Efficient Interpolant Generation Algorithms based on Quantifier Elimination: EUF, Octagons

Deepak Kapur

Department of Computer Science
University of New Mexico
Albuquerque, NM, USA

September 14, 2017

Outline

- Quantifier Elimination and Interpolants for Theories

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas
  - $O(n^3)$ algorithm which can be further improved.

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas
  - $O(n^3)$ algorithm which can be further improved.
  - Strongest Interpolant

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas
  - $O(n^3)$ algorithm which can be further improved.
  - Strongest Interpolant
  - Simple interpolant

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas
  - $O(n^3)$ algorithm which can be further improved.
  - Strongest Interpolant
  - Simple interpolant
- Combination Algorithm

Outline

- Quantifier Elimination and Interpolants for Theories
- Generating Interpolants without a refutation proof.
- Interpolant Generation for quantifier-free theory of equality of uninterpreted symbols (EUF)
  - Strongest interpolant
  - Simple interpolant
  - Congruence closure of conditional equations.
- Octagonal formulas
  - $O(n^3)$ algorithm which can be further improved.
  - Strongest Interpolant
  - Simple interpolant
- Combination Algorithm
- Other theories: Transitive Closure, Difference Logic, Nonlinear Polynomial Inequalities, etc.

Craig's Interpolants

- Given α and β such that $\alpha \implies \beta$, an intermediate formula γ in common symbols of α and β exists and can be constructed such that

$$\alpha \implies \gamma \text{ and } \gamma \implies \beta.$$

Craig's Interpolants

- Given α and β such that $\alpha \implies \beta$, an intermediate formula γ in common symbols of α and β exists and can be constructed such that

$$\alpha \implies \gamma \text{ and } \gamma \implies \beta.$$

- The existence and construction typically relies on a proof.

Craig's Interpolants

- Given α and β such that $\alpha \implies \beta$, an intermediate formula γ in common symbols of α and β exists and can be constructed such that

$$\alpha \implies \gamma \text{ and } \gamma \implies \beta.$$

- The existence and construction typically relies on a proof.
- An alternate formulation: Given α and β' (typically $\neg\beta$) such that $\alpha \wedge \beta'$ is unsatisfiable, there exists a γ such that

$$\alpha \implies \gamma \text{ and } \neg(\gamma \wedge \beta').$$

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T *eliminates quantifiers* if for every formula φ, a quantifier-free formula ψ such that φ and ψ are T-equivalent and $\mathit{vars}(\psi) \subseteq \mathit{vars}(\varphi)$, can be computed.

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T *eliminates quantifiers* if for every formula φ, a quantifier-free formula ψ such that φ and ψ are T-equivalent and $vars(\psi) \subseteq vars(\varphi)$, can be computed.
  - **Examples:** Presburger arithmetic (over reals, integers, rational arithmetic), Tarski's theory of real arithmetic, the theory of recursively defined data structures, and the theory of finite sets, etc.

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T *eliminates quantifiers* if for every formula φ, a quantifier-free formula ψ such that φ and ψ are T-equivalent and $vars(\psi) \subseteq vars(\varphi)$, can be computed.
  - **Examples:** Presburger arithmetic (over reals, integers, rational arithmetic), Tarski's theory of real arithmetic, the theory of recursively defined data structures, and the theory of finite sets, etc.
  - Theories that do not admit quantifier elimination include the theory of equality over uninterpreted symbols (since satisfiability problem is undecidable), theory of arrays and theory of finite multisets, etc.

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T is *interpolating* if, for all pair of formulas φ, ψ such that $\varphi \implies \psi$, an interpolant α of (φ, ψ) can be computed.

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T is *interpolating* if, for all pair of formulas φ, ψ such that $\varphi \Longrightarrow \psi$, an interpolant α of (φ, ψ) can be computed.
- T is *quantifier-free interpolating* if a quantifier-free interpolant α of (φ, ψ) can be computed.

Kapur et al (FSE06): Quantifier elimination and Interpolants

- A theory T is *interpolating* if, for all pair of formulas φ, ψ such that $\varphi \Longrightarrow \psi$, an interpolant α of (φ, ψ) can be computed.
- T is *quantifier-free interpolating* if a quantifier-free interpolant α of (φ, ψ) can be computed.
- An interpolant can be extracted from any first-order proof Π of the unsatisfiability of $\varphi \wedge \psi$ in time linear in the size of the proof Π. Methods for extracting interpolants from first-order proofs exist for Gentzen-like calculi, resolution, and tableaux, etc.

Flattening

Flattening: A quantifier-free formula is *flat* if all atoms occurring in it are of the form $x = y$, $x = f(x_1, \ldots, x_n)$, or $p(x_1, \ldots, x_n)$, where $x, y, x_1, \ldots, x_n$ are variables.
It is easy to see that every formula can be converted to an

equisatisfiable flat formula by introducing new variables in linear time.

Proposition

Let $\varphi \wedge \psi$ be a quantifier-free T-unsatisfiable Σ-formula. Then, in order to compute a T-interpolant of (φ, ψ), one can just do the computation for a flat form $\varphi' \wedge \psi'$ of $\varphi \wedge \psi$.

Interpolant Generation

Theorem

1. *Every recursively enumerable theory is interpolating.*
2. *Every recursively enumerable theory that eliminates quantifiers is quantifier-free interpolating.*
3. *Every quantifier-free interpolating theory eliminates quantifiers.*

Examples of interpolating theories: Presburger arithmetic (over the reals, integers, rationals), the theory of real arithmetic, theories of containers (recursively defined data structures, finite arrays, finite sets, finite multisets).
Examples of quantifier-free interpolating theories: Presburger arithmetic (over the reals, integers, rationals), theories of real arithmetic, recursively defined data structures and finite sets.
In constrast, the theories of finite arrays and finite multisets are not quantifier-free interpolating. However they can be shown to be quantifier-free interpolating with signature extension (for Skolem functions).

Combination Algorithms for Interpolant Generation

- A quantifier-free theory over a container (e.g. finite lists, finite sets, finite multisets) is reduced to (combination of theories over) Presburger arithmetic and EUF. However, there may be issue of getting an interpolant in the original signature.

Combination Algorithms for Interpolant Generation

- A quantifier-free theory over a container (e.g. finite lists, finite sets, finite multisets) is reduced to (combination of theories over) Presburger arithmetic and EUF. However, there may be issue of getting an interpolant in the original signature.
- A combination algorithm to generate interpolants using algorithms over component theories.

A proof is not needed

Given α and a subset UC of symbols in α to be eliminated, let γ be such that symbols of $\gamma \subseteq$ of the symbols of α and no symbol in UC appears in γ:

γ is an interpolant of α if $\alpha \Longrightarrow \gamma$ and for every β implied by α with no symbols from UC, $\gamma \Longrightarrow \beta$.

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,
- interpolants of (α, β) form a lattice using strict implication ordering (mod out equivalence).

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,
- interpolants of (α, β) form a lattice using strict implication ordering (mod out equivalence).
- For every pair (α, β) such that $\alpha \implies \beta$, there is the strongest interpolant as well as the weakest interpolant.

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,
- interpolants of (α, β) form a lattice using strict implication ordering (mod out equivalence).
- For every pair (α, β) such that $\alpha \implies \beta$, there is the strongest interpolant as well as the weakest interpolant.
- Interpolant generation algorithms that extract interpolants from a proof of (α, β) generate interpolants of different strengths depending upon how a proof is traversed as well as the proof used.

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,
- interpolants of (α, β) form a lattice using strict implication ordering (mod out equivalence).
- For every pair (α, β) such that $\alpha \implies \beta$, there is the strongest interpolant as well as the weakest interpolant.
- Interpolant generation algorithms that extract interpolants from a proof of (α, β) generate interpolants of different strengths depending upon how a proof is traversed as well as the proof used.
- Different proofs can lead to different interpolants.

Interpolants form a Lattice under Implication Ordering

- If γ_1 and γ_2 are interpolants of (α, β), then so is $\gamma_1 \wedge \gamma_2$ as well as $\gamma_1 \vee \gamma_2$,
- interpolants of (α, β) form a lattice using strict implication ordering (mod out equivalence).
- For every pair (α, β) such that $\alpha \implies \beta$, there is the strongest interpolant as well as the weakest interpolant.
- Interpolant generation algorithms that extract interpolants from a proof of (α, β) generate interpolants of different strengths depending upon how a proof is traversed as well as the proof used.
- Different proofs can lead to different interpolants.
- Although interpolants have been used extensively in software model checking for abstraction refinement, generating new predicates, generalization of *IC*3 for invariant generation, little is known about how variety of interpolants affect the performance as well as the quality of invariants generated.

Interpolant Generation over EUF

Given α and UC, the set of symbols in α to be eliminated,

- Flatten by introducing new symbols:
  $x = y, x \neq y, x = f(y_1, y_2)$.

Interpolant Generation over EUF

Given α and UC, the set of symbols in α to be eliminated,

- Flatten by introducing new symbols: $x = y, x \neq y, x = f(y_1, y_2)$.
- A new symbol is **common** iff the function term it stands for, has only common symbols. Otherwise, it is **uncommon**. If there is an equation $x = f(y_1, y_2)$ where x is uncommon but f, y_1, y_2 are common, introduce a new common symbol u to stand for $f(y_1, y_2)$ and replace the above equation by $x = u$ and $u = f(y_1, y_2)$.

Interpolant Generation over EUF

Given α and UC, the set of symbols in α to be eliminated,

- Flatten by introducing new symbols:
  $x = y, x \neq y, x = f(y_1, y_2)$.
- A new symbol is **common** iff the function term it stands for, has only common symbols. Otherwise, it is **uncommon**. If there is an equation $x = f(y_1, y_2)$ where x is uncommon but f, y_1, y_2 are common, introduce a new common symbol u to stand for $f(y_1, y_2)$ and replace the above equation by $x = u$ and $u = f(y_1, y_2)$.
- Constant Congruence Closure: Using Union-Find and/or balanced trees.

Interpolant Generation over EUF

Given α and UC, the set of symbols in α to be eliminated,

- Flatten by introducing new symbols: $x = y, x \neq y, x = f(y_1, y_2)$.
- A new symbol is **common** iff the function term it stands for, has only common symbols. Otherwise, it is **uncommon**. If there is an equation $x = f(y_1, y_2)$ where x is uncommon but f, y_1, y_2 are common, introduce a new common symbol u to stand for $f(y_1, y_2)$ and replace the above equation by $x = u$ and $u = f(y_1, y_2)$.
- Constant Congruence Closure: Using Union-Find and/or balanced trees.
- Eliminate uncommon constants that have an equivalent common constant. The result is equations and disequations without any eliminated uncommon constants.

Interpolant Generation over EUF

- Eliminate uncommon function symbols by generating Horn clauses (conditional equations): $x_1 = f(y_1, y_2), x_2 = f(z_1, z_2)$ where f is uncommon, gives:
  $(y_1 = z_1 \wedge y_2 = z_2) \implies x_1 = x_2$.
  These conditional equations are subsequently used to eliminate x_1 or x_2 in case it is an uncommon constant.

Interpolant Generation over EUF

- Eliminate uncommon function symbols by generating Horn clauses (conditional equations): $x_1 = f(y_1, y_2), x_2 = f(z_1, z_2)$ where f is uncommon, gives:
  $(y_1 = z_1 \wedge y_2 = z_2) \Longrightarrow x_1 = x_2$.
  These conditional equations are subsequently used to eliminate x_1 or x_2 in case it is an uncommon constant.
- Delete all equations with a uncommon symbol.

Interpolant Generation over EUF

- Eliminate uncommon function symbols by generating Horn clauses (conditional equations): $x_1 = f(y_1, y_2), x_2 = f(z_1, z_2)$ where f is uncommon, gives:
  $(y_1 = z_1 \wedge y_2 = z_2) \Longrightarrow x_1 = x_2$.
  These conditional equations are subsequently used to eliminate x_1 or x_2 in case it is an uncommon constant.
- Delete all equations with a uncommon symbol.
- If new equalities on constants generated repeat the previous steps to eliminate any additional uncommon symbols.

Interpolant Generation over EUF

- Eliminate uncommon function symbols by generating Horn clauses (conditional equations): $x_1 = f(y_1, y_2), x_2 = f(z_1, z_2)$ where f is uncommon, gives:
  $(y_1 = z_1 \wedge y_2 = z_2) \implies x_1 = x_2$.
  These conditional equations are subsequently used to eliminate x_1 or x_2 in case it is an uncommon constant.
- Delete all equations with a uncommon symbol.
- If new equalities on constants generated repeat the previous steps to eliminate any additional uncommon symbols.
- Expose any uncommon constant under neath a common function symbol by Horn clauses: Given
  $x_1 = f(y_1, y_2), x_2 = f(z_1, z_2)$ where f is common but at least one of y_1, y_2, z_1, z_2 is uncommon, generate
  $(y_1 = z_1 \wedge y_2 = z_2) \implies x_1 = x_2$.

Watch Out

- Successively eliminate uncommon constants by conditional rewriting insofar as conditional rewriting does not introduce new uncommon constants.

Watch Out

- Successively eliminate uncommon constants by conditional rewriting insofar as conditional rewriting does not introduce new uncommon constants.
- This step can cause exponential blow ups in the worst case.

Watch Out

- Successively eliminate uncommon constants by conditional rewriting insofar as conditional rewriting does not introduce new uncommon constants.
- This step can cause exponential blow ups in the worst case.
- May be possible to avoid blow up by relabeling an uncommon symbol as a place holder for a (finite set of) conditional equations.

Watch Out

- Successively eliminate uncommon constants by conditional rewriting insofar as conditional rewriting does not introduce new uncommon constants.
- This step can cause exponential blow ups in the worst case.
- May be possible to avoid blow up by relabeling an uncommon symbol as a place holder for a (finite set of) conditional equations.
- After repeated application of the above rules, eliminate all equations, disequations and Horn clauses that contain uncommon symbols.

Watch Out

- Successively eliminate uncommon constants by conditional rewriting insofar as conditional rewriting does not introduce new uncommon constants.
- This step can cause exponential blow ups in the worst case.
- May be possible to avoid blow up by relabeling an uncommon symbol as a place holder for a (finite set of) conditional equations.
- After repeated application of the above rules, eliminate all equations, disequations and Horn clauses that contain uncommon symbols.
- The result is the interpolant generated from α after all uncommon symbols have been eliminated.

An Illustration

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.

An Illustration

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Flatten: introduce new uncommon symbols $n_1 = f(y_1, v), n_2 = f(y_2, v)$ producing flattened $\alpha_f = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(y_1, v) = n_1, f(y_2, v) = n_2., f(n_1, n_2) = t\}$ with v, n_1, n_2 being uncommon which must be eliminated.

An Illustration

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Flatten: introduce new uncommon symbols $n_1 = f(y_1, v), n_2 = f(y_2, v)$ producing flattened $\alpha_f = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(y_1, v) = n_1, f(y_2, v) = n_2., f(n_1, n_2) = t\}$ with v, n_1, n_2 being uncommon which must be eliminated.
- Trivial Constant congruence closure: no uncommon constants are eliminated.

An Illustration

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Flatten: introduce new uncommon symbols $n_1 = f(y_1, v), n_2 = f(y_2, v)$ producing flattened $\alpha_f = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(y_1, v) = n_1, f(y_2, v) = n_2., f(n_1, n_2) = t\}$ with v, n_1, n_2 being uncommon which must be eliminated.
- Trivial Constant congruence closure: no uncommon constants are eliminated.
- No uncommon function symbol. However, v, n_1, n_2 are hidden under the common function symbol f and must be exposed by generating Horn clauses: $\{1.\ z_1 = z_2 \implies s_1 = s_2,\ 2.\ z_1 = y_1 \implies n_1 = s_1,\ 3.\ z_1 = y_2 \implies n_2 = s_1,\ 4.\ (z_1 = n_1 \wedge v = n_2) \implies s_1 = t,\ 5.\ z_2 = y_1 \implies n_1 = s_2,\ 6.\ z_2 = y_2 \implies n_2 = s_2,\ 7.\ (z_2 = n_1 \wedge v = n_2) \implies s_2 = t,\ 8.\ y_1 = y_2 \implies n_2 = n_1,\ 9.\ (y_1 = n_1 \wedge v = n_2) \implies n_1 = t,\ 10.\ (y_2 = n_1 \wedge v = n_2) \implies n_2 = t\}$.

Good, Bad and Ugly

- v cannot be eliminated as there is no (conditional) equation replacing v. So all Horn clauses with v as well as f-equations are deleted as they do not play any role in computing an interpolant: Delete $\{4, 7, 9, 10\}$ from the above Horn clauses as well as all equations in which v appears in α_f. Only one equation $f(n_1, n_2) = t$ is left.

Good, Bad and Ugly

- v cannot be eliminated as there is no (conditional) equation replacing v. So all Horn clauses with v as well as f-equations are deleted as they do not play any role in computing an interpolant: Delete $\{4, 7, 9, 10\}$ from the above Horn clauses as well as all equations in which v appears in α_f. Only one equation $f(n_1, n_2) = t$ is left.
- This example illustrates the good, bad and the ugly.

Good, Bad and Ugly

- v cannot be eliminated as there is no (conditional) equation replacing v. So all Horn clauses with v as well as f-equations are deleted as they do not play any role in computing an interpolant: Delete $\{4, 7, 9, 10\}$ from the above Horn clauses as well as all equations in which v appears in α_f. Only one equation $f(n_1, n_2) = t$ is left.
- This example illustrates the good, bad and the ugly.
- n_1 can be eliminated using conditional rewriting by Horn clauses 2 and 5. n_2 can be eliminated similarly.
  After that all equations and conditional equations in which n_1, n_2 appear are deleted.

Good, Bad and Ugly

- v cannot be eliminated as there is no (conditional) equation replacing v. So all Horn clauses with v as well as f-equations are deleted as they do not play any role in computing an interpolant: Delete $\{4, 7, 9, 10\}$ from the above Horn clauses as well as all equations in which v appears in α_f. Only one equation $f(n_1, n_2) = t$ is left.
- This example illustrates the good, bad and the ugly.
- n_1 can be eliminated using conditional rewriting by Horn clauses 2 and 5. n_2 can be eliminated similarly.
  After that all equations and conditional equations in which n_1, n_2 appear are deleted.
- The interpolant is:
  $\{ z_1 = z_2 \implies s_1 = s_2, (z_2 = y_1 \wedge z_1 = y_1) \implies f(s_1, s_2) = t, (z_2 = y_2 \wedge z_1 = y_2) \implies f(s_2, s_1) = t, (z_1 = y_1 \wedge z_2 = y_1) \implies f(s_1, s_1) = t, (z_1 = y_2 \wedge z_2 = y_2) \implies f(s_2, s_2) = t\}$.

Good, Bad and Ugly

- v cannot be eliminated as there is no (conditional) equation replacing v. So all Horn clauses with v as well as f-equations are deleted as they do not play any role in computing an interpolant: Delete $\{4, 7, 9, 10\}$ from the above Horn clauses as well as all equations in which v appears in α_f. Only one equation $f(n_1, n_2) = t$ is left.
- This example illustrates the good, bad and the ugly.
- n_1 can be eliminated using conditional rewriting by Horn clauses 2 and 5. n_2 can be eliminated similarly. After that all equations and conditional equations in which n_1, n_2 appear are deleted.
- The interpolant is: $\{\ z_1 = z_2 \implies s_1 = s_2, (z_2 = y_1 \wedge z_1 = y_1) \implies f(s_1, s_2) = t, (z_2 = y_2 \wedge z_1 = y_2) \implies f(s_2, s_1) = t, (z_1 = y_1 \wedge z_2 = y_1) \implies f(s_1, s_1) = t, (z_1 = y_2 \wedge z_2 = y_2) \implies f(s_2, s_2) = t\}$.
- I_α is an interpolant for a family of β's in which $f, z_1, z_2, y_1, y_2, s_1, s_2, t$ appear along with any other symbols different from v insofar as $\alpha \implies \beta$.

Good, Bad and Ugly

- $$\begin{aligned} f(n_1, n_2) = t,\ z_1 = z_2 \implies s_1 = s_2, \\ 2.\ z_1 = y_1 \implies n_1 = s_1, \\ 3.\ z_1 = y_2 \implies n_2 = s_1, \\ 5.\ z_2 = y_1 \implies n_1 = s_2, \\ 6.\ z_2 = y_2 \implies n_2 = s_2, \\ 8.\ y_1 = y_2 \implies n_2 = n_1.\} \end{aligned}$$

Good, Bad and Ugly

- $$\begin{aligned} f(n_1, n_2) = t,\ z_1 = z_2 \implies s_1 = s_2,& \\ 2.\ z_1 = y_1 \implies n_1 = s_1,& \\ 3.\ z_1 = y_2 \implies n_2 = s_1,& \\ 5.\ z_2 = y_1 \implies n_1 = s_2,& \\ 6.\ z_2 = y_2 \implies n_2 = s_2,& \\ 8.\ y_1 = y_2 \implies n_2 = n_1.\}& \end{aligned}$$
- The blow-up is evident here. One way to avoid is to relabel n_1 and n_2 as place holders for conditional pointers and not perform substitutions.

Good, Bad and Ugly

- $$\begin{aligned} f(n_1, n_2) = t,\ z_1 = z_2 \Longrightarrow s_1 = s_2, \\ 2.\ z_1 = y_1 \Longrightarrow n_1 = s_1, \\ 3.\ z_1 = y_2 \Longrightarrow n_2 = s_1, \\ 5.\ z_2 = y_1 \Longrightarrow n_1 = s_2, \\ 6.\ z_2 = y_2 \Longrightarrow n_2 = s_2, \\ 8.\ y_1 = y_2 \Longrightarrow n_2 = n_1.\} \end{aligned}$$
- The blow-up is evident here. One way to avoid is to relabel n_1 and n_2 as place holders for conditional pointers and not perform substitutions.
- Similar to the situation in syntactic unification: $y = f(x_1, x_1), x_1 = f(x_2, x_2), \cdots$.
  The substitution for y if fully expanded blows up. Structure sharing helps.

Good, Bad and Ugly

- $$f(n_1, n_2) = t,\ z_1 = z_2 \Longrightarrow s_1 = s_2,$$
$$2.\ z_1 = y_1 \Longrightarrow n_1 = s_1,$$
$$3.\ z_1 = y_2 \Longrightarrow n_2 = s_1,$$
$$5.\ z_2 = y_1 \Longrightarrow n_1 = s_2,$$
$$6.\ z_2 = y_2 \Longrightarrow n_2 = s_2,$$
$$8.\ y_1 = y_2 \Longrightarrow n_2 = n_1.\}$$
- The blow-up is evident here. One way to avoid is to relabel n_1 and n_2 as place holders for conditional pointers and not perform substitutions.
- Similar to the situation in syntactic unification:
  $y = f(x_1, x_1), x_1 = f(x_2, x_2), \cdots$.
  The substitution for y if fully expanded blows up. Structure sharing helps.
- Conditional structure sharing needed.

Good, Bad and Ugly:Currying to the Rescue?

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.

Good, Bad and Ugly:Currying to the Rescue?

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Curry and Flatten: $A(f, z_1) = f_1, A(f_1, v) = s_1, A(f, z_2) = f_2, A(f_2, v) = s_2, A(f, y_1) = f_3, A(f_3, v) = n_1, A(f, y_2) = f_4, A(f_4, v) = n_2, A(f, n_1) = f_5, A(f_5, n_2) = t$, in which v, n_1, n_2 are uncommon.

Good, Bad and Ugly:Currying to the Rescue?

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Curry and Flatten: $A(f, z_1) = f_1, A(f_1, v) = s_1, A(f, z_2) = f_2, A(f_2, v) = s_2, A(f, y_1) = f_3, A(f_3, v) = n_1, A(f, y_2) = f_4, A(f_4, v) = n_2, A(f, n_1) = f_5, A(f_5, n_2) = t$, in which v, n_1, n_2 are uncommon.
- Eliminating v,
  $f_1 = f_2 \Longrightarrow s_1 = s_2, \quad f_1 = f_3 \Longrightarrow n_1 = s_1, \; f_1 = f_4 \Longrightarrow n_2 = s_1, \; f_2 = f_3 \Longrightarrow n_1 = s_2, \; f_2 = f_4 \Longrightarrow n_2 = s_2$.

Good, Bad and Ugly:Currying to the Rescue?

- $\alpha = \{f(z_1, v) = s_1, f(z_2, v) = s_2, f(f(y_1, v), f(y_2, v)) = t\}$ in which v is uncommon and must be eliminated.
- Curry and Flatten: $A(f, z_1) = f_1, A(f_1, v) = s_1, A(f, z_2) = f_2, A(f_2, v) = s_2, A(f, y_1) = f_3, A(f_3, v) = n_1, A(f, y_2) = f_4, A(f_4, v) = n_2, A(f, n_1) = f_5, A(f_5, n_2) = t$, in which v, n_1, n_2 are uncommon.
- Eliminating v, $f_1 = f_2 \implies s_1 = s_2, \quad f_1 = f_3 \implies n_1 = s_1, \ f_1 = f_4 \implies n_2 = s_1, \ f_2 = f_3 \implies n_1 = s_2, \ f_2 = f_4 \implies n_2 = s_2.$
- Can leave the output in triangular form where n_1, n_2 can be eliminated. Or, do replacement: $f_1 = f_2 \implies s_1 = s_2, \quad (f_1 = f_3 \wedge f_2 = f_3) \implies s_1 = s_2,$ $(f_2 = f_4 \wedge f_1 = f_4) \implies s_1 = s_2, \ f_1 = f_3 \implies A(f, s_1) = f_5,$ $f_2 = f_3 \implies A(f, s_2) = f_5, \ f_1 = f_4 \implies A(f_5, s_1) = t,$ $f_2 = f_4 \implies A(f_5, s_2) = t.$

McMillan's Algorithm and Tinelli et al's Algorithm

- Mutually contradictory $\alpha = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$ and $\beta = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$.
  Commons symbols are: $\{f, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.
  Symbols to be removed from α are $\{x_1, x_2, x_3, x_4\}$.

McMillan's Algorithm and Tinelli et al's Algorithm

- Mutually contradictory $\alpha = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$ and $\beta = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$.
  Commons symbols are: $\{f, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.
  Symbols to be removed from α are $\{x_1, x_2, x_3, x_4\}$.
- Constant congruence closure gives: $\{\{x_1, z_1\}, \{x_2, z_2\}, \{x_3, z_5\}, \{x_4, z_6\}\}$. Uncommon symbols $\{x_1, x_2, x_3, x_4\}$ are deleted after replacing them by their representative common symbols. The remaining f-equations are: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.

McMillan's Algorithm and Tinelli et al's Algorithm

- Mutually contradictory $\alpha = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$ and $\beta = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$.
  Commons symbols are: $\{f, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.
  Symbols to be removed from α are $\{x_1, x_2, x_3, x_4\}$.
- Constant congruence closure gives: $\{\{x_1, z_1\}, \{x_2, z_2\}, \{x_3, z_5\}, \{x_4, z_6\}\}$. Uncommon symbols $\{x_1, x_2, x_3, x_4\}$ are deleted after replacing them by their representative common symbols. The remaining f-equations are: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.
- The interpolant generated is: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.

McMillan's Algorithm and Tinelli et al's Algorithm

- Mutually contradictory $\alpha = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$ and $\beta = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$.
  Commons symbols are: $\{f, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.
  Symbols to be removed from α are $\{x_1, x_2, x_3, x_4\}$.
- Constant congruence closure gives: $\{\{x_1, z_1\}, \{x_2, z_2\}, \{x_3, z_5\}, \{x_4, z_6\}\}$. Uncommon symbols $\{x_1, x_2, x_3, x_4\}$ are deleted after replacing them by their representative common symbols. The remaining f-equations are: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.
- The interpolant generated is: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.
- The interpolant reported by McMillan's algorithm is: $(z_1 = z_2 \wedge (z_3 = z_4 \Longrightarrow z_5 = z_6)) \Longrightarrow (z_3 = z_4 \wedge z_7 = z_8)$.

McMillan's Algorithm and Tinelli et al's Algorithm

- Mutually contradictory $\alpha = \{x_1 = z_1, z_2 = x_2, z_3 = f(x_1), f(x_2) = z_4, x_3 = z_5, z_6 = x_4, z_7 = f(x_3), f(x_4) = z_8\}$ and $\beta = \{z_1 = z_2, z_5 = f(z_3), f(z_4) = z_6, y_1 = z_7, z_8 = y_2, y_1 \neq y_2\}$.
  Commons symbols are: $\{f, z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8\}$.
  Symbols to be removed from α are $\{x_1, x_2, x_3, x_4\}$.
- Constant congruence closure gives: $\{\{x_1, z_1\}, \{x_2, z_2\}, \{x_3, z_5\}, \{x_4, z_6\}\}$. Uncommon symbols $\{x_1, x_2, x_3, x_4\}$ are deleted after replacing them by their representative common symbols. The remaining f-equations are: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.
- The interpolant generated is: $\{f(z_1) = z_3, f(z_2) = z_4, f(z_5) = z_7, f(z_6) = z_8\}$.
- The interpolant reported by McMillan's algorithm is: $(z_1 = z_2 \wedge (z_3 = z_4 \Longrightarrow z_5 = z_6)) \Longrightarrow (z_3 = z_4 \wedge z_7 = z_8)$.
- Tinelli et al's algorithm produces $(z_1 = z_2 \Longrightarrow z_3 = z_4) \wedge (z_5 = z_6 \Longrightarrow z_7 = z_8)$.

Avoiding Exponential Blow-up

- Flattening: n steps. At most n new constants introduced, one for each internal node in the DAG representation.

Avoiding Exponential Blow-up

- Flattening: n steps. At most n new constants introduced, one for each internal node in the DAG representation.
- Constant Congruence: $O(n * log(n)))$ (amortized complexity: $O(n * \alpha(n)))$.

Avoiding Exponential Blow-up

- Flattening: n steps. At most n new constants introduced, one for each internal node in the DAG representation.
- Constant Congruence: $O(n * log(n)))$ (amortized complexity: $O(n * \alpha(n)))$.
- Generation of Horn clauses: $O(n^2)$. Each clause is of constant size to start with (depending upon the maximum arity of a function symbol).

Avoiding Exponential Blow-up

- Flattening: n steps. At most n new constants introduced, one for each internal node in the DAG representation.
- Constant Congruence: $O(n * log(n)))$ (amortized complexity: $O(n * \alpha(n)))$.
- Generation of Horn clauses: $O(n^2)$. Each clause is of constant size to start with (depending upon the maximum arity of a function symbol).
- Conditional Rewriting: If every constant eliminated once, then the complexity is polynomial: $O(n^3)$.
  Otherwise, because of multiple rules for a single uncommon constant, the final result after complete elimination can be exponential.

Avoiding Exponential Blow-up

- Flattening: n steps. At most n new constants introduced, one for each internal node in the DAG representation.
- Constant Congruence: $O(n * log(n)))$ (amortized complexity: $O(n * \alpha(n)))$.
- Generation of Horn clauses: $O(n^2)$. Each clause is of constant size to start with (depending upon the maximum arity of a function symbol).
- Conditional Rewriting: If every constant eliminated once, then the complexity is polynomial: $O(n^3)$.
  Otherwise, because of multiple rules for a single uncommon constant, the final result after complete elimination can be exponential.
- Dependency analysis and presenting the result in triangular form with conditional structure sharing could help (???).

Interpolant Generation over Octagonal Formulas

Given α, a conjunction of $x_i \leq c_i \wedge \pm x_j \pm x_k \leq c_{j,k}$, where x_j, x_k are distinct symbols.

- For every uncommon symbol y in α that needs to be eliminated, consider two distinct octagon atoms in which the sign of y is positive in one and negative in the other. y is eliminated by adding the two formulas. This must be done for every pair of such formulas.

Interpolant Generation over Octagonal Formulas

Given α, a conjunction of $x_i \leq c_i \wedge \pm x_j \pm x_k \leq c_{j,k}$, where x_j, x_k are distinct symbols.

- For every uncommon symbol y in α that needs to be eliminated, consider two distinct octagon atoms in which the sign of y is positive in one and negative in the other. y is eliminated by adding the two formulas. This must be done for every pair of such formulas.
- After this step, delete all atoms in which y appears.

Interpolant Generation over Octagonal Formulas

Given α, a conjunction of $x_i \leq c_i \wedge \pm x_j \pm x_k \leq c_{j,k}$, where x_j, x_k are distinct symbols.

- For every uncommon symbol y in α that needs to be eliminated, consider two distinct octagon atoms in which the sign of y is positive in one and negative in the other. y is eliminated by adding the two formulas. This must be done for every pair of such formulas.
- After this step, delete all atoms in which y appears.
- In case a formula of the form $2y \leq a$ (or $-2y \leq a$) is generated, it is normalized; in the case octagonal formulas are over the integers, then it is replaced by $y \leq \left\lfloor \left(\frac{a}{2}\right) \right\rfloor$ for $2y \leq a$.

Interpolant Generation over Octagonal Formulas

Given α, a conjunction of $x_i \leq c_i \wedge \pm x_j \pm x_k \leq c_{j,k}$, where x_j, x_k are distinct symbols.

- For every uncommon symbol y in α that needs to be eliminated, consider two distinct octagon atoms in which the sign of y is positive in one and negative in the other. y is eliminated by adding the two formulas. This must be done for every pair of such formulas.
- After this step, delete all atoms in which y appears.
- In case a formula of the form $2y \leq a$ (or $-2y \leq a$) is generated, it is normalized; in the case octagonal formulas are over the integers, then it is replaced by $y \leq \lfloor (\frac{a}{2}) \rfloor$ for $2y \leq a$.
- If an uncommon symbol only appears positively or negatively, all octagonal formulas containing it can be eliminated as they do not occur in the interpolant right at the start.

Interpolant Generation over Octagonal Formulas

Given α, a conjunction of $x_i \leq c_i \wedge \pm x_j \pm x_k \leq c_{j,k}$, where x_j, x_k are distinct symbols.

- For every uncommon symbol y in α that needs to be eliminated, consider two distinct octagon atoms in which the sign of y is positive in one and negative in the other. y is eliminated by adding the two formulas. This must be done for every pair of such formulas.
- After this step, delete all atoms in which y appears.
- In case a formula of the form $2y \leq a$ (or $-2y \leq a$) is generated, it is normalized; in the case octagonal formulas are over the integers, then it is replaced by $y \leq \left\lfloor \left(\frac{a}{2}\right) \right\rfloor$ for $2y \leq a$.
- If an uncommon symbol only appears positively or negatively, all octagonal formulas containing it can be eliminated as they do not occur in the interpolant right at the start.
- The result after eliminating all uncommon symbols is an interpolant generated from α.

Examples from Griggio's thesis

Mutually contradictory α and *beta*$'$:
$\alpha = \{(x_1 - x_2 \geq -4, -x_2 - x_3 \geq 5, x_2 + x_6 \geq 4, x_2 + x_5 \geq -3\}$,
$\beta' = \{-x_1 + x_3 \geq -2, -x_4 - x_6 \geq 0, -x_5 + x_4 \geq 0\}$ with uncommon symbol x_2 to be eliminated.

- Eliminate x_2:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.

Examples from Griggio's thesis

Mutually contradictory α and *beta*$'$:
$\alpha = \{(x_1 - x_2 \geq -4, -x_2 - x_3 \geq 5, x_2 + x_6 \geq 4, x_2 + x_5 \geq -3\}$,
$\beta' = \{-x_1 + x_3 \geq -2, -x_4 - x_6 \geq 0, -x_5 + x_4 \geq 0\}$ with uncommon symbol x_2 to be eliminated.

- Eliminate x_2:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- No literal from α is included in I_α since each contains x_2.

Examples from Griggio's thesis

Mutually contradictory α and *beta*$'$:
$\alpha = \{(x_1 - x_2 \geq -4, -x_2 - x_3 \geq 5, x_2 + x_6 \geq 4, x_2 + x_5 \geq -3\}$,
$\beta' = \{-x_1 + x_3 \geq -2, -x_4 - x_6 \geq 0, -x_5 + x_4 \geq 0\}$ with uncommon symbol x_2 to be eliminated.

- Eliminate x_2:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- No literal from α is included in I_α since each contains x_2.
- The interpolant is:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.

Examples from Griggio's thesis

Mutually contradictory α and $beta'$:
$\alpha = \{(x_1 - x_2 \geq -4, -x_2 - x_3 \geq 5, x_2 + x_6 \geq 4, x_2 + x_5 \geq -3\}$,
$\beta' = \{-x_1 + x_3 \geq -2, -x_4 - x_6 \geq 0, -x_5 + x_4 \geq 0\}$ with uncommon symbol x_2 to be eliminated.

- Eliminate x_2:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- No literal from α is included in I_α since each contains x_2.
- The interpolant is:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- Griggio's algorithm gives the conditional interpolant
  $-x_6 - x_5 \geq 0 \Longrightarrow x_1 - x_3 \geq 3$.

Examples from Griggio's thesis

Mutually contradictory α and *beta*$'$:
$\alpha = \{(x_1 - x_2 \geq -4, -x_2 - x_3 \geq 5, x_2 + x_6 \geq 4, x_2 + x_5 \geq -3\}$,
$\beta' = \{-x_1 + x_3 \geq -2, -x_4 - x_6 \geq 0, -x_5 + x_4 \geq 0\}$ with uncommon symbol x_2 to be eliminated.

- Eliminate x_2:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- No literal from α is included in I_α since each contains x_2.
- The interpolant is:
  $\{-x_3 + x_5 \geq 2, x_1 + x_6 \geq 0, x_1 + x_5 \geq -7, -x_3 + x_6 \geq 9\}$.
- Griggio's algorithm gives the conditional interpolant
  $-x_6 - x_5 \geq 0 \implies x_1 - x_3 \geq 3$.
- The strongest interpolant is an octagonal formula and is generated by our algorithm. No need for conditional interpolants.

Transitive Closure, Difference Logic etc.

- The framework works for other theories as well.

Transitive Closure, Difference Logic etc.

- The framework works for other theories as well.
- Interpolant generation algorithm for EUF works well in combination with other theories.

Concluding Remarks

- Quantifier Elimination provides a uniform mechanism for generating interpolants.

Concluding Remarks

- Quantifier Elimination provides a uniform mechanism for generating interpolants.
- Need to study heuristics to perform effective quantifier elimination. A complete quantifier elimination algorithm is not needed.

Concluding Remarks

- Quantifier Elimination provides a uniform mechanism for generating interpolants.
- Need to study heuristics to perform effective quantifier elimination. A complete quantifier elimination algorithm is not needed.
- Generate a variety of interpolants from the strongest using implication ordering.