Pseudo-Boolean Proof Logging for Optimal Classical Planning

Simon Dold[*1], **Malte Helmert**[*1], **Jakob Nordström**[*2, 3], **Gabriele Röger**[*1], **Tanja Schindler**[*1]

[1]University of Basel
[2]University of Copenhagen
[3]Lund University
{simon.dold, malte.helmert, gabriele.roeger, tanja.schindler}@unibas.ch, jn@di.ku.dk

Abstract

We introduce lower-bound certificates for classical planning tasks, which can be used to prove the unsolvability of a task or the optimality of a plan in a way that can be verified by an independent third party. We describe a general framework for generating lower-bound certificates based on pseudo-Boolean constraints, which is agnostic to the planning algorithm used.

As a case study, we show how to modify the A* algorithm to produce proofs of optimality with modest overhead, using pattern database heuristics and $h^{\max}$ as concrete examples. The same proof logging approach works for any heuristic whose inferences can be efficiently expressed as reasoning over pseudo-Boolean constraints.

Introduction

Optimal classical planning algorithms make three promises: that the plans they produce are correct, that no cheaper plans achieving the goal exist, and that any task reported as unsolvable actually is. As McConnell et al. (2011) argue in their seminal work on *certifying algorithms*, there are many good reasons not to accept such promises blindly. Instead, a *certifying planning algorithm* outputs some kind of proof (a *certificate*) that an independent third party can use to verify the truthfulness of the planner's claims.

For the correctness of plans, such a verification is performed routinely: the generated plans themselves serve as certificates for this, and plan validation tools such as VAL (Howey and Long 2003) or INVAL (Haslum 2016) can be used to check that the produced plans are correct. For unsolvability, Eriksson et al. (Eriksson, Röger, and Helmert 2017, 2018; Eriksson and Helmert 2020) recently introduced two forms of unsolvability certificates that cover very diverse planning algorithms.

Certifying the *optimality* of plans, in contrast, is still in its infancy. The only existing work in this direction is a paper by Mugdan, Christen, and Eriksson (2023) which describes two approaches: one based on a compilation to unsatisfiability, and one based on an extension of the unsatisfiability proof system of Eriksson, Röger, and Helmert (2018). The first approach is in general not computationally feasible, as it requires a task reformulation that increases the number of state variables and actions exponentially in the size of the planning task. The second approach does not share this weakness, but is not sufficiently general to encompass a wide range of planning approaches. Mugdan, Christen, and Eriksson identify five essential properties for optimality certificates: *soundness* (if an optimality certificate is accepted by the verifier, then optimality holds), *completeness* (if a solution is optimal, then a certificate of its optimality exists), *efficient generation* (a non-certifying algorithm can be changed to produce certificates with reasonable, at most polynomial overhead), *efficient verification* (the verifier runtime is polynomial in the task and certificate size), and *generality* (certificates can be efficiently produced by a wide variety of different planning algorithms rather than being algorithm-specific).

Mugdan, Christen, and Eriksson show that their approach is sound and complete, but critically comment that "the other three properties for practical usability do not have clear-cut answers". Compared to the unsolvability proof system of Eriksson et al., their approach appears much more specific to heuristic forward search using heuristics that are themselves based on some kind of forward search, such as the $h^{\max}$ heuristic (Bonet and Geffner 2001). For example, it is not at all clear how to certify heuristics computed in a backward direction, such as pattern databases (Edelkamp 2001) or other abstraction heuristics without redoing the abstract state space exploration for every heuristic evaluation.

While in the planning community the interest in certifying algorithms is quite recent, they are standard in the SAT community, where unsatisfiability certificates based on proof systems like DRAT (Heule, Hunt, and Wetzler 2013) and LRAT (Cruz-Filipe et al. 2017) are required for participating in SAT competitions and supported by formally verified checkers (Tan, Heule, and Myreen 2023; Lammich 2020). A recently proposed alternative are pseudo-Boolean proofs based on cutting planes, which are supported by the formally verified checker VeriPB (Bogaerts et al. 2022). Unlike the other proof systems mentioned, cutting planes are able to directly incorporate linear arithmetic, making them very appealing for optimization problems such as optimal classical planning. Indeed, VeriPB has been applied to a wide range of optimization problems such as MaxSAT (Berg et al. 2023), ILP presolving (Hoen et al. 2024), constraint pro-

[*]These authors contributed equally.

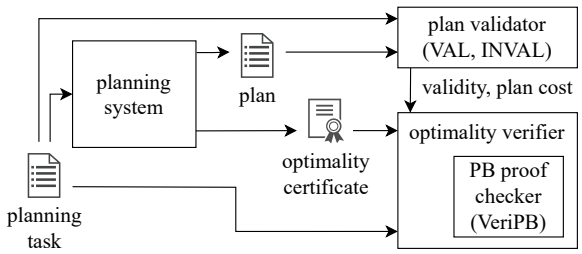

Figure 1: Interaction of Components

gramming (Gocht, McCreesh, and Nordström 2022; McIlree and McCreesh 2023; McIlree, McCreesh, and Nordström 2024), and dynamic programming (Demirović et al. 2024).

In this work, we propose to certify plan optimality by means of such pseudo-Boolean proofs. We claim that our optimality certificates have all five desirable properties discussed above: they are sound, complete, efficiently generatable, efficiently verifiable, and general. (For space reasons, the generality of the proof system must remain a conjecture for the time being, but at least we show that we can cover the techniques covered by Mugdan et al. as well as pattern database heuristics that their approach struggles with.)

The overall concept is shown in Figure 1. In addition to a plan, the certifying planning system produces an optimality certificate that proves that the input task has no cheaper solution. A plan validator such as VAL verifies that the plan solves the task and determines its cost. The optimality verifier has access to the original task, the plan cost determined by the validator, and the certificate from the planning system. On this basis it verifies that the plan is indeed optimal.

The certificate is based on cutting planes proofs with reification (Bogaerts et al. 2023), in which the atomic pieces of knowledge are pseudo-Boolean constraints. Such constraints allow us to reason conveniently about costs of actions and bounds on costs to reach a state. Roughly speaking, the certificate describes an overapproximation of which states can be reached at which cost and shows that no goal state can be reached at a cost below the claimed optimal plan cost. The heart of the proof is a standard pseudo-Boolean proof that can be verified by an off-the-shelf proof checker such as the VeriPB system.

The paper is structured as follows: we introduce the general framework for certifying plan optimality by providing an encoding of planning tasks as pseudo-Boolean constraints and defining lower-bound certificates. Next, we define heuristic certificates, which represent the concept of an admissible heuristic estimate as pseudo-Boolean constraints. We then show how heuristic certificates can be used to integrate proof-logging into A*, capturing the arguments for optimality of A* with an admissible heuristic on a given state space as pseudo-Boolean constraints. Finally, we describe how heuristic certificates for pattern database heuristics and $h^{\max}$ can be generated, which amounts to proving the admissibility of these heuristics with pseudo-Boolean constraints.

Background

We first introduce the STRIPS planning formalism (Fikes and Nilsson 1971), followed by pseudo-Boolean constraints and the cutting planes with reification proof system.

STRIPS Planning Tasks

A STRIPS planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ consists of a finite set $\mathcal{V}$ of propositional *state variables*, a finite set $\mathcal{A}$ of *actions*, the *initial state* $I \subseteq \mathcal{V}$ and the *goal* $G \subseteq \mathcal{V}$.

A *state* $s \subseteq \mathcal{V}$ of Π induces a variable assignment ρ_s that maps every $v \in s$ to 1 and every $v \notin s$ to 0. Each action $a \in \mathcal{A}$ is a tuple $\langle \mathit{pre}(a), \mathit{add}(a), \mathit{del}(a), \mathit{cost}(a) \rangle$, where $\mathit{pre}(a) \subseteq \mathcal{V}$ is the set of *preconditions*, $\mathit{add}(a) \subseteq \mathcal{V}$ is the set of *add effects*, $\mathit{del}(a) \subseteq \mathcal{V} \setminus \mathit{add}(a)$ is the set of *delete effects*, and $\mathit{cost}(a) \in \mathbb{N}_0$ is the *cost* of a. We write $\mathit{evars}(a)$ for the set $\mathit{add}(a) \cup \mathit{del}(a)$ of affected variables. Action a is *applicable* in state s if $\mathit{pre}(a) \subseteq s$. The *successor state* is $s[\![a]\!] = (s \setminus \mathit{del}(a)) \cup \mathit{add}(a)$. For a sequence $\pi = a_1, \ldots, a_n$ of actions that are successively applicable in state s, we write $s[\![\pi]\!]$ for the resulting state $s[\![a_1]\!] \ldots [\![a_n]\!]$. If this state is a *goal state*, i.e. $G \subseteq s[\![\pi]\!]$, then π is a *plan* for s. A plan for the initial state I is a plan for task Π. The cost of π is $\mathit{cost}(\pi) = \sum_{i=1}^{n} \mathit{cost}(a_i)$. A plan is *optimal* if there is no plan of lower cost. Task Π is solvable if there is a plan for Π, otherwise it is *unsolvable*.

Pseudo-Boolean Formulas

A *Boolean* variable has domain $\{0, 1\}$. A literal ℓ is a Boolean variable x or its negation $\bar{x}$. The negation of a literal $\ell = \bar{x}$ is $\bar{\ell} = x$. A *pseudo-Boolean (PB) constraint* (in normalized form) over a finite set $X = \{x_1, \ldots, x_n\}$ of Boolean variables is an inequality

$$\sum\nolimits_i a_i \ell_i \geq A,$$

where all literals ℓ_i are over distinct variables from X, $A \in \mathbb{N}_0$ and all coefficients a_i are from $\mathbb{N}_0$. We will also write linear constraints more flexibly, but they can always be transformed to normalized form by simple algebraic transformations. As syntactic sugar, we will also write $(\ell_1 \wedge \cdots \wedge \ell_n) \rightarrow \ell$ as abbreviation for $\overline{\ell_1} + \ldots + \overline{\ell_n} + \ell \geq 1$.

A *solution* of the constraint is an assignment $\rho : X \rightarrow \{0, 1\}$ such that the inequality is satisfied by replacing every variable x_i with $\rho(x_i)$ and every negated variable $\bar{x}_i$ with $1 - \rho(x_i)$. We also use assignments for all literals, implicitly requiring that $\rho(\bar{x}) = 1 - \rho(x)$ for all variables x.

For constraint $C \doteq \sum_i a_i \ell_i \geq A$, the negation $\neg C$ is the normalized form of $\sum_i a_i \ell_i \leq A - 1$.

A *PB formula* (Buss and Nordström 2021) is a finite set $\mathcal{C}$ of PB constraints over a set X of variables. An assignment $\rho : X \rightarrow \{0, 1\}$ is a *model* of the formula if it is a solution of all constraints. If $\mathcal{C}$ has no model, it is *unsatisfiable*. We say that a constraint C is *implied* by a PB formula $\mathcal{C}$ (written $\mathcal{C} \models C$) if every model of $\mathcal{C}$ is a solution of C and that PB formula $\mathcal{D}$ is implied by $\mathcal{C}$ (written $\mathcal{C} \models \mathcal{D}$) if $\mathcal{C} \models D$ for all $D \in \mathcal{D}$.

For a partial variable assignment $\rho : X \nrightarrow \{0, 1\}$ and constraint C, we write $C{\restriction_\rho}$ for the constraint obtained from C by replacing each variable v in the domain of ρ by $\rho(v)$ and normalizing.

Cutting Planes with Reification Proof System

The VeriPB proof system (Bogaerts et al. 2023) is an extension of the cutting planes proof system. We use a subset of the VeriPB proof system which we call *cutting planes with reification* (CPR).

Proofs in the cutting planes proof system (Cook, Coullard, and Turán 1987) are built from one axiom and three derivation rules. For any literal ℓ, the *literal axiom* allows us to derive $\ell \geq 0$ without prerequisites. If we already have $\sum_i a_i\ell_i \geq A$ and $\sum_i b_i\ell_i \geq B$, we can derive

- $\sum_i (c_A a_i + c_B b_i)\ell_i \geq c_A A + c_B B$ for every $c_A, c_B \in \mathbb{N}_0$ (*linear combination*),
- $\sum_i \lceil a_i/c \rceil \ell_i \geq \lceil A/c \rceil$ for every $c \in \mathbb{N}^+$ (*division*), and
- $\sum_i \min\{a_i, A\}\ell_i \geq A$ (*saturation*).

All constraints that can be derived from a PB formula $\mathcal{C}$ by these rules are implied by $\mathcal{C}$.

Reverse Unit Propagation A constraint $C \in \mathcal{C}$ *unit propagates* literal ℓ under partial assignment ρ if all models of $C|_\rho$ assign ℓ to 1. When this happens, we can extend ρ with $\ell \mapsto 1$ and unit-propagate further literals under the extended assignment. If this process derives a conflict (assigning 0 and 1 to the same variable) starting from the empty variable assignment, then $\mathcal{C}$ is unsatisfiable.

Formula $\mathcal{C}$ implies constraint C by *reverse unit propagation* (RUP) if $\mathcal{C} \cup \{\neg C\}$ unit propagates to a conflict from the empty assignment. Any constraint implied by RUP can be derived by a cutting plane proof. As in VeriPB, our proof system allows adding any constraint implied by RUP directly in a single step. This is a useful shortcut that drastically compresses many cutting plane proofs.

Reification In addition to cutting planes reasoning, our proof system also allows *reification*, i.e., introducing a new variable that represents the truth value of a constraint. If C is a constraint and r is a new variable, we write $r \Leftrightarrow C$ to express that variable r must be 1 if C is true under the assignment and 0 otherwise. If C is $\sum_i a_i\ell_i \geq A$, this is a shorthand notation for the two constraints

$$A\bar{r} + \sum\nolimits_i a_i\ell_i \geq A, \text{ and}$$
$$(M - A + 1)r + \sum\nolimits_i a_i\bar{\ell}_i \geq M - A + 1$$

where $M = \sum_i a_i$. We use the notation $r \Rightarrow C$ for the first and $r \Leftarrow C$ for the second constraint.

To summarize, a *CPR proof* consists of a sequence of derivation steps from the cutting plane proof system (literal axiom, linear combination, division, and saturation), constraints derived by RUP, and reifications. In addition, we allow the *redundance-based strengthening* (RED) rule from VeriPB, which can be understood as a form of proof by contradiction. We only use it in an extended version of this paper (Dold et al. 2025) and therefore describe it there.

Lower-Bound Certificates

We propose certificates that prove that there is no plan of lower cost than a given bound B. A typical, but not the only, application of such lower-bound certificates is to certify that a plan of cost B is optimal. Before we define the full framework, we first introduce how we encode planning tasks by means of PB formulas.

Encoding Planning Tasks

The PB encoding of task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ uses the propositional variables from $\mathcal{V}$ as Boolean variables, and variables $\mathcal{V}_c = \{c_0, \ldots, c_{\lceil \log_2(B) \rceil}\}$ as a binary representation of a number in the range $0, \ldots, B$. These variables allow us to represent pairs $\langle s, c \rangle$ consisting of a state s and a number $c \leq B$. In the following, one can use the intuition that s is a state of the task, which has been reached incurring cost c. In addition, we introduce a number of reification variables.

Reification variable r_I is true in a model iff the state variables encode the initial state:

$$r_I \Leftrightarrow \sum_{v \in I} v + \sum_{v \in \mathcal{V} \setminus I} \bar{v} \geq |\mathcal{V}| \tag{1}$$

For the goal, we introduce a reification variable r_G, which is true in a model iff the state variables encode a goal state:

$$r_G \Leftrightarrow \sum_{v \in G} v \geq |G| \tag{2}$$

For the actions, we encode transitions from a state s to successor state $s[\![a]\!]$ in a similar way to symbolic search (e.g., Edelkamp and Kissmann 2009) or planning as satisfiability (e.g., Rintanen, Heljanko, and Niemelä 2006), encoding the successor state by means of additional variables v' for each state variable v. The variables c_i encode a cost by which state s can be reached, and analogously we use a variable c'_i for each c_i to encode the cost to reach $s[\![a]\!]$ via this transition. For this purpose, we need constraints that ensure that the difference between the two values corresponds to the cost of the action. We do this by means of additional reification variables $\Delta c^{=k}$ that express that the difference between the two numbers is k:

$$\Delta c^{=k} \Leftrightarrow \sum\nolimits_{i=0}^{\lceil \log_2 B \rceil} 2^i c'_i - \sum\nolimits_{i=0}^{\lceil \log_2 B \rceil} 2^i c_i = k \tag{3}$$

To express that the variables c_i or c'_i encode a value that is at least k for some $k \in \{0, \ldots, B\}$, we use reification variables $\mathit{cost}_{\geq k}$ and $\mathit{cost}'_{\geq k}$:

$$\mathit{cost}_{\geq k} \Leftrightarrow \sum\nolimits_{i=0}^{\lceil \log_2 B \rceil} 2^i c_i \geq k \tag{4}$$

$$\mathit{cost}'_{\geq k} \Leftrightarrow \sum\nolimits_{i=0}^{\lceil \log_2 B \rceil} 2^i c'_i \geq k \tag{5}$$

Note that we do not introduce these reification variables for all values of k up to B, which would require an exponential number of variables in the encoding size of B. Rather, we lazily introduce only the variables used by the proof.

For handling the state variables v that are not affected by an action, we introduce reification variables $eq_{v,v'}$ that are true in a model iff it assigns v and v' the same value:

$$\begin{aligned} eq_{v,v'} &\Leftrightarrow leq_{v,v'} + geq_{v,v'} \geq 2 \\ geq_{v,v'} &\Leftrightarrow v + \overline{v'} \geq 1 \\ leq_{v,v'} &\Leftrightarrow \bar{v} + v' \geq 1 \end{aligned} \tag{6}$$

For each action $a \in \mathcal{A}$ we introduce a variable r_a expressing that whenever the action is applied, the cost is increased by $\mathit{cost}(a)$, the action precondition is satisfied, the primed variables truthfully represent the successor state, and the successor cost is within the cost bound:

$$r_a \Rightarrow \Delta c^{=\mathit{cost}(a)} + \sum_{v \in \mathit{pre}(a)} v + \sum_{v \in \mathit{add}(a)} v' + \sum_{v \in \mathit{del}(a)} \overline{v'} + \sum_{v \in \mathcal{V} \setminus \mathit{evars}(a)} eq_{v,v'} + \overline{\mathit{cost}'_{\geq B}} \geq 2 + |\mathit{pre}(a)| + |\mathcal{V}| \tag{7}$$

Observe that this constraint represents a conjunction: all literals must be true to meet the bound $2 + |\mathit{pre}(a)| + |\mathcal{V}|$.

Finally, reification variable r_T encodes that a state transition happens, i.e., some action variable is selected:

$$r_T \Leftrightarrow \sum_{a \in \mathcal{A}} r_a \geq 1 \tag{8}$$

This representation allows selecting several actions at the same time, but only if they all lead to the same state change under the same cost.

Definition 1. *For planning task $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ and cost bound $B \in \mathbb{N}_0$, a* PB task encoding *is a tuple $\mathcal{E}_\Pi = \langle \mathcal{C}_\Pi, r_I, r_G, r_T \rangle$, where $\mathcal{C}_\Pi = \langle \mathcal{C}_{\text{init}}, \mathcal{C}_{\text{goal}}, \mathcal{C}_{\text{trans}}, \mathcal{C}_{\geq} \rangle$ such that $\mathcal{C}_{\text{init}}$, $\mathcal{C}_{\text{goal}}$, $\mathcal{C}_{\text{trans}}$ and $\mathcal{C}_{\geq}$ are sets of reifications from equations* (1)–(8)*, $\mathcal{C}_{\text{init}}$ from* (1) *and* (4)*, $\mathcal{C}_{\text{goal}}$ from* (2) *and* (4)*, $\mathcal{C}_{\text{trans}}$ from* (3)–(8)*, $\mathcal{C}_{\geq}$ from* (3)–(5) *and r_I, r_G, and r_T are the reification variables introduced in* (1)*,* (2)*, and* (8)*.*

Certifying Unsolvability under Cost Bound

A certificate shows that the task is unsolvable under a cost bound B, i.e., there is no plan π with $\mathit{cost}(\pi) < B$.

Intuitively, it is based on an invariant φ which represents an overapproximation of the reachable state-cost pairs. An invariant in general is a property that is preserved through action applications, i.e. (1) whenever φ is true for state s and cost c, and action a is applicable in s, then φ is true for state $s[\![a]\!]$ and cost $c + \mathit{cost}(a)$.[1] In addition, we require that (2) φ is true for the initial state and cost 0. Together, (1) and (2) ensure that φ is true in all reachable state-cost pairs. If in addition, (3) φ is *not* true for any goal state with a cost strictly lower than B, this implies that the task cannot be solved with cost $< B$: initially, the invariant is true (2), it is impossible to change this by an action application (1), so for all reachable goal states the incurred cost is at least B (3).

In our certificates, the invariant will be defined by a sequence of PB reifications, and the certificate must prove the three properties above by means of three separate CPR proofs. We call property (1) the *inductivity lemma*, property (2) the *initial state lemma* and property (3) the *goal lemma*.

We can think of the representation of the invariant as a circuit where each gate evaluates a PB constraint. The state variables and cost bits c_i are the inputs to the circuit, and the output of the circuit determines whether the invariant is true for the input state-cost pair. We now formalize this notion.

[1] For unsolvability under a cost bound, it is sufficient to only require the invariant property for action applications within the cost bound. We implicitly do this because our task encoding only permits such action applications, cf. Eq. (7).

Definition 2. *A* PB circuit *with input variables V is a pair $\mathcal{R} = \langle R, r \rangle$, where*

- *R is a sequence $\langle r_1 \Leftrightarrow \varphi_1, \ldots, r_n \Leftrightarrow \varphi_n \rangle$ of PB reifications such that each PB constraint φ_i only has non-zero coefficients for variables from $V \cup \{r_j \mid j < i\}$, and*
- *$r \in \{r_1, \ldots, r_n\}$ is the* output variable*.*

Slightly abusing notation, we also interpret a sequence of PB constraints as the PB formula consisting of its components (i.e., treat the sequence as a set). For a PB circuit $\langle R, r \rangle$ with input variables $\mathcal{V} \cup \mathcal{V}_c$, we will write $M(R, r)$ for the represented set of pairs $\langle s, c \rangle$. In other words, $M(R, r)$ contains the pair $\langle s, c \rangle$ iff there exists a model ρ of $R \cup \{r = 1\}$, such that $s = \{v \in \mathcal{V} \mid \rho(v) = 1\}$ and $c = \sum_i 2^i \rho(c_i)$.

Putting the pieces together, we get the following formal definition of lower-bound certificates. Remember that in the encoding of the planning task, $\mathcal{V}$ contains the state variables and $\mathcal{V}_c$ the binary variables for representing cost.

Definition 3. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ be a planning task and $B \in \mathbb{N}_0$ a cost bound. Let $\mathcal{E}_\Pi = \langle \mathcal{C}_\Pi, r_I, r_G, r_T \rangle$ be a PB task encoding for Π and B.*

A lower-bound certificate for Π with bound B *is a tuple $\langle \langle \mathcal{C}_\varphi, r_\varphi \rangle, \mathcal{P}_{\text{init}}, \mathcal{P}_{\text{ind}}, \mathcal{P}_{\text{goal}} \rangle$, where*

- *$\langle \mathcal{C}_\varphi, r_\varphi \rangle$ is a PB circuit with input variables $\mathcal{V} \cup \mathcal{V}_c$ not mentioning a primed variable.*
- ***initial state lemma:** $\mathcal{P}_{\text{init}}$ is a CPR proof for $\mathcal{C}_{\text{init}} \cup \mathcal{C}_\varphi \cup \mathcal{C}_{\geq} \models (r_I \wedge \overline{\mathit{cost}_{\geq 1}}) \rightarrow r_\varphi$.*
- ***goal lemma:** $\mathcal{P}_{\text{goal}}$ is a CPR proof for $\mathcal{C}_{\text{goal}} \cup \mathcal{C}_\varphi \cup \mathcal{C}_{\geq} \models (r_G \wedge r_\varphi) \rightarrow \mathit{cost}_{\geq B}$.*
- ***inductivity lemma:** $\mathcal{P}_{\text{ind}}$ is a CPR proof for $\mathcal{C}_{\text{trans}} \cup \mathcal{C}_\varphi \cup \mathcal{C}'_\varphi \cup \mathcal{C}_{\geq} \models (r_\varphi \wedge r_T) \rightarrow r'_\varphi$, where $\mathcal{C}'_\varphi$ is a copy of $\mathcal{C}_\varphi$ where all PB variables are replaced with their primed version.*

Lower-bound certificates are sound:

Theorem 1. *If there is a lower-bound certificate for planning task Π with bound B, then the task has no plan π with $\mathit{cost}(\pi) < B$.*

Proof. Assume there is a plan $\pi = \langle a_1, \ldots, a_n \rangle$ with $\mathit{cost}(\pi) < B$. It induces a sequence of state-cost pairs $\langle s_0, g_0 \rangle, \ldots, \langle s_n, g_n \rangle$ such that s_0 is the initial state, $g_0 = 0$, and for all $i \in \{1, \ldots, n\}$, $s_i = s_{i-1}[\![a_i]\!]$ and $g_i = g_{i-1} + \mathit{cost}(a_i)$. Note that s_n is a goal state and $g_n = \mathit{cost}(\pi) < B$.

By the initial state lemma, $\langle s_0, g_0 \rangle$ is in $M(\mathcal{C}_\varphi, r_\varphi)$. In r_T, the PB encoding captures all action applications that do not exceed the cost bound, which is the case for the action applications in π because $\mathit{cost}(\pi) < B$. Thus, by the inductivity lemma, we get that $\langle s_i, g_i \rangle \in M(\mathcal{C}_\varphi, r_\varphi)$ for all $i \in \{1, \ldots, n\}$. By the goal lemma, we have for all $\langle s, g \rangle \in M(\mathcal{C}_\varphi, r_\varphi)$ that $g \geq B$ if s is a goal state. Since s_n is a goal state, and $\langle s_n, g_n \rangle \in M(\mathcal{C}_\varphi, r_\varphi)$, this implies that $g_n \geq B$, a contradiction to $g_n < B$. □

The verifier component (see Fig. 1) receives the planning task, the lower-bound certificate, and the validated plan cost. Inside, it confirms that the reifications in $\mathcal{E}_\Pi$ are a system of PB constraints that encode the original planning task. Additionally, it generates $\langle \mathcal{C}'_\varphi, r'_\varphi \rangle$ based on $\langle \mathcal{C}_\varphi, r_\varphi \rangle$ by creating

a copy of each PB reification in $\mathcal{C}_\varphi$ where all variables are replaced by their primed counterparts. It then verifies that the proofs $\mathcal{P}_{\text{init}}$, $\mathcal{P}_{\text{goal}}$ and $\mathcal{P}_{\text{ind}}$ actually derive the statements in Definition 3. Only if all these steps are successful, the verifier confirms that the provided plan cost is optimal.

This concludes our formalization of lower-bound certificates and their verification. Lower-bound certificates are efficiently verifiable: the verifier runs in polynomial time because the underlying PB proof checker does and all steps other than the proof checking are easy to perform in polynomial time. In the rest of the paper we describe a case study of how certain optimal classical planning algorithms can be augmented to efficiently generate lower-bound certificates. Because the algorithms we discuss are complete, this also shows that lower-bound certificates are complete.

Proof-Logging Heuristic Search

We now examine how a planner can be transformed into a proof-logging system, in our case a planner that emits a lower-bound certificate for the cost of the found plan. The idea is to already log most relevant information about its internal reasoning during its normal operation, keeping the overhead for generating the proof low.

A* search (Hart, Nilsson, and Raphael 1968) with an admissible heuristic h, i.e. a distance estimator that provides a lower bound on the goal distance of a state, is the most common approach for optimal planning as heuristic search.

It maintains a priority queue *Open* of states s, ordered by $f = g + h(s)$, where g is the best known cost to reach s from the initial state. The open list is initialized with the initial state. A* removes a state s from the list and expands it until it removes a goal state. A state expansion generates all successor states and adds them to the open list if the implicit path via s improves its g-value.

One challenge when transforming the approach into a proof-logging system is that the bound B that should be certified is only known once a plan has been found. To handle this, we will use "placeholder" PB variables $K_{\geq l}$ and $K'_{\geq l}$ with integers l. Once B is known, the proof will be augmented by reifications that give them the intended meaning. The idea is that they represent $\mathit{cost}_{\geq l}$ and $\mathit{cost}'_{\geq l}$, but l will not necessarily be in the range $\{0, \dots, B\}$. For $l < 0$, we will instead use 0, and for $l > B$, we will use B. So the reifications for these variables will be

$$K_{\geq l} \Leftrightarrow \mathit{cost}_{\geq \min\{B, \max\{0,l\}\}} \text{ and} \tag{9}$$

$$K'_{\geq l} \Leftrightarrow \mathit{cost}'_{\geq \min\{B, \max\{0,l\}\}} \tag{10}$$

By $\mathcal{K}$, we denote the (infinite) set of all such variables. Upon termination A* adds a set $\mathcal{C}_K$ of reifications for the finitely many such variables that actually occur in the proof. In the proofs in this paper, we will silently switch from $K_{\geq \dots}$ to the corresponding $\mathit{cost}_{\geq \dots}$ expression, implicitly using (9).

The following two lemmas establish that these clipped costs behave as expected.[2] Intuitively, the first lemma says that if the costs represented by the variables in $\mathcal{V}_c$ exceed $j + k$, they also exceed the smaller value j:

Lemma 1. *Let $j \in \mathbb{Z}$ and $k \in \mathbb{N}_0$. It is possible to derive $\mathit{cost}_{\geq \min\{B, \max\{0, j+k\}\}} \rightarrow \mathit{cost}_{\geq \min\{B, \max\{0,j\}\}}$ from $\mathcal{C}_{\geq}$.*

The second lemma informally states that if the costs already exceed l and we spend cost m, the successor cost exceeds $l + m$ (clipping all costs into $\{0, \dots, B\}$).

Lemma 2. *For $l \in \mathbb{Z}$ and $m \in \mathbb{N}_0$ it is possible to derive $(\mathit{cost}_{\geq \min\{B, \max\{0,l\}\}} \wedge \Delta c^{=m}) \rightarrow \mathit{cost}'_{\geq \min\{B, \max\{0, l+m\}\}}$ from $\mathcal{C}_{\geq}$.*

Proof-Logging Heuristics

Parts of the overall proof must be contributed by the heuristic. Intuitively, this contribution can be seen as a certificate that shows that it is impossible to reach the goal with a total cost strictly less than B from a state s that has been reached incurring a cost of at least c. For an admissible heuristic, this will naturally be the case for all c where $c + h(s) \geq B$. We will later see examples for different heuristics where some use the same invariant for all states and others generate a new invariant for each evaluated state.

Throughout its execution (a potential initialization phase and evaluations for a number of states), the heuristic maintains its own PB circuit $\langle H, r^h \rangle$ with input variables $\mathcal{V}$ and $\mathcal{K}$. We assume that the heuristic uses its own namespace, so it does not introduce reification variables that are also introduced by the search or some other heuristic. Whenever the search uses the heuristic to evaluate a state s, the heuristic not only returns the estimate but also a PB variable r^h_s with the following requirements:

- $M(H, r^h_s)$ contains all pairs $\langle s, \hat{c} \rangle$ with $\hat{c} + h(s) \geq B$.
- If $M(H, r^h_s)$ contains pair $\langle \hat{s}, \hat{c} \rangle$, it contains all pairs $\langle \tilde{s}, \tilde{c} \rangle$ such that $\tilde{s}$ is reachable from $\hat{s}$ with overall cost $\tilde{c} < B$ (i.e., there is an action sequence π with $\hat{s}[\![\pi]\!] = \tilde{s}$ and $\tilde{c} = \hat{c} + \mathit{cost}(\pi) < B$).
- $M(H, r^h_s)$ contains no pair $\langle \hat{s}, \hat{c} \rangle$ where $\hat{s}$ is a goal state and $\hat{c} < B$.

This motivation leads to the following definition:

Definition 4. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$ be a planning task and $B \in \mathbb{N}_0$ a cost bound. Let $\langle \mathcal{C}_\Pi, r_I, r_G, r_T \rangle$ be the PB task encoding for Π and B and s be a state over $\mathcal{V}$.*

A heuristic certificate for state s of Π with bound B is a tuple $\langle \langle H, r^h_s \rangle, \mathcal{P}^h_s, \mathcal{P}^h_{s,\text{ind}}, \mathcal{P}^h_{s,\text{goal}} \rangle$, where

- *$\langle H, r^h_s \rangle$ is a PB circuit with input variables $\mathcal{V} \cup \mathcal{K}$ not mentioning a primed variable.*
- ***state lemma:** $\mathcal{P}^h_s$ is a CPR proof for $\mathcal{C}_s \cup H \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K \models (r_s \wedge \mathit{cost}_{\geq \max\{0, B-h(s)\}}) \rightarrow r^h_s$, where $\mathcal{C}_s = \{r_s \Leftrightarrow \sum_{v \in s} v + \sum_{v \in \mathcal{V} \setminus s} \bar{v} \geq |\mathcal{V}|\}$.*
- ***goal lemma:** $\mathcal{P}^h_{s,\text{goal}}$ is a CPR proof for $\mathcal{C}_{\text{goal}} \cup H \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K \models (r_G \wedge r^h_s) \rightarrow \mathit{cost}_{\geq B}$.*
- ***inductivity lemma:** $\mathcal{P}^h_{s,\text{ind}}$ is a CPR proof for $\mathcal{C}_{\text{trans}} \cup H \cup H' \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K \models (r^h_s \wedge r_T) \rightarrow {r'}^h_s$, where H' is a copy of H where all PB variables are replaced with their primed version.*

[2] Throughout the paper, we omit proofs of technical lemmas that do not provide further insight. These proofs are included in the extended version of this paper (Dold et al. 2025).

As mentioned earlier, heuristic certificates can be seen as certificates for state s where some cost of at least $B - h(s)$ has already been spent to reach s. Thus it is no coincidence that these heuristic certificates also structurally resemble lower-bound certificates under cost bound B.

We will later showcase for pattern database heuristics and $h^{\max}$ how such heuristic certificates can be generated. But first we show how they contribute to the overall lower-bound certificate generated by A*.

Proof-Logging A*

The invariant from a proof-logging A* search will conceptually cover two aspects: (1) the invariant is true for all closed states with their corresponding cost from the initial state. For these states, the heuristic estimates cannot rule out that they could be traversed with the corresponding cost by an optimal plan. (2) the invariant is also true for any state-cost pair for which any invariant is true that was produced by the heuristic to actually rule states out. The first part considers the distances from the initial state to the states in the closed list, while the second part considers the distances from the states in the open list to a goal.

Proof-logging A* maintains a sequence A of reifications and a proof log L of derivations. During its initialization, the heuristic already writes some information to A and L.

Whenever A* removes a state s with g-value g from the open list, it adds a reification

$$r_{s,\mathrm{g}\geq g} \Leftrightarrow \sum_{v\in s} v + \sum_{v\in\mathcal{V}\setminus s} \overline{v} + \mathit{cost}_{\geq g} \geq |\mathcal{V}| + 1 \qquad (11)$$

to A, characterizing all pairs $\langle s, \tilde{g}\rangle$ with $\tilde{g} \geq g$. In addition, it adds $\langle s, g\rangle$ to the initially empty collection *Closed*. It also logs some derivations in L that we describe later. If a successor s' with g-value g' is not added to *Open* because A*'s duplicate detection is aware of an earlier encounter with $g'' \leq g'$, it uses Lemma 1 and logs

$$K'_{\geq g'} \rightarrow K'_{\geq g''}. \qquad (12)$$

Whenever the search uses the heuristic to evaluate a state, the heuristic can extend A with further reifications, adds derivations for the corresponding state lemma, inductivity lemma and goal lemma to L and returns the corresponding PB variable r^h_s to the search. In addition, we require it to log a proof for the state lemma in terms of the primed variables.

When the search terminates, it adds to A a reification

$$r_{\mathrm{A}^*} \Leftrightarrow \sum_{\langle s,g\rangle\in\mathit{Closed}} r_{s,\mathrm{g}\geq g} + \sum_{\langle s,g,h\rangle\in\mathit{Open}} r^h_s \geq 1 \qquad (13)$$

and prepends A with the necessary reifications (9) and (10), where B is the cost of the found plan.

In the following we explain how we can generate the three proofs $\mathcal{P}_{\text{init}}, \mathcal{P}_{\text{ind}}$ and $\mathcal{P}_{\text{goal}}$ for the PB circuit $\langle A, r_{\mathrm{A}^*}\rangle$. This generation relies on the correctness of A*. However, the verifier receiving the generated proofs relies only on the correctness of the derivation rules of CPR. We start by showing how L can be extended to $\mathcal{P}_{\text{init}}$.

Lemma 3. *RUP can derive the initial state lemma* $(r_I \wedge \overline{\mathit{cost}_{\geq 1}}) \rightarrow r_{\mathrm{A}^*}$ *from* $\mathcal{C}_{\text{init}} \cup A \cup \mathcal{C}_{\geq}$.

Proof. Assume (a) $r_I \geq 1$, (b) $\overline{\mathit{cost}_{\geq 1}} \geq 1$, and (c) $\overline{r_{\mathrm{A}^*}} \geq 1$. From (a) and (1), we receive (d) $v \geq 1$ for each $v \in I$ and $\overline{v} \geq 1$ for each $v \in \mathcal{V} \setminus I$. Reification (4) and (b) express $\sum_{i=0}^{\lceil \log_2 B\rceil} 2^i c_i < 1$, from them we get for all $c_i \in \mathcal{V}_c$ that $\overline{c_i} \geq 1$. With (4), we receive $\mathit{cost}_{\geq 0}$, which with (d) and (11) gives $r_{I,\mathrm{g}\geq 0} \geq 0$. Since $\langle I, 0\rangle$ is in *Closed*, with (a) and (13) we get $r_{\mathrm{A}^*} \geq 1$, contradicting (c). □

The proof $\mathcal{P}_{\text{goal}}$ builds on the goal lemmas logged by the heuristic:

Lemma 4. *It is possible to derive the goal lemma* $(r_G \wedge r_{\mathrm{A}^*}) \rightarrow \mathit{cost}_{\geq B}$ *from* $\mathcal{C}_{\text{goal}} \cup A \cup \mathcal{C}_{\geq}$.

Proof. The heuristic certificate can derive the goal lemma for all open states. The rest is then by RUP, assuming (a) $r_G \geq 1$, (b) $r_{\mathrm{A}^*} \geq 1$ and (c) $\overline{\mathit{cost}_{\geq B}} \geq 1$.

From (2) and (a) we get that (d) $v \geq 1$ for all goal variables. For all closed pairs $\langle s, g\rangle$, s is not a goal state or it is the reached goal state $s_\star$ with $g = B$. If s is not a goal state, then some $v \in G$ is false in s and (11) yields with (d) that $\overline{r_{s,\mathrm{g}\geq g}} \geq 1$. We also get $\overline{r_{s_\star,\mathrm{g}\geq B}} \geq 1$ from (11), using (c).

For all variables r^h_s, the goal lemma provided by the heuristic implies with (a) and (c) that $\overline{r^h_s} \geq 1$, so overall we get with (13) that $\overline{r_{\mathrm{A}^*}} \geq 1$, contradicting (b). □

To support the derivation of the inductivity lemma, A* already extends L upon every expansion for every action with the derivation described in the following lemma:

Lemma 5. *For every action a applicable in s and state s closed with cost g, it is possible to derive* $(r_{s,\mathrm{g}\geq g} \wedge r_a) \rightarrow r'_{\mathrm{A}^*}$ *from* $\mathcal{C}_{\text{trans}} \cup A \cup A' \cup \mathcal{C}_{\geq}$.

Proof. We first establish by Lemma 2 that (a) $(\mathit{cost}_{\geq g} \wedge \Delta c^{=\mathit{cost}(a)}) \rightarrow \mathit{cost}'_{\geq \min\{B, g+\mathit{cost}(a)\}}$.

The desired constraint follows by RUP: Assume (b) $r_{s,\mathrm{g}\geq g} \geq 1$, (c) $r_a \geq 1$, and (d) $\overline{r'_{\mathrm{A}^*}} \geq 1$.

From (b), we can derive with (11) that (e) $v \geq 1$ for all $v \in s$, (f) $\overline{v} \geq 1$ for all $v \in \mathcal{V} \setminus s$, and (g) $\mathit{cost}_{\geq g} \geq 1$. We use (c) with (7) to derive (h) $\Delta c^{=\mathit{cost}(a)} \geq 1$. With (h), (g), and (a) we get (i) $\mathit{cost}'_{\geq \min\{B, g+\mathit{cost}(a)\}} \geq 1$. If s is a goal state then $g = B$ and we get from (i) and (7) that $\overline{r_a} \geq 1$, contradicting (c).

Otherwise, we define $\tilde{s} = s[\![a]\!]$ and use (7), (c), (e), (f), (g), and (6) to derive that (j) the primed state variables encode $\tilde{s}$ and that (k) $K'_{\geq g+\mathit{cost}(a)} \geq 1$.

If the successor was not considered because of a duplicate $\langle \tilde{s}, \hat{g}\rangle$ with $\hat{g} < g + \mathit{cost}(a)$, we use the derived constraint (12) to derive (k') $K'_{\geq \hat{g}} \geq 1$.

If $\langle \tilde{s}, g + \mathit{cost}(a)\rangle \in \mathit{Closed}$ or $\langle \tilde{s}, \hat{g}\rangle \in \mathit{Closed}$, respectively, we use the primed version of (11) with (j) and (k) or (k') to derive $r_{\tilde{s},\mathrm{g}\geq g+\mathit{cost}(a)}{}' \geq 1$ or $r_{\tilde{s},\mathrm{g}\geq \hat{g}}{}' \geq 1$. With the primed version of (13) we derive $r'_{\mathrm{A}^*} \geq 1$, contradicting (d).

Otherwise, there is an entry for $\tilde{s}$ in *Open*. Since A* closes all states with $f < B$, we know that in this case $g+\mathit{cost}(a)+h(\tilde{s}) \geq B$, so $g+\mathit{cost}(a) \geq \max\{0, B-h(\tilde{s})\}$. We can thus use (j) and (k) with the primed version of the state lemma for $\tilde{s}$ from the heuristic and receive $r^{h'}_{\tilde{s}} \geq 1$. The primed version of (13) yields $r'_{\mathrm{A}^*} \geq 1$, contradicting (d). □

Lemma 6. *For every state s closed with cost g, it is possible to derive $(r_{s,\mathrm{g}\geq g} \wedge r_T) \rightarrow r'_{\mathrm{A}^*}$ from $\mathcal{C}_{\text{trans}} \cup A \cup A' \cup \mathcal{C}_{\geq}$.*

Proof. We establish by Lemma 5 that (a) $(r_{s,\mathrm{g}\geq g} \wedge r_a) \rightarrow r'_{\mathrm{A}^*}$ for every action $a \in \mathcal{A}$ that is applicable in s.

Then RUP derives the constraint: Assume (b) $r_{s,\mathrm{g}\geq g} \geq 1$, (c) $r_T \geq 1$, and (d) $\overline{r'_{\mathrm{A}^*}} \geq 1$. With (a), (b) and (d) we derive $\overline{r_a} \geq 1$ for each action a applicable in s. From (b), we can derive with (11) that (e) $v \geq 1$ for all $v \in s$, (f) $\overline{v} \geq 1$ for all $v \in \mathcal{V} \setminus s$, and if a is not applicable in s, some precondition is violated and we use (f) and (7) to derive $\overline{r_a} \geq 1$. With (8), this yields $\overline{r_T} \geq 1$, contradicting (c). □

These derivations are already logged during the execution of the search. We extend the log with derivations from the following lemma to generate $\mathcal{P}_{\text{ind}}$:

Lemma 7. *It is possible to derive the inductivity lemma $(r_{\mathrm{A}^*} \wedge r_T) \rightarrow r'_{\mathrm{A}^*}$ from $\mathcal{C}_{\text{trans}} \cup A \cup A' \cup \mathcal{C}_{\geq}$.*

Proof. We first derive by RUP for every s with some $\langle s, g, h\rangle \in$ *Open* that (a) $(r^h_s \wedge r_T) \rightarrow r'_{\mathrm{A}^*}$: From the assumption $\overline{r'_{\mathrm{A}^*}} \geq 1$, we get $\overline{r'^h_s} \geq 1$. Using the inductivity lemma from the heuristic for s with the other assumptions $r^h_s \geq 1$ and $r_T \geq 1$, we get the contradiction.

For every state s expanded with cost g, we get (b) $(r_{s,\mathrm{g}\geq g} \wedge r_T) \rightarrow r'_{\mathrm{A}^*}$ as described in Lemma 6.

Afterwards RUP can derive the constraint, assuming (c) $r_{\mathrm{A}^*} \geq 1$, (d) $r_T \geq 1$, and (e) $\overline{r'_{\mathrm{A}^*}} \geq 1$.

For every open state s, we get from (a,d,e) that (f) $\overline{r^h_s} \geq 1$. For every state s closed with g-value g, we get from (b, d, e) that (g) $\overline{r_{s,\mathrm{g}\geq g}} \geq 1$. With (13), we get from (f) and (g) that $\overline{r_{\mathrm{A}^*}} \geq 1$, contradicting (c). □

To analyze the overhead of proof-logging A*, we assume that each proof from the heuristic is provided with only a constant-factor overhead to the heuristic computation time. Logging a single reification constraint (11) is an operation that requires time linear in $|\mathcal{V}|$. A constraint of this kind has to be logged for each closed state. This is still a constant-factor overhead because the closed state has to be generated first, which is an operation with time linear in $|\mathcal{V}|$, too. The reification of (13) is linear in the number of states generated. Both the goal lemma and the initial state lemma are logged by single RUP statements of constant size.

In the proof of the inductivity lemma we first use as many constant-size statements as there are open states. The second RUP subproof in the inductivity lemma requires the constraint from Lemma 6 specified on each state $s \in$ *Closed*, which in turn requires the constraint from Lemma 5 specified on each applicable action in s. This amortizes with the generation of the successors of s when s moves from *Open* to *Closed*. We see that in total there is a constant-factor overhead to the proof-logging of A*.

Proof-Logging Pattern Database Heuristics

Abstraction heuristics such as *pattern database (PDB) heuristics* use goal distances in an induced abstract task for the heuristic estimates. Let $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G\rangle$ be a STRIPS planning task. A PDB heuristic is defined in terms of a *pattern* $P \subseteq \mathcal{V}$ and its abstraction function α maps each state s over $\mathcal{V}$ to an abstract state $\alpha(s)$ over P as $\alpha(s) = s \cap P$. Each action $a \in \mathcal{A}$ induces the abstract action a^α with $\mathit{pre}(a^\alpha) = \mathit{pre}(a) \cap P$, $\mathit{add}(a^\alpha) = \mathit{add}(a) \cap P$, $\mathit{del}(a^\alpha) = \mathit{del}(a) \cap P$, and $\mathit{cost}(a^\alpha) = \mathit{cost}(a)$. The abstract goal G^α is $G \cap P$. The heuristic estimate of the PDB for state s is the cost of an optimal solution of the abstract task $\langle P, \{a^\alpha \mid a \in \mathcal{A}\}, \alpha(s), G^\alpha\rangle$ or ∞ if it is unsolvable. This is a lower bound of the goal distance of s in Π, as any solution of the concrete task corresponds to a solution of the abstract task with equal cost. In practice, a PDB heuristic does not build an abstract task for each heuristic evaluation but precomputes the abstract goal distances $d(s_\alpha)$ for all abstract states s_α and stores them in a so-called pattern database. When a concrete state s is evaluated, the PDB heuristic computes $\alpha(s)$ and returns the stored goal distance $d(\alpha(s))$ as the heuristic estimate $h(s)$.

The invariant for a PDB heuristic should hold for all pairs $\langle s, c\rangle$ such that the abstract goal distance d of $\alpha(s)$ is already so high that $c + d \geq B$, and thus it is impossible to reach the goal from s with a strictly lower cost than B if reaching s already costs c.

Let S_α be the set of all abstract states. For each $s_\alpha \in S_\alpha$, the heuristic introduces two PB variables. The variable r^{s_α} is true iff the variables from $\mathcal{V}$ encode a state s with $\alpha(s) = s_\alpha$ by adding a reification

$$r^{s_\alpha} \Leftrightarrow \sum\nolimits_{v \in s_\alpha} v + \sum\nolimits_{v \in P \setminus s_\alpha} \bar{v} \geq |P|. \tag{14}$$

The variable $r^{s_\alpha}_{\geq B - d(s_\alpha)}$ is true iff the variables from $\mathcal{V}$ and $\mathcal{V}_c$ encode a pair $\langle s, c\rangle$ such that $\alpha(s) = s_\alpha$ and $c \geq \max\{B - d(s_\alpha), 0\}$ by adding reification

$$r^{s_\alpha}_{\geq B - d(s_\alpha)} \Leftrightarrow r^{s_\alpha} + K_{\geq B - d(s_\alpha)} \geq 2. \tag{15}$$

As a final reification for the invariant the heuristic adds

$$r_{\text{PDB}} \Leftrightarrow \sum\nolimits_{s_\alpha \in S_\alpha} r^{s_\alpha}_{\geq B - d(s_\alpha)} \geq 1. \tag{16}$$

The PB circuit for any state then is $\langle H_{\text{PDB}}, r_{\text{PDB}}\rangle$, where H_{PDB} is the sequence of all reifications (14), (15) and (16). On the evaluation of a state s, the heuristic always returns reification variable r_{PDB} to the search. In the following we discuss how the required proofs can be generated. This generation relies on the correctness and admissibility of the PDB heuristic, but the generated proofs themselves do not.

We begin with the state lemma, which requires a new proof for each evaluated state.

Lemma 8. *RUP can derive the state lemma $(r_s \wedge \mathit{cost}_{\geq \max\{0, B - h(s)\}}) \rightarrow r_{\text{PDB}}$ from $\mathcal{C}_s \cup H_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$, where $\mathcal{C}_s = \{r_s \Leftrightarrow \sum_{v \in s} v + \sum_{v \in \mathcal{V} \setminus s} \bar{v} \geq |\mathcal{V}|\}$.*

Since the PB circuit for the heuristic is the same for all evaluated states, we only need to include the proof for the goal lemma and for the inductivity lemma once in the overall generated proof. For the goal lemma, we use the following:

Lemma 9. *RUP can derive the goal lemma $(r_G \wedge r_{\text{PDB}}) \rightarrow \mathit{cost}_{\geq B}$ from $\mathcal{C}_{\text{goal}} \cup H_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.*

For the inductivity lemma, we develop the derivation by means of four lemmas. The first one derives that applying an induced abstract action a^α in abstract state s_α leads to the abstract successor state.

Lemma 10. *For each action $a \in \mathcal{A}$ and abstract state s_α such that a^α is applicable in s_α, RUP can derive $(r^{s_\alpha} \wedge r_a) \to r^{s_\alpha[\![a^\alpha]\!]\prime}$ from $\mathcal{C}_{\text{trans}} \cup H_{\text{PDB}} \cup H'_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.*

The next lemma again considers individual actions and abstract states, but takes cost into account and derives that the invariant of the certificate is true for the successor state-cost pair.

Lemma 11. *For each action $a \in \mathcal{A}$ and abstract state s_α such that a^α is applicable in s_α, it is possible to derive $(r^{s_\alpha}_{\geq B-d(s_\alpha)} \wedge r_a) \to r'_{\text{PDB}}$ from $\mathcal{C}_{\text{trans}} \cup H_{\text{PDB}} \cup H'_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.*

Proof. We start by deriving some constraints over costs that we will use later in a RUP proof.

First, we derive (a) $(K_{\geq B-d(s_\alpha)} \wedge \Delta c^{=cost(a)}) \to K'_{\geq B-d(s_\alpha)+cost(a)}$ as described in Lemma 2.

We know that for the abstract goal distance it holds that $d(s_\alpha[\![a^\alpha]\!]) + cost(a^\alpha) \geq d(s_\alpha)$. Together with $cost(a^\alpha) = cost(a)$, we get $B - d(s_\alpha[\![a^\alpha]\!]) \leq B - d(s_\alpha) + cost(a)$. We use this to derive (b) $K'_{\geq B-d(s_\alpha)+cost(a)} \to K'_{\geq B-d(s_\alpha[\![a^\alpha]\!])}$ as described in Lemma 1. In addition, we establish (c) $(r^{s_\alpha} \wedge r_a) \to r^{s_\alpha[\![a^\alpha]\!]\prime}$ by RUP (Lemma 10).

Now we can derive the constraint in the claim by RUP, assuming that (d) $r^{s_\alpha}_{\geq B-d(s_\alpha)} \geq 1$, (e) $r_a \geq 1$, and (f) $\overline{r'_{\text{PDB}}} \geq 1$. From (d), we get with (15) that (g) $r^{s_\alpha} \geq 1$ and (h) $K_{\geq B-d(s_\alpha)} \geq 1$. From (e),(g) and (c) we derive (i) $r^{s_\alpha[\![a^\alpha]\!]\prime} \geq 1$.

From (e) and (7), we derive (j) $\Delta c^{=cost(a)} \geq 1$. From (a) together with (h) and (j) we derive that (k) $K'_{\geq B-d(s_\alpha)+cost(a)} \geq 1$. From (b) together with (k) we derive that (l) $K'_{\geq B-d(s_\alpha[\![a^\alpha]\!])} \geq 1$. We get with this, (i) and the primed constraint (15) from H_{PDB} that $r^{s_\alpha[\![a^\alpha]\!]\prime}_{\geq B-d(s_\alpha[\![a^\alpha]\!])}$. With the primed version of (16) from H'_{PDB} this yields $r'_{\text{PDB}} \geq 1$, contradicting (f). □

The third lemma generalizes the previous lemma from individual actions to the entire transition relation.

Lemma 12. *For each abstract state s_α it is possible to derive $(r^{s_\alpha}_{\geq B-d(s_\alpha)} \wedge r_T) \to r'_{\text{PDB}}$ from $\mathcal{C}_{\text{trans}} \cup H_{\text{PDB}} \cup H'_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.*

Proof sketch. For each action $a \in \mathcal{A}$ that is applicable in s_α, establish $(r^{s_\alpha}_{\geq B-d(s_\alpha)} \wedge r_a) \to r'_{\text{PDB}}$ with Lemma 11. Then the desired constraint can be derived by RUP. □

The final step for the inductivity lemma generalizes this from individual abstract states to the entire PB circuit of the heuristic.

Lemma 13. *It is possible to derive the inductivity lemma $(r_{\text{PDB}} \wedge r_T) \to r'_{\text{PDB}}$ from $\mathcal{C}_{\text{trans}} \cup H_{\text{PDB}} \cup H'_{\text{PDB}} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.*

Proof sketch. Establish $(r^{s_\alpha}_{\geq B-d(s_\alpha)} \wedge r_T) \to r'_{\text{PDB}}$ for each abstract state s_α by Lemma 12. The rest follows by RUP. □

We considered PDB heuristics as one example of abstractions because they have a particularly simply structured abstraction function. To adapt the approach to other abstraction heuristics, the overall line of argument still works but we would have to replace (14) with one or more reifications that allow to identify the concrete states that correspond to a given abstract state. In addition, the derivations from Lemmas 8, 9 and 10 need to be adapted accordingly.

The described proof logging only has a constant-factor overhead for a somewhat naive implementation of PDBs. While the state and goal lemma only require a single RUP statement, the inductivity lemma requires effort in $O(|S_\alpha||\mathcal{A}|)$, iterating once over all actions for each abstract state. A better PDB implementation can deal more efficiently with abstract states that are not reverse-reachable. In the extended version of this paper we describe a variant of the certificate that does not explicitly represent such states and which guarantees a constant-factor overhead also for such a more efficient computation of PDBs.

Proof-Logging $h^{\max}$

The maximum heuristic (Bonet and Geffner 2001) is based on a relaxation of the planning task that makes two simplifying assumptions: first, actions do not delete variables; second, the cost of making a set of variables true corresponds to the cost of making its most expensive variable true; since action preconditions are sets of variables, the relaxation affects the cost of enabling action applications, which in turn affects the cost to make the individual variables true. For this reason, the heuristic is mathematically specified by a system of equations. The maximum heuristic $h^{\max}(s)$ is $h^{\max}(s, G)$, where $h^{\max}(s, V)$ is the pointwise greatest function with $h^{\max}(s, V) =$

$$\begin{cases} 0 & \text{if } V \subseteq s \\ \min\limits_{\substack{a \in \mathcal{A}, \\ v \in add(a)}} (cost(a) + h^{\max}(s, pre(a))) & \text{if } |V| = 1 \text{ and } V \not\subseteq s \\ \max\limits_{v \in V} h^{\max}(s, \{v\}) & \text{otherwise} \end{cases}$$

For $v \in \mathcal{V}$, the *max value* is $V^{\max}(v) = h^{\max}(s, \{v\})$, which can be seen as the cost of making v true (starting from s) under this relaxation. The heuristic certificate will build on the following insight: since $h^{\max}$ is admissible, it will be impossible to reach the goal from s with overall cost $< B$ if the cost to reach s already exceeds $B - h^{\max}(s)$. If in a state $\hat{s}$ additional variables are true, the state can be closer to the goal but under the relaxation of the heuristic the advantage gained from an individual variable v will be at most $V^{\max}(v)$, so for arbitrary non-empty states $\hat{s}$, the goal cannot be reached if the cost to reach $\hat{s}$ already exceeds $B - h^{\max}(s) + \max_{v \in \hat{s}} V^{\max}(v)$ (note that $V^{\max}(v) = 0$ for $v \in s$), even under the relaxation of the heuristic. In the delete-relaxed task, the empty state cannot be closer to the goal than s, so we can use bound $B - h^{\max}(s)$.

A typical efficient implementation of the maximum heuristic only determines the max value for all v with

$V^{\max}(v) < h^{\max}(s)$. We can still build a valid heuristic certificate, requiring that the cost B has already been exceeded for states that contain a variable for which $V^{\max}(v) \geq h^{\max}(s)$. For the formal definition of the certificate, we will handle this aspect by an auxiliary value $W^{\max}(v) = \min\{V^{\max}(v), h^{\max}(s)\}$.

For each $v \in \mathcal{V}$, the heuristic adds a reification:

$$r_{v,s} \Leftrightarrow \bar{v} + K_{\geq B - h^{\max}(s) + W^{\max}(v)} \geq 1 \tag{17}$$

In addition, it adds the following reification:

$$r_s^{\max} \Leftrightarrow K_{\geq B - h^{\max}(s)} + \sum_{v \in \mathcal{V}} r_{v,s} \geq |\mathcal{V}| + 1 \tag{18}$$

The PB circuit for state s then is $\langle H_{\max,s}, r_s^{\max} \rangle$, where $H_{\max,s}$ is the sequence of all reifications (17) and (18).

Intuitively, (17) expresses the implication that the cost is at least $B - h^{\max}(s) + W^{\max}(v)$ if v is true. Equation (18) corresponds to a conjunction of $K_{\geq B - h^{\max}(s)}$ and such implications for each variable, so overall $r_s^{\max}$ will be true for all state value pairs $\langle \hat{s}, \hat{c} \rangle$, where $\hat{s} \neq \emptyset$ and $\hat{c} \geq \max_{v \in \hat{s}} \{B - h^{\max}(s) + W^{\max}(v)\}$ or where $\hat{s} = \emptyset$ and $\hat{c} \geq B - h^{\max}(s)$.

Lemma 14. *RUP can derive the state lemma* $(r_s \wedge \mathit{cost}_{\geq \max\{0, B - h^{\max}(s)\}}) \rightarrow r_s^{\max}$ *from* $\mathcal{C}_s \cup H_{\max,s} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$, *where* $\mathcal{C}_s = \{r_s \Leftrightarrow \sum_{v \in s} v + \sum_{v \in \mathcal{V} \setminus s} \bar{v} \geq |\mathcal{V}|\}$.

A single RUP statement is sufficient for the goal lemma:

Lemma 15. *RUP can derive the goal lemma* $(r_G \wedge r_s^{\max}) \rightarrow \mathit{cost}_{\geq B}$ *from* $\mathcal{C}_{\text{goal}} \cup H_{\max,s} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.

For the inductivity lemma, we first show an analogous statement for a single action.

Lemma 16. *For every action a, it is possible to derive* $(r_s^{\max} \wedge r_a) \rightarrow r_s^{\max\prime}$ *from* $\mathcal{C}_{\text{trans}} \cup H_{\max,s} \cup H'_{\max,s} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.

Proof sketch. The overall argument of the proof is as follows:

If the action has a precondition p, we can conclude that $p \geq 1$, which allows us to derive with (17) that $K_{\geq B - h^{\max}(s) + W^{\max}(p)} \geq 1$. We can establish $K'_{\geq B - h^{\max}(s)}$ using Lemmas 1 and 2.

To establish $r'_{v,s} \geq 1$ for all variables, we consider the precondition with maximal $W^{\max}$ value.

If $W^{\max}(p) = h^{\max}(s)$, then $K_{\geq B - h^{\max}(s) + W^{\max}(p)} \geq 1$ corresponds to $K_{\geq B} \geq 1$. We derive $K'_{\geq B} \geq 1$, which is sufficient to establish $r'_{v,s} \geq 1$.

If $W^{\max}(p) < h^{\max}(s)$, we establish $r'_{v,s} \geq 1$ depending on whether and how v occurs in the effect of a. If it is a delete effect, then $\overline{v'} \geq 1$. If it is an add effect, we exploit that p has maximal $W^{\max}$ value among the preconditions, so from the definition of the maximum heuristic we can see that $W^{\max}(v) \leq W^{\max}(p) + \mathit{cost}(a)$. Establishing $K'_{\geq B - h^{\max}(s) + W^{\max}(p) + \mathit{cost}(a)}$ and using this insight gives $K'_{\geq B - h^{\max}(s) + W^{\max}(v)} \geq 1$. If $v \notin \mathit{evars}(a)$, we can carry over the reason for $r_{v,s} \geq 1$ to the primed variables.

If the precondition of the action is empty, we build on $K_{\geq B - h^{\max}(s)}$, which gives $K'_{\geq B - h^{\max}(s) + \mathit{cost}(a)}$ and $K'_{\geq B - h^{\max}(s)}$ using Lemmas 1 and 2. For establishing $r'_{v,s} \geq 1$ for all variables, we proceed as before for the case $W^{\max}(p) < h^{\max}(s)$ (replacing $W^{\max}(p)$ with 0). For the special case $h^{\max}(s) = 0$, we can instead use the shorter argument as in the previous case $W^{\max}(p) = h^{\max}(s)$, building on $K_{\geq B} \geq 1$.

The technical details are included in the extended version of this paper (Dold et al. 2025). □

The full inductivity lemma follows quite directly.

Lemma 17. *It is possible to derive the inductivity lemma* $(r_s^{\max} \wedge r_T) \rightarrow r_s^{\max\prime}$ *from* $\mathcal{C}_{\text{trans}} \cup H_{\max,s} \cup H'_{\max,s} \cup \mathcal{C}_{\geq} \cup \mathcal{C}_K$.

Proof sketch. Establish with Lemma 16 for every $a \in \mathcal{A}$ that $(r_s^{\max} \wedge r_a) \rightarrow r_s^{\max\prime}$. Then derive the constraint by RUP. □

Extending $h^{\max}$ with proof logging only leads to the following overhead: adding reifications (17) for each $v \in \mathcal{V}$ and (18) takes additional time linear in $|\mathcal{V}|$ because $h(s)$ and the values necessary for $W^{\max}$ are already computed during the normal evaluation. In addition, the implementation has to reinitialize some values for each variable before each new evaluation, so the overhead is within a constant factor.

The state and goal lemmas each only require a single RUP statement. For the inductivity lemma, we need to establish the constraint from Lemma 16 for every action. For each action where $h^{\max}(s, \mathit{pre}(a)) \geq h^{\max}(s)$, we require constant effort. For all other actions, the effort for the action is linear in the number of variables.

Conclusion

We introduced lower-bound certificates for classical planning, which can be used to verify the optimality of optimal planners. The certificates are sound, complete and efficiently verifiable. We showed them to be efficiently generatable in a case study on A* with $h^{\max}$ or pattern database heuristics.

We believe these certificates to be much more general than the ones considered in previous research because they work on a more fundamental level of abstraction, build on an established proof systems that has proved its utility in many other areas of AI, and are able to directly incorporate numerical reasoning. In future work, this generality has to be confirmed by applying these certificates to other optimal planning approaches that current approaches cannot handle. In particular, we believe that pseudo-Boolean constraints can generically handle the concept of *cost partitioning* (Katz and Domshlak 2010), perhaps *the* most important optimal planning technique, because they are able to reason numerically at the level of individual state transitions in a way that previously proposed certificates cannot.

Acknowledgments

This research was supported by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored StateSpace Search" (UTA). Jakob Nordström was supported by the Independent Research Fund Denmark grant 9040-00389B.

References

Berg, J.; Bogaerts, B.; Nordström, J.; Oertel, A.; and Vandesande, D. 2023. Certified Core-Guided MaxSAT Solving. In Pientka, B.; and Tinelli, C., eds., *Proceedings of the 29th International Conference on Automated Deduction (CADE 2023)*, volume 14132 of *LNCS*, 1–22. Springer.

Bogaerts, B.; Gocht, S.; McCreesh, C.; and Nordström, J. 2022. Certified Symmetry and Dominance Breaking for Combinatorial Optimisation. In *Proc. AAAI 2022*, 3698–3707.

Bogaerts, B.; Gocht, S.; McCreesh, C.; and Nordström, J. 2023. Certified Dominance and Symmetry Breaking for Combinatorial Optimisation. *JAIR*, 77: 1539–1589.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *AIJ*, 129(1): 5–33.

Buss, S.; and Nordström, J. 2021. Proof Complexity and SAT Solving. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 233–350. IOS Press.

Cook, W. J.; Coullard, C. R.; and Turán, G. 1987. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1): 25–38.

Cruz-Filipe, L.; Heule, M. J. H.; Hunt, W. A., Jr.; Kaufmann, M.; and Schneider-Kamp, P. 2017. Efficient Certified RAT Verification. In de Moura, L., ed., *Proceedings of the 26th International Conference on Automated Deduction (CADE 2017)*, volume 10395 of *LNCS*, 220–236. Springer.

Demirović, E.; McCreesh, C.; McIlree, M. J.; Nordström, J.; Oertel, A.; and Sidorov, K. 2024. Pseudo-Boolean Reasoning About States and Transitions to Certify Dynamic Programming and Decision Diagram Algorithms. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP-24)*, 9:1–9:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Dold, S.; Helmert, M.; Nordström, J.; Röger, G.; and Schindler, T. 2025. Pseudo-Boolean Proof Logging for Optimal Classical Planning. arXiv:2504.18443 [cs.AI].

Edelkamp, S. 2001. Planning with Pattern Databases. In *Proc. ECP 2001*, 84–90.

Edelkamp, S.; and Kissmann, P. 2009. Optimal Symbolic Planning with Action Costs and Preferences. In *Proc. IJCAI 2009*, 1690–1695.

Eriksson, S.; and Helmert, M. 2020. Certified Unsolvability for SAT Planning with Property Directed Reachability. In *Proc. ICAPS 2020*, 90–100.

Eriksson, S.; Röger, G.; and Helmert, M. 2017. Unsolvability Certificates for Classical Planning. In *Proc. ICAPS 2017*, 88–97.

Eriksson, S.; Röger, G.; and Helmert, M. 2018. A Proof System for Unsolvable Planning Tasks. In *Proc. ICAPS 2018*, 65–73.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *AIJ*, 2: 189–208.

Gocht, S.; McCreesh, C.; and Nordström, J. 2022. An Auditable Constraint Programming Solver. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming (CP-22)*, 25:1–25:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Haslum, P. 2016. INVAL: the Independent PDDL Plan Validator. https://github.com/patrikhaslum/INVAL. Accessed May 2, 2025.

Heule, M. J. H.; Hunt, W. A.; and Wetzler, N. 2013. Verifying Refutations with Extended Resolution. In *Proceedings of the Twenty-Fourth International Conference on Automated Deduction (CADE-24)*, 345–359. Springer, Berlin, Heidelberg.

Hoen, A.; Oertel, A.; Gleixner, A.; and Nordström, J. 2024. Certifying MIP-based Presolve Reductions for 0–1 Integer Linear Programs. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR-24)*, 310–328. Springer.

Howey, R.; and Long, D. 2003. VAL's Progress: The Automatic Validation Tool for PDDL2.1 used in the International Planning Competition. In *ICAPS 2003 Workshop on the Competition*.

Katz, M.; and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *AIJ*, 174(12–13): 767–798.

Lammich, P. 2020. Efficient Verified (UN)SAT Certificate Checking. *Journal of Automated Reasoning*, 64(3): 513–532.

McConnell, R. M.; Mehlhorn, K.; Näher, S.; and Schweitzer, P. 2011. Certifying algorithms. *Computer Science Review*, 5(2): 119–162.

McIlree, M. J.; and McCreesh, C. 2023. Proof Logging for Smart Extensional Constraints. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming, (CP-23)*, 26:1–26:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

McIlree, M. J.; McCreesh, C.; and Nordström, J. 2024. Proof Logging for the Circuit Constraint. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR-24)*, 38–55. Springer.

Mugdan, E.; Christen, R.; and Eriksson, S. 2023. Optimality Certificates for Classical Planning. In *Proc. ICAPS 2023*, 286–294.

Rintanen, J.; Heljanko, K.; and Niemelä, I. 2006. Planning as satisfiability: parallel plans and algorithms for plan search. *AIJ*, 170(12–13): 1031–1080.

Tan, Y. K.; Heule, M. J. H.; and Myreen, M. O. 2023. Verified Propagation Redundancy and Compositional UNSAT Checking in CakeML. *Software Tools for Technology Transfer*, 25(2): 167–184.