Recent Improvements of Theory Reasoning in Vampire

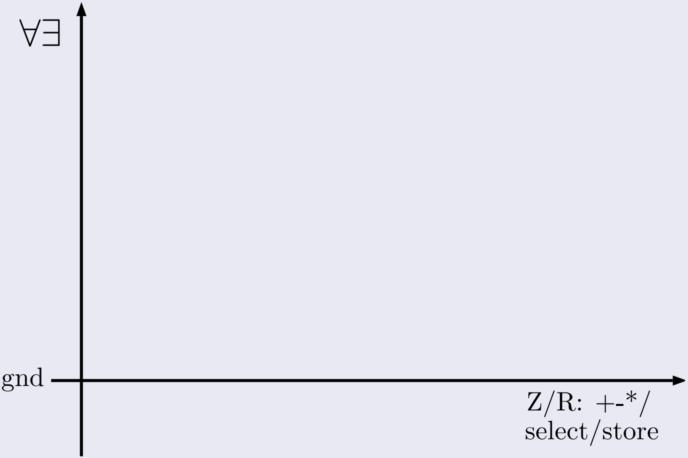
Giles Reger[1], Martin Suda[2], Andrei Voronkov[3,4]

[1]University of Manchester, Manchester, UK
[2]TU Wien, Vienna, Austria
[3]Chalmers University of Technology, Gothenburg, Sweden
[4]EasyChair

IWIL 2017 – Maun, May 7, 2017

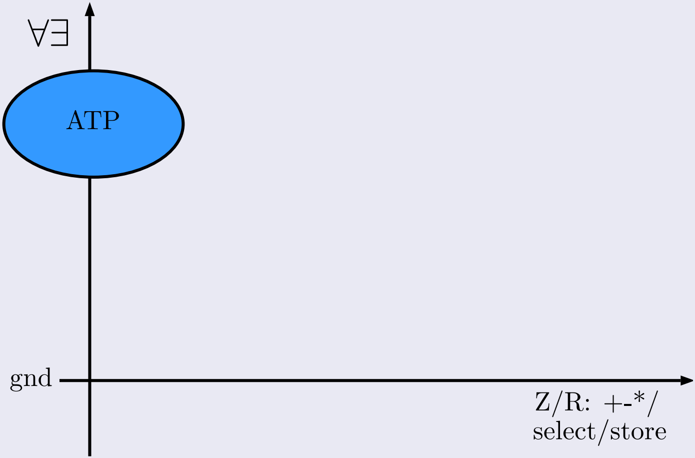
Reasoning with quantifiers and theories

Two Dimensions of Complexity

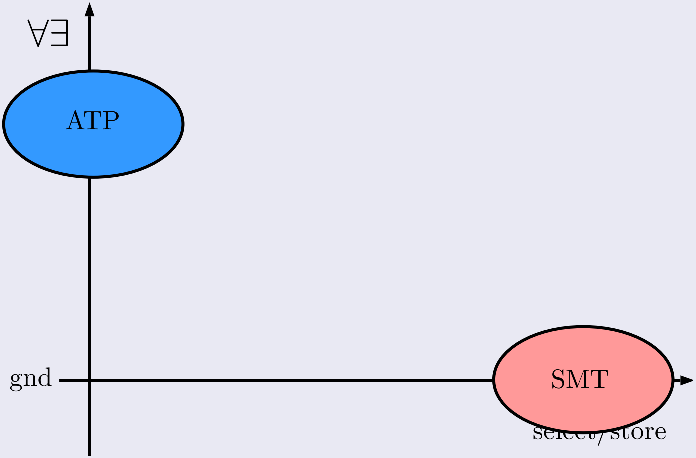


Reasoning with quantifiers and theories

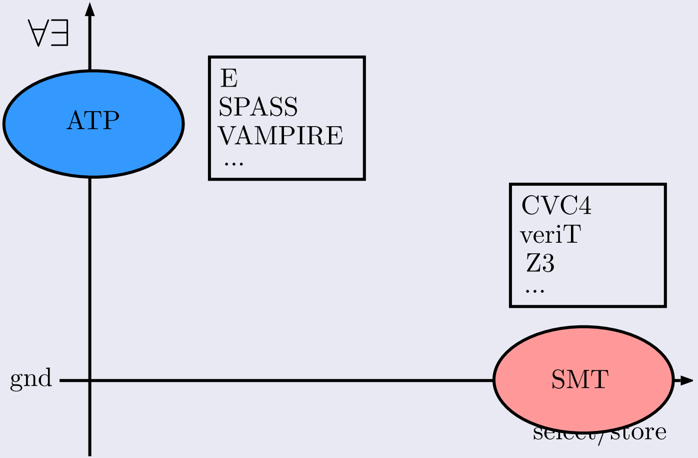
Two Dimensions of Complexity



Two Dimensions of Complexity

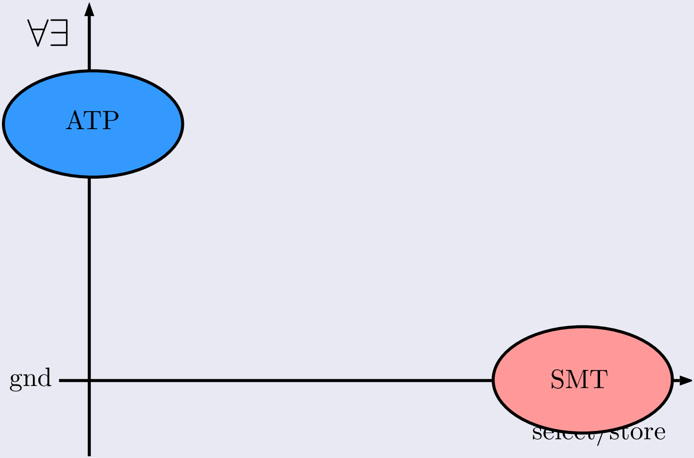


Two Dimensions of Complexity

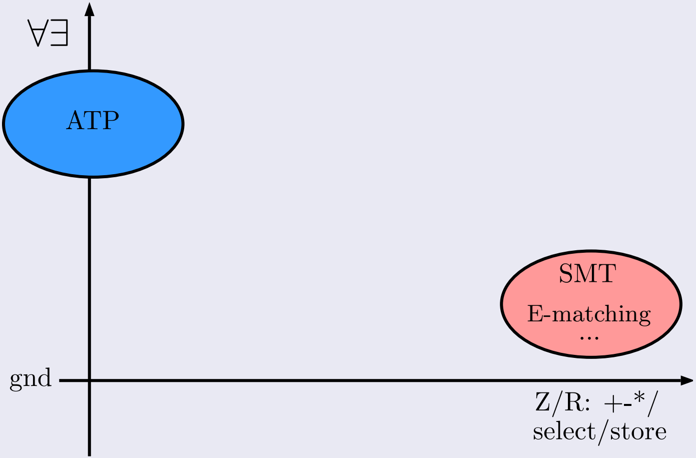


Reasoning with quantifiers and theories

Two Dimensions of Complexity

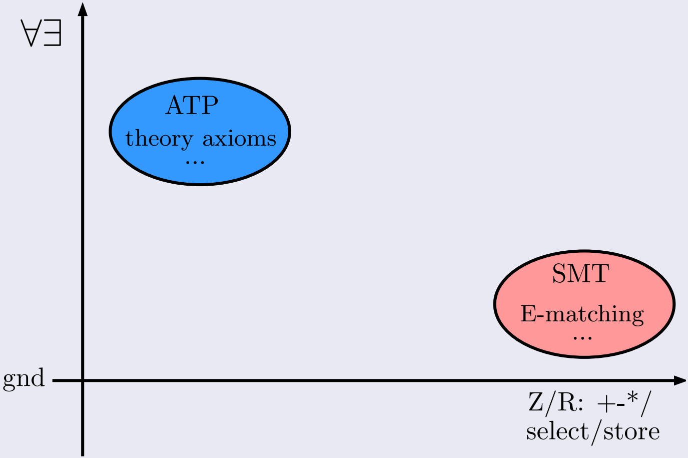


Two Dimensions of Complexity



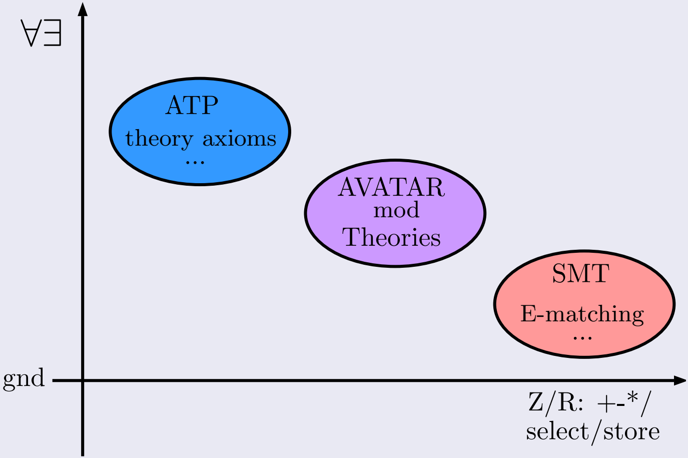
Reasoning with quantifiers and theories

Two Dimensions of Complexity



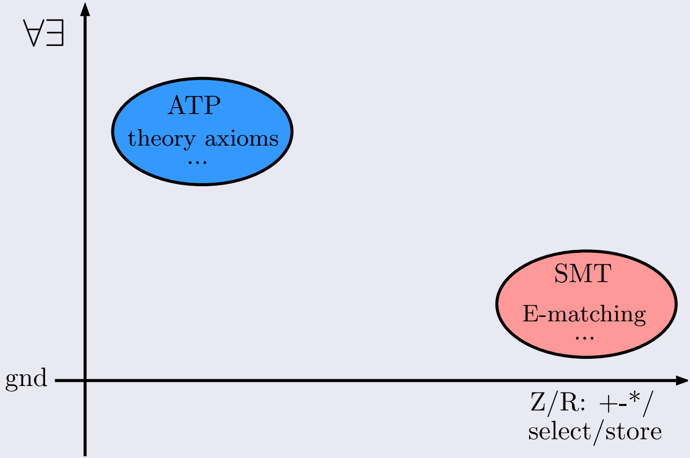
Reasoning with quantifiers and theories

Two Dimensions of Complexity



Reasoning with quantifiers and theories

Two Dimensions of Complexity



This talk in one slide

Contribution 1: Theory Instantiation Rule

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\simeq x^2 + 49 \vee p(x)$$

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \lor p(x) \implies p(7)$$

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \vee p(x) \implies p(7)$$

- by utilising ground SMT solving

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \vee p(x) \implies p(7)$$

- by utilising ground SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \vee p(x) \implies p(7)$$

- by utilising ground SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

Contribution 2: Unification with Abstraction

- extension of unification that introduces theory constraints

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \vee p(x) \implies p(7)$$

- by utilising ground SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

Contribution 2: Unification with Abstraction

- extension of unification that introduces theory constraints
- $p(2x)$ against $\neg p(10)$

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a <u>non-ground</u> clause

$$14x \not\approx x^2 + 49 \lor p(x) \implies p(7)$$

- by utilising <u>ground</u> SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

Contribution 2: Unification with Abstraction

- extension of unification that introduces theory constraints
- $p(2x)$ against $\neg p(10) \implies 2x \not\approx 10$

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\approx x^2 + 49 \lor p(x) \implies p(7)$$

- by utilising ground SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

Contribution 2: Unification with Abstraction

- extension of unification that introduces theory constraints
- $p(2x)$ against $\neg p(10) \implies 2x \not\approx 10$
- a lazy approach to abstraction

This talk in one slide

Contribution 1: Theory Instantiation Rule

- derives a simplifying instance of a non-ground clause

$$14x \not\simeq x^2 + 49 \vee p(x) \implies p(7)$$

- by utilising ground SMT solving
- (current) limitation: complete theories (e.g. arithmetic)

Contribution 2: Unification with Abstraction

- extension of unification that introduces theory constraints
- $p(2x)$ against $\neg p(10) \implies 2x \not\simeq 10$
- a lazy approach to abstraction
- new constrains can be often "discharged" by 1.

Outline

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \Longrightarrow 2$

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \Longrightarrow 2$
- add theory axioms: $x + 0 = x$, $x + y = y + x$, $\ldots$

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \Longrightarrow 2$
- add theory axioms: $x + 0 = x$, $x + y = y + x$, $\ldots$
- AVATAR modulo theories

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \implies 2$
- add theory axioms: $x + 0 = x$, $x + y = y + x$, $\ldots$
- AVATAR modulo theories

Theory abstraction rule

$$L[t] \vee C \implies x \not\simeq t \vee L[x] \vee C,$$

where L is a theory literal, t a non-theory term, and x fresh.

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \Longrightarrow 2$
- add theory axioms: $x + 0 = x$, $x + y = y + x$, ...
- AVATAR modulo theories

Theory abstraction rule

$$L[t] \vee C \implies x \not\simeq t \vee L[x] \vee C,$$

where L is a theory literal, t a non-theory term, and x fresh.

Example

$5 < f(y) \vee p(y) \implies x \not\simeq f(y) \vee 5 < x \vee p(y)$

Short preliminaries

Main Arsenal for Theory reasoning in Vampire

- evaluate ground terms: $1 + 1 \Longrightarrow 2$
- add theory axioms: $x + 0 = x$, $x + y = y + x$, ...
- AVATAR modulo theories

Theory abstraction rule

$$L[t] \vee C \quad \Longrightarrow \quad x \not\simeq t \vee L[x] \vee C,$$

where L is a theory literal, t a non-theory term, and x fresh.

Example

$5 < f(y) \vee p(y) \quad \Longrightarrow \quad x \not\simeq f(y) \vee 5 < x \vee p(y)$

NB: abstraction can be “undone” by the equality factoring rule

Outline

Theory instantiation by examples

Example

Consider the conjecture $(\exists x)(x + x \simeq 2)$ negated and clausified to

$$x + x \not\simeq 2.$$

It takes Vampire 15 seconds to solve using theory axioms deriving lemmas such as

$$x + 1 \simeq y + 1 \vee y + 1 \leq x \vee x + 1 \leq y.$$

Theory instantiation by examples

Example

Consider the conjecture $(\exists x)(x + x \simeq 2)$ negated and clausified to

$$x + x \not\simeq 2.$$

It takes Vampire 15 seconds to solve using theory axioms deriving lemmas such as

$$x + 1 \simeq y + 1 \vee y + 1 \leq x \vee x + 1 \leq y.$$

Example (`ARI120=1`)

Initial clauses:

$$x * x \not\simeq 4 \vee x \simeq y \vee \neg p(y) \qquad p(2)$$

Theory instantiation by examples

Example

Consider the conjecture $(\exists x)(x + x \simeq 2)$ negated and clausified to

$$x + x \not\simeq 2.$$

It takes Vampire 15 seconds to solve using theory axioms deriving lemmas such as

$$x + 1 \simeq y + 1 \vee y + 1 \leq x \vee x + 1 \leq y.$$

Example (ARI120=1)

Initial clauses:

$$x * x \not\simeq 4 \vee x \simeq y \vee \neg p(y) \qquad p(2)$$

immediately resolve to $$x * x \not\simeq 4 \vee 2 \simeq x,$$

but this cannot be solved with axioms only in reasonable time.

Theory instantiation more formally

As an inference rule

$$\frac{C}{(D[\mathsf{x}])\theta}\ \mathit{TheoryInst}$$

where $\mathcal{A}_{(P)}(C) = T[\mathsf{x}] \to D[\mathsf{x}]$ is a (partial) abstraction of C, and θ a subst. such that $T[\mathsf{x}]\theta$ is valid in the underlying theory.

Theory instantiation more formally

As an inference rule

$$\frac{C}{(D[\mathbf{x}])\theta}\ \mathit{TheoryInst}$$

where $\mathcal{A}_{(P)}(C) = T[\mathbf{x}] \to D[\mathbf{x}]$ is a (partial) abstraction of C, and θ a subst. such that $T[\mathbf{x}]\theta$ is valid in the underlying theory.

Implementation:

- Abstract relevant literals
- Collect relevant pure theory literals $L_1, \ldots, L_n$
- Run an SMT solver on $T[\mathbf{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$
- If the SMT solver returns a model, transform it into a substitution θ and produce an instance
- If the SMT solver returns <u>unsatisfiable</u> then C is a theory tautology and can be removed

Theory instantiation more formally

As an inference rule

$$\frac{C}{(D[\mathsf{x}])\theta}\ \mathit{TheoryInst}$$

where $\mathcal{A}_{(P)}(C) = T[\mathsf{x}] \to D[\mathsf{x}]$ is a (partial) abstraction of C, and θ a subst. such that $T[\mathsf{x}]\theta$ is valid in the underlying theory.

Implementation:

- Abstract relevant literals
- Collect relevant pure theory literals $L_1, \ldots, L_n$
- Run an SMT solver on $T[\mathsf{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$
- If the SMT solver returns a model, transform it into a substitution θ and produce an instance
- If the SMT solver returns unsatisfiable then C is a theory tautology and can be removed

When (not) to abstract

Example

Consider a unit clause $p(1,5)$ abstracted as

$$(x \simeq 1 \wedge y \simeq 5) \rightarrow p(x,y).$$

The only "solution substitution" is $\theta = \{x \mapsto 1, y \mapsto 5\}$.

When (not) to abstract

Example

Consider a unit clause $p(1,5)$ abstracted as

$$(x \simeq 1 \wedge y \simeq 5) \rightarrow p(x,y).$$

The only "solution substitution" is $\theta = \{x \mapsto 1, y \mapsto 5\}$.

Example

Consider a theory instantiation step

$$x \not\simeq 1 + y \vee p(x,y) \implies p(1,0).$$

When (not) to abstract

Example

Consider a unit clause $p(1,5)$ abstracted as

$$(x \simeq 1 \land y \simeq 5) \to p(x,y).$$

The only "solution substitution" is $\theta = \{x \mapsto 1, y \mapsto 5\}$.

Example

Consider a theory instantiation step

$$x \not\simeq 1 + y \lor p(x,y) \implies p(1,0).$$

But we can obtain a "more general" instance

$$p(y+1,y)$$

using equality resolution.

Selecting Pure Theory Literals

Example (some literals constrain less/more than others)

$$(x \not\simeq 0) \to p(x)$$

Selecting Pure Theory Literals

Example (some literals constrain less/more than others)

$$(x \not\simeq 0) \rightarrow p(x)$$

Three options for `thi`:

- `strong`: Only select <u>strong</u> literals where a literal is strong if it is a negative equality or an interpreted literal
- `overlap`: Select all strong literals and additionally those theory literals whose variables overlap with a strong literal
- `all`: Select all non-trivial pure theory literals

Interacting with the SMT solver

Recall that we collect relevant pure theory literals $L_1, \ldots, L_n$ to run an SMT solver on $T[\mathsf{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$

- the negation step involves Skolemization
- the we just translate the terms via Z3 API

Interacting with the SMT solver

Recall that we collect relevant pure theory literals $L_1, \ldots, L_n$ to run an SMT solver on $T[\mathrm{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$

- the negation step involves Skolemization
- the we just translate the terms via Z3 API

Example (The Division by zero catch!)

The following two clauses are satisfiable:

$$1/x \not\simeq 0 \vee p(x) \qquad 1/x \simeq 0 \vee \neg p(x).$$

Interacting with the SMT solver

Recall that we collect relevant pure theory literals $L_1, \ldots, L_n$ to run an SMT solver on $T[\mathrm{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$

- the negation step involves Skolemization
- the we just translate the terms via Z3 API

Example (The Division by zero catch!)

The following two clauses are satisfiable:

$$1/x \not\simeq 0 \vee p(x) \qquad 1/x \simeq 0 \vee \neg p(x).$$

However, instances $p(0)$ and $\neg p(0)$ could be obtained by an "unprotected" instantiation rule.

Interacting with the SMT solver

Recall that we collect relevant pure theory literals $L_1, \ldots, L_n$ to run an SMT solver on $T[\mathbf{x}] = \neg L_1 \wedge \ldots \wedge \neg L_n$

- the negation step involves Skolemization
- the we just translate the terms via Z3 API

Example (The Division by zero catch!)

The following two clauses are satisfiable:

$$1/x \not\simeq 0 \vee p(x) \qquad 1/x \simeq 0 \vee \neg p(x).$$

However, instances $p(0)$ and $\neg p(0)$ could be obtained by an "unprotected" instantiation rule.

Evaluation may fail:

- result out of Vampire's internal range
- result is a proper algebraic number

A new redundancy rule

Theory Tautology Deletion

Recall we abstract C as $T[\mathsf{x}] \rightarrow D[\mathsf{x}]$. If the SMT solver shows that $T[\mathsf{x}]$ is unsatisfiable, we can remove C from the search space.

A new redundancy rule

Theory Tautology Deletion

Recall we abstract C as $T[\mathsf{x}] \to D[\mathsf{x}]$. If the SMT solver shows that $T[\mathsf{x}]$ is unsatisfiable, we can remove C from the search space.

Be careful about:

A new redundancy rule

Theory Tautology Deletion

Recall we abstract C as $T[\mathsf{x}] \rightarrow D[\mathsf{x}]$. If the SMT solver shows that $T[\mathsf{x}]$ is unsatisfiable, we can remove C from the search space.

Be careful about:

- the interaction with theory axiom support

A new redundancy rule

Theory Tautology Deletion

Recall we abstract C as $T[\mathsf{x}] \to D[\mathsf{x}]$. If the SMT solver shows that $T[\mathsf{x}]$ is unsatisfiable, we can remove C from the search space.

Be careful about:

- the interaction with theory axiom support
- handling of division by zero

Outline

Abstraction for unification example

Example

Consider two clauses

$$r(14y) \qquad \neg r(x^2 + 49) \vee p(x).$$

Abstraction for unification example

Example

Consider two clauses

$$r(14y) \qquad \neg r(x^2 + 49) \vee p(x).$$

We could fully abstract them to obtain:

$$r(u) \vee u \not\approx 14y \qquad \neg r(v) \vee v \not\approx x^2 + 49 \vee p(x),$$

Abstraction for unification example

Example

Consider two clauses

$$r(14y) \qquad \neg r(x^2 + 49) \vee p(x).$$

We could fully abstract them to obtain:

$$r(u) \vee u \not\approx 14y \qquad \neg r(v) \vee v \not\approx x^2 + 49 \vee p(x),$$

then resolve to get

$$u \not\approx 14y \vee u \not\approx x^2 + 49 \vee p(x).$$

Abstraction for unification example

Example

Consider two clauses

$$r(14y) \qquad \neg r(x^2 + 49) \vee p(x).$$

We could fully abstract them to obtain:

$$r(u) \vee u \not\simeq 14y \qquad \neg r(v) \vee v \not\simeq x^2 + 49 \vee p(x),$$

then resolve to get

$$u \not\simeq 14y \vee u \not\simeq x^2 + 49 \vee p(x).$$

Finally, Theory Instantiation could produce

$$p(7).$$

Why one might not like full abstraction

1. Fully abstracted clauses are much longer

Why one might not like full abstraction

1. Fully abstracted clauses are much longer
2. The AVATAR modulo theories approach cannot help (full abstraction destroys ground literals)

Why one might not like full abstraction

1. Fully abstracted clauses are much longer
2. The AVATAR modulo theories approach cannot help
   (full abstraction destroys ground literals)
3. incompatible with theory axiom reasoning
   (theory part requires special treatment)

Why one might not like full abstraction

1. Fully abstracted clauses are much longer
2. The AVATAR modulo theories approach cannot help
   (full abstraction destroys ground literals)
3. incompatible with theory axiom reasoning
   (theory part requires special treatment)
4. inferences need to be protected from undoing abstraction
   (recall equality resolution)

Unification with constraints

Instead of full abstraction ...

- incorporate the abstraction process into unification
- thus abstractions are "on demand" and lazy

Unification with constraints

Instead of full abstraction ...

- incorporate the abstraction process into unification
- thus abstractions are "on demand" and lazy

We define a function $\mathrm{mgu}_{\mathsf{Abs}}(t, s) = (\theta, \Gamma)$ such that

$$(\Gamma \rightarrow t \simeq s)\theta \text{ is valid}$$

and θ is the most general such substitution given Γ.

Unification with constraints

Instead of full abstraction . . .

- incorporate the abstraction process into unification
- thus abstractions are "on demand" and lazy

We define a function $\mathrm{mgu}_{\mathrm{Abs}}(t, s) = (\theta, \Gamma)$ such that

$$(\Gamma \rightarrow t \simeq s)\theta \text{ is valid}$$

and θ is the most general such substitution given Γ.

Example (strive for generality)

Unifying $a + b$ with $c + d$ should produce $(\{\}, a + b = c + d)$ and not $(\{\}, a = c \wedge b = d)$.

Unification with constraints Algorithm part I

function $\text{mgu}_{\text{Abs}}(s, t)$

    if t is a variable and not $\texttt{occurs}(t, s)$ **then**

        return $(\{t \mapsto s\}, \mathit{true})$

    if s is a variable and not $\texttt{occurs}(s, t)$ **then**

        return $(\{s \mapsto t\}, \mathit{true})$

    if s and t have different top-level symbols **then**

        if $\texttt{canAbstract}(s, t)$ **then**

            return $(\{\}, s = t)$

        return $(\perp, \perp)$

    if s and t are constants **then**

        return $(\{\}, \mathit{true})$

    ...

...

let $s = f(s_1, \ldots, s_n)$ **and** $t = f(t_1, \ldots, t_n)$ **in**

$\theta = \{\}$ **and** $\Gamma = \mathit{true}$

for $i = 1$ **to** n **do**

$\quad (\theta_i, \Gamma_i) = \mathrm{mgu}_{\mathrm{Abs}}((s_i\theta), (t_i\theta))$

$\quad$ **if** $(\theta_i, \Gamma_i) = (\bot, \bot)$ **or** $\mathtt{canAbstract}(s, t)$ **and** $\Gamma_i \neq \mathit{true}$

then

$\qquad$ **if** $\mathtt{canAbstract}(s, t)$ **then**

$\qquad\quad$ **return** $(\{\}, s = t)$

$\qquad$ **return** $(\bot, \bot)$

$\quad \theta = \theta \circ \theta_i$ **and** $\Gamma = \Gamma \wedge \Gamma_i$

return (θ, Γ)

When do we abstract?

Example (do not produce unsatisfiable constraints)

Allowing $p(1)$ and $p(2)$ to unify under the constraint that $1 \simeq 2$ is not useful in any context.

`canAbstract` will always be false if the two terms are always non-equal in the underlying theory.

When do we abstract?

Example (do not produce unsatisfiable constraints)

Allowing $p(1)$ and $p(2)$ to unify under the constraint that $1 \simeq 2$ is not useful in any context.

`canAbstract` will always be false if the two terms are always non-equal in the underlying theory. On top of that

For option to choose from:

- `interpreted_only`: only produce a constraint if the top-level symbol of both terms is a theory-symbol,
- `one_side_interpreted`: only produce a constraint if the top-level symbol of at least one term is a theory symbol,
- `one_side_constant`: as `one_side_interpreted` but if the other side is uninterpreted it must be a constant,
- `all`: allow all terms of theory sort to unify and produce constraints.

Updated calculus

New inference rule: Resolution-wA

$$\frac{\underline{A} \vee C_1 \quad \underline{\neg A'} \vee C_2}{(\Gamma \rightarrow (C_1 \vee C_2))\theta}\ ,$$

where $(\theta, \Gamma) = \mathrm{mgu}_{\mathsf{Abs}}(A, A')$ and A is not an equality literal.

Updated calculus

New inference rule: **Resolution-wA**

$$\frac{\underline{A} \vee C_1 \quad \underline{\neg A'} \vee C_2}{(\Gamma \to (C_1 \vee C_2))\theta} \;,$$

where $(\theta, \Gamma) = \mathrm{mgu}_{\mathrm{Abs}}(A, A')$ and A is not an equality literal.

Example (eager evaluation is not a problem anymore)

When starting from an obvious conflict:

$$p(1+3) \qquad \neg p(x+3)$$

Updated calculus

New inference rule: **Resolution-wA**

$$\frac{\underline{A} \vee C_1 \quad \underline{\neg A'} \vee C_2}{(\Gamma \to (C_1 \vee C_2))\theta},$$

where $(\theta, \Gamma) = \mathrm{mgu}_{\mathsf{Abs}}(A, A')$ and A is not an equality literal.

Example (eager evaluation is not a problem anymore)

When starting from an obvious conflict:

$$p(1+3) \qquad \neg p(x+3)$$

eager evaluation destroys this by simplifying

$$p(1+3) \implies p(4).$$

Updated calculus

New inference rule: **Resolution-wA**

$$\frac{\underline{A} \vee C_1 \quad \underline{\neg A'} \vee C_2}{(\Gamma \to (C_1 \vee C_2))\theta} \ ,$$

where $(\theta, \Gamma) = \text{mgu}_{\text{Abs}}(A, A')$ and A is not an equality literal.

Example (eager evaluation is not a problem anymore)

When starting from an obvious conflict:

$$p(1+3) \qquad \neg p(x+3)$$

eager evaluation destroys this by simplifying

$$p(1+3) \implies p(4).$$

But **Resolution-wA** allows us to still derive

$$4 \not\simeq x + 3$$

and Theory Instantiation could derive the empty clause.

Outline

Experiments

The setup:

- implemented in Vampire
- Manchester cluster: 2x4core @ 2.4 GHz and 24GiB per node
- all SMTLIB with quantifiers and theories (-BV)

Experiments

The setup:

- implemented in Vampire
- Manchester cluster: 2x4core @ 2.4 GHz and 24GiB per node
- all SMTLIB with quantifiers and theories (-BV)

How to assess the value of a new technique?

Experiments

The setup:

- implemented in Vampire
- Manchester cluster: 2x4core @ 2.4 GHz and 24GiB per node
- all SMTLIB with quantifiers and theories (-BV)

How to assess the value of a new technique?

- it can interact in many ways (with around 60 other options)

Experiments

The setup:

- implemented in Vampire
- Manchester cluster: 2x4core @ 2.4 GHz and 24GiB per node
- all SMTLIB with quantifiers and theories (-BV)

How to assess the value of a new technique?

- it can interact in many ways (with around 60 other options)
- our methodology:
  - discard easy and unsolvable problems
  - generate random strategy and vary the interesting part

Experiments

The setup:

- implemented in Vampire
- Manchester cluster: 2x4core @ 2.4 GHz and 24GiB per node
- all SMTLIB with quantifiers and theories (-BV)

How to assess the value of a new technique?

- it can interact in many ways (with around 60 other options)
- our methodology:
  - discard easy and unsolvable problems
  - generate random strategy and vary the interesting part

For this experiment:

- 24 reasonable combinations of option values: `fta`, `uwa`, `thi`
- approx. 100 000 runs in total

Comparison of Three Options

fta	uwa	thi	solutions
on	off	all	252
on	off	overlap	265
on	off	strong	266
on	off	off	276
off	all	all	333
off	all	overlap	351
off	all	strong	354
off	one_side_interpreted	all	364
off	all	off	364
off	one_side_constant	all	374
off	interpreted_only	all	379
off	one_side_interpreted	overlap	385
off	one_side_interpreted	strong	387
off	off	all	392
off	one_side_constant	strong	397
off	one_side_constant	overlap	401
off	interpreted_only	overlap	407
off	one_side_interpreted	off	407
off	interpreted_only	strong	409
off	one_side_constant	off	417
off	off	overlap	428
off	interpreted_only	off	430
off	off	strong	431
off	off	off	450

Contribution of New Options to Strategy Building

A new technique in the context of a portfolio approach:

1. helps to solve problems faster
2. solves previously unsolved problems

Contribution of New Options to Strategy Building

A new technique in the context of a portfolio approach:

1. helps to solve problems faster
2. solves previously unsolved problems

The latter dominates in FO reasoning since . . .

If a problem is solvable by a prover, it is usually solvable with a short running time.

Contribution of New Options to Strategy Building

A new technique in the context of a portfolio approach:

1. helps to solve problems faster
2. solves previously unsolved problems

The latter dominates in FO reasoning since ...

If a problem is solvable by a prover, it is usually solvable with a short running time.

Problems newly solved thanks to `thi` and `uwa`:

- ALIA (arrays and linear integer arithmetic): 2
- AUFNIRA: **3**
- LIA : 10
- LRA : 1
- UFLIA : **1**
- UFNIA : 1

Conclusion

Two new techniques for reasoning with theories and quantifiers

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT
- technique 2 (`uwa`): lazy abstraction during unification

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT
- technique 2 (`uwa`): lazy abstraction during unification
- implemented in Vampire
- promising first results

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT
- technique 2 (`uwa`): lazy abstraction during unification
- implemented in Vampire
- promising first results

Directions for future work

- combine (`thi`) with the information in AVATAR modulo theory

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT
- technique 2 (`uwa`): lazy abstraction during unification
- implemented in Vampire
- promising first results

Directions for future work

- combine (`thi`) with the information in AVATAR modulo theory
- "general solutions" in terms of "parameters"

Conclusion

Two new techniques for reasoning with theories and quantifiers

- technique 1 (`thi`): simplifying instances via SMT
- technique 2 (`uwa`): lazy abstraction during unification
- implemented in Vampire
- promising first results

Directions for future work

- combine (`thi`) with the information in AVATAR modulo theory
- "general solutions" in terms of "parameters"

Thank you for your attention!