

On Interpolation in Local Theory Extensions

Viorica Sofronie-Stokkermans

Max-Planck-Institut für Informatik

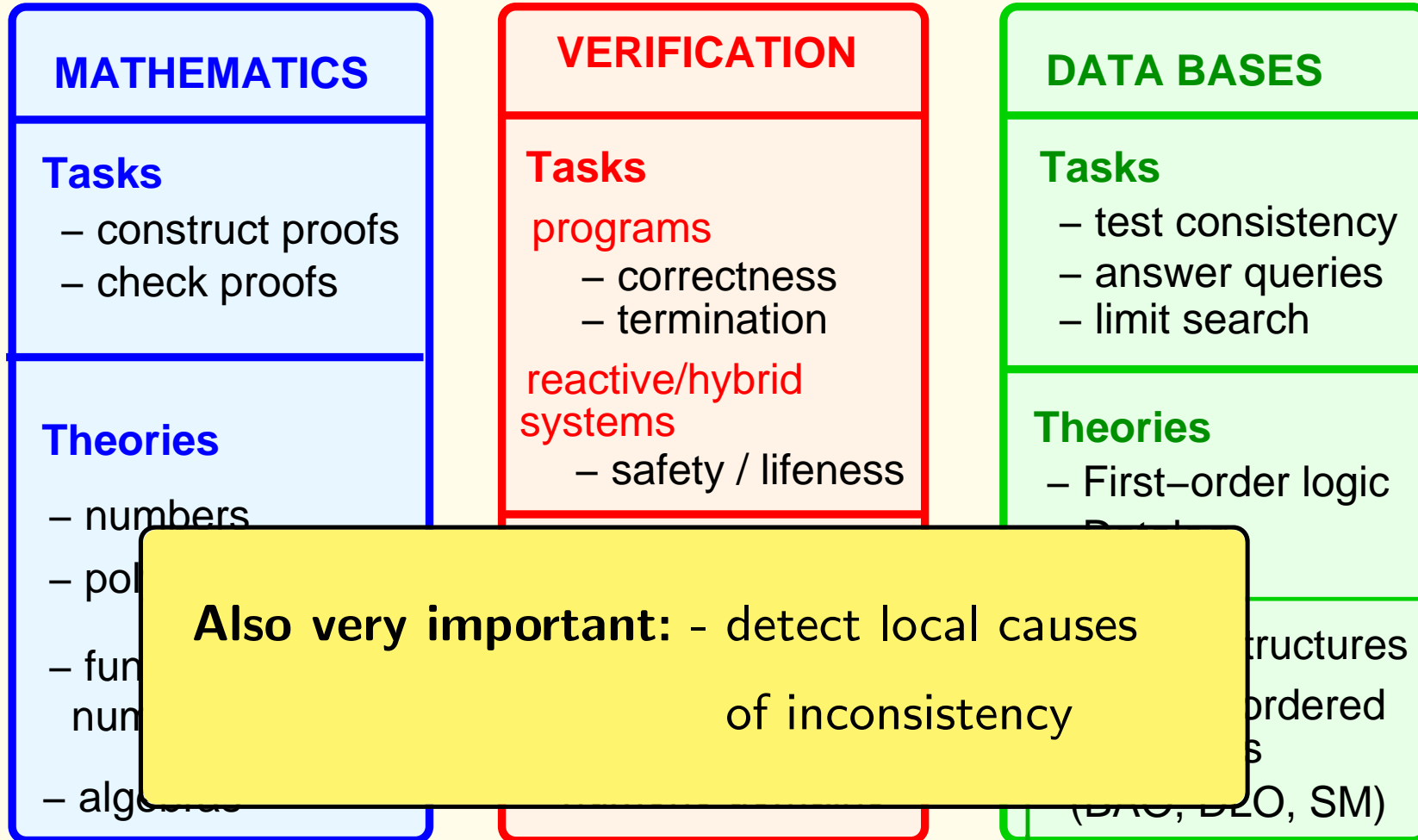
Saarbrücken

IJCAR 2006, August 17-20, 2006, Seattle

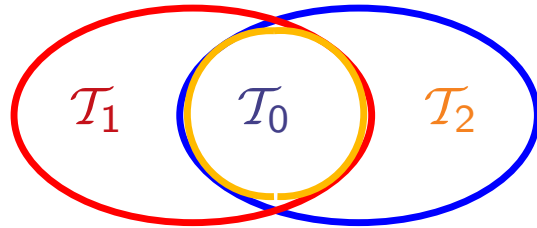
Motivation

MATHEMATICS	VERIFICATION	DATA BASES
Tasks <ul style="list-style-type: none">– construct proofs– check proofs	Tasks <ul style="list-style-type: none">programs<ul style="list-style-type: none">– correctness– terminationreactive/hybrid systems<ul style="list-style-type: none">– safety / liveness	Tasks <ul style="list-style-type: none">– test consistency– answer queries– limit search
Theories <ul style="list-style-type: none">– numbers– polynomials– functions over numeric domains– algebras	Theories <ul style="list-style-type: none">– numbers– data types– functions over numeric domains	Theories <ul style="list-style-type: none">– First-order logic– Datalog– ...– Kripke structures– Lattice-ordered structures (BAO, DLO, SM)
Method: test entailment / satisfiability w.r.t. background theory		

Motivation



Motivation: Modular reasoning



\mathcal{T}_0 : Σ_0 -theory.

\mathcal{T}_i : Σ_i -theory; $\mathcal{T}_0 \subseteq \mathcal{T}_i$ Σ_i extension of Σ_0 .

Can use provers for $\mathcal{T}_1, \mathcal{T}_2$ as black-boxes to prove theorems in $\mathcal{T}_1 \cup \mathcal{T}_2$?

$$G_1 \wedge G_2 \models_{\mathcal{T}_1 \cup \mathcal{T}_2} \perp$$

Which information needs to be exchanged between the provers?

$$G_1 \models_{\mathcal{T}_1} I \quad I \wedge G_2 \models_{\mathcal{T}_2} \perp$$

Example

Reason about **lists** of integers and **monotone** functions over **integers**

Motivation: Distributed databases

Chem	
Primitive concepts (C_0):	process, reaction, subst, organic, anorganic
Constraints (Γ_0):	$\text{organic} \sqcap \text{anorganic} = \emptyset$ $\text{organic} \sqsubseteq \text{subst}, \quad \text{anorganic} \sqsubseteq \text{subst}$

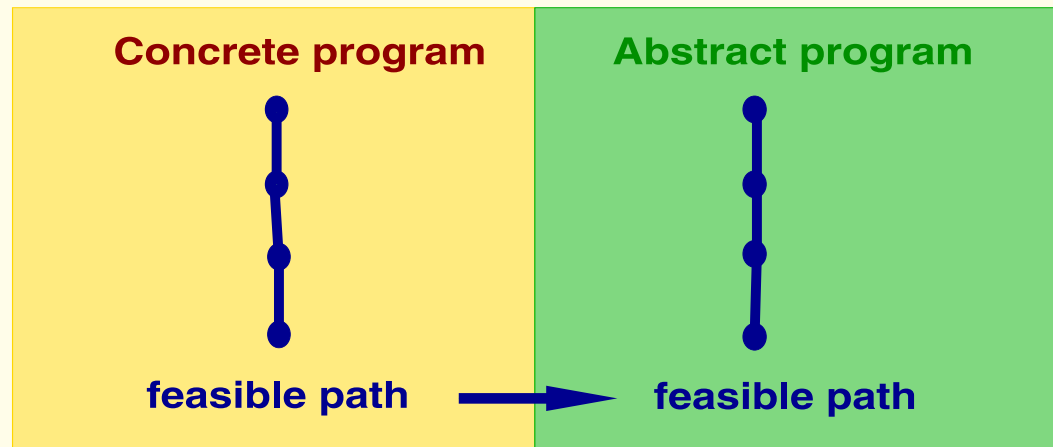
AnorgChem	
C_1 : cat-oxydation, oxydation	R_1 : catalyses
T_1 : cat-oxydation = subst \sqcap \exists catalyzes.oxydation	
Γ_1 : reaction \sqsubseteq oxydation	
$\emptyset \neq \text{cat-oxydation} \sqsubseteq \text{anorganic}$	

BioChem	
C_2 : enzyme	R_2 : produces, catalyses
T_2 : reaction = process \sqcap \exists produces.subst	
enzyme = organic \sqcap \exists catalyzes.reaction	
Γ_2 : enzyme $\neq \emptyset$	

Chem + **AnorgChem** + **BioChem** inconsistent

Find mistake: local explanation for inconsistency (in the common language)

Motivation: Abstraction-based Verification



location unreachable ← location unreachable
check feasibility ← location reachable



conjunction of constraints: $\phi(1) \wedge Tr(1, 2) \wedge \dots \wedge Tr(n - 1, n) \wedge \neg \text{safe}(n)$

- **satisfiable**: feasible path

- **unsatisfiable**: refine abstract program s.t. the path is not feasible

[McMillan 2003-2006] use 'local causes of inconsistency'

↳ compute interpolants

Interpolation

\mathcal{T} theory; A, B formulae such that $A \models_{\mathcal{T}} B$

Does there exist a formula I , containing only symbols occurring in both A and B such that $A \models_{\mathcal{T}} I$ and $I \models_{\mathcal{T}} B$?

If so, I is an **interpolant** for ϕ and ψ .

Theorem [Craig 1957] First order logic has the interpolation property.

(but even if A and B are ground clauses, I may contain quantifiers)

Interpolation

\mathcal{T} theory; A, B formulae such that $A \wedge B \models_{\mathcal{T}} \perp$

Does there exist a formula I , containing only symbols occurring in both A and B such that $A \models_{\mathcal{T}} I$ and $I \wedge B \models_{\mathcal{T}} \perp$?

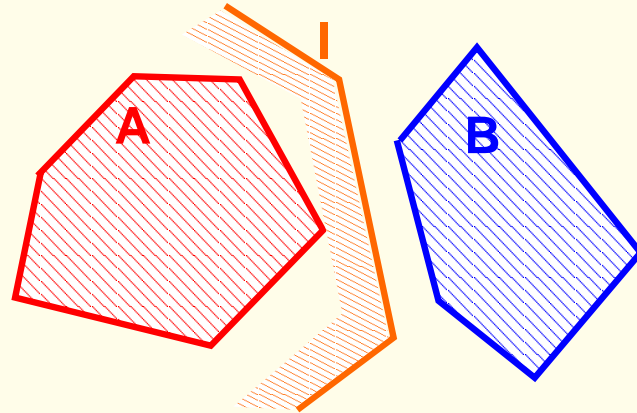
If so, I is an **interpolant** for ϕ and ψ .

Theorem [Craig 1957] First order logic has the interpolation property.

(but even if A and B are ground clauses, I may contain quantifiers)

Ground Interpolation

\mathcal{T} theory; A , B sets of *ground* (unit) clauses in the language of \mathcal{T} such that $A \wedge B \models_{\mathcal{T}} \perp$



Question: Can we construct a *ground* formula I , containing **only constants (and function symbols) common to A and B** such that

$$A \models_{\mathcal{T}} I \quad \text{and} \quad I \wedge B \models_{\mathcal{T}} \perp ?$$

If so, I is a **ground interpolant** of A and B

'local' explanation for the inconsistency of $A \wedge B$

Ground Interpolation

Links with amalgamation, injection transfer property

- in universal algebra [Jónsson'65, Bacsich'75, Wrónski'85]

Ground interpolants exist and can be found fast:

- propositional logic [Pudlak'97, Krajicek'97]
used to SAT-based model checking [McMillan'03]
- linear arithmetic (+ free function symbols) [McMillan'03,'04,'05]
- difference constraints (+ free function symbols) [Jhala, McMillan'06]
- combinations of theories [Yorsh, Musuvathi'05]
(stably infinite, disjoint signatures)

Our contributions

Method for computing interpolants in extensions of a base theory with a set of functions satisfying a set \mathcal{K} of clauses

- The method is general

It can be used if:

- \mathcal{T}_0 has some properties of linear arithmetic
- clauses \mathcal{K} have a special form
- hierarchical reasoning possible for $\mathcal{T}_0 \cup \mathcal{K} \mapsto$ **local extensions**
(test satisfiability of ground clauses \mapsto test satisfiability in \mathcal{T}_0)

Our contributions

Method for computing interpolants in extensions of a base theory with a set of functions satisfying a set \mathcal{K} of clauses

- The method is general
- Interpolants are computed in a hierarchical way
 - reduction to constructing interpolants in the base theory

Our contributions

Method for computing interpolants in local extensions of a base theory with a set of functions satisfying a set \mathcal{K} of clauses

- The method is general
- Interpolants are computed in a hierarchical way
- We identify classes of theory extensions for which this is possible

Our contributions

Method for computing interpolants in local extensions of a base theory with a set of functions satisfying a set \mathcal{K} of clauses

- The method is general
- Interpolants are computed in a hierarchical way
- We identify classes of theory extensions for which this is possible
- We discuss several application domains
 - modular reasoning in combinations of theories
 - reasoning in distributed data bases
 - verification

Structure of the talk

- **Local theory extensions**
- **Computing interpolants in local theory extensions**
- **Applications**
- **Conclusions, perspectives**

Local theory extensions

\mathcal{K} set of equational clauses; \mathcal{T}_0 theory; $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$

(Loc) $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is **local**, if for ground clauses G ,
 $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G$ has no (partial) model

[Ganzinger, VS, Waldmann'04, VS'05]

\mathcal{T}_1 local extension of \mathcal{T}_0 \longleftrightarrow $\text{Emb}(\mathcal{T}_0, \mathcal{T}_1)$

Examples of local extensions

Extensions of a theory \mathcal{T}_0 :

- with **free function symbols**
- with **monotone functions** for:

$\mathcal{T}_0 = \mathbb{R}$ theory of real numbers

$\mathcal{T}_0 \in \{\text{Posets, TotOrd, DenseTotOrd, Lat, SLat, DLat, BoolAlg}\}$

possibly subject to additional constraints

$\phi(x_1, \dots, x_n) \rightarrow f(x_1, \dots, x_n) \leq t(x_1, \dots, x_n)$ [t same monotonicity as f]

$\text{Mon}(f, g) \quad \wedge \quad (x \leq g(y) \rightarrow f(x) \leq y) \quad [f(g(y)) \leq y]$

$\text{Mon}(f, g) \quad \wedge \quad (x \leq g(y) \rightarrow f(x) \leq g(y)) \quad [f(g(y)) \leq g(y)]$

Examples of local extensions

Extensions of a theory \mathcal{T}_0 :

- with **free function symbols**
- with **monotone functions** for

$\mathcal{T}_0 = \mathbb{R}$ theory of real numbers

$\mathcal{T}_0 \in \{\text{Posets, TotOrd, D}\}$

• **Verification:**

sorted arrays \mapsto train positions

[Jacobs, VS, PDPAR'06]

controllers $\text{in}(\text{out}(L)) \leq L$

$0 \leq \text{out}(L), \text{in}(L) \leq L$

• **Knowledge representation:**

\mathcal{EL} description logic \mapsto SLat + Mon

possibly subject to additional constraints

$$\phi(x_1, \dots, x_n) \rightarrow f(x_1, \dots, x_n) \leq t(x_1, \dots, x_n)$$

$$\text{Mon}(f, g) \quad \wedge \quad (x \leq g(y) \rightarrow f(x) \leq y) \quad [f(g(y)) \leq y]$$

$$\text{Mon}(f, g) \quad \wedge \quad (x \leq g(y) \rightarrow f(x) \leq g(y)) \quad [f(g(y)) \leq g(y)]$$

Reasoning in local theory extensions

Locality: $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$ iff $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \perp$

Hierarchical reasoning [VS 2005]

– purify $\mathcal{K}[G]$ and G $\Rightarrow \mathcal{K}_0 \wedge G_0 \wedge \text{Def}$

\mapsto definitions Def for terms starting with extension functions

– reduce to satisfiability in \mathcal{T}_0 $\Rightarrow \mathcal{K}_0 \wedge G_0 \wedge \text{Con}[G]_0$

Example: Reasoning in local theory extensions

$$\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$$

$$\text{where } \mathcal{K}: \forall x (x \leq g(y) \rightarrow f(x) \leq y)$$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

Problem: Check whether $G \models_{\mathcal{T}_1} \perp$

$$\frac{G}{\begin{array}{l} d \leq g(a) \wedge a \leq c \\ b \leq d \wedge f(b) \not\leq c \end{array}}$$

Example: Reasoning in local theory extensions

$$\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$$

where $\mathcal{K}: \forall x (x \leq g(y) \rightarrow f(x) \leq y)$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

Problem: Check whether $G \models_{\mathcal{T}_1} \perp$ local theory extension

G	$\mathcal{K}[G] \wedge \text{Con}[G]$
$d \leq g(a) \wedge a \leq c$	$b \leq g(a) \rightarrow f(b) \leq a$
$b \leq d \wedge f(b) \not\leq c$	$a \triangleleft a \rightarrow g(a) \triangleleft g(a)$ (redundant)
	$b \triangleleft b \rightarrow f(b) \triangleleft f(b)$ $\triangleleft \in \{\leq, =\}$

Example: Reasoning in local theory extensions

$$\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$$

where $\mathcal{K}: \forall x (x \leq g(y) \rightarrow f(x) \leq y)$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

Problem: Check whether $G \models_{\mathcal{T}_1} \perp$ local theory extension

G	$\mathcal{K}[G] \wedge \text{Con}[G]$
$d \leq g(a) \wedge a \leq c$	$b \leq g(a) \rightarrow f(b) \leq a$
$b \leq d \wedge f(b) \not\leq c$	$a \triangleleft a \rightarrow g(a) \triangleleft g(a)$ (redundant)
	$b \triangleleft b \rightarrow f(b) \triangleleft f(b) \quad \triangleleft \in \{\leq, =\}$

Example: Reasoning in local theory extensions

$$\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$$

$$\text{where } \mathcal{K}: \forall x (x \leq g(y) \rightarrow f(x) \leq y)$$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

Problem: Check whether $G \models_{\mathcal{T}_1} \perp$ local theory extension

Def	G	$\mathcal{K}[G] \wedge \text{Con}[G]$
$g(a) = a_1$	$d \leq g(a) \wedge a \leq c$	$b \leq g(a) \rightarrow f(b) \leq a$
$f(b) = b_1$	$b \leq d \wedge f(b) \not\leq c$	

Example: Reasoning in local theory extensions

$$\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$$

where $\mathcal{K}: \forall x (x \leq g(y) \rightarrow f(x) \leq y)$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

Problem: Check whether $G \models_{\mathcal{T}_1} \perp$ local theory extension

Def	G_0	$\mathcal{K}[G]_0 \wedge \text{Con}[G]_0$
$g(a) = a_1$	$d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$ (unsatisfiable)
$f(b) = b_1$	$b \leq d \wedge b_1 \not\leq c$	

Overview

- Local theory extensions
- **Computing interpolants in local theory extensions**
- **Applications**
- **Conclusions, perspectives**

Our goal

Assume $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is **local**, and $A \wedge B \models_{\mathcal{T}_0 \cup \mathcal{K}} \perp$.

Then $A_0 \wedge B_0 \wedge \underbrace{\mathcal{K}[A \wedge B]_0 \wedge \text{Con}[A, B]_0}_{\mathcal{H}_A \wedge \mathcal{H}_B \wedge \mathcal{H}_{\text{mix}}} \models_{\mathcal{T}_0} \perp$

- (1) Separate the clauses in $\mathcal{K}[A \wedge B]_0 \wedge \text{Con}[A, B]_0$
s.t. $A_0 \wedge \mathcal{K}[A \wedge B]_0^A \wedge \text{Con}_0^A \wedge B_0 \wedge \mathcal{K}[A \wedge B]_0^B \wedge \text{Con}_0^B \models_{\mathcal{T}_0} \perp$
- (2) Compute (in \mathcal{T}_0) an interpolant I_0 for the formula above
- (3) From I_0 reconstruct an interpolant I for $A \wedge B$.

Interpolation in theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$

Assumptions

1. \mathcal{T}_0 convex $\Gamma \models_{\mathcal{T}_0} \bigvee R_i(t_i) \Rightarrow \exists i : \Gamma \models_{\mathcal{T}_0} R_i(t_i)$

2. \mathcal{T}_0 P -interpolating $A \wedge B \models a R b \Rightarrow \exists t_{AB} \begin{cases} A \wedge B \models a R t_{AB} \\ A \wedge B \models t_{AB} R b \end{cases}$.

3. \mathcal{T}_0 has ground interpolation

4. \mathcal{K} either has only one function occurrence/clause,
or consists of pairs of rules of the form:

$$\begin{cases} \bigwedge x_i R_i s_i \rightarrow f(x_1, \dots, x_n) R g(y_1, \dots, y_n) \\ \bigwedge x_i R_i y_i \rightarrow f(x_1, \dots, x_n) R f(y_1, \dots, y_n) \end{cases}$$

$R_1, \dots, R_n \in P$; R transitive; $s_i \in \{y_1, \dots, y_n\}$ or $s_i = f_i(y_{i_1}, \dots, y_{i_k})$.

5. $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ local extension

Examples

Only one function occurrence/clause:

- **Free functions (+ boundedness)** over pure equality, posets, Bool, DLat, SLat, linear arithmetic (over \mathbb{R}, \mathbb{Q})
- **Lipschitz functions** at a point c : $|f(x) - f(c)| \leq \lambda \cdot |x - c|$

Clauses of the form:

$$\mathcal{K} : \begin{cases} \bigwedge x_i R_i s_i \rightarrow f(x_1, \dots, x_n) R g(y_1, \dots, y_n) \\ \bigwedge x_i R_i y_i \rightarrow f(x_1, \dots, x_n) R f(y_1, \dots, y_n) \end{cases}$$

$R_1, \dots, R_n \in P$; R transitive; $s_i \in \{y_1, \dots, y_n\}$ or $s_i = f_i(y_{i_1}, \dots, y_{i_k})$.

- **Monotone functions** over posets, Bool, DLat, SLat
- **Semi-Galois connections** (monotone functions): $x \leq g(y) \rightarrow f(x) \leq y$
- **Composition conditions** (monotone functions): $x \leq g(y) \rightarrow f(x) \leq g(y)$

Illustration

Example: $\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$, where \mathcal{K} :

$$\forall x (x \leq g(y) \rightarrow f(x) \leq y)$$

$$\forall x (x \leq y \rightarrow g(x) \leq g(y))$$

$$\forall x (x \leq y \rightarrow f(x) \leq f(y))$$

local extension

Def	$A_0 \wedge B_0$	$\mathcal{K}[A, B]_0 \wedge \text{Con}[A, B]_0$
$g(a)=a_1$	$A_0 : d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$ (unsatisfiable)
$f(b)=b_1$	$B_0 : b \leq d \wedge b_1 \not\leq c$	

Illustration

Example: $\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$

$$A \wedge B \models_{\mathcal{T}_1} \perp$$

Def	$A_0 \wedge B_0$	$\mathcal{K}[A, B]_0 \wedge \text{Con}[A, B]_0$
$g(a)=a_1$	$A_0 : d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$
$f(b)=b_1$	$B_0 : b \leq d \wedge b_1 \not\leq c$	

$A_0 \wedge B_0 \models b \leq a_1$
$B_0 \models b \leq d$
$A_0 \models d \leq a_1$

Illustration

Example: $\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$

$$A \wedge B \models_{\mathcal{T}_1} \perp$$

Def	$A_0 \wedge B_0$	$\mathcal{K}[A, B]_0 \wedge \text{Con}[A, B]_0$
$g(a)=a_1$	$A_0 : d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$
$f(b)=b_1$	$B_0 : b \leq d \wedge b_1 \not\leq c$	

$A_0 \wedge B_0 \models b \leq a_1$
$B_0 \models b \leq d$
$A_0 \models d \leq a_1$

Consider new instances of \mathcal{K}

$$b \leq d \rightarrow f(b) \leq f(d)$$

$$d \leq g(a) \rightarrow f(d) \leq a$$

Illustration

Example: $\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$

$$A \wedge B \models_{\mathcal{T}_1} \perp$$

Def	$A_0 \wedge B_0$	$\mathcal{K}[A, B]_0 \wedge \text{Con}[A, B]_0$
$g(a) = a_1$	$A_0 : d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$
$f(b) = b_1$	$B_0 : b \leq d \wedge b_1 \not\leq c$	$b \leq d \rightarrow b_1 \leq d_1$
$f(d) = d_1$		$d \leq a_1 \rightarrow d_1 \leq a$

$A_0 \wedge B_0 \models b \leq a_1$
$B_0 \models b \leq d$
$A_0 \models d \leq a_1$

Consider new instances of \mathcal{K}

$$b \leq d \rightarrow f(b) \leq f(d)$$

$$d \leq g(a) \rightarrow f(d) \leq a$$

Illustration

Example: $\mathcal{T}_1 = \text{SLat} \cup \mathcal{K}$

$$A \wedge B \models_{\mathcal{T}_1} \perp$$

Def	$A_0 \wedge B_0$	$(\mathcal{K}[A, B]_0 \wedge \text{Con}[A, B]_0)_{\text{sep}}$
$g(a) = a_1$	$A_0 : d \leq a_1 \wedge a \leq c$	$b \leq a_1 \rightarrow b_1 \leq a$
$f(b) = b_1$	$B_0 : b \leq d \wedge b_1 \not\leq c$	$b \leq d \rightarrow b_1 \leq d_1$
$f(d) = d_1$		$d \leq a_1 \rightarrow d_1 \leq a$

Interpolant (w.r.t. SLat): $I_0 = d_1 \leq c$



Interpolant (w.r.t. $\text{SLat} \cup \mathcal{K}$) of $A \wedge B$: $I = f(d) \leq c$

Applications

1. Verification

- Useful for abstraction refinement, widening, invariant generation
- Hierarchical method \mapsto allows to control form of interpolant
- Constrained interpolation in LI + UIF (+ boundedness) and implementation (ARMC, Blast) [Rybalchenko, VS, submitted]

2. Reasoning in combinations of local extensions of a base theory

\mathcal{T}_0 has **ground interpolation** $\mathcal{T}_0 \subseteq \mathcal{T}_i = \mathcal{T}_0 \cup \mathcal{K}_i$ **local**, $i = 1, 2$

$\mathcal{T}_0 \subseteq \mathcal{T}_1 \cup \mathcal{T}_2$ **local** G_i ground \mathcal{T}_i formula.

If $G_1 \wedge G_2 \models_{\mathcal{T}_1 \cup \mathcal{T}_2} \perp$ then there exists a ground formula I containing only \mathcal{T}_0 -functions and constants shared by G_1, G_2 s.t. $G_1 \models I$ and $I \wedge G_2 \models \perp$

Applications

3. Distributed databases

Chem	
Primitive concepts (C_0):	process, reaction, subst, organic, anorganic
Constraints (Γ_0):	$organic \sqcap anorganic = \emptyset$ $organic \sqsubseteq subst, \quad anorganic \sqsubseteq subst$

AnorgChem	
C_1 :	cat-oxydation, oxydation R_1 : catalyses
T_1 :	cat-oxydation = subst \sqcap \exists catalyses.oxydation
Γ_1 :	<div style="border: 1px solid black; padding: 5px; display: inline-block;"> reaction \sqsubseteq oxydation </div> $\emptyset \neq cat-oxydation \sqsubseteq anorganic$

BioChem	
C_2 :	enzyme R_2 : produces, catalyses
T_2 :	reaction = process \sqcap \exists produces.subst enzyme = organic \sqcap \exists catalyses.reaction
Γ_2 :	enzyme $\neq \emptyset$

T.f.a.e: (1) **Chem** + **AnorgChem** + **BioChem** inconsistent

(2) $\Gamma_0 \wedge (T_1 \wedge \Gamma_1) \wedge (T_2 \wedge \Gamma_2) \models_{SLatUMon} \perp$

Interpolant substance \sqcap \exists catalyses.reaction \sqsubseteq anorganic

Conclusions

Method for computing interpolants in local extensions of a base theory with a set of functions satisfying a set \mathcal{K} of clauses

- Interpolants are computed in a hierarchical way
- The method is general
 - applications to theories more general than LI + UIF but method different from McMillan's method [McMillan'04]
 - orthogonal to other methods e.g. [Yorsh, Musuvathi'05])
- We identified classes of theory extensions for which this is possible
- We discussed several application domains
 - large potential of application: can use black-box for base theory
 - current/ongoing work on constrained interpolation, implementations (ARMC/ Blast) [Rybalchenko, VS, submitted]