

Modular Termination and Combinability for Superposition Modulo Counter Arithmetic

C. Ringeissen, V. Senni

LORIA & INRIA Grand Est, Nancy

FTP'11



Outline

- 1 Introduction
- 2 Decision Procedures via Superposition (for Data Structures)
- 3 Decision Procedures via Combination (for Linear Arithmetic)
- 4 Conclusion

Outline

- 1 Introduction
- 2 Decision Procedures via Superposition (for Data Structures)
- 3 Decision Procedures via Combination (for Linear Arithmetic)
- 4 Conclusion

Building Decision Procedures

- Based on **Rewriting** techniques (a uniform big engine)
 - Pros** uniform use of a *superposition calculus* for FOL with **Equality** (requires termination proofs for the theories of interest)
 - Cons** not suitable for arithmetic
 - ➔ Applied to data structures [ARR03, ABRS09, BE07, dMB08]
- Based on **N-O Combination** (several small engines)
 - Pros** use procedures available for individual theories and try to build a procedure for the **union** of theories
 - Cons** hardly applicable to **non-disjoint unions** (complex conditions)
 - ➔ Applied to **disjoint unions** of data structures and arithmetic [KRRT05]

Our mixed approach

- Rewriting** (as much as possible) + **Combination** (when needed)
- ➔ Applied to **non-disjoint unions** of data structures and arithmetic

A general combination framework (à la Nelson-Oppen)

Developed by Ghilardi-Nicolini-Zucchelli [GNZ08] as a generalization of the Nelson-Oppen combination method.

Purification Consider a set of pure constraints $\Gamma_1 \cup \Gamma_2$;

Propagation the T_1 -procedure and the T_2 -procedure fairly exchange **shared positive clauses** entailed by $T_1 \cup \Gamma_1$ and by $T_2 \cup \Gamma_2$

Until an inconsistency is found or saturation is reached

Requirements:

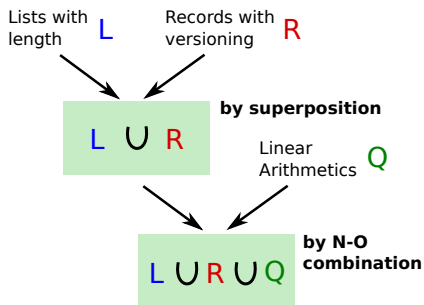
- **disjoint** case : stably infiniteness
- **non-disjoint** case : combinability
= computability of bases + noetherianity + compatibility

An example of application

```

function minmax (l : list) : record {
  while (l != nil) {
    e := car(l);
    if e < rselect_1 (r)
      then r := rstore_1 (r, e);
    if rselect_2 (r) < e
      then r := rstore_2 (r, e);
    l := cdr(l)
  };
  return r
}

```



Property : $\forall l, r (r = \text{minmax}(l) \Rightarrow \text{countR}(r) \leq \text{countL}(l))$

Previous Work

In [NRR10]:

- The **SP_C calculus**, used as a decision procedure for **equational theories** extending **counter arithmetic**
- The **combination framework** as a basic building block for combining **equational theories** extending **counter arithmetic**

Combination as much as possible + **Rewriting** only for single theories

Current work:

Rewriting as much as possible + **Combination** when needed

Advantages:

- **Simplification** of the applicability conditions (termination easier than combinability)
- **Automation** of requirements checking (by saturation)

Outline

- 1 Introduction
- 2 Decision Procedures via Superposition (for Data Structures)**
- 3 Decision Procedures via Combination (for Linear Arithmetic)
- 4 Conclusion

Examples: Data structures with versioning

Lists : $\text{nil} : \text{LISTS}$, $\text{cons} : \text{ELEM} \times \text{LISTS} \rightarrow \text{LISTS}$, $\ell : \text{LISTS} \rightarrow \text{NUM}$

$$\begin{aligned} \text{car}(\text{cons}(x, y)) &\simeq x & \neg \text{atom}(x) \rightarrow \text{cons}(\text{car}(x), \text{cdr}(x)) &\simeq x \\ \text{cdr}(\text{cons}(x, y)) &\simeq y & \neg \text{atom}(\text{cons}(x, y)) & \\ & & \text{atom}(\text{nil}) & \end{aligned}$$

$$\ell(\text{cons}(x, y)) \simeq \mathbf{s}(\ell(y)) \quad \ell(\text{nil}) \simeq \mathbf{0}$$

Records : $\text{rstore}_i : \text{RECORD} \times \text{ELEM}_i \rightarrow \text{RECORD}$, $\text{count}_R : \text{RECORD} \rightarrow \text{NUM}$

$$\begin{aligned} \text{rselect}_i(\text{rstore}_i(x, y)) &\simeq y \\ \text{rselect}_j(\text{rstore}_i(x, y)) &\simeq \text{rselect}_j(x) \\ \bigwedge_{i=1}^n (\text{rselect}_i(x) \simeq \text{rselect}_i(y)) &\rightarrow x \simeq y && \text{(extensionality)} \end{aligned}$$

$$\text{count}_R(\text{rstore}_i(x, y)) \simeq \mathbf{s}(\text{count}_R(x))$$

Note: we use sat-preserving reductions to equational theories

Counter Arithmetic

| | | |
|----------------|---|-------|
| Injectivity | $\forall x, y \quad s(x) = s(y) \rightarrow x = y$ | (Inj) |
| Acyclicity | $\forall x \quad x \neq s^n(x) \text{ for all } n \in \mathbb{N}^+$ | (Acy) |
| No Predecessor | $\forall x \quad s(x) \neq 0$ | (NoP) |

- 1 Theory of Integer Offsets [NRR09b]: $T_I = \{Inj, Acy, S0\}$
- 2 Theory of Increment [NRR09a]: $T_S = \{Inj, Acy\}$

T_C as the generic name for T_I and T_S .

Superposition Calculus - Expansion Rules

| | | |
|-----------------------|--|------------------------|
| <i>Superposition</i> | $\frac{l[u'] = r \quad u = t}{(l[t] = r)\sigma}$ | (i), (ii), (iii), (iv) |
| <i>Paramodulation</i> | $\frac{l[u'] \neq r \quad u = t}{(l[t] \neq r)\sigma}$ | (i), (ii), (iii), (iv) |
| <i>Reflection</i> | $\frac{u' \neq u}{\perp}$ | (i) |

where (i) σ is the most general unifier of u and u' , (ii) u' is not a variable, (iii) $u\sigma \not\leq t\sigma$, (iv) $l[u']\sigma \not\leq r\sigma$.

Figure: Expansion Rules.

Counter Arithmetic - Ground Reduction Rules

Ad hoc rules encoding T_C axioms, applied to **ground** terms.

| | | |
|--------------------|--|--|
| $R1$ (Injectivity) | $\frac{S \cup \{s(u) = s(v)\}}{S \cup \{u = v\}}$ | |
| $R2$ (Injectivity) | $\frac{S \cup \{s(u) = t, s(v) = t\}}{S \cup \{s(v) = t, u = v\}}$ | if $s(u) \succ t$, $s(v) \succ t$ and $u \succ v$ |
| $C1$ (Acyclicity) | $\frac{S \cup \{s^n(t) = t\}}{S \cup \{s^n(t) = t\} \cup \perp}$ | if $n \in \mathbb{N}$ |
| $C2$ (NoP) | $\frac{S \cup \{s(t) = 0\}}{S \cup \{s(t) = 0\} \cup \perp}$ | if t is a ground term and $n \in \mathbb{N}$ |

where S is a set of literals and \perp is the symbol for the inconsistency.

Figure: (Ground) **Reduction** Rules.

Completeness and Termination of SP_C

$SP_C = \mathbf{Expansion} + \mathbf{Reduction} \text{ Rules}$

A syntactic condition on saturations (**safety**) ensures completeness

Refutation Completeness of SP_C

safety of saturations implies: for every set of ground literals G if G is $T \cup T_C$ -unsatisfiable then the saturation of $Ax(T) \cup G$ contains \square

For $T \in \{\text{Lists w length, Records w versioning, Trees w size, } \dots\}$

Termination

for any set of ground flat literals G , the saturation of $Ax(T) \cup G$ is finite.

Termination of SP_C for $T \implies SP_C$ is a **decision procedure** for T -sat

Modular Termination (History)

SP terminates for **Lists** and **Arrays**, does it terminate for **Lists \cup Arrays**?

Problem:

given T_1 and T_2 disjoint, find conditions under which termination of SP for T_1 and T_2 entails termination for $T_1 \cup T_2$

Solution (Disjoint Case):

if T_1 and T_2 are **variable inactive** then SP terminates [ABRS09]
(no **maximal** literal $X = t$, where $X \notin \text{Var}(t)$, in the saturation)

Algorithm

Mimick of (disjoint) Nelson-Oppen combination by superposition

Role of Variable Inactivity:

Limit across-theories inferences to the propagation of equalities between shared constants (finitely many)

Modular Termination (This Paper)

Consider **non**-disjoint unions of theories sharing **Counter Arithmetic**

Problem:

given T_1 and T_2 **non**-disjoint, find conditions under which termination of SP for T_1 and T_2 entails termination for $T_1 \cup T_2$

Solution (non-Disjoint Case):

if T_1 and T_2 are **safe** then SP terminates and $T_1 \cup T_2$ is **safe**
(**maximal terms** in equalities are either **ground** or of **non-shared** sort)

Algorithm

Mimick of (non-disjoint) Nelson-Oppen combination

Role of Safe Termination:

Limit **accross-theories inferences** to the propagation of ground **shared** equalities of the form $a = s^n(b)$

Superposition Strategy

Superposition strategy for $T_1 \cup T_2$:

Purification Consider a set of pure ground constraints $\Gamma_1 \cup \Gamma_2$;

Propagation Across-theories inferences fairly apply paramodulation steps between **shared ground equalities** in the saturations of $T_i \cup \Gamma_i$ and literals of $T_j \cup \Gamma_j$, for $i \neq j$

Until an inconsistency is found or saturation is reached

Fact: safety implies that *shared equalities* are

(1) **ground** and (2) of the form $a = s^n(b)$

Termination proof: shared equalities $a = s^n(b)$ are finitely many

Termination: Across-theories inferences

Safety: **maximal terms** in equalities are either **ground** or belong to a **non-shared** sort (subsumes **variable inactivity**)

Analysis of across-theories inferences, by cases:

- Paramodulation from variables:
impossible by variable inactivity
- Paramodulation from constants:
only for $a = s^n(b)$, by safety (+ good ordering)
- Paramodulation from compound terms:
only for $s^n(a) = b$, by safety (+ good ordering)

Fact: for every constants $a, b \in NUM$, either Γ is T -unsatisfiable or there is at most one $s^n(a) = b$, for some n , in the saturation of $T \cup \Gamma$

There are **finitely many across-theories inferences**. QED

Result

Safety and modular termination provide:

- 1 a method to infer termination of the non-disjoint union $T_1 \cup T_2$ by inspection of T_1 - and T_2 -saturations
- 2 a class of **safe** theories **closed under union**
- 3 an algorithmic test if SP_C is a decision procedure for $T_1 \cup T_2$ -sat.
- 4 an effective, uniform calculus for $T_1 \cup T_2$

Outline

- 1 Introduction
- 2 Decision Procedures via Superposition (for Data Structures)
- 3 Decision Procedures via Combination (for Linear Arithmetic)**
- 4 Conclusion

Combination: critical points and our contributions

Let $T_0 = T_1 \cap T_2$, we have to guarantee:

- ① Computation of the entailed shared ground equalities (T_0 -bases)
 - SP_C computes T_C -bases
 - Safe termination \implies computable T_C -bases
- ② Termination of the exchange loop (Noetherianity of T_0)
 - T_C is **Noetherian** [NRR10]
- ③ Completeness of the combination method (T_0 -compatibility)
 - (extends **stably infiniteness** in the disjoint case)
 - Proper termination \implies T_C -compatibility

Remark: the set of Proper theories is **closed under union**

SP_C preserves **compatibility/computation-of-bases** under **union**

General Compatibility Result

Compatibility of component theories:
ensures **completeness** of their Nelson-Oppen combination

Let $\Sigma_0 = \Sigma_1 \cap \Sigma_2$

disjoint case : if Σ_0 empty, **compatibility = stable infiniteness**
requires satisfiability of T_1, T_2 in a model with infinite domain
➔ **analysis of saturations**: **variable inactivity**

non-disjoint case : if Σ_0 is the signature of **Counter Arithmetic**
compatibility requires satisfiability of T_1, T_2 in a model where each
(non-0) element admits a predecessor
➔ **analysis of saturations**: **proper termination** (stronger than safety)

General Compatibility Result: Sketch of the Proof

Proper termination: safety + avoid shared equations of the form $C[s(X)] = t$ (where X is a variable) in the saturation

1. Define a sequence of models such that the limit is a model of $\forall x \ x \neq 0 \Rightarrow \exists y \ x = s(y)$
 - ▶ Each model is built by model generation techniques, from a saturated set
 - ▶ Each saturated set is extended with a ground equality $s(c') = c$ where c “has no predecessor” and c' is a new constant
2. By **properness**, the extended set is **saturated** (thus, consistent)

Properness is a syntactic property of **saturation**s ensuring that every model of a theory $T \supseteq T_C$ can be extended to ensure **compatibility**

Outline

- 1 Introduction
- 2 Decision Procedures via Superposition (for Data Structures)
- 3 Decision Procedures via Combination (for Linear Arithmetic)
- 4 Conclusion**

Two Contributions

Analysis of saturations can be used to infer

- 1 **Termination** and **Completeness** of SP_C applied to **non-disjoint unions** of theories sharing **counter arithmetic**
(boils down to a superposition-based Nelson-Oppen loop)
- 2 **Combinability** and **Computability of Bases** required for the Non-Disjoint Combination Framework
(needed when we combine with, e.g., arithmetic)

Conclusion and Future Work

- Extension to **clauses** and **non-convex** theories (e.g. arrays)
 - ➔ adapt the superposition calculus to handle non-unit clauses
- Other presentations of counter arithmetic (i.e. with predecessor and successor) complexity?
- Modularity of termination and combinability:
beyond shared counter arithmetic

References



Alessandro Armando, Maria P. Bonacina, Silvio Ranise, and Stephan Schulz.
New results on rewrite-based satisfiability procedures.
ACM Transactions on Computational Logic, 10(1), 2009.



Alessandro Armando, Silvio Ranise, and Michaël Rusinowitch.
A rewriting approach to satisfiability procedures.
Information and Computation, 183(2):140–164, 2003.



Maria Paola Bonacina and Mnacho Echenim.
T-decision by decomposition.
In *Proc. of CADE'07*, volume 4603 of *LNCS*, pages 199–214. Springer, July 2007.



Leonardo Mendonça de Moura and Nikolaj Bjørner.
Engineering DPLL(T) + Saturation.
In *Proc. of IJCAR'08*, volume 5195 of *LNCS*, pages 475–490. Springer, 2008.



Silvio Ghilardi, Enrica Nicolini, and Daniele Zucchelli.
A comprehensive combination framework.
ACM Transactions on Computational Logic, 9(2):1–54, 2008.



Hélène Kirchner, Silvio Ranise, Christophe Ringeissen, and Duc-Khanh Tran.
On superposition-based satisfiability procedures and their combination.
In D. Van Hung and M. Wirsing, editors, *Proc. of ICTAC 2005*, volume 3722 of *LNCS*, pages 594–608, Hanoi (Vietnam), 2005. Springer-Verlag.



Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.

Data structures with arithmetic constraints: a non-disjoint combination.

In *Proc. of FroCoS'09*, LNCS. Springer, 2009.



Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.

Satisfiability procedures for combination of theories sharing integer offsets.

In *Proc. of TACAS'09*, volume 5505 of LNCS, pages 428–442. Springer, 2009.



Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.

Combining satisfiability procedures for unions of theories with a shared counting operator.

Fundam. Inform., 105(1-2):163–187, 2010.