Infinite families of finite string rewriting systems and their confluence

Jean-Pierre Jouannaud, Benjamin Monate

Infinite families of finite string rewriting systems and their confluence

Jean-Pierre Jouannaud
INRIA-LIAMA and Tsinghua University

Benjamin Monate
CEA, LIST

Abstract. We introduce *parameterized rewrite systems* for describing *infinite families of finite string rewrite systems* depending upon non-negative integer parameters, as well as ways to reason *uniformly* over these families. Unlike previous work, the vocabulary on which a rewrite system in the family is built depends itself on the integer parameters. Rewriting makes use of a toolkit for *parameterized words* which allows to describe a rewrite step made independently by all systems in an infinite family by a single, effective parameterized rewrite step. The main result is a confluence test for all systems in a family at once, based on a critical pair lemma classically based on computing finitely many overlaps between left-hand sides of parameterized rules and then checking for their joinability (which decidability is not garanteed).

1 Introduction

Consider a family of groups $\{S_N\}_{N\in\mathbb{N}}$ with generators $a_1, \ldots, a_N$ satisfying:

$$a_i^2 = \epsilon \mid 1 \leq i \leq N, \qquad a_i a_j = a_j a_i \mid i > j+1 \ \wedge\ 1 \leq i, j \leq N$$

This axiomatization depends upon the *parameter* $N \in \mathbb{N}$ in four essential ways: there is one finite set of axioms for each value of the parameter N; and in each set, the number of equations depends on N; the vocabulary depends on N; words in the equations depend on N via integer variables i, j satisfying arithmetic constraints in which N occurs.

The methodology for proving properties of S_N for a given N by machine is well-known: it requires the computation of a complete (confluent and terminating) string rewriting system for S_N. This can be achieved for each given $N \in \mathbb{N}$ by using Knuth-Bendix completion or one of its variants. The study by machine of various finite groups has been carried out in the non-parameterized case, in particular by Le Chenadec [7, 8]. Much apparatus has later been developed to describe and reason about infinite languages of terms by using tailored unification algorithms [2, 10, 5, 9]. Such languages arise for example in Knuth-Bendix completion when the process diverges.

However, all formalisms we know of, *whether mentioned or not*, allow one to represent terms on a *given fixed vocabulary* and specify and reason about a *single algebraic structure*, which does not fit at all our purpose here.

In this paper we show how to deal *at once* with the *infinite family* $\{\mathcal{S}_N\}_{N\in\mathbb{N}}$, *without instantiating the parameter* N. To achieve this goal, we define an extension of the notions of (families of) words, equations and rewrite rules in case the alphabet itself depends on the parameter N. We then show how to mechanize termination proofs and reduce local confluence of such systems to the joinability of finitely many critical pairs.

As a result, the above infinite family $\mathcal{S}_N$ can be directly presented as the complete parameterized string rewriting system:

$$a_i^2 \to \epsilon \mid 1 \leq i \leq N, \qquad a_i a_j \to a_j a_i \mid i > j+1 \ \wedge \ 1 \leq i,j \leq N$$

We stress that our ultimate goal is *not* the study of parameterized groups, which should be seen as an example illustrating techniques which we believe to be of general interest. In this respect, the framework we develop, and the methodology used to lift results from plain rewriting to parameterized rewriting is more important to us than the actual technical results, whatever difficult they indeed are.

We define parameterized words in Section 3, show how to decide equality and factor out parameterized words in Section 4, and introduce parameterized rewriting in Section 5 before we investigate termination and confluence in this setting. An application to dihedral groups is carried out in Section 5 with the rewriting toolkit C*i*ME2 [3], implemented in part by the second author for his PhD thesis, which results are generalized in the present work.

2 Preliminaries

We assume given an infinite alphabet of constant symbols $\mathcal{A} = \{a_i\}_{i \in \mathcal{N}}$ called *generators* or *letters*.

Our formalism relies heavily on the existential fragment of Presburger Arithmetic (PrA) using $0, s, +$ as operations for defining terms, $=, >, <, \geq, \leq$ as predicates for defining formulas, and two disjoint sets of *arithmetic variables*: a set $\mathcal{P}$ of *parameters* denoted by capital letters, and a set I of *dependent variables* denoted by lower-case letters. Values of variables in I depend upon values of parameters via a Presburger constraint, hence their name. We call *solution* of φ an assignment which satisfies φ. We use $\psi \models_{\text{PrA}} \varphi$ for *entailment* in PrA, meaning that any solution of ψ is a solution of φ. We use $\top, \bot$ for the logical constants $true$ and $false$ respectively, $\mathcal{V}ar(e)$ for the set of free variables of an expression e of any kind, $\mathcal{V}arI(e)$ for $\mathcal{V}ar(e) \cap I$ and $\mathcal{V}ar\mathcal{P}(e)$ for $\mathcal{V}ar(e) \cap \mathcal{P}$. We refer to [4, 1, 6] for missing notations and definitions.

3 Parameterized words

3.1 Syntax

Definition 1. Parameterized words *are pairs written* $w \mid \psi$ *made of:*
– *a* word-expression w *defined by the following grammar of axiom* W*:*

$$\begin{array}{rcl} W & := & \epsilon \mid a_f W \mid (F)^e W \\ F & := & a_f \mid a_f F \end{array} \qquad \left\{ \begin{array}{l} \text{where } e, f \textit{ denote arithmetic} \\ \textit{variables or constants in } \mathbb{N}. \end{array} \right.$$

– a quantifier-free formula ψ *of* PrA *s.t.* $\mathcal{V}ar\mathcal{P}(\psi) = \{N_i\}_{i \in [1..n]}$, $\mathcal{V}ar(w) \subseteq \mathcal{V}ar(\psi)$, *and* $\forall k_1 \dots k_n \in \mathbb{N}^n$, *the formula* $\psi \wedge \bigwedge_{i \in [1..n]} N_i = k_i$ *has finitely many solutions.*
A word-expression is: reduced *if all exponents are variables;* constant *if* $\mathcal{V}ar(w) = \emptyset$*;* flat *if it has no exponent; a* word *if it is constant and flat. In* $(w)^e$*,* w *is a non-empty flat word-expression with* exponent e*. In* a_f*,* f *is the* index *of the* letter a*.*

Limitations: the grammar forbids nesting of exponents: a^{i^j} is not a parameterized word. This restriction is also compulsory for terms with integer exponents as defined in [2]: nesting allows for easy encodings of Peano arithmetic. All variables are arithmetic: variables standing for words are out of scope of this paper. There is no theoretical reason for restricting PrA to its existential fragment, apart from its lower complexity. All other syntactic restrictions are for convenience.
Lexicography: we use φ, ψ, θ for Presburger constraints, s, t, u, v, w for arbitrary word-expressions, x, y, z for flat ones, a for the letter a_1 and b for a_2 in examples, and write n for $s^n(0)$ and $n + u$ for $s^n(u)$.
We use bold letters $\mathbf{p}, \mathbf{q}$ to stress constant exponents in word-expressions like $(x)^{\mathbf{p}}$.
Conventions: we sometimes write a^i instead of $(a)^i$ and , $x^{\mathbf{p}}$ instead of $(x)^{\mathbf{p}}$; word-expressions can be easily expanded into reduced ones; we also identify constant word-expressions with words; for convenience, we allow us to write $\mathbf{p} * n$ for the sum $\mathbf{p}$-times of n when $\mathbf{p}$ is a constant in $\mathbb{N}$.

3.2 Semantics

Terms with integer exponents [2], or of a primal grammar [5] denote sets of terms. Because we distinguish *local* dependent variables constrained by a formula of PrA from *global* parameters which can take arbitrary values in $\mathbb{N}$, a parameterized word $w \mid \varphi$ denotes an $\mathbb{N}^{|\mathcal{P}|}$-indexed family of sets of words, and words in each set are obtained by replacing in w the variables in I by the natural numbers satisfying the constraint φ for the considered value of the parameters in $\mathcal{P}$. An *arithmetic valuation* is an application from $\mathcal{P} \cup I$ to $\mathbb{N}^{|\mathcal{P} \cup I|}$ which we split into two, ν for the parameters, and μ for the dependent variables. Given an expression e, we write $\nu(e)$ (resp. $\mu(e)$) for the expression obtained by replacing the variables in $\mathcal{P}$ (resp. I) by their value and possibly eliminating constant exponents. Note that $\mu(\nu(w))$ is a constant word-expression if $w \mid \varphi$ is a parameterized word, and that $\mu(\nu(\varphi))$ is a formula without arithmetic variable, hence evaluates to $\top$ or $\bot$ in PrA. We call *instance* of $w \mid \varphi$ a word $\mu(\nu(w))$ such that $\mu\nu$ satisfies φ. We use a bracketed notation for the semantics of expressions of any kind.

Definition 2. *We define successively the interpretation of a parameterized word $u \mid \psi$ and of a constant word-expression:*

$$[\![u \mid \psi]\!] = \{[\![u \mid \psi]\!]_\nu\}_{\nu \in \mathcal{P} \mapsto \mathbb{N}^{|\mathcal{P}|}}$$
$$[\![u \mid \psi]\!]_\nu = \{[\![\mu(\nu(u))]\!] \mid \mu \in I \mapsto \mathbb{N}^{|I|} \textit{ such that } \mu \models_{\mathrm{PrA}} \nu(\psi)\}$$
$$[\![\epsilon]\!] = \epsilon \qquad [\![a_n x]\!] = a_n [\![x]\!] \qquad [\![(x)^n y]\!] = \underbrace{x \cdots x}_{n \text{ times}} [\![y]\!]$$

Consider for example the parameterized word $(a_i^N \mid 0 < i < N)$. Then,
- $\left[\!\left[a_i^N \mid 0 < i < N \right]\!\right]_{N=0} = \emptyset$ since the formula $0 < i < 0$ is unsatisfiable;
- $\left[\!\left[a_i^N \mid 0 < i < N \right]\!\right]_{N=1} = \emptyset$ since the formula $0 < i < 1$ is unsatisfiable;
- $\left[\!\left[a_i^N \mid 0 < i < N \right]\!\right]_{N=2} = \{a_1 a_1\}$, since $0 < i < 2$ implies $i = 1$;
- $\left[\!\left[a_i^N \mid 0 < i < N \right]\!\right]_{N=3} = \{a_1 a_1 a_1, a_2 a_2 a_2\}$, since $0 < i < 3$ implies $i \in \{1, 2\}$.

Therefore, $[\![(a_i^N \mid 0 < i < N)]\!] = \{\{\}, \{\}, \{a_1 a_1\}, \{a_1 a_1 a_1, a_2 a_2 a_2\}, \ldots\}$. We see that separating arithmetic variables in two sets is used in the semantics to stratify the interpretation of a parameterized word into an infinite family of finite sets of words.

It is *convenient* to consider *conjunctive parameterized words* $(u \mid \varphi) \ \wedge \ (v \mid \psi)$ and *disjunctive parameterized words* $(u \mid \varphi) \vee (v \mid \psi)$, interpreting conjunction and disjunction as set intersection and set union at the set level of interpretation. Conjunctive and disjunctive words do *not* allow for any more expressivity: we shall give an algorithm replacing conjunctive words by disjunctive ones (**Intersection**, page 8), and can always move disjunctions from words to PrA formulas by systematizing the following trick: $\{a^i b \mid i \leq N\} \vee \{b^i a \mid i \leq N\} = \{a^i b^j \mid i \leq N \ \wedge \ j = 1\} \vee \{b^k a^l \mid k \leq N \ \wedge \ l = 1\} = \{a^i b^j b^k a^l \mid (i \leq N \ \wedge \ j = 1 \ \wedge \ k = l = 0) \vee (k \leq N \ \wedge \ l = 1 \ \wedge \ i = j = 0)\}$.

4 The rewriting toolkit for parameterized words

To rewrite a parameterized word, we need to factor it out via a lefthand side of rule. To test for confluence, we need to check equality of parameterized words, which shall require computing their intersection. To compute critical pairs, we need to compute overlaps of parameterized words. Verifying equality, computing factors and overlaps are the main algorithmic difficulties of this framework. We choose to present the rewriting toolkit first, before to introduce parameterized rewriting itself. For lack of space, we treat in details factorization, and then sketch how intersection, equality and overlaps can be derived. Examples are shown for factorization and equality. These algorithms have a non-polynomial complexity, but in our practice rules have usually a small size.

4.1 Auxiliary algorithms

All our algorithms, factorization, equality checking and computing overlaps, use as basic building blocks two auxiliary algorithms, for computing common divisors and non-empty common repeated prefixes of two terms. We start with these two algorithms, taking advantage of their relative simplicity to sketch their description.

gcd takes two non-empty flat word-expressions x, y and returns a possibly empty finite set of *solutions* $\{(z_i, \mathbf{k}_i, \mathbf{l}_i \mid \theta_i)\}_i$ with $z_i \in \mathcal{A}^+; \mathbf{k}_i, \mathbf{l}_i \in \mathbb{N}^+; \theta_i \not\models_{\mathrm{PrA}} \perp$ satisfying:
(i) soundness: $\forall \mu\nu$ such that $\mu\nu \models_{\mathrm{PrA}} \theta_i, \quad \mu\nu(x) = \mu\nu(z_i^{\mathbf{k}_i})$ and $\mu\nu(y) = \mu\nu(z_i^{\mathbf{l}_i})$;
(ii) completeness: $\forall \mu\nu$ such that $\mu\nu(x) = \mu\nu(z^{\mathbf{k}})$ and $\mu\nu(y) = \mu\nu(z^{\mathbf{l}})$ for some triple $(z, \mathbf{k}, \mathbf{l})$, there exists $(z_i, \mathbf{k}_i, \mathbf{l}_i \mid \theta_i)$ such that $\mu\nu \models_{\mathrm{PrA}} \theta_i$.
Consider for example $x = a_k a_l, y = a_i a_j a_i a_j$. Then $\{(a_k a_l, 1, 2 \mid k = i \ \wedge \ l = j)\}$ is a solution of **gcd**(x, y). The solution $(a_k, 2, 4 \mid i = j = k = l)$ is indeed an *instance* of the previous one, since $i = j = k = l \models_{\mathrm{PrA}} k = i \ \wedge \ l = j$.
Let now $x = a_i a_j a_k a_l a_m a_n$ and $y = a_i a_j a_k a_l a_m a_n a_i a_j a_k a_l a_m a_n$. There are two incomparable solutions, $(a_i a_j, 3, 6 \mid i = k = m \ \wedge \ j = l = n)$ and $(a_i a_j a_k, 2, 4 \mid i = l \ \wedge \ j = m \ \wedge \ k = n)$ which must both be returned by **gcd** for sake of completeness. An initial constraint φ can be accomodated by returning $\{(z_i \neq \epsilon, \mathbf{k}_i, \mathbf{l}_i \mid \varphi \ \wedge \ \theta_i)\}_i$.

There is an easy *guess and check* algorithm for **gcd**, in which the only needed guesses are the triples of natural numbers p, k, l such that $|x| = p \times k$ and $|y| = p \times l$. The constraint θ under which $x = z^k$ and $y = z^l$ is then obtained by equating the respective indices of all flat word-expressions to be equated. An initial constraint φ is accomodated by returning $\{(z_i \neq \epsilon, \mathbf{k}_i, \mathbf{l}_i \mid \varphi \ \wedge \ \theta_i)\}_i$. One may of course be willing to pay the price for filtering out redundant guesses. The answer set is empty iff factoring is impossible.

gpref takes two non-empty flat word-expressions x, y with $|x| < |y|$ and returns a (possibly empty) finite complete set of triples written $\{(\mathbf{p}_i \neq 0, t_i \neq \epsilon \mid \theta_i)\}_i$ such that: (i) soundness: $\forall \mu\nu.\mu\nu \models_{PrA} \theta_i$, $\mu\nu(y) = \mu\nu((x)^{\mathbf{p}_i} t_i)$ and $\mu\nu(x)$ is not prefix of $\mu\nu(t_i)$; (ii) completeness: $\forall \mu\nu$ such that $\mu\nu(y) = (\mu\nu(x))^{\mathbf{p}} \mu\nu(t)$ for some pair $(\mathbf{p}, t)$, there exists $(\mathbf{p}_i, t_i \mid \theta_i)$ such that $\mu\nu \models_{PrA} \theta_i$.

Let for example $x = a_i a_j$ and $y = a_i a_j a_k a_l a_j a_i$. Then $\mathbf{gpref}(x, y) = \{(a_i a_j, 1, 1, a_k a_l a_j a_i \mid i \neq k \vee j \neq l), (a_i a_j, 1, 2, a_j a_i \mid i = k \wedge j = l \wedge i \neq j)\}$, which can be obtained by a *guess and check* algorithm as before.

4.2 Factoring

We address now the problem of factoring a parameterized word into one or several quadruples made of a prefix, a given *non-empty* factor, a suffix and a constraint characterizing under which additional condition this decomposition holds. Traditionally, factors are associated with positions. Here, the notion of position is not at all clear: in $a^N b a^N$ the position $N+1$ of b depends on the parameter, but the first position of a to the right of b is $N+2$ if $N > 0$ and is not defined if $N = 0$. This makes it difficult to reduce factoring to equality by first non-deterministically guessing a prefix and a suffix position and then checking the delimited factor for equality.

Let us first look whether the word aba is a factor of the parameterized word $a^N b a^N \mid N > 0$. There is a unique possibility to decompose $a^N b a^N$ so as to obtain the factor aba, namely : $a^N b a^N = a^{N-1} a b a b^{N-1}$. We can therefore write informally that $(a^N b a^N \mid N > 0) = (a^{N-1}, aba, b^{N-1} \mid N > 0)$.
Consider now whether $(ba)^j b \mid j < N$ is a factor of $(ab)^i \mid i \leq N$. This time, we can write $(ab)^i \mid i \leq N = ((ab)^l a, (ba)^j b, \epsilon \mid i \leq N \wedge j < N \wedge i = l+j+k+1) \vee ((ab)^i \mid i \leq N \wedge j < N \wedge i \neq l+j+k+1)$. Factoring here is partial, the constraint of the factorized term being strengthened.
Let us finally check if $a^N b a^N$ is a factor in aba, obtaining this time a disjunction of two possible decompositions: $aba = (\epsilon, a^N b a^N, \epsilon \mid N = 1) \vee (a, a^N b a^N, a \mid N = 0)$, that do not cover all possible values of N: factoring is again partial, the constraint of the factoring term being strengthened this time.

Factoring requires searching where a given factor starts in a given parameterized word. To answer this need, our algorithm is organized in two steps. First, the search for a prefix, from which point on an equality check can start. This search is exhaustive, we then need to check equality of the factor with a prefix of the other parameterized word in the second phase. The suffix is of course obtained at the end of this second phase when successful.

Consider the factorization of the parameterized word $w_0 \mid \varphi$ by the parameterized word $s_0 \mid \psi$. We aim at a *representation* of all solutions as a parameterized factorization. To this end, our rules operate on quintuples (u, v, w, s, φ), written as $(u, v, w, s \mid \varphi)$, by maintaining two invariants: $uvw = w_0$ and $vs = s_0$. The word-expression v is therefore both a factor of w_0 and a prefix of s_0. To control the enumeration of all potential prefixes u, we use a special symbol "$\_$" to block the rules checking for a factor until it is replaced by ϵ from which point on the prefix u is frozen. A factorization is obtained when $s = \epsilon$ and $v = s_0$, the word-expression w being then the suffix of that factorization.

[Factorization.] Input: two parameterized words $w_0 \neq \epsilon \mid \varphi$ and $s_0 \mid \psi$;
Output: a factorization $\vee_i(u_i, v_i, w_i \mid \psi_i)$ such that $\psi_i \not\models_{\mathrm{PrA}} \bot$

$$\text{(Start)}\ \frac{w_0 \mid \varphi,\ s_0 \mid \psi}{\epsilon, \_, w_0, s_0 \mid \varphi \wedge \psi} \quad \text{(Elim)}\ \frac{(u, v, w, s \mid \varphi) \vee P}{P} \quad \text{(Out)}\ \frac{\bigvee_i (u_i, v_i, w_i, \epsilon \mid \varphi_i)}{\bigvee_i (u_i, v_i, w_i \mid \varphi_i)}$$

$$\textbf{if} \qquad \varphi \models_{\mathrm{PrA}} \bot$$

$$\text{FindPref: (1)}\ \frac{u, \_, \epsilon, s \mid \varphi}{u, \epsilon, \epsilon, s \mid \varphi} \qquad (2)\ \frac{u, \_, a_i w, s \mid \varphi}{(u, \epsilon, a_i w, s \mid \varphi) \vee (u a_i, \_, w, s \mid \varphi)}$$

$$(3)\frac{u, \_, (x)^n w, s \mid \varphi}{(\ \bigvee_{x=yz} u(x)^i y, \epsilon, z(x)^j w, s \mid \varphi \wedge n = i + j + 1) \vee (u(x)^n, \_, w, s)}$$

$$\text{Finish: (1)}\ \frac{u, \epsilon, \epsilon, a_i s \mid \varphi}{\bot} \qquad (2)\ \frac{u, \epsilon, \epsilon, (x)^n s \mid \varphi}{u, \epsilon, \epsilon, s \mid \varphi \wedge n = 0}$$

$$\text{CheckFactor:} \qquad (1)\frac{u, v, a_i w, a_j s \mid \varphi}{u, v a_i, w, s \mid \varphi \wedge i = j}$$

$$(2)\frac{u, v, a_i w, (a_j y)^n s \mid \varphi}{(u, v, a_i w, s \mid \varphi \wedge n = 0) \vee (u, v a_i, w, (y a_j)^k y s \mid \varphi \wedge n = k + 1 \wedge i = j)}$$

$$(3)\frac{u, v, (a_i x)^m w, a_j s \mid \varphi}{(u, v, w, a_j s \mid \varphi \wedge m = 0) \vee (u, v a_i, (x a_j)^k x w, s \mid \varphi \wedge m = k + 1 \wedge i = j)}$$

$$(4)\frac{u, v, (x)^m w, (y)^n s \mid \varphi}{\begin{array}{l}[u, v, (x)^m w, s \mid \varphi \wedge n = 0] \vee [u, v, w, (y)^n s \mid \varphi \wedge m = 0] \\ \vee \bigvee_{(z, \mathbf{p}, \mathbf{q} \mid \theta) \in \mathbf{gcd}(x, y)} \\ [u, v(z)^j, (z)^i w, s \mid \varphi \wedge \theta \wedge j = \mathbf{q} * n \wedge i + j = \mathbf{p} * m \wedge i \geq 0 \wedge n \neq 0] \vee \\ [u, v(z)^i, w, (z)^j s \mid \varphi \wedge \theta \wedge i = \mathbf{p} * m \wedge j + i = \mathbf{q} * n \wedge j > 0 \wedge m \neq 0] \\ \vee \bigvee_{(\mathbf{p}, z \mid \theta) \in \mathbf{gpref}(x, y)} \bigvee_{\mathbf{q} \in [1..\mathbf{p}]} [u, v x^{\mathbf{q}}, w, x^{\mathbf{p}-\mathbf{q}} z (y)^l s \mid \varphi \wedge \theta \wedge n = l + 1] \\ \vee \bigvee_{(\mathbf{p}, z \mid \theta) \in \mathbf{gpref}(y, x)} \bigvee_{\mathbf{q} \in [1..\mathbf{p}]} [u, v x^{\mathbf{q}}, x^{\mathbf{p}-\mathbf{q}} z (x)^l w, s \mid \varphi \wedge \theta \wedge m = l + 1]\end{array}}$$

FindPref (3) is slightly redundant to maintain a one line formulation. In **CheckFactor** (4), the word-expressions w, s are maintained in reduced form: $x^{\mathbf{q}}$ and $x^{\mathbf{p}-\mathbf{q}}$ stand for (expanded) words. Note that $j = \mathbf{q} * n \wedge i + j = \mathbf{p} * m \wedge i \geq 0 \wedge n \neq 0$ implies $m \neq 0$. Fresh dependent variables appear in conclusions, making termination non-trivial.

We now illustrate **Factoring** with the simple example of the word-expression $w = a^N b a^N$ with the word $s = aba$. We describe the transformations in a rewriting style, starting with the search for a prefix. The arrow rewriting symbol may use a shortened rule name in index and a disjunct number in exponent to ease the reading. Non-modified disjuncts are replaced by dots:

$$\begin{array}{l}(a^N b a^N, aba) \Rightarrow_{Start} (\epsilon, \_, a^N b a^N, aba) \Rightarrow_{FP(3)} \\ (a^i, \epsilon, a a^j b a^N, aba \mid N = i + j + 1) \vee (a^i a, \epsilon, a^j b a^N, aba \mid N = i + j + 1) \vee \\ \quad (a^N, \_, b a^N, aba) \Rightarrow^3_{FP(2)} \\ \ldots (a^N, \epsilon, b a^N, aba) \vee (a^N b, \_, a^N, aba) \Rightarrow^4_{FP(3)}\end{array}$$

$\ldots (a^N ba^i, \epsilon, aa^j, aba \mid N = i+j+1) \vee (a^N ba^i a, \epsilon, a^j, aba \mid N = i+j+1) \vee$
$(a^N ba^N, \_, \epsilon, aba) \Rightarrow^6_{FP(1)}$
$\ldots (a^N ba^N, \epsilon, \epsilon, aba) \Rightarrow^6_{Finish(1)}$
$(a^i, \epsilon, aa^j ba^N, aba \mid N = i+j+1) \vee (a^i a, \epsilon, a^j ba^N, aba \mid N = i+j+1) \vee$
$(a^N, \epsilon, ba^N, aba) \vee$
$(a^N ba^i, \epsilon, aa^j, aba \mid N = i+j+1) \vee (a^N ba^i a, \epsilon, a^j, aba \mid N = i+j+1) \Rightarrow^3_{CF(1)}$
$(a^i, \epsilon, aa^j ba^N, aba \mid N = i+j+1) \vee (a^i a, \epsilon, a^j ba^N, aba \mid N = i+j+1) \vee$
$(a^N ba^i, \epsilon, aa^j, aba \mid N = i+j+1) \vee (a^N ba^i a, \epsilon, a^j, aba \mid N = i+j+1)$

The last two disjuncts fail quickly. We proceed with the successful first two. Disjuncts resulting in immediate failure are abbreviated by dots and eliminated on the fly:

$(a^i, \epsilon, aa^j ba^N, aba \mid N = i+j+1) \Rightarrow_{CF(1)} (a^i, a, a^j ba^N, ba, \mid N = i+j+1)$
$\Rightarrow_{CF(3)} (a^i, a, ba^N, ba \mid N = i+j+1 \wedge j = 0) \vee (\ldots \mid \bot) \Rightarrow^2_{Elim} \Rightarrow_{CF(1)}$
$(a^i, ab, a^N, a \mid N = i+j+1 \wedge j = 0) \Rightarrow_{CF(3)}$
$(a^i, aba, \epsilon, a \mid \bot) \vee (a^i, aba, a^i, \epsilon \mid N = i+j+1 \wedge j = 0 \wedge N = i+1)$
$\Rightarrow^1_{Elim} \Rightarrow_O (a^i, aba, a^i \mid N = i+1)$

$(a^i a, \epsilon, a^j ba^N, aba \mid N = i+j+1) \Rightarrow_{CF(3)}$
$(a^i a, \epsilon, ba^N, aba \mid N = i+j+1 \wedge j = 0) \vee$
$(a^i a, a, a^k ba^N, ba \mid N = i+j+1 \wedge j = k+1) \Rightarrow^1_{CF(1)} \Rightarrow^1_{Elim}$
$(a^i a, a, a^k ba^N, ba \mid N = i+j+1 \wedge j = k+1) \Rightarrow_{CF(3)}$
$(\ldots \mid \bot) \vee (a^i a, a, ba^N, ba \mid N = i+j+1 \wedge j = k+1 \wedge k = 0) \Rightarrow^1_{Elim} \Rightarrow_{CF(1)}$
$(a^i a, ab, a^N, a \mid N = i+j+1 \wedge j = k+1 \wedge k = 0) \Rightarrow_{CF(3)}$
$(\ldots \mid \bot) \vee (a^i a, aba, a^l, \epsilon \mid N = i+j+1 \wedge j = k+1 \wedge k = 0 \wedge N = l+1)$
$\Rightarrow^1_{Elim} \Rightarrow_O (a^i a, aba, a^l \mid N = i+2 \wedge N = l+1)$

The final result is therefore the redundant factorization
$(a^i, aba, a^i \mid N = i+1) \vee (a^i a, aba, a^l \mid N = i+2 \wedge N = l+1)$
This redundancy originates in our formulation of **FindPref** (3), which can be fixed.

We are left indeed showing that our algorithm factors out parameterized words.

Definition 3. *A triple of words (u, v, w) is a* solution *of the factorization problem of a parameterized word $s \mid \psi$ by a parameterized word $t \mid \varphi$ such that $\mathcal{V}arI(\varphi) \cap \mathcal{V}arI(\psi) = \emptyset$, if there exist a valuation $\mu\nu$ of the arithmetic variables such that $\mu\nu \models_{\mathrm{PrA}} \varphi \wedge \psi$, $\mu\nu(v) = \mu\nu(t)$ and $\mu\nu(uvw) = \mu\nu(s)$.*

Definition 4. *Given two parameterized words $s \mid \varphi$ and $v \mid \psi$ such that $\mathcal{V}arI(\varphi) \cap \mathcal{V}arI(\psi) = \emptyset$, $\bigvee_i (u_i, v_i, w_i \mid \theta_i)$ is a* complete factorization *of $s \mid \varphi$ by $v \mid \psi$ (each disjunct being one particular factorization) iff*
(i)[soundness] for each valuation $\mu\nu$ such that $\mu\nu \models_{\mathrm{PrA}} \theta_i$, the triple $(\mu\nu(u_i, v_i, w_i))$ is a solution of the factorization problem of $s \mid \varphi$ by $v \mid \psi$;
(ii)[completeness] For each valuation $\mu\nu$ such that $\mu\nu \models_{\mathrm{PrA}} (\varphi \wedge \psi)$, either $\exists i$ such that $\mu\nu \models_{\mathrm{PrA}} \theta_i$, or $\mu\nu(v)$ is not a factor of $\mu\nu(s)$.

Theorem 1. *Given two parameterized words $v \mid \psi$ and $s \mid \varphi$ in this order,* Factorization *returns a finite (possibly empty), complete factorization of $s \mid \varphi$ by $v \mid \psi$.*

Proof. (sketch) Termination; we interpret a disjunction of factorization formulas by the multiset of the interpretations of its disjuncts. The interpretation of a disjunct $(u, v, w, s \mid \varphi)$, where w, s are assumed w.l.o.g. to be reduced word-expressions, is defined as the pair $(k + l, m + n)$ of natural numbers, compared lexicographically, in which: k, l are the number of factors of the form $(x)^i$, with i a dependent variable, in w, s respectively, while m, n are the lengths of the longuest flat word prefix of w, s respectively. It is easy to see that **Finish** and **CheckFactor** (4,5) decrease $k+l$, **CheckFactor** (1,2,3) maintain $k+l$ and decrease $m+n$, while other rules can be easily taken care of separately. This shows termination, hence finiteness of the set of answers.
Soundness: it is implied by the two invariants maintained by the rules.
Completeness: first, there is one rule for each possible kind of word-expression for w and s. We justify **CheckFactor** (4), which is the most difficult rule. The first disjunct assumes $m = 0$ or $n = 0$, so we can then assume both $m \neq 0$ and $n \neq 0$. We reason of course (implicitely) on the instances of $(u, v, (x)^m w, (y)^n s \mid \varphi)$, since the algorithms **gcd** and **grpef** will compute them for us. By assumption, $|x| \leq |y|$, hence x is a prefix of y. There are then two cases: either x and y share a common "divisor" z (yielding two possibilities for eliminating one of them), or x "divides" y ($\mathbf{p}$ times with a non-empty reminder t), in which case it is only possible to eliminate x. □

Note that the factorizations rules do not treat parameters differently from dependent variables. So far, the difference between both is only in the semantics. This suggests that the framework should scale to trees using existing toolboxes or variants.

4.3 Intersection, equality and left-overlaps

All these operations can be derived from the previous algorithm.

Intersection. Intersection is a stepping stone for deciding equality. The problem is to compute a description of the words which are common instances of two given parameterized words $u \mid \varphi$ and $v \mid \psi$. The difference with factorization is that the prefix and the suffix must be both empty. It therefore suffices to modify the **Start** rule, which conclusion should be $(\epsilon, \epsilon, w_0, s_0 \mid \varphi \wedge \psi)$ (therefore eliminating the need for the **FindPref** rules, the **Finish (2)** rule which should output $\bot$ as **Finish (1)**, and the **Out** rule in which u_i and v_i should be ϵ, and the conclusion the disjunction $\bigvee_i \varphi_i$. It is then immediate to see that we can simplify the format of formulas in this case, keeping only a triple $(w, s \mid \varphi)$, which we can of course write as $(w = s \mid \varphi)$.

Equality. We need to decide whether two disjunctive parameterized words $u_0 \mid \varphi_0 \vee \ldots u_m \mid \varphi_m$ and $v_0 \mid \varphi_0 \vee \ldots v_n \mid \varphi_n$ have *exactly* the same set of instances. We assume wlog that for all pairs (i, j), $u_i \mid \varphi_i$ and $v_i \mid \varphi_j$ have different sets of dependent variables. In the case of two parameterized words, we can apply **Intersection** to $(u_0 \mid \varphi_0)$ and $(v_0 \mid \psi_0)$, and check the equivalence in PrA of the obtained formula with the starting one $\varphi_0 \wedge \psi_0$. In the case of a disjunction of parameterized words, we can apply **Intersection** to the $(n + 1) \times (m + 1)$ equality problems $u_i \mid \varphi_i, v_j \mid \psi_j$, resulting in $(m + 1) \times (n + 1)$ constraints $\theta_{i,j}$, and then check that $\bigwedge_{i,j} \varphi_i \wedge \psi_j$ is equivalent in PrA to $\bigwedge_i (\bigvee_j \theta_{(i,j)}) \wedge \bigwedge_j (\bigvee_i \theta_{(i,j)})$.

Consider the two words $a(ba)^i \mid i \leq N$ and $(ab)^j a \mid j \leq N$. We explain the use of the rules in words, and in cases of disjunctions, treat the disjuncts in turn.

- Initial formula: $a(ba)^i = (ab)^j a \mid i \leq N \ \wedge \ j \leq N$;
- We split on $j = 0$, using **CheckFactor** (2); in the branch $j > 0$, we simplify the head occurrence of a and permute the word under exponent, yielding the result:
  $(a(ba)^i = a \mid j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N) \vee ((ba)^i = (ba)^{j-1}ba \mid j > 0 \ \wedge \ \ldots)$;
- Second, we simplify a in the obtained first disjunct, and get:
  $((ba)^i = \epsilon \mid j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N) \vee ((ba)^i = (ba)^{j-1}ba \mid j > 0 \ \wedge \ \ldots;$
- Applied to the first disjunct, the rule **Finish** (2) forces the value $i = 0$, yielding:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee ((ba)^i = (ba)^{j-1}ba \mid j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- We now apply the rule **CheckFactor** (4) to the second disjunct (using **gcd**):
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee \epsilon = (ba)^{j-1-i}ba \mid j - 1 - i \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N$
  $\vee (ba)^{i-j+1} = ba \mid i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- The second disjunct now simplifies away by using **Finish** (1), then **Elim**, yielding:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee (ba)^{i-j+1} = ba \mid i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- **CheckFactor** (3) now applies, hence we get:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee \epsilon = ba \mid i - j + 1 = 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N$
  $\vee (ab)^{i-j}a = a \mid i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- Using now **Finish** (2), this simplifies to:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee (ab)^{i-j}a = a \mid i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- **CheckFactor** (3) now applies again resulting in:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee a = a \mid i = j \ \wedge \ i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N$
  $\vee (ba)^{i-j-1}ba = \epsilon \mid i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- The third disjunct now simplifies away by using **Finish** (1), then **Elim**, yielding:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee a = a \mid i = j \ \wedge \ i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- Using **CheckFactor** (1), we finally get:
  $(\epsilon = \epsilon \mid i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee \epsilon = \epsilon \mid i = j \ \wedge \ i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N;$
- which yields the result by using **Output**:
  $(i = 0 \ \wedge \ j = 0 \ \wedge \ i \leq N \ \wedge \ j \leq N)$
  $\vee \, i = j \ \wedge \ i - j \geq 0 \ \wedge \ i - j + 1 \geq 0 \ \wedge \ j > 0 \ \wedge \ i \leq N \ \wedge \ j \leq N.$

The above formula is equivalent to $i = j \ \wedge \ j \leq N \ \wedge \ i \leq N$ in PrA, which expresses the *precise relationship* between the instances of the equal parameterized words.

Left-overlaps. A *left-overlap* of the word s over the word t is any triple (u, v, w) such that $s = uv$ and $t = vw$. Complete sets of overlaps are then disjunctions of quadruples $(u_i, v_i, w_i \mid \theta_i)$ satisfying an induced soundness and completeness condition as before.

The algorithm for computing a *complete set of left-overlaps* of $w_0 \mid \varphi$ over $s_0 \mid \psi$ is very similar to the one for computing a complete factorization, using the same quadruples (u, v, w, s), the same initialization phase, the same search for a prefix u of w_0 before to start the comparison between the obtained suffix w of w_0 and s_0, the same rules for computing the common part v, and maintaining the same invariants $s_0 = uvw$ and $vs = w_0$. The only difference is that the "suffix" w must be empty in the end. The corresponding modifications of the **Finish** rules is left to the reader.

5 Parameterized rewriting

We are now ready for investigating properties of parameterized rewrite systems.

Definition 1 *A* parameterized rewrite rule *is a triple $l \to r \mid \varphi$ made of a lefthand side word-expression l, a righthand side word-expression r and a constraint φ such that $l \mid \varphi$ and $r \mid \varphi$ are parameterized words and $\mathcal{V}ar(l, r) \subseteq \mathcal{V}ar(\varphi)$.*
A parameterized rewrite system *is a set of parameterized rewrite rules $\{l_i \to r_i \mid \varphi_i\}_i$. We shall assume w.l.o.g. that for all $i \in [1..n]$, $\mathcal{V}ar\mathcal{P}(\varphi_i) = \mathcal{P}$.*

A parameterized rewrite system R denotes an infinite family of finite word rewrite systems $\{[\![R_\nu]\!]\}_{\nu \in \mathcal{P} \to \mathbb{N}^{|\mathcal{P}|}}$ defined as follows:

$$
\begin{aligned}
[\![R]\!] &= \{[\![R_\nu]\!] \mid \nu \in \mathcal{P} \to \mathbb{N}^{|\mathcal{P}|}\} \\
R_\nu &= \{(\nu(l) \to \nu(r) \mid \nu(\varphi)) \ \mid \ l \to r \mid \varphi \in R\} \\
[\![R_\nu]\!] &= \{\mu(\nu(l)) \to \mu(\nu(r)) \ \mid \ l \to r \mid \varphi \in R, \text{ and } \mu \in I \to \mathbb{N}^{|I|} \text{ s.t. } \mu \models_{\mathrm{PrA}} \nu(\varphi)\}
\end{aligned}
$$

The rewrite system $[\![R_\nu]\!]$ is called an *instance* of R.
Consider the parameterized rewrite systems $R = \{a_i^i \to a_j \mid 0 \leq i < j \leq N\}$ and $R' = \{(u_i u_j)^N \to u_j u_i \mid i - j \geq 2 \ \wedge \ i, j < N\}$. We have:

$$R_{N=2} = \{a_i^i \to a_j \mid 0 \leq i < j \leq 2\} \quad [\![R_{N=2}]\!] = \{\epsilon \to a_1; \epsilon \to a_2; a_1 \to a_2\}$$

$$R'_{N=5} = \{(u_i u_j)^5 \to u_j u_i \mid i - j \geq 2 \ \wedge \ i, j < 5\}$$

$$[\![R'_{N=5}]\!] = \left\{ \begin{array}{l} u_3u_1u_3u_1u_3u_1u_3u_1u_3u_1 \to u_1u_3, \\ u_4u_1u_4u_1u_4u_1u_4u_1u_4u_1 \to u_1u_4, \\ u_4u_2u_4u_2u_4u_2u_4u_2u_4u_2 \to u_2u_4 \end{array} \right\}$$

There are three ways to understand R: as a set of parameterized rewrite rules operating on parameterized words ; and for each value ν of the parameters, either as a set of parameterized rules R_ν with dependent variables only depending on integer *values*, or as a set $[\![R_\nu]\!]$ of rules on words. Rewriting can then be defined at several levels: on words with rules (both having possibly exponents), on parameterized words with rules, on words with parameterized rules, etc. These definitions need be consistent at all levels, that is, be related by commutation lemmas in order to capture families of critical pairs in $[\![R_\nu]\!]$ and their joinability by critical pairs in R_ν and their joinability, and the latter by critical pairs in R and their joinability. This requires a careful definition of parameterized rewriting, as shown by the coming example.

Consider the parameterized rewrite system $R = \{a^iba^i \to a^ib^i \mid i \leq N, aba \to \epsilon\}$. The parameterized word $a^iba^i \mid i \leq N$ can be seen as the disjunction $(b \mid i = 0) \vee (a^{i-1}abaa^{i-1} \mid i > 0 \ \wedge \ i \leq N)$, and therefore aba is a factor of $a^iba^i \mid i \leq N$ subjected to the additional constraint $i > 0$: we can rewrite the word a^iba^i with the rule $aba \to \epsilon$ if $i > 0$, but we cannot if $i = 0$. The parameterized word $a^iba^i \mid i \leq N$ can therefore be rewritten with the rule $aba \to \epsilon$ into the disjunctive parameterized word $(a^iba^i \mid i \leq N \ \wedge \ i = 0) \vee (a^{i-1}a^{i-1} \mid i \leq N \ \wedge \ i > 0)$, that is $(b \mid i = 0) \vee (a^ia^i \mid 0 < i \leq N)$, hence capturing both cases at once.

Definition 5. *Given a parameterized word $s \mid \varphi$ and a parameterized rewrite rule $l \to r \mid \psi$ such that (i) $\mathcal{V}ar\mathcal{P}(\psi) \subseteq \mathcal{V}ar\mathcal{P}(\varphi)$ and (ii) $\vee_i(u_i, v_i, w_i \mid \theta_i)$ is a complete, non-empty factorization of $s \mid \varphi$ by $l \mid \psi$, then $s \mid \varphi$* rewrites *with $l \to r \mid \psi$ to the parameterized disjunctive word $\bigvee_i(u_irw_i \mid \theta_i) \vee (s \mid \varphi \ \wedge \ \bigwedge_i(\neg\theta_i))$, written $s \mid \varphi \longrightarrow_{l\to r|\psi} (\ \bigvee_i(u_irw_i \mid \theta_i) \vee (s \mid \varphi \ \wedge \ \bigwedge_i(\neg\theta_i))$.*
The formulas $\bigvee_i(u_irw_i \mid \theta_i)$ and $\bigwedge_i(\neg\theta_i)$ characterize respectively the positive *and* negative *parts of the rewrite.*
A rewriting step is called uniform *if the righthand side is a (single) parameterized word, i.e., the factorization of $s \mid \varphi$ by $l \mid \psi$ has the form $(u, l, w \mid \theta)$ with $\varphi \models_{\mathrm{PrA}} \theta$.*
Rewriting by a set of parameterized rules is defined as expected.

Note that we do not allow rewriting with an empty factorization, which would result in a trivial rewrite step. In general, the result of rewriting a parameterized word by a parameterized rule is a disjunction of parameterized words by definition of factorization: first, l does not appear exponentiated in the factorization of $s \mid \varphi$ by $l \mid \psi$; second, the finite number of possible interpretations for the dependent variables, given a value of the parameter variables, is of course maintained as a result of the factorization process. Further, by definition of a factorization, $\theta_i \not\models_{\mathrm{PrA}} \bot$ and therefore the word $u_irw_i \mid \theta_i$ has a non-empty interpretation. On the other hand, it is quite possible that θ and φ are equivalent in Presburger arithmetic in case of a uniform rewrite step, in which case the rewriting result is the single parameterized word $urw \mid \theta$.
We now relate parameterized rewriting with R operating on a parameterized word $u \mid \varphi$ with rewriting the corresponding instances of $u \mid \varphi$ with $[\![R_\nu]\!]$. We therefore skip the intermediate level of rewriting with R_ν. This relationship is expressed by the following key lemma, which will be a main tool in our study of parameterized rewriting:

Lemma 1 (Lifting). *Let $s \mid \varphi$ be a parameterized word. Then, $\mu(\nu(s)) \longrightarrow_{[\![R_\nu]\!]} t$ for some rule instance $\mu(\nu(l)) \to \mu(\nu(r))$ of $l \to r \mid \psi \in R$ iff $(\ \bigvee_i u_i, l, w_i \mid \theta_i)$ is a complete factorization of $s \mid \varphi$ by $l \mid \psi$, $\mu\nu \models_{\mathrm{PrA}} \theta_i$ for some i, and $t = \mu(\nu(u_irw_i))$.*

Proof. Follows easily from Definitions 4 and 5: □

Lifting takes care of positive rewrites. A negative rewrite is nothing but an artefact which reduces the set of instances of a parameterized word without changing the word itself. Negative rewrites play a central role for *derivations*, since they allow us to capture at once all possible derivations on words by derivations on parameterized words.
We write $t \mid \varphi \longrightarrow_R^* s \mid \psi$ for the reflexive, transitive closure of $\longrightarrow_R$, called a *derivation*, and $t \mid \varphi \longleftrightarrow_R^* s \mid \psi$ for its reflexive, symmetric transitive closure, called a *conversion*, for which R is as usual interpreted as a set of equations.

5.1 Termination of parameterized rewriting

Unfortunately, termination of parameterized rewriting does not characterize termination of its instances, as shown by the coming example of a (plain) rewrite system which terminates trivially on words, but does not on parameterized words. Let $R = \{ab \to \epsilon\}$. We have:
$(ab)^i \mid i \leq N \longrightarrow_{ab\to\epsilon} (ab)^k(ab)^l \mid i > 0 \ \wedge \ i = k + l + 1 \ \wedge \ i \leq N = (ab)^j \mid i > 0 \wedge i = j+1 \wedge i \leq N \longrightarrow_{ab\to\epsilon} (ab)^m(ab)^n \mid i > 0 \wedge j > 0 \wedge j = m+n+1 \wedge i > 0 \ \wedge \ i = j + 1 \ \wedge \ i \leq N \longrightarrow_{ab\to\epsilon} \dots$
We cannot therefore expect a parameterized system to be terminating on parameterized words in general (actually in most cases), but we are indeed only interested in the termination of its instances on words. In the above case, we can trivially show that $ab \to \epsilon$ terminates on words. Yet, it may be difficult for parameterized rules. A simple remark shows however that automation is at reach. Consider the two rewrite systems $R = \{a^i \to \epsilon \mid i \leq n\}$ and $S = \{a^i \to \epsilon \mid 1 \leq i \leq n\}$ which describe the same set of instances on words except for the non-terminating instance $\epsilon \to \epsilon$ of R which is not an instance of S. As long as the parameterized rule $a^i \to \epsilon$ does not degenerate (here, into the rule $\epsilon \to \epsilon$), it can be seen as a terminating rule over word-expressions built from the alphabet, concatenation, and exponentiation. The degenerated cases can easily be computed by solving equations of the form $(x)^i = \epsilon$ for x a non-empty word, therefore adding $i = 0$ to the constraint of the considered rule. In the previous two examples, we get the parameterized rule instances $\{\epsilon \to \epsilon \mid i \leq n \ \wedge \ i = 0\}$ and $\{\epsilon \to \epsilon \mid 1 \leq i \leq n \ \wedge \ i = 0\}$, but the second disappears. More generally,

Theorem 2. *Let R a parameterized rewriting system and $C(R)$ be obtained as follows:*
$C(R \cup \{l \to r \mid \theta\}) = C(R) \cup C(l \to r \mid \theta)$
$C(l = x_1(y_1)^{i_1} \dots (y_n)^{i_n} x_{n+1} \to r \mid \theta) = \{(l{\downarrow} \to r \mid \theta \wedge \bigwedge_{j\in J} i_j \neq 0 \mid J \subseteq [1:n]\}$
where $l{\downarrow}$ is obtained from l by replacing i_k by 0 in l for $k \in [1..n] \setminus J$, then $(y_k)^0$ by ϵ and finally eliminating superflous ϵ's.
Then, all instances of R terminate on words if there exists a well-founded rewrite ordering $\succ$ on word-expressions such that:
(i) $s \succ t$ implies $\mu\nu(usv) \succ \mu\nu(utv)$ for all word-expressions u, v and valuations $\mu\nu$;
(ii) $x \succ y$ implies $\mu\nu((x)^i) \succ \mu\nu((y)^i)$ for all valuations $\mu\nu$ such that $\mu\nu \models_{\mathrm{PrA}} i \neq 0$;
(iii) $l \succ r$ for all rules $(l \to r \mid \theta) \in C(R)$ for which $\theta \not\models_{\mathrm{PrA}} \bot$.

Proof. (sketch). First, an arbitrary derivation on words with some rewrite system $[\![R_\nu]\!]$ is also a derivation with $[\![C(R)_\nu]\!]$, which can be seen as a parameterized derivation with $C(R)$ by using the Lifting Lemma. Condition (iii) then allows us to make it an ordered sequence on parameterized-words. Conditions (i,ii) allow finally to move the ordering on parameterized-words back to the original sequence of words. □

Existing techniques apply directly provided they satisfy conditions (i,ii), which is normally the case since exponents cannot be instantiated by 0 in a rule of $C(R)$. The use of exponential interpretations would be a natural choice by allowing to interpret exponentiation on words by arithmetic exponentiation. Polynomial interpretations also do.
In the previous two examples, we ended up checking the pairs $a^i \succ \epsilon, \epsilon \succ \epsilon$ which fails for any interpretation, and $a^i \succ \epsilon$ which succeeds, using for example as interpretation the number of letters in a parameterized-word.

5.2 Confluence of parameterized rewriting

Confluence raises other difficulties. For an example, let $R = \{ac \to \epsilon, def \to \epsilon\}$. R is confluent on words, since it is terminating and has no critical pairs. But positive rewriting (stressed by using $\Longrightarrow$) with R is not confluent on parameterized-words:
$ab^icde^if \mid i \leq N \Longrightarrow_{ac\to\epsilon} df \mid i = 0 \ \wedge \ i \leq N$ and
$ab^icde^if \mid i \leq N \Longrightarrow_{def\to\epsilon} abc \mid i = 1 \ \wedge \ i \leq N$, two words which cannot be joined. Observe however that the factorization constraints $i = 0$ and $i = 1$ are incompatible: the word ab^icde^if does not have instances rewritable by both rules. And indeed, the problem disappears when positive and negative rewriting are carried along together: $ab^icde^if \mid i \leq N \longrightarrow_{ac\to\epsilon} (df \mid i = 0 \ \wedge \ i \leq N) \vee (ab^icde^if \mid i \neq 0 \ \wedge \ i \leq N)$, and $ab^icde^if \mid i \leq N \longrightarrow_{def\to\epsilon} (abc \mid i = 1 \ \wedge \ i \leq N) \vee (ab^icde^if \mid i \neq 1 \ \wedge \ i \leq N)$, and since these rewrites are sort of orthogonal, they both rewrite to a word equal to $(df \mid i = 0 \ \wedge \ i \leq N) \vee (abc \mid i = 1 \ \wedge \ i \leq N) \vee (ab^icde^if \mid i \neq 0 \ \wedge \ i \neq 1 \ \wedge \ i \leq N)$.

Theorem 3. *The instances of a parameterized rewrite system R are confluent (resp. locally confluent) on words iff parameterized rewriting with R is confluent (resp. locally confluent) on parameterized words.*

Proof. Follows from the lifting Lemma 1. □

We now turn our attention to a critical pair analysis of local confluence. Consider the parameterized rewrite system $R = \{a^iba^i \to a^ib^i \mid i \leq N, aba \to \epsilon\}$. Since aba is a factor of $a^iba^i \to a^ib^i \mid i \leq N$ under the additional constraint $i > 0$, the lefthand side of the first rule can be rewritten by the second rule. And since $a^iba^i \to a^ib^i \mid i \leq N$ is a factor of aba under the additional constraint $i = 0$, the lefthand side of the second can be rewritten by the first. We can indeed describe all critical pairs of each rewrite system $[\![R]\!]_\nu$ by computing factorizations and left-overlaps.

Consider the parameterized system $R = \{a_Na_i \to a_N \mid 0 < i < N\}$, which all instances are critical pair-free. Let $a_Na_i \to a_N \mid 0 < i < N$ and $a_Na_j \to a_N \mid 0 < j < N$ be two arbitrary rules in R. Their instances belong to the same system $[\![R_n]\!]$ provided they share the same parameter N, while the dependent variable is renamed (otherwise, we would have the same rule). It is easy to see that a_Na_i and a_Na_j cannot overlap unless $i = j$, in which case we have a trivial critical pair. On the other hand, rules of different systems $[\![R_n]\!]$ and $[\![R_m]\!]$ do overlap and yield non-joinable critical pairs. This example shows the practical need of the two kinds of variables in our model.

Definition 2 *Let $l \to r \mid \varphi$ and $g \to d \mid \psi$ be two rules of a parameterized rewriting system R such that $\mathcal{V}arI(\varphi) \cap \mathcal{V}arI(\psi) = \emptyset$.*
(i) Let $\vee_i(u_i, g_i, v_i \mid \theta_i)$ be a complete factorization of $l \mid \varphi$ by $g \mid \psi$. The pair $(r \mid \bigvee_i \theta_i, \ \bigvee_i u_idv_i \mid \theta_i)$ is called a critical pair *of $g \to d \mid \psi$ on $l \to r \mid \varphi$;*
(ii) Let $(\bigvee_i(u_i, v_i, w_i \mid \theta_i)$ be a complete left-overlap of $l \mid \varphi$ on $g \mid \psi$. The pair $(\bigvee_i u_ir \mid \theta_i, \ \bigvee_i dw_i \mid \theta_i)$ is called a critical overlapping pair *of $l \to r \mid \varphi$ on $g \to d \mid \psi$.*

Lemma 2. *For each valuation ν, the set of critical pairs in $[\![R_\nu]\!]$ is the set of corresponding instances of the critical and overlapping pairs in R.*

Proof. Follows again from the lifting Lemma 1. □

Lemma 3. *Parameterized rewriting with R is locally confluent iff all critical and overlapping pairs of R are joinable by parameterized rewriting.*

Proof. (sketch). The only if direction is straightforward, but the converse is more subtle. Let $s \mid \varphi \longrightarrow (u \mid \theta) \vee (s \mid \varphi \wedge \neg\theta)$, and $s \mid \varphi \longrightarrow (v \mid \gamma) \vee (s \mid \varphi \wedge \neg\gamma)$. Then,
$(u \mid \theta) \vee (s \mid \varphi \wedge \neg\theta) =$
$(u \mid \theta \wedge \gamma) \vee (u \mid \theta \wedge \neg\gamma) \vee (s \mid \varphi \wedge \neg\theta \wedge \gamma) \vee (s \mid \varphi \wedge \neg\theta \wedge \neg\gamma) =$
$(u \mid \theta \wedge \gamma) \vee (u \mid \theta \wedge \neg\gamma) \vee (s \mid \neg\theta \wedge \gamma) \vee (s \mid \varphi \wedge \neg\theta \wedge \neg\gamma)$.
Similarly (but we change the order of disjuncts), $(v \mid \gamma) \vee (s \mid \varphi \wedge \neg\gamma) =$
$(v \mid \theta \wedge \gamma) \vee (s \mid \theta \wedge \neg\gamma) \vee (v \mid \neg\theta \wedge \gamma) \vee (s \mid \varphi \wedge \neg\theta \wedge \neg\gamma)$.
We see that the first disjuncts of both expressions are joinable by a critical pair analysis; the second disjuncts are joinable by a rewrite step with $g \to d \mid \psi$; the third disjuncts are joinable by a rewrite step with $l \to r \mid \varphi$; and the fourth disjuncts are equal. □

Using Lemmas 2, 3 and Newman's lemma, we get the main practical result of this work:

Theorem 4. *Assume the rewrite systems $[R_\nu]$ on words are terminating. Then, they are confluent iff all critical and overlapping pairs of R are joinable.*

Unfortunately, this does not imply the decidability of confluence or local-confluence even under our termination assumption since parameterized rewriting may be non-terminating, and therefore the usual joinability check may not terminate for some pairs. We don't know, however, whether joinability is decidable or undecidable in our model of parameterized rewriting. This problem is left open.

5.3 Implementation and example

As an example, consider the presentation of dihedral groups of order $N > 1$ by $R_N = \{s^2 \to \epsilon; sr \to r^{N-1}; r^N \to \epsilon\}$, which we input to the tool C*i*ME2[3] in the format:

```
let N = parameters "N";
let S = pword_signature N "s | {N>=2}; r | {N>=2}" ;
let R = psrs S "s s -> | {N>=2};
                s r -> r^{N-1} s | {N>=2} ;
                r^{N} -> | {N>=2};"
        ;
let Rnorm = psrs S "s r^{k} -> r^{N-k} s |{N>=2 /\
                                          1<=k<=N-1 };";
pconfluent_ext R Rnorm;
```

The procedure `Rnorm` iterates the second rule, allowing us to overcome some limitations of the current implementation to uniform rewrite proofs introduced in definition 5. C*i*ME2 computes 13 overlapping pairs joinable immediately while 4 others need a few (uniform) rewrite steps. In case $N = 1$, the lefthand side sr of the second rule becomes reducible by the third rule $r^2 \to \epsilon$, while this is not the case if $N > 1$: the restriction $N > 1$ present in the C*i*ME2 specification allows one to comply with the restriction to uniform rewrite proofs: C*i*ME2 is not able to show the joinability of all critical pairs if $N > 1$ is changed to $N > 0$.

6 Conclusion

We have defined a framework of parameterized rewrite systems operating on parameterized words for describing infinite families of rewriting systems on words and mechanize their study, using a sophisticated rewriting toolkit for parameterized words.
We have given a method for showing termination of all instances of a parameterized system R by using an adequate ordering for checking the rules of a transformed system $C(R)$, therefore allowing to reuse existing tools.
We have reduced confluence of all instances of a parameterized rewriting system to the joinability of its finitely many critical or overlapping pairs under termination of the instances. Whether joinability can be decided in this context merits further investigations.
We could have made the choice of a more abstract framework based on parameterized structures for representing infinite families of rewriting systems on that structure, assuming the necessary toolkit for the parameterized structure, and then apply the abstract results to parameterized words as described here or parameterized trees as described in [2, 5]. We indeed conjecture (but have not checked) that our approach scales up, opening up interesting applications for example to multicore hardware modelisations.
Formalisms for representing families of terms, equations or rules fall in two categories: tree automata and term schematizations. Our formalism of parameterized words belongs to the second but its strong closure properties suggest to blend it with automata in the line of [9], a recent bridge between both kinds of worlds.

Acknowledgments: to Evelyne Contejean and Claude Marché for discussions with the second author and to the several anonymous referees who helped shaping this paper.

References

1. R.V. Book and F. Otto. *String Rewriting Systems*. Text and monographs in Computer Science. Springer-Verlag, Berlin, 1993.
2. Hubert Comon. On unification of terms with integer exponents. *Math. Systems Theory*, 28:67–88, 1995.
3. Evelyne Contejean, Claude Marché, Benjamin Monate, and Xavier Urbain. Cime version 2. Technical report, Université Paris-Sud, 2000.
4. Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.
5. Miki Hermann and Roman Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theoretical Computer Science*, 176(1–2):111–158, 1997.
6. Jan Wilhelm Klop et alii. Term rewriting systems. In *Cambridge Tracts in Theoretical Computer Science*, volume 55. Cambridge University Press, 2003.
7. Philippe Le Chenadec. *Canonical forms in finitely presented algebras*. Pitman, London, 1986.
8. Philippe Le Chenadec. Analysis of dehn's algorithm by critical pairs. *Theoretical Computer Science*, 51(1-2):27–52, 1987.
9. Nicolas Peltier. A unified view of tree automata and term schematisations. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *IFIP TCS*, volume 273 of *IFIP*, pages 491–505. Springer, 2008.
10. Gernot Salzer. On unification of infinite sets of terms and its applications. In Voronkov, editor, *Proc. LPAR 92, LNCS 624*, pages 409–421, 1992.