

Hierarchical and modular reasoning in complex theories: The case of local theory extensions

Viorica Sofronie-Stokkermans

Max-Planck-Institut für Informatik

Saarbrücken

Motivation

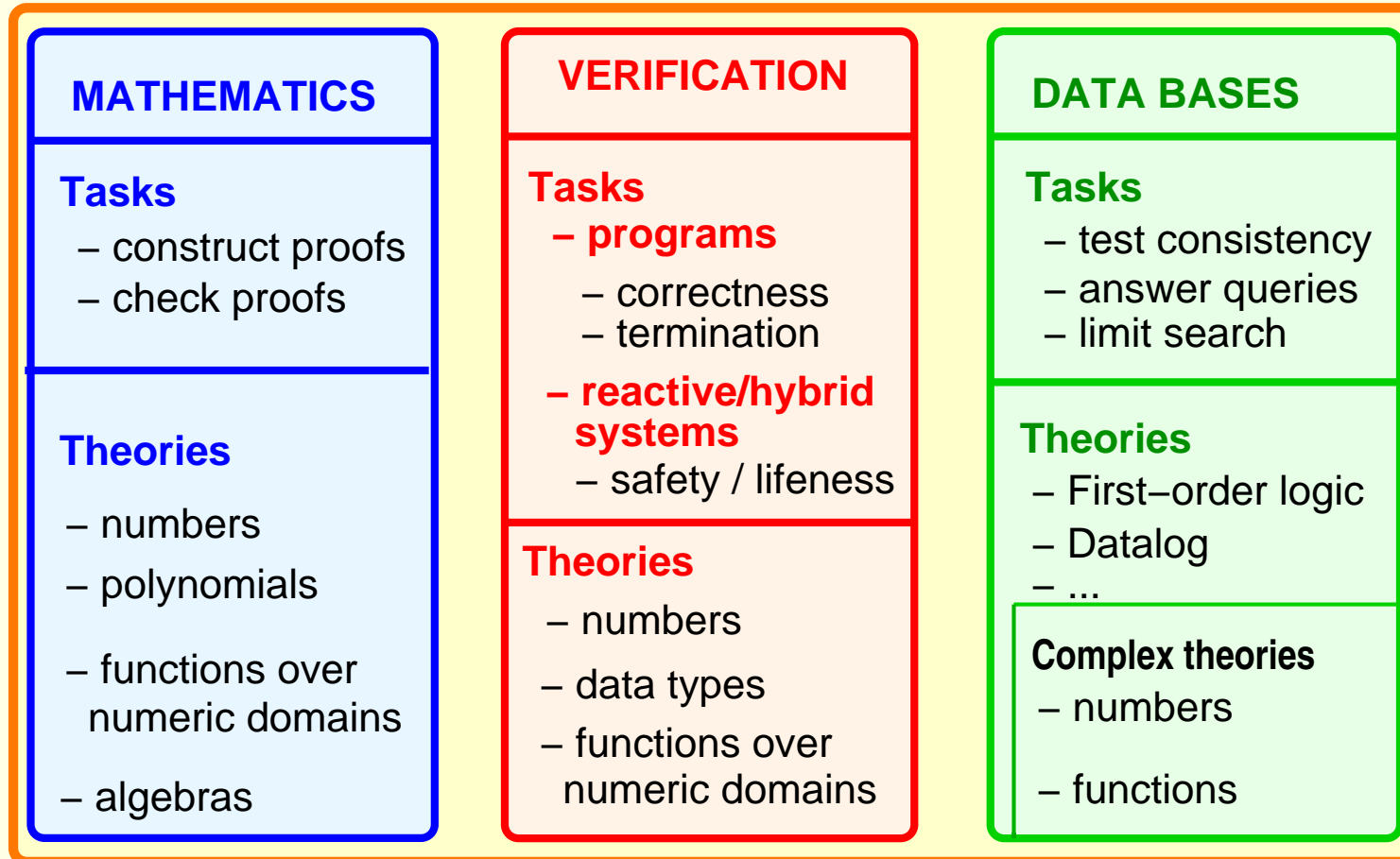
Long-term goal of research in computer science

- use computers as 'intelligent assistants' in
e.g. mathematics, engineering (and other fields)

Main problem

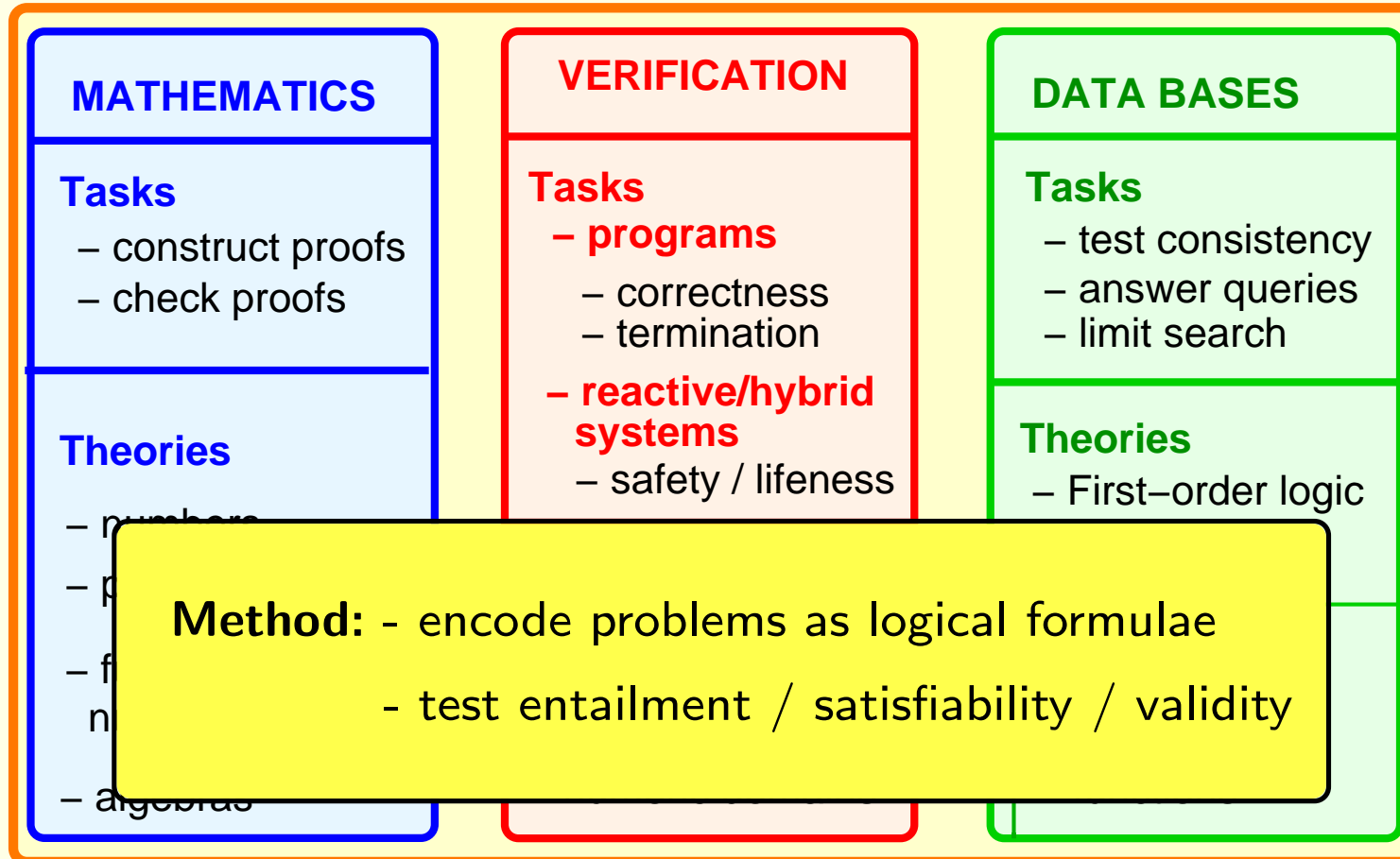
- complex description of problems to be solved
↳ complex systems, complex encoding

Examples of application domains



complex systems (MAS, reactive systems w. embedded software, databases)

Examples of application domains



complex systems (MAS, reactive systems w. embedded software, databases)

Problems and goals

- 1st order logic is undecidable: cannot build an 'all-purpose' program
- + often fragments of theories occurring in applications are decidable

- theories do not occur alone: need to consider combinations of theories
- + often provers for the component theories can be combined efficiently

Important:

Identify theories (and extensions/combinations thereof) which are decidable (with low complexity) and relevant in applications

Problems

$\models \phi$ ϕ : arbitrary

$\neg\phi \models \perp$ ϕ : arbitrary

Problems

$\models_{\mathcal{T}} \phi$	ϕ : arbitrary	$\models_{\mathcal{T}} \forall x \psi(x)$	$\psi(x)$: quantifier-free
$\neg \phi \models_{\mathcal{T}} \perp$	ϕ : arbitrary	$\neg \psi(\bar{c}) \models_{\mathcal{T}} \perp$	$\neg \psi(\bar{c})$: ground

\mathcal{T} can be:

- a standard theory,
- extension of standard theory (with function symbols + axioms)
- a combination of theories

Examples: Mathematics, Logic, Databases, Verification

e.g. **conjunction:** $\phi(1) \wedge Tr(1, 2) \wedge \dots \wedge Tr(n-1, n) \wedge \neg \text{safe}(n)$

- **satisfiable:** path to unsafe state

- **unsatisfiable:** no path to unsafe state

Problems

$\models_{\mathcal{T}} \phi$ ϕ arbitrary	$\models_{\mathcal{T}} \forall x \psi(x)$ $\psi(x)$ quantifier-free
$\neg \phi \models_{\mathcal{T}} \perp$ ϕ arbitrary	$\neg \psi(\bar{c}) \models_{\mathcal{T}} \perp$ $\neg \psi(\bar{c})$ ground

\mathcal{T} can be:

- a standard theory,
- extension of standard theory (with function symbols + axioms)
- a combination of theories

State of the art

- satisfiability checking for **ground formulae** w.r.t. a theory (SMT)
 - $\mapsto DPLL(\mathcal{T})$
- **arbitrary problems**: instances \mapsto ground satisfiability (not complete)

Efficient reasoning

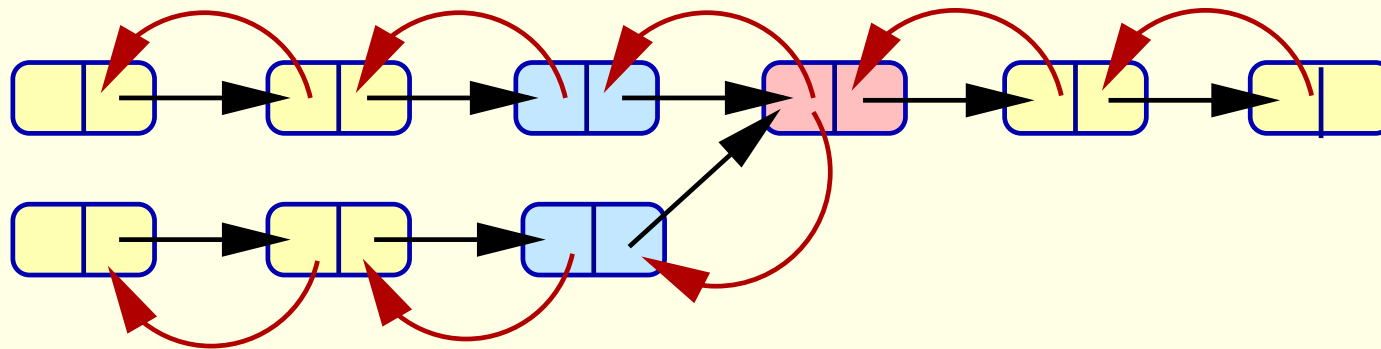
- Possibility of limiting search
- Modular reasoning in complex theories

Efficient reasoning

- Possibility of limiting search
without loss of completeness
- Modular reasoning in complex theories

Limiting search: Example

Reasoning about doubly-linked lists cf. also [Necula, McPeak 2005]



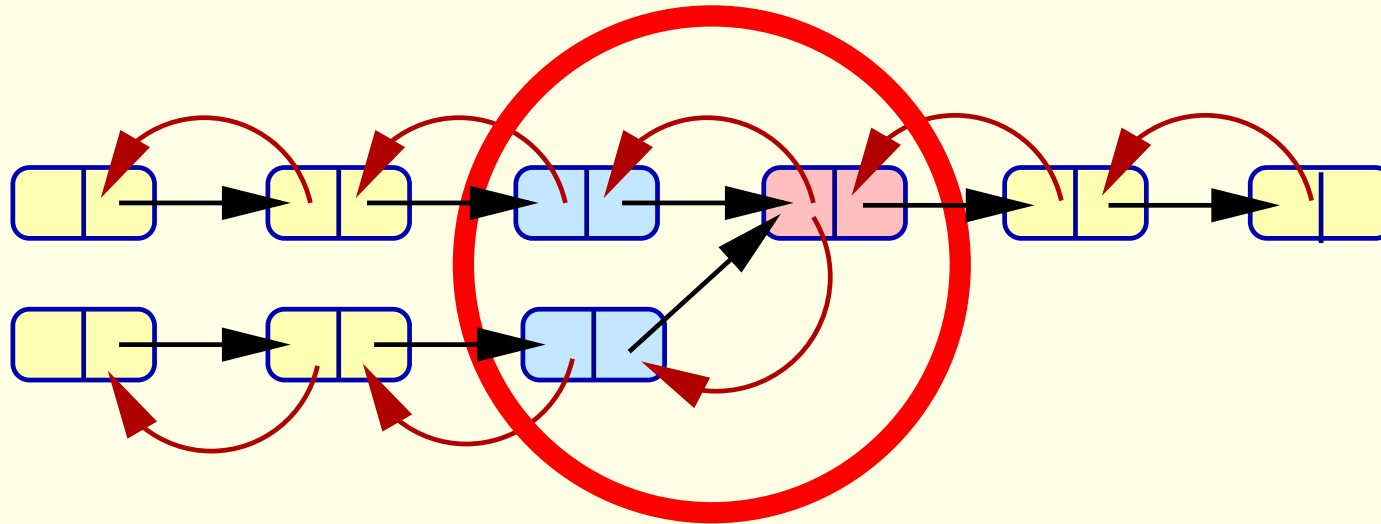
$$\forall p (p \neq \text{null} \wedge p.\text{next} \neq \text{null} \rightarrow p.\text{next}.\text{prev} = p)$$

$$\forall p (p \neq \text{null} \wedge p.\text{prev} \neq \text{null} \rightarrow p.\text{prev}.\text{next} = p)$$

$$\wedge c \neq \text{null} \wedge c.\text{next} \neq \text{null} \wedge d \neq \text{null} \wedge d.\text{next} \neq \text{null} \wedge c.\text{next} = d.\text{next} \wedge c \neq d \quad \models \quad \perp$$

Limiting search: Example

Reasoning about doubly-linked lists cf. also [Necula, McPeak 2005]



$$\begin{aligned} & (c \neq \text{null} \wedge c.\text{next} \neq \text{null} \rightarrow c.\text{next}.\text{prev} = c) \quad (c.\text{next} \neq \text{null} \wedge c.\text{next}.\text{next} \neq \text{null} \rightarrow c.\text{next}.\text{next}.\text{prev} = c.\text{next}) \\ & (d \neq \text{null} \wedge d.\text{next} \neq \text{null} \rightarrow d.\text{next}.\text{prev} = d) \quad (d.\text{next} \neq \text{null} \wedge d.\text{next}.\text{next} \neq \text{null} \rightarrow d.\text{next}.\text{next}.\text{prev} = d.\text{next}) \\ & \wedge c \neq \text{null} \wedge c.\text{next} \neq \text{null} \wedge d \neq \text{null} \wedge d.\text{next} \neq \text{null} \wedge c.\text{next} = d.\text{next} \wedge c \neq d \quad \models \quad \perp \end{aligned}$$

Local Theories

$\mathcal{T}_1 := \mathcal{K}$ set of Horn clauses;

\mathcal{K} is **local**, if for ground Horn clauses C , $\mathcal{K} \models C$ iff $\mathcal{K}[C] \models C$

[McAllester, Givan '92, '93] Local theories capture PTIME

Local Theories

$\mathcal{T}_1 := \mathcal{K}$ set of Horn clauses;

\mathcal{K} is **local**, if for ground Horn clauses C , $\mathcal{K} \models C$ iff $\mathcal{K}[C] \models C$

[McAllester, Givan '92, '93] Local theories capture PTIME

Natural extension of ideas which occurred in deductive databases

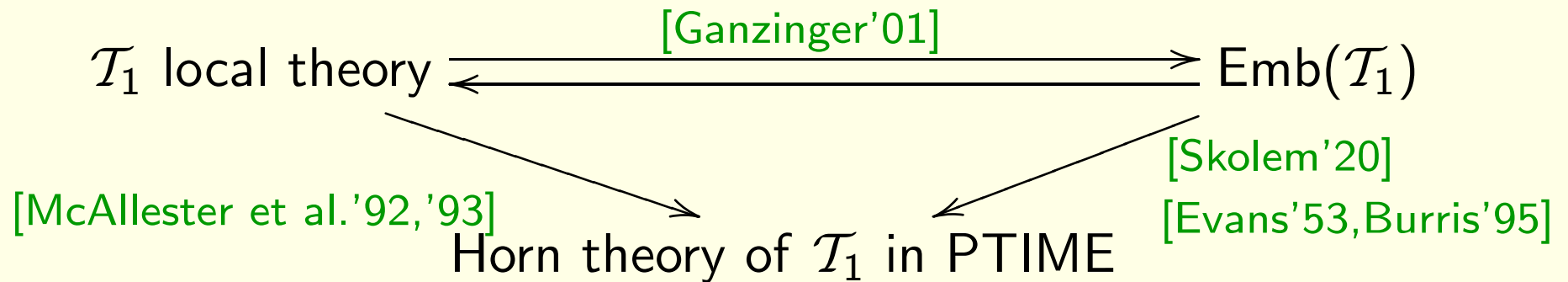
Datalog program:

- set \mathcal{K} of Horn clauses without function symbols
- for ground clauses C : $\mathcal{K} \models C$ iff $\mathcal{K}[C] \models C$
- entailment of ground clauses in PTIME

Local Theories

$\mathcal{T}_1 := \mathcal{K}$ set of Horn clauses;

\mathcal{K} is **local**, if for ground Horn clauses C , $\mathcal{K} \models C$ iff $\mathcal{K}[C] \models C$



How to recognize local theories

- [Ganzinger 2001] test embeddability of partial into total models

Examples:

- theory of lattices [Skolem 1920]
- monotone functions on posets, lattices

- theory of free function symbols

$$- \mathcal{K} : \left\{ \begin{array}{l} s(x) = y \rightarrow p(y) = x \\ p(y) = x \rightarrow s(x) = y \\ s(x) = s(y) \rightarrow x = y \\ p(x) = p(y) \rightarrow x = y \end{array} \right.$$

- theory of doubly linked lists

- theory of lists with cons, car, cdr

How to recognize local theories

- [Ganzinger 2001] test embeddability of partial into total models
- [Basin, Ganzinger '96,'01] test saturation under ordered resolution

Recognize locality [Basin, Ganzinger'96,'01; Ganzinger'01]

= \mapsto EQ (binary predicate) + congruence axioms

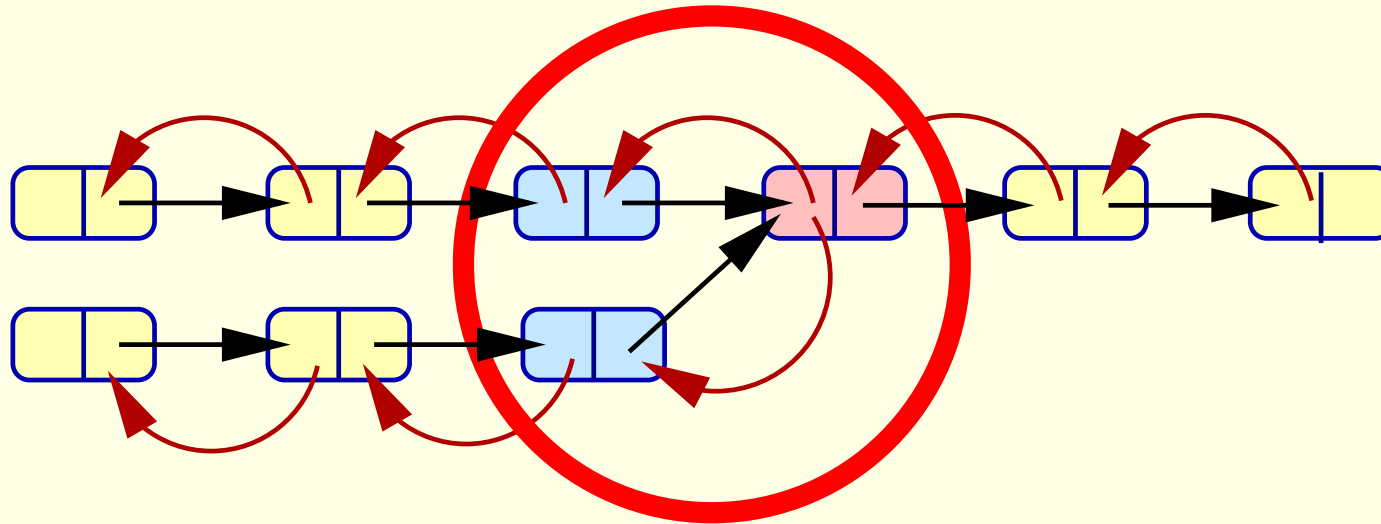
special relations(off) \mapsto saturate

Generate local presentations: add clauses

$$\mathcal{K} : \begin{cases} s(x)=y \rightarrow p(y)=x \\ p(y)=x \rightarrow s(x)=y \\ s(x)=s(y) \rightarrow x=y \\ p(x)=p(y) \rightarrow x=y \end{cases} \quad \mathcal{L} : \begin{cases} \text{cons}(x, y)=z \rightarrow \text{car}(z)=x \\ \text{cons}(x, y)=z \rightarrow \text{cdr}(z)=y \\ \text{cons}(x, y)=\text{cons}(u, v) \rightarrow x=u \wedge y=v \end{cases}$$

Limiting search: Example

Reasoning about doubly-linked lists cf. also [Necula, McPeak 2005]

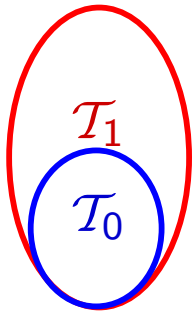


$(c \neq \text{null} \wedge c.\text{next} \neq \text{null})$
 $(d \neq \text{null} \wedge d.\text{next} \neq \text{null})$
 $\wedge c \neq \text{null} \wedge$

Extensions which also consider the **elements** of the list: analogous?
 \mapsto Reasoning in complex theories

$\rightarrow c.\text{next.next.prev} = c.\text{next}$
 $\rightarrow d.\text{next.next.prev} = d.\text{next}$
 $c \neq d \models \perp$

Complex Theories



Hierarchic Reasoning

\mathcal{T}_1 : Σ_1 -theory; $\mathcal{T}_0 \subseteq \mathcal{T}_1$ $\Sigma_0 \subset \Sigma_1$

\mathcal{T}_0 : Σ_0 -theory.

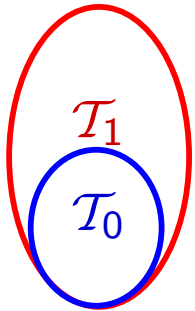
Example:

$f : \mathbb{R} \rightarrow \mathbb{R}$ mon.

\mathbb{R}

Can we use a prover for \mathcal{T}_0 as a blackbox to prove theorems in \mathcal{T}_1 ?

Complex Theories



Hierarchic Reasoning

\mathcal{T}_1 : Σ_1 -theory; $\mathcal{T}_0 \subseteq \mathcal{T}_1$ $\Sigma_0 \subset \Sigma_1$

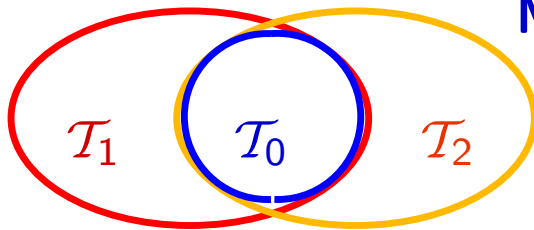
\mathcal{T}_0 : Σ_0 -theory.

Example:

$f : \mathbb{R} \rightarrow \mathbb{R}$ mon.

\mathbb{R}

Can we use a prover for \mathcal{T}_0 as a blackbox to prove theorems in \mathcal{T}_1 ?



Modular Reasoning

\mathcal{T}_0 : Σ_0 -theory.

\mathcal{T}_i : Σ_i -theory; $\mathcal{T}_0 \subseteq \mathcal{T}_i$ $\Sigma_0 \subseteq \Sigma_i$.

Example:

lists(\mathbb{R}) \cup **arrays**(\mathbb{R})

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

Which information needs to be exchanged between the provers?

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

$$\text{Mon}_f \quad \forall x, y (x \leq y \rightarrow f(x) \leq f(y))$$

$$\text{Mon}_g \quad \forall x, y (x \leq y \rightarrow g(x) \leq g(y))$$

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

$$\text{Mon}_f \quad \forall x, y (x \leq y \rightarrow f(x) \leq f(y))$$

$$\text{Mon}_g \quad \forall x, y (x \leq y \rightarrow g(x) \leq g(y))$$

Problems:

- A prover for \mathbb{R} does not know about f, g
- A prover for first-order logic may have problems with the reals
- SMT provers may have problems with the universal quantifiers

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

- Hierarchic reasoning

reduce to the problem of checking a family of constraints over \mathbb{R}

- Modular reasoning

if we separate the information about f and g at the beginning:

no need to combine the information again at a later point

Example

$$(\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g) \wedge \underbrace{c+5 \leq f(c) \wedge c+3 \leq d \wedge f(d) \leq g(e)}_G \rightarrow f(c) \leq f(g(e)) \models \perp$$

Partial models can be made total

- can restrict the search space:

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

without loss of completeness

Hierarchical reasoning is possible

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

1. Refute: $c + 5 \leq f(c) \wedge c + 3 \leq d \wedge f(d) \leq g(e) \wedge f(c) \not\leq f(g(e))$

2. $\text{Mon}_f, \text{Mon}_g, \text{Con}$

$$\begin{array}{lll} c + 5 \leq f(c) & x \leq y \rightarrow f(x) \leq f(y) & x \leq y \rightarrow g(x) \leq g(y) \\ c + 3 \leq d & x = y \rightarrow f(x) = f(y) & x = y \rightarrow g(x) = g(y) \\ f(d) \leq g(e) & & \\ f(c) \not\leq f(g(e)) & & \end{array}$$

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

1. Refute: $c + 5 \leq f(c) \wedge c + 3 \leq d \wedge f(d) \leq g(e) \wedge f(c) \not\leq f(g(e))$

2. Locality

$$\begin{array}{lll} c + 5 \leq f(c) & c \bowtie d \rightarrow f(c) \bowtie f(d) & x \leq y \rightarrow g(x) \leq g(y) \\ c + 3 \leq d & g(e) \bowtie c \rightarrow f(g(e)) \bowtie f(c) & x = y \rightarrow g(x) = g(y) \\ f(d) \leq g(e) & d \bowtie g(e) \rightarrow f(d) \bowtie f(g(e)) & \\ f(c) \not\leq f(g(e)) & & \bowtie \in \{\leq, \geq, =\} \end{array}$$

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

1. Refute: $c + 5 \leq f(c) \wedge c + 3 \leq d \wedge f(d) \leq g(e) \wedge f(c) \not\leq f(g(e))$

2. Locality; Separate symbols (definitions)

$f(c) = c_1$	$c + 5 \leq f(c)$	$c \bowtie d \rightarrow f(c) \bowtie f(d)$	$x \leq y \rightarrow g(x) \leq g(y)$
$f(d) = d_1$	$c + 3 \leq d$	$e_1 \bowtie c \rightarrow f(e_1) \bowtie f(c)$	$x = y \rightarrow g(x) = g(y)$
$g(e) = e_1$	$f(d) \leq g(e)$	$d \bowtie e_1 \rightarrow f(d) \bowtie f(e_1)$	
$f(e_1) = f_1$	$f(c) \not\leq f(g(e))$		$\bowtie \in \{\leq, \geq, =\}$

Example

$$\mathbb{R} \cup \text{Mon}_f \cup \text{Mon}_g \models \forall x, y, z (x+5 \leq f(x) \wedge x+3 \leq y \wedge f(y) \leq g(z) \rightarrow f(x) \leq f(g(z)))$$

1. Refute: $c + 5 \leq f(c) \wedge c + 3 \leq d \wedge f(d) \leq g(e) \wedge f(c) \not\leq f(g(e))$

2. Locality; Separate symbols (definitions); Satisfiability test in \mathbb{R}

$f(c) = c_1$	$c + 5 \leq c_1$	$c \bowtie d \rightarrow c_1 \bowtie d_1$	$x \leq y \rightarrow g(x) \leq g(y)$
$f(d) = d_1$	$c + 3 \leq d$	$e_1 \bowtie c \rightarrow f_1 \bowtie c_1$	$x = y \rightarrow g(x) = g(y)$
$g(e) = e_1$	$d_1 \leq e_1$	$d \bowtie e_1 \rightarrow d_1 \bowtie f_1$	
$f(e_1) = f_1$	$c_1 \not\leq f_1$		$\bowtie \in \{\leq, \geq, =\}$

Idea

Σ_1 extension of Σ_0 with **function symbols** \mathcal{K} set of Σ_1 -clauses;

Task: Check whether $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$

Abstraction: Consider only certain instances

Approximate extension functions with “partial” functions

$$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp \quad \mapsto \quad \mathcal{T}_0 \cup \mathcal{K}[G] \cup \text{Con}[G] \cup G \models \perp$$

Soundness: $\mathcal{T}_0 \cup \mathcal{K}[G] \cup \text{Con}[G] \cup G \models \perp \implies \mathcal{T}_0 \cup \mathcal{K} \cup G \models \perp$

Completeness: $\mathcal{T}_0 \cup \mathcal{K}[G] \cup \text{Con}[G] \cup G \not\models \perp \implies \exists$ “partial” model

If every “partial” model can be embedded into
a total model of $\mathcal{T}_0 \cup \mathcal{K} \cup G$

then $\mathcal{T}_0 \cup \mathcal{K} \cup G \not\models \perp$

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

Reasoning in local theory extensions

Limit search:

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

1. Use *DPLL*($\mathcal{T}_0 + UIF$) (or another SMT method) if $\mathcal{K}[G]$ ground

2. Hierarchical reasoning [VS 2005] $|G| = n$

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

\mapsto definitions for terms starting with extension functions

– reduce to satisfiability in \mathcal{T}_0 $\Rightarrow \mathcal{K}_0 \wedge G_0 \wedge \text{Con}[G]_0$ $+n^2$

Parameterized complexity:

$g(n^k)$

How to recognize local theory extensions

- Test embeddability of partial models into total models

[Ganzinger, VS, Waldmann'04, VS'05]
[VS, Ihlemann'07]

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

Theorem \mathcal{K} set of Σ_1 -flat, Σ_1 -linear clauses. Then for $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$:

(1) $(\text{Emb}_w) \Rightarrow (\text{Loc})$ $(\text{Emb}_w^{\text{fd}}) \Rightarrow (\text{Loc}^{\text{f}})$.

(2) \mathcal{T}_0 locally finite, universal; \mathcal{K} finitely many ground terms: $(\text{Emb}_w^{\text{f}}) \Rightarrow (\text{Loc}^{\text{f}})$.

(Emb_w) Every weak p.model of \mathcal{T}_1 weakly embeds into a total model of \mathcal{T}_1 .

$(\text{Emb}_w^{\text{f}})$ finite weak partial model

$(\text{Emb}_w^{\text{fd}})$ weak partial model where Σ_1 -functions have finite domain

How to recognize local theory extensions

- Test embeddability of partial models into total models

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

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

- Saturation under ordered resolution

Theorem $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ - all extension functions have new output sort
- \mathcal{K} contains only extension symbols and local

Then $\mathcal{T}_0 \subseteq \mathcal{T}_1$ local.

Saturation under ordered resolution:

test locality of \mathcal{K} or generate local presentation if it exists.

How to recognize local theory extensions

- Test embeddability of partial models into total models

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

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

- Saturation under ordered resolution

Theorem $\mathcal{T}_0 \subseteq \mathcal{T}_1$

Then \mathcal{T}_1

Examples

- Extensions with functions +
- definitions, boundedness, monotonicity
- Theories of data structures

sort
local

Saturation under ordered resolution:

test locality of \mathcal{K} or generate local presentation if it exists.

Examples of local extensions

Extensions of an arbitrary theory with free function symbols

Extensions of a theory \mathcal{T}_0 with (strictly) monotone functions if:

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

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

subject to additional constraints e.g.:

- **boundedness:** $f(x_1, \dots, x_n) \leq t(x_1, \dots, x_n)$
 t \mathcal{T}_0 -term, t same monotonicity as f in x_1, \dots, x_n
- **semi-Galois conditions:** $x \leq g(y) \rightarrow f(x) \leq y$

Examples of local extensions

Extensions of an arbitrary theory with free function symbols

Extensions of a theory \mathcal{T}_0 with (strictly) monotone functions if:

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

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

subject to additional constraints

- **boundedness:** $f(x_1, \dots, x_r)$

t \mathcal{T}_0 -term, t same monoid

- **semi-Galois conditions:** x

Applications

• Verification:

sorted arrays \mapsto train positions

[Jacobs, VS, PDPAR'06]

controllers $\text{in}(\text{out}(L)) \leq L$ [VS'06]

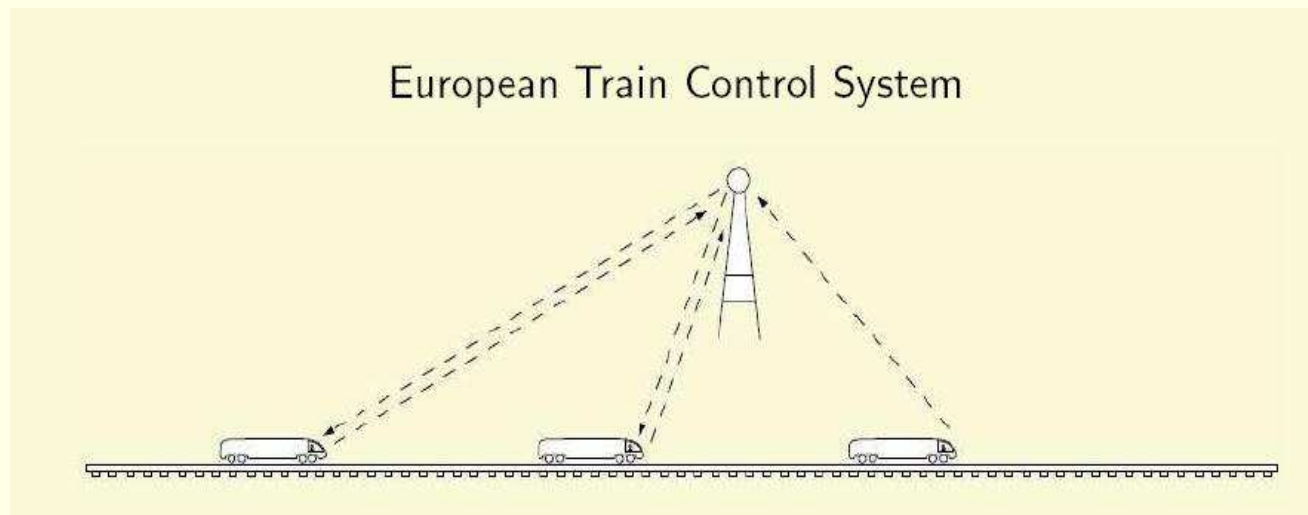
$0 \leq \text{out}(L), \text{in}(L) \leq L$

• Knowledge representation:

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

Application 1: ETCS Case Study

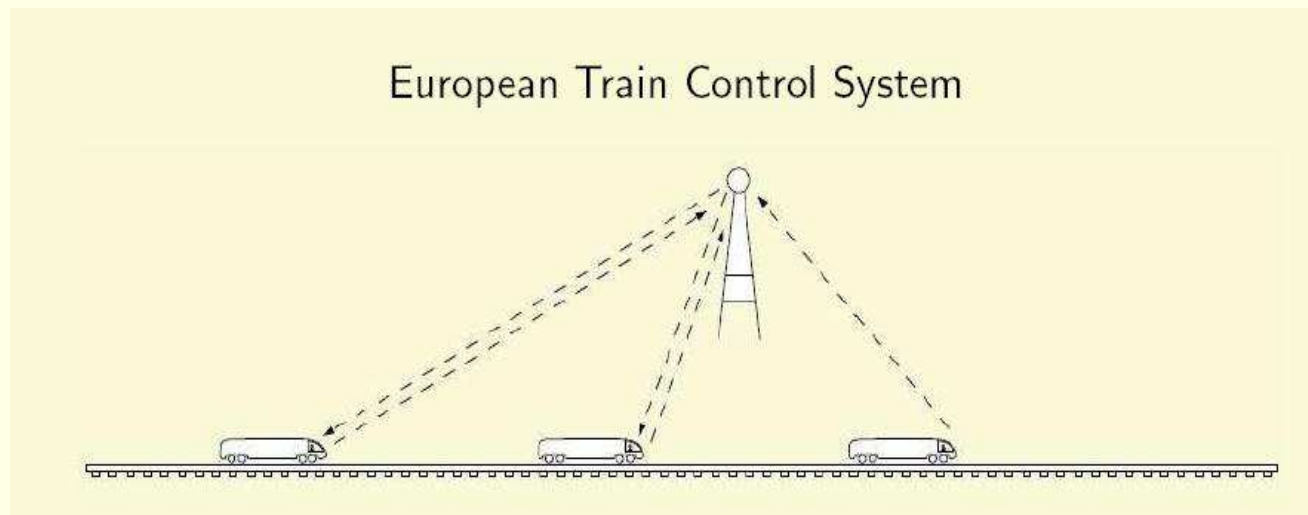
Simplified version of ETCS Case Study [Jacobs,VS'06, Faber,Jacobs,VS'07]



- Number** of trains: $n \geq 0$ \mathbb{Z}
- Minimum and maximum **speed** of trains: $0 \leq \min < \max$ \mathbb{R}
- Minimum **secure distance**: $l_{\text{alarm}} > 0$ \mathbb{R}
- Time** between updates: $\Delta t > 0$ \mathbb{R}
- Train positions** before and after update: $pos(i), pos'(i) : \mathbb{Z} \rightarrow \mathbb{R}$

Application 1: ETCS Case Study

Simplified version of ETCS Case Study [Jacobs,VS'06, Faber,Jacobs,VS'07]



Safe(pos) : $\forall i, j (i < j \rightarrow \text{pos}(i) > \text{pos}(j))$

Update(pos, pos') :

- $\forall i (i = 0 \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$
- $\forall i (0 < i < n \wedge \text{pos}(i - 1) > 0 \wedge \text{pos}(i - 1) - \text{pos}(i) \geq l_{\text{alarm}} \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$

...

Application 1: ETCS Case Study

Safe(pos) : $\forall i, j (i < j \rightarrow \text{pos}(i) > \text{pos}(j))$

Update(pos, pos') : $\forall i (i = 0 \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$
 $\forall i (0 < i < n \wedge \text{pos}(i - 1) > 0 \wedge \text{pos}(i - 1) - \text{pos}(i) \geq l_{\text{alarm}} \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$

...

Inductive invariant: No collisions

$\text{Safe}(\text{pos}) \wedge \text{Update}(\text{pos}, \text{pos}') \models \text{Safe}(\text{pos}')$ iff.

$\text{Safe}(\text{pos}) \wedge \text{Update}(\text{pos}, \text{pos}') \wedge \neg \text{Safe}(\text{pos}') \models \perp$

Problem: Quantified formulae in combinations of theories

\mapsto Instantiation-based methods: incomplete or non-terminating

Tests: Yices \mapsto often answers “unknown” for satisfiable formulae

Application 1: ETCS Case Study

Safe(pos) : $\forall i, j (i < j \rightarrow \text{pos}(i) > \text{pos}(j))$

Update(pos, pos') : $\forall i (i = 0 \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$
 $\forall i (0 < i < n \wedge \text{pos}(i - 1) > 0 \wedge \text{pos}(i - 1) - \text{pos}(i) \geq l_{\text{alarm}} \rightarrow \text{pos}(i) + \Delta t * \text{min} \leq \text{pos}'(i) \leq \text{pos}(i) + \Delta t * \text{max})$

...

Inductive invariant: No collisions

$\text{Safe}(\text{pos}) \wedge \text{Update}(\text{pos}, \text{pos}') \models \text{Safe}(\text{pos}')$ iff.

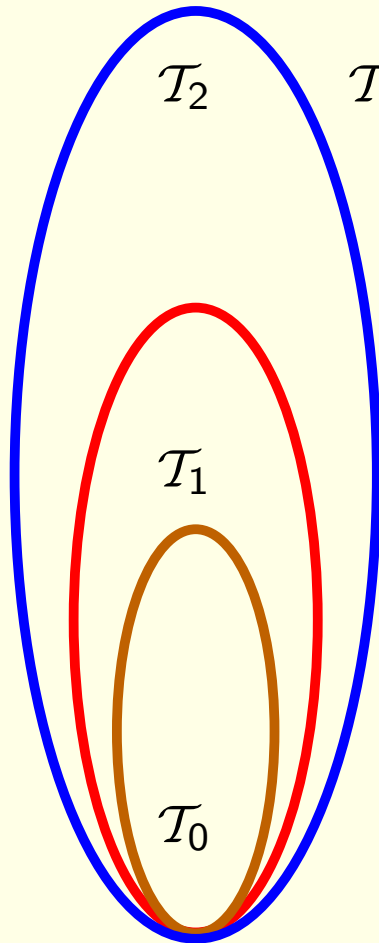
$\text{Safe}(\text{pos}) \wedge \text{Update}(\text{pos}, \text{pos}') \wedge \neg \text{Safe}(\text{pos}') \models \perp$

Problem: Quantified formulae in combinations of theories

↳ Instantiation-based methods: incomplete or non-terminating

Idea: use chains of local theory extensions [Jacobs, VS 2006]

Application 1: ETCS Case Study



$$\mathcal{T}_2 = \mathcal{T}_1 \cup \text{Update}(\text{pos}, \text{pos}')$$

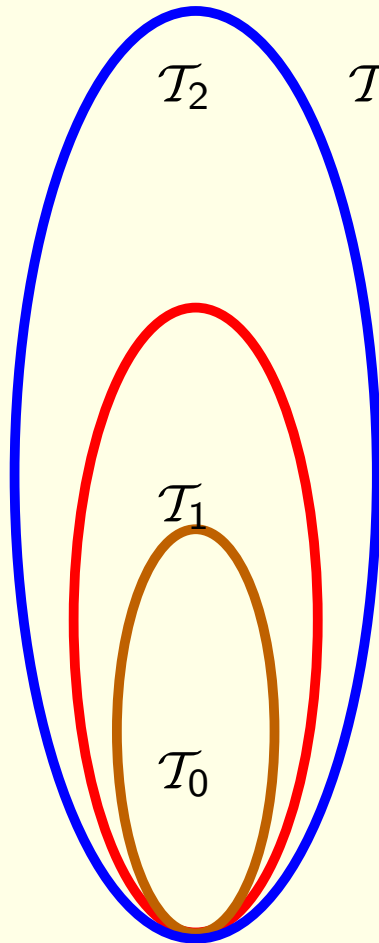
$$\mathcal{T}_1 = \mathcal{T}_0 \cup \text{Safe}(\text{pos})$$

$$\mathcal{T}_0 = \mathbb{R} \cup \mathbb{Z}$$

To show:

$$\mathcal{T}_2 \cup \underbrace{\neg \text{Safe}(\text{pos}')}_G \models \perp$$

Application 1: ETCS Case Study



$$\mathcal{T}_2 = \mathcal{T}_1 \cup \boxed{\text{Update}(\text{pos}, \text{pos}')}$$

Locality:

Hierarchical reasoning:

$$\mathcal{T}_1 = \mathcal{T}_0 \cup \text{Safe}(\text{pos})$$

$$\mathcal{T}_0 = \mathbb{R} \cup \mathbb{Z}$$

To show:

$$\mathcal{T}_2 \cup \underbrace{\neg \text{Safe}(\text{pos}')}_{G} \models \perp$$

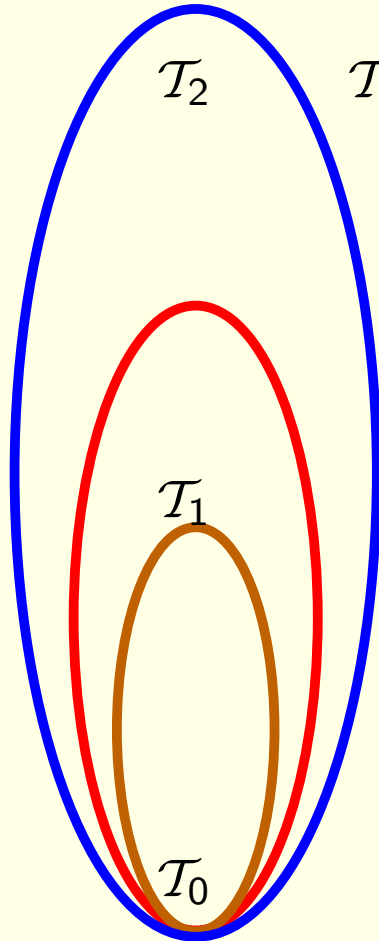
$$\mathcal{T}_1 \cup \boxed{\text{Update}(\text{pos}, \text{pos}')[G]} \cup G \models \perp$$

↓

To show:

$$\mathcal{T}_1 \cup G'(\text{pos}) \models \perp$$

Application 1: ETCS Case Study



$$\mathcal{T}_2 = \mathcal{T}_1 \cup \boxed{\text{Update}(\text{pos}, \text{pos}')}$$

Locality:

Hierarchical reasoning:

$$\mathcal{T}_1 = \mathcal{T}_0 \cup \boxed{\text{Safe}(\text{pos})}$$

Locality:

Hierarchical reasoning:

$$\mathcal{T}_0 = \mathbb{R} \cup \mathbb{Z}$$

To show:

$$\mathcal{T}_2 \cup \underbrace{\neg \text{Safe}(\text{pos}')}_G \models \perp$$

$$\mathcal{T}_1 \cup \boxed{\text{Update}(\text{pos}, \text{pos}') [G]} \cup G \models \perp$$



To show:

$$\mathcal{T}_1 \cup G'(\text{pos}) \models \perp$$

$$\mathcal{T}_0 \cup \boxed{\text{Safe}(\text{pos}) [G']} \cup G' \models \perp$$



To show:

$$\mathcal{T}_0 \cup G'' \models \perp$$

$$\Phi(c, \bar{c}_{\text{pos}'}, \bar{d}_{\text{pos}}, n, l_{\text{alarm}}, \text{min}, \text{max}, \Delta t) \models \perp$$

Standard techniques:	Yices (SAT, CE)	(DPLL(\mathcal{T}_0) + Redlog) \mapsto SAT/QE
----------------------	-----------------	---

Application 1: ETCS Case Study

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%      integer: k, k1, k2, k3, n          %%      real: c1, c2, d1, d2, d3, d4, min, max, lalarm, deltat  %%
%%      c1 = pos'(k)  c2 = pos'(k1)        %%      d1 = pos(k)  d2 = pos(k1)  d3 = pos(k2)  d4 = pos(k3)  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Cond = (min >= 0 and min < max and lalarm > 0 and deltat > 0 and deltat*(max-min) < lalarm);

G = (0 <= k and k<n-1 and c1 <= c2)    and    (k1=k+1 and k2=k-1 and k3=k1-1);

Gf1 = ((k=0 impl ((d1 + deltat*min <= c1) and (c1 <= d1 + deltat*max))) and
      (k1=0 impl ((d2 + deltat*min <= c2) and (c2 <= d2 + deltat*max))));

Gf2 = (((0<k and k<n and d3>0 and d3-d1 >= lalarm) impl ((d1+deltat*min <= c1) and (c1 <= d1+deltat*max))) and
      ((0<k1 and k1<n and d4>0 and d4-d2 >= lalarm) impl ((d2+deltat*min <= c2) and (c2 <= d2+deltat*max))));

Gf3 = (((0<k and k<n and d3>0 and d3-d1 < lalarm) impl (c1 = d1+deltat*min)) and
      ((0<k1 and k1<n and d4>0 and d4-d2 < lalarm) impl (c2 = d2+deltat*min)));

Gf4 = (((0<k and k<n and d3 <= 0) impl c1=d1)    and    ((0<k1 and k1<n and d4 <= 0) impl c2=d2));

Mon = ((k<k1 impl d1>d2) and (k1<k impl d2>d1) and (k1=k impl d2=d1) and
      (k<k2 impl d1>d3) and (k2<k impl d3>d1) and (k=k2 impl d1=d3) and
      (k<k3 impl d1>d4) and (k3<k impl d4>d1) and (k=k3 impl d1=d4) and
      (k1<k2 impl d2>d3) and (k2<k1 impl d3>d2) and (k1=k2 impl d2=d3) and
      (k1<k3 impl d2>d4) and (k3<k1 impl d4>d2) and (k1=k3 impl d2=d4) and
      (k2<k3 impl d3>d4) and (k3<k2 impl d4>d3) and (k2=k3 impl d3=d4) and
      (k=k1 impl c1=c2));
```

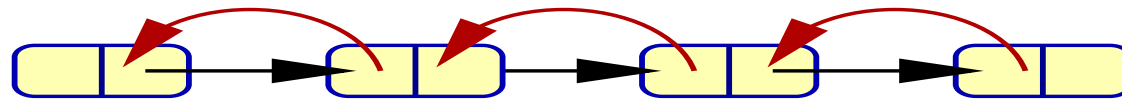
Further examples of local extensions

- **Theories of lists**
- **Theory of arrays** (special type of locality)
 - links to [Bradley,Manna,Sipma] and [Ghilardi,Nicolini,Ranise,Zucchelli'06]
 - ongoing joint work with S.Jacobs and C. Ihlemann
- **Local data structures** [McPeak, Necula 2005]

Examples of local extensions

Local data structures [McPeak, Necula 2005] (stably local)

Example: doubly-linked lists; ordered elements



$$\forall p (p \neq \text{null} \wedge p.\text{next} \neq \text{null} \rightarrow p.\text{next}.\text{prev} = p)$$

$$\forall p (p \neq \text{null} \wedge p.\text{prev} \neq \text{null} \rightarrow p.\text{prev}.\text{next} = p)$$

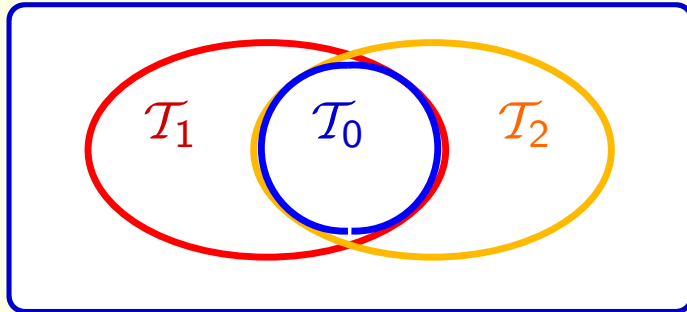
$$\forall p (p \neq \text{null} \wedge p.\text{next} \neq \text{null} \rightarrow p.\text{info} \leq p.\text{next}.\text{info})$$

- two sorts (pointer sort p , scalar sort s).
- pointer fields ($p \rightarrow p$); scalar fields ($p \rightarrow s$);
- axioms: $\forall p \mathcal{E} \wedge \mathcal{C}$; \mathcal{E} contains disjunctions of pointer equalities
 \mathcal{C} contains scalar constraints

Assumption: If $f_1(f_2(\dots f_n(p)))$ occurs in axiom, the axiom also contains:

$$p = \text{null} \vee f_n(p) = \text{null} \vee \dots \vee f_2(\dots f_n(p)) = \text{null}$$

Combinations of Theories



Which information needs to be exchanged between provers for the component theories to guarantee completeness?

Combinations of theories with disjoint signature [Nelson, Oppen'79]

Combinations of theories with non-disjoint signature

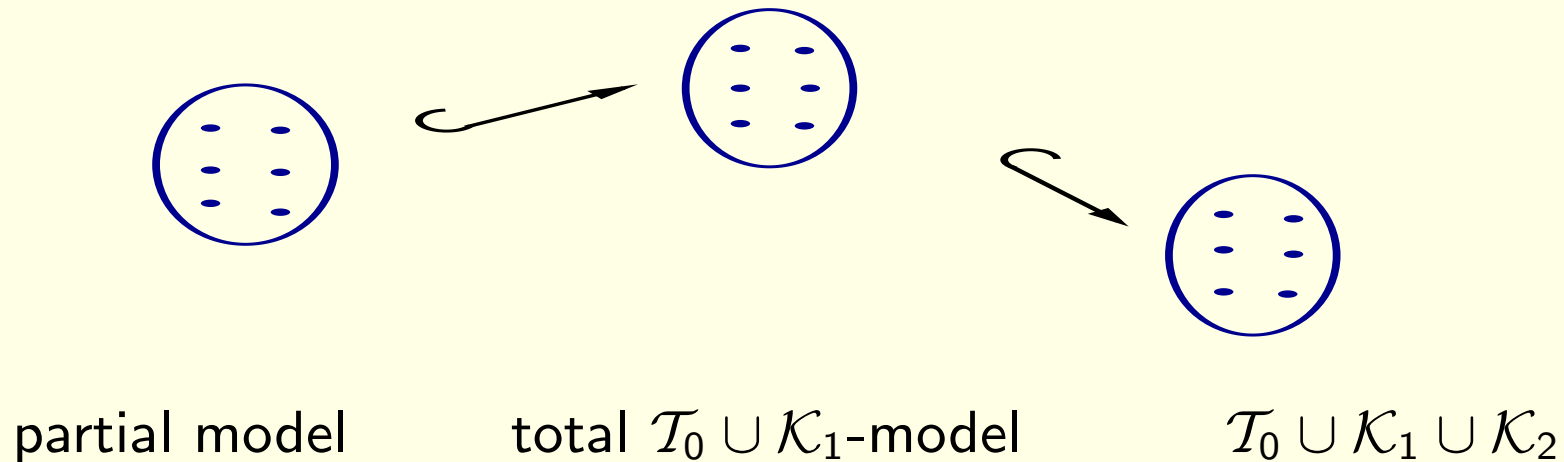
- [Baader, Tinelli'01] [Zarba'02] first results
- [Ghilardi et.al.'03–'05] Model-theoretic method
- [Armando, Bonacina, Ranise, ...] Superposition-based methods
- [Ganzinger, VS, Waldmann'04] Extensions with partial functions
Superposition/pure inferences
- **Here:** Combinations of local theory extensions

Combinations of local extensions

For flat and linear extensions: $\text{Emb}_w \Rightarrow \text{Loc}$ and $\text{Loc} \Rightarrow \text{Emb}_w$

Comp: $P \hookrightarrow A$, same support(s) of base sort(s).

Case 1: $\mathcal{T}_0 \cup \mathcal{K}_i$ satisfy Comp. Then so does $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$.

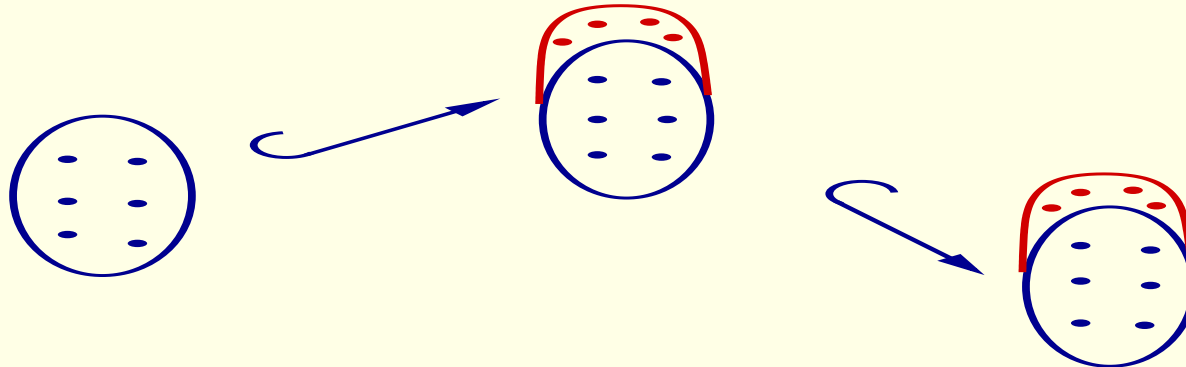


Combinations of local theory extensions

For flat and linear extensions: $\text{Emb}_w \Rightarrow \text{Loc}$ and $\text{Loc} \Rightarrow \text{Emb}_w$

Case 1: $\mathcal{T}_0 \cup \mathcal{K}_i$ satisfy Comp. Then so does $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$.

Case 2: $\mathcal{T}_0 \cup \mathcal{K}_2$ satisfy Comp, $\mathcal{T}_0 \cup \mathcal{K}_1$ satisfy Emb.



partial model

total $\mathcal{T}_0 \cup \mathcal{K}_1$ -model

$\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$

p. model of \mathcal{K}_2 if all vars. in \mathcal{K}_2 below ext. functions

Combinations of local theory extensions

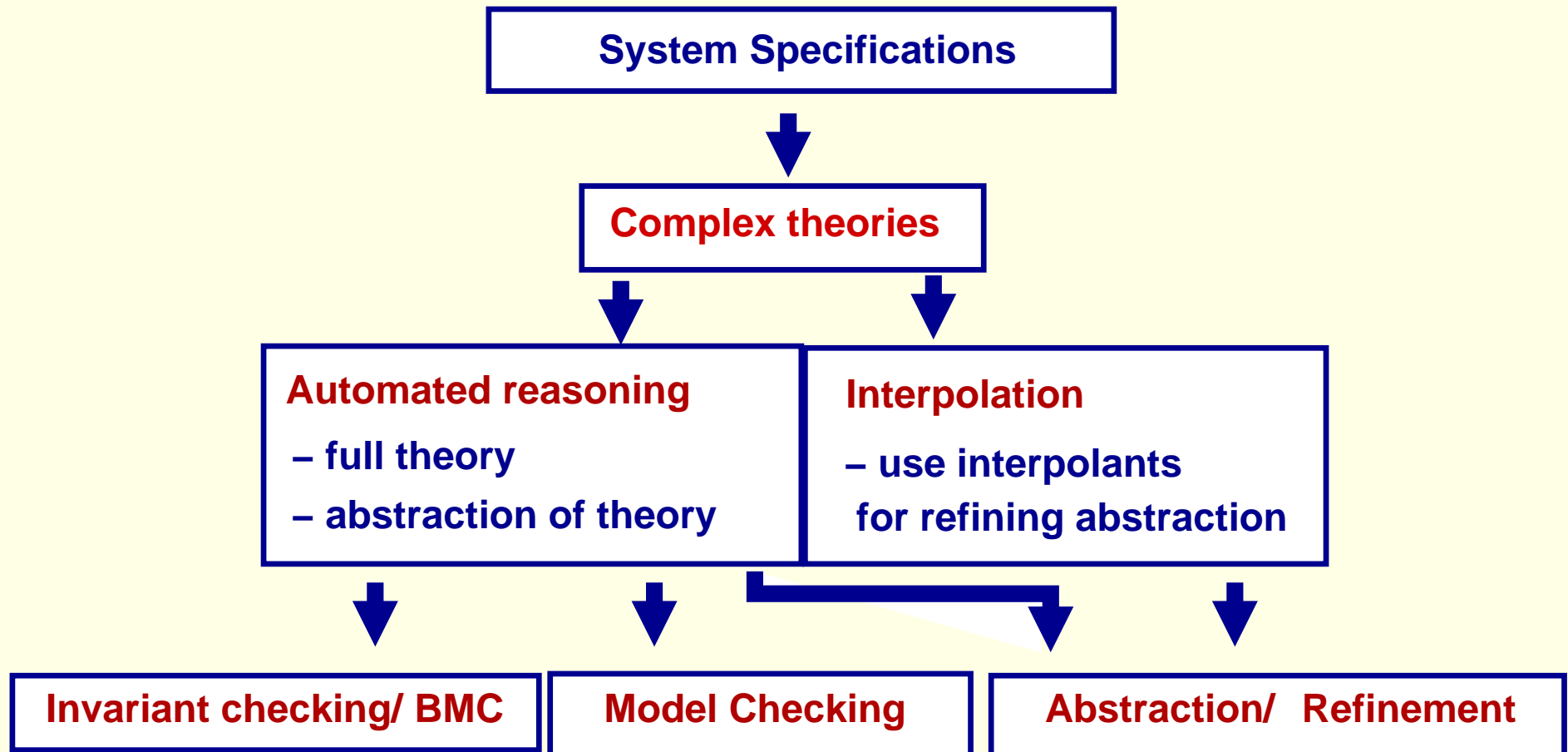
Case 1: $\mathcal{T}_0 \cup \mathcal{K}_i$ satisfy Comp. Then so does $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$.

Case 2: $\mathcal{T}_0 \cup \mathcal{K}_1$ satisfy Comp, $\mathcal{T}_0 \cup \mathcal{K}_1$ satisfy Emb. Then so does $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ if all vars. in \mathcal{K}_2 below ext. functions

Case 3: $\mathcal{T}_0 \cup \mathcal{K}_i$ satisfy Emb. Then so does $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ if all vars. in \mathcal{K}_2 below ext. functions and \mathcal{T}_0 closed under directed limits. (zig-zag argument)

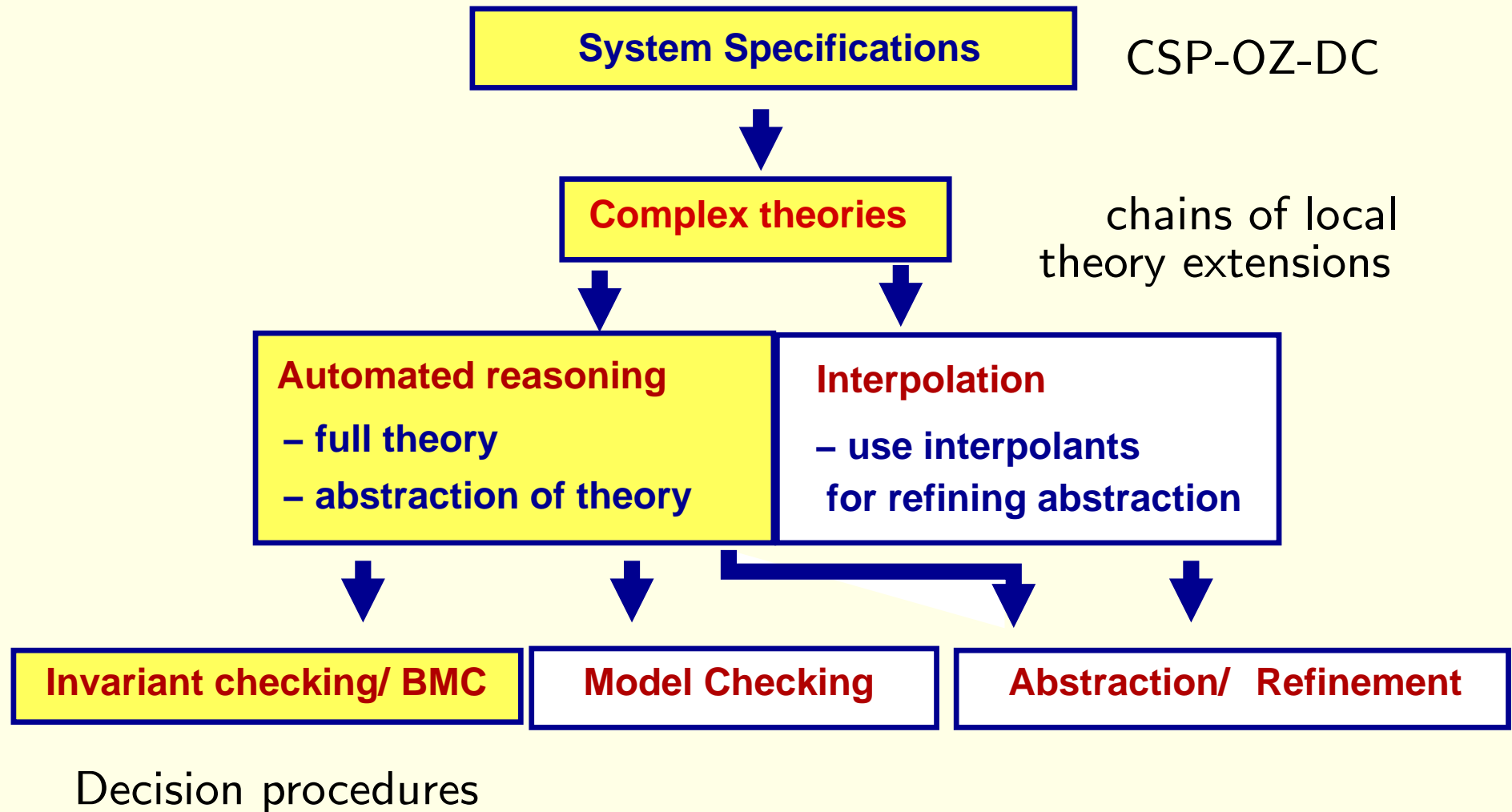
- Examples:**
- Any extension of a theory with $\text{Free}_{\Sigma_1} \cup \text{Mon}_{\Sigma_2}$
 - $\text{Lists}(\mathbb{R}) \cup \text{Mon}(\mathbb{R})$

Verification



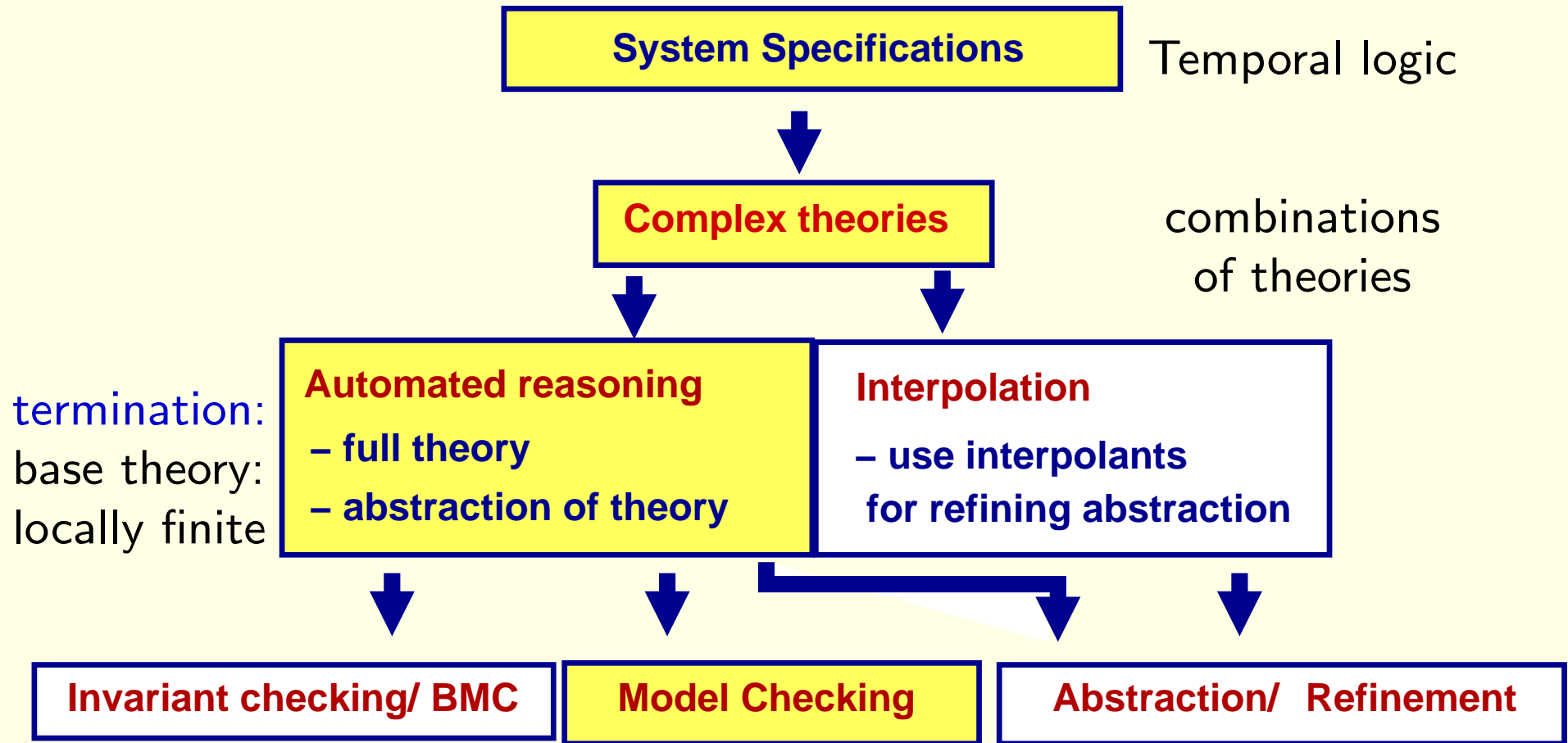
Verification

[Faber, Jacobs, VS '2007]



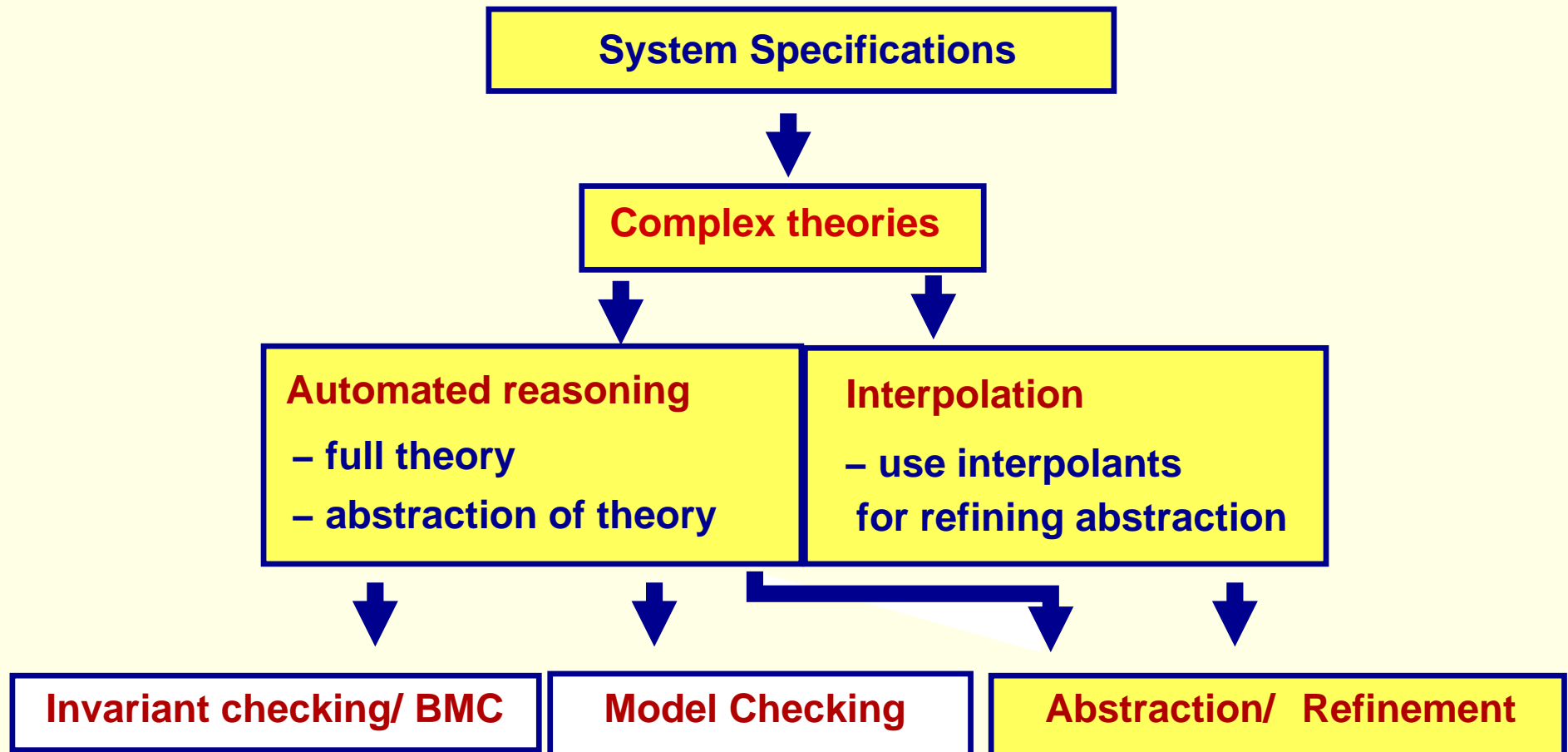
Verification

[Ghilardi, Nicolini, Zucchelli '2007]

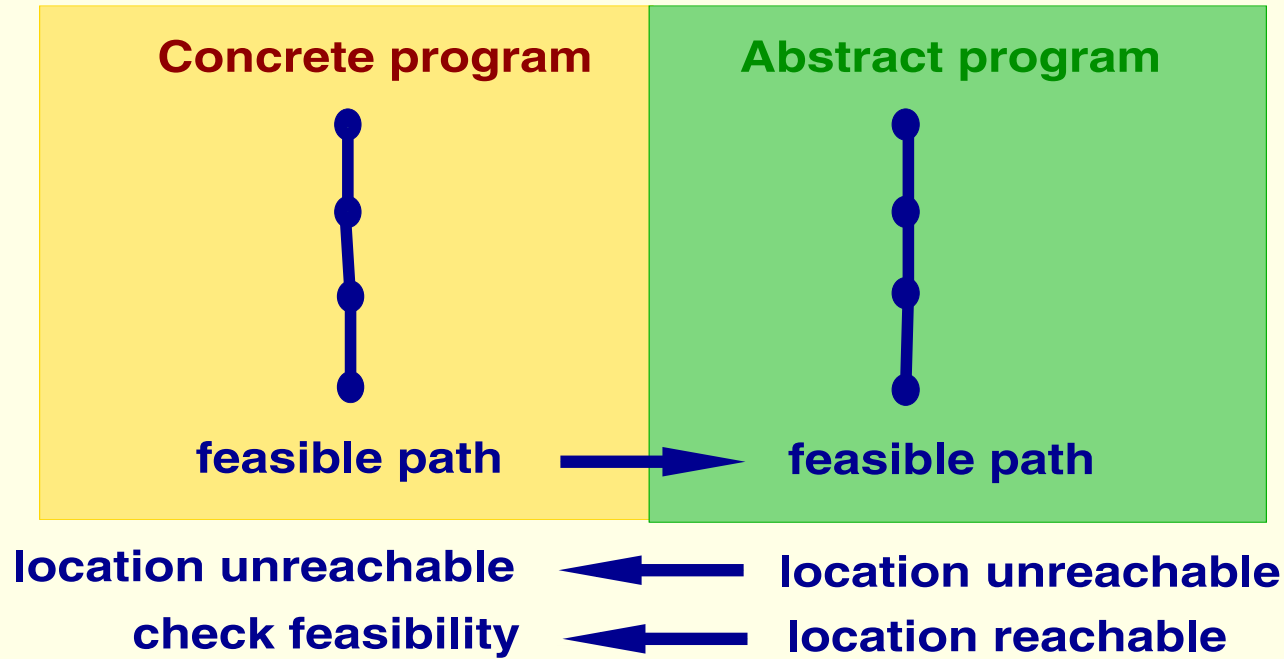


Also hierarchical reasoning applicable

Verification



Abstraction refinement



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

- **satisfiable:** path to unsafe state

- **unsatisfiable:** refine the abstraction

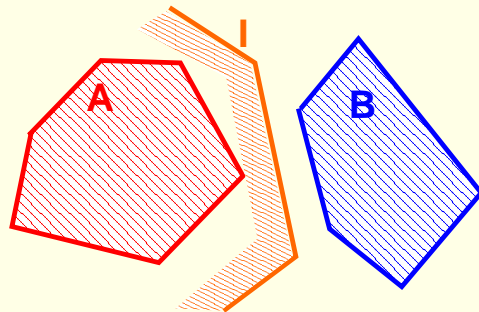
[McMillan 2003-2006] use interpolants (generated from proofs)

Interpolation

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

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

If so, **I** is an **interpolant** for **A** and **B**.



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

... but, for an arbitrary first-order theory \mathcal{T} , **I** may contain quantifiers even if **A** and **B** are ground.

Ground Interpolation

\mathcal{T} theory; A, B sets of **ground** clauses in the language of \mathcal{T} with $A \wedge B \models_{\mathcal{T}} \perp$

Does there exist a **ground** formula I , containing only **constants** (& **function 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 **ground interpolant** for A and B .

Abstraction-based verification [McMillan 2003-2006]

Abstraction refinement \mapsto use interpolants (generated from proofs)

Theories: propositional logic, LI + UIF

Ground Interpolation

\mathcal{T} theory; A, B sets of ground clauses

Does there exist a **ground** formula I (with **no** symbols?) occurring in both A and B ?
If so, I is an **ground interpolant** for A, B .

Applications

1. **Combinations of local extensions**
2. **Distributed databases**
3. **Verification**

- “Constrained interpolation” for $LI(\mathbb{Q}) + UIF$
Implementation (ARMC, Blast)
[Rybalchenko, VS'07]
- Abstraction refinement, widening, invariant generation

Abstraction-based verification [McM]

Abstraction refinement \mapsto use interpolants (generated from proofs)

Theories: propositional logic, $LI + UIF$

Our Contribution [VS 2006]: hierarchical interpolation

in classes of **local** extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$

Conclusions

- **Efficient reasoning in complex theories:** limit search / modularity
 - **Local theory extensions:** hierarchic reasoning; many examples
 - **Combine extensions:** modular reasoning; information exchange

Main advantage: Identify situations when completeness is guaranteed

Conclusions

- **Efficient reasoning in complex theories:** limit search / modularity
 - **Local theory extensions:** hierarchic reasoning; many examples
 - **Combine extensions:** modular reasoning; information exchange

Main advantage: Identify situations when completeness is guaranteed

Standard methods:

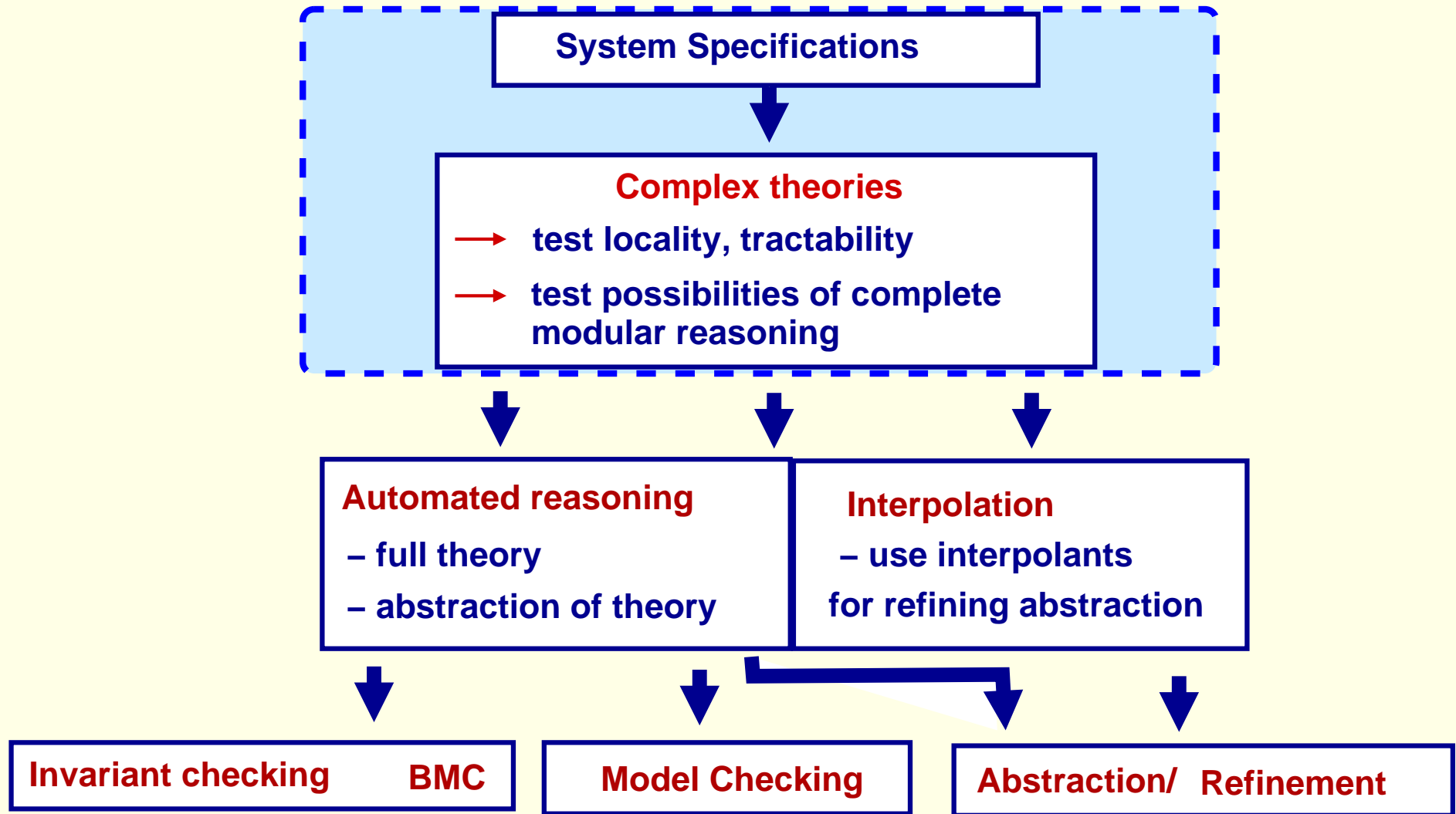
- Combinations of theories: $DPLL(T) \mapsto SAT$ of ground formulae
- Quantified formulae: use instantiations (heuristics; completeness unknown)

- Our method:**
- identifies the instances necessary for preserving completeness
 - **guarantees:** completeness, termination, complexity
(Implementation (with S.Jacobs and C. Ihlemann))
 - parametric systems \mapsto constraints on parameters for safety

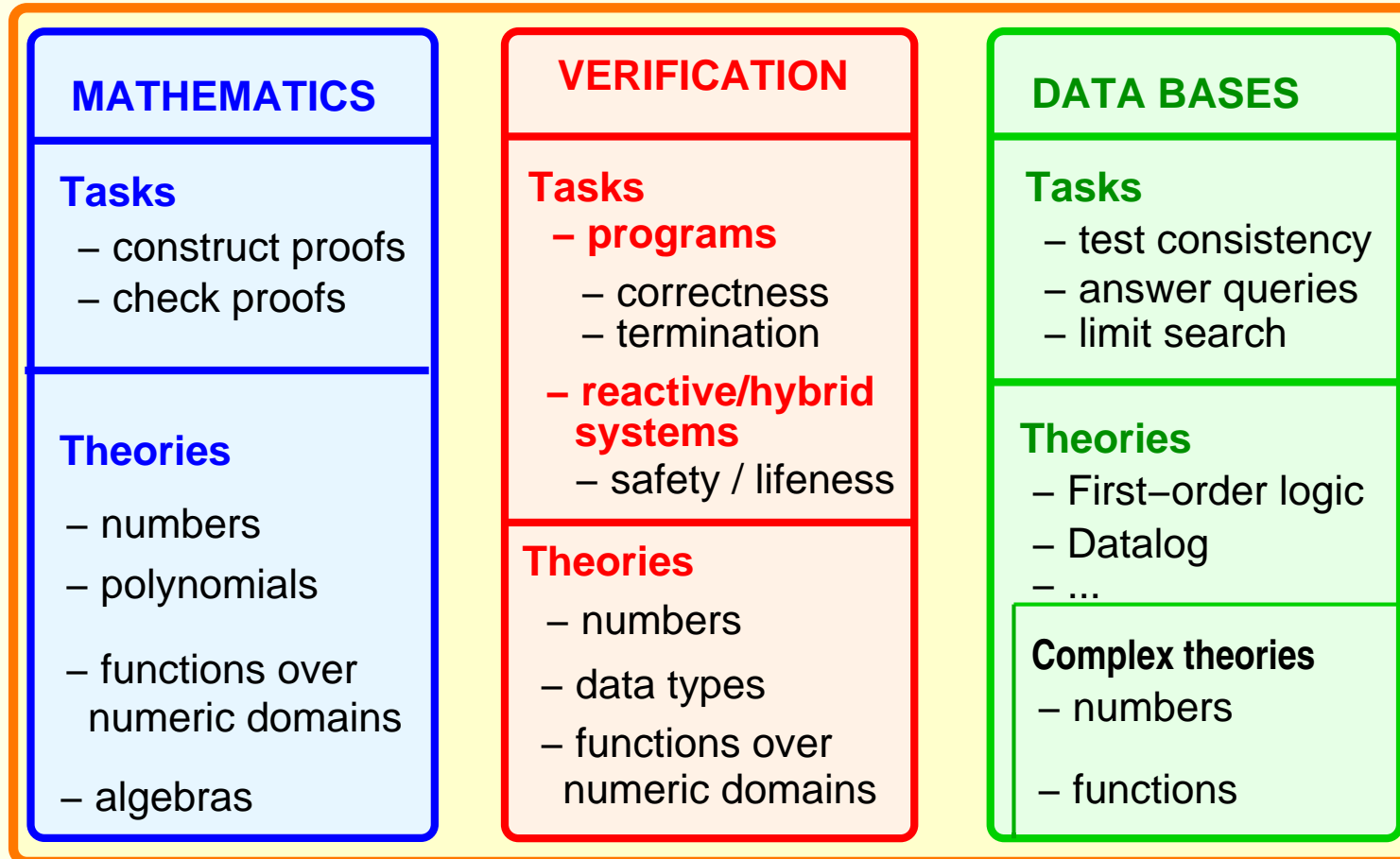
Conclusions

- **Efficient reasoning in complex theories:** limit search / modularity
 - **Local theory extensions:** hierarchic reasoning; many examples
 - **Combine extensions:** modular reasoning; information exchange
- **Interpolation in local theory extensions**
 - hierarchic method
 - more general domains than [McMillan'04]
 - implementation (ARMC, Blast)

Verification



Examples of application domains



complex systems (MAS, reactive systems w. embedded software, databases)