Data structures with arithmetic constraints: non-disjoint combinations

E. Nicolini, C. Ringeissen, and M. Rusinowitch

LORIA & INRIA-Lorraine

ADDCT-UNIF 09
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
Building Decision Procedures

The most investigated approaches:

- **Rewriting** techniques
  - use a *superposition calculus* for FOL with *Equality* and prove its termination for useful cases in verification
  - Application to data structures [ARR03, ABRS09, BE07, dMB08]

- **Combination** techniques
  - use procedures available for individual theories and try to build a procedure for the *union* of theories
  - Application to the union of data structures and fragments of arithmetic [KRRT05]

Our approach: blend both the approaches to combine data structures sharing some arithmetic operators

- Application of the combination method proposed by Ghilardi-Nicolini-Zucchelli [GNZ08]: a combination method à la Nelson-Oppen [NO79] for *non-disjoint unions of theories*
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
Data structures using arithmetic operators

**Lists**: \( \text{nil} : \text{LISTS}, \text{cons} : \text{ELEM} \times \text{LISTS} \rightarrow \text{LISTS}, \ell : \text{LISTS} \rightarrow \text{NUM} \)

\[
\ell(\text{nil}) = 0 \\
\ell(\text{cons}(x, y)) = s(\ell(y))
\]

**Trees**: \( \text{bin} : \text{ELEM} \times \text{TREES} \times \text{TREES} \rightarrow \text{TREES}, \text{null} : \text{TREES}, \text{size} : \text{TREES} \rightarrow \text{NUM} \)

\[
\text{size}(\text{null}) = 0 \\
\text{size}(\text{bin}(e, t_1, t_2)) = \text{size}(t_1) + \text{size}(t_2) + 1 \\
0 \neq 1
\]

**Records**: \( \text{sel}_i : \text{RECS} \rightarrow \text{NUM}, \text{inc} : \text{RECS} \rightarrow \text{RECS} \)

\[
\text{sel}_i(\text{inc}(r)) = s(\text{sel}_i(r))
\]

for any index \( i \) of sort \( \text{NUM} \).
Possible shared theories

(lnj) \( \forall x, y \ s(x) = s(y) \rightarrow x = y \)
(Acy) \( \forall x \ x \neq s^n(x) \) for all \( n \in \mathbb{N}^+ \)
(S0) \( \forall x \ s(x) \neq 0 \)

1. Theory of Integer Offsets [NRR09c]: \( T_I = \{ Inj, Acy, S0 \} \)
2. Theory of Increment [NRR09b]: \( T_S = \{ Inj, Acy \} \)
3. Theory of Abelian Groups [NRR09a]:
\( AG = AC(+) \cup \{ x + (-x) = 0, x + 0 = x \} \)
Superposition Calculus

\[
\begin{align*}
\text{Superposition} & \quad \frac{l[u'] = r \quad u = t}{(l[t] = r) \sigma} \\
& \quad (i), (ii), (iii), (iv) \\
\text{Paramodulation} & \quad \frac{l[u'] \neq r \quad u = t}{(l[t] \neq r) \sigma} \\
& \quad (i), (ii), (iii), (iv) \\
\text{Reflection} & \quad \frac{u' \neq u}{\bot} \\
& \quad (i)
\end{align*}
\]

where (i) $\sigma$ 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 Inference Rules.
Superposition Calculus (for a successor function)

Ad hoc rules to be applied to ground terms:

\[
R1 \text{ (for } \text{Inj)} \quad \frac{S \cup \{s(u) = s(v)\}}{S \cup \{u = v\}}
\]

\[
R2 \text{ (for } \text{Inj)} \quad \frac{S \cup \{s(u) = t, s(v) = t\}}{S \cup \{s(v) = t, u = v\}} \text{ if } s(u) \succ t, \quad s(v) \succ t \text{ and } u \succ v
\]

\[
C1 \text{ (for } \text{Acy)} \quad \frac{S \cup \{s^n(t) = t\}}{S \cup \{s^n(t) = t\} \cup \bot} \text{ if } n \in \mathbb{N}
\]

\[
C2 \text{ (for } \text{S0)} \quad \frac{S \cup \{s(t) = 0\}}{S \cup \{s(t) = 0\} \cup \bot}
\]

where \(S\) is a set of literals and \(\bot\) is the symbol for the inconsistency.

**Figure:** Ground reduction Inference Rules.
Superposition Calculi as Decision Procedures

Result ([NRR09c, NRR09b])
An appropriate Superposition Calculus leads to a decision procedure for a class of theories modelling data-structures with the **unary successor function**.

* Examples: Lists with length, Records with increment

Result ([NRR09a])
A Superposition Calculus modulo \(AG\) leads to a decision procedure for a class of theories modelling data-structures with the **binary addition function**.

* Examples: previous ones + Trees with size.

Proof (\(AG\) case):

1. A many-sorted and unconstrained version the Superposition Calculus modulo \(AG\) developed by Godoy-Nieuwenhuis [GN04]
2. Use of \(AG\)-unification with free symbols
3. Considered theories: unit clauses with no variable of sort \(AG\)
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
Linear Arithmetic

\[ \Sigma_Q := \{0, 1, +, -, \{f_q\}_{q \in \mathbb{Q}}, s, <\}, \] where 0, 1 are constants, \(-, f_q, s\) are unary function symbols. Let \( T_Q \) be the set of all the \( \Sigma_Q \)-sentences that are true in \( \mathbb{Q} \).

Fact

A \( T_Q \)-satisfiability procedure can be obtained by using

1. Fourier-Motzkin Elimination (for inequalities)
   - to detect unsatisfiability or to compute implicit equalities

2. Gauss Elimination (for equalities)
   - a function \texttt{solve} to compute the solved form of a set of equalities

3. Disequality Handler
   - a function \texttt{canon} over arithmetic expressions to check whether an disequality can be canonized into an unsatisfiable disequality \( u \neq u \).
Non-Linear Arithmetic: The Theory of $\mathbb{Q}$-Algebras

$T_{Q-\text{alg}}$ is $\text{AC}(+) \cup \text{AC}(\times) \cup U(+, 0) \cup U(\times, 1)$ plus

\begin{align*}
\forall x \ x + (-x) &= 0 \quad (1) \\
0 &\neq 1 \quad (2) \\
\forall x \ s(x) &= x + 1 \quad (3) \\
\forall x, y, z \ (x + y)z &= xz + yz \quad (4) \\
\forall x, y \ q(x + y) &= qx + qy \quad (5) \\
\forall x \ (q_1 \oplus q_2)x &= q_1 x + q_2 x \quad (6) \\
\forall x \ (q_1 \cdot q_2)x &= q_1 (q_2 x) \quad (7) \\
\forall x \ 1_\mathbb{Q} x &= x \quad (8) \\
\forall x, y \ q(xy) &= x(qy) \quad (9)
\end{align*}

Fact

A $T_{Q-\text{alg}}$-satisfiability procedure can be obtained by using the Buchberger algorithm for the computation of Groebner bases.
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
A combination problem

\[ \Gamma_1 = \left\{ \begin{array}{l} y = \ell(a) \\ b = \text{cons}(e, a) \\ x = \ell(b) \end{array} \right\} \]

\[ \Gamma_2 = \left\{ \begin{array}{l} u \geq 0 \\ x + u = y \end{array} \right\} \]

Satisfiability of \( \Gamma_1 \cup \Gamma_2 \)?

\( \Gamma_1 \cup \Gamma_2 \) is unsatisfiable since

- \( \Gamma_1 \rightarrow x = s(y) \)
- \( \Gamma_2 \cup \{ x = s(y) \} \) is \( T_2 \)-unsatisfiable:

\[ \Gamma_2 \cup \{ x = s(y) \} \leftrightarrow \{ u \geq 0, u = -1 \} \]
A combination problem

\[ \Gamma_1 = \begin{cases} 
    y = \ell(a) \\
    b = \text{cons}(e, a) \\
    x = \ell(b) 
\end{cases} \]

\[ \Gamma_2 = \begin{cases} 
    u \geq 0 \\
    x + u = y 
\end{cases} \]

Satisfiability of \( \Gamma_1 \cup \Gamma_2 \)?

\( \Gamma_1 \cup \Gamma_2 \) is unsatisfiable since

- \( \Gamma_1 \rightarrow x = s(y) \)
- \( \Gamma_2 \cup \{ x = s(y) \} \) is \( T_2 \)-unsatisfiable:

\[ \Gamma_2 \cup \{ x = s(y) \} \leftrightarrow \{ u \geq 0, u = -1 \} \]
Non-disjoint combination method (à la Nelson-Oppen)

Combination method developed by Ghilardi-Nicolini-Zucchelli [GNZ08]: Let $T_0 = T_1 \cap T_2$ and $\Sigma_0 = \Sigma_1 \cap \Sigma_2$

**Purification** Given a set of $T_1 \cup T_2$-constraints $\Gamma$, produce an equisatisfiable set of pure constraints $\Gamma_1 \cup \Gamma_2$;

**Propagation** the $T_1$-constraint solving procedure and the $T_2$-constraint solving procedure fairly exchange shared positive $\Sigma_0$-clauses that are entailed by $T_1 \cup \Gamma_1$ and by $T_2 \cup \Gamma_2$

**Until** an inconsistency is detected or a saturation state is reached.

**Pseudo-code:**

1. If $T_0$-basis $T_i(\Gamma_i) = \Delta_i$ and $\perp \notin \Delta_i$ for each $i \in \{1, 2\}$, then
   1.1. For each $D \in \Delta_i$ such that $T_j \cup \Gamma_j \not\models D$, ($i \neq j$), add $D$ to $\Gamma_j$
   1.2. If $\Gamma_1$ or $\Gamma_2$ has been changed in 1.1, then rerun 1.

   Else return *Unsatisfiable*

2. Return *Satisfiable.*
Non-disjoint combination method (à la Nelson-Oppen)

Combination method developed by Ghilardi-Nicolini-Zucchelli [GNZ08]: Let $T_0 = T_1 \cap T_2$ and $\Sigma_0 = \Sigma_1 \cap \Sigma_2$

Purification Given a set of $T_1 \cup T_2$-constraints $\Gamma$, produce an equisatisfiable set of pure constraints $\Gamma_1 \cup \Gamma_2$.

Propagation the $T_1$-constraint solving procedure and the $T_2$-constraint solving procedure fairly exchange shared positive $\Sigma_0$-clauses that are entailed by $T_1 \cup \Gamma_1$ and by $T_2 \cup \Gamma_2$

Until an inconsistency is detected or a saturation state is reached.

Pseudo-code:
1. If $T_0$-basis $T_i(\Gamma_i) = \Delta_i$ and $\bot \notin \Delta_i$ for each $i \in \{1, 2\}$, then
   1.1. For each $D \in \Delta_i$ such that $T_j \cup \Gamma_j \not\models D$, ($i \neq j$), add $D$ to $\Gamma_j$
   1.2. If $\Gamma_1$ or $\Gamma_2$ has been changed in 1.1, then rerun 1.

   Else return Unsatisfiable

2. Return Satisfiable.
Combination method: critical points

1. How to obtain the $T_0$-bases, which are logical consequences of a constraint $\Gamma$ w.r.t. a theory $T_0$ over a given sub-signature
   ➤ Computability of $T_0$-bases

2. How to guarantee the termination of the exchange loop
   ➤ Noetherianity of $T_0$

3. How to ensure its completeness
   ➤ $T_0$-compatibility (extends the assumption on stably infinite theories used in the disjoint case)

Our work: how to face these issues when dealing with

(i) a combination of two data-structures sharing the theory of Integer Offsets
(ii) a combination of one data structure and one theory of arithmetic sharing the theory of Increment
(iii) a combination of two data-structures sharing the theory of Abelian Groups
Computation of bases for data structures

Result

In case of satisfiability, our Superposition Calculi compute $T_0$-bases for $T_0 = T_I, T_S, AG$.

How to compute $T_0$-bases: collect all the shared equalities in a saturation of $\Gamma$ not containing $\perp$.

Example

The saturation of

$$\Gamma = \{ y = \ell(a), b = cons(e, a), x = \ell(b) \}$$

contains

$$x = s(y)$$
Result

It is possible to compute $T_S$-bases for $T_\mathbb{Q}$ and $T_\mathbb{Q} - \text{alg}$.

Proof Idea:

1. (Linear case) Assume $\Gamma$ is a set of linear equalities. We have

$$T \cup \Gamma \models a_1 = s^n(a_2) \iff \text{canon}(a_1\gamma - a_2\gamma) = n$$

where $\gamma = \text{solve}(\Gamma)$.

2. (Non-linear case) It is possible to compute the set of all entailed linear equalities by using a slight adaptation of the Buchberger algorithm, as shown in Nicolini’s thesis. Then proceed as in (1).
Background on Combination

Computation of $T_S$-bases: example for the arithmetic

Example

\[ \Gamma = \begin{cases} 
    x = c \\
    1 + 2c + y = 2 + 3d \\
    2c = d + x 
\end{cases} \]

$\Gamma$ is equivalent to the solved form:

\[ \text{solve}(\Gamma) = \begin{cases} 
    x = c \\
    y = c + 1 \\
    d' = c 
\end{cases} \]

Therefore:

\[ \Gamma \rightarrow y = s(x) \]
Computation of $T_S$-bases: example for the arithmetic

Example

\[
\Gamma = \begin{cases} 
  x = c \\
  1 + 2c + y = 2 + 3d \\
  2c = d + x 
\end{cases}
\]

\(\Gamma\) is equivalent to the solved form:

\[
\text{solve}(\Gamma) = \begin{cases} 
  x = c \\
  y = c + 1 \\
  d = c 
\end{cases}
\]

Therefore:

\(\Gamma \rightarrow y = s(x)\)
Computation of $T_S$-bases: example for the arithmetic

Example

\[
\Gamma = \begin{cases} 
  x = c \\
  1 + 2c + y = 2 + 3d \\
  2c = d + x 
\end{cases}
\]

$\Gamma$ is equivalent to the solved form:

\[
\text{solve}(\Gamma) = \begin{cases} 
  x = c \\
  y = c + 1 \\
  d = c 
\end{cases}
\]

Therefore:

\[
\Gamma \rightarrow y = s(x)
\]
Non-disjoint extension of Nelson-Oppen applied to the theory of Increment ($T_S$)

We have identified a class of theories $\text{DST}_S$ modelling data structures modulo $T_S$ such that for any $T \in \text{DST}_S \cup \{T_Q, T_Q\text{-alg}\}$: the Ghilardi-Nicolini-Zucchelli combination method is

1. complete
2. terminating

**Theorem ([NRR09b])**

*For any $\Sigma_1$-theory $T_1 \in \text{DST}_S$ and any $\Sigma_2$-theory $T_2 \in \{T_Q, T_Q\text{-alg}\} \cup \text{DST}_S$ such that $\Sigma_1 \cap \Sigma_2 = \Sigma_S$, $T_1 \cup T_S \cup T_2$ has a decidable constraint satisfiability problem.*
Non-disjoint extension of Nelson-Oppen applied to the theory of Integer Offsets ($T_I$)

We have identified a class of theories $\text{DST}_I$, modelling data structures modulo $T_I$, such that for any $T \in \text{DST}_I$:

- the Ghilardi-Nicolini-Zucchelli combination method is complete
- terminating

Theorem ([NRR09c])

For any $\Sigma_1$-theory $T_1 \in \text{DST}_I$ and any $\Sigma_2$-theory $T_2 \in \text{DST}_I$ such that $\Sigma_1 \cap \Sigma_2 = \Sigma_I$, $T_1 \cup T_I \cup T_2$ has a decidable constraint satisfiability problem.
Non-disjoint extension of Nelson-Oppen applied to the theory of Abelian Groups (AG)

We have identified a class of theories $\text{DST}_{AG}$ modelling data structures modulo $AG$ such that for any $T \in \text{DST}_{AG}$:

1. complete
2. terminating

Theorem ([NRR09a])

For any $\Sigma_1$-theory $T_1 \in \text{DST}_{AG}$ and any $\Sigma_2$-theory $T_2 \in \text{DST}_{AG}$ such that $\Sigma_1 \cap \Sigma_2 = \Sigma_{AG}$, $T_1 \cup AG \cup T_2$ has a decidable constraint satisfiability problem.
Outline

1. Introduction
2. Data Structures
3. Arithmetic
4. Background on Combination
5. Conclusion
Conclusion and future work

- What about a theory of arithmetic over the integers?
  ➤ Computation of bases seems more difficult for the integers!
- Possibility of combining a data structure with a theory of arithmetic sharing the + operator?
  ➤ continuation of our work on abelian groups [NRR09a]
- How to deal with a non-convex data structure such as arrays?
  ➤ adaptation of the superposition calculus, to handle clauses instead of unit clauses
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.

Maria Paola Bonacina and Mnacho Echenim.
T-decision by decomposition.

Leonardo Mendonça de Moura and Nikolaj Bjørner.
Engineering DPLL(T) + Saturation.

Guillem Godoy and Robert Nieuwenhuis.
Superposition with completely built-in abelian groups.

Silvio Ghilardi, Enrica Nicolini, and Daniele Zucchelli.
A comprehensive combination framework.

Hélène Kirchner, Silvio Ranise, Christophe Ringeissen, and Duc-Khanh Tran.
On superposition-based satisfiability procedures and their combination.

**Greg Nelson and Derek C. Oppen.**
Simplification by cooperating decision procedures.

**Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.**
Combinable extensions of abelian groups.

**Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.**
Data structures with arithmetic constraints: a non-disjoint combination.

**Enrica Nicolini, Christophe Ringeissen, and Michaël Rusinowitch.**
Satisfiability procedures for combination of theories sharing integer offsets.