# Automatic Decidability for Theories with Counting Operators

**(Presentation-only paper)**

Elena Tushkanova[1,2], Christophe Ringeissen[1], Alain Giorgetti[1,2], and Olga Kouchnarenko[1,2]

[1] Inria, Villers-les-Nancy, F-54600, France
[2] FEMTO-ST Institute (UMR 6174), University of Franche-Comté,
Besançon, F-25030, France

**The full paper will appear in the proceedings of the 24th International Conference on Rewriting Techniques and Applications (RTA 2013), published as LIPIcs proceedings,** `http://www.dagstuhl.de/en/publications/lipics`

## 1 Introduction

Decision procedures for satisfiability modulo background theories of classical datatypes are at the core of many state-of-the-art verification tools. Designing and implementing these satisfiability procedures remains a very hard task. To help the researcher with this time-consuming task, an important approach based on rewriting has been investigated in the last decade [2, 1]. The rewriting-based approach allows building satisfiability procedures in a flexible way by using a general calculus for automated deduction, namely the paramodulation calculus [9] (also called superposition calculus). The paramodulation calculus is a refutation-complete inference system at the core of all equational theorem provers. In general this calculus provides a semi-decision procedure that halts on unsatisfiable inputs by generating an empty clause, but may not terminate on satisfiable ones. However, it also terminates on satisfiable inputs for some theories axiomatising standard datatypes such as arrays, lists, etc, and thus provides a decision procedure for these theories. A classical termination proof consists in considering the finitely many cases of inputs made of the (finitely many) axioms and any set of ground flat literals. This proof can be done by hand, by analysing the finitely many forms of clauses generated by saturation, but the process is tedious and error-prone. To simplify this process, a schematic paramodulation calculus has been developed [5] to build the schematic form of the saturations. It can be seen as an abstraction of the paramodulation calculus: If it halts on one given abstract input, then the paramodulation calculus halts for all the corresponding concrete inputs. More generally, schematic paramodulation is a fundamental tool to check important properties related to decidability and combinability [6].

To ensure efficiency, it is very useful to have built-in axioms in the calculus, and so to design paramodulation calculi modulo theories. This is particularly important for arithmetic fragments due to the ubiquity of arithmetics in applications of formal methods. For instance, paramodulation calculi have been developed for Abelian Groups [4, 7] and Integer Offsets [8]. In [8], the termination of paramodulation modulo Integer Offsets is proved manually. Therefore, there is an obvious need for a method to automatically prove that an input theory admits a decision procedure based on paramodulation modulo Integer Offsets.

In this paper, we introduce theoretical underpinnings that allow us to automatically prove the termination of paramodulation modulo Integer Offsets. To this aim, we design a new schematic paramodulation calculus to describe saturations modulo Integer Offsets. Our approach requires a new form of schematization to cope with arithmetic expressions. The interest of schematic paramodulation relies on a correspondence between a derivation using (concrete) paramodulation and a derivation using schematic paramodulation: Roughly speaking, the set of derivations obtained by schematic paramodulation over-approximates the set of derivations obtained by (concrete) paramodulation.

Our approach has been developed and validated thanks to a proof system [10] implemented in the rewriting logic-based environment Maude.

## 2 Paramodulation Calculus

As in [10] we consider only unitary clauses, i.e. clauses composed of at most one literal. The *Unitary Paramodulation Calculus*, denoted by $\mathcal{UPC}$ [10] corresponds to the standard paramodulation calculus restricted to the case of unit clauses.

The paramodulation-based calculus $\mathcal{UPC}_I$ defined in [8] adapts the paramodulation calculus $\mathcal{UPC}$ to the theory of Integer Offsets, so that it can serve as a basis for the design of decision procedures for Integer Offsets extensions. Technically, the axioms of the theory of Integer Offsets are directly integrated in the simplification rules of $\mathcal{UPC}_I$. The theory of Integer Offsets is axiomatized by the set of axioms $\{\forall X.\ \mathsf{s}(X) \neq 0,\ \forall X, Y.\ \mathsf{s}(X) = \mathsf{s}(Y) \Rightarrow X = Y,\ \forall X.\ X \neq \mathsf{s}^n(X)\ \text{ for all }\ n \geq 1\}$ over the signature $\Sigma_I := \{0 : \textsc{int}, \mathsf{s} : \textsc{int} \rightarrow \textsc{int}\}$. Compared to [3], our theory of Integer Offsets does not consider the predecessor function. Following [8, Section 5], a possible Integer Offsets extension is the theory $LLI$ of lists with length whose signature is $\Sigma_{LLI} = \{\mathsf{car} : \textsc{lists} \rightarrow \textsc{elem}, \mathsf{cdr} : \textsc{lists} \rightarrow \textsc{lists}, \mathsf{cons} : \textsc{elem} \times \textsc{lists} \rightarrow \textsc{lists}, \mathsf{len} : \textsc{lists} \rightarrow \textsc{int}, \mathsf{nil} :\rightarrow \textsc{lists}, 0 :\rightarrow \textsc{int}, \mathsf{s} : \textsc{int} \rightarrow \textsc{int}\}$ and whose set of axioms $Ax(LLI)$ is $\{\mathsf{car}(\mathsf{cons}(X, Y)) = X, \mathsf{cdr}(\mathsf{cons}(X, Y)) = Y, \mathsf{len}(\mathsf{cons}(X, Y)) = \mathsf{s}(\mathsf{len}(Y)), \mathsf{cons}(X, Y) \neq \mathsf{nil}, \mathsf{len}(\mathsf{nil}) = 0\}$.

## 3 Schematic Paramodulation

The *Schematic Unitary Paramodulation Calculus* $\mathcal{SUPC}$ is an abstraction of $\mathcal{UPC}$. Indeed, any concrete saturation computed by $\mathcal{UPC}$ can be viewed as an instance of an abstract saturation computed by $\mathcal{SUPC}$ [6, Theorem 2]. Hence, if $\mathcal{SUPC}$ halts on one given abstract input, then $\mathcal{UPC}$ halts for all the corresponding concrete inputs. More generally, $\mathcal{SUPC}$ is an automated tool to check properties of $\mathcal{UPC}$ such as termination, stable infiniteness and deduction completeness [6].

$\mathcal{SUPC}$ is almost identical to $\mathcal{UPC}$, except that literals are constrained by conjunctions of atomic constraints of the form $const(x)$ which restricts the instantiation of the variable $x$ by only constants. An implementation of *Paramodulation* and *Schematic Paramodulation* calculi $\mathcal{UPC}$ and $\mathcal{SUPC}$ is presented in [10].

In the following, we extend the schematic calculus $\mathcal{SUPC}$ for $\mathcal{UPC}$ to get a schematic calculus for $\mathcal{UPC}_I$, named $\mathcal{SUPC}_I$.

## 4 Schematic Paramodulation Calculus for Integer Offsets

This section introduces a new schematic calculus named $\mathcal{SUPC}_I$. It is a schematization of $\mathcal{UPC}_I$ taking into account the axioms of the theory of Integer Offsets within a framework based on schematic paramodulation [6, 10].

The theory of Integer Offsets allows us to build arithmetic expressions of the form $\mathsf{s}^n(t)$ for $n \geq 1$. The idea investigated here is to represent all terms of this form in a unique way. To this end, we consider a new operator $\mathsf{s}^+ : \text{INT} \to \text{INT}$ such that $\mathsf{s}^+(t)$ denotes the infinite set of terms $\{s^n(t) \mid n \geq 1\}$. Let us introduce the notions of schematic clause and instance of schematic clause handled by $\mathcal{SUPC}_I$. These notions extend the ones used in [6] for the schematization of $\mathcal{PC}$.

**Definition 1 (Schematic Clause)** *A* schematic clause *is a constrained clause built over the signature extended with* $\mathsf{s}^+$. *An* instance of a schematic clause *is a constraint instance where each occurrence of* $\mathsf{s}^+$ *is replaced by some* $\mathsf{s}^n$ *with* $n \geq 1$.

The calculus $\mathcal{SUPC}_I$ takes as input a set of schematic literals, $G_0$, that represents all possible sets of ground literals given as inputs to $\mathcal{UPC}_I$:

$$\begin{aligned} G_0 = \{\bot, x = y \parallel const(x,y), x \neq y \parallel const(x,y), u = \mathsf{s}^+(v) \| \varphi\} \\ \cup \textstyle\bigcup_{f \in \Sigma_T} \{f(x_1, \ldots, x_n) = x_0 \parallel const(x_0, x_1, \ldots, x_n)\} \end{aligned}$$

where $u, v$ are flat terms of sort INT whose variables are all constrained (in $\varphi$), and $x, y$ are constrained variables of the same sort.

The calculus $\mathcal{SUPC}_I$ is depicted in Fig. 1. It re-uses most of the rules of $\mathcal{SUPC}$ – Figs. 1(a) and 1(b) – and complete them with one new contraction rule named *Schematic Deletion* and two reduction rules – presented in Fig. 1(c) – which are simplification rules for Integer Offsets.

Whenever a literal is generated by superposition or simplification, the rewrite system $R\mathsf{s}^+ = \{\ \mathsf{s}^+(\mathsf{s}(x)) \to \mathsf{s}^+(x),\ \mathsf{s}(\mathsf{s}^+(x)) \to \mathsf{s}^+(x),\ \mathsf{s}^+(\mathsf{s}^+(x)) \to \mathsf{s}^+(x)\ \}$ is applied eagerly to simplify terms containing $\mathsf{s}^+$. The rewrite system $R\mathsf{s}^+$ is also applied in the *Schematic Deletion* rule to implement a form of subsumption check via a morphism $\pi$ replacing all the occurences of $\mathsf{s}$ by $\mathsf{s}^+$ ($\pi(\mathsf{s}(t)) = \mathsf{s}^+(\pi(t))$ for any $t$, $\pi(x) = x$ if $x$ is a variable).

It is important to note that $\mathcal{SUPC}_I$ may diverge without the new *Schematic Deletion* rule. To illustrate this point, let us take a look at the theory of lists with length. In fact, the calculus generates a schematic clause $\mathsf{len}(a) = \mathsf{s}(\mathsf{len}(b)) \| const(a, b)$ which will superpose with a renamed copy of itself, i.e. with

*Superposition* $$\frac{l[u'] \bowtie r \| \varphi \qquad u = t \| \psi}{\sigma(l[t] \bowtie r \| \varphi \wedge \psi)}$$

**if** i) $\sigma(u) \not\leq \sigma(t)$, ii) $\sigma(l[u']) \not\leq \sigma(r)$, and iii) $u'$ is not an unconstrained variable.

*Reflection* $$\frac{u' \neq u \| \psi}{\bot}$$ **if** $\sigma(\psi)$ is satisfiable.

Above, $u$ and $u'$ are unifiable and $\sigma$ is the most general unifier of $u$ and $u'$.

(a) Schematic expansion inference rules

*Subsumption* $$\frac{S \cup \{L \| \psi, L' \| \psi'\}}{S \cup \{L \| \psi\}}$$

**if** a) $L \in Ax(T)$, $\psi = \emptyset$ and $L'$ is an instance of $L$; or b) $L' = \sigma(L)$, $\psi' = \sigma(\psi)$, where $\sigma$ is a renaming or a mapping from constrained variables to constrained variables.

*Simplification* $$\frac{S \cup \{C[l'] \| \varphi, l = r\}}{S \cup \{C[\sigma(r)] \| \varphi, l = r\}}$$

**if** i) $l = r \in Ax(T)$, ii) $l' = \sigma(l)$, iii) $\sigma(l) > \sigma(r)$, and iv) $C[l'] > (\sigma(l) = \sigma(r))$.

*Tautology* $$\frac{S \cup \{u = u \| \varphi\}}{S}$$

*Deletion* $$\frac{S \cup \{L \| \varphi\}}{S}$$ **if** $\varphi$ is unsatisfiable.

*Schematic Del.* $$\frac{S \cup \{C' \| \varphi, C[\mathsf{s}^+(t)] \| \psi\}}{S \cup \{C[\mathsf{s}^+(t)] \| \psi\}}$$

**if** $\sigma(\pi(C') \downarrow_{R\mathsf{s}^+}) = C[\mathsf{s}^+(t)]$, $\sigma(\varphi) = \psi$, for a renaming $\sigma$.

(b) Schematic contraction inference rules

*R1* $$\frac{S \cup \{\mathsf{s}(u) = \mathsf{s}(v) \| \varphi\}}{S \cup \{u = v \| \varphi\}}$$

*R2* $$\frac{S \cup \{\mathsf{s}(u) = t \| \varphi, \mathsf{s}(v) = t \| \psi\}}{S \cup \{\mathsf{s}(v) = t \| \psi, u = v \| \psi \wedge \varphi\}}$$ **if** $\mathsf{s}(u) > t, \mathsf{s}(v) > t, u > v$

Above, all the variables in $u, v, t$ are constrained.

(c) Schematic ground reduction inference rules

Fig. 1: Inference rules of $\mathcal{SUPC}_I$

$\mathsf{len}(a') = \mathsf{s}(\mathsf{len}(b'))\ \|const(a', b')$ to generate a schematic clause of a new form $\mathsf{len}(a) = \mathsf{s}(\mathsf{s}(\mathsf{len}(b')))\ \|const(a, b')$. Without the *Schematic Deletion* rule this process continues to generate deeper and deeper schematic clauses so that $\mathcal{SUPC}_I$ will diverge. The *Schematic Deletion* rule applies to the theory of lists with length since $G_0$ already contains the non-flat schematic literal $\mathsf{len}(a) = \mathsf{s}^+(\mathsf{len}(b))\|const(a, b)$.

As in [5, 6], we are interested in satisfying the following properties:

- Any clause in a saturation generated by the paramodulation calculus with any possible input is an instance of a schematic clause in a saturation generated by the schematic paramodulation calculus with the input $G_0$.
- The termination of the schematic paramodulation calculus with the input $G_0$ implies the termination of the paramodulation calculus with any possible input.

The new form of schematization introduced for arithmetic expressions requires adapting the proofs done for the standard case [11]. Our schematic paramodulation calculus for Integer Offsets provides us with an automatic proof method for the theories considered in [8], where the termination proofs are done manually.

## References

1. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures. *ACM Trans. Comput. Logic*, 10(1):1 – 51, 2009.
2. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *J. Inf. Comput*, 183(2):140 – 164, 2003.
3. Maria Paola Bonacina and Mnacho Echenim. On Variable-inactivity and Polynomial T-Satisfiability Procedures. *J. Log. Comput.*, 18(1):77–96, 2008.
4. G. Godoy and R. Nieuwenhuis. Superposition with completely built-in abelian groups. *Journal of Symbolic Computation*, 37(1):1–33, 2004.
5. C. Lynch and B. Morawska. Automatic decidability. In *LICS*, pages 7–16, Copenhagen, Denmark, July 2002. IEEE Computer Society.
6. C. Lynch, S. Ranise, C. Ringeissen, and D.-K. Tran. Automatic decidability and combinability. *J. Inf. Comput*, 209(7):1026–1047, 2011.
7. E. Nicolini, C. Ringeissen, and M. Rusinowitch. Combinable extensions of Abelian groups. In R. Schmidt, editor, *CADE*, volume 5663 of *LNCS*, pages 51–66. Springer, 2009.
8. E. Nicolini, C. Ringeissen, and M. Rusinowitch. Satisfiability procedures for combination of theories sharing integer offsets. In S. Kowalewski and A. Philippou, editors, *TACAS*, volume 5505 of *LNCS*, pages 428–442. Springer, 2009.
9. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–443. Elsevier and MIT Press, 2001.
10. E. Tushkanova, A. Giorgetti, C. Ringeissen, and O. Kouchnarenko. A rule-based framework for building superposition-based decision procedures. In F. Durán, editor, *WRLA*, volume 7571 of *LNCS*, pages 221–239. Springer, 2012.
11. E. Tushkanova, C. Ringeissen, A. Giorgetti, and O. Kouchnarenko. Automatic Decidability for Theories with Counting Operators. In F. van Raamsdonk, editor, *RTA'13*, 2013.