Quantifier Elimination and Craig Interpolation: The Quantitative Way

Kevin Batz, Joost-Pieter Katoen, Nora Orhan

FoSSaCS

May 5, 2025, Hamilton, Canada

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

$$\exists y \colon x \leq y \wedge y \leq z$$

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

$$\exists y \colon x \leq y \wedge y \leq z$$

Quantifier Elimination [Fourier & Motzkin]

For every $\varphi \in \mathsf{LRA}$, there exists effectively constructible $\mathsf{QE}(\varphi) \in \mathsf{LRA}$ such that

1. $\mathsf{QE}(\varphi)$ is quantifier-free, and
2. $\mathsf{QE}(\varphi) \equiv \varphi$.

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

$$\exists y \colon x \leq y \wedge y \leq z$$

Quantifier Elimination [Fourier & Motzkin]

For every $\varphi \in \mathsf{LRA}$, there exists effectively constructible $\mathsf{QE}(\varphi) \in \mathsf{LRA}$ such that

1. $\mathsf{QE}(\varphi)$ is quantifier-free, and
2. $\mathsf{QE}(\varphi) \equiv \varphi$.

$$\mathsf{QE}(\exists y \colon x \leq y \wedge y \leq z) \;=\; x \leq z$$

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

$$\exists y \colon x \leq y \wedge y \leq z$$

Quantifier Elimination [Fourier & Motzkin]

For every $\varphi \in \mathsf{LRA}$, there exists effectively constructible $\mathsf{QE}(\varphi) \in \mathsf{LRA}$ such that

1. $\mathsf{QE}(\varphi)$ is quantifier-free, and
2. $\mathsf{QE}(\varphi) \equiv \varphi$.

- Template-based invariant synthesis [Colón et al. '03]:

$$\exists \text{ template instance} \colon \forall \text{ states} \colon \text{ invariant condition holds}$$

Quantifier Elimination and **Craig Interpolation** for Linear Rational Arithmetic are popular for automatic program verification.

$$\exists y \colon x \leq y \wedge y \leq z$$

Quantifier Elimination [Fourier & Motzkin]

For every $\varphi \in \mathsf{LRA}$, there exists effectively constructible $\mathsf{QE}(\varphi) \in \mathsf{LRA}$ such that

1. $\mathsf{QE}(\varphi)$ is quantifier-free, and
2. $\mathsf{QE}(\varphi) \equiv \varphi$.

- Template-based invariant synthesis [Colón et al. '03]:

$$\exists \text{ template instance} \colon \forall \text{ states} \colon \text{ invariant condition holds}$$

- Symbolic pre-image computations [Komuravelli et al. '14]:

$$\exists \, \text{succ} \colon \text{Trans}(\text{pred}, \text{succ}) \wedge \text{Goal}(\text{succ})$$

Craig Interpolation [Craig '57]

Let $\varphi, \psi \in \mathsf{LRA}$ with $\varphi \models \psi$. There exists effectively constructible $\underbrace{\theta \in \mathsf{LRA}}_{\textbf{Craig interpolant}}$ such that

1. θ is quantifier-free,
2. $\varphi \models \theta \models \psi$, and
3. $\mathsf{FV}(\theta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)$.

Craig Interpolation [Craig '57]

Let $\varphi, \psi \in \mathsf{LRA}$ with $\varphi \models \psi$. There exists effectively constructible $\underbrace{\theta \in \mathsf{LRA}}_{\textbf{Craig interpolant}}$ such that

1. θ is quantifier-free,
2. $\varphi \models \theta \models \psi$, and
3. $\mathsf{FV}(\theta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)$.

$$x = y_1 \wedge y_1 \leq y \quad \models \quad z = x \longrightarrow z \leq y$$

Craig Interpolation [Craig '57]

Let $\varphi, \psi \in \mathsf{LRA}$ with $\varphi \models \psi$. There exists effectively constructible $\underbrace{\theta \in \mathsf{LRA}}_{\text{Craig interpolant}}$ such that

1. θ is quantifier-free,
2. $\varphi \models \theta \models \psi$, and
3. $\mathsf{FV}(\theta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)$.

$$x = y_1 \wedge y_1 \leq y \quad \models \quad \underbrace{x \leq y}_{\text{concise explanation of entailment}} \quad \models \quad z = x \longrightarrow z \leq y$$

Craig Interpolation [Craig '57]

Let $\varphi, \psi \in \mathsf{LRA}$ with $\varphi \models \psi$. There exists effectively constructible $\underbrace{\theta \in \mathsf{LRA}}_{\textbf{Craig interpolant}}$ such that

1. θ is quantifier-free,
2. $\varphi \models \theta \models \psi$, and
3. $\mathsf{FV}(\theta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)$.

Application: Invariant synthesis from finite loop unrollings [McMillan '03, '06]

Craig Interpolation [Craig '57]

Let $\varphi, \psi \in \mathsf{LRA}$ with $\varphi \models \psi$. There exists effectively constructible $\underbrace{\theta \in \mathsf{LRA}}_{\textbf{Craig interpolant}}$ such that

1. θ is quantifier-free,
2. $\varphi \models \theta \models \psi$, and
3. $\mathsf{FV}(\theta) \subseteq \mathsf{FV}(\varphi) \cap \mathsf{FV}(\psi)$.

$$\text{loop unrolling} \quad \models \quad \underbrace{\text{potential loop invariant}}_{\text{concise explanation of entailment}} \quad \models \quad \text{specification}$$

Application: Invariant synthesis from finite loop unrollings [McMillan '03, '06]

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[x \leq y] \cdot x + [x > y] \cdot 0$$

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![[x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \;=\; \begin{cases} \sigma(x) & \text{if } \sigma \models x \leq y \\ 0 & \text{if } \sigma \models x > y \end{cases}$$

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\, [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \;=\; \begin{cases} \sigma(x) & \text{if } \sigma \models x \leq y \\ 0 & \text{if } \sigma \models x > y \end{cases}$$

1. Expected runtimes [Kaminski et al. '16]

How long does a program run on average?

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\, [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \;=\; \begin{cases} \sigma(x) & \text{if } \sigma \models x \leq y \\ 0 & \text{if } \sigma \models x > y \end{cases}$$

1. Expected runtimes [Kaminski et al. '16]

   How long does a program run on average?

2. Expected outcomes [Kozen '83, McIver '99, McIver & Morgan '05]

   What is the expected final value of variable x?

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\, [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \;=\; \begin{cases} \sigma(x) & \text{if } \sigma \models x \leq y \\ 0 & \text{if } \sigma \models x > y \end{cases}$$

1. Expected runtimes [Kaminski et al. '16]

   How long does a program run on average?

2. Expected outcomes [Kozen '83, McIver '99, McIver & Morgan '05]

   What is the expected final value of variable x?

3. Weighted programming [OOPSLA '22]

   What is the competitive ratio of an online algorithm?

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$\mathfrak{Q}x\colon\ [x \leq y] \cdot x + [x > y] \cdot 0$$

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\wp x\colon\ [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \ = \ \sup\{[\![[x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma[x \mapsto q]) \mid q \in \mathbb{Q}\}$$

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\exists x\colon [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \;=\; \sup\{[\![[x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma[x \mapsto q]) \mid q \in \mathbb{Q}\}$$

Contribution 1: Quantifier Elimination for Piecewise Linear Quantities

Given $f \in \mathsf{LinQuant}$, compute *quantifier-free* $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with $\mathsf{QE}(f) \equiv f$.

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\text{ƨ} x\colon\ [x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma) \ = \ \sup\{ [\![[x \leq y] \cdot x + [x > y] \cdot 0]\!](\sigma[x \mapsto q]) \mid q \in \mathbb{Q} \}$$

Contribution 1: Quantifier Elimination for Piecewise Linear Quantities

Given $f \in \mathsf{LinQuant}$, compute *quantifier-free* $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with $\mathsf{QE}(f) \equiv f$.

$$\mathsf{QE}(\text{ƨ} x\colon\ [x \leq y] \cdot x + [x > y] \cdot 0) \ = \ [y \geq 0] \cdot y + [y < 0] \cdot 0$$

Quantitative (probabilistic) program verification requires a shift

"assertions" evaluate to **numbers** instead of truth values

$$[\![\partial x\colon\ [x \le y] \cdot x + [x > y] \cdot 0]\!](\sigma) \ = \ \sup\{[\![[x \le y] \cdot x + [x > y] \cdot 0]\!](\sigma[x \mapsto q]) \mid q \in \mathbb{Q}\}$$

Contribution 1: Quantifier Elimination for Piecewise Linear Quantities

Given $f \in \mathsf{LinQuant}$, compute *quantifier-free* $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with $\mathsf{QE}(f) \equiv f$.

$$\mathsf{QE}(\partial x\colon\ [x \le y] \cdot x + [x > y] \cdot 0) \ = \ [y \ge 0] \cdot y + [y < 0] \cdot 0$$

- **Soundness and termination** for discontinuous or $\{\infty, -\infty\}$-valued quantities
- **Upper space complexity bounds**

Contribution 2: Quantitative Craig Interpolation for Piecewise Linear Quantities

Let $f, f' \in \mathsf{LinQuant}$ with $f \models f'$. There **exists effectively constructible** $g \in \mathsf{LinQuant}$ with

1. g is quantifier-free,
2. $f \models g \models f'$, and
3. $\mathsf{FV}(g) \subseteq \mathsf{FV}(f) \cap \mathsf{FV}(f')$.

Contribution 2: Quantitative Craig Interpolation for Piecewise Linear Quantities

Let $f, f' \in \mathsf{LinQuant}$ with $f \models f'$. There **exists effectively constructible** $g \in \mathsf{LinQuant}$ with

1. g is quantifier-free,
2. $f \models g \models f'$, and
3. $\mathsf{FV}(g) \subseteq \mathsf{FV}(f) \cap \mathsf{FV}(f')$.

$$[x \geq 0] \cdot x + [x \geq 0 \wedge y \leq x] \cdot y$$

$$\models \quad [x \geq 0 \wedge z \geq x] \cdot (2 \cdot x + z + 1) + [z < x] \cdot \infty$$

Contribution 2: Quantitative Craig Interpolation for Piecewise Linear Quantities

Let $f, f' \in \mathsf{LinQuant}$ with $f \models f'$. There **exists effectively constructible** $g \in \mathsf{LinQuant}$ with

1. g is quantifier-free,
2. $f \models g \models f'$, and
3. $\mathsf{FV}(g) \subseteq \mathsf{FV}(f) \cap \mathsf{FV}(f')$.

$$
\begin{aligned}
& [x \geq 0] \cdot x + [x \geq 0 \wedge y \leq x] \cdot y \\
\models \quad & [x \geq 0] \cdot 2 \cdot x \\
\models \quad & [x \geq 0 \wedge z \geq x] \cdot (2 \cdot x + z + 1) + [z < x] \cdot \infty
\end{aligned}
$$

Piecewise Linear Quantities

Quantitative Quantifier Elimination

Quantitative Craig Interpolation

Summary & Future Work

Piecewise Linear Quantities

The quantitative analogue of Linear Rational Arithmetic

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,
2. the φ_i are Boolean combinations of linear inequalities,

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,
2. the φ_i are Boolean combinations of linear inequalities,
3. the a_i are affine expressions or $\infty, -\infty$,

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,
2. the φ_i are Boolean combinations of linear inequalities,
3. the a_i are affine expressions or $\infty, -\infty$,
4. decidable side condition avoiding $\underbrace{-\infty + \infty}_{\text{undefined}}$.

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,
2. the φ_i are Boolean combinations of linear inequalities,
3. the a_i are affine expressions or $\infty, -\infty$,
4. decidable side condition avoiding $\underbrace{-\infty + \infty}_{\text{undefined}}$.

Semantics: Let $\Sigma = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}\}$. Then, for quantifier-free $f \in \mathsf{LinQuant}$ and $\sigma \in \Sigma$,

$$[\![f]\!](\sigma) \quad = \quad \sum_{i=1}^{n} \begin{cases} [\![a_i]\!](\sigma) & \text{if } \sigma \models \varphi_i \\ 0 & \text{if } \sigma \not\models \varphi_i \,. \end{cases}$$

Piecewise Linear Quantities

Syntax: *Piecewise linear quantities* in LinQuant are expressions f of the form

$$Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n \,,$$

where

1. $Q_i \in \{\exists, \forall\}$,
2. the φ_i are Boolean combinations of linear inequalities,
3. the a_i are affine expressions or $\infty, -\infty$,
4. decidable side condition avoiding $\underbrace{-\infty + \infty}_{\text{undefined}}$.

Semantics: Let $\Sigma = \{\sigma \mid \sigma \colon \mathsf{Vars} \to \mathbb{Q}\}$. Then, for quantifier-free $f \in \mathsf{LinQuant}$ and $\sigma \in \Sigma$,

$$[\![\exists x \colon f]\!](\sigma) \;=\; \sup\{[\![f]\!](\sigma[x \mapsto q]) \mid q \in \mathbb{Q}\} \,.$$

Quantitative Quantifier Elimination

Goal:

Given $f = Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

Goal:

Given $f = Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

First Observation: We have

$$\begin{aligned} & \mathsf{QE}(Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n) \\ = \; & \mathsf{QE}(\ldots \mathsf{QE}(Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n)) \end{aligned}$$

Goal:

Given $f = Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

First Observation: We have

$$\begin{aligned} & \mathsf{QE}(Q_1 x_1 \colon \ldots \colon Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n) \\ = \; & \mathsf{QE}(\ldots \mathsf{QE}(Q_k x_k \colon [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n)) \end{aligned}$$

$\Longrightarrow$ **Focus on eliminating a single quantifier.**

New Goal:

Given $f = Qx\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

New Goal:

Given $f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

New Goal:

Given $f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$, compute $\mathsf{QE}(f) \in \mathsf{LinQuant}$ with

1. $\mathsf{QE}(f)$ *quantifier-free*, and
2. $f \equiv \mathsf{QE}(f)$.

Our Approach: Divide-And-Conquer in 3 Stages

Stage 1

The Guarded Normal Form

Definition (Guarded Normal Form)

$f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.

Definition (Guarded Normal Form)

$f = \mathfrak{Q} x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.
2. The φ_i are in Disjunctive Normal Form.

Definition (Guarded Normal Form)

$f = \mathfrak{Q} x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.
2. The φ_i are in Disjunctive Normal Form.
3. Each linear inequality in the φ_i is rearranged for x.

Definition (Guarded Normal Form)

$f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.
2. The φ_i are in Disjunctive Normal Form.
3. Each linear inequality in the φ_i is rearranged for x.

Every f can be transformed into Guarded Normal Form.

Definition (Guarded Normal Form)

$f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.
2. The φ_i are in Disjunctive Normal Form.
3. Each linear inequality in the φ_i is rearranged for x.

Every f can be transformed into Guarded Normal Form.

$$\exists x\colon\ [x \leq y \vee x \leq z_1] \cdot x + [x \leq z_1] \cdot y$$

Definition (Guarded Normal Form)

$f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ is in *Guarded Normal Form*, if:

1. The φ_i partition the set Σ of valuations.
2. The φ_i are in Disjunctive Normal Form.
3. Each linear inequality in the φ_i is rearranged for x.

Every f can be transformed into Guarded Normal Form.

$$
\begin{aligned}
& \exists x\colon\ [x \leq y \vee x \leq z_1] \cdot x + [x \leq z_1] \cdot y \\
\equiv\ & \exists x\colon\ [(x \leq y \vee x \leq z_1) \wedge x \leq z_1] \cdot (x + y) + [(x \leq y \vee x \leq z_1) \wedge \neg(x \leq z_1)] \cdot x \\
& + [\neg(x \leq y \vee x \leq z_1) \wedge x \leq z_1] \cdot y + [\neg(x \leq y \vee x \leq z_1) \wedge \neg(x \leq z_1)] \cdot 0
\end{aligned}
$$

Theorem

If $f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ *is in Guarded Normal Form, then*

$$\mathsf{QE}(f) \ = \ \mathsf{MAX}\left(\mathsf{QE}(\exists x\colon\ [\varphi_i] \cdot a_i + [\neg\varphi_i] \cdot (-\infty)) \mid i \in \{1,\ldots,n\}\right) .$$

Theorem

If $f = \exists x\colon\ [\varphi_1] \cdot a_1 + \ldots + [\varphi_n] \cdot a_n$ *is in Guarded Normal Form, then*

$$\mathsf{QE}(f) \ = \ \mathsf{MAX}\,(\mathsf{QE}(\exists x\colon\ [\varphi_i] \cdot a_i + [\neg\varphi_i] \cdot (-\infty)) \mid i \in \{1,\ldots,n\})\ .$$

$\Longrightarrow$ **Focus on** $\mathsf{QE}(\exists x\colon\ [\varphi_i] \cdot a_i + [\neg\varphi_i] \cdot (-\infty))$**.**

Stage 2

Exploiting the Disjunctive Normal Form

Quantitative Quantifier Elimination

Theorem

For $f = \mathbb{Q}x\colon\ [\varphi] \cdot a + [\neg\varphi] \cdot (-\infty)$, *where*

$$\varphi \quad = \quad \bigvee_{i=1}^{n} D_i$$

is in DNF, we have

$$\mathsf{QE}(f) \ = \ \mathsf{MAX}\left(\mathsf{QE}(\mathbb{Q}x\colon\ [D_i] \cdot a + [\neg D_i] \cdot (-\infty)) \mid i \in \{1,\ldots,n\}\right) .$$

Theorem

For $f = \mathfrak{Q}x\colon\ [\varphi] \cdot a + [\neg\varphi] \cdot (-\infty)$, *where*

$$\varphi \quad = \quad \bigvee_{i=1}^{n} D_i$$

is in DNF, we have

$$\mathsf{QE}(f) \ = \ \mathsf{MAX}\left(\mathsf{QE}(\mathfrak{Q}x\colon\ [D_i] \cdot a + [\neg D_i] \cdot (-\infty)) \mid i \in \{1,\ldots,n\}\right) .$$

$\Longrightarrow$ **Focus on** $\mathsf{QE}(\mathfrak{Q}x\colon\ [D_i] \cdot a + [\neg D_i] \cdot (-\infty))$,

where D_i is a conjunction of linear inequalities.

Stage 3

Focus on $\mathsf{QE}(\partial x\colon [D_i] \cdot a_i + [\neg D_i] \cdot (-\infty))$,

where D_i is a conjunction of linear inequalities.

Quantitative Quantifier Elimination

$$f = \exists x \colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

$$f = \exists x \colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

$$f = \text{Ƨ}x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

$$f = \partial x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

$$f = \text{Ƨ}x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

   Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

2. If $\sigma \models \exists x\colon D$, **maximize** $2x$.

$$f = \partial x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

   Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

2. If $\sigma \models \exists x\colon D$, **maximize** $2x$. **Case distinction on upper bounds on x.**

$$f = \mathcal{Q}x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

   Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

2. If $\sigma \models \exists x\colon D$, **maximize** $2x$. **Case distinction on upper bounds on x.**

$$\mathsf{QE}(f) = \underbrace{[\neg(z < y_1 \wedge z \leq -y_2)]}_{\equiv \neg\exists x\colon D} \cdot (-\infty)$$

$$f = \partial x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

   Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

2. If $\sigma \models \exists x\colon D$, **maximize** $2x$. **Case distinction on upper bounds on x.**

$$\mathsf{QE}(f) = \underbrace{[\neg(z < y_1 \wedge z \leq -y_2)]}_{\equiv \neg \exists x\colon D} \cdot (-\infty) + \underbrace{[z < y_1 \wedge z \leq -y_2}_{\equiv \exists x\colon D} \wedge \underbrace{y_1 \leq -y_2]}_{y_1 \text{ is sup}} \cdot 2y_1$$

$$f \;=\; \mathfrak{Z}x\colon \underbrace{[x < y_1 \wedge x \leq -y_2 \wedge x \geq z]}_{=D} \cdot \underbrace{2x}_{=a} + [\neg D] \cdot (-\infty)$$

Let $\sigma \in \Sigma$ be a variable valuation.

1. If $\sigma \not\models \exists x\colon D$, evaluate to $-\infty$.

   Classic Fourier-Motzkin Elimination: $\exists x\colon D \equiv z < y_1 \wedge z \leq -y_2$

2. If $\sigma \models \exists x\colon D$, **maximize** $2x$. **Case distinction on upper bounds on x.**

$$\begin{aligned}
\mathsf{QE}(f) \;=\; & \underbrace{[\neg(z < y_1 \wedge z \leq -y_2)]}_{\equiv \neg\exists x\colon D} \cdot (-\infty) \\
& + \underbrace{[z < y_1 \wedge z \leq -y_2}_{\equiv \exists x\colon D} \wedge \underbrace{y_1 \leq -y_2]}_{y_1 \text{ is sup}} \cdot 2y_1 \\
& + \underbrace{[z < y_1 \wedge z \leq -y_2}_{\equiv \exists x\colon D} \wedge \underbrace{y_1 > -y_2]}_{-y_2 \text{ is sup}} \cdot 2 \cdot (-y_2)
\end{aligned}$$

Quantitative Quantifier Elimination

Theorem

*Our quantitative quantifier elimination algorithm is **sound** and **terminates**.*

Theorem

*Our quantitative quantifier elimination algorithm is **sound** and **terminates**.*

Theorem

If $f \in \mathsf{LinQuant}$ contains a single quantifier and

1. $|f|_{\rightarrow} = n$ *is the number of summands in* f*, and*
2. $|f|_{\downarrow} = m$ *is the maxmial size of the Boolean expressions in* f*,*

then

$$|\mathsf{QE}(f)|_{\rightarrow} \leq n \cdot 2^m \cdot (m+2)^{n \cdot 2^m} \quad \text{and} \quad |\mathsf{QE}(f)|_{\downarrow} \leq n \cdot 2^m \cdot \left(\left(\frac{m+2}{2}\right)^2 + m + 1\right).$$

Quantitative Craig Interpolation

Quantitative Craig Interpolation

Definition

Given $f, f' \in \mathsf{LinQuant}$ with $f \models f'$, we call $g \in \mathsf{LinQuant}$ a *Craig interpolant* of (f, f'), if

1. g is quantifier-free,
2. $f \models g \models f'$, and
3. $\mathsf{FV}(g) \subseteq \mathsf{FV}(f) \cap \mathsf{FV}(f')$.

Quantitative Craig Interpolation

Definition

Given $f, f' \in \mathsf{LinQuant}$ with $f \models f'$, we call $g \in \mathsf{LinQuant}$ a *Craig interpolant* of (f, f'), if

1. g is quantifier-free,
2. $f \models g \models f'$, and
3. $\mathsf{FV}(g) \subseteq \mathsf{FV}(f) \cap \mathsf{FV}(f')$.

Theorem

Whenever $f \models f'$ there ***exists an effectively constructible*** *Craig interpolant g of (f, f').*

Proof.

Using quantitative quantifier elimination. □

Quantitative Craig Interpolation

$$f \models \underbrace{\mathsf{QE}(\underbrace{\exists x_1 \ldots \exists x_n}_{=\mathsf{FV}(f)\setminus\mathsf{FV}(f')} : f)}_{\textit{strongest}\text{ Craig interpolant}} \models \underbrace{\mathsf{QE}(\underbrace{\forall y_1 \ldots \forall y_m}_{=\mathsf{FV}(f')\setminus\mathsf{FV}(f)} : f')}_{\textit{weakest}\text{ Craig interpolant}} \models f'$$

Quantitative Quantifier Elimination & Craig Interpolation

1. Soundness and termination
2. Upper space complexity bound
3. First Craig interpolation theorem for LinQuant

Quantitative Quantifier Elimination & Craig Interpolation

1. Soundness and termination
2. Upper space complexity bound
3. First Craig interpolation theorem for LinQuant

Future Work:

- Computing **more concise** Craig interpolants, efficiently

Quantitative Quantifier Elimination & Craig Interpolation

1. Soundness and termination
2. Upper space complexity bound
3. First Craig interpolation theorem for LinQuant

Future Work:

- Computing **more concise** Craig interpolants, efficiently
- **Quantitative invariant synthesis:**
  - Craig interpolation-based [McMillan '03, '06]
  - Via abduction using quantifier elimination [Dillig, Li, McMillan '13]

Quantitative Quantifier Elimination & Craig Interpolation

1. Soundness and termination
2. Upper space complexity bound
3. First Craig interpolation theorem for LinQuant

Future Work:

- Computing **more concise** Craig interpolants, efficiently
- **Quantitative invariant synthesis:**
  - Craig interpolation-based [McMillan '03, '06]
  - Via abduction using quantifier elimination [Dillig, Li, McMillan '13]

Thank you!

Backup

Maybe put construction of max