A Primal-Dual Perspective on Program Verification Algorithms

TAKESHI TSUKADA, Chiba University, Japan
HIROSHI UNNO, Tohoku University, Japan
ODED PADON, Weizmann Institute of Science, Israel
SHARON SHOHAM, Tel Aviv University, Israel

Many algorithms in verification and automated reasoning leverage some form of duality between proofs and refutations or counterexamples. In most cases, duality is only used as an intuition that helps in understanding the algorithms and is not formalized. In other cases, duality is used explicitly, but in a specially tailored way that does not generalize to other problems.

In this paper we propose a unified primal-dual framework for designing verification algorithms that leverage duality. To that end, we generalize the concept of a Lagrangian that is commonly used in linear programming and optimization to capture the domains considered in verification problems, which are usually discrete, e.g., powersets of states, predicates, ranking functions, etc. A Lagrangian then induces a primal problem and a dual problem. We devise an abstract primal-dual procedure that simultaneously searches for a primal solution and a dual solution, where the two searches guide each other. We provide sufficient conditions that ensure that the procedure makes progress under certain monotonicity assumptions on the Lagrangian.

We show that many existing algorithms in program analysis, verification, and automated reasoning can be derived from our algorithmic framework with a suitable choice of Lagrangian. The Lagrangian-based formulation sheds new light on various characteristics of these algorithms, such as the ingredients they use to ensure monotonicity and guarantee progress. We further use our framework to develop a new validity checking algorithm for fixpoint logic over quantified linear arithmetic. Our prototype achieves promising results and in some cases solves instances that are not solved by state-of-the-art techniques.

CCS Concepts: • **Theory of computation** → **Logic and verification**; **Program verification**.

Additional Key Words and Phrases: primal-dual method, Lagrangian, verification

ACM Reference Format:
Takeshi Tsukada, Hiroshi Unno, Oded Padon, and Sharon Shoham. 2025. A Primal-Dual Perspective on Program Verification Algorithms. *Proc. ACM Program. Lang.* 9, POPL, Article 68 (January 2025), 32 pages. https://doi.org/10.1145/3704904

1 Introduction

Duality and the primal-dual approach are a corner stone in algorithm design and optimization [Bertsimas and Tsitsiklis 1997; Boyd and Vandenberghe 2014]. In program analysis, verification, and automated reasoning, many algorithms also have a primal-dual flavor. The common duality in these contexts is between a proof and a refutation or counterexample. That is, the algorithm simultaneously searches for a proof and a refutation, and both searches guide each other in a meaningful way. For example, in counterexample guided abstraction refinement (CEGAR) [Ball et al. 2001a;

Authors' Contact Information: Takeshi Tsukada, Chiba University, Chiba, Japan, t.tsukada@acm.org; Hiroshi Unno, Tohoku University, Sendai, Japan, hiroshi.unno@acm.org; Oded Padon, Weizmann Institute of Science, Rehovot, Israel, oded.padon@weizmann.ac.il; Sharon Shoham, Tel Aviv University, Tel Aviv, Israel, sharon.shoham@gmail.com.

This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2025 Copyright held by the owner/author(s).
ACM 2475-1421/2025/1-ART68
https://doi.org/10.1145/3704904

Clarke et al. 2000], the search alternates between searching for a counterexample to the current abstraction and refining the abstraction to eliminate a spurious counterexample. The intuition is that refining the abstraction according to a spurious counterexample makes progress towards a proof (if one exists), and finding counterexamples for more refined abstractions might make progress towards a non-spurious counterexample (if one exists).

A similar interaction between the search for proofs and the search for counterexamples exists in many program analysis, verification, and automated reasoning algorithms, including may-must analysis [Godefroid et al. 2010], ICE learning [Garg et al. 2014], lazy SMT solving [Nieuwenhuis et al. 2006], quantifier instantiation [Ge and de Moura 2009], program synthesis [Jha et al. 2010], game solving [Farzan and Kincaid 2018] and more. In most of these cases, the primal-dual nature of the search algorithm is not made explicit and formal.

Recently, some algorithms were proposed that are explicitly manifested as primal-dual algorithms. In [Padon et al. 2022], a primal-dual algorithm for safety verification is proposed, based on a duality between executions in a transition system and a form of incremental induction proofs. In [Unno et al. 2023], a primal-dual algorithm for validity of formulas in first-order fixpoint logic with background theories is proposed, based on a duality between the validity of a formula and that of its negation, which corresponds to a duality between least and greatest fixpoints. However, while these algorithms are both explicitly primal-dual, the duality and the primal-dual algorithm are bespoke in each case.

In this paper, we propose a unifying perspective that shows that many algorithms in program analysis, verification, and automated reasoning that leverage duality, be it informally and intuitively or explicitly and formally, can be derived from a common principle. Moreover, this principle is connected to the classical notion of duality based on a Lagrangian used in linear programming [Bertsimas and Tsitsiklis 1997; Boyd and Vandenberghe 2014]. We leverage the Lagrangian perspective to study different characteristics of existing algorithms, and to develop a new algorithm for solving formulas in fixpoint logic over quantified linear arithmetic.

In linear programming, a Lagrangian $L(x, \lambda)$ is typically defined as a function from $\mathbb{R}^n \times \mathbb{R}^m$ to $\mathbb{R}$. The Lagrangian induces two related optimization problems: the *primal* problem is finding $\inf_x \sup_\lambda L(x, \lambda)$, and the *dual* problem is finding $\sup_\lambda \inf_x L(x, \lambda)$. It is easy to see that $\sup_\lambda \inf_x L(x, \lambda) \leq \inf_x \sup_\lambda L(x, \lambda)$, which is known as *weak duality* (*strong duality* is when an equality holds). That is, the solution to the dual problem provides a lower bound on the solution of the primal problem. Moreover, for any λ, the value of $\inf_x L(x, \lambda)$ provides such a lower bound. Similarly (dually), the value of $\sup_\lambda L(x, \lambda)$ for any x provides an upper bound on the dual problem. These properties are used in algorithm design to simultaneously search for solutions to the primal and dual problems, where both searches guide each other.

We propose to define a Lagrangian $L(x, y)$ from $X \times Y$ for arbitrary sets X and Y to a totally-ordered complete lattice, such that the induced primal and dual problems correspond, intuitively, to checking validity of a candidate proof and checking validity of a candidate refutation. Unlike in the linear programming case, the sets X and Y that arise in verification contexts are not real vector spaces but discrete sets or lattices (e.g, powerset lattices of sets of states or abstract domains).

For example, we can define a Lagrangian that captures the classic CEGAR algorithm as follows. The set X represents possible (abstract) counterexample traces, i.e, X is the set of sequences of states. The set Y represents possible abstractions, i.e., Y is the set of finite sets of predicates. We define the Lagrangian $L((s_0 s_1 \cdots s_n), \{p_0, p_1, \ldots, p_m\})$ to be -1 if the sequence $(s_0 s_1 \cdots s_n)$ is a counterexample for the abstract transition system defined by the abstraction to $\{p_0, p_1, \ldots, p_m\}$, and 1 otherwise. That is, the Lagrangian is -1 if for every $0 < i < n$ there is a transition from a state s to a state t such that on all the predicates in $\{p_0, p_1, \ldots, p_m\}$, s agrees with s_i and t agrees

with s_{i+1}. With this definition of a Lagrangian, safety is captured by $\sup_y \inf_x L(x, \lambda) = 1$, which holds if there exists a set of predicates y such that for every sequence of states x, $L(x, y) = 1$, i.e., x is not an abstract counterexample in the abstract transition system induced by y.

We further propose an algorithmic framework that given a Lagrangian, tries to solve the primal and dual problems simultaneously with primal-dual guidance between both sides of the duality. The basic algorithm attempts to find $\alpha \in X$ such that $\sup_y L(\alpha, y)$ is small enough, providing an upper bound on the primal solution, or $\beta \in Y$ such that $\inf_x L(x, \beta)$ is large enough, providing a lower bound on the dual problem. For example, in the case of CEGAR, if $\inf_x L(x, \beta)$ is 1 for some set of predicates β, so is the solution of the dual problem, and the system is safe. Starting from some candidates α and β, as long as the aforementioned conditions do not hold, the algorithm iteratively updates α to some α' that witnesses that $\inf_x L(x, \beta)$ is not large enough (i.e., $L(\alpha', \beta)$ is not large enough); or updates β to β' that witnesses that $\sup_y L(\alpha, y)$ is not small enough (i.e., $L(\alpha, \beta')$ is not small enough). That is, the update of the primal candidate α is guided by the dual problem and vice versa. For CEGAR, given a set of predicates β, computing a witness for $\inf_x L(x, \beta) \not\geq 1$, i.e., finding α' such that $L(\alpha', \beta) = -1$, corresponds to finding an abstract counterexample for the abstraction β defines; and for a given sequence α, finding β' such that $L(\alpha, \beta') = 1$ corresponds to finding an abstraction that rules out α as an abstract counterexample, i.e., an abstraction refinement step.

Often, X or Y (or both) are join semilattices, in which case it is possible to increase α or β (or both) in each iteration by joining their new value with the previous one. For example, in the case of the Lagrangian that captures CEGAR, the set Y of possible abstractions (sets of predicates) is a join semilattice, and increasing $\beta \in Y$ corresponds to "accumulating" predicates, as opposed to starting from scratch with a new set of predicates in each iteration. We prove that if the Lagrangian is (anti-)monotone in X or Y the "accumulation" of values ensures that the primal-dual algorithm makes progress in the sense of never generating the same α or β more than once. This is the case for CEGAR, where the Lagrangian is monotone on Y.

We show that many existing algorithms for a variety of problems can be described as instances of our framework with a suitable Lagrangian. For safety verification, we provide Lagrangians that capture CEGAR [Ball et al. 2001a; Clarke et al. 2000] (as described above), ICE learning [Garg et al. 2014], and primal-dual Houdini [Padon et al. 2022]. We show that our Lagrangian-based algorithmic framework can also capture algorithms for termination verification [Podelski and Rybalchenko 2004], and for solving quantified linear real arithmetic (LRA) formulas [Farzan and Kincaid 2016].

The Lagrangians succinctly capture the corresponding algorithms, and make the primal-dual interaction explicit. Furthermore, they allow us to compare the different algorithms. For example, our Lagrangian-based framework makes it clear that CEGAR makes progress by accumulating on the proof side, ICE makes progress by accumulating on the counterexample side, and primal-dual Houdini accumulates on both sides.

In some cases, the Lagrangians shed new light on some of the ingredients of existing techniques. For example, for termination verification, the Lagrangian perspective exposes that the disjunction found in disjunctively well-founded transition invariants [Podelski and Rybalchenko 2004] provides a mechanism to accumulate and make monotonic progress on the proofs side for termination proofs. For solving quantified LRA formulas, we see that the strategy skeletons of [Farzan and Kincaid 2016] play a similar role in enabling monotonic progress.

Another interesting aspect exposed by our unifying framework is the symmetry or asymmetry between the primal and dual problems induced by the Lagrangian. For example, in primal-dual Houdini and in quantified LRA solving, the primal and dual problems are symmetric in the sense that they have an identical high-level structure. In contrast, in CEGAR and ICE the two sides of the duality have different structures.

The Lagrangians also provide a new perspective on the cases where the algorithms captured by them fail, due to a connection between strong duality and the ability of the generic primal-dual algorithm to terminate. Namely, in cases where the primal and dual solutions do not coincide (*i.e.*, strong duality does not hold), the primal-dual algorithm diverges. In the kinds of problems we consider, strong duality often holds for an "idealized" version of the problem, where sets of states can be infinite, or where any set of states can be captured by a predicate, but is lost when restrictions are imposed on the values in X and Y. For example, in the case of CEGAR, this may be the case if the target system is safe, but the set of predicates considered is not sufficient to prove its safety. Thus, from the Lagrangian perspective, the loss of precision or expressiveness that prevents termination is captured by the loss of strong duality.

Equipped with Lagrangian based framework and the insights it provides, we go on to develop a new primal-dual algorithm for solving formulas in fixpoint logic over quantified linear arithmetic. Our new algorithm combines ideas from the aforementioned algorithms for termination verification and quantified LRA solving. We derive it cleanly by defining a Lagrangian that combines elements from the Lagrangians of these prior algorithms. A prototype implementation of this new algorithm achieves promising results and solves some instances that state-of-the-art algorithms do not.

In summary, this paper makes the following contributions:

(1) We define the notions of a Lagrangian and the primal and dual optimization problems it induces in a way that generalizes the classic Lagrangian from linear programming and is general enough for the setting of verification algorithms.
(2) We present a primal-dual search procedure that is parameterized by a Lagrangian and attempts to conclude either an upper bound on the value of the primal problem or a lower bound on the value of the dual problem. We provide a sufficient condition that ensures that the procedure makes progress.
(3) We demonstrate that with suitable Lagrangians, several existing algorithms for a variety of program analysis and automated reasoning problems can be seen as instances of our framework, and some of their characteristics can be explained in terms of the Lagrangians.
(4) We use our framework to derive a new primal-dual algorithm for solving formulas in fixpoint logic over quantified linear arithmetic. We present an initial empirical evaluation that shows the promise of the new algorithm.

The rest of the paper is organized as follows. Section 2 recalls the classical notion of a Lagrangian from linear programming, and introduces our generalization that is applicable to program verification as well as an abstract primal-dual search algorithm that is parameterized by a Lagrangian. Section 3 considers the safety verification problem and derives CEGAR and ICE learning as instances of our framework. Section 4 derives primal-dual Houdini [Padon et al. 2022] as an instance of our framework with a suitable Lagrangian. Section 5 formalizes algorithms for termination verification based on disjunctive well-founded ranking functions [Podelski and Rybalchenko 2004] as instances of the primal-dual procedure with a suitable Lagrangian, and Section 6 introduces a Lagrangian that captures the algorithm of [Farzan and Kincaid 2016] for solving quantified linear real arithmetic (LRA) formulas. Section 7 presents our new algorithm for fixpoint logic over quantified linear arithmetic. We conclude the paper with a discussion of related work in Section 8.

2 Lagrange Duality for Linear Programming and Verification

Duality is a fundamental and useful concept in linear programming problems, and this section aims to generalize this concept to verification problems. However, there are various differences between linear programming problems and verification problems. For example, in a linear programming problem, both the values to be controlled and to be optimized are continuous, whereas in a

verification problem, both are often discrete. This section introduces the Lagrange duality in a general form applicable to both linear programming and verification problems. We also provide a basic procedure to solve the optimization problem with a progress property.

2.1 Duality in Linear Programming

We briefly review the duality in linear programming. Let $\mathbb{R}$ be the set of reals and $\mathbb{R}_{\geq 0}$ be the subset of non-negative reals. For vectors $x, y \in \mathbb{R}^n$, we write $\langle x, y \rangle$ for the inner product and $x \leq y$ to mean $x_i \leq y_i$ for every i. For a matrix $A \in \mathbb{R}^{n \times m}$, its transpose is written as A^T.

Consider the following linear program, which we shall call the *primal* problem:[1]

$$\begin{array}{ll} \text{minimize} & \langle c, x \rangle, \qquad x \in \mathbb{R}^n_{\geq 0}, \\ \text{subject to} & Ax = b, \end{array}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{n \times m}$ are constants. We say $x \in \mathbb{R}^n_{\geq 0}$ is *feasible* if $Ax = b$. A feasible x gives an upper bound $\langle c, x \rangle$ of the optimal value, but a way to obtain a lower bound is not obvious.

The *dual* problem is a useful tool to overcome the situation. It is defined as:

$$\begin{array}{ll} \text{maximize} & \langle \lambda, b \rangle, \qquad \lambda \in \mathbb{R}^m, \\ \text{subject to} & A^T \lambda \leq c. \end{array}$$

An important point is that the optimal value of this dual problem coincides with the optimal value of the primal problem described above. So λ satisfying $A^T \lambda \leq c$ provides a lower bound $\langle \lambda, b \rangle$ of the optimal value of both the primal and dual problems. It is often the case that solving the primal problem takes a long time but the dual problem can be solved quickly (and vice versa). Linear programming algorithms often leverage this duality by exchanging information between the primal and the dual problems.

The primal and dual problems are connected by the *Lagrangian*, a function on x and λ given by

$$L(x, \lambda) \quad := \quad \langle c, x \rangle - \langle \lambda, Ax - b \rangle, \qquad x \in \mathbb{R}^n_{\geq 0},\ \lambda \in \mathbb{R}^m.$$

This is obtained by changing the *hard constraint* $Ax = b$ to a *soft constraint*. The constraint is violated if $Ax - b$ is non-zero for some element, say $(Ax - b)_i \neq 0$, and then the objective function is increased by $-\lambda_i (Ax - b)_i$. Since this penalty can be made arbitrarily large by choosing λ_i to have the same sign as $(Ax - b)_i$ and a large absolute value, the worst choice of the weights λ for the penalty makes the objective function ∞ if the constraint $Ax = b$ is violated. So

$$\sup_{\lambda} L(x, \lambda) \quad = \quad \begin{cases} \langle c, x \rangle & \text{if } Ax = b \\ \infty & \text{if } Ax \neq b. \end{cases}$$

Hence, the optimal value to the original problem is equivalent to $\inf_x \sup_\lambda L(x, \lambda)$, and the optimal solution x^* achieves $\sup_\lambda L(x^*, \lambda) = \inf_x \sup_\lambda L(x, \lambda)$.

The Lagrangian also characterizes the dual problem. We have

$$\begin{aligned} L(x, \lambda) &= \langle c, x \rangle - \langle \lambda, Ax - b \rangle \\ &= \langle c, x \rangle - \langle \lambda, Ax \rangle + \langle \lambda, b \rangle \\ &= \langle c, x \rangle - \langle A^T \lambda, x \rangle + \langle \lambda, b \rangle \\ &= \langle c - A^T \lambda, x \rangle + \langle \lambda, b \rangle. \end{aligned}$$

[1] We implicitly assume that $\{x \in \mathbb{R}^n_{\geq 0} \mid Ax = b\}$ is non-empty and the optimal value is finite.

For a fixed λ, if $c - A^T\lambda$ has a negative element, say $(c - A^T\lambda)_i < 0$, by choosing large $x_i > 0$, the value of $L(x, \lambda)$ can be made arbitrary small. So

$$\inf_x L(x, \lambda) \quad = \quad \begin{cases} \langle \lambda, b \rangle & \text{if } A^T\lambda \leq c \\ -\infty & \text{if } A^T\lambda \not\leq c. \end{cases}$$

Hence $\sup_\lambda \inf_x L(x, \lambda)$ is the optimum value for the dual problem.

The coincidence of the primal and dual is expressed as

$$\inf_x \sup_\lambda L(x, \lambda) \quad = \quad L(x^*, \lambda^*) \quad = \quad \sup_\lambda \inf_x L(x, \lambda),$$

where x^* and λ^* are the optimal solutions to the primal and dual problems, respectively. Furthermore, optimal solutions x^* and λ^* are characterized by $\sup_\lambda L(x^*, \lambda) = L(x^*, \lambda^*) = \inf_x L(x, \lambda^*)$. This non-trivial but useful property is called the *strong duality*.

2.2 Generalized Lagrange Duality

In typical situations in linear programming and convex optimization, a Lagrangian takes vectors and returns a real. Vectors and reals have useful operations and properties, and the development of Lagrange duality exploits these operations and properties. On the contrary, the verification community mainly deals with logical expressions, state transition systems, *etc.*, which significantly differ from reals and vectors. This subsection develops the Lagrange duality that works for such situations with less structure.

The following definition is probably the minimum requirement, assuming only that the upper and lower bounds in the codomain of a Langrangian make sense.

Definition 2.1 (Lagrangian). A *Lagrangian* is a function

$$L \colon X \times Y \longrightarrow P$$

from sets X and Y to a totally-ordered complete lattice $P = (P, \leq)$. □

Similar to the case of linear programming, a Lagrangian $L \colon X \times Y \longrightarrow P$ induces primal and dual objective functions as well as the associated optimization problems.

Definition 2.2. Let $L \colon X \times Y \longrightarrow P$ be a Lagrangian. The *primal objective function* is $\alpha \mapsto \sup_{y \in Y} L(\alpha, y)$ and the *primal optimization problem* is to minimize the primal objective function, *i.e.* to compute $\inf_x \sup_y L(x, y)$. Similarly, the *dual objective function* is $\beta \mapsto \inf_{x \in X} L(x, \beta)$ and the *dual optimization problem* is to compute $\sup_y \inf_x L(x, y)$. □

Example 2.3. Let $L_1 \colon \mathbb{Z} \times \mathbb{Z} \longrightarrow \{-1, 1\}$ (with $-1 \leq 1$) be the Lagrangian defined by

$$L_1(x, y) := \begin{cases} -1 & \text{if } x \geq y \\ 1 & \text{if } x < y. \end{cases}$$

The primal objective function $\alpha \mapsto \sup_{y \in \mathbb{Z}} L_1(\alpha, y)$ is the constant function to 1, so the optimal outcome for the primal optimization problem is 1. The dual objective function $\beta \mapsto \inf_{x \in \mathbb{Z}} L_1(x, \beta)$ is the constant function to -1, so the optimal outcome for the dual optimization problem is -1. Let $L_2 \colon \mathbb{Z} \times (\mathbb{Z} \to \mathbb{Z}) \longrightarrow \{-1, 1\}$ be another Lagrangian defined by

$$L_2(x, y) := \begin{cases} -1 & \text{if } x \geq y(x) \\ 1 & \text{if } x < y(x). \end{cases}$$

The primal objective function $\alpha \mapsto \sup_{y \in (\mathbb{Z} \to \mathbb{Z})} L_2(\alpha, y)$ is the constant function to 1, so the optimal outcome for the primal optimization problem is 1. The dual objective function $\beta \mapsto \inf_{x \in \mathbb{Z}} L_2(x, \beta)$

is 1 if $\forall x \in \mathbb{Z}.x < \beta(x)$ and -1 otherwise. For example, its value on $\beta^*(x) = x + 1$ is 1. So the optimal outcome for the dual optimization problem is 1. □

As example 2.3 shows, the optimal values for the primal and dual problems do not necessarily coincide. However, even this very general setting enjoys a weak form of duality.

Lemma 2.4. *Every Lagrangian $L\colon X \times Y \longrightarrow P$ enjoys the weak duality:*

$$\sup_{y \in Y} \inf_{x \in X} L(x, y) \quad \leq \quad \inf_{x \in X} \sup_{y \in Y} L(x, y).$$

Proof. Since $L(x, y) \leq \sup_x L(x, y)$ for every x and y, we have $\inf_y L(x, y) \leq \inf_y \sup_x L(x, y)$ for every x. So $\sup_x \inf_y L(x, y) \leq \inf_y \sup_x L(x, y)$. □

Example 2.5. Recall the Lagrangian L_1 in example 2.3. We have

$$\sup_{y \in \mathbb{Z}} \inf_{x \in \mathbb{Z}} L_1(x, y) = -1 < 1 = \inf_{x \in \mathbb{Z}} \sup_{y \in \mathbb{Z}} L(x, y),$$

so the inequality in Lemma 2.4 can be strict. In general, for $L\colon X \times Y \longrightarrow \{-1, 1\}$, we have $\inf_x \sup_y L(x, y) = 1$ if and only if $\forall x.\exists y.L(x, y) = 1$ holds, and similarly $(\sup_y \inf_x L(x, y) = 1) \Leftrightarrow (\exists y.\forall x.L(x, y) = 1)$. So the weak duality is a mild generalization of the well-known fact $(\exists x.\forall y.\varphi(x, y)) \Rightarrow (\forall y.\exists x.\varphi(x, y))$. □

We say a Lagrangian $L\colon X \times Y \longrightarrow P$ enjoys *strong duality* if $\sup_y \inf_x L(x, y) = \inf_x \sup_y L(x, y)$. The Lagrangian L_2 in example 2.3 enjoys strong duality. The strong duality is a desirable and useful property, but we do not assume it.

In the examples in this paper, a typical situation is as follows. A Lagrangian $L\colon X \times Y \longrightarrow P$ has an "idealization" $L'\colon X' \times Y' \longrightarrow P$ satisfying $X \subseteq X'$ and $Y \subseteq Y'$, and the idealization L' enjoys the strong duality and its optimal value coincides with the answer to a problem of interest (*e.g.* whether a given system is safe). However, X' and Y' are often not computationally tractable sets such as the set $\mathcal{P}(States)$ of all subsets of an infinite set *States*. The Lagrangian L is a computably tractable approximation of L', typically obtained by replacing an intractable set (*e.g.* $\mathcal{P}(States)$) with a tractable one (*e.g.* the set $\mathcal{P}_{\mathrm{fin}}(States)$ of finite subsets or an appropriate set of logical formulas describing properties on *States*). This approximation, however, may lose desirable properties such as strong duality, so we do not assume the strong duality of L.

2.3 A Primal-Dual Procedure

In the typical setting in this paper, we want to know whether the optimal value of the primal (or dual) optimization problem of a given Lagrangian L is bounded by a specific value, which we write as 0. For example, for the Lagrangian $L_{\mathrm{CEGAR}}\colon X \times Y \longrightarrow \{-1, 1\}$ in Section 3.2, $\inf_x L_{\mathrm{CEGAR}}(x, \beta) = 1$ if and only if β witnesses the safety of the target system, so we are interested in whether the optimal value $\sup_y \inf_x L_{\mathrm{CEGAR}}(x, y)$ of the dual optimization problem exceeds 0 or not, regarding the codomain of L_{CEGAR} as $\{-1 < 0 < 1\}$. This subsection develops a general procedure to solve this problem.

Suppose that we would like to know whether $\sup_y \inf_x L(x, y) \geq 0$ for a given Lagrangian $L\colon X \times Y \longrightarrow \{-1, 1\}$. To affirmatively answer this question, it suffices to find a witness $\beta \in Y$ such that $\inf_x L(x, \beta) \geq 0$. Similarly, a negative answer is confirmed by finding $\alpha \in X$ such that $\sup_y L(\alpha, y) \leq 0$. To find a witness α or β, we start by choosing an arbitrary value $\beta_0 \in Y$ as a candidate for a witness β. If $\inf_x L(x, \beta_0) \geq 0$, we are done; otherwise, there exists $\alpha_1 \in X$ such that $L(\alpha_1, \beta_0) < 0$. Now α_1 can be a candidate for a negative witness α, so we check whether $\sup_y L(\alpha_1, y) \leq 0$. We are done if it is the case, and otherwise, $L(\alpha_1, \beta_1) > 0$ for some $\beta_1 \in Y$, which

is the next candidate for a positive witness. In this way, we can iteratively update the candidates of α and β against each other until we finally find a positive or negative witness.

While the above process might converge, it might also get stuck in a cycle. For example, if β_1 above is not a valid positive witness we will discover α_2 such that $L(\alpha_2, \beta_1) < 0$ as a counter-witness to β_1 and our next candidate negative witness. But it may be the case that $L(\alpha_2, \beta_0) \geq 0$, in which case we may start repeating ourselves. That is, our sequence of candidate positive and negative witnesses will be $\beta_0, \alpha_1, \beta_1, \alpha_2, \beta_0, \ldots$, iterating back and forth between two pairs of candidate positive and negative witnesses. One way to ensure progress, at least in the sense of not revisiting the same candidates, is to assume additional structure on X or Y (or possibly both). Suppose that X or Y is a join semilattice $(X, \sqsubseteq_X, \sqcup_X)$ or $(Y, \sqsubseteq_Y, \sqcup_Y)$ and that the Lagrangian L is anti-monotone on X or monotone on Y. Then, when updating the candidate positive or negative witness, rather than forgetting the previous candidate we can take the join of the previous candidate and the new counter-witness. The option of this monotonic update appears in Algorithm 1, lines 6 and 10. As we prove below, a monotonic update on either side is sufficient to prevent the search from getting stuck and ensures it keeps exploring new possible solutions (on both sides).

Algorithm 1 Primal-Dual Procedure

function PrimalDual(L)
1: let $\alpha \in X, \beta \in Y$
2: **while true do**
3: **if** $\inf_x L(x, \beta) \geq 0$ **then**
4: **return** (D, β)
5: let $\delta \in \{x \in X \mid L(x, \beta) < 0\}$
6: $\alpha \leftarrow \delta$ or $\alpha \leftarrow \alpha \sqcup_X \delta$
7: **if** $\sup_y L(\alpha, y) \leq 0$ **then**
8: **return** (P, α)
9: let $\gamma \in \{y \in Y \mid L(\alpha, y) > 0\}$
10: $\beta \leftarrow \gamma$ or $\beta \leftarrow \beta \sqcup_Y \gamma$

Algorithm 1 formalizes this idea. Algorithm 1 can be divided into two parts, namely lines 3–5 and lines 7–9. The former checks whether the current β is a positive witness and, if it is not the case, produces a counter δ. The latter is the dual of the former, checking whether α is a negative witness and producing a counter γ. We call the former the *dual witness check* and the latter the *primal witness check* (recall that the primal optimization problem is minimization, and the dual is maximization). Algorithm 1 does not tell us how to implement the subprocedures to solve these subproblems and leaves the choice in lines 5 and 9 nondeterministic.

Remark 2.6. There are similar variants of Algorithm 1. E.g., rather than checking the dual witness first, it is possible to start with the primal witness check. Also, instead of computing $\gamma \in \{y \in Y \mid L(\alpha, y) \not\leq 0\}$ and updating $\beta \leftarrow \beta \sqcup \gamma$, we can directly update β to an element from $\{y \in Y \mid L(\alpha, y) \not\leq 0, y \geq \beta\}$. We will refer to Algorithm 1 and these variants collectively as the *basic primal-dual procedure.* □

Remark 2.7. Developing a practical procedure based on Algorithm 1 often requires optimizations. Heuristics for choosing a "good" δ or γ on lines 5 or 9 can have a significant impact on performance. One could also consider managing additional information beyond α and β to solve the primal and dual witness check problems faster. Algorithm 1 just describes a skeleton of practical procedures. □

Partial Correctness and Progress. The partial correctness (i.e., correctness assuming termination) and progress properties of Algorithm 1 are formalized in the following theorems.

Theorem 2.8. *PrimalDual in Algorithm 1 is correct in the following sense:*

- *If* $(\mathsf{P}, \alpha) =$ *PrimalDual*(L)*, then* 0 *is an upper bound of the optimal value of the primal optimization problem and* α *is a witness:* $\inf_x \sup_y L(x, y) \leq \sup_y L(\alpha, y) \leq 0$.
- *If* $(\mathsf{D}, \beta) =$ *PrimalDual*(L)*, then* 0 *is a lower bound of the optimal value of the dual optimization problem and* β *is a witness:* $\sup_y \inf_x L(x, y) \geq \inf_x L(x, \beta) \geq 0$.

Proof. If the procedure returns (P, α), then the value α has passed the condition in line 7, hence $\sup_y L(\alpha, y) \leq 0$. The case of (D, β) is similar. □

Theorem 2.9. *Let $L\colon X \times Y \longrightarrow P$ be a Lagrangian. Assume that X or Y is a join semilattice, and L is anti-monotone on X or monotone on Y. Then,* PrimalDual *in Algorithm 1 enjoys progress, i.e. it does not assign the same value to α or β twice, provided that we always choose $\alpha \leftarrow \alpha \sqcup_X \delta$ in line 6 (or that we always choose $\beta \leftarrow \beta \sqcup_Y \gamma$ in line 10).*

Proof. We focus on the case where Y is a join semilattice and β is updated monotonically (the other case is similar). We write α_n, β_n and γ_n for the values at the n-th iteration of the loop. Assume for contradiction that $\alpha_i = \alpha_j$ for some $i < j$. Then $L(\alpha_i, \gamma_i) \not\leq 0$. Since $\gamma_i \leq \beta_i \leq \beta_{j-1}$, we have $L(\alpha_i, \beta_{j-1}) \not\leq 0$ by the monotonicity of L on Y. We have $L(\alpha_j, \beta_{j-1}) \not\geq 0$ by the definition of α_j, a contradiction. We show that $\beta_i \neq \beta_j$ for every $i \neq j$. Since $\beta_i \leq \beta_{i+1} \leq \cdots \leq \beta_j$ for $i \leq j$, it suffices to show that $\beta_i < \beta_{i+1}$, or equivalently that $\beta_i \neq \beta_{i+1}$. Assume for contradiction that $\beta_i = \beta_{i+1}$. By the definition of β_{i+1}, we have $L(\alpha_{i+1}, \beta_{i+1}) \not\leq 0$. So $L(\alpha_{i+1}, \beta_i) \not\leq 0$ since $\beta_i = \beta_{i+1}$. By the definition of α_{i+1}, we have $L(\alpha_{i+1}, \beta_i) \not\geq 0$, a contradiction. □

Termination. Unlike partial correctness and progress, termination of Algorithm 1 is generally not guaranteed (except for trivial cases where X or Y are finite). This is because Algorithm 1 is a general procedure applicable to a variety of verification problems, many of which are undecidable. Our Lagrangian framework, however, provides a necessary condition for termination, relating termination and strong duality.

Theorem 2.10. *Assume $L\colon X \times Y \longrightarrow \{-1, 1\}$ does not enjoy the strong duality property,* i.e., $\inf_x \sup_y L(x, y) = 1$ *while* $\sup_y \inf_x L(x, y) = -1$. *Then Algorithm 1 cannot terminate. A similar claim holds if 0 is in the duality gap in the sense that*

$$\sup_y \inf_x L(x, y) < 0 < \inf_x \sup_y L(x, y).$$

Proof. Since $\sup_y L(\alpha, y) > 0$ and $\inf_x L(x, \beta) < 0$ for every $\alpha \in X$ and $\beta \in Y$, the conditions in lines 3 and 7 are never met. □

A typical example of the above situation is CEGAR satisfying the following conditions: the target system is safe, and the set of available predicates is strong enough to refute individual suspicious error traces but not enough to prove the safety of the entire system.

Example 2.11. Algorithm 1 may diverge, even for a Lagrangian with strong duality. Let $\mathbb{Z}_\infty := \mathbb{Z} \cup \{\infty\}$ and $L_3\colon \mathbb{Z} \times \mathbb{Z}_\infty \longrightarrow \{-1, 1\}$ be the Lagrangian defined by $L_3(x, y) = 1 \Leftrightarrow (x < y)$. Note that $\inf_x L_3(x, \infty) = 1$ but $\inf_x L_3(x, \beta) = -1$ for every $\beta \neq \infty$. So the loop continues until β becomes ∞. However, one can always choose a finite γ in line 9, resulting in divergence of the procedure. □

Let us now turn our attention to the opposite, *i.e.* ideas to ensure termination in some situations. Assume a Lagrangian $L\colon X \times Y \longrightarrow \{-1, 1\}$ with a positive witness β satisfying $\inf_x L(x, \beta) = 1$. *Stratification* [Jhala and McMillan 2006; Unno et al. 2021] is a technique to ensure the procedure eventually finds a positive witness (possibly different from β). The idea is to decompose $Y = \biguplus_{n \in \mathbb{N}} Y_n$ into an infinite union of finite sets, and to force the procedure to find $\gamma \in Y_l$ with minimum l (in line 9).

By combining monotonicity as formulated in Theorem 2.9 with stratification we can ensure termination, assuming a suitable positive or negative witness exists. The conditions for termination vary between the case where monotonicity and stratification are applied at the same side (i.e., both to X or to Y) or at different sides (i.e., X is a semilattice with monotonicity and Y is stratified, or vice versa). Below we analyze the case where X is a semilattice and Y is stratified and then the case where Y is a semilattice and also stratified. The other two cases are dual.

Theorem 2.12. *Let $L\colon X \times Y \longrightarrow \{-1,1\}$ be a Lagrangian where X is a join semilattice with L anti-monotone on X, and Y is stratified, i.e., $Y = \biguplus_{n\in\mathbb{N}} Y_n$ such that Y_n is finite for each n. Assume that L has a positive witness β^*,* i.e. $\inf_x L(x,\beta^*) = 1$. *Then Algorithm 1 terminates provided that γ in line 9 is chosen from the smallest possible layer,* i.e. $\gamma \in Y_l$ *with* $l = \min\{k \in \mathbb{N} \mid \exists \gamma \in Y_k. L(\alpha,\gamma) > 0\}$.

Proof. Assume $\beta^* \in Y_n$. Then $Y_{\leq n} := \bigcup_{i=0}^{n} Y_i$ is a finite set. Algorithm 1 terminates before $|Y_{\leq n}|$ iterations because of progress (Theorem 2.9) and because $\beta \in Y_{\leq n}$ is an invariant of the procedure, since β^* is always a possible choice in line 9 and we choose γ from the smallest possible layer. □

Theorem 2.12 guarantees termination (when a positive witness exists) but imposes an additional condition on the choice of γ, making the implementation of line 9 more difficult. It is therefore important to design a stratification for which the modified line 9 can be implemented efficiently.

Stratification and monotonicity can also be on the same side (i.e., X or Y). In a stratification of a semilattice the layers need not be finite, only of finite height, and they must be closed under join.

Theorem 2.13. *Let $L\colon X \times Y \longrightarrow \{-1,1\}$ be a Lagrangian and assume that Y is a join semilattice and L is monotone on Y. Assume a join semilattice homomorphism $r\colon Y \longrightarrow \mathbb{N}$,* i.e. $r(y \sqcup y') = \max(r(y), r(y'))$ *and consider the induced stratification $Y_n := \{y \in Y \mid r(y) = n\}$. Suppose that each Y_n is of finite height, i.e., it has no infinite increasing chain $y_0 \sqsubset y_1 \sqsubset \cdots \in Y_n$. We assume that the procedure always chooses $\beta \leftarrow \beta \sqcup_Y \gamma$ in line 10 and that γ in line 9 is chosen from the smallest possible layer. Then the procedure terminates provided that there exists a positive witness β^*.*

Proof. Similar to that of Theorem 2.12, combining progress and the invariant $r(\beta) \leq r(\beta^*)$. □

Remark 2.14. Although we were only interested in the sign of the optimal outcome of a Lagrangian, seeking the optimal outcome can also be useful for verification in some contexts. In this setting, it is natural to choose γ in line 9 in Algorithm 1 from those achieveing the best outcome against α (*i.e.* $L(\alpha,\gamma) = \max\{L(\alpha,y) \mid y \in Y\}$). A Lagrangian $L\colon X \times Y \longrightarrow \{-1,1\}$ with a stratification $Y = \biguplus_{n\in\mathbb{N}} Y_n$ induces the *stratified Lagrangian* $\widehat{L}\colon X \times Y \longrightarrow \mathbb{R}$ defined by

$$\widehat{L}(x,y) \quad := \quad \begin{cases} -1 & \text{if } L(x,y) = -1 \\ 1/n & \text{if } L(x,y) = 1 \text{ and } y \in Y_n. \end{cases}$$

The procedure seeking the optimal outcome of $\widehat{L}$ is the stratified version of Algorithm 1 mentioned in Theorems 2.12 and 2.13. □

Summary: Requirements for Applying our Framework. We conclude this section by summarizing the process to apply Algorithm 1.

- Develop a Lagrangian $L\colon X \times Y \longrightarrow P$ with the following conditions:
  - **Soundness**: At least one of $\inf_x \sup_y L(x,y) \leq 0$ and $\sup_y \inf_x L(x,y) \geq 0$ is related to the problem we would like to solve.
  - **Monotonicity**: At least one of X or Y is a join-lattice, and L is (anti-)monotone on that component.
- Develop solvers for both the primal and dual witness check problems. Possibly design heuristics for the nondeterministic choices in lines 5 and 9 that lead to good generalization in practice.

Remark 2.15. Hereafter, we assume that the codomain P of a Lagrangian is a *finite subset of* $\mathbb{Z}$ with the order inherited from $\mathbb{Z}$. All the examples in this paper satisfy this assumption. □

3 Lagrangians for Safety Verification: CEGAR and ICE

This section provides the first examples of Lagrangians in verification. We provide two Lagrangians for the safety verification problem, which induce CEGAR and ICE.

3.1 Problem setting

This section focuses on the *safety verification problem* of a given transition system, which is a problem asking if one can reach a bad state in a given transition system. We give a formal definition.

Definition 3.1. A *transition system* $\mathbb{S}$ is a tuple $(|\mathbb{S}|, I_{\mathbb{S}}, \rightsquigarrow_{\mathbb{S}}, B_{\mathbb{S}})$ where $|\mathbb{S}|$ is the set of *states*, $I_{\mathbb{S}} \subseteq |\mathbb{S}|$ is the *initial states*, $(\rightsquigarrow_{\mathbb{S}}) \subseteq |\mathbb{S}| \times |\mathbb{S}|$ is the *transition relation*, and $B_{\mathbb{S}} \subseteq |\mathbb{S}|$ is the set of *bad states*. A state $s \in |\mathbb{S}|$ is *reachable* if there exists a transition sequence $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} s_1 \rightsquigarrow_{\mathbb{S}} \dots \rightsquigarrow_{\mathbb{S}} s_n = s$ for some $n \geq 0$. The transition system $\mathbb{S}$ is *safe* if no bad state $s \in B_{\mathbb{S}}$ is reachable. Otherwise $\mathbb{S}$ is *unsafe*. □

Example 3.2. Let $\mathbb{S}_0$ be the transition system given by $|\mathbb{S}_0| = \mathbb{Z}$, $I_{\mathbb{S}_0} = \{0\}$, $(x \rightsquigarrow_{\mathbb{S}_0} y) \Leftrightarrow x + 1 = y$ and $B_{\mathbb{S}_0} = \{-3\}$. The set of reachable states is $\{n \in \mathbb{Z} \mid n \geq 0\}$, so $\mathbb{S}_0$ is safe. □

The unsafety of a transition system can be witnessed by a transition sequence $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} s_1 \rightsquigarrow_{\mathbb{S}} \dots \rightsquigarrow_{\mathbb{S}} s_n \in B_{\mathbb{S}}$ from an initial state s_0 to a bad state s_n. A safety proof cannot always be possible in such an obvious way. A common approach is to consider an appropriate set of *predicates* $\mathbb{P}$ and try to find a *safe inductive invariant*.

Definition 3.3. A *predicate set* over a transition system $\mathbb{S}$ is a set $\mathbb{P}$ together with a *satisfaction relation* $(\models) \subseteq |\mathbb{S}| \times \mathbb{P}$. When $(s, p) \in (\models)$, we write $s \models p$ and say that s *satisfies* p. A subset $X \subseteq |\mathbb{S}|$ is an *inductive invariant* if it satisfies the following conditions: (1) $I_{\mathbb{S}} \subseteq X$ (*initiation*); and (2) $(s \in X) \wedge (s \rightsquigarrow_{\mathbb{S}} s') \Longrightarrow (s' \in X)$ (*consecution*). If an invariant further satisfies (3) $B_{\mathbb{S}} \cap X = \emptyset$ (*safety*), we call it a *safe inductive invariant*. A predicate $p \in \mathbb{P}$ is a *(safe) inductive invariant* if so is $\{s \in |\mathbb{S}| \mid s \models p\}$. □

Example 3.4. Recall the transition system $\mathbb{S}_0$ in example 3.2. An inductive invariant is $p(x) :\Leftrightarrow (x \geq 0)$. There may be more than one inductive invariant for a system. For example, $p'(x) :\Leftrightarrow (x > -2)$ is another inductive invariant for $\mathbb{S}_0$. □

3.2 Lagrangian for CEGAR

Counter-Example Guided Abstraction Refinement (known as *CEGAR*) is a famous technique for the safety verification problem [Ball et al. 2001a; Clarke et al. 2000]. The CEGAR procedure manages a finite subset $A \subseteq \mathbb{P}$ of the predicate set, and it iterates the following two phases.

(a) Abstract the target transition system $\mathbb{S}$ by using the predicate set A and check its safety. If the abstract system is safe, then $\mathbb{S}$ is safe; otherwise, the abstract transition system has a transition sequence to a bad state.
(b) Check the feasibility of the abstract transition sequence to a bad state. If it is feasible, then $\mathbb{S}$ is unsafe; otherwise, add predicates to A that suffice to show the infeasibility of the abstract transition sequence.

The abstract transition system for a predicate set $A \subseteq \mathbb{P}$ is given as follows. The states $s, s' \in |\mathbb{S}|$ are *indistinguishable by* A, written $s \overset{A}{\approx} s'$, if $(s \models p) \Leftrightarrow (s' \models p)$ for every $p \in A$. A state of the abstract transition system is an equivalence class $[s]_A := \{ s' \in |\mathbb{S}| \mid s \overset{A}{\approx} s' \}$ of $\overset{A}{\approx}$. The abstract transition system has a transition from $[s_1]_A$ to $[s_2]_A$ if there exists a transition from a state in $[s_1]_A$ to a state in $[s_2]_A$ (*i.e.* $[s_1]_A \ni s_1' \rightsquigarrow_{\mathbb{S}} s_2' \in [s_2]_A$ for some s_1' and s_2'). An abstract state $[s]_A$ is an initial state (resp. a bad state) if it contains an initial state (resp. a bad state). We write $\mathbb{S}^A = (|\mathbb{S}|^A, I^A_{\mathbb{S}}, \rightsquigarrow^A_{\mathbb{S}}, B^A_{\mathbb{S}})$ for the abstract transition system.

Algorithm 2 CEGAR in the standard description (left) and as a primal-dual method (right)

let $A \leftarrow \emptyset$	let $A \leftarrow \emptyset$
while true do	**while true do**
if $\mathbb{S}^A$ is safe **then**	**if** $\inf_\tau L_{\mathrm{CEGAR}}(\tau, A) \geq 0$ **then**
return $(\mathtt{safe}, A)$	**return** $(\mathtt{safe}, A)$
let τ be an abstract error trace in $\mathbb{S}^A$	let $\tau \in \{\tau \in \lvert\mathbb{S}\rvert^* \mid L_{\mathrm{CEGAR}}(\tau, A) \not\geq 0\}$
if feasibility of τ is not refutable **then**	**if** $\sup_\alpha L_{\mathrm{CEGAR}}(\vec{s}, \alpha) \leq 0$ **then**
return $(\mathtt{unknown}, \tau)$	**return** $(\mathtt{unknown}, \tau)$
let α be an infeasibility witness for τ	let $\alpha \in \{\alpha \in Y \mid L_{\mathrm{CEGAR}}(\tau, \alpha) \not\leq 0\}$
$A \leftarrow A \cup \alpha$	$A \leftarrow A \cup \alpha$

CEGAR can be seen as an instance of the Lagrangian-based primal-dual method. Let

$$X := \lvert\mathbb{S}\rvert^* \quad \text{and} \quad Y := \mathcal{P}_{\mathrm{fin}}(\mathbb{P})$$

where $\lvert\mathbb{S}\rvert^*$ is the set of all finite sequences over $\lvert\mathbb{S}\rvert$ and $\mathcal{P}_{\mathrm{fin}}(\mathbb{P})$ is the set of all finite subsets of $\mathbb{P}$. We regard $Y = \mathcal{P}_{\mathrm{fin}}(\mathbb{P})$ as a poset ordered by set-inclusion $\subseteq$. The Lagrangian is given by

$$L_{\mathrm{CEGAR}}((s_0 s_1 \dots s_n), A) := \begin{cases} -1 & \text{if } I^A_{\mathbb{S}} \ni [s_0]_A \rightsquigarrow^A_{\mathbb{S}} [s_1]_A \rightsquigarrow^A_{\mathbb{S}} \cdots \rightsquigarrow^A_{\mathbb{S}} [s_n]_A \in B^A_{\mathbb{S}} \\ 1 & \text{otherwise.} \end{cases}$$

An "idealization" L'_{CEGAR} of L_{CEGAR} is obtained by setting $\mathbb{P}$ to be $\mathcal{P}(\lvert\mathbb{S}\rvert)$.

Proposition 3.5. *L'_{CEGAR} enjoys strong duality, and its optimal value is 1 if and only if $\mathbb{S}$ is safe.*

Proof. If $\mathbb{S}$ is unsafe, a concrete error trace $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} \dots \rightsquigarrow_{\mathbb{S}} s_n \in B_{\mathbb{S}}$ is an optimal choice of X. Otherwise, the set R of reachable states is a safe inductive invariant and $\{R\}$ is optimal for Y. □

Corollary 3.6. *If $\sup_y \inf_x L_{CEGAR}(x, y) = 1$, then $\mathbb{S}$ is safe.* □

If $\inf_x \sup_y L_{\mathrm{CEGAR}}(x, y) = -1$, then we know that Y is insufficient to prove the safety of $\mathbb{S}$ but not to say that $\mathbb{S}$ is unsafe. However, given α such that $\sup_y L_{\mathrm{CEGAR}}(\alpha, y) = -1$, we can examine it and may conclude that $\mathbb{S}$ is unsafe.

Corollary 3.6 ensures the soundness criterion. The monotonicity criterion is trivially met on Y. We examine the primal and dual witness check problems.

- The dual witness check asks to find τ such that $L_{\mathrm{CEGAR}}(\tau, A) < 0$. This is equivalent to the safety verification of the abstract transition system $\mathbb{S}^A$, and τ is an error trace.
- The primal witness check asks to find A such that $L_{\mathrm{CEGAR}}(\tau, A) > 0$. This is the so-called abstraction refinement and has well-known procedures.

So Algorithm 1 with L_{CEGAR} is applicable to solve the safety verification problem. Algorithm 2 describes the resulting procedure, which is the standard CEGAR procedure.

Remark 3.7. The stratified version of CEGAR [Jhala and McMillan 2006] can be understood in the Lagrangian framework as follows. In this setting, the set of predicates is stratified: $\mathbb{P} = \biguplus_{n \in \mathbb{N}} \mathbb{P}_n$. This induces a stratification on $Y = \mathcal{P}_{\mathrm{fin}}(\mathbb{P})$ as follows: $A \in \mathcal{P}_{\mathrm{fin}}(\mathbb{P})$ belongs to the n-th level where n is the minimum number such that $A \subseteq \bigcup_{i=0}^n \mathbb{P}_i$. By Theorem 2.12, the CEGAR procedure is guaranteed to terminate for instances with positive witnesses, as long as newly added predicates are always chosen from the smallest possible level. □

3.3 Lagrangian for ICE

The *ICE learning* [Garg et al. 2014] is another approach to the safety verification problem. It is described as an interaction between a *teacher* and a *learner*. The teacher tells a finite information on the system, and the leaner generates a candidate of satefy proof that works for the provided information. Then the teacher checks if the candidate is a proof against the whole system, and if not, tells an additional information on the system that explains why the candidate generated by the leaner is insufficient. Example 3.8 explains the interaction by an example.

Example 3.8. We prove the safety of $\mathbb{S}_0$ in example 3.2 by ICE. Information passed from the teacher to the learner is a triple $(I', \rightsquigarrow', B')$ of finite subsets of $I_{\mathbb{S}_0}$, $\rightsquigarrow_{\mathbb{S}_0}$ and $B_{\mathbb{S}_0}$, respectively. Let $S'_0 = (I'_0, \rightsquigarrow'_0, B'_0) = (\emptyset, \emptyset, \emptyset)$ be the initial choice. The learner proposes an inductive invariant for the subsystem S'_0: the subsystem S'_0 does not have a bad state, so $p_0(x) :\Leftrightarrow \top$ is an inductive invariant for the subsystem S'_0. Then the teacher checks if p_0 is an inductive invariant for $\mathbb{S}_0$, but it is obviously not. The teacher adds a bad state -3 to the subsystem, resulting in $S'_1 = (\emptyset, \emptyset, \{-3\})$. The leaner proposes an inductive invariant for the subsystem S'_1: the subsystem does not have an initial state, so $p_1(x) :\Leftrightarrow \bot$ is an inductive invariant for the subsystem S'_1. The teacher then checks if p_1 is an inductive invariant for $\mathbb{S}_0$, but it is not. The teacher adds an initial state 0 to the subsystem: $S'_2 = (\{0\}, \emptyset, \{-3\})$. The learner proposes an inductive invariant for the subsystem S'_2: one can choose $p_2(x) :\Leftrightarrow (x \bmod 2 = 0)$, which separates the initial state 0 from the bad state -3. The teacher then checks if p_2 is an inductive invariant for $\mathbb{S}_0$, but it is not because p_2 is not closed under the transition relation. The teacher adds some examples of transitions, yielding $S'_3 = (\{0\}, \{(0,1), (2,3), (8,9), (-4,-3)\}, \{-3\})$. Now, the learner has a much narrower range of candidates: an inductive invariant p_3 for S'_3 must be true on $0, 1$ and must be false on $-4, -3$. A candidate is $p_3(x) :\Leftrightarrow (x > -2)$. The teacher can ensure that p_3 is an inductive invariant for $\mathbb{S}_0$. □

Then ICE-learning can be seen as an instance of the Lagrangian-based primal-dual method. Let

$$X := \mathcal{P}_{\mathrm{fin}}(I_{\mathbb{S}}) \times \mathcal{P}_{\mathrm{fin}}(\rightsquigarrow_{\mathbb{S}}) \times \mathcal{P}_{\mathrm{fin}}(B_{\mathbb{S}}) \quad \text{and} \quad Y := \mathbb{P}.$$

Intuitively, X tells us partial information on the transition system. The Lagrangian is given by

$$L_{\mathrm{ICE}}(S', p) = \begin{cases} 1 & p \text{ is an inductive invariant for the subsystem } S' \text{ of } \mathbb{S} \\ -1 & \text{otherwise.} \end{cases}$$

So $L_{\mathrm{ICE}}(S', \varphi)$ is 1 if and only if φ witnesses the safety of the subtransition system S'. The "idealization" L'_{ICE} of L_{ICE} is obtained by setting $Y := \mathcal{P}(|\mathbb{S}|)$.

Proposition 3.9. *L'_{ICE} enjoys the strong duality, and its optimal value is 1 iff $\mathbb{S}$ is safe.*

Proof. If $\mathbb{S}$ is safe, an optimal choice of Y is the set of reachable states. If $\mathbb{S}$ is unsafe, an optimal choice of X is an error trace. □

Corollary 3.10. *If $\sup_y \inf_x L_{ICE}(x, y) = 1$, then $\mathbb{S}$ is safe.* □

The Lagrangian L_{ICE} satisfies both soundness and monotonicity criteria. Unlike in the CEGAR case, the X component has a join semilattice structure and L_{ICE} is anti-monotone on X. We examine the primal and dual witness checks.

- The dual witness check asks to find S' such that $L_{\mathrm{ICE}}(S', p) = -1$. This is the teacher.
- The primal witness check asks to find p such that $L_{\mathrm{ICE}}(S', p) = 1$. This is the learner.

So the variant of Algorithm 1 that monotonically updates α is applicable to this setting. Algorithm 3 describes the resulting procedure (which, unlike Algorithm 1, starts from the dual side).

Algorithm 3 ICE in the standard description (left) and as a primal-dual method (right)

let $S' \leftarrow (\emptyset, \emptyset, \emptyset)$	let $S' \leftarrow (\emptyset, \emptyset, \emptyset)$
while true do	**while true do**
if Safety of S' is unprovable **then**	**if** $\sup_p L_{\mathrm{ICE}}(S', p) = -1$ **then**
return $(\mathtt{unknown}, S')$	**return** $(\mathtt{unknown}, S')$
let p be a safety proof for S'	let $p \in \{p \in Y \mid L_{\mathrm{ICE}}(S', p) = 1\}$
if p is a safety proof for $\mathbb{S}$ **then**	**if** $\inf_{S''} L_{\mathrm{ICE}}(S'', p) = 1$ **then**
return $(\mathtt{safe}, p)$	**return** $(\mathtt{safe}, p)$
let S'' be counterexamples for p	let $S'' \in \{S'' \in X \mid L_{\mathrm{ICE}}(S'', p) = -1\}$
$S' \leftarrow S' \cup S''$	$S' \leftarrow S' \cup S''$

4 Lagrangian for Primal-Dual Houdini

This section analyzes *primal-dual Houdini* [Padon et al. 2022], a procedure that motivated this paper, in terms of the Lagrange duality. The Lagrangian for primal-dual Houdini yields monotone progress on both sides of the duality; it is symmetric in the sense that the primal and dual problems have the same high-level structure and can be swapped, changing the sign of the Lagrangian; and interestingly showing that the Lagrangian is well-defined requires a non-trivial lemma.

4.1 Warmup: CEGAR for Cartesian Abstraction

As a stepping stone towards developing a Lagrangian for primal-dual Houdini, we first consider a version of CEGAR that operates not on predicate abstraction as in Section 3.2, but on Cartesian abstraction [Ball et al. 2001b]. Given a set of predicates P, the *Cartesian* or *conjunctive* abstraction it induces considers inductive invariants that can be expressed as a conjunction of predicates from P. This abstraction is coarser than the Boolean or predicate abstraction, which considers invariants that can be expressed as an arbitrary Boolean formula over predicates from P (i.e., using conjunction, disjunction, and negation). While in the case of predicate abstraction the abstract counterexample takes the form of an abstract trace (i.e., a sequence of transitions), in the case of Cartesian abstraction the abstract counterexample can be thought of as a directed acyclic graph (DAG) of transitions. We abstract over the details of the DAG structure, and focus just on the set of states that participate in the abstract counterexample.

As in Section 3, let us fix a transition system $\mathbb{S}$ (Definition 3.1) and a predicate set $\mathbb{P}$ (Definition 3.3), *i.e.* a set with a satisfaction relation $(\models) \subseteq |\mathbb{S}| \times \mathbb{P}$. To define a Lagrangian for CEGAR for Cartesian abstraction, we let

$$X = \mathcal{P}_{\mathrm{fin}}(|\mathbb{S}|) \quad \text{and} \quad Y = \mathcal{P}_{\mathrm{fin}}(\mathbb{P}).$$

We now define the value of $L_{\mathrm{CEGAR}^\wedge}(x, y)$ according to whether there is a conjunctive inductive invariant over predicates from y that proves that $\mathbb{S}|_x$ is safe, where $\mathbb{S}|_x$ is the transition system $\mathbb{S}$ *restricted* to states from x, i.e., $|\mathbb{S}|_x| = x$, $I_{\mathbb{S}|_x} = I_{\mathbb{S}} \cap x$, $\rightsquigarrow_{\mathbb{S}|_x} = \rightsquigarrow_{\mathbb{S}} \cap (x \times x)$, and $B_{\mathbb{S}|_x} = B_{\mathbb{S}} \cap x$. We define the Lagrangian $L_{\mathrm{CEGAR}^\wedge} : X \times Y \longrightarrow \{-1, +1\}$.

$$L_{\mathrm{CEGAR}^\wedge}(x, y) = \begin{cases} -1 & \text{if no } \vartheta \subseteq y \text{ is a safe inductive invariant of } \mathbb{S}|_x \\ +1 & \text{otherwise} \end{cases} \tag{1}$$

Given a finite set of predicates $y \in Y$, checking whether $\inf_x L_{\mathrm{CEGAR}^\wedge}(x, y) \geq 0$ (i.e., the dual witness check) amounts to checking whether $\mathbb{S}$ can be proven safe using the Cartesian abstraction defined by y. The well-known Houdini algorithm [Flanagan et al. 2001; Flanagan and Leino 2001] is a procedure for answering this question, and it operates by a straightforward fixpoint computation to

compute the strongest inductive invariant of $\mathbb{S}$ that can be expressed as a conjunction of predicates from y. The algorithm also computes a set of (at most $|y|$) transitions that show why no larger subset of y is an inductive invariant for $\mathbb{S}$; or, in our terms, it produces a set of states x such that no larger subset of y is an inductive invariant for $\mathbb{S}|_x$.

Given a finite set of states x, the primal witness check, i.e., checking if $\sup_y L_{\text{CEGAR}^\wedge}(x, y) \leq 0$, corresponds to finding the right predicates that can be used to prove safety of $\mathbb{S}|_x$ using Cartesian abstraction. That is, it corresponds to abstraction refinement, similarly to the primal witness check problem in Section 3.2.

4.2 Induction Duality

Let $\mathbb{S}$ be a transition system and $\mathbb{P}$ be a predicate set. For a subset $\vartheta \subseteq \mathbb{P}$ of predicates, $s \models \vartheta$ means $\forall p \in \vartheta.\ s \models p$. Similarly, for a subset $\varpi \subseteq |\mathbb{S}|$ of states, $\varpi \models p$ means $\forall s \in \varpi.\ s \models p$. Finally, $\varpi \models \vartheta$ means $s \models p$ for every $s \in \varpi$ and $p \in \vartheta$. Below we sometimes abuse the notation and use p either for a single predicate or for a finite set of predicates. We extend the definition of an inductive invariant (Definition 3.3) to finite sets of predicates by interpreting the sets conjunctively. That is, a finite set of predicates $\vartheta \subseteq \mathbb{P}$ is an inductive invariant of if its conjunction, denoted $\bigwedge \vartheta$, is an inductive invariant.

Primal-dual Houdini operates by analyzing $\mathbb{S}$ together with another transition system $\mathbb{I} = (|\mathbb{I}|, I_{\mathbb{I}}, \rightsquigarrow_{\mathbb{I}}, B_{\mathbb{I}})$ which is defined over finite sets of predicates, i.e., $|\mathbb{I}| = \mathcal{P}_{\text{fin}}(\mathbb{P})$. $\mathbb{I}$ is assumed to be an *induction dual* of $\mathbb{S}$, meaning that the following conditions are satisfied:[2]

(ID1) $s_0 \models p$ for every $s_0 \in I_{\mathbb{S}}$ and $p \in |\mathbb{I}|$, that is, the initial states satisfy all predicates;
(ID2) $s \models p_0$ for every $s \in |\mathbb{S}|$ and $p_0 \in I_{\mathbb{I}}$, that is, the initial predicates are satisfied by all states;
(ID3) $s \not\models p$ for every $s \in B_{\mathbb{S}}$ and $p \in B_{\mathbb{I}}$, that is, every bad state is excluded by every "bad" predicate set; and
(ID4) if $(s \rightsquigarrow_{\mathbb{S}} s') \wedge (p \rightsquigarrow_{\mathbb{I}} p') \wedge (s \models p) \wedge (s \models p') \wedge (s' \models p)$, then $(s' \models p')$, that is, transitions in $\rightsquigarrow_{\mathbb{S}}$ and in $\rightsquigarrow_{\mathbb{I}}$ restrict each other in a particular way.

The intuition behind this definition is that traces in $\mathbb{I}$ represents incremental induction proofs over $\mathbb{S}$. Specifically, (ID4) means that if we assume p is an invariant (of $\mathbb{S}$) and $p \rightsquigarrow_{\mathbb{I}} p'$, then p' is also an invariant. As the following lemma shows, a path in $\mathbb{I}$ implies an inductive invariant of $\mathbb{S}$.

Lemma 4.1 ([Padon et al. 2022]). *Assume induction-dual transition systems $\mathbb{S}$ and $\mathbb{I}$. If $I_{\mathbb{I}} \ni p_0 \rightsquigarrow_{\mathbb{I}} p_1 \rightsquigarrow_{\mathbb{I}} \ldots \rightsquigarrow_{\mathbb{I}} p_n$, then $\bigcup_{i=0}^{n} p_i$ is an inductive invariant for $\mathbb{S}$.*

Proof. Initiation follows from (ID1). For consecution, suppose $I_{\mathbb{I}} \ni p_0 \rightsquigarrow_{\mathbb{I}} p_1 \rightsquigarrow_{\mathbb{I}} \ldots \rightsquigarrow_{\mathbb{I}} p_n$ and assume $s \models p_0 \cup p_1 \cup \cdots \cup p_n$ and $s \rightsquigarrow_{\mathbb{S}} s'$. We prove $s' \models p_i$ for every $0 \leq i \leq n$ by induction on i. For $i = 0$, the claim follows from (ID2). For $i > 0$, since $s \models p_{i-1} \cup p_i$ and $s' \models p_{i-1}$, we conclude $s' \models p_i$ from (ID4)—the duality condition on transitions. □

Corollary 4.2. *For $\mathbb{S}$ and $\mathbb{I}$ that are induction dual, if some $p \in B_{\mathbb{I}}$ is reachable in $\mathbb{I}$ then $\mathbb{S}$ is safe.*

Proof. Follows from Lemma 4.1 and (ID3). □

Dually, a path $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} s_1 \rightsquigarrow_{\mathbb{S}} \ldots \rightsquigarrow_{\mathbb{S}} s_n$ in $\mathbb{S}$ induces an inductive invariant for $\mathbb{I}$, and reachability to a bad state in $\mathbb{S}$ implies safety of the dual transition system $\mathbb{I}$.

[2]This presentation of induction duality is a slight adaptation of [Padon et al. 2022]. For example, the original paper assumes that the only element in $I_{\mathbb{I}}$ is the empty set (representing $\top$), which (ID2) generalizes; it also assumes that $B_{\mathbb{S}}$ is represented by a single predicate $p_0 \in \mathbb{P}$, *i.e.* $B_{\mathbb{S}} = \{s \in |\mathbb{S}| \mid s \not\models p_0\}$, and that $B_{\mathbb{I}} = \{y \in \mathcal{P}_{\text{fin}}(\mathbb{P}) \mid p_0 \in y\}$, which (ID3) generalizes. The original paper also assumes that transitions in $\mathbb{I}$ only grow the set of predicates, which we do not assume here.

4.3 Lagrangian for Primal-Dual Houdini

We can understand primal-dual Houdini as a simultaneous CEGAR for Cartesian abstraction for both $\mathbb{S}$ and $\mathbb{I}$, where the dual witness check in CEGAR for $\mathbb{S}$ also solves the primal witness check for $\mathbb{I}$, and the dual witness check for $\mathbb{I}$ solves the primal witness check for $\mathbb{S}$. The fact that this can be done requires a non-trivial lemma about the connection between $\mathbb{S}$ and $\mathbb{I}$ (and their restrictions).

Formally, let

$$X = \mathcal{P}_{\text{fin}}(|\mathbb{S}|) \quad \text{and} \quad Y = \mathcal{P}_{\text{fin}}(\mathbb{P}).$$

The Lagrangian $L_{\text{pdH}} \colon X \times Y \longrightarrow \{-1, 0, 1\}$ is defined by

$$L_{\text{pdH}}(x, y) := \begin{cases} -1 & \text{if no } \vartheta \subseteq y \text{ is a safe inductive invariant of } \mathbb{S}|_x \\ 1 & \text{if no } \varpi \subseteq x \text{ is a safe inductive invariant of } \mathbb{I}|_y \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbb{S}|_x$ is the subsystem of $\mathbb{S}$ consisting of states in x and $\mathbb{I}|_y$ is the subsystem of $\mathbb{S}$ consisting of sets of predicates from y (i.e., more formally this can be written as $\mathbb{I}|_{\mathcal{P}(y)}$). We need a slightly advanced result of the dual transition system to understand this definition, as well as to confirm its well-definedness. The following theorem is new to this paper, an abstraction of the progress proof in Padon et al. [2022].

Theorem 4.3. *Assume dual transition systems $\mathbb{S}$ and $\mathbb{I}$. Then at least one of the following holds:*

- *some $\vartheta \subseteq \mathbb{P}$ is a safe inductive invariant for $\mathbb{S}$, or*
- *some $\varpi \subseteq |\mathbb{S}|$ is a safe inductive invariant for $\mathbb{I}$.*

Proof. See [Tsukada et al. 2024, Appendix A]. □

Theorem 4.3 shows that the conditions for $L_{\text{pdH}}(x, y) = 1$ and for $L_{\text{pdH}}(x, y) = -1$ do not hold simultaneously. So L_{pdH} given by eq. (2) is well-defined. Both X and Y are join semilattices, and L is anti-monotone in X and monotone in Y. So the general method applies to this Lagrangian, yielding a procedure in Algorithm 4 that enjoys the progress property. The resulting procedure is the core of primal-dual Houdini in [Padon et al. 2022].

Proposition 4.4. $\sup_y \inf_x L_{pdH}(x, y) \geq 0$ *implies the satefy of* $\mathbb{S}$.

Proof. If $\mathbb{S}$ is unsafe, there exists an error trace $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} \ldots \rightsquigarrow_{\mathbb{S}} s_n \in B_{\mathbb{S}}$. Letting $x = \{s_0, \ldots, s_n\}$, there is no safe inductive invariant of $\mathbb{S}\restriction_x$, so $\sup_y L(x, y) = -1$. □

Even if $\sup_y \inf_x L_{\text{pdH}}(x, y) = -1$, we cannot conclude that $\mathbb{S}$ is unsafe: it just shows that $\mathbb{P}$ is not sufficient to prove the safety of $\mathbb{S}$. Similar to the case of CEGAR and ICE, analysis of $S_G \in X$ satisfying $\sup_y L_{\text{pdH}}(S_G, y) \leq 0$ may provide an unsafety proof.

The Lagrangian L_{pdH} is symmetric in the sense that swapping $\mathbb{S}$ and $\mathbb{I}$ does not essentially change the Lagrangian. That means, writing $L^{\mathbb{S},\mathbb{I}}_{\text{pdH}}$ for the Lagrangian for the dual transition systems $(\mathbb{S}, \mathbb{I})$, we have $L^{\mathbb{S},\mathbb{I}}_{\text{pdH}}(x, y) = -L^{\mathbb{I},\mathbb{S}}_{\text{pdH}}(y, x)$. By this symmetry, the primal and dual witness check problems are essentially the same. Primal-dual Houdini [Padon et al. 2022] solves these subproblems by using the famous Houdini procedure [Flanagan et al. 2001; Flanagan and Leino 2001].

Theorem 4.3 also suggests a connection to CEGAR. By Theorem 4.3, if $L_{\text{pdH}}(x, y) = 1$ (*i.e.* no $\varpi \subseteq x$ is a safe inductive invariant of $\mathbb{I}|_y$), then some $\vartheta \subseteq y$ is a safe inductive invariant of $\mathbb{S}$. So $L_{\text{pdH}}(x, y) \geq 0$ if and only if some $\vartheta \subseteq y$ is a safe inductive invariant of $\mathbb{S}$. Therefore finding P such that $L_{\text{pdH}}(S_G, P) \geq 0$ is essentially the abstraction refinement problem, which asks to find a predicate set that proves the safety of an approximation of the transition system $\mathbb{S}$. In primal-dual Houdini (Algorithm 4), line 10 computes P that satisfies $L_{\text{pdH}}(x, y) \geq 1$ instead of $L_{\text{pdH}}(x, y) \geq 0$,

Algorithm 4 Primal-Dual Houdini

function Primal-Dual Houdini
1: let $S_G \leftarrow \emptyset$
2: let $P_G \leftarrow \{p\}$ for some $p \in B_{\mathbb{I}}$
3: **while true do**
4: **if** $\inf_S L_{\text{pdH}}(S, P_G) \geq 0$ **then**
5: **return** $(\texttt{safe}, P_G)$
6: let $S \in \{S \in X \mid L_{\text{pdH}}(S, P_G) = -1\}$
7: $S_G \leftarrow S_G \cup S$
8: **if** $\sup_P L_{\text{pdH}}(S_G, P) \leq 0$ **then**
9: **return** $(\text{unknown}, S_G)$
10: let $P \in \{P \in Y \mid L_{\text{pdH}}(S_G, P) = 1\}$
11: $P_G \leftarrow P_G \cup P$

and this is a requirement stronger than the standard abstraction refinement. This strategy for updating the predicate set is the characteristic feature of primal-dual Houdini.

4.4 A More Abstract View

Our exposition above was rather close to that of [Padon et al. 2022]. Specifically, we used sets of predicates for $\mathbb{I}$. The use of sets of predicates is consistent with the perspective of incremental induction and constructing invariants in a conjunctive domain. However, the Lagrangian formulation suggests that a more abstract structure is enough to define primal-dual Houdini, as follows.

In the more abstract form, we assume transition systems $\mathbb{S}$ and $\mathbb{I}$, but do not assume anything on the state space $|\mathbb{I}|$, i.e., we do not assume it consists of predicates or sets of predicates. The only assumption is that there is a binary relation between $|\mathbb{S}|$ and $|\mathbb{I}|$, and that the two transition systems are induction duals, that is, conditions (ID1)–(ID4) hold. This is very similar to the induction-dual graphs of [Padon et al. 2022, Section 3.1].

Next, we assume arbitrary join semilattices X and Y. That is, X is not assumed to be finite sets of states, and Y is not assumed to be finite sets of predicates. Instead, we assume a *restriction mapping*, which maps every element of X to a transition system $\mathbb{S}|_x$ that is a subsystem of $\mathbb{S}$ in the following sense: $|\mathbb{S}|_x| \subseteq |\mathbb{S}|$, $I_{\mathbb{S}|_x} = I_{\mathbb{S}} \cap |\mathbb{S}|_x|$, $\rightsquigarrow_{\mathbb{S}|_x} = \rightsquigarrow_{\mathbb{S}} \cap (|\mathbb{S}|_x| \times |\mathbb{S}|_x|)$, and $B_{\mathbb{S}|_x} = B_{\mathbb{S}} \cap |\mathbb{S}|_x|$. We further require that the restriction mapping is monotone, i.e., if $x \sqsubseteq_X x'$ then $|\mathbb{S}|_x| \subseteq |\mathbb{S}|_{x'}|$, and that the restriction is *unbounded* in the following sense: $\sup_x |\mathbb{S}|_x| = |\mathbb{S}|$. We assume a similar restriction mapping for Y and $\mathbb{I}$.

With this structure over the arbitrary sets X and Y, we can define the Lagrangian by the following generalization of eq. (2):

$$L_{\text{pdH}}(x, y) = \begin{cases} -1 & \text{if no } \vartheta \subseteq |\mathbb{I}|_y| \text{ is a safe inductive invariant of } \mathbb{S}|_x \\ 1 & \text{if no } \varpi \subseteq |\mathbb{S}|_x| \text{ is a safe inductive invariant of } \mathbb{I}|_y \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

It is not too hard to see that Theorem 4.3 generalizes to the setting eq. (3) and that this Lagrangian is well-defined. Note that the Lagrangian of Section 4.3 is obtained from the more abstract one by letting $|\mathbb{I}| = \mathcal{P}_{\text{fin}}(\mathbb{P})$, $X = \mathcal{P}_{\text{fin}}(|\mathbb{S}|)$, $Y = \mathcal{P}_{\text{fin}}(\mathbb{P})$, and defining $|\mathbb{S}|_x| = x$, and $|\mathbb{I}|_y| = \mathcal{P}(y)$ (note the difference in the definition of the restriction mapping between X and Y).

5 Lagrangians for Termination Verification

So far, we have focused on the safety verification problem. This section discusses how our Lagrangian-based approach applies to termination verification, presenting ICE-based and CEGAR-based procedures. Intuitively, the Lagrangians can be obtained by replacing predicates in the safety verification with ranking functions. This idea works well for ICE, but for CEGAR, a problem arises with the monotonicity requirement. From our perspective, the concept of *disjunctive well-foundedness* [Podelski and Rybalchenko 2004] is an idea to address the monotonicity requirement.

5.1 Termination and Ranking Function

Let $\mathbb{S} = (|\mathbb{S}|, I_{\mathbb{S}}, \rightsquigarrow_{\mathbb{S}})$ be a transition system. In this section, we omit the bad states $B_{\mathbb{S}}$ as this component is irrelevant to the verification problem in this section. An *infinite trace* is an infinite sequence $s_0 s_1 s_2 \cdots \in |\mathbb{S}|^\omega$ of states such that $s_0 \in I_{\mathbb{S}}$ and $s_i \rightsquigarrow_{\mathbb{S}} s_{i+1}$ for every i. A transition system is *terminating* if it has no infinite trace.

The termination of a transition system $\mathbb{S}$ can be witnessed by a *ranking function*. In this paper, it is a function $r\colon |\mathbb{S}| \longrightarrow \kappa$ from states to an ordinal number κ such that $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}}^* s \rightsquigarrow_{\mathbb{S}} s'$ implies $r(s) > r(s')$ (readers who are not familiar with ordinal numbers may set $\kappa = \mathbb{N}$). So r must decrease for every reachable transition (but a transition $s \rightsquigarrow_{\mathbb{S}} s'$ from an unreachable state s does not require anything on r). The existence of a ranking function is a sound and complete criterion for termination.

5.2 ICE for Termination

It is straightforward to modify the Lagrangian for the safety verification in Section 3.3 to handle the termination verification. Let $\mathfrak{R} \subseteq (|\mathbb{S}| \to \mathbb{N})$ be a set of candidates for ranking functions, and let

$$X := \mathcal{P}_{\mathrm{fin}}(I_{\mathbb{S}}) \times \mathcal{P}_{\mathrm{fin}}(\rightsquigarrow_{\mathbb{S}}) \quad \text{and} \quad Y := \mathfrak{R}.$$

The X-component is the same as the X-component for ICE (but the bad-state information is omitted). The Y-component is the set of candidate proofs, which in the case of termination verification is the set of candidate ranking functions. (Recall that in the case of safety verification Y is the set of predicates, thus in termination verification ranking functions assume the role of predicates in safety verification.) The Lagrangian $L_{\text{T-ICE}}\colon X \times Y \longrightarrow \{-1, 1\}$ is defined by

$$L_{\text{T-ICE}}(S', r) = \begin{cases} 1 & r \text{ is a ranking function for the subsystem } S' \text{ of } \mathbb{S} \\ -1 & \text{otherwise.} \end{cases}$$

The X-component has an obvious join semilattice structure, and $L_{\text{T-ICE}}$ is anti-monotone on X. So the basic primal-dual procedure is applicable, provided that the primal and dual witness check problems are tractable.

We now discuss soundness. An "idealized" version $L'_{\text{T-ICE}}\colon X \times Y' \longrightarrow \{-1, 1\}$ of $L_{\text{T-ICE}}$ is given by setting $X' := \mathcal{P}(I_{\mathbb{S}}) \times \mathcal{P}(\rightsquigarrow_{\mathbb{S}})$ and $Y' := (|\mathbb{S}| \to \kappa)$ for a sufficiently large ordinal number κ.[3] We define $L'_{\text{T-ICE}}(S', r) = 1$ if and only if r is a ranking function for the subsystem S' of $\mathbb{S}$.

Proposition 5.1. *$L'_{\text{T-ICE}}$ enjoys strong duality, and $\inf_x \sup_y L'_{\text{T-ICE}}(x, y) = 1$ iff $\mathbb{S}$ terminates.*

Proof. If $\mathbb{S}$ terminates, a ranking function r for $\mathbb{S}$ is given by

$$\begin{aligned} r(s) &= \text{(The number of steps remaining before termination)} \\ &= \sup\{r(s') + 1 \mid s \rightsquigarrow_{\mathbb{S}} s'\}. \end{aligned}$$

[3] One can choose κ as the minimum ordinal that has the same cardinality as $\mathcal{P}(|\mathbb{S}|)$.

This is a definition by induction on $\rightsquigarrow_{\mathbb{S}}$. Then $L'_{\text{T-ICE}}(S', r) = 1$ since S' is a subsystem of $\mathbb{S}$. If $\mathbb{S}$ is not terminating, $L'_{\text{T-ICE}}(\mathbb{S}, r) = -1$ for every $r \in Y'$. □

Lemma 5.2. $\inf_{S'} L_{\text{T-ICE}}(S', r) = 1$ *implies the termination of* $\mathbb{S}$. □

Proof. If r is not a ranking function of $\mathbb{S}$, there exists a transition $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} \dots \rightsquigarrow_{\mathbb{S}} s_n \rightsquigarrow_{\mathbb{S}} s_{n+1}$ such that $r(s_n) \not> r(s_{n+1})$. Then the subsystem $S' = (\{s_0\}, \{(s_i, s_{i+1}) \mid 0 \le i \le n\})$ satisfies $L'_{\text{T-ICE}}(S', r) = -1$. □

We discuss the tractability of the primal and dual witness check.

The dual witness check is tractable because whether $\inf_{S'} L_{\text{T-ICE}}(S', r) = -1$ is reducible to a safety verification problem. Let $\mathbb{S}^{(r)}$ be the transition system (with bad states) given by $|\mathbb{S}^{(r)}| := |\mathbb{S}| \times |\mathbb{S}|$, $I_{\mathbb{S}^{(r)}} := \{(s, s') \mid s \in I_{\mathbb{S}}, s \rightsquigarrow_{\mathbb{S}} s'\}$, $(\rightsquigarrow_{\mathbb{S}^{(r)}}) := \{((s, s'), (s', s'')) \mid s' \rightsquigarrow_{\mathbb{S}} s''\}$ and $B_{\mathbb{S}^{(r)}} := \{(s, s') \mid r(s) \le r(s')\}$. A state of $\mathbb{S}^{(r)}$ is a pair (s, s') of a previous state s and a current state s', and a pair is bad if and only if it violates the ranking condition with respect to r. If the system $\mathbb{S}^{(r)}$ is safe, r is a ranking function for $\mathbb{S}$. The safety verification of $\mathbb{S}^{(r)}$ can be solved by procedures such as those discussed in Sections 3 and 4. If $\mathbb{S}^{(r)}$ is unsafe, an error trace $(s_0, s_1)(s_1, s_2) \dots (s_n, s_{n+1}) \in |\mathbb{S}^{(r)}|^*$ induces $S' = (\{s_0\}, \{(s_0, s_1)(s_1, s_2) \dots (s_n, s_{n+1})\})$.

On the contrary, it is hard to check the primal witness, which asks to find a ranking function r that works well on the subsystem S'. Since S' is a terminating finite transition system, a ranking function can in theory be given by mapping each point to the number of remaining steps, but this would not be expected to behave well for points outside of S'. To find generalized ranking functions, methods have been proposed that use template-based synthesis [Unno et al. 2021] as well as machine learning techniques, such as decision tree learning [Kura et al. 2021] and support vector machines [Li et al. 2020].

5.3 CEGAR for Termination

The Lagrangian for CEGAR has a semilattice structure on Y and L_{CEGAR} is monotone on Y. In the context of termination verification, Y consists of (sets of) candidate ranking functions, but it is not straightforward to introduce the join semilattice structure to the Y-component.

A *disjunctively well-founded relation* [Podelski and Rybalchenko 2004] is a concept to address this issue: a relation $\succ$ is *disjunctively well-founded* if it is a finite union of well-founded relations. For a finite set $R \in \mathcal{P}_{\text{fin}}(\mathfrak{R})$ of candidate ranking functions, the relation $\succ_R$ defined by $(s \succ_R s') :\Leftrightarrow \exists r \in R. r(s) > r(s')$ is a disjunctively well-founded relation. The set $\mathcal{P}_{\text{fin}}(\mathfrak{R})$ has an obvious lattice structure, and $R \subseteq R' \in \mathcal{P}_{\text{fin}}(\mathfrak{R})$ implies $(\succ_R) \subseteq (\succ_{R'})$.

Let $X := \{s_0 s_1 \dots s_n \in |\mathbb{S}| \mid I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} \dots \rightsquigarrow_{\mathbb{S}} s_n\}$, $Y := \mathcal{P}_{\text{fin}}(\mathfrak{R})$, and

$$L_{\text{T-CEGAR}}(s_0 \dots s_n, R) = \begin{cases} 1 & \forall i. \forall j. (i < j) \Rightarrow (s_i \succ_R s_j) \\ -1 & \text{otherwise.} \end{cases}$$

Note that $\succ_R$ must satisfy $s_i \succ_R s_j$ for every $i < j$, including the case that $i + 1 \neq j$, *i.e.* s_j is not the immediate successor of s_i, unlike the condition for ranking functions. Then Y is a join semilattice and $L_{\text{T-CEGAR}}$ is monotone on Y. When $\sup_R \inf_\tau L_{\text{T-CEGAR}}(\tau, R) = 1$, then $\mathbb{S}$ is terminating by the following theorem.

Theorem 5.3 (Podelski and Rybalchenko [2004, Theorem 1]). *A transition system* $\mathbb{S}$ *is terminating if there exists a disjunctively well-founded relation* $\succ$ *such that* $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}}^* s \rightsquigarrow_{\mathbb{S}}^+ s'$ *implies* $s \succ s'$. □

Corollary 5.4. *If* $\sup_y \inf_x L_{\text{T-CEGAR}}(x, y) = 1$, *then* $\mathbb{S}$ *is terminating.* □

The dual witness check, which asks to find τ such that $L_{\text{T-ICE}}(\tau, R) = -1$, is reducible to a safety verification problem. The idea is similar to the above case, but the first component s of a pair (s, s') is now a past state that is not necessarily the previous state. Formally, let $\mathbb{S}^{(R)}$ be the transition system (with bad states) given by $|\mathbb{S}^{(R)}| := |\mathbb{S}| \times |\mathbb{S}|$, $I_{\mathbb{S}^{(R)}} := \{(s, s') \mid s \in I_{\mathbb{S}}, s \rightsquigarrow_{\mathbb{S}} s'\}$, $(\rightsquigarrow_{\mathbb{S}^{(R)}}) := \{((s_1, s_1'), (s_2, s_2')) \mid s_1' \rightsquigarrow_{\mathbb{S}} s_2' \text{ and } s_2 = s_1 \vee s_2 = s_1'\}$ and $B_{\mathbb{S}^{(R)}} := \{(s, s') \mid s \not\succ_R s'\}$. For an error trace $(s_0, s_0')(s_1, s_1') \ldots (s_n, s_n') \in |\mathbb{S}^{(R)}|^*$ for $\mathbb{S}^{(R)}$, we have $I_{\mathbb{S}} \ni s_0 \rightsquigarrow_{\mathbb{S}} s_0' \rightsquigarrow_{\mathbb{S}} s_1' \rightsquigarrow_{\mathbb{S}} \ldots \rightsquigarrow_{\mathbb{S}} s_n'$, $s_n \not\succ_R s_n'$ and $s_n \in \{s_0, s_0', s_1', \ldots, s_{n-1}'\}$.

The primal witness check is more tractable than $L_{\text{T-ICE}}$ since it suffices to find a ranking function that works for a single trace.

The basic primal-dual procedure applied to $L_{\text{T-CEGAR}}$ is (the core of) the procedure known as Terminator [Cook et al. 2005, 2006].

6 Lagrangian for Quantified Linear Arithmetic Solver

This section deals with a problem that is closely related to verification, but different in nature from the safety/termination verification problems we have dealt with so far. The problem is validity checking of first-order predicate logic formulas, and this section analyses the procedure given by Farzan and Kincaid [2016] from the viewpoint of Lagrange duality. Basically, the Lagrangian corresponds to Skolemization, but as discussed in Farzan and Kincaid [2016], Skolemization alone is not sufficient. Interestingly, our framework of Lagrangian duality and basic primal-dual procedure clarifies the issue: as we shall see, it is the monotonicity criterion.

6.1 First-Order Formula and Skolemization

This section focuses on first-order formulas. We consider the theory of linear rational arithmetic, LRA, which is the theory that Farzan and Kincaid [2016] mainly dealt with. A term is given by $t ::= x \mid c \mid t_1 + t_2 \mid c \cdot t$ (where $c \in \mathbb{Q}$) and atomic predicates are $t_1 < t_2$ and $t_1 \leq t_2$. We assume that formulas are in prenex normal form, *i.e.*, $\varphi = Q_1 x_1 \ldots Q_k x_k . \vartheta$ where $Q_i \in \{\forall, \exists\}$ and ϑ is quantifier-free. Since LRA enjoys effective quantifier elimination (*i.e.*, given a quantified formula $\varphi(x, \vec{y})$, one can effectively construct a quantifier-free formula $\psi(\vec{y})$ such that $\psi(\vec{y}) \leftrightarrow \exists x . \varphi(x, \vec{y})$), the validity checking problem for first-order logic formulas over LRA is decidable.

Skolemization is a satisfiability-preserving translation of first-order logic formulas, introducing new functional symbols. Consider a quantified formula:

$$\forall a \in \mathbb{Q} . \exists b \in \mathbb{Q} . \forall c \in \mathbb{Q} . \varphi(a, b, c). \tag{4}$$

If this formula is true, there exists a function $f_b : \mathbb{Q} \to \mathbb{Q}$ that maps a value assigned to a to an appropriate value for b, *i.e.*, for every $v \in \mathbb{Q}$,

$$\forall c \in \mathbb{Q} . \varphi(v, f_b(v), c),$$

and f_b is called a *Skolem function* for b. By using the Skolem function, the validity of eq. (4) is reduced to the problem to find an appropriate function f_b such that

$$\forall a \in \mathbb{Q} . \forall c \in \mathbb{Q} . \varphi(a, f_b(a), c)$$

is valid, and this validity problem can be solved by SMT solvers (provided that f_b is describable as a term in LRA). Conversely, if the formula in eq. (4) is false, its negation $\exists a \in \mathbb{Q} . \forall b \in \mathbb{Q} . \exists c \in \mathbb{Q} . \neg\varphi(a, b, c)$ is true, so there exist Skolem functions $f_a : \mathbb{Z}$ and $f_c : \mathbb{Z} \to \mathbb{Z}$ such that

$$\forall b \in \mathbb{Z} . \neg\varphi(f_a, b, f_c(b)).$$

The above observation yields the following Lagrangian: letting

$$X = (\text{Skolem functions for } a \text{ and } c) = (\mathbb{Z} \times (\mathbb{Z} \to \mathbb{Z}))$$
$$Y = (\text{Skolem function for } b) = (\mathbb{Z} \to \mathbb{Z}),$$

the Lagrangian $L \colon X \times Y \longrightarrow \{-1, 1\}$ is given by

$$L((f_a, f_c), f_b) = 1 \quad :\Leftrightarrow \quad \varphi(f_a, f_b(f_a), f_c(f_b(f_a))) \text{ is true.}$$

The optimal value of the dual optimization problem is 1 if and only if there exists an appropriate Skolem function for b. To see this, note that

$$\begin{aligned} \inf_{f_a, f_c} L((f_a, f_c), \beta) = -1 \quad &\Leftrightarrow \quad \exists f_a \in \mathbb{Q}.\exists f_c \in (\mathbb{Q} \to \mathbb{Q}).\varphi(f_a, \beta(f_a), f_c(\beta(f_a))) \text{ is false} \\ &\Leftrightarrow \quad \exists a \in \mathbb{Q}.\exists c \in \mathbb{Q}.\varphi(a, \beta(a), c) \text{ is false,} \end{aligned}$$

so, by negating the both sides,

$$\inf_{f_a, f_c} L((f_a, f_c), \beta) = 1 \quad \Leftrightarrow \quad \forall a \in \mathbb{Q}.\forall c \in \mathbb{Q}.\varphi(a, \beta(a), c).$$

Hence, $\beta \in Y$ witnessing $\sup_{y \in Y} \inf_{x \in X} L(x, y) = 1$ is a Skolem function for b that witnesses the truth of the formula in eq. (4). Dually, a witness $\alpha \in X$ of $\inf_{x \in X} \sup_{y \in Y} L(x, y) = -1$ is a pair of Skolem functions for a and c that witnesses the falsity of the formula.

In the above setting, both X and Y are the sets of all possible Skolem functions in the semantic domains, so there is an appropriate choice for exactly one of X and Y. Therefore, we have the strong duality:

$$\sup_{f_b} \inf_{f_a, f_c} L((f_a, f_c), f_b) \quad = \quad \inf_{f_a, f_c} \sup_{f_b} L((f_a, f_c), f_b),$$

and the optimal value coincides with the truth of the formula.

This argument can be applied to arbitrary formulas. Given a first-order predicate logic formula ψ with quantifiers, there exists a Lagrangian $L_\psi \colon X_\psi \times Y_\psi \longrightarrow \{-1, 1\}$, where X_ψ (resp. Y_ψ) is the set of Skolem functions for universally (resp. existentially) quantified variables in ψ, such that L_ψ satisfies the strong duality and the optimal value of L_ψ is the truth of ψ.

6.2 Quantified LRA Solver by Farzan and Kincaid

Very roughly, the procedure by Farzan and Kincaid [2016] is an instance of the basic primal-dual procedure for Lagrangians in Section 6.1. However, we cannot directly apply the basic primal-dual procedure since the Lagrangian does not satisfy the monotonicicy criterion, *i.e.* the sets X and Y of Skolem functions are not poset (and the Lagrangian is not monotone). Another, relatively minor issue is that the sets X and Y are too large, containing elements that have no finite representations.

Farzan and Kincaid [2016] introduced the notion of *strategy skeletons* to address the above issues. Intuitively, a strategy skeleton is a variant of Skolem functions, albeit with the additional capacity to select multiple values, claiming that at least one of the selected values is appropriate. The inclusion of the set of selected values is an order for strategy skeletons, and the Lagrangian is monotone (or antimonotone) with respect to this order.

Definition 6.1 (Strategy Skeleton). The set of *SAT strategy skeletons* is defined by the grammar

$$\pi \quad ::= \quad \bullet \mid \forall y.\pi \mid \bigsqcup_{i=1,\ldots,n} t_i.\pi_i$$

where t_i is a term of LRA and $t_i \neq t_j$ if $i \neq j$. For a formula ψ in prenex normal form we write $\vec{x} \vdash \psi \lhd \pi$ to denote that π is a SAT strategy skeleton for ψ and satisfies $\mathrm{fv}(\pi) \subseteq \{\vec{x}\}$. Formally, it is

the relation defined by the following rules:

$$\frac{QF(\vartheta)}{\vec{x} \vdash \vartheta \lhd \bullet} \qquad \frac{\vec{x}, y \vdash \psi \lhd \pi}{\vec{x} \vdash (\forall y.\psi) \lhd (\forall y.\pi)} \qquad \frac{\vec{x} \vdash \psi \lhd \pi_i \text{ for every } i \qquad \forall i.\mathrm{fv}(t_i) \subseteq \{\vec{x}\}}{\vec{x} \vdash (\exists y.\psi) \lhd (\bigsqcup_i t_i.\pi_i)}$$

The strategy skeleton $\bullet$ is for quantifier-free formulas, to which we have nothing to choose. The strategy skeleton $\forall y.\pi$ is for universally-quantified formulas $\forall y.\psi$: a SAT strategy skeleton does nothing on $\forall y$, and π describes a way to choose values for $\exists$-quantified variables in ψ. The strategy skeleton $\bigsqcup_{i=1,\ldots,n} t_i.\pi_i$ is for $\exists y.\psi$, meaning that an appropriate choice for y should be found in $\{t_1, \ldots, t_n\}$. An UNSAT strategy skeleton can be defined similarly:

$$\varrho \quad ::= \quad \bullet \mid \exists y.\varrho \mid \bigsqcap_{i=1,\ldots,n} t_i.\varrho_i$$

$$\frac{QF(\vartheta)}{\vec{x} \vdash \bullet \rhd \vartheta} \qquad \frac{\vec{x}, y \vdash \varrho \rhd \psi}{\vec{x} \vdash (\exists y.\varrho) \rhd (\exists y.\psi)} \qquad \frac{\vec{x} \vdash \varrho_i \rhd \psi \text{ for every } i \qquad \forall i.\mathrm{fv}(t_i) \subseteq \{\vec{x}\}}{\vec{x} \vdash (\bigsqcap_i t_i.\varrho_i) \rhd (\forall y.\psi)}.$$

For a sentence φ in prenex normal form, let $\mathit{Skeleton}_{\mathsf{SAT}}(\varphi)$ and $\mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi)$ be the set of SAT and UNSAT strategy sketelons for φ with no free variable, *i.e.* $\mathit{Skeleton}_{\mathsf{SAT}}(\varphi) := \{\pi \mid \vdash \varphi \lhd \pi\}$ and $\mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi) := \{\varrho \mid \vdash \varrho \rhd \varphi\}$. □

Example 6.2 ([Farzan and Kincaid 2016]). Let

$$\varphi \quad := \quad \exists w.\forall x.\exists y.\forall z.\vartheta(w,x,y,z), \qquad \vartheta(w,x,y,z) \;\equiv\; ((y < 1 \vee 2w < y) \wedge (z < y \vee x < z)).$$

A SAT strategy skeleton is

$$\pi \quad := \quad 0 \,.\, \forall x \,.\, \big((x \,.\, \forall z \,.\, \bullet) \sqcup (2x \,.\, \forall z \,.\, \bullet)\big).$$

This skeleton selects 0 for w and x or $2x$ for y depending on the branch of $\sqcup$. □

The set $\mathit{Skeleton}_{\mathsf{SAT}}(\varphi)$ is a preordered set by the order defined by the following rules:

$$\frac{}{\bullet \le \bullet} \qquad \frac{\pi \le \pi'}{\forall y.\pi \le \forall y.\pi'} \qquad \frac{\forall i.\exists j.t_i = t'_j \wedge \pi_i \le \pi'_j}{\bigsqcup_i t_i.\pi_i \le \bigsqcup_j t'_j.\pi'_j}$$

The join operation $\cup$ is defined by $\bullet \cup \bullet := \bullet$, $(\forall y.\pi) \cup (\forall y.\pi') := \exists y.(\pi \cup \pi')$ and

$$(t.\pi) \cup (\bigsqcup_{i\in I} t_i.\pi_i) := \begin{cases} (t.\pi) \sqcup (\bigsqcup_{i\in I} t_i.\pi_i) & \text{if } t \neq t_i \text{ for every } i \\ (t.(\pi \cup \pi_j)) \sqcup (\bigsqcup_{i\in(I\setminus\{j\})} t_i.\pi_i) & \text{if } t = t_j. \end{cases}$$

Dually $\mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi)$ has a similar structure, obtained by replacing $\sqcup$ with $\sqcap$ (*e.g.* $\varrho \le (\varrho \sqcap \varrho')$). We write the join in $\mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi)$ as $\cup$, since it computes the union of candidates.

A pair $(\varrho, \pi) \in \mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi) \times \mathit{Skeleton}_{\mathsf{SAT}}(\varphi)$ of SAT and UNSAT skeletons for the same formula φ induces a quantifier-free formula $\langle \varrho \mid \varphi \mid \pi \rangle$ defined by induction on φ as follows:

$$\begin{array}{rcl} \langle \bullet \mid \quad \vartheta \quad \mid \bullet \rangle & := & \vartheta \\ \langle \bigsqcap_i t_i.\varrho_i \mid \forall x.\varphi \mid \forall x.\pi \rangle & := & \bigwedge_i \langle \varrho_i \mid \varphi[t_i/x] \mid \pi[t_i/x] \rangle \\ \langle \exists x.\varrho \mid \exists x.\varphi \mid \bigsqcup_i t_i.\pi_i \rangle & := & \bigvee_i \langle \varrho[t_i/x] \mid \varphi[t_i/x] \mid \pi_i \rangle. \end{array}$$

Let $L_{\mathrm{FK}} \colon \mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi) \times \mathit{Skeleton}_{\mathsf{SAT}}(\varphi) \longrightarrow \{-1, 1\}$ be the Lagrangian defined by

$$L_{\mathrm{FK}}(\varrho, \pi) = 1 \quad :\Leftrightarrow \quad \langle \varrho \mid \varphi \mid \pi \rangle \text{ is true.}$$

This Lagrangian is written as $L^{\varphi}_{\mathrm{FK}}$ if the formula φ should be clarified. The procedure by Farzan and Kincaid [2016] is an instance of the basic primal-dual procedure for L_{FK}.

Example 6.3 ([Farzan and Kincaid 2016]). Recall the formulas φ and ϑ and the SAT strategy skeleton π in example 6.2. An example of an UNSAT strategy skeleton is

$$\varrho \quad := \quad \exists w\,.\,(-1)\,.\,\exists y\,.\,\big((u_1\,.\,\bullet) \sqcap (u_2\,.\,\bullet)\big), \qquad u_1(w,y) = y, \quad u_2(w,y) = (w+y)/2$$

Then, the calculation of $\langle \varrho \mid \varphi \mid \pi \rangle$ proceeds as

$$\begin{aligned}
&\left\langle \exists y.((u_1(0,y)\,.\,\bullet) \sqcap (u_2(0,y)\,.\,\bullet)) \,\middle|\, \exists y.\forall z.\vartheta(0,-1,y,z) \,\middle|\, ((-1)\,.\,\forall z\,.\,\bullet) \sqcup ((-2)\,.\,\forall z\,.\,\bullet) \right\rangle \\
&= \left\langle ((u_1(0,-1)\,.\,\bullet) \sqcap (u_2(0,-1)\,.\,\bullet)) \,\middle|\, \forall z.\vartheta(0,-1,-1,z) \,\middle|\, \forall z\,.\,\bullet \right\rangle \\
&\quad \vee \left\langle ((u_1(0,-2)\,.\,\bullet) \sqcap (u_2(0,-2)\,.\,\bullet)) \,\middle|\, \forall z.\vartheta(0,-1,-2,z) \,\middle|\, \forall z\,.\,\bullet \right\rangle \\
&= \Big(\vartheta\big(0,-1,-1,u_1(0,-1)\big) \;\wedge\; \vartheta\big(0,-1,-1,u_2(0,-1)\big)\Big) \\
&\quad \vee \Big(\vartheta\big(0,-1,-2,u_1(0,-2)\big) \;\wedge\; \vartheta\big(0,-1,-2,u_2(0,-2)\big)\Big)
\end{aligned}$$

so we have $L_{\mathrm{FK}}(\varrho,\pi) = \langle \varrho \mid \varphi \mid \pi \rangle = \big(\bot \wedge \top\big) \vee \big(\bot \wedge \bot\big) = -1$. □

Lemma 6.4. *$L_{\mathrm{FK}} \colon Skeleton_{\mathsf{UNSAT}}(\varphi) \times Skeleton_{\mathsf{SAT}}(\varphi) \longrightarrow \{-1,1\}$ is anti-monotone on the first argument and monotone on the second argument.*

Proof. Intuitively, if $\pi \leq \pi'$, then π' has more components than π connected by $\sqcup$. Then $\langle \varrho | \varphi | \pi' \rangle$ has more components than $\langle \varrho | \varphi | \pi \rangle$ connected by $\vee$. So, the validity of the latter implies the validity of the former. The anti-monotonicity on $Skeleton_{\mathsf{UNSAT}}(\varphi)$ is similar. □

Prior to the discussion of correctness, we discuss the decidability of the primal/dual witness check problems, following Farzan and Kincaid [2016]. We discuss the dual witness check here; the primal is essentially equivalent. Assume a SAT strategy skeleton $\pi \in Skeleton_{\mathsf{SAT}}(\varphi)$. Then $\inf_{\varrho} L^{\varphi}_{\mathrm{FK}}(\varrho,\pi) = 1$ is reducible to the validity of the formula $\varphi|\pi\rangle$ given as follows:

$$\vartheta|\bullet\rangle := \vartheta \qquad (\forall x.\varphi)|\forall x.\pi\rangle := \forall x.(\varphi|\pi\rangle) \qquad (\exists x.\varphi)|\bigsqcup_i t_i.\pi_i\rangle := \bigvee_i (\varphi[t_i/x])|\pi_i\rangle.$$

Since $\varphi|\pi\rangle$ is a formula with no $\exists$, its validity can be checked by an SMT solver.

Example 6.5. Recall the formulas φ and ϑ and a SAT strategy skeleton π in example 6.2. Then

$$\varphi|\pi\rangle \quad = \quad \forall x.\big((\forall z_1.\vartheta(0,x,x,z_1)) \vee (\forall z_2.\vartheta(0,x,2x,z_2))\big),$$

which is invalid. For example, consider the assignment $x = -1, z_1 = -1$ and $z_2 = -2$. This assignment gives a UNSAT strategy skeleton $\varrho := \forall w.(-1).\forall y.\big((-1).\bullet \sqcup (-2).\bullet\big)$ such that $L_{\mathrm{FK}}(\varrho,\pi) = -1$. For another SAT strategy skeleton $\pi' := (-2).\forall x.(x+1).\forall z.\bullet$, we have $\varphi|\pi'\rangle = \forall x.\forall z.\vartheta(-2,x,x+1,z)$, which is valid. □

Lemma 6.6. *$\inf_{\varrho} L^{\varphi}_{\mathrm{FK}}(\varrho,\pi) = 1$ if and only if $\varphi|\pi\rangle$ is valid.*

Proof. See [Tsukada et al. 2024, Appendix B]. □

Theorem 6.7. *For every closed formula φ over LRA, the Lagrangian $L^{\varphi}_{\mathrm{FK}}$ enjoys the strong duality, and its optimal value coincides with the validity of φ.*

Proof. The result follows from Farzan and Kincaid [2016]. To show the strong duality, it suffices to prove that, for a valid formula φ, there exists a SAT strategy skeleton $\pi \in Skeleton_{\mathsf{SAT}}(\varphi)$ such that $\inf_{\varrho} L^{\varphi}_{\mathrm{FK}}(\varrho,\pi) = 1$. The basic observation in Farzan and Kincaid [2016] is that the quantifier elimination $\vartheta(y) \leftrightarrow \exists x.\psi(x,y)$ can be achieved by substitution, *i.e.* there exists a finite

set $\{t_1, \ldots, t_n\}$ of terms such that $(\exists x.\psi(x, y)) \leftrightarrow (\psi(t_1, y) \vee \cdots \vee \psi(t_n, y))$.[4] Iterative application of this result yields a SAT strategy skeleton π such that $\varphi \leftrightarrow (\varphi|\pi\rangle)$.

Since $\psi(t) \rightarrow (\exists x.\psi(x))$ is valid for every ψ and t, the formula $\varphi|\pi\rangle$, obtained by instantiating $\exists$-variables, is stronger than φ. By Lemma 6.6, $\inf_\varrho L^\varphi_{\mathrm{FK}}(\varrho, \pi) = 1$ implies the validity of $\varphi|\pi\rangle$ and hence the validity of φ. □

The algorithm given by Farzan and Kincaid [2016] is an instance of the basic primal-dual procedure for the Lagrangian L_{FK}. Given a SAT strategy skeleton π, the check of $\inf_\varrho L_{\mathrm{FK}}(\varrho, \pi) = 1$ is reducible to the validity of $\varphi|\pi\rangle$, which can de solved by an SMT solver. If $\inf_\varrho L_{\mathrm{FK}}(\varrho, \pi) = -1$, or equivalently, $\varphi|\pi\rangle$ is invalid, the SMT solver generates a counter-model for $\varphi|\pi\rangle$, from which a UNSAT strategy skeleton ϱ can be constructed as illustrated in example 6.5. But their procedure has an important twist here: the assignment to each variable in the counter-model, which is a concrete rational number, is converted to a term. The terms are chosen from those playing an important role in quantifier elimination or model-based projection (*cf.* [Farzan and Kincaid 2016, Section 4.1]). As the set of such terms is finite, Farzan and Kincaid [2016] in effect considered finite subsets of $\mathit{Skeleton}_{\mathrm{SAT}}(\varphi)$ and $\mathit{Skeleton}_{\mathrm{UNSAT}}(\varphi)$, and this finiteness guarantees that Farzan and Kincaid's procedure is terminating.

7 Lagrangian for Fixed-Point Logic over Quantified Linear Arithmetic

This section demonstrates the usefulness of our Lagrangian-based approach by developing a solver of the validity problem for a fixed-point logic over quantified linear arithmetic. The validity problem is closely related to game solving against liveness winning criteria as in Heim and Dimitrova [2024].

Our observation is that the difficulty of this problem can actually be decomposed into the difficulties of termination analysis and of quantifiers, each of which has been addressed in Sections 5 and 6. So a Lagrangian for the validity problem of fixed-point logic with quantifiers is obtained as a simple combination of ideas in these sections.

We have developed a prototype implementation of the basic primal-dual procedure for the proposed Lagrangian and provide an evaluation.

7.1 Fixed-Point Logic

We define the syntax and semantics of the fixed-point logic studied in this section. Although the discussion in this section is applicable to first-order structures in general to some extent, we will focus on the fixed-point logic over linear integer arithmetic, LIA. An atomic predicate p is either $=$ or $\leq$, and a term is given by $t ::= n \mid x \mid t + t' \mid n \times t \mid t \bmod n$ (where n is an integer and x is a variable).

A *(fixed-point-free) formula* is defined by the following grammar:

$$\varphi, \psi \quad ::= \quad p(\vec{t}) \mid \neg p(\vec{t}) \mid P(\vec{t}) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \forall x.\varphi \mid \exists x.\varphi,$$

where t is a term in LIA, p is an atomic predicate in LIA (*i.e.* $=$ or $\leq$), and P is a user-defined predicate. For notational simplicity, we will also use logical formulas that do not follow the above syntax in the strict sense but can be transformed into the above form (*e.g.* $(x < 42 \wedge i \neq 0) \Rightarrow P(x + i)$, which is equivalent to $(42 \leq x \vee i = 0 \vee P(x + 1))$). Predicates P are defined by mutual recursion:

$$P_1(\vec{x}_1) \overset{\xi_1}{=} \varphi_1 \; ; \; P_2(\vec{x}_2) \overset{\xi_2}{=} \varphi_2 \; ; \ldots ; \; P_n(\vec{x}_n) \overset{\xi_n}{=} \varphi_n$$

where $\vec{x}_i$ is a sequence of variables, $\xi_i \in \{\nu, \mu\}$, and φ_i is a fixed-point-free formula with free variables in $\{\vec{x}_i, P_1, \ldots, P_n\}$. If $\xi_i = \nu$ (resp. μ), then P_i is the greatest solution (resp. least solution) of the

[4] Farzan and Kincaid [2016] was inspired by *model-based projection* [Komuravelli et al. 2014], which is closely related to quantifier elimination (see [Farzan and Kincaid 2016, Section 4.1]).

equation $P_i(\vec{x}_i) = \varphi_i$ (for details, see the formal semantics defined below). The order of definitions matters: $\cdots ; P_i(\vec{x}_i) \overset{\xi_i}{=} \varphi_i; P_{i+1}(\vec{x}_{i+1}) \overset{\xi_{i+1}}{=} \varphi_{i+1}; \cdots$ differs from $\cdots ; P_{i+1}(\vec{x}_{i+1}) \overset{\xi_{i+1}}{=} \varphi_{i+1}; P_i(\vec{x}_i) \overset{\xi_i}{=} \varphi_i; \cdots$ when $\xi_i \neq \xi_{i+1}$. For readers familiar with parity conditions, the left has higher priority. A formula is a pair $(\varphi, \mathcal{R})$ of a fixed-point free formula φ with a mutually-recursive definition $\mathcal{R}$ of predicates in φ.

Example 7.1 (Heim and Dimitrova [2024, Example 1.1]). This example is a formalization of the following control problem. Consider a system with a single integer variable $x \colon \mathbb{Z}$. For each step, the environment chooses $i \in \mathbb{Z}$. If $x < 42$ or $i = 0$, the system successfully terminates. If $x \geq 42$ and $i \neq 0$, the system proceeds to the next step, updating x. In this case, the controller can choose the next value of x from $(x + i)$ and $(x - i)$. The problem is to synthesize a controller that will always make the system terminate (or, if such a controller does not exist, to answer that it does not exist).

The existence of a controller is equivalent to the validity of $\forall x.P(x)$ where

$$P(x) \quad \overset{\mu}{=} \quad \forall i.\Big(\big((x < 42 \vee i = 0) \Rightarrow \top\big) \wedge \big((x \geq 42 \wedge i \neq 0) \Rightarrow (P(x + i) \vee P(x - i))\big)\Big)$$

The user-defined predicate $P(n)$ means that the system with an appropriate controller terminates from the state $x = n$. The right-hand-side of the definition describes the one-step transition of the system; the universal quantification $\forall i$ expresses that i is chosen by the environment. The predicate P is defined as the least fixed-point; this reflects the requirement that the system should not perform infinite step transitions. The formula $\forall x.P(x)$ is valid, as we shall see. □

We formally define the semantics of a mutually-recursive definition of predicates. First, we deal with a single equation $P(\vec{x}) \overset{\xi}{=} \varphi$. We write $\varphi(P)(\vec{x})$ to make explicit that φ depends on P and $\vec{x}$. Let $\mathcal{D} := (\mathbb{Z}^\ell \to \{0, 1\})$ be the semantic domain for the predicate P, where ℓ is the length of $\vec{x}$. This is a complete lattice by the point-wise order (*i.e.*, for $f, g \in \mathcal{D}$, $f \leq g$ if and only if $f(\vec{n}) \leq g(\vec{n})$ for every $\vec{n} \in \mathbb{Z}^\ell$). Since $\neg P$ does not appear in φ, the function $\varphi \colon \mathcal{D} \longrightarrow \mathcal{D}$ mapping $d \in \mathcal{D}$ to $\varphi(d)$ (which is a predicate $\mathbb{Z}^\ell \ni \vec{n} \mapsto \varphi(d)(\vec{n}) \in \{0, 1\}$) is a monotone function. Then, Knaster-Tarski theorem shows that the set $\{d \in \mathcal{D} \mid d = \varphi(d)\}$ has both the least and greatest elements. When $\xi = \mu$ (resp. $\xi = \nu$), then P is defined as the least solution (resp. greatest solution).

The semantics of a mutually-recursive definition

$$\mathcal{R} \quad = \quad \big(P_1(\vec{x}_1) \overset{\xi_1}{=} \varphi_1 \; ; \; P_2(\vec{x}_2) \overset{\xi_2}{=} \varphi_2 \; ; \ldots ; \; P_n(\vec{x}_n) \overset{\xi_n}{=} \varphi_n\big)$$

is given as follows. Let $\mathcal{D}_i := (\mathbb{Z}^{\ell_i} \to \{0, 1\})$ be the semantic domain for the predicate P_i, where ℓ_i is the length of $\vec{x}_i$. We solve the equations in the right-to-left direction. A subtlety here is that φ_n may contain $P_1, \ldots, P_{n-1}$; we write $\varphi_n(P_1, \ldots, P_{n-1}, P_n)(\vec{x}_n)$ to clarify the dependency. For each $(d_1, \ldots, d_{n-1}) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1}$, the equation $P_n(\vec{x}_n) \overset{\xi_n}{=} \varphi_n(d_1, \ldots, d_{n-1}, P_n)(\vec{x}_n)$ determines a value $\lambda_n(d_1, \ldots, d_{n-1}) \in \mathcal{D}_n$ depending on $d_1, \ldots, d_{n-1}$. The solutions $\lambda_n(d_1, \ldots, d_{n-1})$ parameterized by $(d_1, \ldots, d_{n-1}) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1}$ determine a function $\lambda_n \colon \mathcal{D}_1 \times \cdots \times \mathcal{D}_{n-1} \to \mathcal{D}_n$. Substituting P_n with $\lambda_n(P_1, \ldots, P_{n-1})$ yields a system of equations without P_n, and then we solve the equation for P_{n-1}. Iteratively applying this process yields $P_k = \lambda_k(P_1, \ldots, P_{k-1})$ for each $k = 1, \ldots, n$ depending on $P_1, \ldots, P_{k-1}$. Let $\delta_1 = \lambda_1()$, $\delta_2 = \lambda_2(\delta_1)$, and $\delta_k = \lambda_k(\delta_1, \ldots, \delta_{k-1})$ for general k. The assignment on $P_1, \ldots, P_n$ determined by $\mathcal{R}$ is $(\delta_1, \ldots, \delta_n) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$. The formula $(\varphi, \mathcal{R})$ is *valid* if $\varphi = \varphi(\delta_1, \ldots, \delta_n)$ is valid.

7.2 Lagrangian and Primal-Dual Procedure

As we have mentioned at the end of the previous subsection, the validity problem becomes easier if one can remove the least fixed-point μ and existential quantifier $\exists$. We have already discussed a

way to remove an existential quantifier $\exists$ in Section 6, in which we used a SAT strategy skeleton to guide a process of removal. The removal of the least fixed-point has been addressed in Section 5, albeit in a slightly different form: a ranking function or disjunctively-well-founded relation is used to reduce the termination problem (a typical μ-property) to a safety problem (a typical ν-property).

The Lagrangian of this section simply combines these data, so the Y-component is a pair of a SAT strategy skeleton and a disjunctively-well-founded relation. The X-component is similar: it describes choices of $\forall$-variables and ranking information for ν-recursions.

We formally describe the details. Assume a pair $(\varphi, \mathcal{R})$, where

$$\mathcal{R} \quad = \quad \left(P_1(\vec{x}_1) \overset{\xi_1}{=} \varphi_1 \,;\, P_2(\vec{x}_2) \overset{\xi_2}{=} \varphi_2 \,;\, \ldots \,;\, P_n(\vec{x}_n) \overset{\xi_n}{=} \varphi_n\right).$$

Let ℓ_i be the length of $\vec{x}_i$. For simplicity, we assume that $\varphi, \varphi_1, \ldots, \varphi_n$ are in prenex normal form. Let $\mathfrak{R}_i$ be the set of candidate ranking functions available for P_i. Let $L := \{i \mid \xi_i = \mu\}$ and $G := \{i \mid \xi_i = \nu\}$ be the sets of indices of μ- and ν-predicates, repectively.

The Lagrangian is a function $L_{\mathrm{Fix}} \colon X \times Y \longrightarrow \{-1, 1\}$, where

$$\begin{aligned} X \quad &:= \textstyle\prod_{i\in G} \mathcal{P}_{\mathrm{fin}}(\mathfrak{R}_i) \times \mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi) \times \prod_{i=1}^{n} \mathit{Skeleton}_{\mathsf{UNSAT}}(\varphi_i) \\ Y \quad &:= \textstyle\prod_{i\in L} \mathcal{P}_{\mathrm{fin}}(\mathfrak{R}_i) \times \mathit{Skeleton}_{\mathsf{SAT}}(\varphi) \times \prod_{i=1}^{n} \mathit{Skeleton}_{\mathsf{SAT}}(\varphi_i). \end{aligned}$$

Recall that in Section 6, which dealt with logical expressions with quantifiers, the two arguments of the Lagrangian produce a logical formula whose validity is trivially computable (*i.e.* a logical formula without quantifiers). A similar situation can be observed for fixed-point logic: two arguments of the Lagrangian induce a logic formula whose validity is computable. The problematic constructs in fixed-point logic formulas are quantifiers and recursions; the quantifiers are eliminated by the strategy skeletons, and we avoid the divergence in the evaluation of the recursion by immediately stopping the evaluation when a ranking information violation is detected. Then, we define the value of the Lagrangian as the outcome of the evaluation.

Given $x = ((R_i)_{i\in G}, \varrho, (\varrho_i)_{i=1,\ldots,n}) \in X$ and $y = ((R_i)_{i\in L}, \pi, (\pi_i)_{i=1,\ldots,n})$, we formally define the value $L_{\mathrm{Fix}}(x, y)$ of the Lagrangian as follows. We first apply the strategy skeletons $(\varrho, (\varrho_i)_i)$ and $(\pi, (\pi_i)_i)$ and construct a quantifier-free approximation $(\varphi', \mathcal{R}')$ of the input $(\varphi, \mathcal{R})$. This is simply obtained by applying the translation $\langle \varrho \mid \varphi \mid \pi \rangle$ defined in Section 6. That is, $\varphi' := \langle \varrho \mid \varphi \mid \pi \rangle$ and $P_i'(\vec{x}_i) \overset{\xi_i}{=} \langle \varrho_i \mid \varphi_i \mid \pi_i \rangle$. Note that $(\varphi', \mathcal{R}')$ is neither over- nor under-approximation.

The formula $(\varphi', \mathcal{R}')$ has no quantifier nor free variable, which we regard as a "program" with recursively defined functions $P_1', \ldots, P_n'$ with two kinds of non-deterministic branches, demonic $\wedge$ and angelic $\vee$. We run the program, recording all visited states and monitoring them for violations of the disjunctively-well-founded relations R_i. So, a configuration is a pair $(\psi, \vec{V})$ of a formula ψ and a list of sets $\vec{V} = (V_1, \ldots, V_n)$ in which V_i consists of actual arguments $\vec{n}_i$ such that $P_i(\vec{n}_i)$ has been visited. The reduction relation is defined by

$$\begin{aligned}
&(\psi_1 \,\square\, \psi_2, (V_1, \ldots, V_n)) \longrightarrow (\psi_i, (V_1, \ldots, V_n)) && \square \in \{\wedge, \vee\},\ i \in \{1, 2\} \\
&(P_i'(\vec{n}_i), (V_1, \ldots, V_n)) \longrightarrow (\varphi_i'[\vec{n}_i/\vec{V}_i], (V_1, \ldots, V_i \cup \{(\vec{n}_i)\}, \ldots, V_n)) && \forall \vec{m} \in V_i.\ \vec{m} \succ_{R_i} \vec{n}_i \\
&(P_i'(\vec{n}_i), (V_1, \ldots, V_n)) \longrightarrow \mathtt{false} && \xi_i = \mu \wedge \exists \vec{m} \in V_i.\ \vec{m} \not\succ_{R_i} \vec{n}_i \\
&(P_i'(\vec{n}_i), (V_1, \ldots, V_n)) \longrightarrow \mathtt{true} && \xi_i = \nu \wedge \exists \vec{m} \in V_i.\ \vec{m} \not\succ_{R_i} \vec{n}_i
\end{aligned}$$

(where φ' is the formula defined above, stemming from the definition of P_i') and this reduction process always terminates because $\succ_{R_i}$ is disjunctively well-founded. A normal form is `true`, `false` or $(\mathit{atomic}, \vec{V})$ for some closed atomic formula atomic, so a normal form is associated to a truth value. Then we can reverse the reduction backwards and assign a truth value to each configuration. A configuration $(\psi, \vec{V})$ with $\psi = \psi_1 \wedge \psi_2$ or $\psi_1 \vee \psi_2$ has successors, and its truth value is the

conjuction or the disjunction of those of the successors depending on ψ. We define $L_{\mathrm{Fix}}(x, y) = 1$ if and only if the value assigned to $(\varphi', (\emptyset, \ldots, \emptyset))$ is true. Note that $L_{\mathrm{Fix}}(x, y)$ is computable (provided that every ranking function in $\mathfrak{R}_i$ is computable).

We say that a subset $\mathfrak{R} \subseteq (Z \to \mathbb{N})$ of ranking functions on a set Z is *fine* if, for every $x, y \in Z$, there exists $r \in \mathfrak{R}$ such that $r(x) > r(z)$.

Theorem 7.2. *Assume that $\mathfrak{R}_i$ is fine for every i. Then $\sup_y \inf_x L_{\mathrm{Fix}}(x, y) = 1$ implies the truth of the input formula.*

Proof Sketch. We prove the contraposition. Assume that the input formula is false, and assume $y \in Y$. It suffices to find $x \in X$ such that $\sup_y \inf_x L_{\mathrm{Fix}}(x, y) = -1$. By the game semantics of the fixed-point logic, the opponent has a winning strategy σ for the game induced by the input formula. For every proponent strategy σ', the play determined by σ and σ' reaches a false atomic formula or diverges with an infinitely deep recursive calls to a μ-predicate. The evaluation of such a recursive call eventually violates the disjunctively-well-founded relation given by x. So the fact that σ' loses against σ is witnessed by a finite set of finite reduction sequences. This is particularly true when σ' chooses each assignment of $\exists$ following the SAT strategy skeletons in x. When the proponent follows the SAT strategy skeleton, the proponent always has finitely many options at each moment. So the fact that the proponent cannot win the game following the SAT strategy skeletons in x can be witnessed by a finitely branching tree with no infinite path, so it is a finite tree by König's lemma. Thanks to the finiteness, together with the fineness assumption on $\mathfrak{R}$, we can give y that beats x. □

Both of the X- and Y-components are join semilattices, and the Lagrangian L_{fix} is anti-monotone on X and monotone on Y. So the monotonicity requirement is satisfied. Our implementation uses a variant of the basic primal-dual procedure that is monotone on both $\alpha \in X$ and $\beta \in Y$.

Let us examine the primal and dual witness check problems. An interesting observation for the termination analysis with disjunctively-well-founded relations is that the dual witness check is reducible to a safety problem, using a transition system with an additional component that records one of the already visited states. One may expect that the same idea should apply to the problem of this section as well, but here is a subtlety. The soundness of the reduction relies on the setting of Section 5, in which all non-deterministic branches in the transition system are demonic branching $\wedge$ (*i.e.* whatever the choice, the system should terminate). In the presence of angelic branches $\vee$, the reduction is unsound in general, because the proponent can make a choice that depends on the value of the additional component, which is a kind of cheating as the additional component is not a part of the real state of the transition system. One way to restore soundness is to ensure that choices are independent of additional components by making the dependencies explicit using (fresh) functional symbols. This is the approach of our implementation.

Example 7.3. Conisder the predicate P defined by $P(x) \stackrel{\mu}{=} \exists z.P(z)$. Then $P(x)$ is false for every x. Applying the translation in Section 5.3, we obtain $P'(x, y) \stackrel{\nu}{=} \exists z.(y \succ_R z) \wedge P'(z, y) \wedge P'(z, z)$, but this formula becomes true when we appropriately choose a disjunctively-well-founded relation R. Note that P' is the greatest fixed-point, corresponding to the fact that Section 5.3 translates the termination problem to a safety problem. Let $r(x) := \max(x, 0)$ and $r'(x) := \max(3 - x, 0)$. Our strategy is to choose z from $0, 1$ so that $z \neq y$. Then $y \succ_R z$ holds for every y, and thus $P'(x, y)$ is true for every x and y. Note that the choice of z depends on the additional component y.

Our implementation reduces the problem to find an appropriate function f such that $P''(x, y) \stackrel{\nu}{=} (y \succ_R f(x)) \wedge P'(f(x), y) \wedge P'(f(x), f(x))$. The definition of P'' is obtained by replacing z with $f(x)$, expressing that the choice of z should only depend on x. This approach is sound, and there is no pair (f, R) that makes $P''(x, y)$ true. □

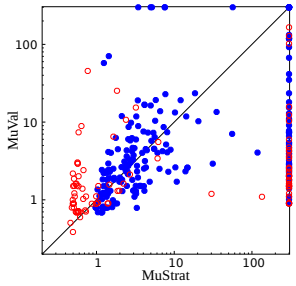

Fig. 1. **MuStrat** vs. **MuVal** on the (non-)term. benchmark set from termCOMP (C Integer).

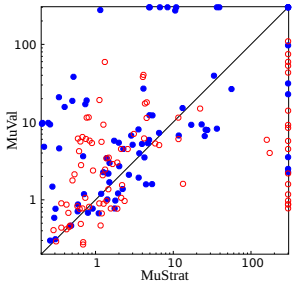

Fig. 2. **MuStrat** vs. **MuVal** on the fixed-point logic benchmark set from [Unno et al. 2023].

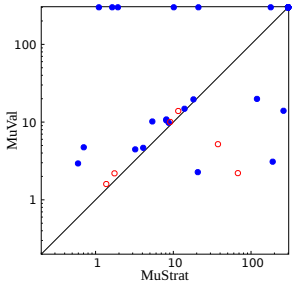

Fig. 3. **MuStrat** vs. **MuVal** on the game solving benchmark set from [Heim and Dimitrova 2024].

7.3 Propotype Implementation and Evaluation

We implemented the new method proposed in this section as MuStrat, a prototype validity checker for a fixpoint-logic over quantified linear integer and real arithmetic. Additionally, for evaluation, we conducted comparative experiments with MuVal [Unno et al. 2023], the existing state-of-the-art fixpoint logic validity checker. We ran the tools on (1) the 335 (non-)termination benchmarks from termCOMP (C Integer category), (2) the 202 fixpoint logic benchmarks provided by the authors of MuVal [Unno et al. 2023], and (3) the 47 game-solving benchmarks from [Heim and Dimitrova 2024], with a timeout of 300 seconds in the StarExec environment. The results are summarized in scatter plots, respectively in Figures 1, 2, and 3. In (1) and (2), although the total number of solved problems is slightly lower than MuVal, MuStrat successfully solved 6 and 7 problems respectively that MuVal timed out on. Additionally, in problems where strategy synthesis is crucial, such as non-termination verification and branching-time properties verification, MuStrat was able to find witnesses faster than MuVal in more cases. (Note that in Figure 1, the red dots represent non-terminating instances.) MuStrat also has an engineering advantage as it uses the highly efficient Spacer as the backend CHC solver, allowing it to quickly solve examples that require the synthesis of complex inductive invariants. On the other hand, MuVal, through CEGIS iterations, can simultaneously synthesize the necessary ranking functions and inductive invariants, which may be inter-dependent, allowing it to solve problem instances requiring complex ranking functions that MuStrat times out on. On the other hand, the dedicated game solver rpgsolve, proposed in the paper that provided the benchmark set (3) [Heim and Dimitrova 2024], successfully solved all but two problems. To achieve performance close to rpgsolve, implementation tuning unrelated to the core algorithm (e.g., efficiently handling the DAG-shaped resolution proofs returned by Spacer without expanding them into trees) is necessary. This is future work for tool development and is out of scope for the present paper. Additionally, it should be noted that rpgsolve targets a limited subset of fixpoint logic, making it inapplicable to the benchmark sets (1) and (2).

8 Related Work and Concluding Remarks

The development of unifying frameworks for algorithms in program analysis and verification is a common theme. Abstract Interpretation [Cousot and Cousot 1977, 1979] is perhaps the most prominent example of a mathematical framework that describes a variety of program analysis algorithms. While in some instances the primal or dual problems in our framework can be seen as solved by computing an abstract fixpoint in a suitable domain (e.g., in CEGAR and in primal-dual Houdini), it seems to us that the entirety of Algorithm 1 is not an instance of abstract

fixpoint computation. We consider it an interesting avenue for future work to better understand the connection between our Lagrangian-based algorithmic framework and abstract interpretation. In particular, primal-dual search may be related to widening and narrowing.

In optimization and constraint solving, the so called inference dual and the relaxation dual (which is a special case of the inference dual) have been observed to underlie many algorithms [Hooker 2000, 2006]. In these terms, the duality of our generalized Lagrangian can be seen as both an inference dual and a relaxation dual, in the sense that, e.g., some $x \in X$ such that $\sup_y L(x, y) = v$ can be seen as a proof that $\inf_x \sup_y L(x, y) \leq v$. In some cases Algorithm 1 can yield what [Hooker 2006] calls a nogood-based search method, but Algorithm 1 seems to be more general. Exploring the connection between our framework and more algorithms and mathematical frameworks for constraint satisfaction and optimization is an interesting avenue for future work.

Applications of linear/semidefinite programming and their (relaxation) duals to verification can be found in the literature. Cousot [2005] proposed an approach to reduce a verification problem into linear or semidefinite programming using Lagrangian relaxation as a key component. Gonnord et al. [2015] synthesizes ranking functions by formulating the problem as a linear programming problem. Their use of extremal counterexamples to guarantee finiteness and therefore termination bears some resemblance to the choice of terms (mentioned at the end of Section 6) that guarantee termination in the Quantified LRA solver discussed in Section 6.

The duality in verification and constraint solving often appears as the duality between the proponent and opponent of a game. For example, the quantified SMT solver by Farzan and Kincaid [2016] studied in Section 6 is presented in terms of games. Many quantified SMT solvers Bjørner and Janota [2015]; Bonacina et al. [2023]; Murphy and Kincaid [2024] are inspired by the game semantics of logical formulas and the duality inherent in it. For example, Murphy and Kincaid [2024] improved the approach by Farzan and Kincaid [2016]: whereas the original method adds a strategy skeleton that could beat the current opponent strategy, the new method seeks a strategy skeleton that not only beats the current opponent strategy but also is winning on a subgame. It is an interesting challenge to see if these advanced ideas can be understood and developed within a Lagrangian-based framework.

An important family of procedures that this paper does not deal with is IC3/PDR [Bradley 2011; Een et al. 2011] and its relatives such as GPDR [Hoder and Bjørner 2012] and Spacer [Komuravelli et al. 2015, 2014; Tsukada and Unno 2024; Vediramana Krishnan et al. 2023]. We developed some candidate Lagrangians aiming to capture the behavior of PDR, some of which seem to capture PDR to a certain extent. Detailed analysis of PDR based on Lagrangians is an important topic for future work. According to Tsukada and Unno [2022], a game solving procedure by Farzan and Kincaid [2018] can be related to PDR, so the analysis of PDR would be beneficial to understand their procedure as well.

The primal-dual algorithms of [Padon et al. 2022; Unno et al. 2023] have inspired the development of our Lagrangian-based framework. Section 4 presented a suitable Lagrangian that shows that the high-level structure of primal-dual Houdini [Padon et al. 2022] is an instance of our framework. We believe the algorithm of [Unno et al. 2023] can similarly be captured by our framework, and we are still developing a suitable Lagrangian. One challenge there is that the primal and dual sides exchange information only under certain validity conditions. Therefore, either developing a suitable Lagrangian or possibly extending our framework is a planned future work.

Acknowledgments

We thank the anonymous reviewers for comments which improved the paper. The research leading to these results has received funding from the European Research Council under the European

Union's Horizon 2020 research and innovation programme (grant agreement No [759102-SVIS]). This research was partially supported by the Israel Science Foundation (ISF) grant No. 2117/23 and JSPS KAKENHI Grant Numbers JP20H04162, JP20H05703, JP19H04084, JP24H00699, JP23K24826, and JP23K24820. This research was partially supported by a research grant from the Center for New Scientists at the Weizmann Institute of Science and by a grant from the Azrieli Foundation.

References

Thomas Ball, Rupak Majumdar, Todd D. Millstein, and Sriram K. Rajamani. 2001a. Automatic Predicate Abstraction of C Programs. In *Proceedings of the 2001 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Snowbird, Utah, USA, June 20-22, 2001*, Michael Burke and Mary Lou Soffa (Eds.). ACM, 203–213. https://doi.org/10.1145/378795.378846

Thomas Ball, Andreas Podelski, and Sriram K. Rajamani. 2001b. Boolean and Cartesian Abstraction for Model Checking C Programs. In *Tools and Algorithms for the Construction and Analysis of Systems, 7th International Conference, TACAS 2001 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2001 Genova, Italy, April 2-6, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2031)*, Tiziana Margaria and Wang Yi (Eds.). Springer, 268–283. https://doi.org/10.1007/3-540-45319-9_19

Dimitris Bertsimas and John N. Tsitsiklis. 1997. *Introduction to linear optimization.* Athena scientific optimization and computation series, Vol. 6. Athena Scientific.

Nikolaj Bjørner and Mikoláš Janota. 2015. Playing with Quantified Satisfaction. In *20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2015) (EPiC Series in Computing)*. EasyChair. https://doi.org/10.29007/vv21

Maria Paola Bonacina, Stéphane Graham-Lengrand, and Christophe Vauthier. 2023. QSMA: A New Algorithm for Quantified Satisfiability Modulo Theory and Assignment. In *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14132)*, Brigitte Pientka and Cesare Tinelli (Eds.). Springer, 78–95. https://doi.org/10.1007/978-3-031-38499-8_5

Stephen P. Boyd and Lieven Vandenberghe. 2014. *Convex Optimization.* Cambridge University Press. https://doi.org/10.1017/CBO9780511804441

Aaron R. Bradley. 2011. SAT-Based Model Checking without Unrolling. In *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 70–87. https://doi.org/10.1007/978-3-642-18275-4_7

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-Guided Abstraction Refinement. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1855)*, E. Allen Emerson and A. Prasad Sistla (Eds.). Springer, 154–169. https://doi.org/10.1007/10722167_15

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. 2005. Abstraction Refinement for Termination. In *Static Analysis Symposium (SAS 2005) (LNCS, Vol. 3672)*. Springer Berlin Heidelberg, 87–101. https://doi.org/10.1007/11547662_8

Byron Cook, Andreas Podelski, and Andrey Rybalchenko. 2006. Termination proofs for systems code. In *Proceedings of the 27th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '06)*. ACM, 415–426. https://doi.org/10.1145/1133981.1134029

Patrick Cousot. 2005. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming. In *Verification, Model Checking, and Abstract Interpretation (LNCS, Vol. 3385)*. Springer Berlin Heidelberg, 1–24. https://doi.org/10.1007/978-3-540-30579-8_1

Patrick Cousot and Radhia Cousot. 1977. Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (Los Angeles, California) *(POPL '77)*. ACM, New York, NY, USA, 238–252. https://doi.org/10.1145/512950.512973

Patrick Cousot and Radhia Cousot. 1979. Systematic Design of Program Analysis Frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (San Antonio, Texas) *(POPL '79)*. ACM, New York, NY, USA, 269–282. https://doi.org/10.1145/567752.567778

Niklas Een, Alan Mishchenko, and Robert Brayton. 2011. Efficient implementation of property directed reachability. In *2011 Formal Methods in Computer-Aided Design (FMCAD)*. 125–134.

Azadeh Farzan and Zachary Kincaid. 2016. Linear Arithmetic Satisfiability via Strategy Improvement. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, Subbarao Kambhampati (Ed.). IJCAI/AAAI Press, 735–743. http://www.ijcai.org/Abstract/16/110

Azadeh Farzan and Zachary Kincaid. 2018. Strategy synthesis for linear arithmetic games. *Proc. ACM Program. Lang.* 2, POPL (2018), 61:1–61:30. https://doi.org/10.1145/3158149

Cormac Flanagan, Rajeev Joshi, and K. Rustan M. Leino. 2001. Annotation inference for modular checkers. *Inf. Process. Lett.* 77, 2-4 (2001), 97–108. https://doi.org/10.1016/S0020-0190(00)00196-4

Cormac Flanagan and K. Rustan M. Leino. 2001. Houdini, an Annotation Assistant for ESC/Java. In *FME 2001: Formal Methods for Increasing Software Productivity, International Symposium of Formal Methods Europe, Berlin, Germany, March 12-16, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2021)*, José Nuno Oliveira and Pamela Zave (Eds.). Springer, 500–517. https://doi.org/10.1007/3-540-45251-6_29

Pranav Garg, Christof Löding, P. Madhusudan, and Daniel Neider. 2014. ICE: A Robust Framework for Learning Invariants. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8559)*, Armin Biere and Roderick Bloem (Eds.). Springer, 69–87. https://doi.org/10.1007/978-3-319-08867-9_5

Yeting Ge and Leonardo Mendonça de Moura. 2009. Complete Instantiation for Quantified Formulas in Satisfiabiliby Modulo Theories. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5643)*, Ahmed Bouajjani and Oded Maler (Eds.). Springer, 306–320. https://doi.org/10.1007/978-3-642-02658-4_25

Patrice Godefroid, Aditya V. Nori, Sriram K. Rajamani, and Sai Deep Tetali. 2010. Compositional may-must program analysis: unleashing the power of alternation. In *Proceedings of the 37th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL '10)*. ACM. https://doi.org/10.1145/1706299.1706307

Laure Gonnord, David Monniaux, and Gabriel Radanne. 2015. Synthesis of ranking functions using extremal counterexamples. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '15)*. ACM, 608–618. https://doi.org/10.1145/2737924.2737976

Philippe Heim and Rayna Dimitrova. 2024. Solving Infinite-State Games via Acceleration. *Proceedings of the ACM on Programming Languages* 8, POPL (Jan. 2024), 1696–1726. https://doi.org/10.1145/3632899

Kryštof Hoder and Nikolaj Bjørner. 2012. Generalized Property Directed Reachability. In *Theory and Applications of Satisfiability Testing – SAT 2012*. Springer Berlin Heidelberg, 157–171. https://doi.org/10.1007/978-3-642-31612-8_13

J. N. Hooker. 2000. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction.* Wiley, New York. https://doi.org/10.1002/9781118033036

J. N. Hooker. 2006. Duality in Optimization and Constraint Satisfaction. In *The 3rd International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR 2006) (LNCS, Vol. 3990)*. Springer Berlin Heidelberg, 3–15. https://doi.org/10.1007/11757375_3

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. 2010. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, Jeff Kramer, Judith Bishop, Premkumar T. Devanbu, and Sebastián Uchitel (Eds.). ACM, 215–224. https://doi.org/10.1145/1806799.1806833

Ranjit Jhala and Kenneth L. McMillan. 2006. A Practical and Complete Approach to Predicate Refinement. In *TACAS '06 (Lecture Notes in Computer Science, Vol. 3920)*. Springer, 459–473.

Anvesh Komuravelli, Nikolaj Bjørner, Arie Gurfinkel, and Kenneth L. Mcmillan. 2015. Compositional verification of procedural programs using horn clauses over integers and arrays. In *2015 Formal Methods in Computer-Aided Design (FMCAD)*. IEEE. https://doi.org/10.1109/fmcad.2015.7542257

Anvesh Komuravelli, Arie Gurfinkel, and Sagar Chaki. 2014. SMT-Based Model Checking for Recursive Programs. In *Computer Aided Verification (CAV)*. Springer International Publishing, 17–34. https://doi.org/10.1007/978-3-319-08867-9_2

Satoshi Kura, Hiroshi Unno, and Ichiro Hasuo. 2021. Decision Tree Learning in CEGIS-Based Termination Analysis. In *CAV '21 (Lecture Notes in Computer Science)*. Springer, 75–98.

Xie Li, Yi Li, Yong Li, Xuechao Sun, Andrea Turrini, and Lijun Zhang. 2020. *SVMRanker: A General Termination Analysis Framework of Loop Programs via SVM.* ACM, 1635–1639.

Charlie Murphy and Zachary Kincaid. 2024. Quantified Linear Arithmetic Satisfiability via Fine-Grained Strategy Improvement. In *CAV 2024*. Springer, Cham, 89–109.

Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. 2006. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *J. ACM* 53, 6 (2006), 937–977. https://doi.org/10.1145/1217856.1217859

Oded Padon, James R. Wilcox, Jason R. Koenig, Kenneth L. McMillan, and Alex Aiken. 2022. Induction duality: primal-dual search for invariants. *Proceedings of the ACM on Programming Languages* 6, POPL (Jan. 2022), 1–29. https://doi.org/10.1145/3498712

A. Podelski and A. Rybalchenko. 2004. Transition invariants. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS 2004)*. IEEE. https://doi.org/10.1109/lics.2004.1319598

Takeshi Tsukada and Hiroshi Unno. 2022. Software model-checking as cyclic-proof search. *Proceedings of the ACM on Programming Languages* 6, POPL (jan 2022), 1–29. https://doi.org/10.1145/3498725

Takeshi Tsukada and Hiroshi Unno. 2024. Inductive Approach to Spacer. *Proc. ACM Program. Lang.* 8, PLDI, Article 227 (June 2024), 24 pages. https://doi.org/10.1145/3656457

Takeshi Tsukada, Hiroshi Unno, Oded Padon, and Sharon Shoham. 2024. A Primal-Dual Perspective on Program Verification Algorithms (extended version). (2024). Available on arXiv.

Hiroshi Unno, Tachio Terauchi, Yu Gu, and Eric Koskinen. 2023. Modular Primal-Dual Fixpoint Logic Solving for Temporal Verification. *Proc. ACM Program. Lang.* 7, POPL (2023), 2111–2140. https://doi.org/10.1145/3571265

Hiroshi Unno, Tachio Terauchi, and Eric Koskinen. 2021. Constraint-Based Relational Verification. In *CAV '21 (Lecture Notes in Computer Science)*. Springer, 742–766.

Hari Govind Vediramana Krishnan, YuTing Chen, Sharon Shoham, and Arie Gurfinkel. 2023. Global guidance for local generalization in model checking. *Formal Methods in System Design* (March 2023). https://doi.org/10.1007/s10703-023-00412-3