



Quantifier Elimination and Craig Interpolation: The Quantitative Way^{*}

Kevin Batz^{1,2} , Joost-Pieter Katoen¹ , and Nora Orhan¹

¹ RWTH Aachen University, Aachen, Germany
`{kevin.batz,katoen}@cs.rwth-aachen.de`, `nora.hiseni@rwth-aachen.de`
² University College London, London, UK

Abstract. Quantifier elimination (QE) and Craig interpolation (CI) are central to various state-of-the-art automated approaches to hardware- and software verification. They are rooted in the Boolean setting and are successful for, e.g., first-order theories such as linear rational arithmetic. What about their applicability in the quantitative setting where formulae evaluate to numbers and quantitative supremum/infimum quantifiers are the natural pendant to traditional Boolean quantifiers? Applications include establishing quantitative properties of programs such as bounds on expected outcomes of probabilistic programs featuring unbounded non-determinism and analyzing the flow of information through programs. In this paper, we present the — to the best of our knowledge — first QE algorithm for possibly unbounded, ∞ - or $(-\infty)$ -valued, or discontinuous *piecewise linear quantities*. They are the quantitative counterpart to linear rational arithmetic, and are a popular quantitative assertion language for probabilistic program verification. We provide rigorous soundness proofs as well as upper space complexity bounds. Moreover, our algorithm yields a quantitative CI theorem: Given arbitrary piecewise linear quantities f, g with $f \models g$, both the strongest and the weakest Craig interpolant of f and g are quantifier-free and effectively constructible.

Keywords: Quantifier Elimination · Craig Interpolation · Quantitative Program Verification

1 Introduction

Quantifier elimination algorithms take as input a first-order formula φ over some background theory $\mathcal{T}$ and output a quantifier-free formula $\text{QE}(\varphi)$ equivalent to φ modulo $\mathcal{T}$. Prime examples include Fourier-Motzkin variable elimination [45,25] and virtual substitution [40] for linear rational arithmetic, Cooper’s method [18] for linear integer arithmetic, and Cylindrical Algebraic Decomposition [16] for non-linear real arithmetic. Quantifier elimination is extensively leveraged by automatic hard- and software verification techniques for, e.g., computing images

^{*} This research was supported by the ERC AdG project 787914 FRAPPANT.

of state sets [35,36], or for synthesizing loop invariants either from templates [27,17,31] or by solving abduction problems [22].

Craig interpolation [19] is vital to various automatic hard- and software verification techniques. A first-order theory $\mathcal{T}$ is called (*quantifier-free*) *interpolating* [32], if for all formulae φ, ψ with $\varphi \models_{\mathcal{T}} \psi$, there is an effectively constructible (quantifier-free) formula ϑ — called *Craig interpolant* of (φ, ψ) — with $\varphi \models_{\mathcal{T}} \vartheta \models_{\mathcal{T}} \psi$ and such that all free variables occurring free in ϑ also occur free in both φ and ψ . Intuitively, $\varphi \models_{\mathcal{T}} \psi$ encodes some (desirable or undesirable) reachability information and ϑ is a concise explanation of this fact, abstracting away irrelevant details. The computation of (quantifier-free) Craig interpolants is a vivid area of research [1,26,15,57,13] with applications ranging from symbolic finite-state model checking [43,56,39] over computing transition power abstractions [12] to automatic infinite-state software verification [28,11,29,52,53,44,2].

Quantitative program verification includes reasoning about expected outcomes of probabilistic programs via *weakest pre-expectations* [37,38,42,30], reasoning about the quantitative flow of information via *quantitative strongest post* [59], and reasoning about competitive ratios of online algorithms via *weighted programming* [7]. Quantitative reasoning requires a shift: Predicates, i.e., maps from program states to truth values in $\{\text{true}, \text{false}\}$, are replaced by *quantities*³, which map program states to *extended reals* in $\mathbb{R} \cup \{-\infty, \infty\}$.

The classical quantifiers “there exists” $\exists$ and “for all” $\forall$ from predicate logic are replaced by *quantitative* supremum $\mathcal{Z}$ and infimum $\mathcal{L}$ quantifiers [8]. These quantifiers naturally occur when reasoning with quantitative program logics: Very much like *classical* strongest post-conditions introduce an $\exists$ -quantifier for assignments [21], *quantitative* strongest post introduces a $\mathcal{Z}$ -quantifier (cf. [59, Table 2]). Similarly, whereas *classical* weakest pre-conditions introduce a $\forall$ -quantifier for demonically resolving unbounded non-determinism of the form $x := \mathbb{Q}$ (read: assign to x an *arbitrary* rational number) [20,47], *quantitative* weakest pre-expectations introduce an $\mathcal{L}$ -quantifier [9,51].

Example 1. In this paper, we focus on *piecewise linear quantities* such as

$$g = \underbrace{[y_1 \geq z \longrightarrow (x - 2 < y_1 \wedge -x \geq y_3 \wedge x \geq y_2)]}_{=\varphi} \cdot \underbrace{(2 \cdot x + z)}_{=\tilde{a}},$$

evaluate to $\tilde{a}$ on variable valuation σ if $\sigma \models \varphi$, and to 0 otherwise

where $x, y_1, \dots$ are $\mathbb{Q}$ -valued variables. We can think of g as a formula that evaluates to extended rationals from $\mathbb{Q} \cup \{-\infty, \infty\}$ instead of truth values. By prefixing g with, e.g., a supremum quantifier, we obtain a new piecewise linear quantity $\mathcal{Z}x: g$, which, on variable valuation σ , evaluates to the (variable valuation-dependent) supremum of g under all possible values for x , i.e.,

$$\sigma[\mathcal{Z}x: g] = \sup \left\{ \sigma[x \mapsto q][g] \mid q \in \mathbb{Q} \right\}. \quad \triangle$$

³ We adopt this term from Zhang and Kaminski [59]. In the realm of weakest pre-expectations, quantitative assertions are usually referred to as *expectations* [42]. In weighted programming, they are called *weightings* [7].

Our Contribution: Quantitative Quantifier Elimination and Craig Interpolation. Piecewise linear quantities over $\mathbb{Q}$ -valued variables are to quantitative probabilistic program verification what first-order linear rational arithmetic is to classical program verification: Their entailment problem, i.e.,

$$\begin{array}{ccc} \text{Given piecewise linear quantities } f \text{ and } f', & & \\ \text{does } \underbrace{f \models f'} & & \text{hold?} \\ \text{for all variable valuations } \sigma: \sigma[f] \leq \sigma[f'] & & \end{array}$$

is decidable [33,6], they are effectively closed under, e.g., weakest pre-expectations of loop-free linear probabilistic programs [6], and they have been shown to be sufficiently expressive for the verification of various probabilistic programs [5,6,58,49,14]. These facts render piecewise linear quantities one of the most prevalent quantitative assertion languages in automatic reasoning.

Reasoning with piecewise linear quantities containing the quantitative $\mathfrak{Q}$ or $\mathcal{L}$ quantifiers has, however, received scarce attention, let alone algorithmically. Similarly, the field of *quantitative Craig interpolation* is rather unexplored, as well. The goal of this paper is to lay the foundations for (i) developing quantitative quantifier elimination- and Craig interpolation-based approaches to automatic quantitative program verification and (ii) for simplifying the reasoning with quantitative assertions involving quantitative quantifiers. Towards this end:

1. We contribute the — to the best of our knowledge — first quantitative quantifier elimination algorithm for *arbitrary, possibly unbounded, ∞ or $-(\infty)$ -valued, or discontinuous* piecewise linear quantities. Put more formally, given an *arbitrary* piecewise linear quantity f possibly containing quantitative quantifiers, our algorithm computes a *quantifier-free equivalent* of f . For $\mathfrak{Q}x: g$ from Example 1, our algorithm yields (after simplification)

$$\begin{aligned} & [y_1 < z] \cdot \infty \\ & + [y_1 \geq z \wedge y_2 < y_1 + 2 \wedge y_2 \leq -y_3 \wedge y_1 + 2 \leq y_3] \cdot (2 \cdot y_1 + z + 4) \\ & + [y_1 \geq z \wedge y_2 < y_1 + 2 \wedge y_2 \leq -y_3 \wedge y_1 + 2 > y_3] \cdot (-y_3 + z) . \end{aligned}$$

2. We provide rigorous soundness proofs, illustrative examples, and upper space-complexity bounds on our algorithm.
3. We contribute the — to the best of our knowledge — first *Craig interpolation theorem* for piecewise linear quantities: Using our quantifier elimination algorithm, we prove that for two *arbitrary* piecewise linear quantities f, f' with $f \models f'$, both the strongest and the weakest g such that

$$f \models g \models f' \quad \text{and} \quad \text{the free variables in } g \text{ are free in both } f \text{ and } f'$$

are quantifier-free and effectively constructible.

Example 2. Consider the following piecewise linear quantities:

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

$$f' = [x \geq 0 \wedge z \geq x] \cdot (2 \cdot x + z + 1) + [z < x] \cdot \infty$$

We have $f \models f'$. Using our quantifier elimination technique, we effectively construct both the strongest and the weakest⁴ Craig interpolants of (f, f') given by

$$\underbrace{[x \geq 0] \cdot 2 \cdot x}_{\text{strongest Craig interpolant of } (f, f')} \quad \text{and} \quad \underbrace{[x \geq 0] \cdot (3 \cdot x + 1)}_{\text{weakest Craig interpolant of } (f, f')} \quad . \quad \triangle$$

Remark 1. When applying *classical* Craig interpolation for a first-order theory $\mathcal{T}$ to, e.g., loop invariant generation, “simple” Craig interpolants, i.e., interpolants that lie strictly “between” (w.r.t. $\models_{\mathcal{T}}$) the strongest and the weakest ones, are often very useful [1]. Our *quantitative* Craig interpolation technique presented in this paper does *not* aim for obtaining such “simple” interpolants. Rather, our goal is to prove that quantitative Craig interpolants at all exist and that they are effectively constructible. We discuss possible directions for obtaining simpler quantitative Craig interpolants in Section 5.

Related Work. Our quantifier elimination algorithm is based on ideas related to Fourier-Motzkin variable elimination [45,25]. Most closely related is the work by Zamani, Sanner, and Fang on symbolic dynamic programming [50]. They introduce the symbolic $\max_x$ operator on piecewise defined functions of type $\mathbb{R}^n \rightarrow \mathbb{R}$, which also exploits the partitioning property (similar to Theorem 1) and disjunctive normal forms (similar to Theorem 2). We identify the following key differences: the functions considered in [50] must be (i) continuous, (ii) bounded (so that all suprema are actually maxima), and (iii) they must not contain ∞ or $-\infty$. We do not impose these restrictions since they do not apply to piecewise defined functions obtained from, e.g., applying the program logics mentioned in Section 1. [50], on the other hand, also considers piecewise quadratic functions, whereas we focus on piecewise linear functions. Extending our approach to piecewise quadratic functions is an interesting direction for future work. Finally, we provide a rigorous formalization and soundness proofs alongside upper space complexity bounds. Quantitative quantifier elimination is moreover related to *parametric programming* [54]. We are, however, not aware of an approach which tackles the computational problem we investigate as it is required from the perspective of a quantitative quantifier elimination problem.

Khatami, Pourmahdian, and Tavana [55] investigate a Craig interpolation property of first-order Gödel logic, where formulae evaluate to real numbers in the unit interval $[0, 1]$. Apart from the more restrictive semantic codomain, [55] operates in an uninterpreted setting whereas we operate within linear rational arithmetic extended by ∞ and $-\infty$. Baaz and Veith [4] investigate quantifier elimination of first-order logic over fuzzy algebras over the same semantic codomain. Teige and Fränzle [41] investigate Craig interpolation for stochastic Boolean satisfiability problems, where formulae also evaluate to numbers instead of truth values. Quantified variables are assumed to range over a finite domain.

⁴ We say that g is *stronger* (resp. *weaker*) than g' , if $g \models g'$ (resp. $g' \models g$).

Outline. In Section 2, we introduce piecewise linear quantities. We present our quantifier elimination algorithm alongside essential theorems and a complexity analysis in Section 3. Our quantitative Craig interpolation results are presented in Section 4. Finally, we discuss potential applications in Section 5.

2 Piecewise Linear Quantities

Throughout this paper, we fix a finite non-empty set $\mathbf{Vars} = \{x, y, z, \dots\}$ of variables. We denote by $\mathbb{N}_0$ the set of natural numbers including 0 and let $\mathbb{N} = \mathbb{N}_0 \setminus 0$. The set of rationals (resp. reals) is denoted by $\mathbb{Q}$ (resp. $\mathbb{R}$) and we denote by $\mathbb{Q}^{\pm\infty} = \mathbb{Q} \cup \{-\infty, \infty\}$ (resp. $\mathbb{R}^{\pm\infty} = \mathbb{R} \cup \{-\infty, \infty\}$) the set of *extended* rationals (resp. reals). A *(variable) valuation* $\sigma: \mathbf{Vars} \rightarrow \mathbb{Q}$ assigns a rational number to each variable. The set of all valuations is denoted by Σ .

Towards defining piecewise linear quantities and their semantics, we briefly recap linear arithmetic and Boolean expressions.

Definition 1 (Linear Arithmetic Expressions). *The set LinAX of linear arithmetic expressions consists of all expressions a of the form*

$$a = q_0 + \sum_{i=1}^{|\mathbf{Vars}|} q_i \cdot x_i ,$$

where $q_0, \dots, q_{|\mathbf{Vars}|} \in \mathbb{Q}$ and $x_1, \dots, x_{|\mathbf{Vars}|} \in \mathbf{Vars}$. Moreover, we define the set $\text{LinAX}^{\pm\infty}$ of extended linear arithmetic expressions as

$$\text{LinAX}^{\pm\infty} = \text{LinAX} \cup \{-\infty, \infty\} . \quad \triangle$$

Notice that every arithmetic expression is normalized in the sense that every variable occurs exactly once. We often omit summands $q_i \cdot x_i$ (resp. q_0) with $q_i = 0$ (resp. $q_0 = 0$) for the sake of readability. Given a as above, we denote by

$$\text{FV}(a) = \{x_i \in \mathbf{Vars} \mid q_i \neq 0\}$$

the set of (necessarily free) variables occurring in a . For $\tilde{a} = \infty$ or $\tilde{a} = -\infty$, we define $\text{FV}(\tilde{a}) = \emptyset$. Given $\tilde{a} \in \text{LinAX}^{\pm\infty}$ and $x_j \in \mathbf{Vars}$, we say that

$$\begin{cases} x_j \text{ occurs positively in } \tilde{a} & \text{if } \tilde{a} = q_0 + \sum_{i=1}^{|\mathbf{Vars}|} q_i \cdot x_i \text{ and } q_j > 0 \\ x_j \text{ occurs negatively in } \tilde{a} & \text{if } \tilde{a} = q_0 + \sum_{i=1}^{|\mathbf{Vars}|} q_i \cdot x_i \text{ and } q_j < 0 . \end{cases}$$

Finally, given $\sigma \in \Sigma$, the *semantics* $\sigma[\tilde{a}] \in \mathbb{Q}^{\pm\infty}$ of $\tilde{a}$ under σ is defined as

$$\sigma[\tilde{a}] = \begin{cases} q_0 + \sum_{i=1}^{|\mathbf{Vars}|} q_i \cdot \sigma(x_i) & \text{if } \tilde{a} = q_0 + \sum_{i=1}^{|\mathbf{Vars}|} q_i \cdot x_i \\ -\infty & \text{if } \tilde{a} = -\infty \\ \infty & \text{if } \tilde{a} = \infty . \end{cases}$$

Definition 2 (Boolean Expressions). Boolean expressions φ in the set **Bool** adhere to the following grammar, where $\tilde{a} \in \text{LinAX}^{\pm\infty}$:

$$\begin{array}{ll} \varphi & \longrightarrow \quad \tilde{a} < \tilde{a} \mid \tilde{a} \leq \tilde{a} \mid \tilde{a} > \tilde{a} \mid \tilde{a} \geq \tilde{a} & \text{(linear inequalities)} \\ & \mid \neg\varphi & \text{(negation)} \\ & \mid \varphi \wedge \varphi & \text{(conjunction) } \triangle \end{array}$$

The Boolean constants **true**, **false** and the Boolean connectives $\vee$ and $\longrightarrow$ are syntactic sugar. We assume that $\neg$ binds stronger than $\wedge$ binds stronger than $\vee$, and we use parentheses to resolve ambiguities. The set $\text{FV}(\varphi)$ of (necessarily free) variables in φ is defined as usual. Given a valuation σ , we write $\sigma \models \varphi$ if σ satisfies φ and $\sigma \not\models \varphi$ otherwise, which is defined in the standard way. Finally, if $\sigma \not\models \varphi$ for all $\sigma \in \Sigma$, then we say that φ is *unsatisfiable*⁵.

Definition 3 (Piecewise Linear Quantities (adapted from [8,33])). The set **LinQuant** of (piecewise linear) quantities consists of all expressions

$$f = Q_1 x_1 \dots Q_k x_k : \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i ,$$

where $k \in \mathbb{N}_0$, $n \in \mathbb{N}$, and where

1. $Q_j \in \{\mathfrak{Z}, \mathfrak{L}\}$ and $x_j \in \text{Vars}$ for all $j \in \{1, \dots, k\}$,
2. $\varphi_i \in \text{Bool}$ and $\tilde{a}_i \in \text{LinAX}^{\pm\infty}$ for all $i \in \{1, \dots, n\}$,
3. for all $\sigma \in \Sigma$ and all $i, j \in \{1, \dots, n\}$ with $i \neq j$, we have⁶

$$\sigma \models \varphi_i \text{ and } \sigma \models \varphi_j \quad \text{implies} \quad \tilde{a}_i \neq \infty \text{ or } \tilde{a}_j \neq -\infty . \quad \triangle$$

Here $[\varphi]$ is the *Iverson bracket* [34] of the Boolean expression φ , which evaluates to 1 under valuation σ if $\sigma \models \varphi$, and to 0 otherwise. $\mathfrak{Z}$ is called the *supremum quantifier* and $\mathfrak{L}$ is the *infimum quantifier*. The quantitative quantifiers take over the role of the classical $\exists$ - and $\forall$ -quantifiers from first-order predicate logic. Their semantics is detailed further below. If $k = 0$, then we call f *quantifier-free*. Given f as above, the set of *free variables* in f is

$$\text{FV}(f) = \bigcup_{j=1}^n (\text{FV}(\varphi_j) \cup \text{FV}(\tilde{a}_j)) \setminus \{x_1, \dots, x_k\} .$$

For quantifier-free f , we introduce the shorthand $[\varphi] \cdot f = \sum_{i=1}^n [\varphi \wedge \varphi_i] \cdot \tilde{a}_i$.

Towards defining the semantics of quantities, we use the following notions: Given a valuation $\sigma \in \Sigma$, a variable $x \in \text{Vars}$, and $q \in \mathbb{Q}$, we define the valuation obtained from updating the value of x under σ to q as

$$\sigma[x \mapsto q](y) = \begin{cases} q & \text{if } y = x \\ \sigma(y) & \text{otherwise} . \end{cases}$$

As is standard [3] in the realm of extended reals, we define for all $r \in \mathbb{R}$:

⁵ Unsatisfiability of Boolean expressions is decidable by SMT solving over linear rational arithmetic (LRA) as is implemented, e.g., by the solver Z3 [46].

⁶ This is decidable by SMT solving over LRA. Hence, the set **LinQuant** is computable.

1. $r + \infty = \infty + r = \infty$
2. $r + (-\infty) = -\infty + r = -\infty$
3. $\infty + \infty = \infty$
4. $-\infty + (-\infty) = -\infty$
5. $-\infty \cdot 0 = 0 \cdot (-\infty) = 0 = 0 \cdot \infty = \infty \cdot 0$
6. if $r > 0$, then $r \cdot \infty = \infty \cdot r = \infty$
7. if $r > 0$, then $r \cdot (-\infty) = -\infty \cdot r = -\infty$
8. if $r < 0$, then $r \cdot \infty = \infty \cdot r = -\infty$
9. if $r < 0$, then $r \cdot (-\infty) = -\infty \cdot r = \infty$

The terms $\infty + (-\infty)$ and $-\infty + \infty$ are undefined. The condition from Definition 3.3 ensures that we never encounter such undefined terms, which yields the semantics of piecewise linear quantities to be well-defined:

Definition 4 (Semantics of Piecewise Linear Quantities). *Let $f \in \text{LinQuant}$ and $\sigma \in \Sigma$. The semantics⁷ $\sigma \llbracket f \rrbracket \in \mathbb{R}^{\pm\infty}$ of f under σ is defined inductively:*

$$\begin{aligned}
 \sigma \llbracket \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i \rrbracket &= \sum_{i=1}^n \begin{cases} \sigma \llbracket \tilde{a}_i \rrbracket & \text{if } \sigma \models \varphi_i \\ 0 & \text{if } \sigma \not\models \varphi_i \end{cases} \\
 \sigma \llbracket \mathcal{Z}x : f' \rrbracket &= \sup \left\{ \sigma[x \mapsto q] \llbracket f' \rrbracket \mid q \in \mathbb{Q} \right\} \\
 \sigma \llbracket \mathcal{L}x : f' \rrbracket &= \inf \left\{ \sigma[x \mapsto q] \llbracket f' \rrbracket \mid q \in \mathbb{Q} \right\} \quad \triangle
 \end{aligned}$$

In words: if f is quantifier-free, then $\sigma \llbracket f \rrbracket$ evaluates to the sum of all extended arithmetic expressions $\tilde{a}_j$ for which $\sigma \models \varphi_j$. The semantics of $\mathcal{Z}$ and $\mathcal{L}$ makes it evident that the quantitative quantifiers generalize the classical quantifiers: Whereas $\exists$ maximizes — so to speak — a truth value, the $\mathcal{Z}$ -quantifier maximizes a quantity by evaluating to the supremum obtained from evaluating f under all possible values for x . $\mathcal{L}$ behaves analogously by evaluating to an infimum.

Finally, we say that two piecewise linear quantities $f, f' \in \text{LinQuant}$ are (*semantically*) *equivalent*, denoted by $f \equiv f'$, if $\sigma \llbracket f \rrbracket = \sigma \llbracket f' \rrbracket$ for all $\sigma \in \Sigma$.

3 Quantitative Quantifier Elimination

In this section, we detail our quantifier elimination procedure alongside illustrative examples. Given an *arbitrary* piecewise linear quantity

$$f = Q_1 x_1 \dots Q_k x_k : \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i \in \text{LinQuant} ,$$

we aim to automatically compute some $\text{QE}(f) \in \text{LinQuant}$ satisfying

$$\text{QE}(f) \text{ is quantifier-free} \quad \text{and} \quad \text{QE}(f) \text{ is equivalent to } f, \text{ i.e., } f \equiv \text{QE}(f) .$$

⁷ It follows from the soundness of our quantifier elimination algorithm (Theorem 5) that all $f \in \text{LinQuant}$ evaluate to extended rationals in $\mathbb{Q}^{\pm\infty}$.

As with quantifier elimination for (theories of) classical first-order predicate logic, it suffices being able to eliminate piecewise linear quantities containing *a single* quantifier, which then enables to process quantities containing an *arbitrary* number of quantifiers in an inner- to outermost fashion, i.e.,

$$\text{QE}(f) = \text{QE}(Q_1 x_1 : \text{QE}(\dots \text{QE}(Q_k x_k : \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i))) .$$

Throughout the next sections, we thus fix an f of the form

$$f = Qx : \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i , \quad (1)$$

where $Q \in \{\mathcal{E}, \mathcal{U}\}$. We proceed by means of a 3-level divide-and-conquer approach. We describe each of the involved stages in Sections 3.1-3.3. In Section 3.4, we summarize our approach by providing an algorithm.

3.1 Stage 1: Exploiting the Guarded Normal Form

Our first step is to transform the input f into a normal form (an extension of [33, Section 5.1]), which enables us to subdivide the quantifier elimination problem into simpler sub-problems. Intuitively, this normal form enforces a more explicit representation of the $\mathbb{R}^{\pm\infty}$ -valued function a piecewise linear quantity represents.

Definition 5 (Guarded Normal Form). *Let $g \in \text{LinQuant}$ be given by*

$$g = Q_1 x_1 \dots Q_k x_k : \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i$$

and fix some variable $x \in \text{Vars}$. We say that g is in guarded normal form w.r.t. x , denoted by $g \in \text{GNF}_x$, if all of the following conditions hold:

1. *the φ_i partition the set Σ of valuations, i.e., for all $\sigma \in \Sigma$ there exists exactly one $i \in \{1, \dots, n\}$ such that $\sigma \models \varphi_i$,*
2. *the φ_i are in disjunctive normal form (DNF), i.e.,*

$$\forall i \in \{1, \dots, n\}: \quad \varphi_i \text{ is of the form } \bigvee_j \bigwedge_{j'} L_{j,j'} ,$$

- where each $L_{j,j'} \in \text{LinAX}^{\pm\infty}$ is a (strict or non-strict) linear inequality,*
3. *for each linear inequality L in g , it holds that if $x \in \text{FV}(L)$, then*

$$L = x \sim \tilde{b}$$

for some $\tilde{b} \in \text{LinAX}^{\pm\infty}$ with $x \notin \text{FV}(\tilde{b})$ and $\sim \in \{>, \geq, <, \leq\}$. $\triangle$

If Condition 5.1 holds, then we say that g is *partitioning*. Speaking of a *normal form* is justified by the fact that every piecewise linear quantity $g \in \text{LinQuant}$ can effectively be transformed into a semantically equivalent $g' \in \text{LinQuant}$ in guarded normal form with respect to variable $x \in \text{Vars}$, i.e., such that $g' \in \text{GNF}_x$ and $g \equiv g'$. To see this, let g be given as above. Towards obtaining g' , we first establish the partitioning property by enumerating the possible assignments of truth values to the φ_i . Put more formally, we construct

$$\sum_{((\rho_1, \tilde{e}_1), \dots, (\rho_n, \tilde{e}_n)) \in \times_{i=1}^n \{(\varphi_i, \tilde{a}_i), (\neg \varphi_i, 0)\}} \begin{cases} \epsilon & \text{if } \bigwedge_{i=1}^n \rho_i \text{ is unsat.} \\ [\bigwedge_{i=1}^n \rho_i] \cdot \sum_{i=1}^n \tilde{e}_i & \text{otherwise,} \end{cases}$$

where we let $\epsilon + \tilde{e} = \tilde{e} = \tilde{e} + \epsilon$ for all $\tilde{e} \in \text{LinAX}^{\pm\infty}$ and obey the rules for treating ∞ and $-\infty$, respectively, as given in Section 2. We then obtain g' by transforming the so-obtained Boolean expressions into DNF and isolating x in every inequality where x occurs. Notice that if g satisfies the conditions from Definition 3, then so does g' . In particular, when constructing sums of the form $\sum_{i=1}^n \tilde{e}_i$, we never encounter expressions of the form $\infty + (-\infty)$ or $-\infty + \infty$.

Example 3. Recall the piecewise linear quantity from Example 1 given by

$$\mathfrak{Z}x: [y_1 \geq z \longrightarrow (x - 2 < y_1 \wedge -x \geq y_3 \wedge x \geq y_2)] \cdot (2 \cdot x + z),$$

which is *not* in GNF_x . Applying the construction from above yields

$$\begin{aligned} \mathfrak{Z}x: & [y_1 < z \vee (x < y_1 + 2 \wedge x \leq -y_3 \wedge x \geq y_2)] \cdot (2 \cdot x + z) \\ & + [(y_1 \geq z \wedge x \geq y_1 + 2) \vee (y_1 \geq z \wedge x > -y_3) \vee (y_1 \geq z \wedge x < y_2)] \cdot 0 \end{aligned}$$

which *is* in GNF_x and will serve us as a running example. $\triangle$

Now assume w.l.o.g. that the input quantity f is in GNF_x . Each of the Conditions 5.1-3 is essential to our approach. We will now exploit that f is partitioning. Given $\varphi \in \text{Bool}$ and $\tilde{a} \in \text{LinAX}^{\pm\infty}$, we define the shorthands

$$\varphi \searrow \tilde{a} = [\varphi] \cdot \tilde{a} + [\neg \varphi] \cdot (-\infty) \quad \text{and} \quad \varphi \nearrow \tilde{a} = [\varphi] \cdot \tilde{a} + [\neg \varphi] \cdot \infty.$$

Notice that these quantities are always partitioning. Now consider the following:

Theorem 1. *Let $x \in \text{Vars}$ and let $\sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i \in \text{GNF}_x$. We have for all $\sigma \in \Sigma$:*

1. $\sigma \llbracket \mathfrak{Z}x: \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i \rrbracket = \max \{ \sigma \llbracket \mathfrak{Z}x: (\varphi_i \searrow \tilde{a}_i) \rrbracket \mid i \in \{1, \dots, n\} \}$
2. $\sigma \llbracket \mathfrak{L}x: \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i \rrbracket = \min \{ \sigma \llbracket \mathfrak{L}x: (\varphi_i \nearrow \tilde{a}_i) \rrbracket \mid i \in \{1, \dots, n\} \}$

Proof. This is a consequence of the fact that the quantity is partitioning and that $-\infty$ (resp. ∞) are neutral wr.t. max (resp. min). See [10, App. A.1] for details. $\square$

We may thus consider each summand of the input quantity f separately. Assuming we can compute $\text{QE}(\mathcal{Q}x: \varphi_i \searrow \tilde{a}_i)$ and $\text{QE}(\mathcal{L}x: \varphi_i \nearrow \tilde{a}_i)$, we obtain the sought-after quantifier-free equivalent of f by effectively constructing valuation-wise minima and maxima of finite sets of partitioning quantities as follows:

Lemma 1. *Let $M = \{h_1, \dots, h_n\} \subseteq \text{LinQuant}$ for some $n \geq 1$, where each*

$$h_i = \sum_{j=1}^{m_i} [\varphi_{i,j}] \cdot \tilde{a}_{i,j}$$

is partitioning. Then:

1. *There is an effectively constructible $\text{MAX}(M) \in \text{LinQuant}$ such that*

$$\forall \sigma \in \Sigma: \quad \sigma \llbracket \text{MAX}(M) \rrbracket = \max \{ \sigma \llbracket h_i \rrbracket \mid i \in \{1, \dots, n\} \}.$$

2. *There is an effectively constructible $\text{MIN}(M) \in \text{LinQuant}$ such that*

$$\forall \sigma \in \Sigma: \quad \sigma \llbracket \text{MIN}(M) \rrbracket = \min \{ \sigma \llbracket h_i \rrbracket \mid i \in \{1, \dots, n\} \}.$$

Moreover, both $\text{MAX}(M)$ and $\text{MIN}(M)$ are partitioning.

Proof. Write $\underline{m}_i = \{1, \dots, m_i\}$. We construct⁸

$$\begin{aligned} \text{MAX}(M) = & \sum_{(j_1, \dots, j_n) \in \underline{m}_1 \times \dots \times \underline{m}_n} \sum_{i=1}^n \\ & \left[\underbrace{\bigwedge_{k=1}^n \varphi_{k,j_k}}_{h_k \text{ evaluates to } \tilde{a}_{k,j_k}} \wedge \underbrace{\bigwedge_{k=1}^{i-1} \tilde{a}_{i,j_i} > \tilde{a}_{k,j_k} \wedge \bigwedge_{k=i+1}^n \tilde{a}_{i,j_i} \geq \tilde{a}_{k,j_k}}_{h_i \text{ is the quantity with smallest index evaluating to the sought-after maximum}} \right] \cdot \tilde{a}_{i,j_i}. \end{aligned}$$

$\text{MAX}(M)$ iterates over all combinations of summands, which determine the value each of the h_i evaluate to (first summand). For each of these combinations, we check, for each $i \in \{1, \dots, n\}$, whether h_i evaluates to the sought-after maximum (second summand). $\text{MAX}(M)$ is partitioning since the h_i are and due to the fact that $\text{MAX}(M)$ selects the maximizing quantity with the *smallest index*. The construction of $\text{MIN}(M)$ is analogous and provided in [10, App. B.1]. $\square$

Combining Theorem 1 and Lemma 1 thus yields:

1. $\text{QE}(\mathcal{Q}x: \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i) = \text{MAX}(\{ \text{QE}(\mathcal{Q}x: (\varphi_i \searrow \tilde{a}_i)) \mid i \in \{1, \dots, n\} \})$
2. $\text{QE}(\mathcal{L}x: \sum_{i=1}^n [\varphi_i] \cdot \tilde{a}_i) = \text{MIN}(\{ \text{QE}(\mathcal{L}x: (\varphi_i \nearrow \tilde{a}_i)) \mid i \in \{1, \dots, n\} \})$

Example 4. Continuing Example 3, we have for every $\sigma \in \Sigma$,

$$\begin{aligned} \text{QE}(f) = & \text{MAX}(\{ \text{QE}(\mathcal{Q}x: (y_1 < z \vee (x < y_1 + 2 \wedge x \geq y_2)) \searrow 2 \cdot x + z), \\ & \text{QE}(\mathcal{Q}x: ((y_1 \geq z \wedge x \geq y_1 + 2) \vee \dots) \searrow 0) \}). \quad \triangle \end{aligned}$$

⁸ As usual, the empty conjunction is equivalent to **true**.

3.2 Stage 2: Exploiting the Disjunctive Normal Form

In this stage, we aim to eliminate the quantifiers from the simpler quantities

$$\mathcal{E}x: (\varphi \searrow \tilde{a}) \quad \text{or} \quad \mathcal{L}x: (\varphi \nearrow \tilde{a}) .$$

Recall that we assume the input quantity f to be in guarded normal form w.r.t. x , which yields the Boolean expression φ to be in DNF (cf. Definition 5.2). Exploiting the disjunctive shape of φ yields the following:

Theorem 2. *Let $\tilde{a} \in \text{LinAX}^{\pm\infty}$ be an extended arithmetic expression and let*

$$\varphi = \bigvee_{i=1}^n D_i \in \text{Bool}$$

be a Boolean expression in DNF for some $n \geq 1$. We have for all $\sigma \in \Sigma$:

1. $\sigma \llbracket \mathcal{E}x: (\varphi \searrow \tilde{a}) \rrbracket = \max \{ \sigma \llbracket \mathcal{E}x: (D_i \searrow \tilde{a}) \rrbracket \mid i \in \{1, \dots, n\} \}$
2. $\sigma \llbracket \mathcal{L}x: (\varphi \nearrow \tilde{a}) \rrbracket = \min \{ \sigma \llbracket \mathcal{L}x: (D_i \nearrow \tilde{a}) \rrbracket \mid i \in \{1, \dots, n\} \}$

Proof. We first observe that

$$\sigma \llbracket \mathcal{E}x: (\varphi \searrow \tilde{a}) \rrbracket = \sup \left(\bigcup_{i=1}^n \left\{ \sigma[x \mapsto q] \llbracket [D_i] \cdot \tilde{a} \rrbracket \mid q \in \mathbb{Q} \text{ and } \sigma[x \mapsto q] \models D_i \right\} \right)$$

and then make use of the fact that the supremum of a finite union of extended reals is the maximum of the individual suprema, i.e., the above is equal to

$$\begin{aligned} & \max \left(\bigcup_{i=1}^n \left\{ \sup \left\{ \sigma[x \mapsto q] \llbracket [D_i] \searrow \tilde{a} \rrbracket \mid q \in \mathbb{Q} \right\} \right\} \right) \quad (-\infty \text{ is neutral w.r.t. sup}) \\ &= \max \left(\bigcup_{i=1}^n \left\{ \sigma \llbracket \mathcal{E}x: (D_i \searrow \tilde{a}) \rrbracket \right\} \right) \quad (\text{Definition 4}) \\ &= \max \{ \sigma \llbracket \mathcal{E}x: (D_i \searrow \tilde{a}) \rrbracket \mid i \in \{1, \dots, n\} \} . \quad (\text{rewrite set}) \end{aligned}$$

See [10, App. A.2] for a detailed proof. The reasoning for $\mathcal{L}$ is analogous. $\square$

Hence, combining Theorem 2 with Lemma 1 reduces our problem further to eliminating quantifiers from the above simpler quantities. Put formally:

1. $\text{QE}(\mathcal{E}x: (\varphi \searrow \tilde{a})) = \text{MAX}(\{ \text{QE}(\mathcal{E}x: (D_i \searrow \tilde{a})) \mid i \in \{1, \dots, n\} \})$
2. $\text{QE}(\mathcal{L}x: (\varphi \nearrow \tilde{a})) = \text{MIN}(\{ \text{QE}(\mathcal{L}x: (D_i \nearrow \tilde{a})) \mid i \in \{1, \dots, n\} \})$

Example 5. Continuing Example 4, we have

$$\begin{aligned} & \text{QE}(\mathcal{E}x: (y_1 < z \vee (x < y_1 + 2 \wedge x \geq y_2)) \searrow 2 \cdot x + z) \\ &= \text{MAX}(\{ \text{QE}(\mathcal{E}x: y_1 < z \searrow 2 \cdot x + z), \\ & \quad \text{QE}(\mathcal{E}x: (x < y_1 + 2 \wedge x \geq y_2) \searrow 2 \cdot x + z) \}) . \end{aligned}$$

The second argument of MAX from Example 4 is treated analogously. $\triangle$

3.3 Stage 3: Computing Valuation-Dependent Suprema and Infima

This is the most involved stage since we need to operate on the atomic level of the given expressions. We aim to eliminate the quantifiers from quantities of the form

$$\mathfrak{Z}x: \left(\bigwedge_{i=1}^n L_i \searrow \tilde{a} \right) \quad \text{or} \quad \mathfrak{L}x: \left(\bigwedge_{i=1}^n L_i \nearrow \tilde{a} \right),$$

where each L_i is a linear inequality. We start with an example.

Example 6. Continuing Example 5, we perform quantifier elimination on

$$g = \mathfrak{Z}x: \underbrace{(x < y_1 + 2 \wedge x \leq -y_3 \wedge x \geq y_2)}_{=D} \searrow \underbrace{2 \cdot x + z}_{=\tilde{a}}.$$

Fix some valuation σ . First observe that if there is no $q \in \mathbb{Q}$ such that $\sigma[x \mapsto q] \models D$ — or, phrased in predicate logic, if $\sigma \not\models \exists x: D$ —, then $\sigma[g]$ evaluates to $-\infty$. Otherwise, we need to inspect D and $\tilde{a}$ closer in order to determine $\sigma[g]$. Hence, eliminating the $\mathfrak{Z}x$ -quantifier involves characterizing whether $\sigma \models \exists x: D$ holds *without referring to x* . This boils down to performing *classical* quantifier elimination on the formula $\exists x: D$. We leverage classical Fourier-Motzkin variable elimination: Compare the bounds D imposes on x and encode whether they are consistent. Towards this end, we construct

$$\varphi_{\exists}(D, x) = \underbrace{y_2 < y_1 + 2 \wedge y_2 \leq -y_3}_{\text{equivalent to } \exists x: D} \in \text{Bool}.$$

Now, how can we characterize $\sigma[g]$ in case $\sigma \models \varphi_{\exists}(D, x)$? We first observe that x occurs positively in $\tilde{a}$. Therefore, intuitively, the $\mathfrak{Z}x$ -quantifier aims to maximize the value of x under all possible assignments satisfying D . Since we isolate x in every inequality where x occurs, we can readily read off from D that x 's maximal (or, in fact, *supremal*) value is given by the *minimum* of $\sigma(y_1) + 2$ and $-\sigma(y_3)$ — the least of all upper bounds imposed on x . Overall, we get

$$\begin{aligned} g \equiv & [\varphi_{\exists}(D, x)] \cdot ([y_1 + 2 \leq -y_3] \cdot (2 \cdot y_1 + z + 4) \\ & + [y_1 + 2 > -y_3] \cdot (-2 \cdot y_3 + z)) + [\neg \varphi_{\exists}(D, x)] \cdot (-\infty). \end{aligned}$$

The above quantifier-free equivalent of g indeed evaluates to $-\infty$ if $\sigma \not\models \varphi_{\exists}(D, x)$ and, otherwise, performs a case distinction on said least upper bounds on x .

Finally, consider the quantity

$$g' = \mathfrak{Z}x: \underbrace{y_1 < z}_{=D'} \searrow \underbrace{2 \cdot x + z}_{=\tilde{a}'}$$

and observe that D' does not impose any bound on x whatsoever. This highlights the need for a careful treatment of ∞ (or, in dual situations, $-\infty$): Since x occurs positively in $\tilde{a}$, we have $\sigma[g'] = \infty$ whenever $\sigma \models y_1 < z$, and $\sigma[g'] = -\infty$ otherwise. We thus have

$$g' \equiv [y_1 < z] \cdot \infty + [y_1 \geq z] \cdot (-\infty).$$

When considering $\mathcal{L}x$ -quantifiers or when x occurs negatively in $\tilde{a}$, the above described observations need to be dualized, which we detail further below. $\triangle$

We condense the following steps for eliminating the $\mathcal{Z}x$ - or $\mathcal{L}x$ -quantifiers:

1. Extract lower and upper bounds on x imposed by the Boolean expression D .
2. Construct the Boolean expression $\varphi_{\exists}(D, x)$ via classical Fourier-Motzkin.
3. Characterize least upper- and greatest lower bounds on x admitted by D .
4. Eliminate the $\mathcal{Z}x$ - or $\mathcal{L}x$ -quantifiers by gluing the above concepts together.

We detail these steps in the subsequent paragraphs. Fix $x \in \text{Vars}$, $n \geq 1$, and

$$D = \bigwedge_{i=1}^n L_i .$$

Extracting Lower and Upper Bounds. Given $\sim \in \{>, \geq, <, \leq\}$, we define

$$\begin{aligned} & \text{Bnd}_{x \sim \cdot}(D) \\ = & \begin{cases} \{\tilde{a} \in \text{LinAX}^{\pm\infty} \mid \exists i \in \{1, \dots, n\}: L_i = x \sim \tilde{a}\} & \text{if } \sim \in \{>, <\} \\ \{\tilde{a} \in \text{LinAX}^{\pm\infty} \mid \exists i \in \{1, \dots, n\}: L_i = x \sim \tilde{a}\} \cup \{-\infty\} & \text{if } \sim = \geq \\ \{\tilde{a} \in \text{LinAX}^{\pm\infty} \mid \exists i \in \{1, \dots, n\}: L_i = x \sim \tilde{a}\} \cup \{\infty\} & \text{if } \sim = \leq \end{cases} \end{aligned}$$

and let $\text{UBnd}_x = \text{Bnd}_{x < \cdot}(D) \cup \text{Bnd}_{x \leq \cdot}(D)$ and $\text{LBnd}_x = \text{Bnd}_{x > \cdot}(D) \cup \text{Bnd}_{x \geq \cdot}(D)$. Including ∞ and $-\infty$, respectively, by default will be convenient when characterizing least upper- and greatest lower bounds on x admitted by D : If there is no upper (resp. lower) bound on x whatsoever imposed by D , our construction will automatically default to ∞ (resp. $-\infty$).

Classical Fourier-Motzkin Quantifier Elimination with Infinity. We define

$$\begin{aligned} \varphi_{\exists}(D, x) = & \bigwedge_{\substack{\tilde{a} \in \text{Bnd}_{x \geq \cdot}(D), \\ \tilde{b} \in \text{Bnd}_{x \leq \cdot}(D)}} \tilde{a} \leq \tilde{b} \wedge \bigwedge_{\substack{\tilde{a} \in \text{Bnd}_{x \geq \cdot}(D), \\ \tilde{b} \in \text{Bnd}_{x < \cdot}(D)}} \tilde{a} < \tilde{b} \wedge \bigwedge_{\substack{\tilde{a} \in \text{Bnd}_{x > \cdot}(D), \\ \tilde{b} \in \text{Bnd}_{x \leq \cdot}(D)}} \tilde{a} < \tilde{b} \\ & \wedge \bigwedge_{\substack{\tilde{a} \in \text{Bnd}_{x > \cdot}(D), \\ \tilde{b} \in \text{Bnd}_{x < \cdot}(D)}} \tilde{a} < \tilde{b} \wedge \bigwedge_{\substack{i \in \{1, \dots, n\}, \\ x \notin \text{FV}(L_i)}} L_i \end{aligned}$$

as is standard in Fourier-Motzkin variable elimination. The soundness of this construction generalizes to Boolean expressions involving ∞ or $-\infty$:

Lemma 2 ([25, 45]). *For all $\sigma \in \Sigma$, we have*

$$\sigma \models \varphi_{\exists}(D, x) \quad \text{iff} \quad \{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} \neq \emptyset .$$

Characterizing Suprema and Infima. Fix some ordering on the bounds on x , i.e., let $\text{UBnd}_x = \{\tilde{u}_1, \dots, \tilde{u}_{m_1}\}$ and $\text{LBnd}_x = \{\tilde{\ell}_1, \dots, \tilde{\ell}_{m_2}\}$. We define:

1. $\varphi_{\text{sup}}(D, x, \tilde{u}_i) = \bigwedge_{k=1}^{i-1} \tilde{u}_i < \tilde{u}_k \wedge \bigwedge_{k=i+1}^{m_1} \tilde{u}_i \leq \tilde{u}_k$
2. $\varphi_{\text{inf}}(D, x, \tilde{\ell}_i) = \bigwedge_{k=1}^{i-1} \tilde{u}_i > \tilde{u}_k \wedge \bigwedge_{k=i+1}^{m_2} \tilde{u}_i \geq \tilde{u}_k$

Intuitively, $\varphi_{\text{sup}}(D, x, \tilde{u}_i)$ evaluates to **true** under valuation σ if $^\sigma\llbracket \tilde{u}_i \rrbracket$ evaluates to the *least upper bound* on x admitted by D under σ with *minimal* i . The intuition for $\varphi_{\text{inf}}(D, x, \tilde{\ell}_i)$ is analogous. Put more formally:

Theorem 3. *Let $\sigma \in \Sigma$ such that*

$$\{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} \neq \emptyset .$$

Then all of the following statements hold:

1. *There is exactly one $\tilde{u} \in \text{UBnd}_x$ such that*

$$\sigma \models \varphi_{\text{sup}}(D, x, \tilde{u}) .$$

2. *If $\tilde{u} \in \text{UBnd}_x$ and $\sigma \models \varphi_{\text{sup}}(D, x, \tilde{u})$, then*

$$^\sigma\llbracket \tilde{u} \rrbracket = \sup\{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} .$$

3. *There is exactly one $\tilde{\ell} \in \text{LBnd}_x$ such that*

$$\sigma \models \varphi_{\text{inf}}(D, x, \tilde{\ell}) .$$

4. *If $\tilde{\ell} \in \text{LBnd}_x$ and $\sigma \models \varphi_{\text{inf}}(D, x, \tilde{\ell})$, then*

$$^\sigma\llbracket \tilde{\ell} \rrbracket = \inf\{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} .$$

Proof. See [10, App. A.3].

An immediate consequence of Theorem 3 is that, for every $\sigma \models \varphi_{\exists}(D, x)$,

$$\begin{aligned} ^\sigma\llbracket \sum_{i=1}^{m_1} [\varphi_{\text{sup}}(D, x, \tilde{u}_i)] \cdot \tilde{u}_i \rrbracket &= \sup\{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} \\ ^\sigma\llbracket \sum_{i=1}^{m_2} [\varphi_{\text{inf}}(D, x, \tilde{\ell}_i)] \cdot \tilde{\ell}_i \rrbracket &= \inf\{q \in \mathbb{Q} \mid \sigma[x \mapsto q] \models D\} \end{aligned}$$

It is in this sense that φ_{sup} and φ_{inf} characterize least upper- and greatest lower bounds on x admitted by D .

Eliminating the Quantitative Quantifiers Equipped with the preceding prerequisites, we formalize our construction and prove it sound. Given extended arithmetic expressions $\tilde{a}, \tilde{e} \in \text{LinAX}^{\pm\infty}$ and a variable $x \in \text{Vars}$, we define

$$\tilde{e}(x, \tilde{a}) = \begin{cases} \infty & \text{if } x \text{ occurs positively in } \tilde{e} \text{ and } \tilde{a} = \infty \text{ or} \\ & x \text{ occurs negatively in } \tilde{e} \text{ and } \tilde{a} = -\infty \\ -\infty & \text{if } x \text{ occurs positively in } \tilde{e} \text{ and } \tilde{a} = -\infty \text{ or} \\ & x \text{ occurs negatively in } \tilde{e} \text{ and } \tilde{a} = \infty \\ \tilde{e} & \text{if } x \notin \text{FV}(\tilde{e}) \\ \tilde{e}[x/\tilde{a}] & \text{otherwise,} \end{cases}$$

where in the last case we have $\tilde{a} \in \text{LinAX}$ so $\tilde{e}[x/\tilde{a}]$ is the standard syntactic replacement⁹ of x by $\tilde{a}$ in $\tilde{e}$. Our sought-after quantifier-free equivalents are¹⁰

$$\text{QE}(\mathcal{Z}x: (D \searrow \tilde{e})) = [\neg\varphi_{\exists}(D, x)] \cdot (-\infty) \quad (2)$$

$$+ [\varphi_{\exists}(D, x)] \cdot \begin{cases} \sum_{i=1}^{m_1} [\varphi_{\text{sup}}(D, x, \tilde{u}_i)] \cdot \tilde{e}(x, \tilde{u}_i) & \text{if } x \text{ occurs positively in } \tilde{e} \\ \sum_{i=1}^{m_2} [\varphi_{\text{inf}}(D, x, \tilde{\ell}_i)] \cdot \tilde{e}(x, \tilde{\ell}_i) & \text{if } x \text{ occurs negatively in } \tilde{e} \\ \tilde{e} & \text{if } x \notin \text{FV}(\tilde{e}) \end{cases}$$

and

$$\text{QE}(\mathcal{L}x: (D \nearrow \tilde{e})) = [\neg\varphi_{\exists}(D, x)] \cdot \infty \quad (3)$$

$$+ [\varphi_{\exists}(D, x)] \cdot \begin{cases} \sum_{i=1}^{m_2} [\varphi_{\text{inf}}(D, x, \tilde{\ell}_i)] \cdot \tilde{e}(x, \tilde{\ell}_i) & \text{if } x \text{ occurs positively in } \tilde{e} \\ \sum_{i=1}^{m_1} [\varphi_{\text{sup}}(D, x, \tilde{u}_i)] \cdot \tilde{e}(x, \tilde{u}_i) & \text{if } x \text{ occurs negatively in } \tilde{e} \\ \tilde{e} & \text{if } x \notin \text{FV}(\tilde{e}). \end{cases}$$

These constructions comply with our intuition from Example 6. We apply classical Fourier-Motzkin variable elimination to check whether the respective supremum (infimum) evaluates to $-\infty$ (resp. ∞). If $\varphi_{\exists}(D, x)$ is satisfied, then we inspect $\tilde{e}$ closer, select the right bound $\tilde{u}$ (resp. $\tilde{\ell}$) on x in D via φ_{sup} (resp. φ_{inf}), and substitute x in $\tilde{e}$ by $\tilde{u}$ ($\tilde{\ell}$) while obeying the arithmetic laws for the extended reals given in Section 2. The resulting quantities are partitioning and:

Theorem 4. *Let $x \in \text{Vars}$ and $\tilde{e} \in \text{LinAX}^{\pm\infty}$. We have:*

1. $\mathcal{Z}x: (D \searrow \tilde{e}) \equiv \text{QE}(\mathcal{Z}x: (D \searrow \tilde{e}))$
2. $\mathcal{L}x: (D \nearrow \tilde{e}) \equiv \text{QE}(\mathcal{L}x: (D \nearrow \tilde{e}))$

Proof. Fixing some $\sigma \in \Sigma$, we first distinguish the cases $\sigma \models \varphi_{\exists}(D, x)$ and $\sigma \not\models \varphi_{\exists}(D, x)$. For $\sigma \models \varphi_{\exists}(D, x)$, we then distinguish the cases $x \notin \text{FV}(\tilde{e})$ and

⁹ provided in [10, App. B.2]

¹⁰ recall that $\text{UBnd}_x = \{\tilde{u}_1, \dots, \tilde{u}_{m_1}\}$ and $\text{LBnd}_x = \{\tilde{\ell}_1, \dots, \tilde{\ell}_{m_2}\}$.

Algorithm 1: $\text{Elim}(\cdot)$ — Quantitative Quantifier Elimination

```

1 Input: partitioning  $f \in \text{LinQuant}$ 
2 Output: quantifier-free partitioning  $\text{Elim}(f) \in \text{LinQuant}$  with  $\text{Elim}(f) \equiv f$ 
3 if  $f$  is quantifier-free then
4   | return  $f$ 
5 else if  $f$  is of the form  $Qx: g$  then
6   |  $g \leftarrow \text{Elim}(g)$ ;
7   | transform  $g$  into  $\text{GNF}_x$ ;
   | // let  $g = \sum_{i=1}^n [\bigvee_{j=1}^{m_i} D_{i,j}] \cdot \tilde{a}_{i,j}$ 
8   | if  $Q = \mathcal{Z}$  then
9   |   | return  $\text{MAX}(\bigcup_{i=1}^n \bigcup_{j=1}^{m_i} \{ \underbrace{\text{QE}(\mathcal{Z}x: D_{i,j} \searrow \tilde{a}_{i,j})}_{\text{given by Equation (2) on page 190}} \})$ 
10  | else if  $Q = \mathcal{L}$  then
11  |   | return  $\text{MIN}(\bigcup_{i=1}^n \bigcup_{j=1}^{m_i} \{ \underbrace{\text{QE}(\mathcal{L}x: D_{i,j} \nearrow \tilde{a}_{i,j})}_{\text{given by Equation (3) on page 190}} \})$ 

```

$x \in \text{FV}(\tilde{e})$, the latter case being the most interesting. The key insight is that if $\tilde{e}$ is of the form $q_0 + \sum_{y \in \text{Vars}} q_y \cdot y$ and $\sigma \models \varphi_{\exists}(D, x)$, then

$$\begin{aligned} & \sigma[\![\mathcal{Z}x: (D \searrow \tilde{e})]\!] \\ &= q_0 + \left(\sum_{y \in \text{Vars} \setminus \{x\}} q_y \cdot \sigma(y) \right) + q_x \cdot \begin{cases} \sigma[\![\sum_{i=1}^{m_1} [\varphi_{\text{sup}}(D, x, \tilde{u}_i)] \cdot \tilde{u}_i]\!] & \text{if } q_x > 0 \\ \sigma[\![\sum_{i=1}^{m_2} [\varphi_{\text{inf}}(D, x, \tilde{\ell}_i)] \cdot \tilde{\ell}_i]\!] & \text{if } q_x < 0 \end{cases} \end{aligned}$$

by Theorem 3. See [10, App. A.4] for a detailed proof. $\square$

3.4 Algorithmically Eliminating Quantitative Quantifiers

We summarize our quantifier elimination technique in Algorithm 1, which takes as input a partitioning $f \in \text{LinQuant}$ and computes a quantifier-free equivalent $\text{Elim}(f)$ of f by proceeding in a recursive inner- to outermost fashion. Since f is partitioning and since both MAX and MIN always return partitioning quantities, the transformation of g into GNF_x only involves transforming every Boolean expression into DNF and isolating x in every inequality where x occurs. The soundness of Algorithm 1 is an immediate consequence of our observations from the preceding sections. Moreover, the algorithm terminates because the number of recursive invocations equals the number of quantifiers occurring in f .

In order to upper-bound the space complexity of Algorithm 1, we agree on the following: The size $|\varphi|$ of a Boolean expression φ is the number of (not necessarily distinct) inequalities it contains. The *width* $|f|_{\rightarrow}$ of $f \in \text{LinQuant}$ is its number of summands, and its *depth* $|f|_{\downarrow}$ is the maximum of the sizes of the Boolean expressions f contains.

Theorem 5. *Algorithm 1 is sound and terminates. Moreover, for partitioning¹¹ $f \in \text{LinQuant}$ with $|f|_{\rightarrow} = n$ and $|f|_{\downarrow} = m$ containing exactly one quantifier,*

$$|\text{Elim}(f)|_{\rightarrow} \leq n \cdot 2^m \cdot (m+2)^{n \cdot 2^m} \quad \text{and} \quad |\text{Elim}(f)|_{\downarrow} \leq n \cdot 2^m \cdot ((m+2/2)^2 + m + 1) .$$

Proof. We exploit that (i) transforming a Boolean expression φ of size l into DNF produces at most 2^l disjuncts, each consisting of at most l linear inequalities and (ii) if D is of size l , then $|\varphi_{\exists}(D, x)| \leq (l+2/2)^2$. See [10, App. A.6] for details. $\square$

Fixing m and l as above, the resulting upper bounds for quantities containing k quantifiers are thus non-elementary in k . Investigating *lower* space complexity bounds of Algorithm 1 or the computational complexity of the quantitative quantifier elimination problem is left for future work.

4 Quantitative Craig Interpolation

We now employ our quantifier elimination procedure Elim from Algorithm 1 to derive a quantitative Craig interpolation theorem. Let us first agree on a notion of quantitative Craig interpolants, which is a quantitative analogue of the notion from [32]. Given $f, f' \in \text{LinQuant}$, we say that f (quantitatively) *entails* f' , denoted by $f \models f'$, if $\forall \sigma \in \Sigma: \sigma \llbracket f \rrbracket \leq \sigma \llbracket f' \rrbracket$.

Definition 6 (Quantitative Craig Interpolant). *Given $f, f', g \in \text{LinQuant}$ with $f \models f'$, we say that g is a (quantitative) Craig interpolant of (f, f') , if*

$$f \models g \text{ and } g \models f' \quad \text{and} \quad \text{FV}(g) \subseteq \text{FV}(f) \cap \text{FV}(f') . \quad \triangle$$

In words, g sits between f and f' and the free variables occurring in g also occur free in *both* f and f' . We will now see that piecewise linear quantities enjoy the property of being *quantifier-free interpolating* [32]: For all $f, f' \in \text{LinQuant}$ with $f \models f'$, there exists a quantifier-free Craig interpolant of (f, f') . More precisely, we prove that both the *strongest* and the *weakest* Craig interpolants of (f, f') are *quantifier-free and effectively constructible*. Our construction is inspired by existing techniques for constructing *classical* Craig interpolants via *classical* quantifier elimination [23,24]: By “projecting-out” the free variables in f which are *not* free in f' via $\mathcal{Z}$, we obtain the *strongest* Craig interpolant of (f, f') . Dually, by “projecting-out” the free variables in f' which are *not* free in f via $\mathcal{L}$, we obtain the *weakest* Craig interpolant of (f, f') . Put formally:

Theorem 6. *Let $f, f' \in \text{LinQuant}$ with $f \models f'$. We have:*

1. For $\{x_1, \dots, x_n\} = \text{FV}(f) \setminus \text{FV}(f')$,

$$g = \text{Elim}(\mathcal{Z}x_1 \dots \mathcal{Z}x_n: f)$$

is the strongest quantitative Craig interpolant of (f, f') , i.e.,

$$\forall \text{ Craig interpolants } g' \text{ of } (f, f'): \quad g \models g' .$$

¹¹ If f needs to be pre-processed to make it partitioning via the construction from Section 3.1, then n is to be substituted by 2^n and m is to be substituted by $n \cdot m$.

2. For $\{y_1, \dots, y_m\} = \text{FV}(f') \setminus \text{FV}(f)$,

$$g = \text{Elim}(\mathcal{L}_{y_1} \dots \mathcal{L}_{y_m} : f')$$

is the weakest quantitative Craig interpolant of (f, f') , i.e.,

$$\forall \text{ Craig interpolants } g' \text{ of } (f, f') : g' \models g .$$

Proof. See [10, App. A.5].

Example 7. Consider the following quantities f, f' which satisfy $f \models f'$:

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

Pre-processing f and f' to make them partitioning and simplifying yields

$$\underbrace{\text{Elim}(\mathcal{Q}y : f) = [x \geq 0] \cdot 2 \cdot x}_{\text{strongest Craig interpolant}} \quad \text{and} \quad \underbrace{\text{Elim}(\mathcal{L}z : f') = [x \geq 0] \cdot (3 \cdot x + 1)}_{\text{weakest Craig interpolant}} .$$

△

5 Conclusion

We have investigated both quantitative quantifier elimination and quantitative Craig interpolation for piecewise linear quantities — an assertion language in automatic quantitative software verification. We have provided a sound and complete quantifier elimination algorithm, proved it sound, and analyzed upper space-complexity bounds. Using our algorithm, we have derived a quantitative Craig interpolation theorem for arbitrary piecewise linear quantities.

We see ample space for future work. First, we could investigate alternative quantifier elimination procedures: Our algorithm can be understood as a quantitative generalization of Fourier-Motzkin variable elimination [45, 25]. It would be interesting to apply, e.g., virtual substitution [40] in the quantitative setting and to compare the so-obtained approaches — both empirically and theoretically. We might also benefit from improvements of Fourier-Motzkin variable elimination such as FMPLEX [48] to improve the practical feasibility of our approach. Moreover, we have focussed on $\mathbb{Q}$ -valued variables. We plan to investigate techniques which apply to integer-valued variables using, e.g., Cooper’s method [18] and in how far our results can be generalized to a non-linear setting.

Finally, we plan to investigate potential applications of our techniques:

1. Dillig et al. [22] present a quantifier elimination-based algorithm for generating inductive loop invariants of classical programs abductively. Generalizing this algorithm to the probabilistic setting, where weakest *preconditions* are replaced by weakest *preexpectations*, might yield a promising application of our quantifier elimination algorithm.

2. We are currently investigating the applicability of McMillan’s interpolation and SAT-based model checking algorithm [43] to *probabilistic* program verification. One of the major challenges is to obtain suitable quantitative interpolants and we hope that our results on the existence of interpolants spark the development of suitable techniques.
3. In the light of the above application and Remark 1, we plan to adapt Albarghouthi’s and McMillan’s technique for computing [1] “simpler” interpolants.

References

1. Albarghouthi, A., McMillan, K.L.: Beautiful interpolants. In: CAV. Lecture Notes in Computer Science, vol. 8044, pp. 313–329. Springer (2013)
2. Alberti, F., Bruttomesso, R., Ghilardi, S., Ranise, S., Sharygina, N.: Lazy abstraction with interpolants for arrays. In: LPAR. Lecture Notes in Computer Science, vol. 7180, pp. 46–61. Springer (2012)
3. Aliprantis, C., Burkinshaw, O.: Principles of Real Analysis. North Holland (1981)
4. Baaz, M., Veith, H.: Quantifier elimination in fuzzy logic. In: CSL. Lecture Notes in Computer Science, vol. 1584, pp. 399–414. Springer (1998)
5. Batz, K., Chen, M., Junges, S., Kaminski, B.L., Katoen, J., Matheja, C.: Probabilistic program verification via inductive synthesis of inductive invariants. In: TACAS (2). Lecture Notes in Computer Science, vol. 13994, pp. 410–429. Springer (2023)
6. Batz, K., Chen, M., Kaminski, B.L., Katoen, J., Matheja, C., Schröder, P.: Latticed k-induction with an application to probabilistic programs. In: CAV (2). Lecture Notes in Computer Science, vol. 12760, pp. 524–549. Springer (2021)
7. Batz, K., Gallus, A., Kaminski, B.L., Katoen, J.P., Winkler, T.: Weighted programming: a programming paradigm for specifying mathematical models. Proc. ACM Program. Lang. **6**(OOPSLA1) (2022)
8. Batz, K., Kaminski, B.L., Katoen, J., Matheja, C.: Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. Proc. ACM Program. Lang. **5**(POPL), 1–30 (2021)
9. Batz, K., Kaminski, B.L., Katoen, J.P., Matheja, C., Noll, T.: Quantitative separation logic: A logic for reasoning about probabilistic pointer programs. Proc. ACM Program. Lang. **3**(POPL), 34:1–34:29 (2019)
10. Batz, K., Katoen, J.P., Orhan, N.: Quantifier elimination and craig interpolation: The quantitative way (technical report) (2025), <https://arxiv.org/abs/2501.15156>
11. Beyer, D., Lee, N., Wendler, P.: Interpolation and sat-based model checking revisited: Adoption to software verification. CoRR **abs/2208.05046** (2022)
12. Britikov, K., Blich, M., Sharygina, N., Fedyukovich, G.: Reachability analysis for multiloop programs using transition power abstraction. In: FM (1). Lecture Notes in Computer Science, vol. 14933, pp. 558–576. Springer (2024)
13. ten Cate, B., Comer, J.: Craig interpolation for decidable first-order fragments. In: FoSSaCS (2). Lecture Notes in Computer Science, vol. 14575, pp. 137–159. Springer (2024)
14. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV. Lecture Notes in Computer Science, vol. 8044, pp. 511–526. Springer (2013)

15. Chen, M., Wang, J., An, J., Zhan, B., Kapur, D., Zhan, N.: NIL: learning nonlinear interpolants. In: CADE. Lecture Notes in Computer Science, vol. 11716, pp. 178–196. Springer (2019)
16. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages. Lecture Notes in Computer Science, vol. 33, pp. 134–183. Springer (1975)
17. Colón, M., Sankaranarayanan, S., Sipma, H.: Linear invariant generation using non-linear constraint solving. In: CAV. Lecture Notes in Computer Science, vol. 2725, pp. 420–432. Springer (2003)
18. Cooper, D.: Theorem proving in arithmetic without multiplication. Machine Intelligence (1972)
19. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic* **22**(3), 269–285 (1957)
20. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall (1976)
21. Dijkstra, E.W., Scholten, C.S.: Predicate Calculus and Program Semantics. Texts and Monographs in Computer Science, Springer (1990)
22. Dillig, I., Dillig, T., Li, B., McMillan, K.L.: Inductive invariant generation via abductive inference. In: OOPSLA. pp. 443–456. ACM (2013)
23. Esparza, J., Kiefer, S., Schwoon, S.: Abstraction refinement with Craig interpolation and symbolic pushdown systems. In: TACAS. Lecture Notes in Computer Science, vol. 3920, pp. 489–503. Springer (2006)
24. Esparza, J., Kiefer, S., Schwoon, S.: Abstraction refinement with Craig interpolation and symbolic pushdown systems. *J. Satisf. Boolean Model. Comput.* **5**(1-4), 27–56 (2008)
25. Fourier, J.: Analyse des travaux de l'Académie royale des sciences pendant l'année 1824: rapport lu dans la séance publique de l'Institut le 24 avril 1825. Partie mathématique. Institut (Paris), Institut royal de France (1825)
26. Gan, T., Dai, L., Xia, B., Zhan, N., Kapur, D., Chen, M.: Interpolant synthesis for quadratic polynomial inequalities and combination with EUF. In: IJCAR. Lecture Notes in Computer Science, vol. 9706, pp. 195–212. Springer (2016)
27. Gretz, F., Katoen, J., McIver, A.: Prinsys - on a quest for probabilistic loop invariants. In: QEST. Lecture Notes in Computer Science, vol. 8054, pp. 193–208. Springer (2013)
28. Henzinger, T.A., Jhala, R., Majumdar, R., McMillan, K.L.: Abstractions from proofs. In: POPL. pp. 232–244. ACM (2004)
29. Jhala, R., McMillan, K.L.: A practical and complete approach to predicate refinement. In: TACAS. Lecture Notes in Computer Science, vol. 3920, pp. 459–473. Springer (2006)
30. Kaminski, B.L.: Advanced Weakest Precondition Calculi for Probabilistic Programs. Ph.D. thesis, RWTH Aachen University, Germany (2019)
31. Kapur, D.: Automatically generating loop invariants using quantifier elimination. In: Deduction and Applications (2005)
32. Kapur, D., Majumdar, R., Zarba, C.G.: Interpolation for data structures. p. 105–116. SIGSOFT '06/FSE-14, ACM (2006)
33. Katoen, J.P., McIver, A., Meinicke, L., Morgan, C.C.: Linear-invariant generation for probabilistic programs: - automated support for proof-based methods. In: SAS. Lecture Notes in Computer Science, vol. 6337, pp. 390–406. Springer (2010)
34. Knuth, D.E.: Two notes on notation. *The American Mathematical Monthly* **99**(5), 403–422 (1992)

35. Komuravelli, A., Gurfinkel, A., Chaki, S.: SMT-based model checking for recursive programs. In: CAV. Lecture Notes in Computer Science, vol. 8559, pp. 17–34. Springer (2014)
36. Komuravelli, A., Gurfinkel, A., Chaki, S.: SMT-based model checking for recursive programs. *Formal Methods Syst. Des.* **48**(3), 175–205 (2016)
37. Kozen, D.: A probabilistic PDL. In: STOC. pp. 291–297. ACM (1983)
38. Kozen, D.: A probabilistic PDL. *J. Comput. Syst. Sci.* **30**(2), 162–178 (1985)
39. Krishnan, H.G.V., Vizel, Y., Ganesh, V., Gurfinkel, A.: Interpolating strong induction. In: CAV (2). Lecture Notes in Computer Science, vol. 11562, pp. 367–385. Springer (2019)
40. Loos, R., Weispfenning, V.: Applying linear quantifier elimination. *Comput. J.* **36**(5), 450–462 (1993)
41. Mahdi, A., Fränzle, M.: Generalized Craig interpolation for stochastic satisfiability modulo theory problems. In: RP. Lecture Notes in Computer Science, vol. 8762, pp. 203–215. Springer (2014)
42. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science, Springer (2005)
43. McMillan, K.L.: Interpolation and sat-based model checking. In: CAV. Lecture Notes in Computer Science, vol. 2725, pp. 1–13. Springer (2003)
44. McMillan, K.L.: Lazy abstraction with interpolants. In: CAV. Lecture Notes in Computer Science, vol. 4144, pp. 123–136. Springer (2006)
45. Motzkin, T.: Beitrage zur Theorie der linearen Ungleichungen. Universitaet Basel (1936)
46. de Moura, L.M., Bjørner, N.S.: Z3: an efficient SMT solver. In: TACAS. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer (2008)
47. Müller, P.: Building Deductive Program Verifiers - Lecture Notes. Engineering Secure and Dependable Software Systems (2019)
48. Nalbach, J., Promies, V., Ábrahám, E., Kobialka, P.: Fmplex: A novel method for solving linear real arithmetic problems. In: GandALF. EPTCS, vol. 390, pp. 16–32 (2023)
49. Ngo, V.C., Carbonneaux, Q., Hoffmann, J.: Bounded expectations: resource analysis for probabilistic programs. In: PLDI. pp. 496–512. ACM (2018)
50. Sanner, S., Delgado, K.V., de Barros, L.N.: Symbolic dynamic programming for discrete and continuous state MDPs. In: UAI. pp. 643–652. AUAI Press (2011)
51. Schröer, P., Batz, K., Kaminski, B.L., Katoen, J.P., Matheja, C.: A deductive verification infrastructure for probabilistic programs. *Proc. ACM Program. Lang.* **7**(OOPSLA2), 2052–2082 (2023)
52. Sery, O., Fedyukovich, G., Sharygina, N.: Interpolation-based function summaries in bounded model checking. In: Haifa Verification Conference. Lecture Notes in Computer Science, vol. 7261, pp. 160–175. Springer (2011)
53. Sery, O., Fedyukovich, G., Sharygina, N.: Incremental upgrade checking by means of interpolation-based function summaries. In: FMCAD. pp. 114–121. IEEE (2012)
54. Still, G.: Lectures on parametric optimization: An introduction (2018)
55. Tavana, N., Pourmahdian, M., Khatami, S.A.: The Craig interpolation property in first-order Gödel logic. *Fuzzy Sets Syst.* **485**, 108958 (2024)
56. Vizel, Y., Gurfinkel, A.: Interpolating property directed reachability. In: CAV. Lecture Notes in Computer Science, vol. 8559, pp. 260–276. Springer (2014)
57. Wu, H., Wang, J., Xia, B., Li, X., Zhan, N., Gan, T.: Nonlinear Craig interpolant generation over unbounded domains by separating semialgebraic sets. In: FM (1). Lecture Notes in Computer Science, vol. 14933, pp. 92–110. Springer (2024)

58. Yang, T., Fu, H., Ke, J., Zhan, N., Wu, S.: Piecewise linear expectation analysis via k-induction for probabilistic programs. CoRR **abs/2403.17567** (2024)
59. Zhang, L., Kaminski, B.L.: Quantitative strongest post: a calculus for reasoning about the flow of quantitative information. Proc. ACM Program. Lang. **6**(OOPSLA1) (2022)

Open Access. This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution, and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

