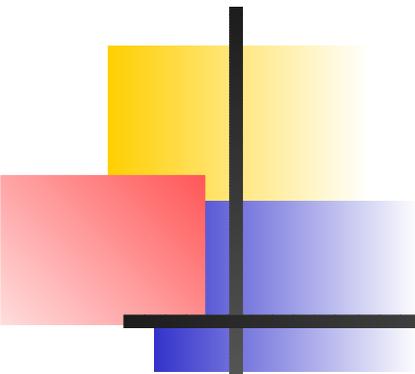


# Combined Satisfiability Modulo Parametric Theories

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

\*Strategic CAD Labs, Intel

\*\*The University of Iowa

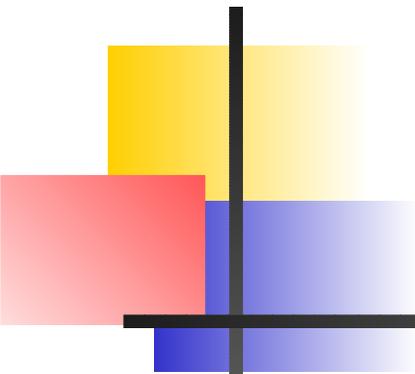


# This Talk

---

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

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

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

## Output:

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

# SMT Solvers over Multiple Theories

## Input:

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

## Output:

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

## Main Idea:

1. Input  $S$  is *purified* into equisatisfiable  $S_1, \dots, S_n$ ;
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

$\mathcal{T}_2$  = linear arithmetic

**Input set:**

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

**Purified sets:**

$$S_1 = \begin{cases} l_1 \neq l_2, \\ y_1 = \text{head}(l_2), \\ l = \text{tail}(l_2), \\ l_1 = x :: l, \\ y_2 = \text{head}(l), y_3 = \text{head}(\text{tail } l_1) \end{cases}$$

$$S_2 = \begin{cases} y_1 \leq x, \\ y_2 - y_3 + x \leq y_1 \end{cases}$$

# Nelson-Oppen: Example

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

# Nelson-Oppen: Example

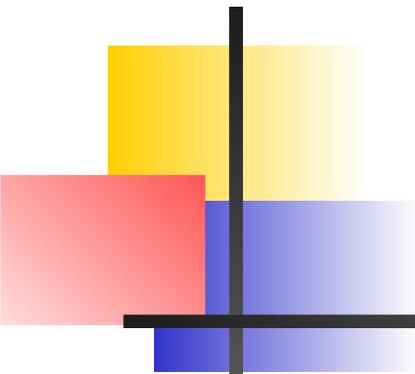
$S_1$	$S_2$
$l_1 \neq l_2$	$y_1 \leq x$
$y_1 = \text{head}(l_2)$	$y_2 - y_3 + x \leq y_1$
$l = \text{tail}(l_2)$	
$l_1 = x :: l$	
$y_2 = \text{head}(l)$	
$y_3 = \text{head}(\text{tail } l_1)$	
$\longrightarrow$	$y_2 = y_3$

# Nelson-Oppen: Example

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

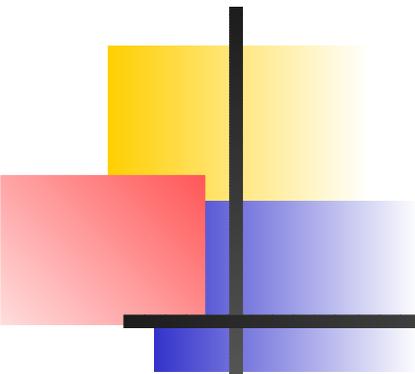
# Nelson-Oppen: Example

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



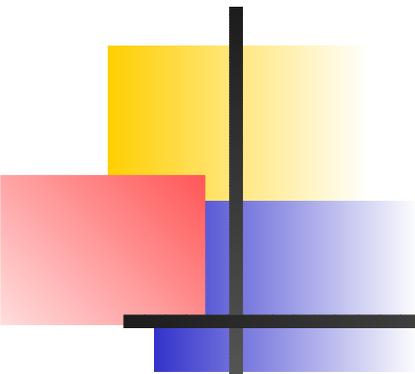
# Correctness of Nelson-Oppen

- The combination procedure is **sound** for any  $\mathcal{T}_1, \dots, \mathcal{T}_n$ :  
if it returns “Unsatisfiable”, then its input  $S$  is unsatisfiable in  $\mathcal{T}_1 + \dots + \mathcal{T}_n$



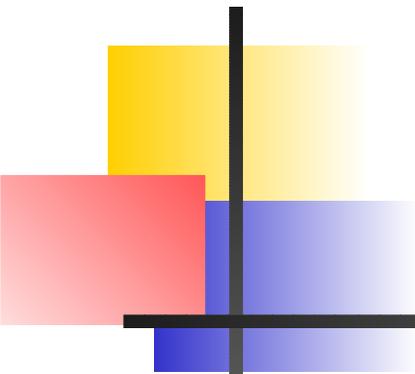
# Correctness of Nelson-Oppen

- The combination procedure is **sound** for any  $\mathcal{T}_1, \dots, \mathcal{T}_n$ :  
if it returns “Unsatisfiable”, then its input  $S$  is unsatisfiable in  $\mathcal{T}_1 + \dots + \mathcal{T}_n$
- It is **complete** when



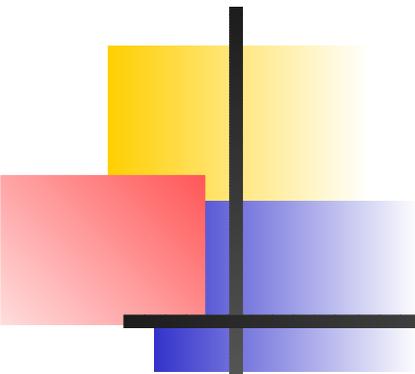
# Correctness of Nelson-Oppen

- The combination procedure is **sound** for any  $\mathcal{T}_1, \dots, \mathcal{T}_n$ :  
if it returns “Unsatisfiable”, then its input  $S$  is unsatisfiable in  $\mathcal{T}_1 + \dots + \mathcal{T}_n$
- It is **complete** when
  1.  $\mathcal{T}_1, \dots, \mathcal{T}_n$  are pairwise signature-disjoint, and



# Correctness of Nelson-Oppen

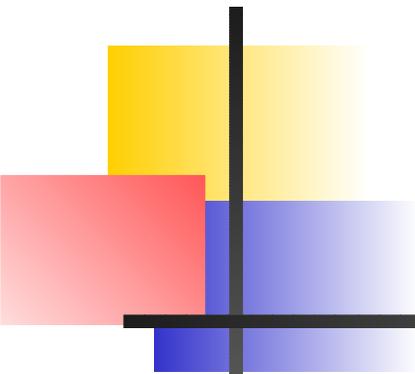
- The combination procedure is **sound** for any  $\mathcal{T}_1, \dots, \mathcal{T}_n$ :  
if it returns “Unsatisfiable”, then its input  $S$  is unsatisfiable in  $\mathcal{T}_1 + \dots + \mathcal{T}_n$
- It is **complete** when
  1.  $\mathcal{T}_1, \dots, \mathcal{T}_n$  are pairwise signature-disjoint, 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  $\varphi$  can be amalgamated into a model of  $\varphi$

# 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  $\varphi$  can be amalgamated into a model of  $\varphi$
- 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  $\varphi$  can be amalgamated into a model of  $\varphi$
- 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  $\varphi$  can be amalgamated into a model of  $\varphi$
- 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”

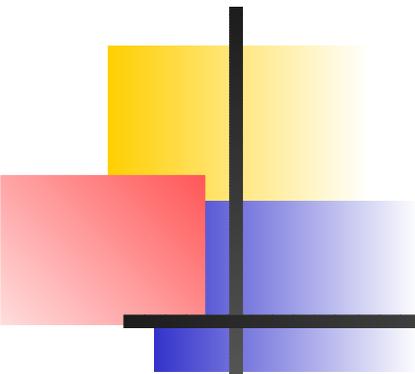
$\mathcal{T}_2$  = theory of Boolean rings (not stably-infinite)

## 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

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

# Parametricity, Not Stable Infiniteness: Example

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

$$\left( \begin{array}{ccccc} x_1 & x_2 & x & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{array} \right) \models_{\mathcal{T}_{\text{List}}} \Phi_{\text{List}} \cup \Delta \quad \left( \begin{array}{cccc} x_1 & x_2 & x & z \\ 1 & 2 & 2 & 1 \end{array} \right) \models_{\mathcal{T}_{\text{Int}}} \Phi_{\text{Int}} \cup \Delta$$

# Parametricity, Not Stable Infiniteness: Example

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

$$\left( \begin{array}{cccccc} x_1 & x_2 & x & l_1 & l_2 & \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] & \end{array} \right) \models_{\mathcal{T}_{\text{List}}} \Phi_{\text{List}} \cup \Delta \quad \left( \begin{array}{cccccc} x_1 & x_2 & x & z & & \\ 1 & 2 & 2 & 1 & & \end{array} \right) \models_{\mathcal{T}_{\text{Int}}} \Phi_{\text{Int}} \cup \Delta$$

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

# Parametricity, Not Stable Infiniteness: Example

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

$$\left( \begin{array}{ccccc} x_1 & x_2 & x & l_1 & l_2 \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] \end{array} \right) \models_{\mathcal{T}_{\text{List}}} \Phi_{\text{List}} \cup \Delta \quad \left( \begin{array}{ccccc} x_1 & x_2 & x & z & \\ 1 & 2 & 2 & 1 & \end{array} \right) \models_{\mathcal{T}_{\text{Int}}} \Phi_{\text{Int}} \cup \Delta$$

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

$$\left( \begin{array}{ccccc} x_1 & x_2 & x & l_1 & l_2 \\ m & n & n & [m, n] & [n, n] \end{array} \right) \models \Phi_{\text{List}} \cup \Delta \text{ as well}$$

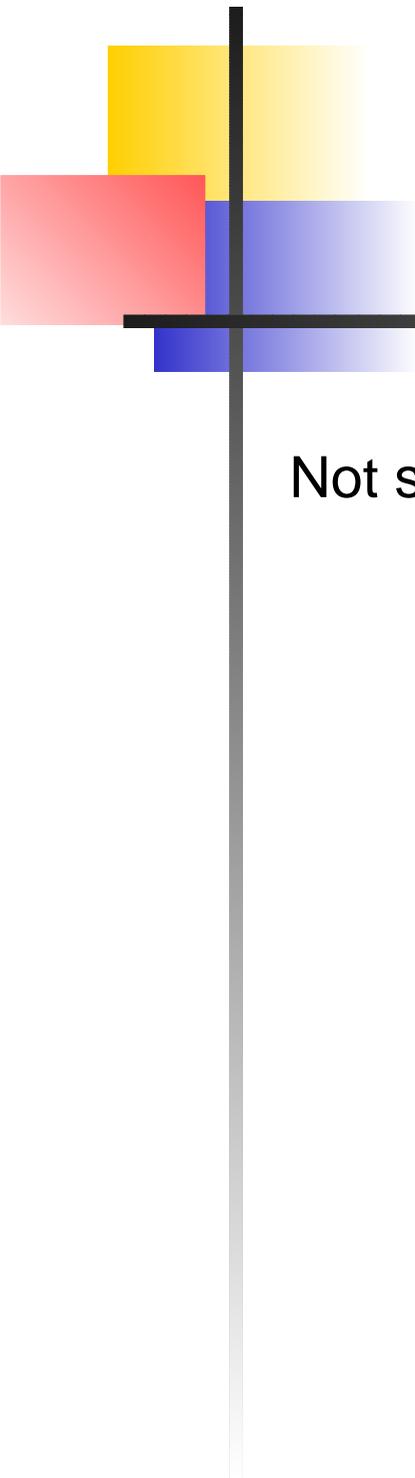
# Parametricity, Not Stable Infiniteness: Example

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

$$\left( \begin{array}{cccccc} x_1 & x_2 & x & l_1 & l_2 & \\ \blacktriangle & \bullet & \bullet & [\blacktriangle, \bullet] & [\bullet, \bullet] & \end{array} \right) \models_{\mathcal{T}_{\text{List}}} \Phi_{\text{List}} \cup \Delta \quad \left( \begin{array}{cccccc} x_1 & x_2 & x & z & & \\ 1 & 2 & 2 & 1 & & \end{array} \right) \models_{\mathcal{T}_{\text{Int}}} \Phi_{\text{Int}} \cup \Delta$$

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

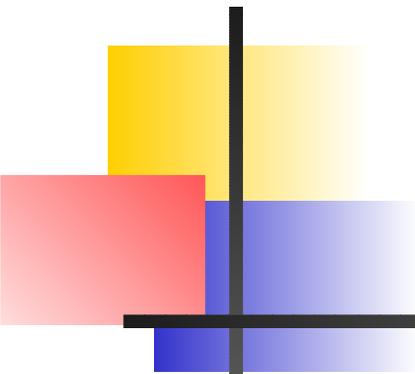
to construct a model for  $\Phi_{\text{List}} \cup \Phi_{\text{Int}} \cup \Delta$ , we can use the blue assignment to  $x_1, x_2, x$



# 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:**  $\alpha, \beta, \alpha_1, \beta_1, \dots$

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

**Ex:** Bool/0, Int/0, List/1, Arr/2,  $\Rightarrow$ /2, ...

Types( $O, V$ ), set of *types*, terms over  $O, V$

**Ex:** Int, List( $\alpha$ ), List(Int), Arr(Int, List( $\alpha$ )), List( $\alpha$ )  $\Rightarrow$  Int, ...

# FOLP Syntax

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

**Constants:**  $K$ , set of symbols each with an associated *principal* type  $\tau$

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

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

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

# FOLP Syntax

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

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

**$\Sigma$ -Terms of First-order Type  $\tau$ :**  $T_\tau(K, X)$ , defined as usual

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

**First-order (Quantifier-free) Formulas:** Terms in  $T_{\text{Bool}}(K, X)$

# 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  $\text{TypeVars}(\tau)$  where  $f^\tau$ )
- s.t. Bool,  $\Rightarrow$ , and =, ite,  $\wedge$ , ... 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  $\text{TypeVars}(\tau)$  where  $f^{\tau}$ )

s.t.  $\text{Bool}$ ,  $\Rightarrow$ ,  $\text{and}$ ,  $=$ ,  $\text{ite}$ ,  $\wedge$ ,  $\dots$  are the interpreted in the usual way.

**Ex 1:**

$\text{Int}^{\mathcal{S}}$  equals the integers

$\text{List}^{\mathcal{S}}$  maps an input set  $A$  to the set of finite lists over  $A$

$\text{Arr}^{\mathcal{S}}$  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  $\text{TypeVars}(\tau)$  where  $f^\tau$ )

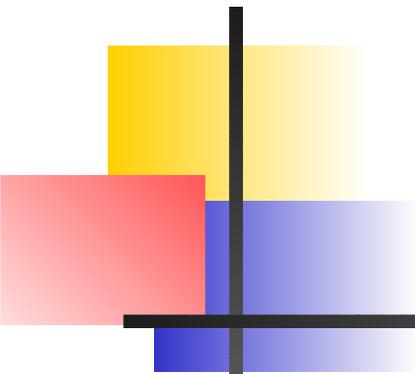
s.t.  $\text{Bool}$ ,  $\Rightarrow$ ,  $\text{and}$ ,  $=$ ,  $\text{ite}$ ,  $\wedge$ ,  $\dots$  are the interpreted in the usual way.

**Ex 2:**

$\text{head}^\mathcal{S}$  family  $\{\text{head}[A_1] \mid A_1 \text{ is a set}\}$  (since  $\text{head}^{\text{List}(\alpha) \Rightarrow \alpha}$ )

$\text{read}^\mathcal{S}$  family  $\{\text{read}[A_1, A_2] \mid A_1, A_2 \text{ are sets}\}$   
(since  $\text{read}^{\text{Arr}(\alpha_1, \alpha_2), \alpha_1 \Rightarrow \alpha_2}$ )

$+^\mathcal{S}$  singleton family (since  $+^{\text{Int}, \text{Int} \Rightarrow \text{Int}}$ )



# FOLP Semantics

For every signature  $\Sigma = \langle O \mid K \rangle$ ,  $\Sigma$ -structure  $\mathcal{S}$ , **type environment**  $\iota$ , **term environment**  $\rho$ , and  $\Sigma$ -formula  $\varphi$ ,

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

# FOLP Semantics

For every signature  $\Sigma = \langle O \mid K \rangle$ ,  $\Sigma$ -structure  $\mathcal{S}$ , **type environment**  $\iota$ , **term environment**  $\rho$ , and  $\Sigma$ -formula  $\varphi$ ,

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

## Satisfiability

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

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

# FOLP Semantics

For every signature  $\Sigma = \langle O \mid K \rangle$ ,  $\Sigma$ -structure  $\mathcal{S}$ , **type environment**  $\iota$ , **term environment**  $\rho$ , and  $\Sigma$ -formula  $\varphi$ ,

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

## Satisfiability

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

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

## Cardinality Constraints

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

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

# The Equality Structure

Let

$$K_{\text{Eq}} = \{ \lambda \alpha, \alpha \Rightarrow \text{Bool}, \top^{\text{Bool}}, \neg^{\text{Bool} \Rightarrow \text{Bool}}, \text{ite}^{\text{Bool}, \alpha, \alpha \Rightarrow \alpha}, \dots \}$$

$$\Sigma_{\text{Eq}} = \langle \text{Bool}, \Rightarrow \mid K_{\text{Eq}} \rangle$$

$$\mathcal{S}_{\text{Eq}} = \text{the unique } \Sigma_{\text{Eq}}\text{-structure}$$

# The Equality Structure

Let

$$K_{\text{Eq}} = =_{\alpha, \alpha \Rightarrow \text{Bool}}, \top^{\text{Bool}}, \neg^{\text{Bool} \Rightarrow \text{Bool}}, \text{ite}^{\text{Bool}, \alpha, \alpha \Rightarrow \alpha}, \dots$$

$$\Sigma_{\text{Eq}} = \langle \text{Bool}, \Rightarrow \mid K_{\text{Eq}} \rangle$$

$$\mathcal{S}_{\text{Eq}} = \text{the unique } \Sigma_{\text{Eq}}\text{-structure}$$

**Note:**  $\mathcal{S}_{\text{Eq}}$  models

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

# The Equality Structure

Let

$$K_{\text{Eq}} = =_{\alpha, \alpha \Rightarrow \text{Bool}}, \top^{\text{Bool}}, \neg^{\text{Bool} \Rightarrow \text{Bool}}, \text{ite}^{\text{Bool}, \alpha, \alpha \Rightarrow \alpha}, \dots$$

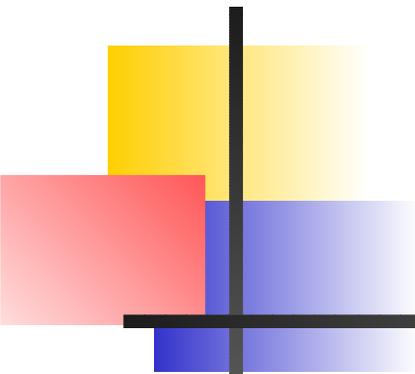
$$\Sigma_{\text{Eq}} = \langle \text{Bool}, \Rightarrow \mid K_{\text{Eq}} \rangle$$

$$\mathcal{S}_{\text{Eq}} = \text{the unique } \Sigma_{\text{Eq}}\text{-structure}$$

**Note:**  $\mathcal{S}_{\text{Eq}}$  models

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

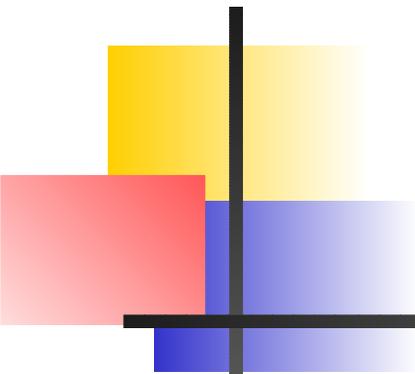
**Fact:** The satisfiability in  $\mathcal{S}_{\text{Eq}}$  of first-order  $\Sigma_{\text{Eq}}$ -formulas is **decidable** (with the usual congruence closure algorithms)



# 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*

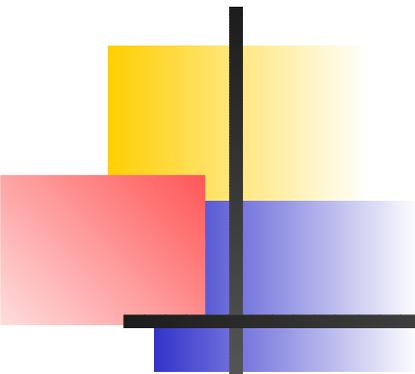


# Parametricity [TACAS'07]

---

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

$$\Sigma_{\text{Int}} = \langle \text{Int} \mid 0^{\text{Int}}, 1^{\text{Int}}, +^{\text{Int}^2 \rightarrow \text{Int}}, -^{\text{Int}^2 \rightarrow \text{Int}}, \times^{\text{Int}^2 \rightarrow \text{Int}}, \leq^{\text{Int}^2 \rightarrow \text{Bool}}, \dots \rangle$$

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

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

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

$$\Sigma_{\text{BitVec32}} = \dots$$

$$\Sigma_{\text{Sets}} = \dots$$

$$\Sigma_{\text{Multisets}} = \dots$$

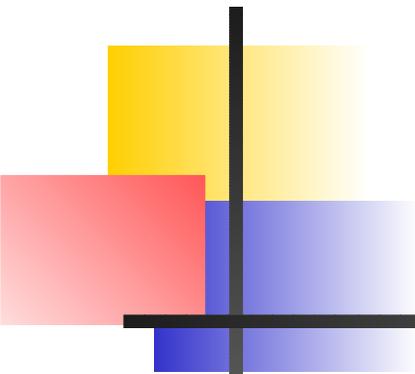
(All the above signatures implicitly include the signature  $\Sigma_{\text{Eq}}$ )



# Combining Signatures and Structures

## Disjoint Signatures

Signatures that share exactly the symbols of  $\Sigma_{Eq}$



# Combining Signatures and Structures

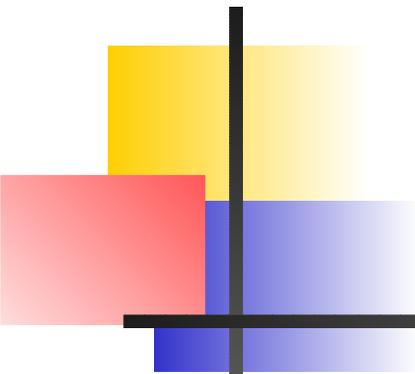
---

## Disjoint Signatures

Signatures that share exactly the symbols of  $\Sigma_{Eq}$

## Combination of Disjoint Signatures $\Sigma_1, \Sigma_2$

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



# Combining Signatures and Structures

## Disjoint Signatures

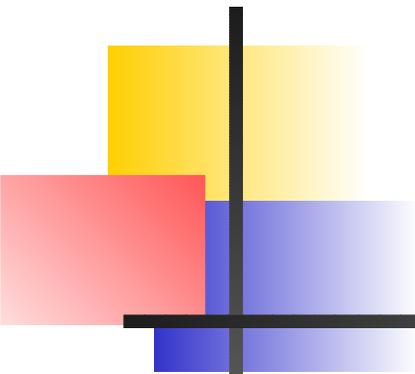
Signatures that share exactly the symbols of  $\Sigma_{Eq}$

## Combination of Disjoint Signatures $\Sigma_1, \Sigma_2$

$\Sigma_1 + \Sigma_2 = \langle O_1 \cup O_2 \mid K_1 \cup K_2 \rangle$  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  $\Sigma_i$ -symbols exactly like  $\mathcal{S}_i$  for  $i = 1, 2$ .



# Combining Signatures and Structures

## Disjoint Signatures

Signatures that share exactly the symbols of  $\Sigma_{Eq}$

## Combination of Disjoint Signatures $\Sigma_1, \Sigma_2$

$\Sigma_1 + \Sigma_2 = \langle O_1 \cup O_2 \mid K_1 \cup K_2 \rangle$  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  $\Sigma_i$ -symbols exactly like  $\mathcal{S}_i$  for  $i = 1, 2$ .

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

# Pure and Semipure Terms

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

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

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

**Ex**

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

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

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

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

# Pure and Semipure Terms

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

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

- ***i-semipure*** if it has signature  $\langle O_1 \cup \dots \cup O_n \mid K_i \rangle$
- ***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  $t^{\text{pure}}$  of  $t$

**Ex**

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

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

# Pure and Semipure Terms

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

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

iff

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

for some suitable set  $\Phi_i^{\text{card}}$  of cardinality constraints computable from  $\Phi_i$

**Ex**

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

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

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

# Why Cardinality Constraints are Needed

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

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

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

- $\Phi$  and  $\Phi_1$  are  $(\mathcal{S}_{\text{Int}} + \mathcal{S}_{\text{List}})$ -satisfiable,  $\Phi_2$  is not

# Why Cardinality Constraints are Needed

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

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

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

- $\Phi$  and  $\Phi_1$  are  $(\mathcal{S}_{\text{Int}} + \mathcal{S}_{\text{List}})$ -satisfiable,  $\Phi_2$  is not
- $\mathcal{S}_{\text{List}}$ -solver can't take  $\Phi_1$  or  $\Phi_2$  as input: they are not  $\Sigma_{\text{List}}$ -pure

# Why Cardinality Constraints are Needed

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

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

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

- $\Phi$  and  $\Phi_1$  are  $(\mathcal{S}_{\text{Int}} + \mathcal{S}_{\text{List}})$ -satisfiable,  $\Phi_2$  is not
- $\mathcal{S}_{\text{List}}$ -solver can't take  $\Phi_1$  or  $\Phi_2$  as input: they are not  $\Sigma_{\text{List}}$ -pure
- Instead of  $\Phi_1$ , it gets  $\Phi = \Phi_1^{\text{pure}}$  with **cardinality constraint**  $\emptyset$

# Why Cardinality Constraints are Needed

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

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

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

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

# Towards Nelson-Oppen Combination: Purification

We turn each query  $\Phi$  into the *purified form*

$$\Phi_B \cup \Phi_E \cup \Phi_1 \cup \dots \cup \Phi_n$$

where

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

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

$$\Phi_B = \{p \vee q\}$$

$$\Phi_E = \{p \equiv y = x\},$$

$$\Phi_{\text{Eq}} = \{y = f(x), u = f(z)\}$$

$$\Phi_{\text{Int}} = \{q \equiv u > x \ z = 2 * x - y, \}$$

# Towards a Combination Theorem

Let

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

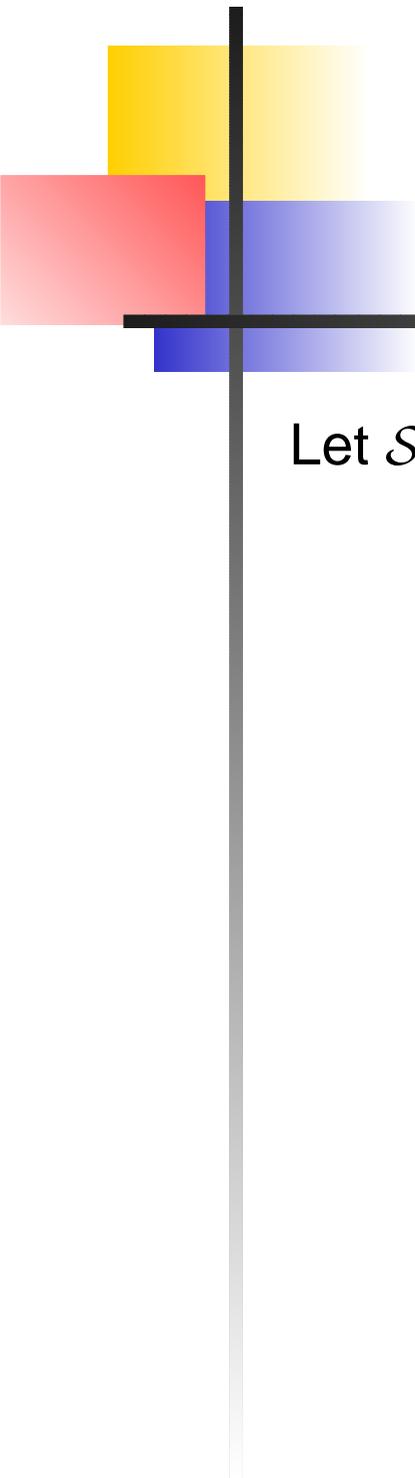
An *assignment*  $M$  of  $A$  is a consistent set of literals with atoms in  $A$

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

**Ex**

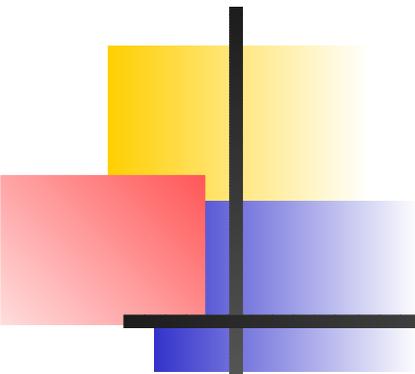
Partition:  $\{\{x^{\tau_1}, y^{\tau_1}, z^{\tau_1}\}, \{u^{\tau_2}, v^{\tau_2}\}, \{w^{\tau_3}\}\}$

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



# Main Result: A Combination Theorem for FOLP

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



# Main Result: A Combination Theorem for FOLP

---

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

A query

$$\Phi = \Phi_B \cup \Phi_E \cup \Phi_1 \cup \dots \cup \Phi_n$$

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

# Main Result: A Combination Theorem for FOLP

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

A query

$$\Phi = \Phi_B \cup \Phi_E \cup \Phi_1 \cup \dots \cup \Phi_n$$

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

- an assignment  $M$  of the atoms in  $\Phi_B$  and
- an arrangement  $\Delta$  of the non-Bool variables in  $\Phi$

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  $i = 1, \dots, n$

# Main Theoretical Requirement: Flexible Structures

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

- every query  $\Phi$ ,
- 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  $\langle \iota^{\text{up}(\kappa)}, \rho^{\text{up}(\kappa)} \rangle$  and  $\langle \iota^{\text{down}}, \rho^{\text{down}} \rangle$  satisfying  $\Phi$  s.t.

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

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

# Main Theoretical Requirement: Flexible Structures

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

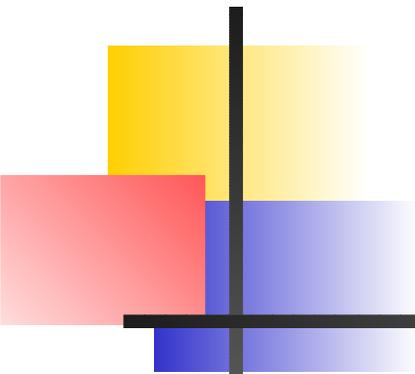
- every query  $\Phi$ ,
- every injective  $\langle \iota, \rho \rangle$  such that  $\langle \iota, \rho \rangle \models_{\mathcal{S}} \Phi$ ,
- every  $\alpha \in V$ ,
- every  $\kappa > |\iota(\alpha)|$

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

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

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

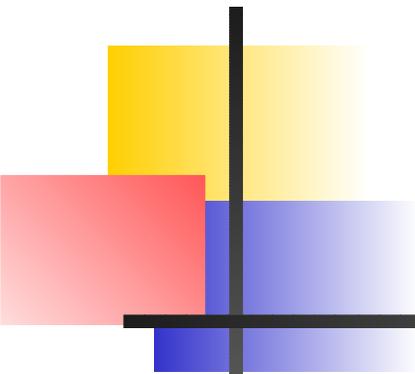
# Closest Related Work [Ranise et al., FroCoS'05]

## 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:

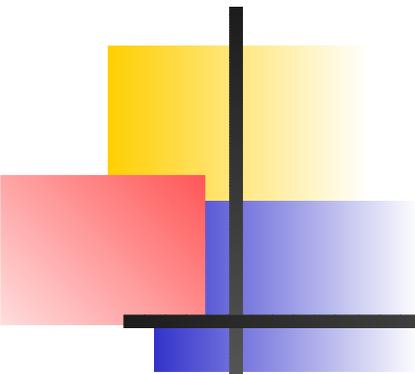
Theory solvers are combined, soundly and completely, with a Nelson-Oppen style method that also guesses *equalities over some additional terms* computed from the input query.



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

## That work vs. This work

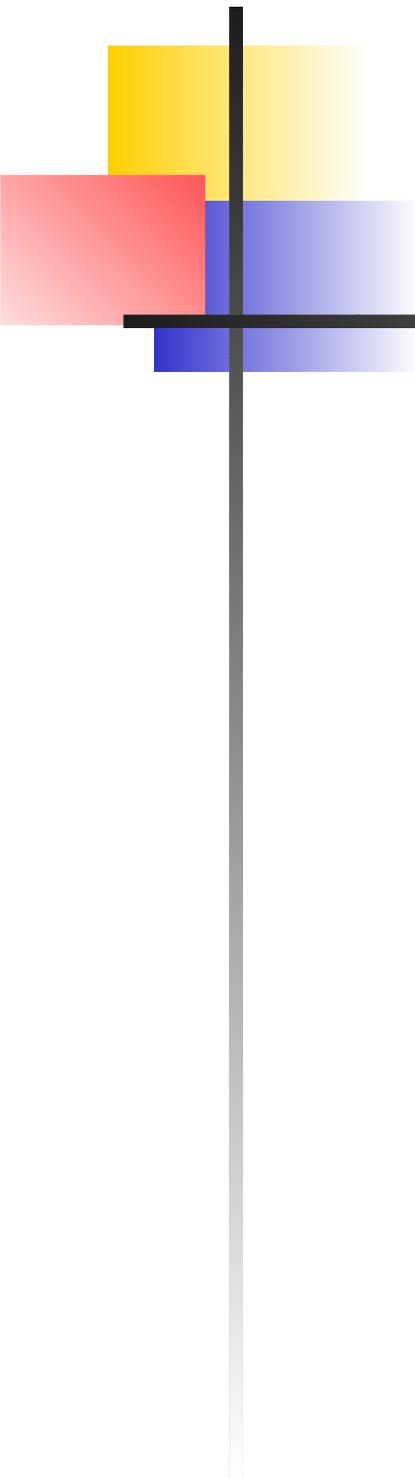
- Theory combinations via signature push-outs  
Theory combinations via type parameter instantiation
- Politeness assumption on theories  
Flexibility assumption on structures
- Politeness proven per theory  
Parametricity as general sufficient condition for flexibility
- Idea of parametricity is implicit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



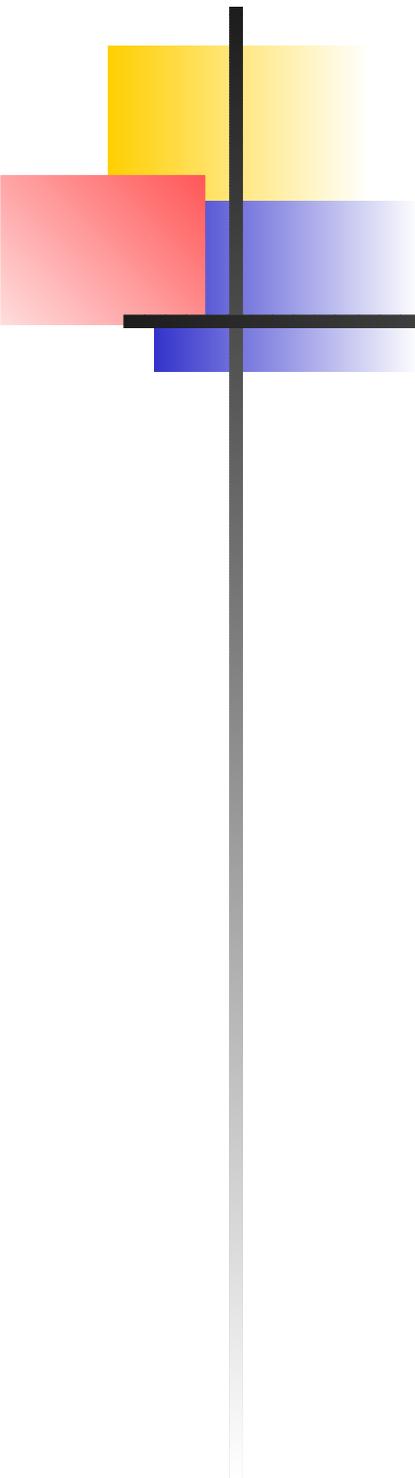
# Some Future Work

---

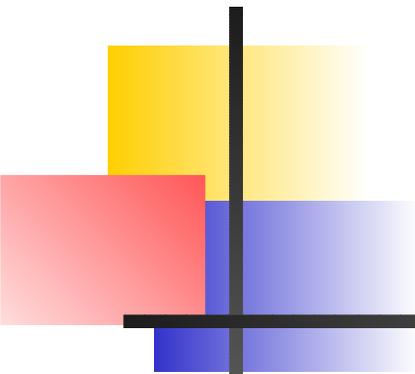
- Method(s) for turning solvers into strong solvers
- Implementation (CVC3, DPT)
- Extension to non-disjoint combination  
(possibly built on combination framework of [Ghilardi et al., 2007])



**Thank you**



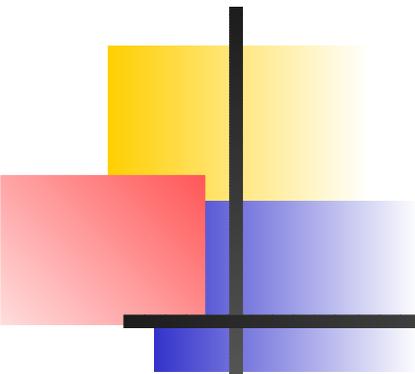
# Parametricity



# Parametric Type Operators

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

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



# Parametric Type Operators

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

An  $n$ -ary operator  $F \in O$  is *parametric in  $\mathcal{S}$*  if there exists a related  $n$ -ary operation  $F^\#$  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  $\Sigma$ -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**  $R_1 : A_1 \leftrightarrow B_1, \dots, R_n : A_n \leftrightarrow B_n,$   
 $S_1 : C_1 \leftrightarrow A_1, \dots, S_n : C_n \leftrightarrow A_n,$

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

# Parametric Type Operators: Example

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

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

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

# Parametric Type Operators: Example

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

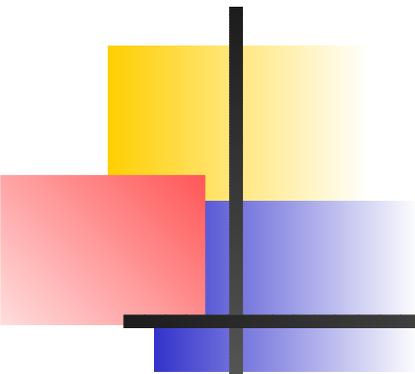
Define  $\text{List}^{\sharp}$  so that for all  $R : A \leftrightarrow B$

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

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

for all composable partial bjections  $R$  and  $S$  and sets  $C$

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



# Parametric Structures

Fix a signature  $\Sigma = \langle O \mid K \rangle$  and a  $\Sigma$ -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