} \models D'$, where $D'' := C' \vee l_1 r l_2 \perp \not\approx v\perp$ is a g-clause of $C \vee l_1 r l_2 \not\approx v$ and $l l_2 \perp \approx r l_2 \perp$ is a g-equation of $l \approx r$. We also have $D' \succ_g D''$ and $D' \succ_g l l_2 \perp \approx r l_2 \perp$, and thus the conclusion follows.

Otherwise, suppose that $l \approx r$ simplifies $D := C \vee l_1 l l_2 \approx v$ into $C \vee l_1 r l_2 \approx v$. Then every g-clause D' of D has the form $D' := C' \vee l_1 l l_2 w \perp \approx v w \perp$ for some $w \in \Sigma^*$, where C' is a g-clause of C. Now, we have $\{D'', l l_2 w \perp \approx r l_2 w \perp\} \models D'$, where $D'' := C' \vee l_1 r l_2 w \perp \approx v w \perp$ is a g-clause of $C \vee l_1 r l_2 \approx v$ for some $w \in \Sigma^*$ and $l l_2 w \perp \approx r l_2 w \perp$ is a g-equation of $l \approx r$. We also have $D' \succ_g D''$ and $D' \succ_g l l_2 w \perp \approx r l_2 w \perp$ because l_1 is not λ in the condition of the rule (i.e., $l_1 l l_2 w \perp \succ_g l l_2 w \perp$), and thus the conclusion follows. □

We see that if C subsumes C' with C and C' containing the same number of literals, then they are the same when viewed as the finite multisets, so we can remove C'. Therefore, we exclude this case in the following lemma.

Lemma 6. *If a clause C subsumes a clause D and C contains fewer literals than D, then D is redundant w.r.t. $\{C\}$.*

Proof. Suppose that C subsumes D and C contains fewer literals than D. Then D can be denoted by $C \vee B$ for some nonempty clause B. Now, for every g-clause $D' := C' \vee B'$ of D, we have $\{C'\} \models D'$ with $D' \succ_g C'$, where C' and B' are g-clauses of C and B, respectively. Thus, D is redundant w.r.t. $\{C\}$. □

Lemma 7. *A tautology $C \vee s \approx s$ is redundant.*

Proof. It is easy to see that for every g-clause $C' \vee su\perp \approx su\perp$ of $C \vee s \approx s$, we have $\models C' \vee su\perp \approx su\perp$, where $u \in \Sigma^*$ and C' is a g-clause of C. Thus, $C \vee s \approx s$ is redundant. □

5 Refutational Completeness

In this section, we adapt the model construction and equational theorem proving techniques used in [3, 18, 22] and show that $\mathfrak{S}$ with the contraction rules is refutationally complete.

Definition 8. A g-equation $s\bot \approx t\bot$ is *reductive* for a g-clause $C := D \vee s\bot \approx t\bot$ if $s\bot \approx t\bot$ is strictly maximal (w.r.t. $\succ_g$) in C with $s\bot \succ_g t\bot$.

Definition 9. (Model Construction) Let S be a set of clauses over Σ^*. We use induction on $\succ_g$ to define the sets R_C, E_C, and I_C for all g-clauses C of clauses in S. Let C be such a g-clause of a clause in S and suppose that $E_{C'}$ has been defined for all g-clauses C' of clauses in S for which $C \succ_g C'$. Then we define by $R_C = \bigcup_{C \succ_g C'} E_{C'}$. We also define by I_C the equality interpretation R_C^*, which denotes the least congruence containing R_C.

Now, let $C := D \vee s\bot \approx t\bot$ such that C is not a g-clause of a clause with a selected literal in S. Then C produces $E_C = \{s\bot \to t\bot\}$ if the following conditions are met: (1) $I_C \not\models C$, (2) $I_C \not\models t\bot \approx t'\bot$ for every $s\bot \approx t'\bot$ in D, (3) $s\bot \approx t\bot$ is reductive for C, and (4) $s\bot$ is irreducible by R_C. We say that C is *productive* and produces E_C if it satisfies all of the above conditions. Otherwise, $E_C = \emptyset$. Finally, we define I_S as the equality interpretation R_S^*, where $R_S = \bigcup_C E_C$ is the set of all g-rules produced by g-clauses of clauses in S.

Lemma 10. *(i) R_S has the Church-Rosser property.*
(ii) R_S is terminating.
(iii) For g-terms $u\bot$ and $v\bot$, $I_S \models u\bot \approx v\bot$ if and only if $u\bot \downarrow_{R_S} v\bot$.
(iv) If $I_S \models s \approx t$, then $I_S \models usv \approx utv$ for nonempty strings $s, t, u, v \in \Sigma^*$.

Proof. (i) R_S is left-reduced because there are no overlaps among the left-hand sides of rewrite rules in R_S, and thus R_S has the Church-Rosser property.
(ii) For each rewrite rule $l\bot \to r\bot$ in R_S, we have $l\bot \succ_g r\bot$, and thus R_S is terminating.
(iii) Since R_S has the Church-Rosser property and is terminating by (i) and (ii), respectively, R_S is convergent. Thus, $I_S \models u\bot \approx v\bot$ if and only if $u\bot \downarrow_{R_S} v\bot$ for g-terms $u\bot$ and $v\bot$.
(iv) Suppose that $I_S \models s \approx t$ for nonempty strings s and t. Then, we have $I_S \models svw\bot \approx tvw\bot$ for all strings v and w by Definition 3(i). Similarly, since I_S is an equality Herbrand interpretation, we also have $I_S \models usvw\bot \approx utvw\bot$ for all strings u, which means that $I_S \models usv \approx utv$ by Definition 3(i). □

Lemma 10(iv) says that the monotonicity assumption used in this paper holds w.r.t. a model constructed by Definition 9.

Definition 11. Let S be a set of clauses over Σ^*. We say that S is *saturated* under $\mathfrak{S}$ if every inference by $\mathfrak{S}$ with premises in S is redundant w.r.t. S.

Definition 12. Let $C := s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ be a clause over Σ^*, and $C' = s_1w_1\bot \approx t_1w_1\bot \vee \cdots \vee s_mw_m\bot \approx t_mw_m\bot \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$ for some strings $w_1, \ldots, w_m$ be a g-clause of C. We say that C' is a *reduced g-clause* of C w.r.t. a rewrite system R if every $w_i\bot$, $1 \le i \le m$, is not reducible by R.

In the proof of the following lemma, we write $s[t]_{suf}$ to indicate that t occurs in s as a suffix and (ambiguously) denote by $s[u]_{suf}$ the result of replacing the occurrence of t (as a suffix of s) by u.

Lemma 13. *Let S be saturated under $\mathfrak{S}$ not containing the empty clause and C be a g-clause of a clause in S. Then C is true in I_S. More specifically,*
(i) If C is redundant w.r.t. S, then it is true in I_S.

(ii) If C is not a reduced g-clause of a clause in S w.r.t. R_S, then it is true in I_S.
(iii) If $C := C' \vee s\bot \approx t\bot$ produces the rule $s\bot \to t\bot$, then C' is false and C is true in I_S.
(iv) If C is a g-clause of a clause in S with a selected literal, then it is true in I_S.
(v) If C is non-productive, then it is true in I_S.

Proof. We use induction on $\succ_g$ and assume that (i)–(v) hold for every g-clause D of a clause in S with $C \succ_g D$.

(i) Suppose that C is redundant w.r.t. S. Then there exist g-clauses $C'_1, \ldots, C'_k$ of clauses $C_1, \ldots, C_k$ in S, such that $\{C'_1, \ldots, C'_k\} \models C$ and $C \succ_g C'_i$ for all $1 \le i \le k$. By the induction hypothesis, each C'_i, $1 \le i \le k$, is true in I_S. Thus, C is true in I_S.

(ii) Suppose that C is a g-clause of a clause $B := s_1 \approx t_1 \vee \cdots \vee s_m \approx t_m \vee u_1 \not\approx v_1 \vee \cdots \vee u_n \not\approx v_n$ in S but is not a reduced g-clause w.r.t. R_S. Then C is of the form $C := s_1w_1\bot \approx t_1w_1\bot \vee \cdots \vee s_mw_m\bot \approx t_mw_m\bot \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$ for $w_1, \ldots, w_m \in \Sigma^*$ and some $w_k\bot$ is reducible by R_S. Now, consider $C' = s_1w'_1\bot \approx t_1w'_1\bot \vee \cdots \vee s_mw'_m\bot \approx t_mw'_m\bot \vee u_1\bot \not\approx v_1\bot \vee \cdots \vee u_n\bot \not\approx v_n\bot$, where $w'_i\bot$ is the normal form of $w_i\bot$ w.r.t. R_S for each $1 \le i \le m$. Then C' is a reduced g-clause of B w.r.t. R_S, and is true in I_S by the induction hypothesis. Since each $w_i\bot \approx w'_i\bot$, $1 \le i \le m$, is true in I_S by Lemma 10(iii), we may infer that C is true in I_S.

In the remainder of the proof of this lemma, we assume that C is neither redundant w.r.t. S nor is it a reducible g-clause w.r.t. R_S of some clause in S. (Otherwise, we are done by (i) or (ii).)

(iii) Suppose that $C := C' \vee s\bot \approx t\bot$ produces the rule $s\bot \to t\bot$. Since $s\bot \to t\bot \in E_C \subset R_S$, we see that C is true in I_S. We show that C' is false in I_S. Let $C' := \Gamma \to \Delta$. Then $I_C \not\models C'$ by Definition 9, which implies that $I_C \cap \Delta = \emptyset, I_C \supseteq \Gamma$, and thus $I_S \supseteq \Gamma$. It remains to show that $I_S \cap \Delta = \emptyset$. Suppose to the contrary that Δ contains an equation $s'\bot \approx t'\bot$ which is true in I_S. Since $I_C \cap \Delta = \emptyset$, we must have $s'\bot \approx t'\bot \in I \setminus I_C$, which is only possible if $s\bot = s'\bot$ and $I_C \models t\bot \approx t'\bot$, contradicting condition (2) in Definition 9.

(iv) Suppose that C is of the form $C := B' \vee s\bot \not\approx t\bot$, where $s\bot \not\approx t\bot$ is a g-literal of a selected literal in a clause in S and B' is a g-clause of B.

(iv.1) If $s\bot = t\bot$, then B' is an equality resolvent of C and the Equality Resolution inferences can be lifted. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, B' is true in I_S. Thus, C is true in I_S.

(iv.2) If $s\bot \neq t\bot$, then suppose to the contrary that C is false in I_S. Then we have $I_S \models s\bot \approx t\bot$, which implies that $s\bot$ or $t\bot$ is reducible by R_S by Lemma 10(iii). Without loss of generality, we assume that $s\bot$ is reducible by R_S with some rule $lu\bot \to ru\bot$ for some $u \in \Sigma^*$ produced by a productive g-clause $D' \vee lu\bot \approx ru\bot$ of a clause $D \vee l \approx r \in S$. This means that $s\bot$ has a suffix $lu\bot$. Now, consider the following inference by Rewriting:

$$\frac{B \vee s[lu]_{suf} \not\approx t \qquad D \vee l \approx r}{B \vee D \vee s[ru]_{suf} \not\approx t}$$

where $s[lu]_{suf} \not\approx t$ is selected for the left premise. The conclusion of the above inference has a g-clause $C' := B' \vee D' \vee s\bot[ru\bot]_{suf} \not\approx t\bot$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C' must be true in I_S. Moreover, we see that $s\bot[ru\bot]_{suf} \not\approx t\bot$ is false in I_S by Lemma 10(iii), and D' are false in I_S by (iii). This means that B' is true in I_S, and thus C (i.e., $C = B' \vee s\bot \not\approx t\bot$) is true in I_S, which is the required contradiction.

(v) If C is non-productive, then we assume that C is not a g-clause of a clause with a selected literal. Otherwise, the proof is done by (iv). This means that C is of the form $C := B' \vee su\bot \approx tu\bot$, where $su\bot \approx tu\bot$ is maximal in C and B' contains no selected literal. If $su\bot = tu\bot$, then we are done. Therefore, without loss of generality, we assume that $su\bot \succ_g tu\bot$. As C is non-productive, it means that (at least) one of the conditions in Definition 9 does not hold.

If condition (1) does not hold, then $I_C \models C$, so we have $I_S \models C$, i.e., C is true in I_S. If condition (1) holds but condition (2) does not hold, then C is of the form $C := B'_1 \vee su\bot \approx tu\bot \vee svw\bot \approx t'vw\bot$, where $su = svw$ (i.e., $u = vw$) and $I_C \models tu\bot \approx t'vw\bot$.

Suppose first that $tu\bot = t'vw\bot$. Then we have $t = t'$ since $u = vw$. Now, consider the following inference by Factoring:

$$\frac{B_1 \vee s \approx t \vee sv \approx tv}{B_1 \vee sv \approx tv}$$

The conclusion of the above inference has a g-clause $C' := B'_1 \vee svw\bot \approx tvw\bot$, i.e., $C' := B'_1 \vee su\bot \approx tu\bot$ since $u = vw$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C' is true in I_S, and thus C is true in I_S.

Otherwise, suppose that $tu\bot \neq t'vw\bot$. Then we have $tu\bot \downarrow_{R_C} t'vw\bot$ by Lemma 10(iii) and $tu\bot \succ_g t'vw\bot$ because $su\bot \approx tu\bot$ is maximal in C. This means that $tu\bot$ is reducible by R_C by some rule $l\tau\bot \to r\tau\bot$ produced by a productive g-clause $D' \vee l\tau\bot \approx r\tau\bot$ of a clause $D \vee l \approx r \in S$. Now, we need to consider two cases:

(v.1) If t has the form $t := u_1u_2$ and l has the form $l := u_2u_3$, then consider the following inference by Paramodulation:

$$\frac{B \vee s \approx u_1u_2 \qquad D \vee u_2u_3 \approx r}{B \vee D \vee su_3 \approx u_1r}$$

The conclusion of the above inference has a g-clause $C' := B' \vee D' \vee su_3\tau\bot \approx u_1r\tau\bot$ with $u = u_3\tau$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C' is true in I_S. Since D' is false in I_S by (iii), either B' or $su_3\tau\bot \approx u_1r\tau\bot$ is true in I_S. If B' is true in I_S, so is C. If $su_3\tau\bot \approx u_1r\tau\bot$ is true in I_S, then $su\bot \approx tu\bot$ is also true in I_S by Lemma 10(iii), where $t = u_1u_2$ and $u = u_3\tau$. Thus, C is true in I_S.

(v.2) If t has the form $t := u_1u_2u_3$ and l has the form $l := u_2$, then consider the following inference by Rewrite:

$$\frac{B \vee s \approx u_1u_2u_3 \qquad D \vee u_2 \approx r}{B \vee D \vee s \approx u_1ru_3}$$

The conclusion of the above inference has a g-clause $C'' := B' \vee D' \vee su\bot \approx u_1ru_3u\bot$ with $\tau = u_3u$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C'' is true in I_S. Since D' is false in I_S by (iii), either B' or $su\bot \approx u_1ru_3u\bot$ is true in I_S. Similarly to case (v.1), if B' is true in I_S, so is C. If $su\bot \approx u_1ru_3u\bot$ is true in I_S, then $su\bot \approx tu\bot$ is also true in I_S by Lemma 10(iii), where $t = u_1u_2u_3$. Thus, C is true in I_S.

If conditions (1) and (2) hold but condition (3) does not hold, then $su\bot \approx tu\bot$ is only maximal but is not strictly maximal, so we are in the previous case. (Since $\succ_g$ is total on g-clauses, condition (2) does not hold.) If conditions (1)–(3) hold but condition (4) does not hold, then $su\bot$ is reducible by R_C by some rule $l\tau\bot \to r\tau\bot$ produced by a productive g clause $D' \vee l\tau\bot \approx r\tau\bot$ of a clause $D \vee l \approx r \in S$. Again, we need to consider two cases:

(v.1') If s has the form $s := u_1u_2$ and l has the form $l := u_2u_3$, then consider the following inference by Superposition:

$$\frac{B \vee u_1u_2 \approx t \qquad D \vee u_2u_3 \approx r}{B \vee D \vee u_1r \approx tu_3}$$

The conclusion of the above inference has a g-clause $C' := B' \vee D' \vee u_1 r\tau\bot \approx tu_3\tau\bot$ with $u = u_3\tau$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C' is true in I_S. Since D' is false in I_S by (iii), either B' or $u_1 r\tau\bot \approx tu_3\tau\bot$ is true in I_S. If B' is true in I_S, so is C. If $u_1 r\tau\bot \approx tu_3\tau\bot$ is true in I_S, then $su\bot \approx tu\bot$ is also true in I_S by Lemma 10(iii), where $s = u_1u_2$ and $u = u_3\tau$. Thus, C is true in I_S.

(v.2') If s has the form $s := u_1u_2u_3$ and l has the form $l := u_2$, then consider the following inference by Rewrite:

$$\frac{B \vee u_1u_2u_3 \approx t \qquad D \vee u_2 \approx r}{B \vee D \vee u_1 r u_3 \approx t}$$

The conclusion of the above inference has a g-clause $C'' := B' \vee D' \vee u_1 r u_3 u\bot \approx tu\bot$ with $\tau = u_3u$. By saturation of S under $\mathfrak{S}$ and the induction hypothesis, C'' is true in I_S. Since D' is false in I_S by (iii), either B' or $u_1 r u_3 u\bot \approx tu\bot$ is true in I_S. Similarly to case (v.1'), If B' is true in I_S, so is C. If $u_1 r u_3 u\bot \approx tu\bot$ is true in I_S, then $su\bot \approx tu\bot$ is also true in I_S by Lemma 10(iii), where $s = u_1u_2u_3$. Thus, C is true in I_S. □

Definition 14. (i) A *theorem proving derivation* is a sequence of sets of clauses $S_0 = S, S_1, \ldots$ over Σ^* such that:

(i.1) Deduction: $S_i = S_{i-1} \cup \{C\}$ if C can be deduced from premises in S_{i-1} by applying an inference rule in $\mathfrak{S}$.

(i.2) Deletion: $S_i = S_{i-1} \setminus \{D\}$ if D is redundant w.r.t. S_{i-1}.[5]

(ii) The set $S_\infty := \bigcup_i (\bigcap_{j \geq i} S_j)$ is the *limit* of the theorem proving derivation.

We see that the soundness of a theorem proving derivation w.r.t. the proposed inference system is straightforward, i.e., $S_i \models S_{i+1}$ for all $i \geq 0$.

Definition 15. A theorem proving derivation $S_0, S_1, S_2, \ldots$ is *fair* w.r.t. the inference system $\mathfrak{S}$ if every inference by $\mathfrak{S}$ with premises in S_∞ is redundant w.r.t. $\bigcup_j S_j$.

Lemma 16. *Let S and S' be sets of clauses over Σ^*.*
(i) If $S \subseteq S'$, then any clause which is redundant w.r.t. S is also redundant w.r.t. S'.
(ii) If $S \subseteq S'$ and all clauses in $S' \setminus S$ are redundant w.r.t. S', then any clause or inference which is redundant w.r.t. S' is also redundant w.r.t. S.

Proof. The proof of part (i) is obvious. For part (ii), suppose that a clause C is redundant w.r.t. S' and let C' be a g-clause of it. Then there exists a minimal set $N := \{C'_1, \ldots, C'_n\}$ (w.r.t. $\succ_g$) of g-clauses of clauses in S' such that $N \models C'$ and $C' \succ_g C'_i$ for all $1 \leq i \leq n$. We claim that all C'_i in N are not redundant w.r.t. S', which shows that C' is redundant w.r.t. S. Suppose to the contrary that some C'_j is redundant w.r.t. S'. Then there exist a set $N' := \{D'_1, \ldots, D'_m\}$ of g-clauses of clauses in S' such that $N' \models C'_j$ and $C'_j \succ_g D'_i$ for all $1 \leq i \leq m$. This means that we have $\{C'_1, \ldots, C'_{j-1}, D'_1, \ldots, D'_m, C'_{j+1}, \ldots, C'_n\} \models C'$, which contradicts our minimal choice of the set $N = \{C'_1, \ldots, C'_n\}$.

Next, suppose an inference π with conclusion D is redundant w.r.t. S' and let π' be a g-instance of it such that B is the maximal premise and D' is the conclusion of π' (i.e., a g-clause of D). Then there exists a minimal set $P := \{D'_1, \ldots, D'_n\}$ (w.r.t. $\succ_g$) of g-clauses of clauses in S' such that $P \models D'$ and $B \succ_g D'_i$ for all $1 \leq i \leq n$. As above, we may infer that all D'_i in P are not redundant w.r.t. S', and thus π' is redundant w.r.t. S. □

Lemma 17. *Let $S_0, S_1, \ldots$ be a fair theorem proving derivation w.r.t. $\mathfrak{S}$. Then S_∞ is saturated under $\mathfrak{S}$.*

[5] Here, an inference by Simplification combines the Deduction step for $C \vee l_1 r l_2 \bowtie v$ and the Deletion step for $C \vee l_1 l l_2 \bowtie v$ (see the Simplification rule).

Proof. If S_∞ contains the empty clause, then it is obvious that S_∞ is saturated under $\mathfrak{S}$. Therefore, we assume that the empty clause is not in S_∞.

If a clause C is deleted in a theorem proving derivation, then C is redundant w.r.t. some S_j. By Lemma 16(i), it is also redundant w.r.t. $\bigcup_j S_j$. Similarly, every clause in $\bigcup_j S_j \setminus S_\infty$ is redundant w.r.t. $\bigcup_j S_j$.

By fairness, every inference π by $\mathfrak{S}$ with premises in S_∞ is redundant w.r.t. $\bigcup_j S_j$. Using Lemma 16(ii) and the above, π is also redundant w.r.t. S_∞, which means that S_∞ is saturated under $\mathfrak{S}$. ☐

Theorem 18. *Let $S_0, S_1, \ldots$ be a fair theorem proving derivation w.r.t. $\mathfrak{S}$. If S_∞ does not contain the empty clause, then $I_{S_\infty} \models S_0$ (i.e., S_0 is satisfiable.)*

Proof. Suppose that $S_0, S_1, \ldots$ is a fair theorem proving derivation w.r.t. $\mathfrak{S}$ and that its limit S_∞ does not contain the empty clause. Then S_∞ is saturated under $\mathfrak{S}$ by Lemma 17. Let C' be a g-clause of a clause C in S_0. If $C \in S_\infty$, then C' is true in I_{S_∞} by Lemma 13. Otherwise, if $C \notin S_\infty$, then C is redundant w.r.t. some S_j. It follows that C redundant w.r.t. $\bigcup_j S_j$ by Lemma 16(i), and thus redundant w.r.t. S_∞ by Lemma 16(ii). This means that there exist g-clauses $C'_1, \ldots, C'_k$ of clauses $C_1, \ldots, C_k$ in S_∞ such that $\{C'_1, \ldots, C'_k\} \models C'$ and $C' \succ_g C'_i$ for all $1 \leq i \leq k$. Since each C'_i, $1 \leq i \leq k$, is true in I_{S_∞} by Lemma 13, C' is also true in I_{S_∞}, and thus the conclusion follows. ☐

The following theorem states that $\mathfrak{S}$ with the contraction rules is refutationally complete for clauses over Σ^*.

Theorem 19. *Let $S_0, S_1, \ldots$ be a fair theorem proving derivation w.r.t. $\mathfrak{S}$. Then S_0 is unsatisfiable if and only if the empty clause is in some S_j.*

Proof. Suppose that $S_0, S_1, \ldots$ be a fair theorem proving derivation w.r.t. $\mathfrak{S}$. By the soundness of the derivation, if the empty clause is in some S_j, then S_0 is unsatisfiable. Otherwise, if the empty clause is not in S_k for all k, then S_∞ does not contain the empty clause by the soundness of the derivation. Applying Theorem 18, we conclude that S_0 is satisfiable. ☐

6 Conditional Completion

In this section, we present a saturation procedure under $\mathfrak{S}$ for a set of conditional equations over Σ^*, where a conditional equation is naturally written as an equational Horn clause. A saturation procedure under $\mathfrak{S}$ can be viewed as *conditional completion* [12] for a set of conditional equations over Σ^*. If a set of conditional equations over Σ^* is simply a set of equations over Σ^*, then the proposed saturation procedure (w.r.t. $\succ$) corresponds to a completion procedure for a string rewriting system. Conditional string rewriting systems were considered in [11] in the context of embedding a finitely generated monoid with decidable word problem into a monoid presented by a finite convergent conditional presentation. It neither discusses a conditional completion (or a saturation) procedure, nor considers the word problems for conditional equations over Σ^* in general.

First, it is easy to see that a set of equations over Σ^* is consistent. Similarly, a set of conditional equations R over Σ^* is consistent because each conditional equation has always a positive literal and we cannot derive the empty clause from R using a saturation procedure under $\mathfrak{S}$ that is refutationally complete (cf. Section 9 in [13]). Since we only consider Horn clauses in this section, we neither need to consider the Factoring rule nor the Paramodulation rule in $\mathfrak{S}$. In the remainder of this section, by a conditional equational theory R, we mean a set of conditional equations R over Σ^*.

Definition 20. Given a conditional equational theory R and two finite words $s,t \in \Sigma^*$, a *word problem* w.r.t. R is of the form $\phi := s \approx^? t$. The *goal* of this word problem is $s \not\approx t$. We say that a word problem $s \approx^? t$ w.r.t. R is *decidable* if there is a decision procedure for determining whether $s \approx t$ is entailed by R (i.e., $R \models s \approx t$) or not (i.e., $R \not\models s \approx t$) .

Given a conditional equational theory R, let $G := s \not\approx t$ be the goal of a word problem $s \approx^? t$ w.r.t. R. (Note that G does not have any positive literal.) Then we see that $R \cup \{s \approx t\}$ is consistent if and only if $R \cup \{G\}$ is inconsistent. This allows one to decide a word problem w.r.t. R using the equational theorem proving procedure discussed in Section 5.

Lemma 21. *Let R be a conditional equational theory finitely saturated under $\mathfrak{S}$. Then Rewrite together with Equality Resolution is terminating and refutationally complete for $R \cup \{G\}$, where G is the goal of a word problem w.r.t. R.*

Proof. Since R is already saturated under $\mathfrak{S}$, inferences among Horn clauses in R are redundant and remain redundant in $R \cup \{G\}$ for a theorem proving derivation starting with $R \cup \{G\}$. (Here, $\{G\}$ can be viewed as a *set of support* [3] for a refutation of $R \cup \{G\}$.) Now, observe that G is a negative literal, so it should be selected. The only inference rules in $\mathfrak{S}$ involving a selected literal are the Rewrite and Equality Resolution rule. Furthermore, the derived literals from G w.r.t. Rewrite will also be selected eventually. Therefore, it suffices to consider positive literals as the right premise (because they contain no selected literal), and G and its derived literals w.r.t. Rewrite as the left premise for the Rewrite rule. Observe also that if G' is an immediate derived literal from G w.r.t. Rewrite, then we see that $G \succ G'$. If G or its derived literal from G w.r.t. Rewrite becomes of the form $u \not\approx u$ for some $u \in \Sigma^*$, then it will also be selected and an Equality Resolution inference yields the empty clause. Since $\succ$ is terminating and there are only finitely many positive literals in R, we may infer that the Rewrite and Equality Resolution inference steps on G and its derived literals are terminating. (The number of positive literals in R remains the same during a theorem proving derivation starting with $R \cup \{G\}$ using our selection strategy.)

Finally, since $\mathfrak{S}$ is refutationally complete by Thereom 19, Rewrite together with Equality Resolution is also refutationally complete for $R \cup \{G\}$. □

Given a finitely saturated conditional equational theory R under $\mathfrak{S}$, we provide a decision procedure for the word problems w.r.t. R in the following theorem.

Theorem 22. *Let R be a conditional equational theory finitely saturated under $\mathfrak{S}$. Then the word problems w.r.t. R are decidable by Rewrite together with Equality Resolution.*

Proof. Let $\phi := s \approx^? t$ be a word problem w.r.t. R and G be the goal of ϕ. We know that by Lemma 21, Rewrite together with Equality Resolution is terminating and refutationally complete for $R \cup \{G\}$. Let $R_0 := R \cup \{G\}, R_1, \ldots, R_n$ be a fair theorem proving derivation w.r.t. Rewrite together with Equality Resolution such that R_n is the limit of this derivation. If R_n contains the empty clause, then R_n is inconsistent, and thus R_0 is inconsistent, i.e., $\{s \not\approx t\} \cup R$ is inconsistent by the soundness of the derivation. Since R is consistent and $\{s \not\approx t\} \cup R$ is saturated under $\mathfrak{S}$, we may infer that $R \models s \approx t$.

Otherwise, if R_n does not contain the empty clause, then R_n is consistent, and thus R_0 is consistent by Theorem 19, i.e., $\{s \not\approx t\} \cup R$ is consistent. Since R is consistent and $\{s \not\approx t\} \cup R$ is saturated under $\mathfrak{S}$, we may infer that $R \not\models s \approx t$. □

The following corollary is a consequence of Theorem 22 and the following observation. Let $R = R_0, R_1, \ldots, R_n$ be a finite fair theorem proving derivation w.r.t. $\mathfrak{S}$ for an initial conditional equational theory R with the limit $\bar{R} := R_n$. Then $R \cup \{G\}$ is inconsistent if and only if $\bar{R} \cup \{G\}$ is inconsistent by the soundness of the derivation and Theorem 19.

Corollary 23. *Let $R = R_0, R_1, \ldots$ be a fair theorem proving derivation w.r.t. $\mathfrak{S}$ for a conditional equational theory R. If R can be finitely saturated under $\mathfrak{S}$, then the word problems w.r.t. R are decidable.*

Example 5. Let $a \succ b \succ c$ and R be a conditional equational theory consisting of the following rules 1: $aa \approx \lambda$, 2: $bb \approx \lambda$, 3: $ab \approx \lambda$, 4: $ab \not\approx ba \vee ac \approx ca$, and 5: $ab \not\approx ba \vee ac \not\approx ca \vee bc \approx cb$. We first saturate R under $\mathfrak{S}$:

6: $\lambda \not\approx ba \vee ac \approx ca$ ($ab \not\approx ba$ is selected for 4. Rewrite of 4 with 3)
7: $\lambda \not\approx ba \vee ac \not\approx ca \vee bc \approx cb$ ($ab \not\approx ba$ is selected for 5. Rewrite of 5 with 3)
8: $a \approx b$ (Superposition of 1 with 3)
9: $\lambda \not\approx bb \vee ac \approx ca$ ($\lambda \not\approx ba$ is selected for 6. Rewrite of 6 with 8)
10: $\lambda \not\approx \lambda \vee ac \approx ca$ ($\lambda \not\approx bb$ is selected for 9. Rewrite of 9 with 2)
11: $ac \approx ca$ ($\lambda \not\approx \lambda$ is selected for 10. Equality Resolution on 10)
12: $\lambda \not\approx bb \vee ac \not\approx ca \vee bc \approx cb$ ($\lambda \not\approx ba$ is selected for 7. Rewrite of 7 with 8)
13: $\lambda \not\approx \lambda \vee ac \not\approx ca \vee bc \approx cb$ ($\lambda \not\approx bb$ is selected for 12. Rewrite of 12 with 2)
14: $ac \not\approx ca \vee bc \approx cb$ ($\lambda \not\approx \lambda$ is selected for 13. Equality Resolution on 13)
15: $ca \not\approx ca \vee bc \approx cb$ ($ac \not\approx ca$ is selected for 14. Rewrite of 14 with 11)
16: $bc \approx cb$ ($ca \not\approx ca$ is selected for 15. Equality Resolution on 15)
$\ldots$

After some simplification steps, we have a saturated set $\bar{R}$ for R under $\mathfrak{S}$ using our selection strategy (i.e., the selection of negative literals). We may infer that the positive literals in $\bar{R}$ are as follows. $1' : bb \approx \lambda$, $2' : a \approx b$, and $3' : bc \approx cb$. Note that only the positive literals in $\bar{R}$ are now needed to solve a word problem w.r.t. R because of our selection strategy.

Now, consider the word problem $\phi := acbcba \approx^? bccaba$ w.r.t. R, where the goal of ϕ is $G := acbcba \not\approx bccaba$. We only need the Rewrite and Equality Resolution steps on G and its derived literals from G using $1'$, $2'$, and $3'$. Note that all the following literals are selected except the empty clause.

$4'$: $bcbcbb \not\approx bccbbb$ (Rewrite steps of G and its derived literals from G using $2'$).
$5'$: $bcbc \not\approx bccb$ (Rewrite steps of $4'$ and its derived literals from $4'$ using $1'$).
$6'$: $ccbb \not\approx ccbb$ (Rewrite steps of $5'$ and its derived literals from $5'$ using $3'$).
$7'$: $\square$ (Equality Resolution on $6'$)

Since $\bar{R} \cup G$ is inconsistent, we see that $R \cup G$ is inconsistent by the soundness of the derivation, where R and $\bar{R}$ are consistent. Therefore, we may infer that $R \models acbcba \approx bccaba$.

7 Related Work

Equational reasoning on strings has been studied extensively in the context of string rewriting systems and Thue systems [8] and their related algebraic structures. The monotonicity assumption used in this paper is found in string rewriting systems and Thue systems in the form of a congruence relation (see [8, 17]). See [7, 10, 21, 23] also for the completion of algebraic structures and decidability results using string rewriting systems, in particular *cross-sections* for finitely presented monoids discussed by Otto et al [23]. However, those systems are not concerned with equational theorem proving for general clauses over strings. If the monotonicity assumption is discarded, then equational theorem proving for clauses over strings can be handled by traditional superposition calculi or SMT with the theory of equality with uninterpreted functions (EUF) and their variants [6] using a simple translation into first-order ground terms. Also, efficient SMT solvers for various string constraints were discussed in the literature (e.g., [20]).

Meanwhile, equational theorem proving modulo associativity was studied in [26]. (See also [19] for equational theorem proving with *sequence variables* and fixed or variadic arity symbols). This approach is not tailored towards (ground) strings, so we need an additional encoding for each string. Also, it is probably less efficient since it is not tailored towards ground strings, and it does not provide a similar decision procedure discussed in Section 6.

The proposed calculus is the first sound and refutationally complete equational theorem proving calculus for general clauses over strings under the monotonicity assumption. One may attempt to use the existing superposition calculi for clauses over strings with the proposed translation scheme, which translates clauses over strings into clauses over first-order terms discussed in Section 3.2. However, this does not work because of the Equality Factoring rule [3, 22] or the Merging Paramodulation rule [3], which is essential for first-order superposition theorem proving calculi in general. For example, consider a clause $a \approx b \vee a \approx c$ with $a \succ b \succ c$, which is translated into a first-order clause $a(x) \approx b(x) \vee a(y) \approx c(y)$. The Equality Factoring rule yields $b(z) \not\approx c(z) \vee a(z) \approx c(z)$ from $a(x) \approx b(x) \vee a(y) \approx c(y)$, which cannot be translated back into a clause over strings (see Lemma 1). Similarly, a first-order clause produced by Merging Paramodulation may not be translated back into a clause over strings. If one is only concerned with refutational completeness, then the existing superposition calculi[6] can be adapted by using the proposed translation scheme. In this case, a saturated set may not be translated back into clauses over strings in some cases, which is an obvious drawback for its applications (see *programs* in [3]).

8 Conclusion

This paper has presented a new refutationally complete superposition calculus with strings and provided a framework for equational theorem proving for clauses over strings. The results presented in this paper generalize the results about completion of string rewriting systems and equational theorem proving using equations over strings. The proposed superposition calculus is based on the simple string matching methods and the efficient length-lexicographic ordering that allows one to compare two finite strings in linear time for a fixed signature with its precedence.

The proposed approach translates for a clause over strings into the first-order representation of the clause by taking the monotonicity property of equations over strings into account. Then the existing notion of redundancy and model construction techniques for the equational theorem proving framework for clauses over strings has been adapted. This paper has also provided a decision procedure for word problems over strings w.r.t. a set of conditional equations R over strings if R can be finitely saturated under the Superposition, Rewrite and Equality Resolution rule. (The complexity analysis of the proposed approach is not discussed in this paper. It is left as a future work for this decision procedure.)

Since strings are fundamental objects in mathematics, logic, and computer science including formal language theory, developing applications based on the proposed superposition calculus with strings may be a promising future research direction. Also, the results in this paper may have potential applications in verification systems and solving satisfiability problems [1].

In addition, it would be an interesting future research direction to extend our superposition calculus with strings to superposition calculi with strings using built-in equational theories, such as commutativity, *idempotency* [8], *nilpotency* [15], and their various combinations. For example, research on superposition theorem proving for *commutative monoids* [25] is one such direction.

[6] The reader is also encouraged to see *AVATAR modulo theories* [24], which is based on the concept of splitting.

References

[1] Alessandro Armando, Silvio Ranise & Michael Rusinowitch (2003): *A rewriting approach to satisfiability procedures*. Information and Computation 183(2), pp. 140 – 164, doi:10.1016/S0890-5401(03)00020-8.

[2] Franz Baader & Tobias Nipkow (1998): *Term Rewriting and All That*. Cambridge University Press, Cambridge, UK, doi:10.1017/CBO9781139172752.

[3] Leo Bachmair & Harald Ganzinger (1994): *Rewrite-based Equational Theorem Proving with Selection and Simplification*. J. Log. Comput. 4(3), pp. 217–247, doi:10.1093/logcom/4.3.217.

[4] Leo Bachmair & Harald Ganzinger (1995): *Associative-commutative superposition*. In Nachum Dershowitz & Naomi Lindenstrauss, editors: Conditional and Typed Rewriting Systems, Springer, Berlin, Heidelberg, pp. 1–14, doi:10.1007/3-540-60381-6_1.

[5] Leo Bachmair & Harald Ganzinger (1998): *Equational Reasoning in Saturation-Based Theorem Proving*. In Wolfgang Bibel & Peter H. Schmitt, editors: Automated Deduction. A basis for applications, chapter 11, Volume I, Kluwer, Dordrecht, Netherlands, p. 353–397, doi:10.1007/978-94-017-0437-3.

[6] Clark Barrett, Roberto Sebastiani, Sanjit A Seshia & Cesare Tinelli (2009): *Satisfiability Modulo Theories*. In: Handbook of satisfiability, Frontiers in Artificial Intelligence and Applications 185, IOS Press, pp. 825–885, doi:10.3233/978-1-58603-929-5-825.

[7] Ronald V. Book & Colm P. O'Dunlaing (1981): *Testing for the Church-Rosser property*. Theoretical Computer Science 16(2), pp. 223–229, doi:10.1145/321850.321862.

[8] Ronald V. Book & Friedrich Otto (1993): *String-Rewriting Systems*. Springer, New York, NY, doi:10.1007/978-1-4613-9771-7.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein (2001): *Introduction to Algorithms*, second edition. The MIT Press.

[10] Robert Cremanns & Friedrich Otto (2002): *A Completion Procedure for Finitely Presented Groups That Is Based on Word Cycles*. J. Autom. Reason. 28(3), pp. 235–256, doi:10.1023/A:1015741511536.

[11] Thomas Deiß (1992): *Conditional Semi-Thue systems for Presenting Monoids*. In: Annual Symposium on Theoretical Aspects of Computer Science, STACS 1992, Springer, Berlin, Heidelberg, pp. 557–565, doi:10.1007/3-540-55210-3_212.

[12] Nachum Dershowitz (1991): *Canonical sets of Horn clauses*. In Javier Leach Albert, Burkhard Monien & Mario Rodríguez Artalejo, editors: Automata, Languages and Programming, Springer Berlin Heidelberg, pp. 267–278, doi:10.1007/3-540-54233-7_140.

[13] Nachum Dershowitz & David A. Plaisted (2001): *Rewriting*. In: Handbook of Automated Reasoning, chapter 9, Volume I, Elsevier, Amsterdam, pp. 535 – 610, doi:10.1016/b978-044450813-3/50011-4.

[14] David B. A. Epstein, Mike Paterson, James W. Cannon, Derek F. Holt, Silvio V. F. Levy & William Thurston (1992): *Word Processing in Groups*. A. K. Peters, Ltd., Natick, MA, doi:10.1201/9781439865699.

[15] Qing Guo, Paliath Narendran & David A. Wolfram (1996): *Unification and matching modulo nilpotence*. In Michael A. McRobbie & John K. Slaney, editors: The 13th International Conference on Automated Deduction (CADE-13), Lecture Notes in Computer Science 1104, Springer, Berlin, Heidelberg, pp. 261–274, doi:10.1007/3-540-61511-3_90.

[16] Derek F. Holt, Bettina Eick & Eamonn A. O'Brien (2005): *Handbook of computational group theory*. CRC Press, Boca Raton, FL, doi:10.1201/9781420035216.

[17] Deepak Kapur & Paliath Narendran (1985): *The Knuth-Bendix Completion Procedure and Thue Systems*. SIAM Journal on Computing 14(4), pp. 1052–1072, doi:10.1137/0214073.

[18] Dohan Kim & Christopher Lynch (2021): *Equational Theorem Proving Modulo*. In: The 28th International Conference on Automated Deduction (CADE-28), Lecture Notes in Computer Science 12699, Springer, pp. 166–182, doi:10.1007/978-3-030-79876-5_10.

[19] Temur Kutsia (2002): *Theorem Proving with Sequence Variables and Flexible Arity Symbols*. In Matthias Baaz & Andrei Voronkov, editors: *Logic for Programming, Artificial Intelligence, and Reasoning, 9th International Conference, LPAR 2002, Tbilisi, Georgia, October 14-18, 2002, Proceedings*, *Lecture Notes in Computer Science* 2514, Springer, pp. 278–291, doi:10.1007/3-540-36078-6_19.

[20] Tianyi Liang, Andrew Reynolds, Nestan Tsiskaridze, Cesare Tinelli, Clark Barrett & Morgan Deters (2016): *An Efficient SMT Solver for String Constraints*. *Form. Methods Syst. Des.* 48(3), p. 206–234, doi:10.1007/s10703-016-0247-6.

[21] Klaus Madlener, Paliath Narendran & Friedrich Otto (1991): *A Specialized Completion Procedure for Monadic String-Rewriting Systems Presenting Groups*. In Javier Leach Albert, Burkhard Monien & Mario Rodríguez-Artalejo, editors: *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, *Lecture Notes in Computer Science* 510, Springer, pp. 279–290, doi:10.1007/3-540-54233-7_141.

[22] Robert Nieuwenhuis & Albert Rubio (2001): *Paramodulation-based theorem proving*. In: *Handbook of Automated Reasoning*, chapter 7, Volume I, Elsevier, Amsterdam, pp. 371–443, doi:10.1016/b978-044450813-3/50009-6.

[23] Friedrich Otto, Masashi Katsura & Yuji Kobayashi (1997): *Cross-Sections for Finitely Presented Monoids with Decidable Word Problems*. In Hubert Comon, editor: *Rewriting Techniques and Applications, 8th International Conference, RTA-97, Sitges, Spain, June 2-5, 1997, Proceedings*, *Lecture Notes in Computer Science* 1232, Springer, pp. 53–67, doi:10.1007/3-540-62950-5_61.

[24] Giles Reger, Nikolaj Bjøner, Martin Suda & Andrei Voronkov (2016): *AVATAR Modulo Theories*. In Christoph Benzmüller, Geoff Sutcliffe & Raul Rojas, editors: *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, *EPiC Series in Computing* 41, pp. 39–52, doi:10.29007/k6tp.

[25] José Carlos Rosales, Pedro A. García-Sánchez & Juan M. Urbano-Blanco (1999): *On Presentations of Commutative Monoids*. *International Journal of Algebra and Computation* 09(05), pp. 539–553, doi:10.1142/S0218196799000333.

[26] Albert Rubio (1996): *Theorem Proving modulo Associativity*. In Hans Kleine Büning, editor: *Computer Science Logic*, Springer, Berlin, Heidelberg, pp. 452–467, doi:10.1007/3-540-61377-3_53.