Combined Satisfiability Modulo Parametric Theories

Sava Krstić*, Amit Goel*, Jim Grundy*, and **Cesare Tinelli****

*Strategic CAD Labs, Intel

**The University of Iowa

This Talk

Based on work in

- S. Krstić, A. Goel, J. Grundy, and C. Tinelli.
  Combined Satisfiability Modulo Parametric Theories.
  TACAS'07, 2007.

- S. Krstić and A. Goel.
  Architecting Solvers for SAT Modulo Theories: Nelson-Oppen with DPLL.
  FroCoS, 2007.

Contribution

Nelson-Oppen framework for theories in parametrically polymorphic logics—a fresh foundation for design of SMT solvers

Highlights

- Endowing SMT with a rich typed input language that can model arbitrarily nested data structures
- Completeness of a Nelson-Oppen-style combination method proved for theories of all common datatypes
- Troublesome *stable infiniteness* condition replaced by a natural notion of type parametricity
- Issue of handling *finite-cardinality constraints* exposed as crucial for completeness

SAT Modulo Theories (SMT)

There are decision procedures for (fragments of) logical theories of common datatypes

Use them to decide validity/satisfiability of *queries*, quantifier-free formulas, that involve symbols from several theories

- $f(x) = x \;\Rightarrow\; f(2x - f(x)) = x$ $[\mathcal{T}_{\mathsf{UF}} + \mathcal{T}_{\mathsf{Int}}]$
- $\mathsf{head}(a) = f(x) + 1 \;\ldots$ $[\mathcal{T}_{\mathsf{UF}} + \mathcal{T}_{\mathsf{Int}} + \mathcal{T}_{\mathsf{List}}]$

The underlying logic is classical (unsorted or many-sorted) first-order logic

SMT Solvers over Multiple Theories

G. Nelson, D. C. Oppen Simplification by cooperating decision procedures, 1979

Input:

- theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$ with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$
- decision procedures P_i for the $\mathcal{T}_i$-satisfiability of sets of Σ_i-literals

Output:

- a decision procedure for $(\mathcal{T}_1 + \cdots + \mathcal{T}_n)$-satisfiability of sets of $(\Sigma_1 + \cdots + \Sigma_n)$-literals.

SMT Solvers over Multiple Theories

Input:

- theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$ with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$
- decision procedures P_i for the $\mathcal{T}_i$-satisfiability of sets of Σ_i-literals

Output:

- a decision procedure for $(\mathcal{T}_1 + \cdots + \mathcal{T}_n)$-satisfiability of sets of $(\Sigma_1 + \cdots + \Sigma_n)$-literals.

Main Idea:

1. Input S is *purified* into equisatisfiable $S_1, \ldots S_n$;
2. each P_i works on S_i but propagates to the others any entailed equalities between shared variables.

Nelson-Oppen: Example

$\mathcal{T}_1$ = theory of lists $\qquad$ $\mathcal{T}_2$ = linear arithmetic

Input set:

$$S = \begin{cases} l_1 \neq l_2, \\ \mathsf{head}(l_2) \leq x, \\ l = \mathsf{tail}(l_2), \\ l_1 = x :: l, \\ \mathsf{head}(l) - \mathsf{head}(\mathsf{tail}\ l_1) + x \leq \mathsf{head}(l_2) \end{cases}$$

Purified sets:

$$S_1 = \begin{cases} l_1 \neq l_2, \\ y_1 = \mathsf{head}(l_2), \\ l = \mathsf{tail}(l_2), \\ l_1 = x :: l, \\ y_2 = \mathsf{head}(l),\ y_3 = \mathsf{head}(\mathsf{tail}\ l_1) \end{cases} \qquad S_2 = \begin{cases} y_1 \leq x, \\ y_2 - y_3 + x \leq y_1 \end{cases}$$

Nelson-Oppen: Example

S_1	S_2
$l_1 \neq l_2$	$y_1 \leq x$
$y_1 = \mathsf{head}(l_2)$	$y_2 - y_3 + x \leq y_1$
$l = \mathsf{tail}(l_2)$	
$l_1 = x :: l$	
$y_2 = \mathsf{head}(l)$	
$y_3 = \mathsf{head}(\mathsf{tail}\ l_1)$	

Nelson-Oppen: Example

S_1	S_2
$l_1 \neq l_2$	$y_1 \leq x$
$y_1 = \mathsf{head}(l_2)$	$y_2 - y_3 + x \leq y_1$
$l = \mathsf{tail}(l_2)$	
$l_1 = x :: l$	
$y_2 = \mathsf{head}(l)$	
$y_3 = \mathsf{head}(\mathsf{tail}\ l_1)$	
$\longrightarrow$	$y_2 = y_3$

Nelson-Oppen: Example

S_1	S_2
$l_1 \neq l_2$	$y_1 \leq x$
$y_1 = \mathsf{head}(l_2)$	$y_2 - y_3 + x \leq y_1$
$l = \mathsf{tail}(l_2)$	
$l_1 = x :: l$	
$y_2 = \mathsf{head}(l)$	
$y_3 = \mathsf{head}(\mathsf{tail}\ l_1)$	
$\longrightarrow$	$y_2 = y_3$
$x = y_1$	$\longleftarrow$

Nelson-Oppen: Example

S_1	S_2
$l_1 \neq l_2$	$y_1 \leq x$
$y_1 = \mathsf{head}(l_2)$	$y_2 - y_3 + x \leq y_1$
$l = \mathsf{tail}(l_2)$	
$l_1 = x :: l$	
$y_2 = \mathsf{head}(l)$	
$y_3 = \mathsf{head}(\mathsf{tail}\ l_1)$	
$\longrightarrow$	$y_2 = y_3$
$x = y_1$	$\longleftarrow$
Unsatisfiable!	

Correctness of Nelson-Oppen

- The combination procedure is sound for any $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

  if it returns “Unsatisfiable”, then its input S is unsatisfiable in $\mathcal{T}_1 + \cdots + \mathcal{T}_n$

Correctness of Nelson-Oppen

- The combination procedure is sound for any $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

  if it returns "Unsatisfiable", then its input S is unsatisfiable in $\mathcal{T}_1 + \cdots + \mathcal{T}_n$

- It is complete when

Correctness of Nelson-Oppen

- The combination procedure is sound for any $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

  if it returns "Unsatisfiable", then its input S is unsatisfiable in $\mathcal{T}_1 + \cdots + \mathcal{T}_n$

- It is complete when

  1. $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are pairwise signature-disjoint, and

Correctness of Nelson-Oppen

- The combination procedure is sound for any $\mathcal{T}_1, \ldots, \mathcal{T}_n$:

  if it returns "Unsatisfiable", then its input S is unsatisfiable in $\mathcal{T}_1 + \cdots + \mathcal{T}_n$

- It is complete when

  1. $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are pairwise signature-disjoint, and
  2. each $\mathcal{T}_i$ is *stably-infinite*

The Notorious Stable Infiniteness Restriction

A first-order theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable ground formula is satisfiable in an infinite model of $\mathcal{T}$.

The Notorious Stable Infiniteness Restriction

A first-order theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable ground formula is satisfiable in an infinite model of $\mathcal{T}$.

- Helps guarantee that models of pure parts of a query φ can be amalgamated into a model of φ

The Notorious Stable Infiniteness Restriction

A first-order theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable ground formula is satisfiable in an infinite model of $\mathcal{T}$.

- Helps guarantee that models of pure parts of a query φ can be amalgamated into a model of φ
- Yields completeness of N-O, but
  - it's not immediate to prove
  - it's not true in some important cases (e.g., bit vectors)

The Notorious Stable Infiniteness Restriction

A first-order theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable ground formula is satisfiable in an infinite model of $\mathcal{T}$.

- Helps guarantee that models of pure parts of a query φ can be amalgamated into a model of φ
- Yields completeness of N-O, but
  - it's not immediate to prove
  - it's not true in some important cases (e.g., bit vectors)
- General understanding: the condition doesn't matter much—if you know what you are doing

The Notorious Stable Infiniteness Restriction

A first-order theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable ground formula is satisfiable in an infinite model of $\mathcal{T}$.

- Helps guarantee that models of pure parts of a query φ can be amalgamated into a model of φ
- Yields completeness of N-O, but
  - it's not immediate to prove
  - it's not true in some important cases (e.g., bit vectors)
- General understanding: the condition doesn't matter much—if you know what you are doing
- Lot of research shows completeness of N-O variants without it: [Tinelli-Zarba'04], [Fontaine-Gribomont'04], [Zarba'04], [Ghilardi et al.'07], [Ranise et al.'05]

Why Stable Infiniteness is Needed

$\mathcal{T}_1 \;=\;$ theory of "uninterpreted functions"

$\mathcal{T}_2 \;=\;$ theory of Boolean rings (not stably-infinite)

Purified Input:

S_1	S_2
$f(x_1) \neq x_1$	$x_1 = 0$
$f(x_1) \neq x_2$	$x_2 = 1$

There are no equations to propagate: the procedure returns "satisfiable"

Is that correct?

Our Main Points

In combining theories of different data types

1. a typed logic (with parametric types) is a more adequate underlying logic than unsorted logic
2. parametricity is the key notion not stable infiniteness

Parametricity, Not Stable Infiniteness: Example

$$\Phi_{\mathsf{List}} = \begin{cases} \mathsf{tail}\, l_1 = \mathsf{tail}\, l_2 \\ x_1 = \mathsf{head}\, l_1 \\ x_2 = \mathsf{head}\, l_2 \\ x = \mathsf{head}(\mathsf{tail}\, l_1) \end{cases} \qquad \Phi_{\mathsf{Int}} = \begin{cases} x = x_1 + z \\ x_2 = x_1 + z \end{cases} \qquad \Delta = \begin{cases} x = x_2 \\ x \neq x_1 \end{cases}$$

Parametricity, Not Stable Infiniteness: Example

$$\Phi_{\mathsf{List}} = \begin{cases} \mathsf{tail}\, l_1 = \mathsf{tail}\, l_2 \\ x_1 = \mathsf{head}\, l_1 \\ x_2 = \mathsf{head}\, l_2 \\ x = \mathsf{head}(\mathsf{tail}\, l_1) \end{cases} \qquad \Phi_{\mathsf{Int}} = \begin{cases} x = x_1 + z \\ x_2 = x_1 + z \end{cases} \qquad \Delta = \begin{cases} x = x_2 \\ x \neq x_1 \end{cases}$$

$$\begin{pmatrix} x_1 & x_2 & x & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{pmatrix} \models_{\mathcal{T}_{\mathsf{List}}} \Phi_{\mathsf{List}} \cup \Delta \qquad \begin{pmatrix} x_1 & x_2 & x & z \\ 1 & 2 & 2 & 1 \end{pmatrix} \models_{\mathcal{T}_{\mathsf{Int}}} \Phi_{\mathsf{Int}} \cup \Delta$$

Parametricity, Not Stable Infiniteness: Example

$$\Phi_{\mathsf{List}} = \begin{cases} \mathsf{tail}\, l_1 = \mathsf{tail}\, l_2 \\ x_1 = \mathsf{head}\, l_1 \\ x_2 = \mathsf{head}\, l_2 \\ x = \mathsf{head}(\mathsf{tail}\, l_1) \end{cases} \qquad \Phi_{\mathsf{Int}} = \begin{cases} x = x_1 + z \\ x_2 = x_1 + z \end{cases} \qquad \Delta = \begin{cases} x = x_2 \\ x \neq x_1 \end{cases}$$

$$\begin{pmatrix} {\color{red}x_1} & {\color{red}x_2} & {\color{red}x} & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{pmatrix} \models_{\mathcal{T}_{\mathsf{List}}} \Phi_{\mathsf{List}} \cup \Delta \qquad \begin{pmatrix} {\color{blue}x_1} & {\color{blue}x_2} & {\color{blue}x} & z \\ 1 & 2 & 2 & 1 \end{pmatrix} \models_{\mathcal{T}_{\mathsf{Int}}} \Phi_{\mathsf{Int}} \cup \Delta$$

$\mathcal{T}_{\mathsf{List}}$ knows nothing about $\mathbb{Z}$ and cannot distinguish $(\blacktriangle, \bullet)$ from any pair (m, n) of distinct integers:

Parametricity, Not Stable Infiniteness: Example

$$\Phi_{\mathsf{List}} = \begin{cases} \mathsf{tail}\, l_1 = \mathsf{tail}\, l_2 \\ x_1 = \mathsf{head}\, l_1 \\ x_2 = \mathsf{head}\, l_2 \\ x = \mathsf{head}(\mathsf{tail}\, l_1) \end{cases} \qquad \Phi_{\mathsf{Int}} = \begin{cases} x = x_1 + z \\ x_2 = x_1 + z \end{cases} \qquad \Delta = \begin{cases} x = x_2 \\ x \neq x_1 \end{cases}$$

$$\begin{pmatrix} x_1 & x_2 & x & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{pmatrix} \models_{\mathcal{T}_{\mathsf{List}}} \Phi_{\mathsf{List}} \cup \Delta \qquad \begin{pmatrix} x_1 & x_2 & x & z \\ 1 & 2 & 2 & 1 \end{pmatrix} \models_{\mathcal{T}_{\mathsf{Int}}} \Phi_{\mathsf{Int}} \cup \Delta$$

$\mathcal{T}_{\mathsf{List}}$ knows nothing about $\mathbb{Z}$ and cannot distinguish $(\blacktriangle, \bullet)$ from any pair (m, n) of distinct integers:

$$\begin{pmatrix} x_1 & x_2 & x & l_1 & l_2 \\ m & n & n & [m, n] & [n, n] \end{pmatrix} \models \Phi_{\mathsf{List}} \cup \Delta \text{ as well}$$

Parametricity, Not Stable Infiniteness: Example

$$\Phi_{\mathsf{List}} = \begin{cases} \mathsf{tail}\, l_1 = \mathsf{tail}\, l_2 \\ x_1 = \mathsf{head}\, l_1 \\ x_2 = \mathsf{head}\, l_2 \\ x = \mathsf{head}(\mathsf{tail}\, l_1) \end{cases} \qquad \Phi_{\mathsf{Int}} = \begin{cases} x = x_1 + z \\ x_2 = x_1 + z \end{cases} \qquad \Delta = \begin{cases} x = x_2 \\ x \neq x_1 \end{cases}$$

$$\begin{pmatrix} \color{red}{x_1} & \color{red}{x_2} & \color{red}{x} & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{pmatrix} \models_{\mathcal{T}_{\mathsf{List}}} \Phi_{\mathsf{List}} \cup \Delta \qquad \begin{pmatrix} \color{blue}{x_1} & \color{blue}{x_2} & \color{blue}{x} & z \\ 1 & 2 & 2 & 1 \end{pmatrix} \models_{\mathcal{T}_{\mathsf{Int}}} \Phi_{\mathsf{Int}} \cup \Delta$$

$\mathcal{T}_{\mathsf{List}}$ knows nothing about $\mathbb{Z}$ and cannot distinguish $(\blacktriangle, \bullet)$ from any pair (m, n) of distinct integers:

to construct a model for $\Phi_{\mathsf{List}} \cup \Phi_{\mathsf{Int}} \cup \Delta$, we can use the blue assignment to x_1, x_2, x

Real Issue in NO Combination

Not so much getting stable-infiniteness right, but

getting underlying logic right

Real Issue in NO Combination

Not so much getting stable-infiniteness right, but

getting underlying logic right

Our proposal

FOLP: A first order logic with parametrized type constructors and type variables

Essentially, the applicative fragment of HOL

FOLP Syntax

Types

V, an infinite set of *type variables*

Ex: $\alpha,\ \beta,\ \alpha_1,\ \beta_1,\ \ldots$

O, a set of *type operators*, symbols with associated arity $n \geq 0$

Ex: $\mathsf{Bool}/0,\ \ \mathsf{Int}/0,\ \ \mathsf{List}/1,\ \ \mathsf{Arr}/2,\ \ \Rightarrow/2,\ \ \ldots$

$\mathrm{Types}(O, V)$, set of *types*, terms over O, V

Ex: $\mathsf{Int},\ \mathsf{List}(\alpha),\ \mathsf{List}(\mathsf{Int}),\ \mathsf{Arr}(\mathsf{Int}, \mathsf{List}(\alpha)),\ \mathsf{List}(\alpha) \Rightarrow \mathsf{Int},\ \ldots$

FOLP Syntax

First-order Types: Types over $O \setminus \{\Rightarrow\}, V$

Constants: K, set of symbols each with an associated *principal* type τ

Ex: $\top^{\mathsf{Bool}}$, $\neg^{\mathsf{Bool}\Rightarrow\mathsf{Bool}}$, $=^{\alpha,\alpha\Rightarrow\mathsf{Bool}}$, $+^{\mathsf{Int},\mathsf{Int}\Rightarrow\mathsf{Int}}$, $\mathsf{cons}^{\alpha,\mathsf{List}(\alpha)\Rightarrow\mathsf{List}(\alpha)}$, $\mathsf{read}^{\mathsf{Arr}(\alpha,\beta),\alpha\Rightarrow\beta}$, ...

Term Variables: X_τ, for each $\tau \in \mathrm{Types}(O, V)$, an infinite set of symbols annotated with τ

Ex: x^α, $y^{\mathsf{List}(\beta)}$, $z^{\alpha\Rightarrow\alpha}$, $x^{\mathsf{Arr}(\mathsf{Int},\mathsf{Bool})}$, ...

FOLP Syntax

Signatures: pairs $\Sigma = \langle O \mid K \rangle$ with

- O always containing $\Rightarrow$ and Bool
- K always containing $=^{\alpha,\alpha\Rightarrow\mathsf{Bool}}$, $\mathsf{ite}^{\mathsf{Bool},\alpha,\alpha\Rightarrow\alpha}$, and the usual logical constants $\neg^{\mathsf{Bool}\Rightarrow\mathsf{Bool}}$, $\wedge^{\mathsf{Bool},\mathsf{Bool}\Rightarrow\mathsf{Bool}}$, …

Σ-Terms of First-order Type τ: $\mathrm{T}_\tau(K, X)$, defined as usual

Ex: $x^{\mathsf{Int}\Rightarrow\mathsf{Bool}}\ y^{\mathsf{Int}}$, $(\mathsf{read}\ a^{\mathsf{Arr}(\mathsf{Int},\mathsf{List}(\beta))}\ i^{\mathsf{Int}}) = x^{\mathsf{List}(\beta)}$,

First-order (Quantifier-free) Formulas: Terms in $\mathrm{T}_{\mathsf{Bool}}(K, X)$

FOLP Semantics

Structures of signature $\Sigma = \langle O \mid K \rangle$

Pair $\mathcal{S}$ of

1. an interpretation $(\_)^{\mathcal{S}}$ of type operators F as set operators
2. an interpretation $(\_)^{\mathcal{S}}$ of constants f as set-indexed families of functions (with index determined by $\mathrm{TypeVars}(\tau)$ where f^{τ})

s.t. Bool, $\Rightarrow$, and $=$, ite, $\wedge, \ldots$ are the interpreted in the usual way.

FOLP Semantics

Structures of signature $\Sigma = \langle O \mid K \rangle$

Pair $\mathcal{S}$ of

1. an interpretation $(\_)^{\mathcal{S}}$ of type operators F as set operators
2. an interpretation $(\_)^{\mathcal{S}}$ of constants f as set-indexed families of functions (with index determined by $\mathrm{TypeVars}(\tau)$ where f^{τ})

s.t. Bool, $\Rightarrow$, and $=$, ite, $\wedge, \ldots$ are the interpreted in the usual way.

Ex 1:

$\mathsf{Int}^{\mathcal{S}}$	equals the integers
$\mathsf{List}^{\mathcal{S}}$	maps an input set A to the set of finite lists over A
$\mathsf{Arr}^{\mathcal{S}}$	maps input sets I and A to the set of arrays with index set I and element set A

FOLP Semantics

Structures of signature $\Sigma = \langle O \mid K \rangle$

Pair $\mathcal{S}$ of

1. an interpretation $(\_)^{\mathcal{S}}$ of type operators F as set operators
2. an interpretation $(\_)^{\mathcal{S}}$ of constants f as set-indexed families of functions (with index determined by $\mathrm{TypeVars}(\tau)$ where f^{τ})

s.t. Bool, $\Rightarrow$, and $=$, ite, $\wedge, \ldots$ are the interpreted in the usual way.

Ex 2:

$\mathsf{head}^{\mathcal{S}}$	family $\{head[A_1] \mid A_1$ is a set$\}$ (since $\mathsf{head}^{\mathsf{List}(\alpha) \Rightarrow \alpha}$)
$\mathsf{read}^{\mathcal{S}}$	family $\{read[A_1, A_2] \mid A_1, A_2$ are sets$\}$ (since $\mathsf{read}^{\mathsf{Arr}(\alpha_1, \alpha_2), \alpha_1 \Rightarrow \alpha_2}$)
$+^{\mathcal{S}}$	singleton family (since $+^{\mathsf{Int}, \mathsf{Int} \Rightarrow \mathsf{Int}}$)

FOLP Semantics

For every signature $\Sigma = \langle O \mid K \rangle$, Σ-structure $\mathcal{S}$, type environment ι, term environment ρ, and Σ-formula φ,

we can define $[\_]^{\mathcal{S}}_{\iota,\rho}$ (as expected) to map Σ-formulas to $\{\mathrm{true}, \mathrm{false}\}$

FOLP Semantics

For every signature $\Sigma = \langle O \mid K \rangle$, Σ-structure $\mathcal{S}$, type environment ι, term environment ρ, and Σ-formula φ,

we can define $[\_]^{\mathcal{S}}_{\iota,\rho}$ (as expected) to map Σ-formulas to $\{\mathrm{true}, \mathrm{false}\}$

Satisfiability

φ is *satisfied in $\mathcal{S}$* by ι and ρ, written $\iota, \rho \models_{\mathcal{S}} \varphi$, if $[\varphi]^{\mathcal{S}}_{\iota,\rho} = \mathrm{true}$

φ is *satisfiable in $\mathcal{S}$* if $\iota, \rho \models_{\mathcal{S}} \varphi$ for some ι and ρ

FOLP Semantics

For every signature $\Sigma = \langle O \mid K \rangle$, Σ-structure $\mathcal{S}$, type environment ι, term environment ρ, and Σ-formula φ,

we can define $[\_]^{\mathcal{S}}_{\iota,\rho}$ (as expected) to map Σ-formulas to $\{\mathrm{true}, \mathrm{false}\}$

Satisfiability

φ is *satisfied in $\mathcal{S}$* by ι and ρ, written $\iota, \rho \models_{\mathcal{S}} \varphi$, if $[\varphi]^{\mathcal{S}}_{\iota,\rho} = \mathrm{true}$

φ is *satisfiable in $\mathcal{S}$* if $\iota, \rho \models_{\mathcal{S}} \varphi$ for some ι and ρ

Cardinality Constraints

(Meta)Expressions of the form $\alpha \doteq n$ with $n > 0$

$\alpha \doteq n$ is *satisfied in $\mathcal{S}$* by ι, ρ, written $\iota, \rho \models_{\mathcal{S}} \alpha \doteq n$, if $|\iota(\alpha)| = n$

The Equality Structure

Let

$$
\begin{aligned}
K_{\mathsf{Eq}} &= =^{\alpha,\alpha\Rightarrow\mathsf{Bool}}, \top^{\mathsf{Bool}}, \neg^{\mathsf{Bool}\Rightarrow\mathsf{Bool}}, \mathsf{ite}^{\mathsf{Bool},\alpha,\alpha\Rightarrow\alpha}, \ldots \\
\Sigma_{\mathsf{Eq}} &= \langle \mathsf{Bool}, \Rightarrow \mid K_{\mathsf{Eq}} \rangle \\
\mathcal{S}_{\mathsf{Eq}} &= \text{the unique } \Sigma_{\mathsf{Eq}}\text{-structure}
\end{aligned}
$$

The Equality Structure

Let

$$
\begin{aligned}
K_{\mathsf{Eq}} &= \; =^{\alpha,\alpha\Rightarrow\mathsf{Bool}}, \top^{\mathsf{Bool}}, \neg^{\mathsf{Bool}\Rightarrow\mathsf{Bool}}, \mathsf{ite}^{\mathsf{Bool},\alpha,\alpha\Rightarrow\alpha}, \ldots \\
\Sigma_{\mathsf{Eq}} &= \langle \mathsf{Bool}, \Rightarrow \mid K_{\mathsf{Eq}} \rangle \\
\mathcal{S}_{\mathsf{Eq}} &= \text{ the unique } \Sigma_{\mathsf{Eq}}\text{-structure}
\end{aligned}
$$

Note: $\mathcal{S}_{\mathsf{Eq}}$ models

- the logical constants of FOL= and
- the "uninterpreted functions" data type, by means of higher-order term variables ($x^{\alpha_1,\ldots,\alpha_n\Rightarrow\alpha}$)

The Equality Structure

Let

$$
\begin{aligned}
K_{\mathsf{Eq}} &= \; =^{\alpha,\alpha\Rightarrow\mathsf{Bool}}, \top^{\mathsf{Bool}}, \neg^{\mathsf{Bool}\Rightarrow\mathsf{Bool}}, \mathsf{ite}^{\mathsf{Bool},\alpha,\alpha\Rightarrow\alpha}, \ldots \\
\Sigma_{\mathsf{Eq}} &= \langle \mathsf{Bool}, \Rightarrow \mid K_{\mathsf{Eq}} \rangle \\
\mathcal{S}_{\mathsf{Eq}} &= \text{ the unique } \Sigma_{\mathsf{Eq}}\text{-structure}
\end{aligned}
$$

Note: $\mathcal{S}_{\mathsf{Eq}}$ models

- the logical constants of FOL= and
- the "uninterpreted functions" data type, by means of higher-order term variables ($x^{\alpha_1,\ldots,\alpha_n\Rightarrow\alpha}$)

Fact: The satisfiability in $\mathcal{S}_{\mathsf{Eq}}$ of first-order Σ_{Eq}-formulas is decidable (with the usual congruence closure algorithms)

Parametricity [TACAS'07]

A structure is *parametric* if it interprets
all its type operators, except $\Rightarrow$, as *parametric set operators* and
all its constants as *parametric function families*

Parametricity [TACAS'07]

A structure is *parametric* if it interprets
all its type operators, except $\Rightarrow$, as *parametric set operators* and
all its constants as *parametric function families*

- Parametricity of type operators and constants similar (but not comparable) to Reynold's parametricity

Parametricity [TACAS'07]

A structure is *parametric* if it interprets
all its type operators, except $\Rightarrow$, as *parametric set operators* and
all its constants as *parametric function families*

- Parametricity of type operators and constants similar (but not comparable) to Reynold's parametricity
- Natural property of data types

Parametricity [TACAS'07]

A structure is *parametric* if it interprets
all its type operators, except $\Rightarrow$, as *parametric set operators* and
all its constants as *parametric function families*

- Parametricity of type operators and constants similar (but not comparable) to Reynold's parametricity
- Natural property of data types
- States precisely the informal notion that

  certain type operators and function symbols have a uniform interpretation over the possible values of the type variables

Parametricity [TACAS'07]

A structure is *parametric* if it interprets
all its type operators, except $\Rightarrow$, as *parametric set operators* and
all its constants as *parametric function families*

- Parametricity of type operators and constants similar (but not comparable) to Reynold's parametricity
- Natural property of data types
- States precisely the informal notion that

  certain type operators and function symbols have a uniform interpretation over the possible values of the type variables

- Plays the role of stable-infiniteness in Nelson-Oppen

Parametric Structures

Fact: All structures of practical interest are parametric in our sense

$\Sigma_{\mathsf{Int}} = \langle \mathsf{Int} \mid 0^{\mathsf{Int}}, 1^{\mathsf{Int}}, +^{\mathsf{Int}^2 \to \mathsf{Int}}, -^{\mathsf{Int}^2 \to \mathsf{Int}}, \times^{\mathsf{Int}^2 \to \mathsf{Int}}, \leq^{\mathsf{Int}^2 \to \mathsf{Bool}}, \ldots \rangle$

$\Sigma_{\mathsf{Arr}} = \langle \mathsf{Arr} \mid \mathsf{mk\_arr}^{\beta \to \mathsf{Arr}(\alpha,\beta)}, \mathsf{read}^{[\mathsf{Arr}(\alpha,\beta),\alpha] \to \beta}, \mathsf{write}^{[\mathsf{Arr}(\alpha,\beta),\alpha,\beta] \to \mathsf{Arr}(\alpha,\beta)} \rangle$

$\Sigma_{\mathsf{List}} = \langle \mathsf{List} \mid \mathsf{cons}^{[\alpha,\mathsf{List}(\alpha)] \to \mathsf{List}(\alpha)}, \mathsf{nil}^{\mathsf{List}(\alpha)}, \mathsf{head}^{\mathsf{List}(\alpha) \to \alpha}, \mathsf{tail}^{\mathsf{List}(\alpha) \to \mathsf{List}(\alpha)} \rangle$

$\Sigma_{\times} = \langle \times \mid \langle \_, \_ \rangle^{[\alpha,\beta] \to \alpha \times \beta}, \mathsf{fst}^{\alpha \times \beta \to \alpha}, \mathsf{snd}^{\alpha \times \beta \to \beta} \rangle$

$\Sigma_{\mathsf{BitVec32}} = \ldots$

$\Sigma_{\mathsf{Sets}} = \ldots$

$\Sigma_{\mathsf{Multisets}} = \ldots$

(All the above signatures implicitly include the signature Σ_{Eq})

Combining Signatures and Structures

Disjoint Signatures

Signatures that share exactly the symbols of Σ_{Eq}

Combining Signatures and Structures

Disjoint Signatures

Signatures that share exactly the symbols of Σ_{Eq}

Combination of Disjoint Signatures Σ_1, Σ_2

$\Sigma_1 + \Sigma_2 = \langle O_1 \cup O_2 \mid K_1 \cup K_2 \rangle$ where $\Sigma_i = \langle O_i \mid K_i \rangle$

Combining Signatures and Structures

Disjoint Signatures

Signatures that share exactly the symbols of Σ_{Eq}

Combination of Disjoint Signatures Σ_1, Σ_2

$\Sigma_1 + \Sigma_2 = \langle O_1 \cup O_2 \mid K_1 \cup K_2 \rangle$ where $\Sigma_i = \langle O_i \mid K_i \rangle$

Combination of Signature-Disjoint Structures $\mathcal{S}_1, \mathcal{S}_2$

$(\Sigma_1 + \Sigma_2)$-structure $\mathcal{S}_1 + \mathcal{S}_2$ that interprets Σ_i-symbols exactly like $\mathcal{S}_i$ for $i = 1, 2$.

Combining Signatures and Structures

Disjoint Signatures

Signatures that share exactly the symbols of Σ_{Eq}

Combination of Disjoint Signatures Σ_1, Σ_2

$\Sigma_1 + \Sigma_2 = \langle O_1 \cup O_2 \mid K_1 \cup K_2 \rangle$ where $\Sigma_i = \langle O_i \mid K_i \rangle$

Combination of Signature-Disjoint Structures $\mathcal{S}_1, \mathcal{S}_2$

$(\Sigma_1 + \Sigma_2)$-structure $\mathcal{S}_1 + \mathcal{S}_2$ that interprets Σ_i-symbols exactly like $\mathcal{S}_i$ for $i = 1, 2$.

Note: Modulo isomorphism, + is an ACU operator with unit $\mathcal{S}_{\mathsf{Eq}}$

Pure and Semipure Terms

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be structures with disjoint signatures $\Sigma_i = \langle O_i \mid K_i \rangle$

We call a $(\Sigma_1 + \cdots + \Sigma_n)$-term

- *i-semipure* if it has signature $\langle O_1 \cup \cdots \cup O_n \mid K_i \rangle$
- *i-pure* if it has signature $\langle O_i \mid K_i \rangle$

Ex

$\Sigma_1 = \langle \mathsf{Int} \mid 0^{\mathsf{Int}}, 1^{\mathsf{Int}}, +^{\mathsf{Int},\mathsf{Int}\Rightarrow\mathsf{Int}}, -^{\mathsf{Int}\Rightarrow\mathsf{Int}}, \leq^{\mathsf{Int},\mathsf{Int}\Rightarrow\mathsf{Bool}}, \ldots \rangle$

$\Sigma_2 = \langle \mathsf{Arr} \mid \mathsf{read}^{\mathsf{Arr}(\alpha,\beta),\alpha\Rightarrow\beta}, \mathsf{write}^{\mathsf{Arr}(\alpha,\beta),\alpha,\beta\Rightarrow\mathsf{Arr}(\alpha,\beta)} \rangle$

1-semipure: $\mathsf{read}(a^{\mathsf{Arr}(\mathsf{Int},\mathsf{Int})}, i^{\mathsf{Int}}),\quad a^{\mathsf{Arr}(\mathsf{Int},\beta)},\quad a^{\mathsf{Arr}(\mathsf{Int},\mathsf{Arr}(\mathsf{Int},\mathsf{Int}))}$

1-pure: $\mathsf{read}(a^{\mathsf{Arr}(\alpha,\alpha)}, i^{\alpha}),\quad a^{\mathsf{Arr}(\alpha,\beta)},\quad a^{\mathsf{Arr}(\alpha,\mathsf{Arr}(\beta_1,\beta_2))}$

Pure and Semipure Terms

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be structures with disjoint signatures $\Sigma_i = \langle O_i \mid K_i \rangle$

We call a $(\Sigma_1 + \cdots + \Sigma_n)$-term

- *i-semipure* if it has signature $\langle O_1 \cup \cdots \cup O_n \mid K_i \rangle$
- *i-pure* if it has signature $\langle O_i \mid K_i \rangle$

Fact For each i-semipure term t we can compute a most specific pure generalization t^{pure} of t

Ex

$$\varphi : \quad \mathsf{read}(a^{\mathsf{Arr}(\mathsf{Int},\mathsf{Pair}(\mathsf{Arr}(\mathsf{Bool},\mathsf{Bool})))}, i^{\mathsf{Int}}) = x^{\mathsf{Pair}(\mathsf{Arr}(\mathsf{Bool},\mathsf{Bool}))}$$

$$\varphi^{\mathrm{pure}} : \quad \mathsf{read}(a^{\mathsf{Arr}(\alpha,\beta)}, i^{\alpha}) = x^{\beta}$$

Pure and Semipure Terms

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be parametric structures with disjoint signatures $\Sigma_i = \langle O_i \mid K_i \rangle$

Proposition A set Φ_i of i-semipure formulas is $(\mathcal{S}_1 + \cdots + \mathcal{S}_n)$-satisfiable

iff

$\Phi_i^{\mathrm{pure}} \cup \Phi_i^{\mathrm{card}}$ is $\mathcal{S}_i$-satisfiable

for some suitable set Φ_i^{card} of cardinality constraints computable from Φ_i

Ex

$$\Phi_i : \quad \{ \mathsf{read}(a^{\mathsf{Arr}(\mathsf{Int},\mathsf{Pair}(\mathsf{Arr}(\mathsf{Bool},\mathsf{Bool})))}, i^{\mathsf{Int}}) = x^{\mathsf{Pair}(\mathsf{Arr}(\mathsf{Bool},\mathsf{Bool}))} \}$$

$$\Phi_i^{\mathrm{pure}} : \quad \{ \mathsf{read}(a^{\mathsf{Arr}(\alpha,\beta)}, i^{\alpha}) = x^{\beta} \}$$

$$\Phi_i^{\mathrm{card}} : \quad \{ \beta \doteq 16 \}$$

Why Cardinality Constraints are Needed

$$\Phi : \quad \{x_i^{\mathsf{List}(\alpha)} \neq x_j^{\mathsf{List}(\alpha)}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\alpha)}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_1 : \quad \{x_i^{\mathsf{List}(\mathsf{Int})} \neq x_j^{\mathsf{List}(\mathsf{Int})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Int})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_2 : \quad \{x_i^{\mathsf{List}(\mathsf{Bool})} \neq x_j^{\mathsf{List}(\mathsf{Bool})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Bool})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

- Φ and Φ_1 are $(\mathcal{S}_{\mathsf{Int}} + \mathcal{S}_{\mathsf{List}})$-satisfiable, Φ_2 is not

Why Cardinality Constraints are Needed

$$\Phi : \quad \{x_i^{\mathsf{List}(\alpha)} \neq x_j^{\mathsf{List}(\alpha)}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\alpha)}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_1 : \quad \{x_i^{\mathsf{List}(\mathsf{Int})} \neq x_j^{\mathsf{List}(\mathsf{Int})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Int})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_2 : \quad \{x_i^{\mathsf{List}(\mathsf{Bool})} \neq x_j^{\mathsf{List}(\mathsf{Bool})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Bool})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

- Φ and Φ_1 are $(\mathcal{S}_{\mathsf{Int}} + \mathcal{S}_{\mathsf{List}})$-satisfiable, Φ_2 is not
- $\mathcal{S}_{\mathsf{List}}$-solver can't take Φ_1 or Φ_2 as input: they are not Σ_{List}-pure

Why Cardinality Constraints are Needed

$$\Phi : \quad \{x_i^{\mathsf{List}(\alpha)} \neq x_j^{\mathsf{List}(\alpha)}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\alpha)}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_1 : \quad \{x_i^{\mathsf{List}(\mathsf{Int})} \neq x_j^{\mathsf{List}(\mathsf{Int})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Int})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_2 : \quad \{x_i^{\mathsf{List}(\mathsf{Bool})} \neq x_j^{\mathsf{List}(\mathsf{Bool})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Bool})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

- Φ and Φ_1 are $(\mathcal{S}_{\mathsf{Int}} + \mathcal{S}_{\mathsf{List}})$-satisfiable, Φ_2 is not
- $\mathcal{S}_{\mathsf{List}}$-solver can't take Φ_1 or Φ_2 as input: they are not Σ_{List}-pure
- Instead of Φ_1, it gets $\Phi = \Phi_1^{\text{pure}}$ with cardinality constraint $\emptyset$

Why Cardinality Constraints are Needed

$$\Phi: \quad \{x_i^{\mathsf{List}(\alpha)} \neq x_j^{\mathsf{List}(\alpha)}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\alpha)}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_1: \quad \{x_i^{\mathsf{List}(\mathsf{Int})} \neq x_j^{\mathsf{List}(\mathsf{Int})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Int})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

$$\Phi_2: \quad \{x_i^{\mathsf{List}(\mathsf{Bool})} \neq x_j^{\mathsf{List}(\mathsf{Bool})}\}_{0 \leq i < j \leq 5} \cup \{\mathsf{tail}(\mathsf{tail}\ x_i^{\mathsf{List}(\mathsf{Bool})}) = \mathsf{nil}\}_{1 \leq i \leq 5}$$

- Φ and Φ_1 are $(\mathcal{S}_{\mathsf{Int}} + \mathcal{S}_{\mathsf{List}})$-satisfiable, Φ_2 is not
- $\mathcal{S}_{\mathsf{List}}$-solver can't take Φ_1 or Φ_2 as input: they are not Σ_{List}-pure
- Instead of Φ_1, it gets $\Phi = \Phi_1^{\mathrm{pure}}$ with cardinality constraint $\emptyset$
- Instead of Φ_2, it gets $\Phi = \Phi_2^{\mathrm{pure}}$ with the cardinality constraint $\{\alpha \doteq 2\}$

Towards Nelson-Oppen Combination: Purification

We turn each query Φ into the *purified form*

$$\Phi_B \cup \Phi_E \cup \Phi_1 \cup \cdots \cup \Phi_n$$

where

- Φ_B is a set of propositional formulas
- $\Phi_E = \{p^{\mathsf{Bool}} \equiv x^\tau = y^\tau\}_{p^{\mathsf{Bool}}, x^\tau, y^\tau}$ with $\tau \neq \mathsf{Bool}$
- $\Phi_i = \{p^{\mathsf{Bool}} \equiv \psi\}_{p^{\mathsf{Bool}}, \psi} \cup \{x^\tau = t\}_{x^\tau, t}$ with ψ, t non-variables, i-semipure, and not containing logical constants

Ex: $f(x) = x \vee f(2 * x - f(x)) > x$ becomes

$\Phi_B = \{p \vee q\}$ $\Phi_E = \{p \equiv y = x\},$

$\Phi_{\mathsf{Eq}} = \{y = f(x),\ u = f(z)\}$ $\Phi_{\mathsf{Int}} = \{q \equiv u > x\ z = 2 * x - y,\}$

Towards a Combination Theorem

Let

- A be a set of propositional atoms (i.e., Bool-variables)
- X a set of of variables

An *assignment* M of A is a consistent set of literals with atoms in A

An *arrangement* Δ of X is a set of equational literals corresponding to a well-typed partition of X

Ex

Partition: $\{\{x^{\tau_1}, y^{\tau_1}, z^{\tau_1}\},\ \{u^{\tau_2}, v^{\tau_2}\},\ \{w^{\tau_3}\}\}$

Δ : $\{x^{\tau_1} = y^{\tau_1}, x^{\tau_1} = z^{\tau_1},\ u^{\tau_2} = v^{\tau_2},\ x^{\tau_1} \neq u^{\tau_2},\ x^{\tau_1} \neq w^{\tau_3}\}$

Main Result: A Combination Theorem for FOLP

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be signature-disjoint, flexible structures

Main Result: A Combination Theorem for FOLP

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be signature-disjoint, flexible structures

A query

$$\Phi \;=\; \Phi_B \cup \Phi_E \cup \Phi_1 \cup \cdots \cup \Phi_n$$

is $(\mathcal{S}_1 + \cdots + \mathcal{S}_n)$-satisfiable iff

Main Result: A Combination Theorem for FOLP

Let $\mathcal{S}_1, \ldots, \mathcal{S}_n$ be signature-disjoint, flexible structures

A query

$$\Phi \;=\; \Phi_B \cup \Phi_E \cup \Phi_1 \cup \cdots \cup \Phi_n$$

is $(\mathcal{S}_1 + \cdots + \mathcal{S}_n)$-satisfiable iff
there is

- an assignment M of the atoms in Φ_B and
- an arrangement Δ of the non-Bool variables in Φ

s.t.

1. $M \models \Phi_B$
2. $M, \Delta \models \Phi_E$
3. $(\Phi_i \cup M \cup \Delta)^{\text{pure}} \cup {\Phi_i}^{\text{card}}$ is $\mathcal{S}_i$-satisfiable for all $i = 1, \ldots, n$

Main Theoretical Requirement: Flexible Structures

A structure $\mathcal{S}$ is *flexible* if for

- every query Φ,
- every injective $\langle \iota, \rho \rangle$ such that $\langle \iota, \rho \rangle \models_{\mathcal{S}} \Phi$,
- every $\alpha \in V$,
- every $\kappa > |\iota(\alpha)|$

there exist injective $\langle \iota^{\mathrm{up}(\kappa)}, \rho^{\mathrm{up}(\kappa)} \rangle$ and $\langle \iota^{\mathrm{down}}, \rho^{\mathrm{down}} \rangle$ satisfying Φ s.t.

$\iota^{\mathrm{up}(\kappa)}(\beta) = \iota(\beta) = \iota^{\mathrm{down}}(\beta)$ for every $\beta \neq \alpha$, and

1. $\iota^{\mathrm{up}(\kappa)}(\alpha)$ has cardinality κ *[up-flexibility]*
2. $\iota^{\mathrm{down}}(\alpha)$ is countable *[down-flexibility]*

Main Theoretical Requirement: Flexible Structures

A structure $\mathcal{S}$ is *flexible* if for

- every query Φ,
- every injective $\langle \iota, \rho \rangle$ such that $\langle \iota, \rho \rangle \models_{\mathcal{S}} \Phi$,
- every $\alpha \in V$,
- every $\kappa > |\iota(\alpha)|$

there exist injective $\langle \iota^{\mathrm{up}(\kappa)}, \rho^{\mathrm{up}(\kappa)} \rangle$ and $\langle \iota^{\mathrm{down}}, \rho^{\mathrm{down}} \rangle$ satisfying Φ s.t.

$\iota^{\mathrm{up}(\kappa)}(\beta) = \iota(\beta) = \iota^{\mathrm{down}}(\beta)$ for every $\beta \neq \alpha$, and

1. $\iota^{\mathrm{up}(\kappa)}(\alpha)$ has cardinality κ *[up-flexibility]*
2. $\iota^{\mathrm{down}}(\alpha)$ is countable *[down-flexibility]*

Lemma Every parametric structure is flexible

Main Computational Requirement: Strong Solvers

We call a solver for $\mathcal{S}$-satisfiability *strong* if it can process queries with cardinality constraints.

- Typical $\mathcal{S}$-solvers are not strong
- however, they can be effectively converted into strong solvers by preprocessing each query
- currently this can be done, specifically for a number of structures, as in [Ranise et al., FroCoS'05]
- we are working on a (possibly less efficient but) generic preprocessing mechanism

Closest Related Work [Ranise et al., FroCoS'05]

Setting (2-theory case):

- Many-sorted logic (with sorts being 0-ary type operators)
- Signatures share at most a set of sorts
- One theory is *polite* over shared sorts, other theory is arbitrary

Main Result:

Theory solvers are combined, soundly and completely, with a Nelson-Oppen style method that also guesses equalities over some additional terms computed from the input query.

Comparisons with [Ranise et al., FroCoS'05]

That work vs. This work

- Theory combinations via signature push-outs
  Theory combinations via type parameter instantiation
- Politeness assumption on theories
  Flexibility assumption on structures
- Politeness proven per theory
  Parametricity as general sufficient condition for flexibility
- Idea of parametricity is implicit in politeness
  Parametricity notion fully fleshed out
- Model finiteness issues addressed directly by combination method
  Model finiteness issues encapsulated into strong solvers

Some Future Work

- Method(s) for turning solvers into strong solvers
- Implementation (CVC3, DPT)
- Extension to non-disjoint combination
  (possibly built on combination framework of [Ghilardi et al., 2007])

Thank you

Parametricity

Parametric Type Operators

Fix a signature $\Sigma = \langle O \mid K \rangle$ and a Σ-structure $\mathcal{S}$

An n-ary operator $F \in O$ is *parametric in $\mathcal{S}$* if there exists a related n-ary operation $F^\sharp$ on binary relations

Parametric Type Operators

Fix a signature $\Sigma = \langle O \mid K \rangle$ and a Σ-structure $\mathcal{S}$

An n-ary operator $F \in O$ is *parametric in* $\mathcal{S}$ if there exists a related n-ary operation $F^\sharp$ on binary relations

that

1. preserves partial bijections
2. preserves identity relations
3. distributes over relational composition

Parametric Type Operators

Fix a signature $\Sigma = \langle O \mid K \rangle$ and a Σ-structure $\mathcal{S}$

An n-ary operator $F \in O$ is *parametric in $\mathcal{S}$* if there exists a related n-ary operation $F^\sharp$ on binary relations

such that

for all partial bijections $R_1 : A_1 \leftrightarrow B_1,\ \ldots,\ R_n : A_n \leftrightarrow B_n,$
$S_1 : C_1 \leftrightarrow A_1,\ \ldots,\ S_n : C_n \leftrightarrow A_n,$

1. $F^\sharp(R_1, \ldots, R_n)$ is a partial bijection in $F^{\mathcal{S}}(A_1, \ldots, A_n) \leftrightarrow F^{\mathcal{S}}(B_1, \ldots, B_n)$
2. $F^\sharp(R_1, \ldots, R_n) \circ F^\sharp(S_1, \ldots, S_n) = F^\sharp(R_1 \circ S_1, \ldots, R_n \circ S_n)$
3. $F^\sharp(id_{A_1}, \ldots, id_{A_1}) = id_{F(A_1, \ldots, A_n)}$

Parametric Type Operators: Example

Assume List $\in O$ and List$^{\mathcal{S}}$ is the list operator

Define List$^{\sharp}$ so that for all $R : A \leftrightarrow B$

- List$^{\sharp}(R)$: List$^{\mathcal{S}}(A) \leftrightarrow$ List$^{\mathcal{S}}(B)$
- $(l_A, l_B) \in$ List$^{\sharp}(R)$ iff $l_A = [a_1, \ldots, a_n]$, $l_B = [b_1, \ldots, b_n]$ and $(a_i, b_i) \in R$ for all i.

Parametric Type Operators: Example

Assume $\mathsf{List} \in O$ and $\mathsf{List}^{\mathcal{S}}$ is the list operator

Define $\mathsf{List}^{\sharp}$ so that for all $R : A \leftrightarrow B$

- $\mathsf{List}^{\sharp}(R) : \mathsf{List}^{\mathcal{S}}(A) \leftrightarrow \mathsf{List}^{\mathcal{S}}(B)$
- $(l_A, l_B) \in \mathsf{List}^{\sharp}(R)$ iff $l_A = [a_1, \ldots, a_n]$, $l_B = [b_1, \ldots, b_n]$ and $(a_i, b_i) \in R$ for all i.

Then List is parametric in $\mathcal{S}$:

for all composable partial bjections R and S and sets C

1. $\mathsf{List}^{\sharp}(R)$ is a partial bijection
2. $\mathsf{List}^{\sharp}(R) \circ \mathsf{List}^{\sharp}(S) = \mathsf{List}^{\sharp}(R \circ S)$
3. $\mathsf{List}^{\sharp}(id_C) = id_{\mathsf{List}^{\mathcal{S}}(C)}$

Parametric Structures

Fix a signature $\Sigma = \langle O \mid K \rangle$ and a Σ-structure $\mathcal{S}$

We can define a natural notion of parametricity for function symbols as well (see [Krstic et al., TACAS'07])

The structure $\mathcal{S}$ is *parametric* if every $F \in O \setminus \{\Rightarrow\}$ and every $f \in K$ are parametric