Heuristic Search Planning With Multi-Objective Probabilistic LTL Constraints

Peter Baumgartner, Sylvie Thiébaux, and Felipe Trevizan

Data61/CSIRO and Research School of Computer Science, ANU, Australia

Email: first.last@anu.edu.au

October 18, 2018

Abstract

We present an algorithm for computing cost-optimal stochastic policies for Stochastic Shortest Path problems (SSPs) subject to multi-objective PLTL constraints, i.e., conjunctions of probabilistic LTL formulas. Established algorithms capable of solving this problem typically stem from the area of probabilistic verification, and struggle with the large state spaces and constraint types found in automated planning. Our approach differs in two crucial ways. Firstly it operates entirely on-the-fly, bypassing the expensive construction of Rabin automata for the formulas and their prohibitive prior synchronisation with the full state space of the SSP. Secondly, it extends recent heuristic search algorithms and admissible heuristics for cost-constrained SSPs, to enable pruning regions made infeasible by the PLTL constraints. We prove our algorithm correct and optimal, and demonstrate encouraging scalability results.

1 Introduction

The problem of computing optimal but safe policies for autonomous agents operating in uncertain environments has recently attracted significant attention from the fields of automated verification [KP13], robotics [DSBR14], and artificial intelligence [SKT14]. Such policies must minimise the agent's expected cost to reach a goal, whilst providing probabilistic guarantees about the sequence of visited states. For instance, a policy for a search and rescue UAV mission might need to minimise the expected time to get survivors to safety, whilst avoiding dangerous areas at all times, and circling the affected locations to correctly determine the presence of survivors with high probability.

Such planning problems can be modelled as stochastic shortest path problems (SSPs) augmented with multi-objective probabilistic LTL (MO-PLTL) constraints. Such constraints are conjunctions $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ of *objectives*, where ψ_i is an LTL formula and $z_i \subseteq [0, 1]$ is an interval bounding its probability. An optimal solution takes the form of a finite-memory stochastic policy π whose execution: (a) satisfies the MO-PLTL constraints; (b) reaches a goal state with probability 1; and (c) has minimal expected cost, subject to (a,b).

Fulfilling requirement (a) alone, and to a lesser degree in combination with (c), has been studied in the area of policy synthesis for Markov Decision Processes (MDPs) [FKNP11, EKVY08], leading to what we call the static approach. This consists in constructing k automata whose accepting runs correspond exactly to the satisfying runs of the k LTL constraints, and then building the $k{+}1$-ary cross-product of these k automata with the state space of the decision process. As these automata have

to be deterministic for the cross product to remain an MDP, deterministic Rabin automata (DRA) are required, which are obtained from nondeterministic Büchi automata (NBA) for the formulas ψ_i. Finally, linear programming is applied to the cross-product, to generate (cost-optimal) policies that reach certain sets of states (bottom-end components) with given probability bounds of the form $> u_i$ or $\geq u_i$, where $u_i \in [0, 1]$. A policy for the original problem can be recovered from the solution of the LP.

There are two practical issues that make the static approach inapplicable to the probabilistic planning problems we are interested in. Firstly, the DRA compilation can be prohibitive for certain common type of planning objectives, e.g., multiple maintenance objectives. Secondly and more significantly, the explicit construction of the cross-product is completely infeasible for planning problems which usually have huge state spaces.

In this paper, we resolve these two issues in the particular case where the goal requirement (b) is present by leveraging and extending efficient heuristic search algorithms and heuristics from the field of constrained SSPs (C-SSPs) [TTSW16, TTH17]. Specifically, the recent *i-dual* algorithm solves C-SSPs *on-the-fly*, by running linear programming on small state space fragments of increasing size, guided by an admissible heuristic function to prune regions that are too costly or cannot satisfy the constraints. When guided by effective C-SSP heuristics, such as the *occupation measure* heuristics [TTH17], i-dual only expands a fraction of the state space. The *i^2-dual* algorithm additionally embeds the computation of occupation measure heuristics in the LP, making optimisation and heuristic estimation synergic and avoiding repeated calls to heuristic estimators. These algorithms and heuristics are however currently limited to much simpler constraint types and do not handle MO-PLTL constraints.

To harness the benefits of this "on-the-fly approach" for our problem, we must (1) bypass the need for computing the DRA and the cross product, and (2) extend the heuristics and algorithm to handle MO-PLTL constraints. We achieve (1) by embedding on-the-fly progression of LTL formulae [BK98] or determinisation of NBA in the state space expansion performed by i^2-dual. This avoids doing work that is not relevant to the traces expanded by the heuristic search and, in the case of progression, leads to improved worst-case complexity. We achieve (2) by computing occupation measure heuristics from the exponentially smaller NBAs for each of the formulas, and optionally embedding this computation into i^2-dual. Our experiments show that the on-the-fly approach is capable of solving planning problems that are out of reach of the state-of-the-art implementation of the static approach in the Prism probabilistic model-checker [KNP11].

Due to the lack of space, proofs of theorems are given in appendix.

2 Background and Problem Definition

Let $Dist(X)$ be the set of all distributions on a set X. Given a finite set S of states, we write S^+ for the set of (non-empty) finite sequences of states over S, and S^ω for the set of infinite state sequences, thus $S^+ \cap S^\omega = \emptyset$. For a state sequence $p \in S^+ \cup S^\omega$ and a natural number i, p_i denotes the state of index i in p, and $p(i)$ the suffix $p_i p_{i+1} \cdots$ of p. For a finite sequence $p \in S^+$, last(p) represents the last state of p. We write $p; p'$ for the concatenation of $p \in S^+$ and $p' \in S^+ \cup S^\omega$.

SSPs. A *Stochastic Shortest Path problem* (SSP) is a tuple $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ where: S is the finite set of states; $s_{\text{init}} \in S \setminus G$ is the initial state; $G \subset S$ is the non-empty set of goal states; A is a finite set of *actions* and $A(s) \subseteq A$ is the *set of actions enabled* in $s \in S$. We assume $A(s) \neq \emptyset$ for $s \in S \setminus G$ and $A(s_g) = \emptyset$ for $s_g \in G$; $P(\bullet|s, \alpha) \in Dist(S)$ is the probability of transitioning to $t \in S$ after applying $\alpha \in A(s)$ in state s; $C(\alpha) \in \mathbb{R}^*_+$ is the *cost* of α; and $T : G \to \mathbb{R}$ is the one-time *terminal cost* of reaching

a goal state.[1]

Policies. A solution to an SSP is a policy. In this paper, we consider various types of policies for SSPs and for more complex sequential decision problems to be defined later. These policy types differ in their ability to incorporate memory and randomisation. The most general of these are *stochastic history-dependent policies*, or *unrestricted policies*, which are (partial) functions $\pi : S^+ \mapsto Dist(A)$ mapping the finite state history p to a probability distribution over the actions enabled in the current state. We abbreviate $\pi(p)(\alpha)$ as $\pi(p, \alpha)$ and stipulate that $\pi(p, \alpha) = 0$ if $\alpha \notin A(\text{last}(p))$. A policy is *deterministic* if, for every p in its domain, there is $\alpha \in A(\text{last}(p))$ such that $\pi(p, \alpha) = 1$; hence these policies are (partial) functions $\pi : S^+ \mapsto A$ and we write $\pi(p)$ for the unique action prescribed by the policy in a given state history. A policy is *memoryless* (i.e., stationary) if $\pi(p)$ only depends on $\text{last}(p)$, and has *finite memory* if it additionally depends on a mode which has finitely many values and is updated along with the current state after each transition.

A *run* r is a path $p = s_1 s_2 \cdots \in S^+ \cup S^\omega$ annotated with the actions executed between states, that is, $r = s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} s_3 \cdots$ such that $\alpha_i \in A(s_i)$ and $P(s_{i+1}|s_i, \alpha_i) > 0$, for all $i \geq 1$. The *cost* and *probability* of a run r are defined as $C(r) = \sum_{i \geq 1} C(\alpha_i)$ and $P(r) = \prod_{i \geq 1} P(s_{i+1}|s_i, \alpha_i)$, respectively. Since a path p can be trivially obtained from a run r by removing the action annotation, we use r as a path (e.g., $\pi(r, a)$) when clear from context.

Given a policy π, a run r is an *exhaustive run of* π if $\pi(s_1 \cdots s_i, \alpha_i) > 0$ for all $i \geq 1$ and either r is infinite or π is not defined for the finite path represented by r. An exhaustive run of π only stops when π is unable to recommend an action to be executed next (in particular when a goal state is reached). We write $\text{Runs}(s, \pi)$ for the *set of all exhaustive runs of* π *starting from* $s \in S$, and $\text{GRuns}(s, \pi) \subseteq \text{Runs}(s, \pi)$ for those runs that additionally reach a goal state. For any $s \in S$ and $r \in \text{Runs}(s, \pi)$, the *probability of* r *being produced by* π is $P(r|\pi) = P(r) \prod_{i \geq 1} \pi(s_1 \cdots s_i, \alpha_i)$.

A policy π is *proper* if $\sum_{r \in \text{GRuns}(s_{\text{init}}, \pi)} P(r|\pi) = 1$, i.e., if the probability of reaching the goal when using π from s_{init} is 1. To simplify notation, we assume that there is at least one proper policy for $\mathcal{S}$, or equivalently, that there are no reachable dead ends from s_{init}. Dead ends can easily be handled as in [TTKT17].

Optimal Policies. Given a proper policy π, its *total expected cost* $V^\pi(s)$ to reach a goal state from a state $s \in S$ is:

$$V^\pi(s) = \sum_{r \in \text{GRuns}(s, \pi)} \Big[C(r) + T(\text{last}(r)) \Big] P(r|\pi).$$

We abbreviate $V^\pi(s_{\text{init}})$ with V^π. An *optimal policy* π^* is a proper policy such that $V^{\pi^*} \leq V^\pi$ for all proper policies π.

It is well-known that, in the case of SSPs, at least one optimal policy is memoryless and deterministic. For such policies, their total expected cost can be expressed as the following set of fixed-point equations that is at the core of most solution methods for SSPs:

$$V^\pi(s) = \begin{cases} T(s) & \text{if } s \in G \\ C(\pi(s)) + \sum_{t \in S} P(t|s, \pi(s)) V^\pi(t) & \text{otherwise} \end{cases}$$

C-SSPs. Many extensions of SSPs require stochastic policies. We consider in particular *cost-constrained SSPs* (C-SSPs), for which stochastic policies are needed to optimally account for trade-offs between

[1] An SSP with terminal cost can be trivially encoded as an SSP without terminal costs by adding extra actions. We use terminal costs to simplify our notation.

various cost functions representing, e.g., fuel, money, or time. A C-SSP $C = (S, s_{\text{init}}, G, A, P, \boldsymbol{C}, \boldsymbol{T}, \boldsymbol{z})$ is an SSP where: (i) the action cost function is replaced by a vector of $k+1$ action cost functions $\boldsymbol{C} = [C_0, \ldots, C_k]$ ($C_0 \colon A \to \mathbb{R}_+^*$ and $C_j \colon A \to \mathbb{R}_+$ for all $j \geq 1$); (ii) the terminal cost function is also replaced by a vector of $k+1$ terminal cost functions $\boldsymbol{T}$; and (iii) a vector of k intervals $\boldsymbol{z} = [z_1, \ldots, z_k]$ ($z_j \subseteq \mathbb{R}_+$ for all $j \geq 1$) is added. We refer to C_0 and T_0 as the *primary* action (resp. terminal) cost and to the other elements of the cost vectors as the *secondary* costs. The optimal solution for a C-SSP is any stochastic memoryless policy $\pi : S \mapsto \mathit{Dist}(A)$ which minimises the total expected primary cost to reach a goal state in G from the initial state s_{init} subject to the total expected j-th cost lying within interval z_j for $j \geq 1$. That is, there are $k+1$ total expected cost functions:

$$V_j^\pi(s) = \begin{cases} T_j(s) & \text{if } s \in G \\ \sum_{\alpha \in A(s)} \pi(s,\alpha) \left(C_j(\alpha) + \sum_{t \in S} P(t|s,\alpha) V_j^\pi(t) \right) & \text{otherwise} \end{cases} \tag{1}$$

and an optimal policy π^* minimizes $V_0^{\pi^*}$ subject to $V_j^{\pi^*} \in z_j$, for all $j \geq 1$.

Algorithms. The worst-case complexity of computing an optimal policy for SSPs or C-SSPs is polynomial in the size of the state space S [DD05]. For planning problems however, the state space is much too large to be explicitly enumerated. Therefore, the field represents SSPs using exponentially more compact factored representations, and has moved away from methods that completely expand the state space, including in particular from vanilla linear programming and dynamic programming methods such as value and policy iteration. Instead it focuses on Monte Carlo tree search [BG12, KE12], or on heuristic search approaches such as (L)RTDP [BBS95, BG03] and LAO* [HZ01] for SSPs, and i-dual for C-SSPs [TTSW16]. These start from the factored representation and expand the state space on-the-fly, guided by an *admissible heuristic* function derived from a polynomial time analysis of the factored representation [BG05, TKVI11, TTH17]. Such a heuristic provides us with a lower bound on the optimal expected cost $V^{\pi^*}(s)$ to reach the goal from the current state s, and enables large regions of the state space to be pruned. When equipped with an informative heuristic, heuristic search algorithms often expand only a small fraction of the state space.

Factored Representation. We adopt a probabilistic variant of the SAS$^+$ formalism as our factored representation [Bac92, TTH17]. A *probabilistic SAS$^+$ task* is a tuple $\langle \mathcal{V}, \mathsf{A}, s_\bullet, s_\star, C \rangle$. $\mathcal{V}$ is a finite set of *state variables*, and each $v \in \mathcal{V}$ has a finite domain D_v. A *valuation* (or partial state) is a function s on a subset $\mathcal{V}_s$ of $\mathcal{V}$, such that $s[v] \in \mathsf{D}_v$ for $v \in \mathcal{V}_s$ and $v = \bot$ otherwise. If $\mathcal{V}_s = \mathcal{V}$, then s is a *state*. $s_\bullet$ is the initial state and $s_\star$ is a partial state representing the goal. Given partial states s and s', we write $s' \subseteq s$ if $s'[v] = s[v]$ for all $v \in \mathcal{V}_{s'}$.

The *result* of applying a valuation e to valuation s is the valuation $\mathit{res}(s, e)$ such that $\mathit{res}(s, e)[v] = e[v]$ if $e[v] \neq \bot$ and $\mathit{res}(s, e)[v] = s[v]$ otherwise. A is a finite set of *probabilistic actions*. Each $\alpha \in \mathsf{A}$ consists of a *precondition* $\mathit{pre}(\alpha)$ given by a valuation over $\mathcal{V}$, a set $\mathit{eff}(\alpha)$ of *effects*, each of which is a valuation over $\mathcal{V}$, and a probability distribution $\Pr_\alpha(\cdot) \in \mathit{Dist}(\mathit{eff}(\alpha))$ such that $\Pr_\alpha(e)$ represents the probability of $\mathit{res}(s, e)$ being the state resulting from applying α in state s. $C(\alpha) \in \mathbb{R}_+^*$ is the immediate cost of applying α.

A probabilistic SAS$^+$ task $\langle \mathcal{V}, \mathsf{A}, s_\bullet, s_\star, C \rangle$ defines an SSP $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ where $s_{\text{init}} = s_\bullet$, $S = \times_{v \in \mathcal{V}} \mathsf{D}_v$, $G = \{s \in \mathsf{S} | s_\star \subseteq s\}$, $A(s) = \{\alpha \in \mathsf{A} | \mathit{pre}(\alpha) \subseteq s\}$, $P(t|s,\alpha) = \sum_{e \in \mathit{eff}(\alpha) \text{ s.t. } t = \mathit{res}(s,e)} \Pr_\alpha(e)$, and $T(s) = 0$ for all states $s \in G$. A C-SSP can be compactly represented by a probabilistic SAS$^+$ task whose cost function has been replaced with the corresponding vectors of cost functions and intervals.

2.1 SSPs with Multi-Objective PLTL Constraints

The main contribution of this paper is to extend the scope of heuristic search to SSPs with multi-objective probabilistic LTL constraints, which we define next.

LTL. Let $\langle \mathcal{V}, \mathsf{A}, s_\bullet, s_\star, C \rangle$ be a probabilistic SAS$^+$ task and $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ the SSP it defines. Let $AP = \{(v, d) \mid v \in \mathcal{V}, d \in \mathsf{D}_v\}$ be the finite set of *atoms*. LTL formulas ψ are described by the following grammar:

$$\psi = \mathsf{true} \mid (v, d) \in AP \mid \psi \wedge \psi \mid \psi \vee \psi \mid \neg\psi \mid \mathbf{X}\,\psi \mid \psi\,\mathbf{U}\,\psi$$

The standard semantics of LTL interprets formulas over infinite sequences of states. Formally, for a formula ψ and a path $p \in S^\omega$, the satisfaction relation $p \models \psi$ is defined as follows:

$p \models \top$
$p \models (v, d)$ iff $p_1[v] = d$
$p \models \neg\psi$ iff $p \not\models \psi$
$p \models \psi_1\,\mathbf{U}\,\psi_2$ iff $\exists i \geq 1$ s.t. $p(i) \models \psi_2$ and $p(j) \models \psi_1\ \forall 1 \leq j < i$
$p \models \psi_1 \wedge \psi_2$ iff $p \models \psi_1$ and $p \models \psi_2$
$p \models \psi_1 \vee \psi_2$ iff $p \models \psi_1$ or $p \models \psi_2$
$p \models \mathbf{X}\,\psi$ iff $p(2) \models \psi$

Planning, however, seeks finite state sequences ending in a goal state, and therefore usually interprets LTL over *finite* paths. A popular semantics for that is the *infinite extension semantics* [BK98, BH10], which gives a formula the truth value it would have under the standard semantics by infinite repetition of a path's last state. (Other popular semantics include f-FOLTL [BM06] and LTL$_f$ [DV13], but they do not appear to offer any advantage in our context.) That is, for any $p \in S^+$, we define the satisfaction relation $p \models_{IE} \psi$ as follows:

$$p \models_{IE} \psi := \begin{cases} \mathsf{true} & \text{if } p;(\text{last}(p))^\omega \models \psi \\ \bot & \text{Otherwise} \end{cases}$$

PLTL. A *probabilistic LTL (PLTL) formula* is of the form $\mathbf{P}_{\in z}\,\psi$ where $z \subseteq [0, 1]$. Informally, $\mathbf{P}_{\in z}\,\psi$ states that the LTL formula ψ holds with a probability that lies in the interval z. A *multi-objective PLTL (MO-PLTL) formula* is a conjunction $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\,\psi_i$ of PLTL formulas, for some *arity* $k \geq 0$. If $k = 0$ then φ is equivalent to the constant true.

The probability of ψ being satisfied by $\mathcal{S}$ under π is defined as

$$\Pr^\pi_{\mathcal{S}}(\psi) = \sum_{r \in \text{GRuns}(s_{\text{init}}, \pi) \text{ s.t. } r \models_{IE} \psi} P(r|\pi).$$

We say that $\mathcal{S}$ and π *satisfy* $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\,\psi_i$ and write $\mathcal{S}, \pi \models \varphi$ iff $\Pr^\pi_{\mathcal{S}}(\psi_i) \in z_i$ for all $i = 1..k$.

We can now state the problem we want to solve:

Definition 1 (MO-PLTL SSP Problem) *Let $\mathcal{T}$ be a probabilistic SAS$^+$ task, $\mathcal{S}$ the SSP it defines, and φ an MO-PLTL formula. Find an optimal policy π^* for $\mathcal{S}$ and φ, i.e., $\mathcal{S}, \pi^* \models \varphi$ and $V^{\pi^*} \leq V^\pi$ for all unrestricted proper policies π for $\mathcal{S}$ such that $\mathcal{S}, \pi \models \varphi$. Return failure if no unrestricted proper policy π for $\mathcal{S}$ exists such that $\mathcal{S}, \pi \models \varphi$.*

3 Related Work

Algorithms that can be used for the MO-PLTL SSP problem (Def. 1) are given in [KP13, FKNP11, EKVY08]. The overall approach is similar to single-LTL policy synthesis in that it constructs a product automaton and reduces to certain reachability problems determined by the end-components of

that automaton. These algorithms can be used for both synthesis and model checking, i.e., to check that all policies satisfy the given MO-PLTL formula φ; see, e.g., [KP13]. They are based on building NBAs for each LTL formula ψ_i followed by transformation into DRAs. Both steps require, worst-case exponential time and space each, and are together double-exponential in $|\varphi|$ (and polynomial in $|S|$). Also, they accept single-sided bounds only, $> u_i$ or $\geq u_i$, where $u_i \in [0, 1]$. Our interval bounds "$\in z_i$" can be compiled away into that form at the cost of doubling the size of φ. This can worsen the overall costs, hence, to double-exponential in $2|\varphi|$ which can be problematic even for small φ.

It is important to note that the mentioned algorithms solve the more general problem of policy synthesis for MDPs as opposed to SSPs, that is, probabilistic problems with no goal states. This problem is known to be complete for double-exponential time [CY95]. Indeed, we can show that our tailored SSP synthesis algorithm is strictly less complex (Theorem 4) and no extra penalty needs to be paid for accepting intervals.

It could be argued that MO-PLTL in the context of SSPs is closer to synthesis for probabilistic LTL over *finite* traces. Indeed, automata-based approaches to planning problems for LTL constraints over finite trace semantics have been proposed in [DV13, DV15]. These techniques could be employed for MO-PLTL policy synthesis following the architecture outlined above (cf. [KP13]). However, this would again incur double-exponential worst-time costs, for building NFAs and determinising them afterwards.

In yet another approach [LPH15, LPH17] consider syntactic co-safe MO-PLTL constraints, which are reachability queries and can be represented by DFAs of double-exponential size in the size of the contraints. Notice that our MO-PLTL constraints are not restricted in that way and allow for, e.g., the formulation of *maintenance goals* (see below for examples).

The existence of goals make MO-PLTL SSPs amenable (also) to heuristic search that builds only a small fraction of the state space on-the-fly. Our approach extends existing work from the planning community on deterministic and non-deterministic planning with LTL constraints to the probabilistic case. Relevant work include in particular compilation approaches which translate LTL and finite LTL variants into various types of automata whose states, transitions and accepting conditions are then incorporated as additional variables, actions, or constraints in the factored planning problem description [Ede06, BM06, TB15, CTM+17]. The two key advantages of these approaches and ours are: a) they deal with exponentially more compact non-deterministic automata, leaving to the planner the choice of how to resolve the non-determinism, and b) they use efficient planning algorithms (e.g. planning via heuristic search), which do not explicitly generate the whole state space.

The use of progression as an alternative to automata to generate modes originated in the TLPlan planner [BK98]. Progression has been used in the probabilistic planning setting, for instance to accommodate non-Markovian rewards [TGS+06], but not to handle PLTL constraints.

4 MO-PLTL SSPs as Constrained SSPs

This section contains our formal framework for translating MO-PLTL SSPs into Constrained SSPs (C-SSPs) to be solved for finite-memory policies. The translation abstracts from how MO-PLTL constraints are dealt with. We then describe two instances of this framework which respectively use automata and progression to represent the policy modes.

In this section let $\mathcal{T}$ be a probabilistic SAS$^+$ task, $\mathcal{S}$ the SSP it defines, and $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ an MO-PLTL formula.

Finite-Memory Policies. [BK08] A *stochastic finite-memory policy for an SSP $\mathcal{S}$* is a DFA $\pi_{\text{fin}} = (M, \text{start}, \text{mod}, \text{act})$ where: M is a finite set of *modes*, $\text{start} \in M$ is an initial mode, $\text{mod}\colon M \times S \mapsto M$ is the mode transition function, and $\text{act}\colon M \times S \mapsto \mathit{Dist}(A)$ is the action probability function such

that, for all $\langle m, s\rangle \in M \times S$, $\text{act}(m, s)(\alpha) \geq 0$ only if $\alpha \in A(s)$ and 0 otherwise. A finite-memory policy π_{fin} can be used whenever an unrestricted policy π is required by defining $\pi(s_1 \cdots s_n) = \text{act}(m_n, s_n)$ where $m_1 = \text{start}$ and $m_i = \text{mod}(m_{i-1}, s_i)$ for $i = 2..n$.[2]

Below we define modes M_i and components start_i and mod_i, for $i = 1..k$. These are compounded into a partial finite-memory policy $(\boldsymbol{M}, \textbf{start}, \textbf{mod}, \cdot)$ for $\mathcal{S}$, where $\boldsymbol{M} = M_1 \times \cdots \times M_k$, $\textbf{start} = \langle \text{start}_1, \ldots, \text{start}_k\rangle$, and $\textbf{mod}(\langle m_1, \ldots, m_k\rangle, s) = \langle \text{mod}_1(m_1, s), \ldots, \text{mod}_k(m_k, s)\rangle$. Then we compile into a C-SSP $C^\times$ such that any optimal policy $\pi^*_{C^\times}$ for $C^\times$ can be used as the act-component for the finite-memory policy:

Definition 2 (Product C-SSP $C^\times$) *Given an MO-PLTL SSP $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ and $(\boldsymbol{M}, \textbf{start}, \textbf{mod}, \cdot)$ as above, the product C-SSP $C^\times$ is the C-SSP $(S^\times, s^\times_{\text{init}}, G^\times, A, P^\times, \boldsymbol{C}^\times, \boldsymbol{T}^\times, \boldsymbol{z})$ with k secondary costs where: the state space $S^\times = \boldsymbol{M} \times S$; the initial state $s^\times_{\text{init}} = \langle \textbf{start}, s_{\text{init}}\rangle$; the goal set $G^\times = \boldsymbol{M} \times G$; $A(\langle \boldsymbol{m}, s\rangle) = A(s)$ for all $\boldsymbol{m} \in \boldsymbol{M}$; the transition probability function is $P^\times(\langle \boldsymbol{n}, t\rangle | \langle \boldsymbol{m}, s\rangle, \alpha) = P(t|s, \alpha)$ if $\alpha \in A(s)$ and $\boldsymbol{n} = \textbf{mod}(\boldsymbol{m}, t)$, otherwise 0; the main action cost $C^\times_0 = C$, and $C^\times_i(\alpha) = 0$ for all $\alpha \in A$ and $i \in 1..k$; $T^\times_0(\bullet) = 0$ and, for $i \in 1..k$, $T^\times_i(\langle \boldsymbol{m}, s\rangle) = 1$ if $\langle \boldsymbol{m}, s\rangle \in \text{Accept}_i$ and 0 otherwise; and z_i is the probability interval for ψ_i in φ.*

The mentioned sets $\text{Accept}_i \subseteq G^\times$, defined below, are mode specific. Informally, $\langle \boldsymbol{m}, s\rangle \in \text{Accept}_i$ means that ψ_i is IE-satisfied by all finite runs of $\mathcal{S}$ from s_{init} to the goal s.

As $C^\times$ is an ordinary C-SSP, any off-the-shelf C-SSP solver can be used to compute an optimal policy $\pi^*_{C^\times}$ for $C^\times$. Moreover, $\pi^*_{C^\times}$ is stationary and stochastic [Alt99], that is, $\pi^*_{C^\times}$ maps states $\langle \boldsymbol{m}, s\rangle \in S^\times$ to probability distributions over the set of actions $A(s)$. Thus, $\pi^*_{C^\times}$ can be used as action probability function, i.e., $\text{act}(\boldsymbol{m}, s)(\alpha) = \pi^*_{C^\times}(\langle \boldsymbol{m}, s\rangle, \alpha)$.

4.1 Büchi Automaton Mode

In this section we instantiate our C-SSP framework with NBA-based modes. Unlike the policy synthesis methods in "Related Work", we avoid NBA determinisation at up-front exponential costs by using heuristic search and on-the-fly determinisation of the NBA.

In order to employ existing LTL to NBA algorithms we need to equip the given formula with IE-semantics: for an LTL formula ψ let $\psi^{IE} = \psi \wedge \mathbf{F}(\bigwedge_{a \in AP}(a \rightarrow \mathbf{G}\, a) \wedge (\neg a \rightarrow \mathbf{G}\, \neg a))$, which is a faithful encoding of ψ wrt. the IE-semantics in standard LTL [BH10]. Let $\mathcal{B}_{\psi^{IE}}$ denote an NBA for ψ^{IE}. Then it follows that $\mathcal{B}_{\psi^{IE}}$ accepts a run r iff $r = p; (\text{last}(p))^\omega$ and $p \models_{IE} \psi$, for some finite path p.

In more detail, let $\mathcal{B}_{\psi^{IE}} = (Q, S, \Delta, Q_{\text{init}}, F)$, where: Q is the finite set of states; S is the input alphabet (i.e., the set of states of the given SSP $\mathcal{S}$); $\Delta\colon Q \times S \rightarrow 2^Q$ is the non-deterministic transition function; $Q_{\text{init}} \subseteq Q$ are the initial states; and $F \subseteq Q$ is the acceptance set. As a non-standard notion, we say that a state $q \in Q$ is *accepting with $s \in S$* iff (1) starting from q some strongly connected component $scc \subseteq Q$ is reachable by 0 or more transitions with s only, (2) $scc \cap F \neq \emptyset$, and (3) $\Delta(q', s') = \emptyset$ for all $q' \in SCC(q)$ and $s' \in S$ with $s \neq s$.[3] Now we can characterize satisfaction in a pleasant way:

Lemma 3 *Let $p \in S^+$ be a path with $p_1 = s_{\text{init}}$, ψ an LTL formula and $\mathcal{B}_{\psi^{IE}}$ as defined above. Then $p \models_{IE} \psi$ iff starting from some state in Q_{init} and following the states in p a state $q \in Q$ is reachable that is accepting with* $\text{last}(p)$.

For each LTL formula ψ_i of the given MO-PLTL constraint φ we need one NBA, denoted by $\mathcal{B}_{\psi_i^{IE}} = (Q_i, S, \Delta_i, Q_{\text{init},i}, F_i)$. The *Büchi-Automata based finite-memory policy (NBA policy)* is $\pi^{\text{NBA}} =$

[2] The literature usually defines the mode transition function mod dependent on the source state s_{i-1}, not the target state s_i, i.e., $m_i = \text{mod}(m_{i-1}, s_{i-1})$. This change is theoretically inconsequential, but technically more convenient for us.

[3] In words, condition 3 says that the only transitions among any states in *scc* are with s.

$(\boldsymbol{M}^{\text{NBA}}, \mathbf{start}^{\text{NBA}}, \mathbf{mod}^{\text{NBA}}, \pi^*_{C^\times})$ where: $M_i^{\text{NBA}} = 2^{Q_i}$ (the powerset of the NBA state space); $\text{start}_i^{\text{NBA}} = \bigcup_{q \in Q_{\text{init}}} \Delta_i(q, s_{\text{init}})$; $\text{mod}_i^{\text{NBA}}(m, s) = \bigcup_{q \in m} \Delta_i(q, s)$; $\pi^*_{C^\times}$ is an optimal solution for $C^\times$ obtained using $\boldsymbol{M}^{\text{prog}}$, $\mathbf{start}^{\text{prog}}$, $\mathbf{mod}^{\text{prog}}$ and $\text{Accept}_i = \{\langle \boldsymbol{m}, s\rangle \in G^\times \mid \text{some } q \in m_i \text{ is accepting with } s\}$.

To save space we do not spell out soundness and completeness results. They are analogous to Theorems 5 and 6 below. The main difference is in the definition of Accept_i, which, as per Lemma 3 correctly identifies IE-satisfaction of ψ_i.

4.2 Formula Progression Mode

By $\text{CNF}(\psi, s)$ we denote the conversion of the LTL formula ψ into conjunctive normal form with on-the-fly simplification by evaluating non-temporal formulas in $s \in S$. Thanks to simplification and equivalences like $\psi_1 \mathbf{U} \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge \mathbf{X}(\psi_1 \mathbf{U} \psi_2))$, every formula in $\text{CNF}(\psi, s)$ is an $\mathbf{X}$-formula. This allows us to take $\text{CNF}(\psi, s)$ to successor states by means of an un$\mathbf{X}$-operator, which strips each formula in $\text{CNF}(\psi, s)$ of its $\mathbf{X}$-operator. By $\Sigma(\psi_i)$ we denote a certain set of formulas obtained from the i-th PLTL constraint ψ_i (of size quadratic in the size of ψ_i.) See the appendix for details.

The *progression-based finite-memory policy is* $\pi^{\text{prog}} = (\boldsymbol{M}^{\text{prog}}, \mathbf{start}^{\text{prog}}, \mathbf{mod}^{\text{prog}}, \pi^*_{C^\times})$ where: $M_i^{\text{prog}} = 2^{\Sigma(\psi_i)^3}$; $\text{start}_i^{\text{prog}} = \text{CNF}(\psi_i, s_{\text{init}})$; $\text{mod}_i^{\text{prog}}(m, s) = \text{CNF}(\text{un}\mathbf{X}(m), s)$; and $\pi^*_{C^\times}$ is an optimal solution for $C^\times$ obtained using $\boldsymbol{M}^{\text{prog}}$, $\mathbf{start}^{\text{prog}}$, $\mathbf{mod}^{\text{prog}}$ and $\text{Accept}_i = \{\langle \boldsymbol{m}, s\rangle \in G^\times \mid s \models_{IE} m_i\}$. Notice that π^{prog} exists iff $C^\times$ has a solution.

Notice that, every mode is a set of sets formulas of size at most 3 over an a priori fixed domain $\Sigma(\psi_i)$ whose size is polynomial in the size of ψ. This is possible thanks to a polynomial "Tseitin-style" CNF transformation [Tse68].

Theorem 4 (Complexity) *Let $\mathcal{S}$ be an SSP, φ an MO-PLTL constraint, and $C^\times$ the C-SSP obtained using $\boldsymbol{M}^{\text{prog}}$, $\mathbf{start}^{\text{prog}}$, and $\mathbf{mod}^{\text{prog}}$. Then there is a linear program $\text{LP}_{C^\times}$ whose solution, if any, defines the optimal policy $\pi^*_{C^\times}$ for $C^\times$. $\text{LP}_{C^\times}$ can be obtained in time and space at most $O(|S|) \cdot (2^{O(|\varphi|)^7})$.*

The mentioned linear program $\text{LP}_{C^\times}$ and the mapping of its solution to the solution of $C^\times$ is defined in the Appendix. We emphasize that Theorem 4 is a marked improvement over using generic model-checking algorithms of double-exponential complexity (see "Related Work"). Our qualitative main results are as follows.

Theorem 5 (Soundness) *If π^{prog} exists then it is an optimal policy for $\mathcal{S}$ and φ, i.e., $\mathcal{S}, \pi^{\text{prog}} \models \varphi$ and $V^{\pi^{\text{prog}}} \leq V^\pi$ for all unrestricted proper policies π such that $\mathcal{S}, \pi \models \varphi$.*

Theorem 6 (Completeness) *If π is an unrestricted proper policy for $\mathcal{S}$ such that $\mathcal{S}, \pi \models \varphi$ then π^{prog} exists and is a proper policy such that $\pi^{\text{prog}} \models \varphi$ and $V^{\pi^{\text{prog}}} \leq V^\pi$.*

Theorems 5 and 6 entail solvability of the MO-PLTL SSP Problem (Def. 1). By Theorem 4, $\text{LP}_{C^\times}$ can be used for that. Finally, we need to add that the complexity results in this section are valid in the context of the progression mode but not when NBAs are used, whether to compute modes or heuristics as we do in the next section.

5 Heuristic Search Algorithms

In the previous sections, we showed that an MO-PLTL SSP can be compiled into a Constrained SSP (C-SSP) and, in this section, we leverage this result in order to solve MO-PLTL SSPs using heuristic search algorithms for C-SSPs.

Traditionally, C-SSPs are solved as a single LP representing all the (reachable) search space at once similarly to Value Iteration [D'E63, Alt99]. In the AI community this LP is referred as the dual LP for (C-)SSPs and its variables are the policy's occupation measures $x_{s,a}$ representing the expected number of times action $a \in A(s)$ will be executed in state s. The main advantage of the dual formulation is that the expectation of any function $f : S \times A \to \mathbb{R}$ (e.g., the cost functions C_j) over the policy encoded by x can be easily computed by $\sum_{s,a} x_{s,a} f(s, a)$.

Another important feature of the dual formulation that we exploit in this section is that it can be interpreted as a *probabilistic flow problem*, where $x_{s,a}$ describes the flow leaving state s via action a. Using this interpretation, we can see the dual LP as a flow problem where the expected total cost to reach a goal (sink) from the initial state (source) is minimised.

The drawback of this single dual LP approach is the same as that of Value Iteration, namely, the whole reachable search space of the problem must the computed a priori, making this approach not viable for large problems. To address this issue, i-dual [TTSW16] and its successor i^2-dual [TTH17] were introduced. Both algorithms solve C-SSPs using heuristic search by generating and solving increasingly large LPs.

5.1 i^2-dual for Product C-SSPs

In our context, given a Product C-SSP $C^\times = (S^\times, s^\times_{\text{init}}, G^\times, A, P^\times, \boldsymbol{C}^\times, \boldsymbol{T}^\times, \boldsymbol{z})$ and $\mathcal{T}$ the SAS$^+$ task associated with $C^\times$ as input, i^2-dual incrementally generates and explores larger *partial problems* of $C^\times$ starting from $s^\times_{\text{init}}$. Given a set $\hat{S} \subseteq S^\times$ of explored states and a set $\Gamma \subseteq \hat{S}$ of fringe states of the search, the partial problem solved by i^2-dual is shown in LP1 where, for readability, we abbreviate $\text{in}(\langle \boldsymbol{m}, s \rangle)$ as $\text{in}(\boldsymbol{m}, s)$, $\text{out}(\langle \boldsymbol{m}, s \rangle)$ as $\text{out}(\boldsymbol{m}, s)$ and $x_{\langle \boldsymbol{m}, s \rangle, \alpha}$ as $x_{\boldsymbol{m}, s, \alpha}$. The constraints of LP1 can be categorized as follows (Fig. 1):

Probabilistic Flow Network (C3–C6). Representation of the states explored so far using occupation measures.

Multiplexer (C7–C8). These constraints extract the flow from Γ and, for each SAS$^+$ variable $v \in \mathcal{V}$, they redirect a copy of this flow to the appropriate state $d \in D_v$ of *the projection of $\mathcal{T}$ onto v*.

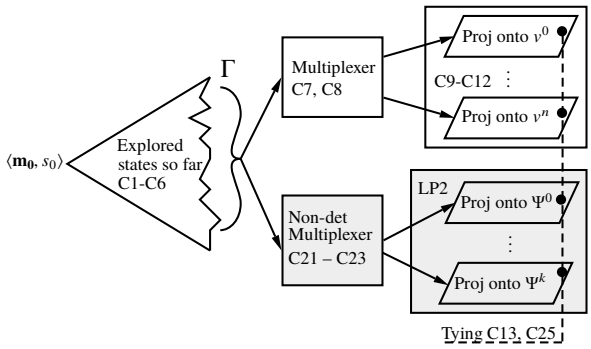


Figure 1: Representation of the flow network solved in each iteration of i^2-dual (non-shaded network – LP1) and of our novel algorithm PLTL-dual (full network – LP3).

$$\min_{x} \sum_{\langle \boldsymbol{m},s\rangle \in \hat{S}, \alpha \in A(s)} x_{\boldsymbol{m},s,\alpha} C_0^{\times}(\alpha) + \sum_{\langle \boldsymbol{m},s\rangle \in \hat{S} \cap G^{\times}} \mathrm{in}(\boldsymbol{m}, s) T_0^{\times}(\langle \boldsymbol{m}, s\rangle) + \sum_{d \in \mathsf{D}_{v'}, \alpha \in A} x^{v'}_{d,\alpha} C_0^{\times}(\alpha) \tag{LP1}$$

$$\text{s.t.} \quad x_{\boldsymbol{m},s,\alpha} \geq 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{S}, \alpha \in A(s) \quad \text{(C1)}$$

$$x^{v}_{d,\alpha} \geq 0 \qquad \forall v \in \mathcal{V}, d \in D_v, \alpha \in A \quad \text{(C2)}$$

$$\mathrm{in}(\boldsymbol{m}, s) = \sum_{\langle \boldsymbol{n},t\rangle \in \hat{S}, \alpha \in A(s')} x_{\boldsymbol{n},t,\alpha} P^{\times}(\langle \boldsymbol{m}, s\rangle | \langle \boldsymbol{n}, t\rangle, \alpha) \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{S} \quad \text{(C3)}$$

$$\mathrm{out}(\boldsymbol{m}, s) = \sum_{\alpha \in A(s)} x_{\boldsymbol{m},s,\alpha} \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{S} \setminus G^{\times} \quad \text{(C4)}$$

$$\mathrm{out}(\mathbf{start}, s_{\mathrm{init}}) - \mathrm{in}(\mathbf{start}, s_{\mathrm{init}}) = 1 \quad \text{(C5)}$$

$$\mathrm{out}(\boldsymbol{m}, s) - \mathrm{in}(\boldsymbol{m}, s) = 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in \hat{S} \setminus G^{\times} \quad \text{(C6)}$$

$$p^{v}_0(g) = \sum_{\langle \boldsymbol{m},s\rangle \in \hat{S} \cap \mathsf{G}^{\times}} \mathrm{in}(\boldsymbol{m}, s) \qquad \forall v \in \mathcal{V} \quad \text{(C7)}$$

$$p^{v}_0(d) = \sum_{\langle \boldsymbol{m},s\rangle \in \Gamma, s[v]=d} \mathrm{in}(\boldsymbol{m}, s) \qquad \forall v \in \mathcal{V}, d \in \mathsf{D}_v \quad \text{(C8)}$$

$$in^{v}(d) = \sum_{d' \in \mathsf{D}_v, \alpha \in A \cup \{a_g\}} x^{v}_{d',\alpha} P(d|d', \alpha) \qquad \forall v \in \mathcal{V}, d \in \mathsf{D}_v \cup \{g\} \quad \text{(C9)}$$

$$out^{v}(d) = \sum_{\alpha \in A \cup \{a_g\}} x^{v}_{d,\alpha} \qquad \forall d \in \mathsf{D}_v \text{(C10)}$$

$$out^{v}(d) - in^{v}(d) = p^{v}_0(d) \qquad \forall v \in \mathcal{V}, d \in \mathsf{D}_v \text{(C11)}$$

$$in^{v}(g) = 1 \qquad \forall v \in \mathcal{V} \text{(C12)}$$

$$\sum_{d_i \in \mathsf{D}_{v_i}} x^{v_i}_{d,\alpha} = \sum_{d_j \in \mathsf{D}_{v_j}} x^{v_j}_{d,\alpha} \qquad \forall v_i, v_j \in \mathcal{V}, \alpha \in A \text{(C13)}$$

$$\sum_{\langle \boldsymbol{m},s\rangle \in \hat{S} \cap Accept(i)} in(\boldsymbol{m}, s) \in z_i \qquad \forall i \in \{1, \ldots, k\} \text{(C14)}$$

Projection Occupation Measure Heuristic (C9–C12). For each SAS$^+$ variable $v \in \mathcal{V}$, these constraints represent the projection of $\mathcal{T}$ onto v. This projection is an SSP in itself over the state space $D_v \cup \{g\}$ where g represents the sink of the overall problem (see [TTH17] for formal definition). The variables $x^{v}_{d,\alpha}$ are the occupation measures for the projection onto v. The projections are used for

obtaining a lower bound on the expected cost of reaching the goal set $G^\times$ of the overall problem from Γ.

Tying constraints (C13). These constraints tie the projections onto the different $v \in \mathcal{V}$ together. In order to avoid the complexity of the original problem, the projections are tied using the following relaxation: for all $\alpha \in A$, the expected number of times action a is applied over the entire projection must be the same for every projection.

PLTL Constraint (C14). These linear constraints enforce the PLTL constraints. The left-hand side of the constraints follows from the definition of $C_i^\times$ and $T_i^\times$. Due to the lack of domain-independent lower bounds for PLTL constraints, i^2-dual can only handle intervals z_i that are right-closed and finishing at 1, thus $\mathbf{P}_{\in z_i}\, \psi_i$ is represented using two constraints: $\mathbf{P}_{\in [\underline{z}_i, 1]}\, \psi_i$ and $\mathbf{P}_{\in [1-\overline{z}_i, 1]}\, \neg\psi_i$.

The objective function of LP1 minimises the expected primary cost of reaching the fringe Γ and the goal states in $\hat{S}$ (first and second summation, respectively) plus the heuristic estimate to solve the problem from the states reached in Γ to the goal set $G^\times$ of the original problem (third summation). Due to the tying constraints (C13), any variable $v' \in \mathcal{V}$ can be used in the third summation. Notice the updates in the search space explored so far and the heuristics estimates happens together and in the same LP, thus the search and heuristic computation work in unison instead of the search method driving the heuristic computation.

Once LP1 is solved, the fringe states reachable by its optimal solution are expanded and a new iteration of i^2-dual is performed. i^2-dual stops when all the flow injected in the initial state reaches the goal set $G^\times$. The optimal policy $\pi^*_{C^\times}$ is encoded in the optimal solution x^* of LP1 in the last iteration of i^2-dual: $\pi^*_{C^\times}(\langle \boldsymbol{m}, s\rangle, \alpha) = \frac{x^*_{\boldsymbol{m},s,\alpha}}{\text{out}(\boldsymbol{m},s)}$ for all $\langle \boldsymbol{m}, s\rangle \in S^\times$ and $\alpha \in A(s)$ such that $\text{out}(\boldsymbol{m}, s) > 0$.

5.2 Heuristic for PLTL Constraints

Since the secondary action costs $C_i^\times$ of $C^\times$ are always zero, the heuristics embedded in i^2-dual is unable to estimate the probability of ψ_i being true; therefore, i^2-dual performs heuristic search only for the primary cost. As we show in our experiments, this approach is not enough to solve large MO-PLTL SSPs. We address this issue by introducing a heuristic for the PLTL constraints, i.e., a function that can prioritize the search on $S^\times$ according to a given PLTL constraint ψ_i.

Formally, a *heuristic for ψ_i* is any function $h^{\psi_i} : S^\times \to \mathbb{R}_+$ that estimates the probability of ψ_i being satisfied. h^{ψ_i} is an *admissible heuristic* if, for all $s^\times \in S^\times$, $h^{\psi_i}(s^\times) \geq \max_{\pi \in \Pi^*} V_i^\pi(s^\times)$ where Π^* is the set of optimal policies for $C^\times$ and V_i^π is the policy π value function for the secondary cost associated with ψ_i (Eq. 1). Thus, the trivial admissible heuristic for ψ_i is the always-1 function. In this paper, we consider only admissible heuristics because they do not prune feasible solutions from the search, thus, the soundness, completeness and optimally guarantees of i^2-dual are preserved.

In order to be able to integrate our novel heuristic to i^2-dual and take advantage of the unison search property of i^2-dual, we propose a heuristic based on projections onto the Non-deterministic Büchi Automaton (NBA) of each LTL formula. To get a good estimate from an NBA projection, we need to handle both the non-determinism of the NBA transitions as well as the probabilistic effects of the SAS$^+$ actions. We accomplish this by modelling the NBA projection as a relaxed SSP over the NBA states.

Formally, let $\mathcal{B}_i = (Q_i, S, \Delta_i, \chi_{\text{init},i}, F_i)$ be the NBA for the PLTL constraint ψ_i. Given an effect e of a probabilistic SAS$^+$ action $\alpha \in A$ and $s \in S$, *e is consistent with s* if $res(pre(\alpha), e) \subseteq s$. We call $\text{Comp}(e, q)$ the set of successor states of $q \in Q_i$ compatible with effect e of action α. Formally $\text{Comp}(e, q) = \{q' \in Q_i | \exists (q, s, q') \in \Delta_i, e \text{ is consistent with } s\}$.

The relaxed SSP representing the projection of $C^{\times}$ onto $\mathcal{B}_i$ is $\mathcal{S}^{\psi_i} = (Q_i, \chi_{\text{start}}, F_i, B^{\psi_i}, P^{\psi_i})$ whose set of states Q_i is that of $\mathcal{B}_i$ and whose set of goal states F_i is the set of accepting states of $\mathcal{B}_i$. This relaxed SSP has a **non-deterministic initial state** $\chi_{\text{start}} \subseteq Q_i$, i.e., no probability distribution over χ_{start} is given. B^{ψ_i} is the set of actions and $B^{\psi_i}(q) = \{\beta_{\alpha,p} | p \in \bigtimes_{e \in \mathit{eff}(\alpha)} \text{Comp}(e, q)\}$ is the set of actions applicable in any $q \in Q_i$. Lastly, $P^{\psi_i}(q'|q, \beta_{\alpha,p}) = \sum_{e \in \mathit{eff}(\alpha), p[e]=q'} \Pr_{\alpha}(e)$, is the probability of transitioning from q to q' after applying $\beta_{\alpha,p} \in B^{\psi_i}(q)$, where $p[e]$ is the element in the e-th coordinate of p. $\mathcal{S}^{\psi_i}$ does not have a cost function and we are interested in finding a solution to $\mathcal{S}^{\psi_i}$ that maximizes the probability of reaching the goal set, that is, the accepting states of $\mathcal{B}_i$.

Our heuristic for a PLTL constraint, called the NBA projection heuristic and denoted by $h^{\psi_i}_{\text{BA}}$ for the i-th PLTL constraint, is formally presented in LP2. $h^{\psi_i}_{\text{BA}}$ takes as input a non-empty subset χ_{start} of Q_i and returns an upper bound on the probability of ψ_i being true when χ_{start} is used as the non-deterministic initial state of $\mathcal{B}_i$.

$$\min_{x^{\psi_i}, p^{\psi_i}} \quad \sum_{q \in Q_i \setminus F_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} \tag{LP2}$$

$$\text{s.t.} \quad x^{\psi_i}_{q,\beta} \geq 0 \qquad \forall q \in Q_i, \beta \in B^{\psi_i}(q) \cup \{\alpha_{\text{sink}}\} \tag{C15}$$

$$p^{\psi_i}_{q} \geq 0 \qquad \forall q \in \chi_{\text{start}} \tag{C16}$$

$$x^{\psi_i}_{q,\alpha_{\text{sink}}} + \sum_{\beta \in B^{\psi_i}(q)} x^{\psi_i}_{q,\beta} - \sum_{\substack{q' \in Q_i \\ \beta \in B^{\psi_i}(q')}} P(q|q', \beta) x^{\psi_i}_{q',\beta} = p^{\psi_i}_{q} \qquad \forall q \in \chi_{\text{start}} \tag{C17}$$

$$x^{\psi_i}_{q,\alpha_{\text{sink}}} + \sum_{\beta \in B^{\psi_i}(q)} x^{\psi_i}_{q,\beta} - \sum_{\substack{q' \in Q_i \\ \beta \in B^{\psi_i}(q')}} P(q|q', \beta) x^{\psi_i}_{q',\beta} = 0 \qquad \forall q \in Q_i \setminus \chi_{\text{start}} \tag{C18}$$

$$\sum_{q \in \chi_{\text{start}}} p^{\psi_i}_{q} = \sum_{q \in Q_i} x^{\psi_i}_{q,\alpha_{\text{sink}}} = 1 \tag{C19}$$

The variables of LP2 are: the occupation measures $x^{\psi_i}_{q,\alpha}$ for this projection; and the input flows $p^{\psi_i}_{q}$ representing how the flow injected into the network is distributed within the states $q \in \chi_{\text{start}}$ of the non-deterministic initial state. While $x^{\psi_i}_{q,\alpha}$ is analogous to $x^{v}_{d,\alpha}$ of LP1, $p^{\psi_i}_{q}$ have no counterpart since they encode the non-deterministic initial state of $\mathcal{B}_i$.

Notice that an artificial action α_{sink} is used in LP2 (C15, C17, C18 and C19). This action represents the deterministic transition from a state $q \in Q_i$ to the sink and it is applicable in all states (C17 and C18). The source and sink constraints C19 enforces that 1 unit enters and leaves the network. C18 is the set of flow preservation constraints that, for all states $q \notin \chi_{\text{start}}$, forces the flow leaving q to the sink and other states $q' \in Q_i$ to equal the flow entering q. Similarly, C17 is the flow preservation constraint for source states.

The objective function of LP2 minimises the flow going into the sink from the non-accepting states of $\mathcal{B}_i$, thus it maximises the probability of reaching an accepting state $q \in F_i$ from χ_{start}. The admissibility of $h^{\psi_i}_{\text{BA}}$ is formalized by Theorem 7 and its proof is based on the fact that the trivial admissible heuristic for the PLTL constraints is the always-1 function and that the LP can move the flow from any state $q \in Q_i$ to a state $q' \in F_i$ except when q is part of a non-accepting bottom end component of $\mathcal{B}_i$. We refer to the Appendix for the complete proof.

Theorem 7 *$h^{BA}_{\psi_i}$ is an admissible heuristic for ψ_i.*

As the insights for the proof for Theorem 7 suggest, LP2 is encoding the problem of avoiding the non-accepting bottom end components of $\mathcal{B}_i$. Although there are more efficient ways of solving this problem (e.g., a look-up table), our encoding as a projection of the MO-PLTL SSP over ψ_i

pays off by being able to be integrated to i^2-dual. The integration is done by tying the actions in the projection onto each ψ_i to the projections onto the state variables $v \in \mathcal{V}$ of the probabilistic SAS$^+$ task. By tying all projections together, both SAS$^+$ and ψ_i projections, we force them to reach a relaxed agreement over their solutions for reaching the original set of goals $G^\times$ from the current fringe states Γ. This agreement provides better heuristic estimates for both the primary cost function and PLTL constraints.

5.3 PLTL-dual

Our last contribution is the integration of $h^{\psi_i}_{\text{BA}}$ to i^2-dual. The obtained algorithm, PLTL-dual, consists of the same iterative procedure as i^2-dual but LP1 is replaced by LP3. We assume that the mode space is the NBA mode. A visual representation of LP3 is presented in Figure 1 and its constraints can be categorized as:

i^2-dual constraints (C3 – C13). All constraints except the PLTL constraint (C14)

Non-deterministic Multiplexer (C21 – C23). The non-deterministic counterpart of the state variable multiplexer (C7 – C8). This multiplexer represents the non-deterministic distribution of flow from the mode m_i to the states $q \in m_i$. $\Gamma^{\psi_i} = \{m_i \in M_i | \exists \langle \boldsymbol{n}, s \rangle \in \Gamma \text{ s.t. } n_i = m_i\}$ represents all the observed NBA modes m_i of ψ_i in the fringe Γ and $\mathcal{D}(m_i, \Gamma^{\psi_i})$ denotes the NBA states $q \in m_i$ not present in any other mode $n_i \in \Gamma^{\psi_i}$, formally, $\mathcal{D}(m_i, \Gamma^{\psi_i}) = m_i \setminus (\cup_{n_i \in \Gamma^{\psi_i} | n_i \neq m_i} n_i)$.

NBA Projection Heuristic (C17–C18). The flow preservation constraints for $h^{\psi_i}_{\text{BA}}$ for each ψ_i.

PLTL sink constraint (C24). The replacement of the sink constraint of $h^{\psi_i}_{\text{BA}}$. The new sink extracts the same amount of flow that the non-deterministic multiplexer injected in each NBA projection.

NBA Tying constraints (C25). These constraints tie the NBA projections to the state variable projections. Any state variable $v \in \mathcal{V}$ can be used in these constraints since all state variables are already tied together by C13.

PLTL constraints (C26). These constraints replace C14 and contain the heuristic estimation provided by $h^{\psi_i}_{\text{BA}}$.

$$\min_{x} \sum_{\langle \boldsymbol{m},s\rangle \in \hat{S},\, \alpha \in A(s)} x_{\boldsymbol{m},s,\alpha} C_0^{\times}(\alpha) + \sum_{\langle \boldsymbol{m},s\rangle \in \hat{S} \cap G^{\times}} \mathrm{in}(\boldsymbol{m},s) T_0^{\times}(\langle \boldsymbol{m},s\rangle) + \sum_{d \in \mathsf{D}_{v'},\, \alpha \in A} x_{d,\alpha}^{v'} C_0^{\times}(\alpha) \tag{LP3}$$

$$\text{s.t.} \quad x_{s,\alpha} \geq 0,\ x_{d,\alpha}^{v} \geq 0,\ x_{q,\alpha}^{\psi_i} \geq 0,\ p_q^{\psi_i} \geq 0 \tag{C20}$$

constraints C3 – C13

$$\sum_{\langle \boldsymbol{m},s\rangle \in \Gamma \mid m_i = \{q\}} \mathrm{in}(\boldsymbol{m},s) \ \leq\ p_q^{\psi_i} \ \leq \sum_{\langle \boldsymbol{m},s\rangle \in \Gamma \mid q \in m_i} \mathrm{in}(\boldsymbol{m},s) \qquad \forall i \in \{1,\ldots,k\},\, q \in Q_i \tag{C21}$$

$$\sum_{q \in \mathcal{D}(m,\Gamma^{\psi_i})} p_q^{\psi_i} \ \leq \sum_{\langle \boldsymbol{n},s\rangle \in \Gamma \mid n_i = m} \mathrm{in}(\boldsymbol{n},s) \leq \sum_{q \in m} p_q^{\psi_i} \qquad \forall i \in \{1,\ldots,k\},\, m \in \Gamma^{\psi_i} \tag{C22}$$

$$\sum_{q \in Q_i} p_q^{\psi_i} = \sum_{\langle \boldsymbol{m},s\rangle \in \Gamma} \mathrm{in}(\boldsymbol{m},s) \qquad \forall i \in \{1,\ldots,k\} \tag{C23}$$

$$\sum_{q \in Q_i} x_{q,\alpha_{\mathrm{sink}}}^{\psi_i} = \sum_{\langle \boldsymbol{m},s\rangle \in \Gamma} \mathrm{in}(\boldsymbol{m},s) \qquad \forall i \in \{1,\ldots,k\} \tag{C24}$$

constraints C17 – C18 $\qquad \forall i \in \{1,\ldots,k\}$

$$\sum_{d \in \mathsf{D}_v} x_{d,\alpha}^{v} = \sum_{q \in Q_i} x_{q,\alpha}^{\psi_i} \qquad \forall i \in \{1,\ldots,k\},\, \alpha \in A \tag{C25}$$

$$\sum_{\langle \boldsymbol{m},s\rangle \in \hat{S} \cap \mathrm{Accept}(i)} in(\boldsymbol{m},s) + \sum_{q \in F_i} x_{q,\alpha_{\mathrm{sink}}}^{\psi_i} \in z_i \qquad \forall i \in \{1,\ldots,k\} \tag{C26}$$

LP3 assumes that the mode for each PLTL constraint ψ_i is the NBA $\mathcal{B}_i$ associated with ψ_i. This presents two key computational advantages: (i) the NBA projection heuristics do not need to compute $\mathcal{B}_i$ since they were already computed to be used as modes; and (ii) the non-deterministic multiplexer can easily relate the mode $m_i \subseteq Q_i$ and the NBA projection states $q \in Q_i$ since they are defined over the same set Q_i of $\mathcal{B}_i$. Nonetheless, PLTL-dual can be used with any mode as long as a translation from the non-NBA mode m_i to a subset of Q_i is provided for each PLTL constraint.

6 Experiments

In this section we empirically evaluate our heuristic search algorithms and compare their performance with that of Prism, a state-of-the-art model-checker implementing the static approach [KNP11]. We enforce a 30-minutes and 4-Gb cut-off for all experiments, and report results averaging 30 runs of each algorithm for each problem taken from the following two domains.

Factory. This domain features an assembly line with n machines, where each machine m_i produces a part p_i starting from the part p_{i-1} produced by machine m_{i-1}. There are actions for turning a machine on or off (cost 1), and for producing p_i using machine m_i once m_i is on and p_{i-1} is available. This makes p_{i-1} unavailable. The k machines $m_2 \ldots m_{k+1}$ are unreliable and fail to produce the new part p_i 20% of the time, but still make p_{i-1} unavailable. Production cost for reliable (resp. unreliable) machines is 5 (resp. 3). Initially, all machines are off and p_0 is available. In the goal, the machines are off again and part p_n has been produced. The two MO-PLTL constraints ($\mathbf{P}=1$) are that (1) m_1 is eventually started, and (2) once m_{i-1} stops for good, m_i has to be on, and then the same hand-shake applies to the next machine down the line and so on: $\mathbf{G}\,(on(m_1) \Rightarrow (on(m_1)\,\mathbf{U}\,(on(m_2) \wedge \mathbf{G}\,\neg on(m1) \wedge (on(m_2)\,\mathbf{U}\,(on(m_3) \wedge \mathbf{G}\,\neg on(m_2) \wedge \ldots \wedge (on(m_{n-1}) \wedge\ \mathbf{U}\,(on(m_n) \wedge \mathbf{G}\,\neg on(m_n)))\ldots)))))$.

Wall-e. In this domain, Wall-e and Eve are in a corridor with n distinct locations $l_1 \ldots l_n$ and n rooms $r_1 \ldots r_n$. Location l_i is connected to room r_i and to locations l_{i-1} and l_{i+1}. Wall-e and Eve can

be in any of these locations and rooms, either separately or together. They can move to a connected location, enter or exit a connected room, either separately or together. All actions are deterministic and have cost 1, except exiting a room together which has cost 5 and fails with 10% probability. Wall-e starts in l_1 and Eve in r_2, and the goal has Wall-e in l_n. The MO-PLTL constraints specify that: (1) they must eventually be together ($\mathbf{P} \geq 0.5$); (2) once they're together they remain together ($\mathbf{P} = 1$); (3) Eve must be at most 3 steps away from a room until they are together ($\mathbf{P} \geq 0.8$); (4) Eve visits the first 3 rooms r_1, r_2, r_3 ($\mathbf{P} = 1$); and (5) Wall-e never visits any room twice, except possibly r_n ($\mathbf{P} \geq 0.8$).

Algorithms. The algorithms considered are: (a) PLTL-dual, that is, i^2-dual with the NBA mode and NBA projection heuristic; (b) PLTL-dual(100), which is like PLTL-dual except that the NBA projection heuristic is only used for NBAs with less than 100 states – for the other formulas, the trivial heuristic is used; (c) i^2-dual with the NBA mode and the trivial heuristic for all formulas; (d) i^2-dual with the progression mode and the trivial heuristic for all formulas; and (e) the multi-objective version of Prism with the -lp option.[4] In our implementation, we use ltl3ba to produce the NBAs.

Results. The graphs in Figure 2 show the time spent by each algorithm on problems from the Factory domain (log scale) with $n = 2..8$ machines including $k = 0..(n-1)$ unreliable machines, and from the Wall-e domain with $n = 4..7$ rooms. We find it convenient to analyse the results by means of answering the following questions:

What is the best mode? To answer this question, we need to compare both modes using the same heuristics, i.e., the trivial heuristics. In this case, the progression mode outperforms the NBA mode in the Wall-e domain (25% faster for $n = 7$) while being statistically tied in the Factory domain. The reason for this performance advantage is that progression uses the observed information so far to simplify its search space thus reducing the overall number of states expanded, e.g., 8% less expanded states in Wall-e #7.

Is the NBA projection heuristic effective? Yes when the formulas are not trivial to check and the NBAs are not too large. The first condition is usual for heuristic search algorithms because, in easy search problems, a less accurate but cheap to compute heuristic is good enough. Constraint (2) of the Factory domain is an example of a constraint trivial to check since turning on a machine in the wrong order violates it. The second condition is an issue because large NBAs result in projections with a large number of LP variables. For instance, the NBA for constraint (5) of Wall-e #6 has 112 states and its projection uses 19863 LP variables. This represents about 40% of the total LP variables in the last LP solved by PLTL-dual in Wall-e #6. An example of good guidance provided by the NBA projection heuristic is constraint (3) of the Wall-e domain. This constraint has a small NBA (11 states) and violations can be detected early by using a look-ahead of size 3. Since the NBA projection heuristic performs search in the mode space, it is able to do early pruning for this constraint while the trivial heuristic, which cannot look beyond the current state, is unable to do.

What is the best algorithm? Although progression is the best mode when using the trivial heuristic, the best algorithm is PLTL-dual(100): it is statistically tied with i^2-dual with progression for the large Factory problems ($n = 8$) while it dominates all other algorithms in the Wall-e domain. As expected, the static approach employed by Prism is outperformed by our heuristic search approaches due to the prohibitive size of the DRA required by it. For instance, the DRA for constraint (2) of Factory #4,3 and constraint (5) of Wall-e #5 has, respectively, 29979 and 2857 states while their NBA has only 13 and 48 states, respectively.

[4] Without this option, Prism produces an error on both domains.

7 Conclusion and Future Work

We have provided the first heuristic search algorithm for SSPs with MO-PLTL constraints, and demonstrated both practical and in certain cases worst-case complexity improvements over state-of-the-art algorithms. Our approach performs an on-the-fly translation of the MO-PLTL SSP into a C-SSP, which is solved by extending recent heuristic search approaches for C-SSPs and guiding them using novel linear programming heuristics for MO-PLTL constraints. In the future, we plan to improve our NBA heuristics, design heuristics that work together with progression, and extend the scope of heuristic search to SSPs with larger subsets of PCTL*.

Acknowledgement

This research was funded by AFOSR grant FA2386-15-1-4015.

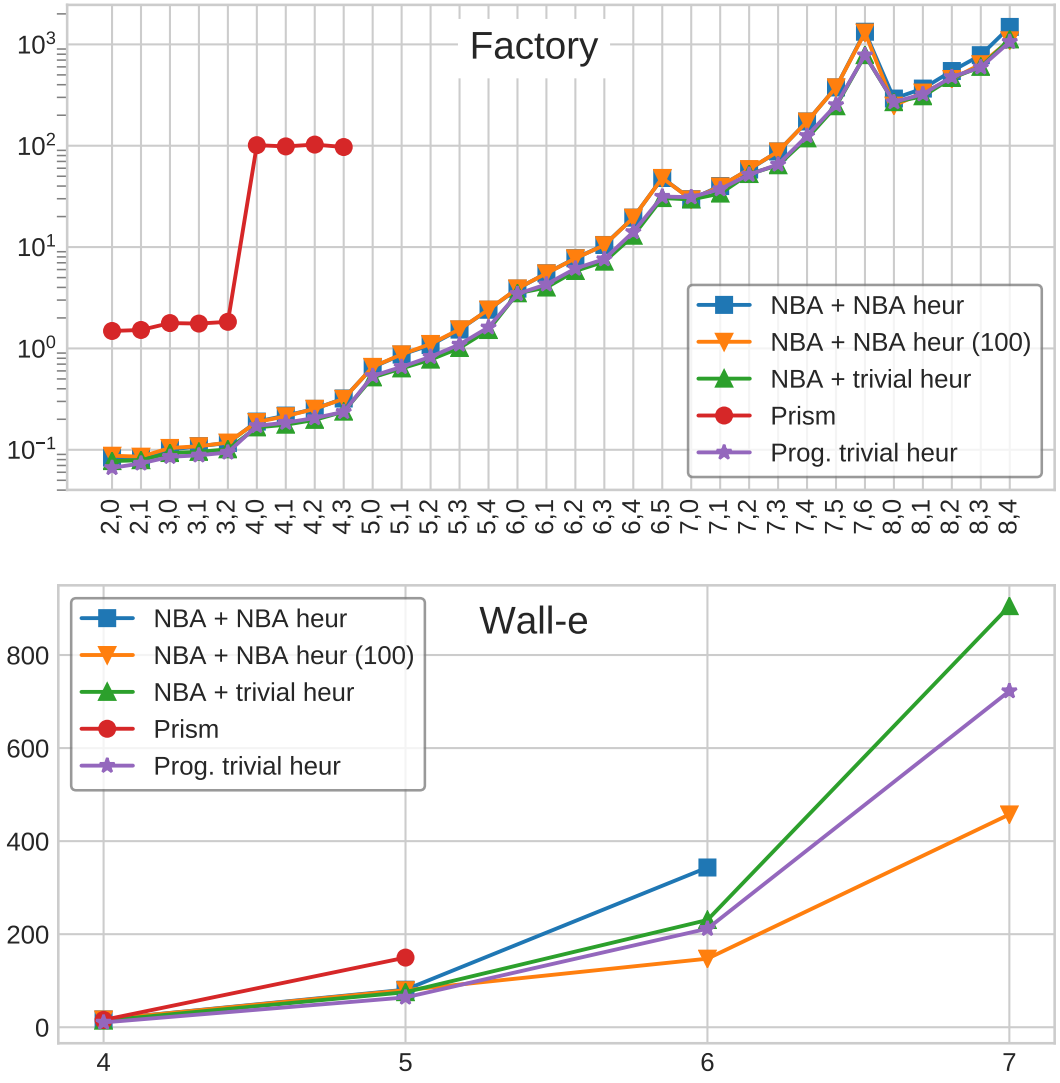


Figure 2: Time in seconds to solve: factory problems n, k ($n \in 2..8$, $k \in 0..(n-1)$); and Wall-e problem for $n \in 4..7$.

A Additional Concepts and Proofs

In this appendix we fill in formal details left open in the main part and provide proofs for the soundness and completeness results, Theorems 5 and 6.

A.1 Büchi Automaton Mode

Lemma 3 *Let $p \in S^+$ be a path with $p_1 = s_{\text{init}}$, ψ an LTL formula and $\mathcal{B}_{\psi^{IE}}$ as defined above* [*in the main part of the paper*]. *Then $p \models_{IE} \psi$ iff starting from some state in Q_{init} by following the states in p a state $q \in Q$ is reachable that is accepting with* $\text{last}(p)$.

Proof. $\Rightarrow$-direction: assume $p \models_{IE} \psi$, equivalently $p;(\text{last}(p)^\omega) \models \psi$. By correctness of the construction $\mathcal{B}_{\psi^{IE}}$ accepts $r = p;(\text{last}(p)^\omega)$. By definition of acceptance, the postfix $\text{last}(p)^\omega$ must visit a state from F infinitely often. Let q' be that state. Trivially, q' is contained in some strongly connected component *scc*. Again by acceptance the state q' is reachable from q by transitions with $\text{last}(p)$ only, proving (1). Because r is accepted, (2) follows. For (3) let $s = \text{last}(p)$ and assume, to the contrary, that *scc* has a transition different to s. This entails there is an accepting run visiting the states of *scc* but that infinitely often alternates between s and some other $s' \in S$. But then that run cannot satisfy the subformula $\mathbf{F}\,(\bigwedge_{a \in AP}(a \rightarrow \mathbf{G}\,a) \wedge (\neg a \rightarrow \mathbf{G}\,\neg a))$.

$\Leftarrow$-direction: assume $q \in Q$ is accepting with $\text{last}(p)$. With the definition of accepting state it is easy to see that $p;(\text{last}(p)^\omega)$ is an accepting run of $\mathcal{B}_{\psi^{IE}}$. It follows $p;(\text{last}(p)^\omega) \models \psi$ and hence $p \models_{IE} \psi$. □

A.2 Formula Progression

Our main data structure for formula progression is that of a set formula. Formally, a *set formula* is a set $\Gamma = \{\Psi_1, \ldots, \Psi_n\}$ of sets of (LTL) formulas. It represents a conjunction of disjunctions, i.e., it stands for the formula $\text{form}(\Gamma) := \bigwedge_{\Psi \in \Gamma} \bigvee_{\psi \in \Psi} \psi$. Any formula ψ can trivially be cast as an equivalent set formula by taking $\Gamma = \{\{\psi\}\}$. By a slight abuse of notation we call each $\Psi \in \Gamma$ a *clause (of Γ)* and each $\psi \in \Psi$ a *literal*.[5] We say that Ψ is a *Σ-clause*, or that *Ψ has domain Σ* iff $\Psi \subseteq \Sigma$. Similarly, we say that *Γ has domain Σ* iff every clause in Γ has domain Σ. We need this notion of domain below, where we show that Σ can be "small", polynomial in the size of ψ, and that clauses can be bound to contain at most three literals.

Standard formula progression as defined in [BK98] recursively descends into a given LTL formula ψ with respect to a current state s. Essentially, it evaluates atemporal subformulas to true or false with respect to the state s, and it expands temporal subformulas using (well-known) equivalences such as $\psi_1\,\mathbf{U}\,\psi_2 \equiv \psi_2 \vee (\psi_1 \wedge \mathbf{X}\,(\psi_1\,\mathbf{U}\,\psi_2))$. The recursion terminates at the logical constants true and false and at $\mathbf{X}$-formulas.

In contrast to [BK98] we are interested in a complete algorithm. This requires being able to detect loops as formula progression proceeds from state to state, which is trivial with the set formula representation. Such a progression algorithm can be based on naïve CNF (conjunctive normal form) transformation. It takes an initial LTL formula ψ and starting from $\{\{\psi\}\}$ converts it to a set formula representing the CNF of ψ by applying certain transformation rules as long as possible. The rules are a combination of the distributivity law for Boolean operators, the expansion law for the $\mathbf{U}$-operator, and rules for simplification by evaluation of atemporal subformulas (as in [BK98]). Associativity, commutativity an idempotence of Boolean operators is dealt with implicitly, by properties

[5] Usually, a literal is an atomic formula or the negation of an atomic formula.

of sets. The rule application stops at **X**-formulas. The resulting set formula will be stripped all its **X**-operators before it is passed on to the next state, where progression from that state continues. See Section A.2.3 for details.

While this algorithm works in principle, it has the serious drawback of exponentially increasing the given set-formula in the worst case. This is because the expansion law $(\psi_1 \wedge \psi_2) \vee \psi \equiv (\psi_1 \vee \psi) \wedge (\psi_2 \vee \psi)$, which is needed along the way to obtain a CNF, duplicates its subformula ψ. This phenomenon can be avoided by a "Tseitin-syle" CNF transformation which has been well-known in the automated theorem proving literature for a long time [Tse68]. See [AW13] for more recent developments.

Our progression algorithm below is essentially the rule-based algorithm outlined above, however with Tseitin-style CNF transformation supplanting naïve CNF transformation. The underlying idea is to define, or *name*, a complex subformula, such as $\psi_1 \wedge \psi_2$ in $(\psi_1 \wedge \psi_2) \vee \psi$, in terms of a fresh propositional variable $A_{\psi_1 \wedge \psi_2}$ that uniquely stands for ("names") that subformula. The formula $(\psi_1 \wedge \psi_2) \vee \psi$ then becomes the conjunction of $A_{\psi_1 \wedge \psi_2} \vee \psi$ and the "definitions" $\neg A_{\psi_1 \wedge \psi_2} \vee \psi_1$ and $\neg A_{\psi_1 \wedge \psi_2} \vee \psi_2$. Notice that this transformation eliminates the offending conjunction $(\psi_1 \wedge \psi_2)$ without duplicating subformulas. The Tseitin-style algorithm returns a formula that is equi-satisfiable with (but usually not equivalent to) the given formula and runs in time and space linear in the size of the input formula.

A.2.1 Tseitin-Style CNF transformation

In this section we define our Tseitin-Style CNF transformation, and in Section A.2.3 below we define progression on top of it. The overall goal is to make sure progression is tightly controlled: (1), that no "new" subformulas are ever derived, ones that are not already a subformula of the initial formula, (2) that the set of names ever needed is "small", polynomial in the size of the initial formula, and (3) that the length of clauses is suitably bound a priori, which is three in our case.

For a formula ψ, $\mathrm{sub}(\psi)$ denotes the set of all subformulas of ψ. Formally, $\mathrm{sub}(\top) = \{\top\}$, $\mathrm{sub}((v, d)) = \{(v, d)\}$, for $(v, d) \in AP$, $\mathrm{sub}(\mathbf{X}\,\psi) = \{\mathbf{X}\,\psi\} \cup \mathrm{sub}(\psi)$, $\mathrm{sub}(\neg\psi) = \{\neg\psi\} \cup \mathrm{sub}(\psi)$, and $\mathrm{sub}(\psi_1 \circ \psi_2) = \{\psi_1 \circ \psi_2\} \cup \mathrm{sub}(\psi_1) \cup \mathrm{sub}(\psi_2)$, for $\circ \in \{\wedge, \vee, \mathbf{U}\}$. Let $\overline{\cdot}$ be the complement operator, i.e., for negated formulas $\overline{\neg\psi} = \psi$ and otherwise $\overline{\psi} = \neg\psi$. Define

$$\mathrm{cl}'(\psi) = \mathrm{sub}(\psi) \cup \{\overline{\varphi},\ \mathbf{X}\,\varphi,\ \mathbf{X}\,\overline{\varphi},\ |\ \varphi \in \mathrm{sub}(\psi)\}, \text{ and} \qquad \text{(Closure of } \mathrm{sub}(\psi) \text{ under } \overline{\cdot} \text{ and } \mathbf{X})$$

$$\mathrm{cl}(\varphi) = \mathrm{cl}'(\psi) \cup \{\psi_1 \wedge \psi_2 \mid \psi_1, \psi_2 \in \mathrm{cl}'(\psi)\}\ . \qquad \text{(Closure of } \mathrm{cl}'(\psi) \text{ under conjunction)}$$

Let ψ be an LTL formula, the initially given constraint. Recall that $AP = \{(v, d) \mid v \in \mathcal{V}, d \in \mathsf{D}_v\}$ is a finite set of atoms and every state s is equipped with an evaluation function $[\cdot]$ so that $s[v] \in \mathsf{D}_v$. We need to extend AP to be able to name (sub)formulas of ψ. Define $\mathcal{V}_\psi = \mathcal{V} \cup \{v_\varphi \mid \varphi \in \mathrm{cl}(\psi)\}$ as the extension of the variables $\mathcal{V}$ by the designated variables v_φ for each $\varphi \in \mathrm{cl}(\psi)$. Each v_φ is Boolean, i.e., $\mathsf{D}_{v_\varphi} = \{\mathsf{true}, \mathsf{false}\}$. We commonly write A_φ as a shorthand for the pair $(v_\varphi, \mathsf{true})$ and say that A_φ is the *name of* φ. Define

$$N(\psi) = \{A_\varphi \mid \varphi \in \mathrm{cl}(\psi)\}, \qquad \text{(Names for closure of } \psi)$$

$$\Sigma(\psi) = N(\psi) \cup \{\neg A_\varphi \mid A_\varphi \in N(\psi)\} \cup \mathrm{cl}(\psi). \qquad \text{(Domain obtained from } \psi)$$

The intuition is that a name A_φ is defined to be true in a state p_i of a given run p if and only if $p(i) \models \varphi$. See Section A.2.3 below for details.

Progression is formalized with the help of the relation $\Rightarrow_s$ between set formulas as follows:

$$\{\{\}\} \uplus \Gamma \Rightarrow_s \{\{\}\} \quad \text{if } \Gamma \neq \emptyset \tag{Triv}$$

$$\{\{\top\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \Gamma \tag{$\top$}$$

$$\{\{\neg\top\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\Psi\} \cup \Gamma \tag{$\neg\top$}$$

$$\{\{(v, d)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \Gamma \qquad \text{if } (v, d) \in AP \text{ and } s[v] = d \tag{Eval1}$$

$$\{\{(v, d)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\Psi\} \cup \Gamma \qquad \text{if } (v, d) \in AP \text{ and } s[v] \neq d \tag{Eval2}$$

$$\{\{\neg(v, d)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\Psi\} \cup \Gamma \qquad \text{if } (v, d) \in AP \text{ and } s[v] = d \tag{Eval3}$$

$$\{\{\neg(v, d)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \Gamma \qquad \text{if } (v, d) \in AP \text{ and } s[v] \neq d \tag{Eval4}$$

$$\{\{\neg\neg\psi\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{\psi\} \cup \Psi\} \cup \Gamma \tag{$\neg\neg$}$$

$$\{\{\psi_1 \vee \psi_2\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{A_{\psi_1\vee\psi_2}\} \cup \Psi, \{\neg A_{\psi_1\vee\psi_2}, \psi_1, \psi_2\}\} \cup \Gamma \tag{$\vee$}$$

$$\{\{\neg(\psi_1 \vee \psi_2)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{\neg A_{\psi_1\vee\psi_2}\} \cup \Psi, \{A_{\psi_1\vee\psi_2}, \overline{\psi_1}\}, \{A_{\psi_1\vee\psi_2}, \overline{\psi_2}\}\} \cup \Gamma \tag{$\neg\vee$}$$

$$\{\{\psi_1 \wedge \psi_2\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{A_{\psi_1\wedge\psi_2}\} \cup \Psi, \{\neg A_{\psi_1\wedge\psi_2}, \psi_1\}, \{\neg A_{\psi_1\wedge\psi_2}, \psi_2\}\} \cup \Gamma \tag{$\wedge$}$$

$$\{\{\neg(\psi_1 \wedge \psi_2)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{\neg A_{\psi_1\wedge\psi_2}\} \cup \Psi, \{A_{\psi_1\wedge\psi_2}, \overline{\psi_1}, \overline{\psi_2}\}\} \cup \Gamma \tag{$\neg\wedge$}$$

$$\{\{\psi_1 \mathbf{U} \psi_2\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{A_{\psi_1 \mathbf{U} \psi_2}\} \cup \Psi, \{\neg A_{\psi_1 \mathbf{U} \psi_2}, \psi_2, A_{\psi_1\wedge\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}\}, \{\neg A_{\psi_1\wedge\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}, \psi_1\}, \{\neg A_{\psi_1\wedge\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}, \mathbf{X}(\psi_1 \mathbf{U} \psi_2)\}\} \cup \Gamma \tag{$\mathbf{U}$}$$

$$\{\{\neg(\psi_1 \mathbf{U} \psi_2)\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{\neg A_{\psi_1 \mathbf{U} \psi_2}\} \cup \Psi, \{A_{\psi_1 \mathbf{U} \psi_2}, \overline{\psi_2}\}, \{A_{\psi_1 \mathbf{U} \psi_2}, \neg A_{\psi_1\wedge\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}\}, \{A_{\psi_1\wedge\mathbf{X}(\psi_1 \mathbf{U} \psi_2)}, \overline{\Psi_1}, \mathbf{X}\neg(\psi_1 \mathbf{U} \psi_2)\}\} \cup \Gamma \tag{$\neg\mathbf{U}$}$$

$$\{\{\neg\mathbf{X}\psi\} \uplus \Psi\} \uplus \Gamma \Rightarrow_s \{\{\mathbf{X}\overline{\psi}\} \cup \Psi\} \cup \Gamma \tag{$\neg\mathbf{X}$}$$

The singled-out literal in the left-hand side of the rule is called the *pivot*.

In the definition of $\Rightarrow_s$ above we left its domain unspecified. The following lemma is a first step and helps by clarifying that $\Rightarrow_s$ preserves the domain $\Sigma(\psi)$.

Lemma 9 *Let Γ_1 be a set formula with domain $\Sigma(\psi)$, for some LTL formula ψ. If $\Gamma_1 \Rightarrow_s \Gamma_2$ then Γ_2 has domain $\Sigma(\psi)$.*

Proof. Assume $\Gamma_1 \Rightarrow_s \Gamma_2$. We need to check all rules in $\Rightarrow_s$ and argue in each case that every clause in Γ_2 has domain $\Sigma(\psi)$. In doing that, we argue with closure properties such as "$\Sigma(\Psi)$ is closed under subterms" by which we mean the fact that $\mathrm{sub}(\psi) \subseteq \Sigma(\Psi)$.

For the rules (Triv)–(Eval4) the proof is trivial, as they either remove clauses or remove literals from clauses. For the rule ($\neg\neg$) use the fact that $\Sigma(\psi)$ is closed under subformulas. For the rules

$(\vee)$–$(\neg\wedge)$ use the fact that $\Sigma(\psi)$ is closed under subformulas, naming subformulas, and complement of subformulas. An important detail here is that no rule can ever be applied to a pivot containing a name from $N(\Psi)$. This is because names or their negation are introduced in top-level positions of clauses *only*, i.e., as literals, and they are never a pivot. It is, hence, enough to include in $\Sigma(\Psi)$ names in a non-recursive way, as defined.

The argumentation is similar for the rules ($\mathbf{U}$) and ($\neg\mathbf{U}$). This works because $\Sigma(\psi)$ is additionally closed under prefixing subformulas with $\mathbf{X}$ and forming conjunctions. Notice these rules differ from the other rules in that instead of decomposing a formula they add an operator $\mathbf{X}$ in front of it. However, they can be applied only to non-$\mathbf{X}$ formulas, which is why closing $\Sigma(\psi)$ under one-time prefixing with $\mathbf{X}$ is enough. The case of the ($\neg\mathbf{X}$) follows again from $\Sigma(\psi)$ being closed under subformulas and complementation. □

Lemma 10 (Termination of $\Rightarrow_I$) *The rule system $\Rightarrow_s$ is terminating. That is, there is no infinite sequence $\Gamma_1 \Rightarrow_s \Gamma_2 \Rightarrow_s \Gamma_3 \cdots$, for no set formula Γ_1 and no state s.*

Proof. Consider any sequence $\Gamma_1 \Rightarrow_s \Gamma_2 \Rightarrow_s \Gamma_3 \cdots$. We show it must be terminating. We can ignore the (Triv) rule as it obviously terminates the sequence. Each of the rules $(\top)$ – $(\neg\wedge)$ replaces some clause $\Psi \in \Gamma_i$ by some clauses $\Psi_1, \ldots, \Psi_k \in \Gamma_{i+1}$. The clauses Ψ_j are either proper subsets of Ψ, or a formula in Ψ is replaced by a proper subformula (the $(\neg\neg)$ rule), or a (possibly negated) formula $\psi_1 \circ \psi_2$ where $\circ \in \{\vee, \wedge\}$ is removed in terms of names from $\mathrm{cl}^+(\psi)$ and introducing definitions for these names, all possibly negated. Notice that the formulas ψ_1 and ψ_2 are moved into different clauses, modulo negation, but not duplicated. Clearly, hence, the rule system $(\top)$ – $(\neg\wedge)$ is terminating. Formally, it is straightforward to define a well-wounded ordering that shows that theses rules work in a strictly order-decreasing way.

The argumentation extends to the rules ($\mathbf{U}$) and ($\neg\mathbf{U}$). If ($\mathbf{U}$) is applied to $\psi_1 \,\mathbf{U}\, \psi_2$, which removes it, the result contains the formulas $\mathbf{X}(\psi_1 \,\mathbf{U}\, \psi_2)$, ψ_1 and ψ_2. The first formula can be ignored because its $\mathbf{X}$-operator shields it from further expansion. The formulas ψ_1 and ψ_2 are again both proper subformulas of $\psi_1 \,\mathbf{U}\, \psi_2$. Similarly for the ($\neg\mathbf{U}$) rule. □

Lemma 10 justifies to talk about normal forms. Let $\Rightarrow_s^*$ denote the reflexive-transitive closure of $\Rightarrow_s$. If Γ is a set formula we write $\Delta = \mathrm{CNF}(\Gamma, s)$ iff $\Gamma \Rightarrow_s^* \Delta$ and no rule of $\Rightarrow_s$ is applicable to Δ.[6] For an LTL formula ψ let $\mathrm{CNF}(\psi, s) = \mathrm{CNF}(\{\{\psi\}\}, s)$.

We say that a set formula Γ *is in 3-CNF* iff $|\Psi| \leq 3$ for every clause $\Psi \in \Gamma$.

Lemma 11 (Basic facts about $\Rightarrow_s$) *Let Γ be a 3-CNF set formula with domain $\Sigma(\psi)$, for some LTL formula Ψ. If $\Delta = \mathrm{CNF}(\Gamma, s)$ then Δ is a 3-CNF set formula with domain $\Sigma(\psi)$.*

Proof. Suppose $\Delta = \mathrm{CNF}(\Gamma, s)$. The domain preservance claim follows immediately from Lemma 9. That Δ is in 3-CNF follows from the fact that Γ is in 3-CNF and that the rule system $\Rightarrow_s$ either removes clauses, shrinks clauses or adds clauses with length at most 3. □

A.2.2 Complexity

As usual, the *size of ψ*, denoted as $|\psi|$, is the number of operators occurring in ψ. Although quite obvious, we show that the size of $\Sigma(\psi)$ is polynomial in the size of ψ.

Lemma 12 $|\Sigma(\psi)| \in O(|\psi|^2)$.

[6] This is a slight abuse of notation, as we have not shown that $\Rightarrow_s^*$ is confluent.

Proof. The cardinality of the set $\mathrm{sub}(\psi)$ is bound by $|\mathrm{sub}(\psi)| \leq 2|\psi|$. This (well-known) fact is easily shown by induction on the formula structure. Obviously, then, $|\mathrm{cl}'(\psi)| \leq 8|\psi|$ and $|\mathrm{cl}(\psi)| \leq 8|\psi| + 8|\psi| \cdot 8|\psi| = 8|\psi| + 64|\psi|^2 \in O(|\psi|^2)$. It follows $|\Sigma(\psi)| \leq 2|N(\psi)| + |\mathrm{cl}(\psi)| = 2|\mathrm{cl}(\psi)| + |\mathrm{cl}(\psi)| \in O(|\psi|^2)$. □

Lemma 13 (Complexity of CNF Transformation) *Let Γ be a 3-CNF set formula Γ with domain $\Sigma(\psi)$ and $\Delta = \mathrm{CNF}(\Gamma, s)$. Then the following holds*:

1. *Δ is derived with $O(|\psi|)$ rule applications.*
2. *$|\Delta| \in O(|\Sigma(\psi)|^3)$ $(= O(|\psi|^6))$.*

Proof. (1) The number of rule applications to obtain Δ from Γ is bound linearly by the number of subformulas occurring in Γ that are also contained in $\mathrm{sub}(\psi)$, which is of size $O(|\psi|)$. To see this, observe that the rule system $\Rightarrow_s$ does not duplicate subformulas and the pivot is always from $\mathrm{sub}(\psi)$. In the worst case every removal of a negated formula made with a binary operator $\circ \in \{\mathbf{U}, \wedge, \vee\}$, of which there can be only linearly many wrt. $|\psi|$, leads to two complemented immediate subformulas of that formula. That is, the number of operators and subformulas shrinks or remains the same. In the latter case a binary operator has been treated in for a negation operator. The negated formulas may need another sweep of rule applications, but again at most linearly many wrt. $|\psi|$.

(2) From Lemma 11 we know that Δ is in 3-CNF and has domain $\Sigma(\psi)$. Every clause $\Psi \in \Delta$, hence, has at most three literals, each taken from $\Sigma(\psi)$. There are $O(|\Sigma(\psi)|^3)$ different clauses with at most 3 literals and domain $\Sigma(\psi)$, which bounds, in particular the size of Δ to $O(|\Sigma(\psi)|^3)$. With Lemma 12 it follows $O(|\Sigma(\psi)|^3) = O(|\psi|^6)$. □

With Lemma 13 we see that $\Delta = \mathrm{CNF}(\Gamma, s)$ can be obtained in polynomial space and time wrt. $|\psi|$.[7] As said in the proof of (2), there are $O(|\Sigma(\psi)|^3) = O(|\psi|^6)$ different clauses with at most 3 literals and domain $\Sigma(\psi)$. There are, hence, $O(2^{|\psi|^6})$ different 3-CNF set formulas. This will allow us to prove that our synthesis algorithm has a time and space complexity of the same order, a markedly improved result over the double-exponential complexity of the established algorithms.

A.2.3 Progression of CNF Formulas

It is not difficult to see that a set formula $\Delta = \mathrm{CNF}(\varphi, s)$ is either the set $\{\{\}\}$, which stands for "false", or for every $\Psi \in \Delta$ and every $\psi \in \Psi$, the formula ψ is an $\mathbf{X}$-formula. The second case includes the possibility that $\Delta = \{\}$, which stands for "true". Let us call set formulas of this form, where every literal is an $\mathbf{X}$-formula, *expanded*.

Define the operator $\mathrm{un}\mathbf{X}$ on expanded set formulas for stripping off the outermost $\mathbf{X}$-operators as follows: $\mathrm{un}\mathbf{X}(\{\{\}\}) = \{\{\}\}$, and $\mathrm{un}\mathbf{X}(\{\Psi_1, \ldots, \Psi_n\}) = \{\mathrm{un}\mathbf{X}(\Psi_1), \ldots, \mathrm{un}\mathbf{X}(\Psi_n)\}$, where $\mathrm{un}\mathbf{X}(\{\mathbf{X}\,\psi_1, \ldots, \mathbf{X}\,\psi_m\}) = \{\psi_1, \ldots, \psi_m\}$, where $n, m \geq 1$.

Notice that an expanded set formula Δ is equivalent to $\mathbf{X}\,\mathrm{un}\mathbf{X}(\Delta)$ thanks to the equivalences $\mathbf{X}\,(\psi_1 \vee \psi_2) \equiv (\mathbf{X}\,\psi_1) \vee (\mathbf{X}\,\psi_2)$ and $\mathbf{X}\,(\psi_1 \wedge \psi_2) \equiv (\mathbf{X}\,\psi_1) \wedge (\mathbf{X}\,\psi_2)$.

In other words, $\mathrm{un}\mathbf{X}(\Delta)$ takes Δ to the next state. By *progression* we mean the combined application of $\mathrm{un}\mathbf{X}$ and CNF, in this order. More precisely:

Definition 14 (Progression) *Let ψ be an LTL formula and p be a path. The* progression of ψ along

[7] The set operations can be done in time linear wrt. $|\Sigma(\psi)|$.

p is the sequence $\mathrm{prog}(\psi, p) := (\Gamma_i)_{i\geq 1}$, *where*

$$\Gamma_1 = \mathrm{CNF}(\{\{\psi\}\}, p_1),$$
$$\Gamma_2 = \mathrm{CNF}(\mathrm{un}\mathbf{X}(\Gamma_1), p_2),$$
$$\Gamma_3 = \mathrm{CNF}(\mathrm{un}\mathbf{X}(\Gamma_2), p_3), \ldots$$

If p is finite, then so is $\mathrm{prog}(\psi, p)$.

Lemma 15 *Let* $\mathrm{prog}(\psi, p) = (\Gamma_i)$ *be the progression of* ψ *along p. Then every* Γ_i *is a 3-CNF set formula with domain* $\Sigma(\psi)$.

Proof. The initial set $\{\{\psi\}\}$ is trivially a set formula with domain $\Sigma(\psi)$. By Lemma 9, the CNF-operator preserves the domain $\Sigma(\psi)$. Trivially, the un$\mathbf{X}$-operator also preserves the domain $\Sigma(\psi)$. Hence, every Γ_i has the domain $\Sigma(\psi)$. (This shows that progression is well-defined.) Moreover, with Lemma 11 every Γ_i is in 3-CNF. □

The semantics of LTL provides evaluation relations $p \models \psi$ and $p \models_{IE} \psi$ for LTL formulas ψ and infinite or finite paths p, respectively. We cannot use these relations on the progression $\mathrm{prog}(\psi, p)$ because our Tseitin-style CNF transformation requires on-the-fly introduction of the new state variables $\mathcal{V}_\psi \setminus \mathcal{V}$, which are undefined in the states of p.

This problem is solved by making the states p_i of p total on $\mathcal{V}_\psi$ in a unique way and that preserves the semantics of LTL formulas. The resulting states $(p_i)_\psi$ are defined in such a way that $(p_i)_\psi \models A_\varphi$ iff $p(i) \models \varphi$. That is, A_φ is true in a state p_i iff the path starting from p_i satisfies φ. Formally:

Definition 16 *Let* ψ *be an LTL formula and p be a path. For all* $i \geq 1$ *define the* extension $(p_i)_\psi$ of p_i to $\mathcal{V}_\psi$, *as*

$$(p_i)_\psi = \mathit{res}(p_i, e_i), \text{ where } e_i \text{ is such that for every } \varphi \in \mathrm{cl}(\psi),$$
$$e_i[v_\varphi] = \begin{cases} \mathsf{true} & \text{if } p(i) \models \varphi \\ \mathsf{false} & \text{otherwise.} \end{cases}$$

The extension p_ψ of p to $\mathcal{V}_\psi$ *is obtained from p by replacing each state* p_i *by* $(p_i)_\psi$, *i.e.,* $(p_\psi)_i = (p_i)_\psi$, *for all* $i \geq 1$.

For example, suppose $\psi = B \vee \mathbf{F}\, C$ and $p \models \psi$. Then we have $(p_1)_\psi[v_\psi] = \mathsf{true}$, as $p(1) = p$ and $p \models \psi$. In other words $(p_1)_\psi \models A_\psi$, as A_ψ is a shorthand for $v_\psi = \mathsf{true}$. This is consistent with the result of the CNF transformation of ψ, which is, as a formula, $\gamma = A_\psi \wedge (\neg A_\psi \vee (B \vee \mathbf{F}\, C))$. Indeed, $p_\psi \models \gamma$ as $p_\psi \models A_\psi$ and $p_\psi \models (\neg A_\psi \vee (B \vee \mathbf{F}\, C))$. The former holds because A_ψ is classical and $(p_\psi)_i \models A_\psi$. The latter is equivalent to $p_\psi \models \psi$ which is equivalent to $p \models \psi$, as p_ψ is the same as p on formulas over the variables $\mathcal{V}$. If instead $p \not\models \psi$ then we would have $p_\psi \not\models A_\psi$ and hence $p_\psi \not\models \gamma$, as desired. We are going to formalize this correspondence.

Proposition 17 *Let* ψ *be an LTL formula, p be a path and* (Γ_i) *the progression of* ψ *along p. Then,*

1. $p \models \psi$ *iff* $p_\psi \models \Gamma_1$, *and*
2. $p(i) \models \Gamma_i$ *iff* $p(i+1) \models \Gamma_{i+1}$.

Proof. (Sketch.) Item (1) requires induction over the formula structure and generalizes the argumentation given in the example above. Item (2) is proven by induction on i. The induction step uses the well-known equivalences

$$\psi_1 \mathbf{U}\, \psi_2 \equiv \psi_2 \vee (\psi_1 \wedge \mathbf{X}\,(\psi_1 \mathbf{U}\, \psi_2)) \qquad \neg(\psi_1 \mathbf{U}\, \psi_2) \equiv \neg\psi_2 \wedge (\neg\psi_1 \vee \mathbf{X}\,\neg(\psi_1 \mathbf{U}\, \psi_2))$$
$$\mathbf{X}\,(\psi_1 \vee \psi_2) \equiv (\mathbf{X}\,\psi_1) \vee (\mathbf{X}\,\psi_2) \qquad \mathbf{X}\,(\psi_1 \wedge \psi_2) \equiv (\mathbf{X}\,\psi_1) \wedge (\mathbf{X}\,\psi_2)$$
$$\neg\mathbf{X}\,\psi \equiv \mathbf{X}\,\neg\psi$$

and argues with the logical structure of the definitions of subformulas by means of names for them. This argumentation generalizes the example above. This is straightforward, but somewhat tedious. □

Proposition 17 states in (1) that the evaluation of an (initial) LTL formula ψ is preserved under CNF transformation by using extended states, and vice versa. Item (2), applied inductively when starting with (1) shows that our progression algorithm is correct: $p \models \psi$ holds iff $p_\psi(j) \models \Gamma_j$ at any time j. In particular, if p_j is a goal state then $p_\psi(j) \models \Gamma_j$ can be evaluated in a straightforward way and in polynomial time, with a minor adaption to the "idle" algorithm in [BK98].

A.3 Formal Concepts Related to SSPs

Throughout this and the following section, let $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ be a given SSP, π an unrestricted policy and $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ a MO-PLTL constraint. We repeat and extend some basic definitions from the main part of the paper.

A *path (of $\mathcal{S}$) from s* is a sequence of states $s_1 s_2 \cdots \in S^+ \cup S^\infty$ such that $s_1 = s$. Given $s \in S$, a *run from s (of $\mathcal{S}$)* is a finite or infinite sequence $r = (s = s_1) \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} s_3 \cdots$ of states $s_i \in S$ and actions $\alpha_i \in A(s_i)$ such that $P(s_{i+1}|s_i, \alpha_i) > 0$, for all $i \geq 1$. The *cost* and *probability* of a run r are defined as, respectively,

$$C(r) = \sum_{i=1} C(\alpha_i), \text{ and}$$
$$P(r) = \prod_{i \geq 1} P(s_{i+1}|s_i, \alpha_i) \ .$$

A run r is an *exhaustive run of π from s*, or a *π-run from s* , if $\pi(s_1 \cdots s_i, \alpha_i) > 0$ for all $i \geq 1$ and either r is infinite or π is not defined for the finite path represented by r. Let *Runs(s, π)* denote the set of all π-runs from s, and *GRuns(s, π)* $\subseteq$ *Runs*(s, π) the set of all finite π-runs from s that end in a goal state, or just *goal runs from s*.

Given $r \in \text{Runs}(s, \pi)$, the *probability of r being produced by π* is

$$P(r|\pi) = P(r) \prod_{i \geq 1} \pi(s_1 \cdots s_i, \alpha_i)$$
$$\left(= \prod_{i \geq 1} \pi(s_1 \cdots s_i, \alpha_i)\, P(s_{i+1}|s_i, \alpha_i) \right)$$

A policy π is *proper* if $\sum_{r \in \text{GRuns}(s_{\text{init}}, \pi)} P(r|\pi) = 1$, that is, if the probability of reaching the goal when following π from s_{init} is 1.

Given a proper policy π, its *total expected costs* are defined as follows:

$$V^\pi(s) = \sum_{r \in GRuns(s,\pi)} (C(r) + T(\text{last}(r)))\, P(r|\pi).$$

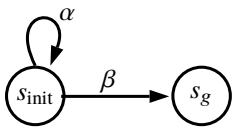
For simplicity, we abbreviate $V^\pi(s_{\text{init}})$ with V^π.

Note 18 (Infinite costs) The definition of V^π ignores non-goal runs, that is, ignores costs of infinite runs. Had we instead defined

$$W^\pi(s) = \sum_{r \in Runs(s,\pi)} (C(r) + U(r))\, P(r|\pi), \text{ where}$$
$$U(r) = \begin{cases} T(r) & \text{if } r \text{ is finite and last}(r) \in G \\ 0 & \text{otherwise} \end{cases}$$
$$W^\pi = W^\pi(s_{\text{init}})$$

then this leads to the problem that there are proper policies with even infinite costs. In other words, obviously, $V^\pi \leq W^\pi$ always holds, but $V^\pi = W^\pi$ does in general not hold, even if π is proper.

As an example take the SSP $\mathcal{S} = (\{s_{\text{init}}, s_g\}, s_{\text{init}}, \{s_g\}, \{\alpha, \beta\}, C, T)$ where $C(\alpha) = C(\beta) = 1$ and $T(s_g) = 0$. The actions α and β are defined as follows:



The (unrestricted) policy π is defined on every path $s^i_{\text{init}} = s_{\text{init}} \cdots s_{\text{init}}$ (i.e., i repetitions of s_{init}) as

$$\pi(s^i_{\text{init}}, \alpha) = \frac{\text{ld}(i+1)}{\text{ld}(i+2}$$
$$\pi(s^i_{\text{init}}, \beta) = 1 - \pi(s^i_{\text{init}}, \alpha)$$

The set $Runs(s, \pi) \setminus GRuns(s, \pi)$ is a singleton consisting of the run $r = s_{\text{init}} \xrightarrow{\alpha} s_{\text{init}} \xrightarrow{\alpha} s_{\text{init}} \cdots$. The probability of r being produced by π is

$$P(r, \pi) = \frac{1}{\text{ld}\,3} \cdot \frac{\text{ld}\,3}{\text{ld}\,4} \cdot \frac{\text{ld}\,4}{\text{ld}\,5} \cdots \frac{\text{ld}(n-1)}{\text{ld}\,n} \cdots$$
$$= \lim_{n\to\infty} \frac{1}{\text{ld}\,n} = 0 \ .$$

The expected costs of any prefix s^n_{init} of r is $C(s^n_{\text{init}}) \frac{1}{\text{ld}\,n} = \frac{n-1}{\text{ld}\,n}$, which converges to infinity. Thus $C(r)\,P(r|\pi) = \lim_{n\to\infty}(C(s^n_{\text{init}}) \frac{1}{\text{ld}\,n})_n = \infty$. As π is proper and $r \in Runs(s, \pi)$ It follows $V^\pi = \infty$. But notice that r, being infinite, cannot be a goal run and hence does not contribute to W^π. □

A.4 Formal Concepts Related to Markov Chains

Let $\mathcal{S}^\pi = (S^+, s_{\text{init}}, A, P_{\mathcal{S}^\pi})$ be the (infinite-state) Markov chain induced by $\mathcal{S}$ and π in the standard way. The construction "compiles away" $\mathcal{S}$'s actions into the state transitions of $\mathcal{S}^\pi$. Accordingly, it is possible there are different π-runs of $\mathcal{S}$ with transitions, say, $s_i \xrightarrow{\alpha_i} s_{i+1}$ and $s_i \xrightarrow{\beta_i} s_{i+1}$ that are collapsed in $\mathcal{S}^\pi$ into the single transition $(s_1 \cdots s_i)(s_1 \cdots s_i s_{i+1})$. The transition probability function $P_{\mathcal{S}^\pi}$ of $\mathcal{S}^\pi$ hence is defined as

$$P_{\mathcal{S}^\pi}(ps|p) = \sum_{\alpha \in A(\text{last}(p))} \pi(p, \alpha)\, P(s|\text{last}(p), \alpha) \text{ , for all } p \in S^+ \text{ and } s \in S,$$

and $P_{\mathcal{S}^\pi}(q|p) = 0$ in all other cases.

The only non-standard aspect is with the runs of $\mathcal{S}$, even if exhausted, that end in goal states and that are always finite (as no action is enabled at goal states). This is a minor techical issue, though, which we deal with by corresponding finite runs in $\mathcal{S}^\pi$. Thus, a *run (of $\mathcal{S}^\pi$) from s* is a sequence $r = (s_1)(s_1s_2)\cdots(s_1s_2\cdots s_n)\cdots$ of finite paths (elements from S^+) such that $s_1 = s$ and $P_{\mathcal{S}^\pi}(s_1\cdots s_is_{i+1}|s_1\cdots s_i) > 0$, for all $i \geq 1$. It can be finite or infinite.

We use $\Pr_{\mathcal{S}^\pi}$ for the probability measure on runs of $\mathcal{S}^\pi$. Let $r = (s_1)(s_1s_2)\cdots(s_1s_2\cdots s_n)$ be a finite run of $\mathcal{S}^\pi$. Then $\Pr_{\mathcal{S}^\pi}(r)$, the probability of the cylinder set spanned by r, is the same as the probability of all runs of $\mathcal{S}$ from s_1 to s_n (this follows immediately from the definition of $P_{\mathcal{S}^\pi}$):

$$\begin{aligned}\Pr_{\mathcal{S}^\pi}(r) &= \prod_{i=1}^{n-1} P_{\mathcal{S}^\pi}(s_1\cdots s_{i+1}|s_1\cdots s_i)\\ &= \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} P(s_1 \xrightarrow{\alpha_1} s_2\cdots s_{n-1}\xrightarrow{\alpha_{n-1}} s_n|\pi)\end{aligned}$$

(The equality follows from the definitions above.)

We can capture the costs of sets of policy runs, those that are collapsed by the Markov Chain under the same finite run $r = (s_1)(s_1s_2)\cdots(s_1s_2\cdots s_n)$ by defining:

$$\mathrm{ExpTot}_{\mathcal{S}^\pi}(r) = \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} C(q)\,P(q|\pi), \text{ where } q = s_1 \xrightarrow{\alpha_1} s_2\cdots s_{n-1}\xrightarrow{\alpha_{n-1}} s_n \ .$$

Similarly, we can capture the terminal costs of sets of policy runs reaching a goal state. Let $r = (s_1)(s_1s_2)\cdots(s_1s_2\cdots s_n)$ as above with additionally $s_n \in G$ and define

$$\begin{aligned}\mathrm{ExpT}_{\mathcal{S}^\pi}(r) &= T(\mathrm{last}(r))\,Pr_{\mathcal{S}^\pi}(r)\\ &= \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} T(s_n)\,P(s_1 \xrightarrow{\alpha_1} s_2\cdots s_{n-1}\xrightarrow{\alpha_{n-1}} s_n|\pi)\end{aligned}$$

(The equality follows from the definitions above.)

Because every finite run $r = (s_1)(s_1s_2)\cdots(s_1s_2\cdots s_n)$ of $\mathcal{S}^\pi$ is uniquely identified by its last state $s_1s_2\cdots s_n$ we can use that state when a run is required. For instance, we write $\Pr_{\mathcal{S}^\pi}(s_1s_2\cdots s_n)$ and mean $\Pr_{\mathcal{S}^\pi}(r)$; similarly for $\mathrm{ExpTot}_{\mathcal{S}^\pi}(s_1s_2\cdots s_n)$ and $\mathrm{ExpT}_{\mathcal{S}^\pi}(s_1s_2\cdots s_n)$.

Paths and Runs. Let $Paths = \{p \in S^+ \mid p_1 = s_{\mathrm{init}} \text{ and } \Pr_{\mathcal{S}^\pi}(p) > 0\}$ be the set of all non-0 probability, finite paths of $\mathcal{S}$ from s_{init}. We introduce some handy notation for describing specific subsets of *Paths*. In that, we allow states $s \in S$ and sets of states $T \subseteq S$ as arguments of $\mathbf{F}$-formulas:

$$\begin{aligned}Paths(\psi) &= \{p \in Paths \mid \mathrm{last}(p) \in G \text{ and } p \models_{IE} \psi\}\\ Paths(\mathbf{F}\,s) &= \{p \in Paths \mid \mathrm{last}(p) = s, \text{ and } p_i \neq s \text{ for all } i = 1..|p|-1\}\\ Paths(\mathbf{F}\,T) &= \cup_{s\in T}\, Paths(\mathbf{F}\,s)\end{aligned}$$

For any set of paths $\mathcal{P} \subseteq Paths$ define

$$\begin{aligned}\Pr^\pi_{\mathcal{S}}(\mathcal{P}) &= \Pr_{\mathcal{S}^\pi}(\mathcal{P})\ , \text{where}\\ \Pr_{\mathcal{S}^\pi}(\mathcal{P}) &= \sum_{p\in\mathcal{P}} \Pr_{\mathcal{S}^\pi}(p)\ .\end{aligned}$$

The expression $\Pr_{\mathcal{S}^\pi}(p)$ is well-defined as the path p stands for the run $(p_1)(p_1p_2)\cdots p$ of $\mathcal{S}^\pi$ from p_1 to p.

As a convenience we write $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$ as a shorthand for $\Pr_{\mathcal{S}^\pi}(\mathit{Paths}(\mathbf{F}\, s))$; similarly, $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, T)$ means $\Pr_{\mathcal{S}^\pi}(\mathit{Paths}(\mathbf{F}\, T))$, and $\Pr_{\mathcal{S}^\pi}(\psi)$ means $\Pr_{\mathcal{S}^\pi}(\mathit{Paths}(\psi))$.

In the main part of the paper we defined $Pr^\pi_{\mathcal{S}}(\psi) = \sum_{r \in \mathrm{GRuns}(s_{\mathrm{init}},\pi) \text{ s.t. } r \models_{IE} \psi} P(r|\pi)$ as the probability mass of the execution paths of π that statisfy LTL formula ψ. In fact, that definition is equivalent to expanding the definitions above. (Notice that $\Pr^\pi_{\mathcal{S}}(p) = 0$ if $p \in S^+$, $p_1 = s_{\mathrm{init}}$ and $\mathrm{last}(p) \in G$ but $p \notin \mathit{Paths}$.)

Lemma 19 *A policy π for $\mathcal{S}$ is proper iff* $\Pr^\pi_{\mathcal{S}}(\mathsf{true}) = 1$ *iff* $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, G) = 1$.

Proof. Straightforward from the definitions. □

For $p \in \mathit{Paths}$ define $\mathit{Succ}(p) = \{s \in S \mid P_{\mathcal{S}^\pi}(ps|p) > 0\}$ as the successor states of $\mathrm{last}(p)$ that can be reached by executing π. In the proof of Lemma 23 below we need to compute the costs of extending p by all states $\mathit{Succ}(p)$. The following lemma is helpful for that:

Lemma 20 *For every $p = s_1 \cdots s_n \in \mathit{Paths}$,*

$$\sum_{s \in \mathit{Succ}(p)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(ps) = \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \Pr_{\mathcal{S}^\pi}(p) \sum_{\alpha \in A(s_n)} C(\alpha)\, \pi(p|\alpha)$$

Proof. Abbreviate $q := s_1 \xrightarrow{\alpha_1} s_2 \cdots s_{n-1} \xrightarrow{\alpha_{n-1}} s_n$ and $\boldsymbol{\alpha}_{n-1} := \alpha_1 \in A(s_1), \ldots, \alpha_{n-1} \in A(s_{n-1})$. Compute

$$\begin{aligned}
&\sum_{s \in \mathit{Succ}(p)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(ps) \\
&\quad= \sum_{s \in \mathit{Succ}(p)} \sum_{\boldsymbol{\alpha}_{n-1}, \alpha \in A(s_n)} C(q \xrightarrow{\alpha} s)\, P(q \xrightarrow{\alpha} s|\pi) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}, \alpha \in A(s_n), s \in \mathit{Succ}(p)} C(q \xrightarrow{\alpha} s)\, P(q \xrightarrow{\alpha} s|\pi) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}, \alpha \in A(s_n), s \in \mathit{Succ}(p)} (C(q) + C(\alpha))\, P(q|\pi)\, \pi(p, \alpha)\, P(s|s_n, \alpha) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} \sum_{\alpha \in A(s_n),\ s \in \mathit{Succ}(p)} (C(q) + C(\alpha))\, P(q, \pi)\, \pi(p, \alpha)\, P(s|s_n, \alpha) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi) \sum_{\alpha \in A(s_n),\ s \in \mathit{Succ}(p)} (C(q) + C(\alpha))\, \pi(p, \alpha)\, P(s|s_n, \alpha) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi) \Big(\sum_{\alpha \in A(s_n),\ s \in \mathit{Succ}(p)} C(q)\, \pi(p, \alpha)\, P(s|s_n, \alpha) + \sum_{\alpha \in A(s_n),\ s \in \mathit{Succ}(p)} C(\alpha)\, \pi(p, \alpha)\, P(s|s_n, \alpha) \Big) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi) \Big(C(q) \underbrace{\sum_{\alpha \in A(s_n),\ s \in \mathit{Succ}(p)} \pi(p, \alpha)\, P(s|s_n, \alpha)}_{=1} + \sum_{\alpha \in A(s_n)} C(\alpha)\, \pi(p, \alpha) \underbrace{\sum_{s \in \mathit{Succ}(p)} P(s|s_n, \alpha)}_{=1} \Big) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi) \Big(C(q) + \sum_{\alpha \in A(s_n)} C(\alpha)\, \pi(p, \alpha) \Big) \\
&\quad= \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi)\, C(q) + \sum_{\boldsymbol{\alpha}_{n-1}} P(q|\pi) \sum_{\alpha \in A(s_n)} C(\alpha)\, \pi(p, \alpha) \\
&\quad= \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \Pr_{\mathcal{S}^\pi}(p) \sum_{\alpha \in A(s_n)} C(\alpha)\, \pi(p, \alpha)
\end{aligned}$$

□

A.5 Linear Program for $\mathrm{LP}_{C^\times}$ and Complexity

Let $\mathcal{T}$ be a probabilistic SAS$^+$ task, $\mathcal{S}$ the SSP it defines, and $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i} \psi_i$. Fix the partial progression-based finite-memory policy $\pi^{\text{prog}} = (\boldsymbol{M}^{\text{prog}}, \mathbf{start}^{\text{prog}}, \mathbf{mod}^{\text{prog}}, \cdot)$ and $\text{Accept}_i = \{\langle \boldsymbol{m}, s\rangle \in G^\times \mid s \models_{IE} m_i\}$. Let $C^\times$ be the the C-SSP obtained according to Definition 2.

The following linear program, $\mathrm{LP}_{C^\times}$, is the standard dual LP for an SSP [Alt99] applied to $C^\times$.

In the statement of $\mathrm{LP}_{C^\times}$ *and all proofs we leave out the treatment of secondary action costs. As they can be added without effort, spelling out the details would unnecessarily clutter up the presentation.*

$$\min_{x} \quad \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A(s)} x_{\boldsymbol{m},s,\alpha}\, C(\alpha) \;+\; \sum_{\langle \boldsymbol{m},s\rangle \in G^\times} \text{in}(\boldsymbol{m}, s)\, T(s) \tag{LP4}$$

$$\text{s.t.} \quad x_{\boldsymbol{m},s,\alpha} \geq 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in S^\times, \alpha \in A(s) \tag{C27}$$

$$\text{out}(\boldsymbol{m}, s) - \text{in}(\boldsymbol{m}, s) = 0 \qquad \forall \langle \boldsymbol{m}, s\rangle \in S^\times \setminus (G^\times \cup \{s^\times_{\text{init}}\}) \tag{C28}$$

$$\text{out}(s^\times_{\text{init}}) - \text{in}(s^\times_{\text{init}}) = 1 \tag{C29}$$

$$\sum_{\langle \boldsymbol{m},s\rangle \in G^\times} \text{in}(\boldsymbol{m}, s) = 1 \tag{C30}$$

$$\sum_{\langle \boldsymbol{m},s\rangle \in G^\times} \text{in}(\boldsymbol{m}, s)\, T_i^\times(\boldsymbol{m}, s) \in z_i \qquad \forall i = 1..k \tag{C31}$$

Each constraint (C5), $e \in z_i$, where $e = \sum_{\langle \boldsymbol{m},s\rangle \in G^\times} \text{in}(\boldsymbol{m}, s)\, T_i^\times(\boldsymbol{m}, s)$ stands for the two constraints $e \geq l_i$ and $u_i \leq e$, where l_i and u_i are the lower and upper bounds, respectively, of the interval z_i.

$\mathrm{LP}_{C^\times}$, as any dual LP for SSPs, can be seen as a probabilistic flow network in which every state is a node and flows split according to the actions probabilities. Formally, the flow entering and leaving a state $\langle \boldsymbol{m}, s\rangle \in \mathcal{S}^\times$ are defined, respectively, as

$$\text{in}(\boldsymbol{m}, s) = \sum_{\langle \boldsymbol{n},t\rangle \in S^\times,\, \alpha \in A(t)} P^\times(\langle \boldsymbol{m}, s\rangle \mid \langle \boldsymbol{n}, t\rangle, \alpha)\, x_{\boldsymbol{n},t,\alpha}$$

$$\text{out}(\boldsymbol{m}, s) = \sum_{\alpha \in A(s)} x_{\boldsymbol{m},s,\alpha}$$

The variables $x_{\boldsymbol{m},s,\alpha}$ of $\mathrm{LP}_{C^\times}$ are known as occupation measures and they represents the expected number of times action $\alpha \in A(s)$ is applied in state $\langle \boldsymbol{m}, s\rangle \in \mathcal{S}^\times$.

The solution x^* of $\mathrm{LP}_{C^\times}$ induces the optimal policy $\pi^*_{C^\times} : S^\times \times A \mapsto \mathbb{Q}$ of $C^\times$ defined pointwise as follows (this is well-known):

$$\pi^*_{C^\times}(\boldsymbol{m}, s, \alpha) = \frac{x^*_{\boldsymbol{m},s,\alpha}}{\sum_{\alpha' \in A(s)} x^*_{\boldsymbol{m},s,\alpha'}}$$

for all $\langle \boldsymbol{m}, s\rangle \in S^\times$ and $\alpha \in A(s)$. Moreover, the cost of $\pi_{C^\times}$ is the term minimized in (LP1).

Constraint (C28) represents the conservation of flow that is enforce for all states except the initial state $s^\times_{\text{init}}$ which is the source of flow (C29) and the goal states that are the sinks (C30). Constraint (C30) also uses two key properties of the dual LP for SSPs: (i) exactly one unit of flow is inject in the source of the network ($s^\times_{\text{init}}$), and (ii) the flow reaching any goal state $\langle \boldsymbol{m}, s\rangle \in G^\times$ is trapped there, i.e., it is not moved back into the network. Properties (i) and (ii) together imply that under the policy $\pi^*_{C^\times}$ the probability of reaching a goal state $\langle \boldsymbol{m}, s\rangle \in G^\times$ is one, i.e., that $\pi^*_{C^\times}$ is proper, as required for optimal policies.

Theorem 4 (Complexity) *Let $\mathcal{S}$ be an SSP, φ an MO-PLTL constraint, and $C^\times$ the C-SSP obtained using $\boldsymbol{M}^{\text{prog}}$, $\mathbf{start}^{\text{prog}}$, and $\mathbf{mod}^{\text{prog}}$. Then there is a linear program $\mathrm{LP}_{C^\times}$ whose solution, if any, defines the optimal policy $\pi^*_{C^\times}$ for $C^\times$. $\mathrm{LP}_{C^\times}$ can be obtained in time and space at most $O(|S|) \cdot (2^{O(|\varphi|)^7})$.*

Proof. Above we defined the linear program $\mathrm{LP}_{C^\times}$and referred to well-known results concerning the correspondence between the solution of $\mathrm{LP}_{C^\times}$and the optimal policy $\pi^*_{C^\times}$ for $C^\times$.

Regarding complexity, the claim follows form the following: for every $i = 1..k$, the mode space is of size $|M_i| = 2^{O(|\Sigma(\psi_i)|)^3} = 2^{O(|\psi_i|)^6}$, as $|\Sigma(\psi_i)| = O(|\psi_i|^2)$. The state space of $\mathcal{S}^\times$ hence is of size $|S| \cdot \prod_{i=1}^{k} 2^{O(|\psi_i|)^6} = |S| \cdot (2^{O(|\varphi|)^6})^k = |S| \cdot (2^{k \cdot O(|\varphi|)^6}) \leq |S| \cdot (2^{|\varphi| \cdot O(|\varphi|)^6}) = |S| \cdot (2^{O(|\varphi|)^7})$. Moreover, all required (set) operations can be carried out in polynomial time. □

Notice that Theorem 4 does not talk about satisfaction of the MO-PLTL constraint φ. That will be taken care of in the soundness and completeness theorems below. More specifically, the constraint (C5) enforces the MO-PLTL constraints $\mathbf{P}_{\in z_i}\, \psi_i$, for $i = 1..k$ in terms of the fraction of the flow entering the goal states (sinks) that satisfies ψ_i. This is perhaps easier to see with the following constraint (C5alt), which is equivalent to (C31) (use the definition of $T^\times$):

$$\sum_{\langle \boldsymbol{m}, s\rangle \in \mathrm{Accept}_i} \mathrm{in}(\boldsymbol{m}, s) \;\in\; z_i \quad \text{, where}$$
$$\mathrm{Accept}_i = \{\langle m_1, \ldots, m_k, s\rangle \in G^\times \mid s \models_{IE} m_i\} \qquad \forall i = 1..k \tag{C5alt}$$

Moreover, the left-hand side of (C5alt) (and hence of C31) is equivalent to the probability of ψ_i being satisfied under the infinite extension semantics. See the proof below of the soundness theorem (Theorem 5) for that.

A.6 Soundness

Theorem 5 (Soundness) *If π^{prog} exists then it is an optimal policy for $\mathcal{S}$ and φ, i.e., $\mathcal{S}, \pi^{\mathrm{prog}} \models \varphi$ and $V^{\pi^{\mathrm{prog}}} \leq V^\pi$ for all unrestricted proper policies π such that $\mathcal{S}, \pi \models \varphi$.*

Proof. Suppose that π^{prog} exists. That is, $C^\times$ is solvable and its optimal policy solution $\pi_{C^\times}$ is the act-component of π^{prog}. As every optimal policy is proper, $\pi_{C^\times}$ is proper, too. It follows that π^{prog} is proper (π^{prog} is nothing but a reformulation of $\pi_{C^\times}$ as a finite-memory history).

As $C^\times$ is solvable, with the correspondence between $C^\times$ and $\mathrm{LP}_{C^\times}$(cf. Theorem 4) we conclude that $\mathrm{LP}_{C^\times}$is feasible (has a solution) and argue with $\mathrm{LP}_{C^\times}$in the following.

The claim $\mathcal{S}, \pi^{\mathrm{prog}} \models \varphi$ follows from the satisfaction of the constraint (C5alt). (Recall from Section A.5 that the constraint (C31) of $\mathrm{LP}_{C^\times}$is equivalent to (C5alt).) This can be explained as follows: take any goal state $\langle \boldsymbol{m}, s\rangle \in G^\times$. Consider the set P of all paths of $\mathcal{S}^\times$ from $s^\times_{\mathrm{init}}$ that end in $\langle \boldsymbol{m}, s\rangle$. Recall that $\boldsymbol{m} = \langle m_1, \ldots, m_k\rangle$ and each m_i is a set formula representing the current progression of the initial formula ψ_i. Now, either $\langle \boldsymbol{m}, s\rangle \in \mathrm{Accept}_i$ or not. If not, none of paths in P IE-satisfies ψ_i. If yes, every path in P IE-satisfies ψ_i. These claims follow from Proposition 17. In the former case the probability of P is 0, and in the latter case it is exactly the inflow into $\langle \boldsymbol{m}, s\rangle$.

The probability of all paths from $s^\times_{\mathrm{init}}$ that IE-satisfy ψ_i hence is the sum of the probabilities of reaching some goal state $\langle \boldsymbol{m}, s\rangle \in G^\times$ that accept at i. The constraint (C5alt) does just that in its summation formula and wraps it into an interval membership condition to represent the constraint $\mathbf{P}_{\in z_i}\, \psi_i$ for each constraint of φ. It follows $\mathcal{S}, \pi^{\mathrm{prog}} \models \varphi$.

Regarding optimality, suppose by way of contradiction that π^{prog} is not optimal. Then there is a (possibly unrestricted) proper policy π for $\mathcal{S}$ with strictly smaller cost $V^\pi < V^{\pi^{\mathrm{prog}}}$. By the completeness theorem below, Theorem 6, conclude $V^{\pi^{\mathrm{prog}}} \leq V^\pi$, a plain contradiction. □

A.7 Completeness

In this section we prove Theorem 6, the completess result wrt. progression-based finite-memory policies $\pi^{\mathrm{prog}} = (\boldsymbol{M}^{\mathrm{prog}}, \mathbf{start}^{\mathrm{prog}}, \mathbf{mod}^{\mathrm{prog}}, \pi^*_{C^\times})$. With the correspondence between the optimal policy

$\pi^*_{C^\times}$ for $C^\times$ and the solution of the linear program $\mathrm{LP}_{C^\times}$(cf. Theorem 4 in Section A.5) we will build our arguments on properties of $\mathrm{LP}_{C^\times}$. For that, we replace the constraint (C31) in $\mathrm{LP}_{C^\times}$ by the equivalent constraint (C5alt).

Throughout this section let $\mathcal{S} = (S, s_{\text{init}}, G, A, P, C, T)$ be a given MO-PLTL SSP and $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$ a given MO-PLTL formula.

We split the proof of the completness theorem in two main parts, one concerning "probabilities" and another one concerning "costs".

A.7.1 Probabilities

Lemma 23 *Let π be a proper policy for $\mathcal{S}$. Let $\mathrm{LP}^-_{C^\times}$ denote the subset (C1)-(C3) of $\mathrm{LP}_{C^\times}$. Then $\mathrm{LP}^-_{C^\times}$ is satisfiable with a solution σ that satisfies all of the following*:[8]

(*1*) *for every $s \in G$, $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) = \sum_{\boldsymbol{m} \in \boldsymbol{M}} \sigma(\mathrm{in}(\boldsymbol{m}, s))$.*

(*2*) *for every $i = 1..k$, $\Pr_{S^\pi}(\psi_i) = \sum_{\langle \boldsymbol{m}, s\rangle \in \mathrm{Accept}_i} \sigma(\mathrm{in}(\boldsymbol{m}, s))$.*

Lemma 23 states in its item (1) that the probability of reaching a goal state s via an unrestricted policy π can as well be found as the solution of $\mathrm{LP}^-_{C^\times}$ in terms of inflow into states of $\mathcal{S}^\times$ whose state component is s. Item (2) says that the probability of each LTL formula Ψ_i in the objective can be obtained by summing up the inflow into those goal states of $\mathcal{S}^\times$ with a state component s such that the run $ss\cdots$ satisfies the progressed version of ψ_i at that goal states, i.e., $s \models_{IE} \psi_i$.

As the proof of Lemma 23 is quite lengthy we provide an overview first. Quite obviously, we need to analyze the set of runs of $\mathcal{S}^\pi$, which we simply denote by R in the following. The end product of this analysis will be the claimed solutionσ of $\mathrm{LP}^-_{C^\times}$.

We visualize R as the obvious (infinite) tree and approximate it in a stepwise way by a sequence $\mathcal{P}_0, \mathcal{P}_1, \cdots$ of finite paths $\mathcal{P}_i \subseteq \mathit{Paths}$. Each $\mathcal{P}_i$ consists of the leaves of a finite subtree of R, and hence each $p \in \mathcal{P}_i$ represents a finite run of $\mathcal{S}^\pi$ of the form $((s_{\text{init}}) = (p_1))\,(p_1 p_2),\ \cdots\ ((p_1 p_2 \cdots p_n) = p)$.

At each stage of the the approximation, such a path p ends in a goal state, or not. If p ends in a non-goal state $s \in S \setminus G$ we *do* temporarily think of s as a goal state for the purpose of the proof. This does not match with the SSP $\mathcal{S}$, of course. As a fix, we add a designated goal state s_ε to S, for every $s \in S \setminus G$ that represents the runs of S^π that get "stuck" at a path ending in s. We call the resulting MO-LTL SSP $\mathcal{T}$ and get its product SSP $\mathcal{T}^\times$ with the same construction as described for $\mathcal{S}$.

At any stage of the approximation, a current solution σ_i of (C1)–(C3) of the LP for $\mathcal{T}^\times$ will be in sync with the runs of $\mathcal{T}^\times$ described by the paths in $\mathcal{P}_i$. More precisely, σ_i assigns to the inflow into a state of $\mathcal{T}^\times$ exactly the probability of the paths in $\mathcal{P}^i$ ending in the same state. This property holds initially and is preserved in the transition from $\mathcal{P}_i$ to $\mathcal{P}_{i+1}$. In each such transition, a $p \in \mathcal{P}_i$ ending in a non-goal state is replaced by all its extensions with the immediate successors as prescribed by π. Accordingly, the probability of p is distributed into these successors. (If such a successor state s is a non-goal state s then the flow is redirected into s_ε.) Similarly, the total expected costs of the paths in $\mathcal{P}_i$ is the same as the expected total costs given by the current solution σ_i. Below we explain in detail how σ_{i+1} is obtained from σ_i so that these invariants are preserved.

As this construction proceeds we get with the σ_i's better and better approximations of π. Because π is proper, the inflow into the artificial goal states of $\mathcal{T}^\times$ corresponding to non-goal states of $\mathcal{S}$ will be zero in the limit of the construction, when i approaches infinity. But then, $\mathcal{T}^\times$ and $\mathcal{S}^\times$ are the same, and the lemma claims (1) and (2) follow.

The rest of this section is a rigorous proof of Lemma 23.

[8] See Section A.4 for the definitions of $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$ and $\Pr_{\mathcal{S}^\pi}(\psi_i)$.

The construction. Formally, let $S_\varepsilon = \{s_\varepsilon \mid s \in S \setminus G\}$ be an ε-indexed copy of the non-goal states. We use it do define the SSP $\mathcal{T} = (T, s_{\text{init}}, G \cup S_\varepsilon, A_\varepsilon, P_\varepsilon, C_\varepsilon)$ where $T = S \cup S_\varepsilon$, $A_\varepsilon = A \cup \{\varepsilon\}$, and C_ε differs from C only by the extension $C(\varepsilon) := 0$. The states s_ε are sink states (only) for transitioning out from non-goal states s. Correspondingly we enable ε as per $A_\varepsilon(s) := \emptyset$ if $s \in G \cup S_\varepsilon$ (recall that $A(s) = \emptyset$ for all $s \in G$) and $A_\varepsilon(s) := A(s) \cup \{\varepsilon\}$ if $s \in S \setminus G$. The transition probability function P_ε is the same as P but extended by $P(\cdot|s, \varepsilon) = \{s_\varepsilon\}$. The evaluation function at s_ε is the same as at s, i.e., $s_\varepsilon[v] = s[v]$ for all $v \in \mathcal{V}$ (but that does not matter).

In the following we work with the product SSP $\mathcal{T}^\times$ of $\mathcal{T}$. When we speak of "(C1)–(C3)" we mean the constraints (C1)–(C3) in the linear program for the SSP $\mathcal{T}^\times$, not $\mathcal{S}^\times$. By $T^\times$ we denote the state space of $\mathcal{T}^\times$. By an *s-state* we mean any state of $T^\times$ whose state component is s, i.e., a state of the form $\langle \boldsymbol{m}, s\rangle \in T^\times$.

We define inductively sequences $(\mathcal{P}_i)_{i\geq 1}$ with $\mathcal{P}_i \subseteq \mathit{Paths}$ and $(\sigma_i)_{i\geq 1}$ such that σ_i is a solution of (C1)–(C3) . The set $\mathcal{P}_i$ is the current set of paths, and those that end in non-goal states need to be worked on. On transition from i to $i+1$ such a $p \in \mathcal{P}_i$ is replaced by all one-step extensions of p. Here are the details:

$i = 1$**:** set $\mathcal{P}_1 = \{s_{\text{init}}\}$ and σ_1 as

$$x_{\mathbf{start}, s_{\text{init}}, \varepsilon} := 1 \qquad x_{\boldsymbol{m}, s, \alpha} := 0 \quad \text{in all other cases}$$
$$\text{in}(\mathbf{start}, (s_{\text{init}})_\varepsilon) := 1 \qquad \text{in}(\boldsymbol{m}, s) := 0 \quad \text{in all other cases}$$

Strictly speaking, we do not need to specify the values of the in-variables, as they are uniquely determined by the x-variables (and $\mathcal{T}^\times$). We do so nevertheless for clarity, noting that the specified values are indeed the uniquely determined ones. The values for the out-variables we do not specify explicitly. The same goes for the induction step below.

$i \to i+1$**:** Assume $i \geq 1$ and that $\mathcal{P}_j$ and σ_j have already been defined for all $j = 1..i$. Pick a shortest $p \in \mathcal{P}_i$ such that $\text{last}(p) \in S \setminus G$. As defined earlier, let $\mathit{Succ}(p) = \{s \in S \mid P_{\mathcal{S}^\pi}(ps|p) > 0\}$ denote the reachable successor states of p.

We replace p in $\mathcal{P}_i$ by its extension by successors. That is, set

$$\mathcal{P}_{i+1} = (\mathcal{P}_i \setminus \{p\}) \cup \{ps \mid s \in \mathit{Succ}(p)\} \ .$$

We need to update σ_i accordingly. Let $p = s_1 \cdots s_n$ (where $s_1 = s_{\text{init}}$). Corresponding to p there is a unique path $p^\times = \langle \boldsymbol{m}_1, s_1\rangle\langle \boldsymbol{m}_2, s_2\rangle \cdots \langle \boldsymbol{m}_{n-1}, s_{n-1}\rangle\langle \boldsymbol{m}_n, s_n\rangle$ of $\mathcal{S}^\times$ (and of $\mathcal{T}^\times$), where $\boldsymbol{m}_1 = \mathbf{start}$ and $\boldsymbol{m}_{i+1} = \mathbf{mod}(\boldsymbol{m}_i, s_{i+1})$ for $i = 1..n-1$. Looking at the last state $\langle \boldsymbol{m}_n, s_n\rangle$ we modify σ_i so that it reflects the new $\mathcal{P}_{i+1}$. First set

$$x_{\boldsymbol{m}_n, s_n, \varepsilon} := x_{\boldsymbol{m}_n, s_n, \varepsilon} - \Pr_{\mathcal{S}^\pi}(p)$$
$$\text{in}(\boldsymbol{m}_n, (s_n)_\varepsilon) := \text{in}(\boldsymbol{m}_n, (s_n)_\varepsilon) - \Pr_{\mathcal{S}^\pi}(p)$$

in accordance with p being removed from $\mathcal{P}_i$. (Recall that p ends in a non-goal state, s_n, and its probability went into $(s_n)_\varepsilon$.) That is, the outflow $x_{\boldsymbol{m}_n, s_n, \varepsilon}$ of $\langle \boldsymbol{m}_n, s_n\rangle$ into the ε-successor state $\langle \boldsymbol{m}_n, (s_n)_\varepsilon\rangle$ is reduced by the probability of p. This loss of outflow is compensated by distributing it into the successor states $\langle \mathbf{mod}(\boldsymbol{m}_n, s), s\rangle$ of $\langle \boldsymbol{m}_n, s_n\rangle$ as prescribed by π, for all $s \in \mathit{Succ}(p)$. More precisely, set

$$x_{\boldsymbol{m}_n, s_n, \alpha} := x_{\boldsymbol{m}_n, s_n, \alpha} + \Pr_{\mathcal{S}^\pi}(p)\,\pi(p, \alpha) \qquad (\text{for all } \alpha \in A(s_n))$$
$$\text{in}(\mathbf{mod}(\boldsymbol{m}_n, s), s) := \text{in}(\mathbf{mod}(\boldsymbol{m}_n, s), s) + \Pr_{\mathcal{S}^\pi}(p)\,P_{\mathcal{S}^\pi}(ps|p) \qquad (\text{for all } s \in \mathit{Succ}(p))$$
$$\left(= \text{in}(\mathbf{mod}(\boldsymbol{m}_n, s), s) + \Pr_{\mathcal{S}^\pi}(p)\,\Big(\sum_{\alpha \in A(s_n)} \pi(p, \alpha)\,P(s|s_n, \alpha)\Big)\right)$$

Now consider the states $s \in \mathit{Succ}(p)$. If $s \in G$ then the corresponding state $\langle \mathbf{mod}(\boldsymbol{m}_n, s), s\rangle$ is a goal state in $\mathcal{T}^\times$. In this case nothing more needs to be done to preserve solutions, the goal state just receives somewhat more inflow. If $s \in S \setminus G$ then $\langle \mathbf{mod}(\boldsymbol{m}_n, s), s\rangle$ is a non-goal state in $\mathcal{T}^\times$ as well and we need to re-balance its outflow with the just increased inflow. We do this by sending the increased inflow into the ε-successor states of $\langle \mathbf{mod}(\boldsymbol{m}_n, s), s\rangle$, which are goal states in $\mathcal{T}^\times$. That is, set

$$x_{\mathbf{mod}(\boldsymbol{m}_n,s),s,\varepsilon} := x_{\mathbf{mod}(\boldsymbol{m}_n,s),s,\varepsilon} + \Pr_{\mathcal{S}^\pi}(p)\, P_{\mathcal{S}^\pi}(ps|p) \qquad \text{(for all } s \in S \setminus G\text{)}$$
$$\mathrm{in}(\mathbf{mod}(\boldsymbol{m}_n, s_\varepsilon), s_\varepsilon) := \mathrm{in}(\mathbf{mod}(\boldsymbol{m}_n, s_\varepsilon), s_\varepsilon) + \Pr_{\mathcal{S}^\pi}(p)\, P_{\mathcal{S}^\pi}(ps|p) \qquad \text{(for all } s \in S \setminus G\text{)}$$

This construction defines the new solution σ_{i+1} of (C1)–(C3).

It is not difficult to verify (by induction) that σ_i is a solution of (C1)–(C3) for all $i \geq 1$.

We are going to introduce some definitions that will facilitate the proofs. For $i \geq 1$ define probabilities of reaching a state $s \in S$ or some state from a set of states G or $S \setminus G$ at stage i as follows:

$$\Pr^\times(\mathbf{F}\, s)_i = \sum_{\boldsymbol{m}\in\boldsymbol{M}} \sigma_i(\mathrm{in}(\boldsymbol{m}, s)) \quad \text{if } s \in G \qquad\qquad \Pr^\times(\mathbf{F}\, G)_i = \sum_{s\in G} \Pr^\times(\mathbf{F}\, s)_i$$
$$\Pr^\times(\mathbf{F}\, s_\varepsilon)_i = \sum_{\boldsymbol{m}\in\boldsymbol{M}} \sigma_i(\mathrm{in}(\boldsymbol{m}, s_\varepsilon)) \quad \text{if } s \in S \setminus G \qquad\qquad \Pr^\times(\mathbf{F}\,(S \setminus G))_i = \sum_{s\in S\setminus G} \Pr^\times(\mathbf{F}\, s_\varepsilon)_i$$
$$\Pr^\times(\psi_j)_i = \sum_{\langle \boldsymbol{m}, s\rangle \in \mathrm{Accept}_j} \sigma_i(\mathrm{in}(\boldsymbol{m}, s)) \qquad \text{if } 1 \leq j \leq k$$

For instance, $\Pr^\times(\mathbf{F}\, s)_i$ is the inflow into all goal s-states of $\mathcal{T}^\times$ at stage i in the construction. We write, e.g., $(\Pr^\times(\mathbf{F}\, s)_i)$ to denote the sequence $\Pr^\times(\mathbf{F}\, s)_1, \Pr^\times(\mathbf{F}\, s)_2, \ldots, \Pr^\times(\mathbf{F}\, s)_n, \ldots$.

We prove some interesting facts.

(0) $\Pr^\times(\mathbf{F}\, G)_i + \Pr^\times(\mathbf{F}\,(S \setminus G))_i = 1$. Proof: Initially, when $i = 1$, $\Pr^\times(\mathbf{F}\, G)_i = 0$ and $\Pr^\times(\mathbf{F}\,(S \setminus G))_i = 1$ as per $\mathrm{in}(\mathbf{start}, (s_{\mathrm{init}})_\varepsilon) = 1$. On transitions from i to $i + 1$ an inflow into some $\mathrm{in}(\boldsymbol{m}_n, (s_n)_\varepsilon)$ is first reduced by an amount $\Pr_{\mathcal{S}^\pi}(p)$. No other state can have an inflow $\geq 1 - \Pr_{\mathcal{S}^\pi}(p)$ as then the property (0) would not hold at time i. For time $i+1$ the amount $\Pr_{\mathcal{S}^\pi}(p)$ will be redistributed into final or non-final states. In any case the mass flowing, in sum, into some nodes $\mathrm{in}(\boldsymbol{m}, s)$ where $s \in G$ or $\mathrm{in}(\boldsymbol{m}, s_\varepsilon)$ where $s \in S \setminus G$ is preserved for σ_{i+1}.

Properties of goal states. In the following let $s \in G$ arbitrary.

(a) The sequence $(\Pr^\times(\mathbf{F}\, s)_i)$ is monotonically increasing, i.e., $\Pr^\times(\mathbf{F}\, s)_i \leq \Pr^\times(\mathbf{F}\, s)_{i+1}$.

Proof: Initially $\Pr^\times(\mathbf{F}\, s)_1 = 0$, and either $\Pr^\times(\mathbf{F}\, s)_{i+1} = \Pr^\times(\mathbf{F}\, s)_i$ or the construction adds a positive amount $\Pr_{\mathcal{S}^\pi}(p)\, P_{\mathcal{S}^\pi}(ps|p)$ to the variable $\mathrm{in}(\mathbf{mod}(\boldsymbol{m}_n, s), s)$, for some p and $\boldsymbol{m}_n$.

(b) $\Pr^\times(\mathbf{F}\, s)_i = \Pr_{\mathcal{S}^\pi}(\mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_i)$ and $\Pr^\times(\mathbf{F}\, s)_i \leq \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$. Proof: the first claim follows immediately from the construction: $\mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_i$ contains exactly all paths from s_{init} to s considered up to and including timepoint i. The value $\Pr_{\mathcal{S}^\pi}(\mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_i)$ is the sum of the probabilities of these paths. By construction, each such probability has been added to a variable $\mathrm{in}(\langle \boldsymbol{m}, s\rangle)$, for some $\boldsymbol{m}$. Their sum is exactly the value $\Pr^\times(\mathbf{F}\, s)_i$.

The second claim follows from the first claim and the trivial fact $\mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_i \subseteq \mathit{Paths}(\mathbf{F}\, s)$.

(c) $\lim_{i\to\infty} \Pr^\times(\mathbf{F}\, s)_i = \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$. Proof: With the monotonicity result (a) and the upper bound result (b) it suffices to show that, given $\varepsilon > 0$, there is an N such that $\Pr^\times(\mathbf{F}\, s)_i > \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) - \varepsilon$ for all $i \geq N$. Consider an arbitrary enumeration $\mathit{Paths}(\mathbf{F}\, s) = \{p_1, p_2, \ldots\}$. Then we can present $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$ as follows:

$$\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) = \sum_{p\in \mathit{Paths}(\mathbf{F}\, s)} \Pr_{\mathcal{S}^\pi}(p) = \sum_{j=1}^{\infty} \Pr_{\mathcal{S}^\pi}(p_j) = \lim_{k\to\infty} \sum_{j=1}^{k} \Pr_{\mathcal{S}^\pi}(p_j) \ .$$

By definition of limits, there is a K such that $\sum_{j=1}^{K} \Pr_{\mathcal{S}^\pi}(p_j) > \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) - \varepsilon$ for all $k \geq K$. Now chose N big enough such that $\{p_1, \ldots, p_K\} \subseteq \mathcal{P}_N$. Because the construction is fair wrt. selecting paths from S^π, by always taking a shortest one for the next iteration, the set $\mathcal{P}_N$ exists. Because every path p_j ends in s we have $\{p_1, \ldots, p_K\} \subseteq \mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_N$ Together we obtain

$$\Pr^{\times}(\mathbf{F}\, s)_i \overset{(b)}{=} \sum_{p \in \mathit{Paths}(\mathbf{F}\, s) \cap \mathcal{P}_i} \Pr_{\mathcal{S}^\pi}(p) \geq \sum_{j=1}^{K} \Pr_{\mathcal{S}^\pi}(p_j) > \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) - \varepsilon \quad \text{for all } i \geq N.$$

(d) $\lim_{i\to\infty} \Pr^{\times}(\mathbf{F}\, G)_i = 1$. That is, the product SSP will, in the limit of the construction, almost certainly reach a goal state. Proof:

$$\lim_{i\to\infty} \Pr^{\times}(\mathbf{F}\, G)_i = \lim_{i\to\infty} \sum_{s\in G} \Pr^{\times}(\mathbf{F}\, s)_i = \sum_{s\in G} \lim_{i\to\infty} \Pr^{\times}(\mathbf{F}\, s)_i \overset{(c)}{=} \sum_{s\in G} \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) = \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, G) = 1 \ .$$

The last equality is given by Lemma 19.

(e) For every $\boldsymbol{m} \in \boldsymbol{M}$, $\lim_{i\to\infty} \sigma_i(\mathrm{in}(\boldsymbol{m}, s))$ exists.

Proof: initially $\sigma_1(\mathrm{in}(\boldsymbol{m}, s)) = 0$, and $(\sigma_i(\mathrm{in}(\boldsymbol{m}, s)))$ is bound from above, as $\sigma_i(\mathrm{in}(\boldsymbol{m}, s)) \leq \sum_{\boldsymbol{m}\in\boldsymbol{M}} \sigma_i(\mathrm{in}(\boldsymbol{m}, s)) = \Pr^{\times}(\mathbf{F}\, s)_i \overset{(a,c)}{\leq} \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)$.

(f) for every $\boldsymbol{m} \in \boldsymbol{M}$, $\sigma_i(x_{\boldsymbol{m},s,\alpha}) = 0$. Proof: this is true initially, for $i = 1$, and it is preserved from $i \mapsto i + 1$ as the construction does not distribute flow coming into goal states G. In other words, $\sigma_{i+1}(x_{\boldsymbol{m},s,\alpha}) = \sigma_i(x_{\boldsymbol{m},s,\alpha})$.

(g) for every $j = 1..k$, $\lim_{i\to\infty} \Pr^{\times}(\psi_j)_i = \Pr_{\mathcal{S}^\pi}(\psi_j)$. The proof is analogous to the proofs of (a) - (c), with ψ_j replacing $\mathbf{F}\, s$, and is omitted.

Properties of non-goal states. In the following let $s \in S \setminus G$ arbitrary.

(h) The sequence $(\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i)$ is decreasing, i.e., $\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i \geq \Pr^{\times}(\mathbf{F}\,(S \setminus G))_{i+1}$. Moreover, $\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i \geq 0$. Proof: Initially, when $i = 1$, $\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i = 1$ as per $\mathrm{in}(\mathbf{start}, (s_{\mathrm{init}})_\varepsilon) = 1$. On transitions from i to $i+1$ an inflow into some $\mathrm{in}(\boldsymbol{m}_n, (s_n)_\varepsilon)$ is first diminished by an amount $\Pr_{\mathcal{S}^\pi}(p)$. This can only happen if the same amount $\Pr_{\mathcal{S}^\pi}(p)$ had been added to $\mathrm{in}(\boldsymbol{m}_n, (s_n)_\varepsilon)$ in an earlier step. It follows $\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i \geq 0$. The amount $\Pr_{\mathcal{S}^\pi}(p)$ then is redistributed into goal or non-goal states. In any case the mass flowing into nodes $\mathrm{in}(\boldsymbol{m}, s_\varepsilon)$ does not increase. Hence $\Pr^{\times}(\mathbf{F}\,(S \setminus G))_i \geq \Pr^{\times}(\mathbf{F}\,(S \setminus G))_{i+1}$.

(i) $\lim_{i\to\infty} \Pr^{\times}(\mathbf{F}\,(S \setminus G))_i = 0$. In other words, the probability of getting stuck at a non-goal state is zero. Proof: follows immediately from (0), (d) and the trivial fact $S = (S \setminus G) \cup G$.

(j) $\lim_{i\to\infty} \Pr^{\times}(\mathbf{F}\, s_\varepsilon)_i = 0$. Proof: follows immediately from (i).

(k) For every $\boldsymbol{m} \in \boldsymbol{M}$, $\lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m},s,\varepsilon}) = 0$. Proof: Choose $\boldsymbol{m} \in \boldsymbol{M}$ arbitrarily. By definition, the only ε-successor state of $\langle \boldsymbol{m}, s\rangle$ is $\langle \boldsymbol{m}, s_\varepsilon\rangle$. That is, $\mathrm{in}(\boldsymbol{m}, s_\varepsilon) = x_{\boldsymbol{m},s,\varepsilon}$. From (j) it follows that the limit inflow into $\langle \boldsymbol{m}, s_\varepsilon\rangle$ is 0, i.e., $\lim_{i\to\infty} \sigma_i(\mathrm{in}(\boldsymbol{m}, s_\varepsilon)) = 0$. Together it follows $\lim_{i\to\infty} \sigma_i(\mathrm{in}(\boldsymbol{m}, s_\varepsilon)) = \lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m},s,\varepsilon}) = 0$.

(l) For every $\boldsymbol{m} \in \boldsymbol{M}$ and $\alpha \in A_\varepsilon(s)$, $\lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m},s,\alpha}) = r$ for some finite real number r. Proof: choose $\boldsymbol{m} \in \boldsymbol{M}$ and $\alpha \in A_\varepsilon(s)$ arbitrarily. If $\alpha = \varepsilon$ then $r = 0$ as per (k). Otherwise, $\alpha \neq \varepsilon$ and the variable $x_{\boldsymbol{m},s,\alpha}$ is the expected number of times the action α is carried out in state $\langle \boldsymbol{m}, s\rangle$. It is possible to quantify an a-priori finite upper bound for $x_{\boldsymbol{m},s,\alpha}$ by graph analysis of $\mathcal{T}^{\times}$. As this is rather involved and we do not need the upper bound as such we provide a simpler argument.

As a preliminary step we need to establish the monotonicity property

$$\sigma_i(x_{\boldsymbol{n},t,\alpha}) \leq \sigma_{i+1}(x_{\boldsymbol{n},t,\alpha}), \text{ for all } \langle \boldsymbol{n}, t\rangle \in S^{\times} \text{ and } \alpha \in A(t). \tag{2}$$

Proof of (2): let $\langle \boldsymbol{n}, t\rangle \in S^{\times}$ and $\alpha \in A(t)$ arbitrary. Notice that $\alpha \neq \varepsilon$, which is the key to why this holds: at any transition $i \mapsto i + 1$ in the construction, the values of some variables $x_{\boldsymbol{m}_n,s_n,\varepsilon}$ and

$\mathrm{in}(\boldsymbol{m}_n, (s_n)_\varepsilon)$ *only* are diminished, by a value $Pr_{\mathcal{S}^\pi}(p)$, which is distributed additively into x- and in-variables (the same variables possibly among them). Hence (2) follows.

Suppose $\sigma_i(\mathrm{in}(\boldsymbol{m}, s)) > 0$ at some time i in the construction. We claim there is a path in the transition graph of $\mathcal{T}^\times$ from $\langle \boldsymbol{m}, s\rangle$, using enabled actions, to a goal state of $\mathcal{T}^\times$. Each such goal state is of the form $\langle \boldsymbol{n}, t\rangle$ for some $t \in G$ or $\langle \boldsymbol{n}, t_\varepsilon\rangle$ for some $t \in (S \setminus G)$. Let us call the former the non-ε-states and the latter the ε-states. The existence of paths to ε-states is obvious from the definition of $\mathcal{T}^\times$, but one of these paths must lead to a non-ε-state. This is, because otherwise the (non-zero) inflow $\sigma_i(\mathrm{in}(\boldsymbol{m}, s))$ would be distributed, as the construction proceeds, only to ε-states, if at all. But this is impossible with the property (i) which entails that the summed-up inflow into the ε-states approaches 0 in the limit. Hence there is at least one path to a non-ε-state $\langle \boldsymbol{n}, t\rangle$ for some $t \in G$.

The inflow into that state $\langle \boldsymbol{n}, t\rangle$, i.e., $\sigma_i(\mathrm{in}(\boldsymbol{n}, t))$, is at any time i a certain fraction of the x-value of each non-goal state along the path from $s^\times_{\mathrm{init}}$ to $\langle \boldsymbol{n}, t\rangle$ passing through non-goal states (including $\langle \boldsymbol{m}, s\rangle$) and using enabled actions $\alpha \neq \varepsilon$. If $\langle \boldsymbol{n}', t'\rangle$ is any such non-goal state and $\sigma_i(x_{\boldsymbol{n}',t',\alpha})$ should not converge to a finite value, then with (2) it follows $\sigma_i(x_{\boldsymbol{n}',t',\alpha})$ converges to infinity. But this is impossible as then $\sigma_i(\mathrm{in}(\boldsymbol{n}, t))$ would converge to infinity as well, a contradiction to property (d).

To wrap up, we have shown that if $\sigma_i(\mathrm{in}(\boldsymbol{m}, s)) > 0$, for some i, then $\sigma_i(x_{\boldsymbol{m},s,\alpha})$ converges to a finite value. Trivially, if $\sigma_i(x_{\boldsymbol{m},s,\alpha}) = 0$, for all i, then convergence (to 0) holds as well. Thus, in any case, $\sigma_i(x_{\boldsymbol{m},s,\alpha})$ converges to a finite value r.

Definition of σ. We are going to define the desired solution σ of (C1)–(C5alt). Like in the construction of σ_i we define it explicitly only on the variables $\mathrm{in}(\boldsymbol{m}, s)$ and $x_{\boldsymbol{m},s,\alpha}$.

First define $\sigma(x_{\boldsymbol{m}s,\alpha}) = \lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m}s,\alpha})$ for every $\langle \boldsymbol{m}, s\rangle \in S^\times$ and $\alpha \in A_\varepsilon(s)$. Notice that the limit is well-defined. For $s \in (S \setminus G)$ use property (l), and for $s \in G$ this is trivial with (f).

Then define σ on the variables $\mathrm{in}(\boldsymbol{m}, s)$ for each $\langle \boldsymbol{m}, s\rangle \in S^\times$ as

$$\sigma(\mathrm{in}(\boldsymbol{m}, s)) = \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A_\varepsilon(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, P^\times(\langle \boldsymbol{n}, t\rangle | \langle \boldsymbol{m}, s\rangle, \alpha) \quad .$$

We can characterize σ on the in-variables also as the "limit σ_i":

$$\sigma(\mathrm{in}(\boldsymbol{n}, t)) = \lim_{i\to\infty} \sigma_i(\mathrm{in}(\boldsymbol{n}, t)), \text{ for all } \langle \boldsymbol{n}, t\rangle \in S^\times \tag{3}$$

Proof of (3): choose $\langle \boldsymbol{n}, t\rangle \in S^\times$ arbitrarily and obtain:

$$\begin{aligned}
\sigma(\mathrm{in}(\boldsymbol{n}, t)) &= \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A_\varepsilon(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, P^\times(\langle \boldsymbol{n}, t\rangle | \langle \boldsymbol{m}, s\rangle, \alpha) && \text{(def. of } \sigma(\mathrm{in}(\boldsymbol{n}, t))) \\
&= \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A_\varepsilon(s)} (\lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m},s,\alpha})\, P^\times(\langle \boldsymbol{n}, t\rangle | \langle \boldsymbol{m}, s\rangle, \alpha) && \text{(def. of } \sigma(x_{\boldsymbol{m},s,\alpha})) \\
&= \lim_{i\to\infty} \sigma_i(\sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A_\varepsilon(s)} x_{\boldsymbol{m},s,\alpha}\, P^\times(\langle \boldsymbol{n}, t\rangle | \langle \boldsymbol{m}, s\rangle, \alpha)) && \text{(limit property)} \\
&= \lim_{i\to\infty} \sigma_i(\mathrm{in}(\boldsymbol{n}, t)) && (\sigma_i \text{ is solution of (C1)–(C3)})
\end{aligned}$$

σ is a solution of (C1)–(C3). The lemma statement requires to prove that σ is a solution of the subset (C1)–(C3) of $\mathrm{LP}_{C^\times}$ for $\mathcal{S}^\times$. For this, we can first show that σ is a solution of the subset (C1)–(C3) of $\mathrm{LP}_{C^\times}$ for $\mathcal{T}^\times$. This is done in a similar way as the proof of lemma item (2), see below. As there, one needs the facts that limits distribute over sums and multiplication by constants and that σ_i is a solution of (C1)–(C3), for all $i \geq 1$. We omit the details.

Then, to show that σ is a solution of the subset (C1)–(C3) of $\mathrm{LP}_{C^\times}$ for $\mathcal{S}^\times$, recall property (j) which states $\lim_{i\to\infty} \Pr^\times(\mathbf{F}\, s_\varepsilon)_i = 0$ for all $s \in (S \setminus G)$. Using (3) it is easy to take (j) to the limit and conclude

$\sigma(\text{in}(\boldsymbol{m}, s_\varepsilon)) = 0$, for all $\boldsymbol{m} \in \boldsymbol{M}$ and $s \in (S \setminus G)$. It follows $\sigma(x_{\boldsymbol{m},s,\varepsilon}) = 0$. In other words, we can ignore the additional ε-actions and s_ε-states and still have a solution of $\mathcal{T}^\times$. As these are the only differences between $\mathcal{S}$ and $\mathcal{T}$, σ is a solution of the subset (C1)–(C3) of $\text{LP}_{C^\times}$ for $\mathcal{S}^\times$.

We turn now to the proofs of the lemma claims (1) and (2).

Item (1). We need to show $\Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s) = \sum_{\boldsymbol{m} \in \boldsymbol{M}} \sigma(\text{in}(\boldsymbol{m}, s))$, for every $s \in G$. Again this follows easily, by taking property (c) to the limit, using (3).

Item (2). Let $j \in \{1..k\}$ arbitrary. We show $\Pr_{S^\pi}(\psi_j) = \sum_{\langle \boldsymbol{m},s\rangle \in \text{Accept}_j} \sigma(\text{in}(\boldsymbol{m}, s))$ as follows:

$$
\begin{aligned}
\Pr_{S^\pi}(\psi_j) &\overset{\text{(g)}}{=} \lim_{i\to\infty} \Pr^\times(\psi_j)_i \\
&= \lim_{i\to\infty} \sum_{\langle \boldsymbol{m},s\rangle \in \text{Accept}_j} \sigma_i(\text{in}(\boldsymbol{m}, s)) \\
&= \sum_{\langle \boldsymbol{m},s\rangle \in \text{Accept}_j} \lim_{i\to\infty} \sigma_i(\text{in}(\boldsymbol{m}, s)) && \text{(as by (e) } \lim_{i\to\infty} \sigma_i(\text{in}(\boldsymbol{m}, s)) \text{ exists)} \\
&= \sum_{\langle \boldsymbol{m},s\rangle \in \text{Accept}_j} \sigma(\text{in}(\boldsymbol{m}, s)) \ . && \text{(by (3))}
\end{aligned}
$$

This concludes the proof of Lemma 23.

A.7.2 Costs

Lemma 24 *Let π be a proper policy for $\mathcal{S}$ and suppose σ is the solution of $\text{LP}^-_{C^\times}$ (the subset (C1)–(C3) of $\text{LP}_{C^\times}$) as constructed in the proof of Lemma 23. Then*

$$
V^\pi = \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, C(\alpha) \ + \sum_{\langle \boldsymbol{m},s\rangle \in G^\times} \sigma(\text{in}(\boldsymbol{m}, s))\, T(s) \ .
$$

The lemma states that the expected total costs of reaching a goal state of $\mathcal{S}$ under π is the same as the computed ones by the solution σ of $\text{LP}^-_{C^\times}$.

For the proof of Lemma 24 we need additional concepts, referring to the construction in the proof of Lemma 23. For every $i \geq 1$ define

$$
\begin{aligned}
\text{ExpTot}_i &= \sum_{p \in \mathcal{P}_i} \text{ExpTot}_{\mathcal{S}^\pi}(p) \\
\text{ExpTot}^\times_i &= \sum_{\langle \boldsymbol{m},s\rangle \in S^\times, \alpha \in A(s)} \sigma_i(x_{\boldsymbol{m},s,\alpha})\, C(\alpha)
\end{aligned}
$$

Notice that the paths p in the equation for ExpTot_i not neccessarily end in a goal state from G. ExpTot_i is the expected total cost of all finite runs from S^π at timepoint i, and $\text{ExpTot}^\times_i$ is the expected total cost computed with the solution σ_i so far.

We first establish some properties regarding ExpTot_i and $\text{ExpTot}^\times_i$. The proof of Lemma 24 is at the end of this section.

Lemma 25 *For all $i \geq 1$ and $p \in \mathcal{P}_i$, $\Pr_{\mathcal{S}^\pi}(p) > 0$. For all $i \geq 2$ and $p \in \mathcal{P}_i$, $\text{ExpTot}_{\mathcal{S}^\pi}(p) > 0$.*

Proof. By induction. If $i = 1$ then $\mathcal{P}_1 = \{s_{\text{init}}\}$ and $\Pr_{\mathcal{S}^\pi}(s_{\text{init}}) = 1$. If $i = 2$ then $\mathcal{P}_2 = \{s_{\text{init}} s \mid s \in \mathit{Succ}(s_{\text{init}})\}$. By definition of *Succ*, each path in $\mathcal{P}_2$ has a non-0 probability, and one action applied to it. No action can have 0 costs by definition SSPs. It follows $\text{ExpTot}_{\mathcal{S}^\pi}(p) > 0$ for each $p \in \mathcal{P}_2$.

For $i \mapsto i+1$, consider the first claim and let $p \in \mathcal{P}_i$ arbitrary. By induction we have $\Pr_{\mathcal{S}^\pi}(p) > 0$. The case $p \in \mathcal{P}_{i+1}$ is trival. Hence suppose p is the chosen path in the construction. By definition, then, $\mathcal{P}_{i+1} = (\mathcal{P}_i \setminus \{p\}) \cup \{ps \mid s \in \mathit{Succ}(p)\}$. The claim follows immediately with the definition of *Succ*. The proof of the second claim is similar and again uses the requirement that no action can have 0 costs. □

Lemma 26 (Montonicity) *For all* $i \geq 1$, $\mathrm{ExpTot}_{i+1} > \mathrm{ExpTot}_i$.
(*Equivalently,* $\sum_{p \in \mathcal{P}_{i+1}} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \sum_{p \in \mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$).

Proof. Let $q = s_1 \cdots s_n \in \mathcal{P}_i$ be the chosen path to work on at timepoint i. Compute

$$\begin{aligned}
\mathrm{ExpTot}_{i+1} &= \sum_{p \in \mathcal{P}_{i+1}} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) \\
&= \sum_{p \in (\mathcal{P}_i \setminus \{q\}) \cup \{qs \mid s \in \mathit{Succ}(q)\}} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) && \text{(Def. of } \mathcal{P}_{i+1}) \\
&= \sum_{p \in \mathcal{P}_i \setminus \{q\}} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \sum_{s \in \mathit{Succ}(q)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(qs) \\
&= \sum_{p \in \mathcal{P}_i \setminus \{q\}} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \big(\mathrm{ExpTot}_{\mathcal{S}^\pi}(q) + \Pr_{\mathcal{S}^\pi}(q) \sum_{\alpha \in A(s_n)} C(\alpha)\,\pi(q,\alpha)\big) && \text{(Lemma 20)} \\
&= \sum_{p \in \mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \Pr_{\mathcal{S}^\pi}(q) \sum_{\alpha \in A(s_n)} C(\alpha)\,\pi(q,\alpha) \\
&= \mathrm{ExpTot}_i + \underbrace{\Pr_{\mathcal{S}^\pi}(q) \sum_{\alpha \in A(s_n)} C(\alpha)\,\pi(q,\alpha)}_{=:\delta}
\end{aligned}$$

The first lemma claim follows from $\delta > 0$, which in turn follows from the following facts:

- $\Pr_{\mathcal{S}^\pi}(q) > 0$ for all $q \in \mathcal{P}_i$ and all $i \geq 1$, cf. Lemma 25.
- The last state s_n of q cannot be a goal state (otherwise it would not be touched by the construction from i to $i+1$). By definition of SSP, $A(s_n) \neq \emptyset$ and $C(\alpha) > 0$, for all $\alpha \in A$. Furthermore, $\pi(q)$ is a distribution on the actions A. Altogether it follows there is at least one $\alpha \in A(s_n)$ such that $C(\alpha)\,\pi(q,\alpha) > 0$.

The second lemma claim is just unfolding the definition of ExpTot_i. □

Lemma 27 *For all* $i \geq 1$, $\mathrm{ExpTot}_i = \mathrm{ExpTot}_i^\times$.

Proof.(of Lemma 27) By induction. For $i = 1$, $\mathcal{P}_1 = \{s_{\text{init}}\}$ and $\mathrm{ExpTot}_1 = 0$ with the definition of $\Pr_{\mathcal{S}^\pi}$ (the empty sum is 0). Likewise $\mathrm{ExpTot}_1^\times = 0$ as the only non-0 x-variable is $x_{\mathbf{start}, s_{\text{init}}, \varepsilon} = 1$ but the cost for executing ε is $C(\varepsilon) = 0$.

For $i \mapsto i+1$ assume $\mathrm{ExpTot}_i = \mathrm{ExpTot}_i^\times$ and let $p = s_1 \cdots s_n \in \mathcal{P}_i$ be the chosen path. The corresponding path from $\mathcal{S}^\times$ (and from $\mathcal{T}^\times$) is $p^\times = \langle \boldsymbol{m}_1, s_1\rangle\langle \boldsymbol{m}_2, s_2\rangle \cdots \langle \boldsymbol{m}_{n-1}, s_{n-1}\rangle\langle \boldsymbol{m}_n, s_n\rangle$, where $\langle \boldsymbol{m}_1, s_1\rangle = \langle \mathbf{start}, s_{\text{init}}\rangle$ and $\boldsymbol{m}_{i+1} = \mathbf{mod}(\boldsymbol{m}_i, s_{i+1})$ for $i = 1..n-1$.

The construction makes $\mathcal{P}_{i+1}$ from $\mathcal{P}_i$ by replacing p by the paths ps, for all $s \in \mathit{Succ}(p)$. The new value for ExpTot_{i+1} is correspondingly obtained from ExpTot_i as follows:

$$\begin{aligned}
\mathrm{ExpTot}_{i+1} &= \mathrm{ExpTot}_i - \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \sum_{s \in \mathit{Succ}(p)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(ps) \\
&= \mathrm{ExpTot}_i - \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \big(\mathrm{ExpTot}_{\mathcal{S}^\pi}(p) + \Pr_{\mathcal{S}^\pi}(p) \sum_{\alpha \in A(s_n)} \pi(p,\alpha)\, C(s_n,\alpha)\big) && \text{(by Lemma 20)} \\
&= \mathrm{ExpTot}_i + \Pr_{\mathcal{S}^\pi}(p) \sum_{\alpha \in A(s_n)} \pi(p,\alpha)\, C(\alpha)
\end{aligned}$$

Analogously, the new value for $\mathrm{ExpTot}^{\times}_{i+1}$ is obtained from $\mathrm{ExpTot}^{\times}_{i}$ as follows:

$$\begin{aligned}
\mathrm{ExpTot}^{\times}_{i+1} &= \sum_{\langle \boldsymbol{m},s\rangle \in S^{\times},\alpha\in A_\varepsilon(s)} \sigma_{i+1}(x_{\boldsymbol{m},s,\alpha})\, C_\varepsilon(\alpha) \\
&= \sum_{\langle \boldsymbol{m},s\rangle \in S^{\times},\alpha\in A_\varepsilon(s)} \sigma_{i}(x_{\boldsymbol{m},s,\alpha})\, C_\varepsilon(\alpha) - \big(\mathrm{Pr}_{\mathcal{S}^\pi}(p)\ \underbrace{C_\varepsilon(\varepsilon)}_{=0} + \sum_{\alpha\in A(s_n)} (\mathrm{Pr}_{\mathcal{S}^\pi}(p)\,\pi(p,\alpha))\ \underbrace{C_\varepsilon(\alpha)}_{=C(\alpha)}\big) + \text{“0”} \\
&= \mathrm{ExpTot}^{\times}_{i} + \sum_{\alpha\in A(s_n)} (\mathrm{Pr}_{\mathcal{S}^\pi}(p)\,\pi(p,\alpha))\, C_\varepsilon(\alpha) \\
&= \mathrm{ExpTot}^{\times}_{i} + \mathrm{Pr}_{\mathcal{S}^\pi}(p) \sum_{\alpha\in A(s_n)} \pi(p,\alpha)\, C(\alpha)
\end{aligned}$$

Together, $\mathrm{ExpTot}_{i+1} = \mathrm{ExpTot}^{\times}_{i+1}$ follows immediately from $\mathrm{ExpTot}_i = \mathrm{ExpTot}^{\times}_i$. The expression “0” in the equations above is meant to stand for the added expected costs into ε-states when going from i to $i+1$. Because $C_\varepsilon(\varepsilon) = 0$ their value is 0. □

Lemma 28 $\sum_{\langle \boldsymbol{m},s\rangle\in S^{\times},\alpha\in A(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, C(\alpha) = \sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$.

Proof. In the proof of Lemma 23 under “σ is a solution of (C1)-(C3)” we concluded $\sigma(\mathrm{in}(\boldsymbol{m}, s_\varepsilon)) = 0$ ($= \lim_{i\to\infty} \sigma(\mathrm{in}(\boldsymbol{m}, s_\varepsilon))$) and $\sigma(x_{\boldsymbol{m},s,\varepsilon}) = 0$ ($= \lim_{i\to\infty} \sigma_i(x_{\boldsymbol{m},s,\varepsilon})$). That is, the ε-actions disappear in the limit. We obtain

$$\begin{aligned}
&\sum_{\langle \boldsymbol{m},s\rangle\in S^{\times},\alpha\in A(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, C(\alpha) \\
&\quad= \sum_{\langle \boldsymbol{m},s\rangle\in S^{\times},\alpha\in A_\varepsilon(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, C_\varepsilon(\alpha) && \text{(as } \sigma(x_{\boldsymbol{m},s,\varepsilon}) = 0 \text{ everywhere)} \\
&\quad= \sum_{\langle \boldsymbol{m},s\rangle\in S^{\times},\alpha\in A_\varepsilon(s)} \lim_{i\to\infty}\sigma_i(x_{\boldsymbol{m},s,\alpha})\, C_\varepsilon(\alpha) && \text{(by definition of } \sigma) \\
&\quad= \lim_{i\to\infty} \sum_{\langle \boldsymbol{m},s\rangle\in S^{\times},\alpha\in A_\varepsilon(s)} \sigma_i(x_{\boldsymbol{m},s,\alpha})\, C_\varepsilon(\alpha) && \text{(limit property)} \\
&\quad= \lim_{i\to\infty} \mathrm{ExpTot}^{\times}_i = \lim_{i\to\infty} \mathrm{ExpTot}_i && \text{(by Lemma 27)} \\
&\quad= \lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) && \text{(by definition of } \mathrm{ExpTot}_i) \\
&\quad= \sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) && \text{(see text)}
\end{aligned}$$

The last equality needs an explanation. In fact, we need several facts:

(1) $\lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$ is a finite value. This follows from the equalities above and the fact that $\sigma(x_{\boldsymbol{m},s,\alpha})$ is a finite value, for all $\langle \boldsymbol{m}, s\rangle \in S^{\times}$ and $\alpha \in A_\varepsilon(s)$ (cf. “Definition of σ” above).

(2) $\sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$ is a finite value. Proof: Enumerate $\mathit{Paths}(\mathbf{F}\,G) = \{p_1, p_2, \ldots\}$. Suppose, by way of contradiction, that claim (2) is false. Then, for every $\iota > 0$ there is a $k \geq 0$ such that $\sum_{i=1}^{k} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p_i) > \iota$. By construction, each of the paths $\{p_1, \ldots, p_k\}$ will be contained eventually in the set $\mathcal{P}_K$, for some $K \geq 1$. It follows $\sum_{p\in\mathcal{P}_K} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \iota$. Together with monotonicity (Lemma 26) it follows that $\lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$ converges to infinity, a plain contradiction to (1).

(3) $\lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) \leq \sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$. Proof: suppose this is not the case. With (2) then $\lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$. Choose $K \geq 1$ big enough such that $\mathcal{P}_K \subset \mathit{Paths}(\mathbf{F}\,G)$ and $\sum_{p\in\mathcal{P}_K} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \sum_{p\in \mathit{Paths}(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$. Such a set $\mathcal{P}_K$ must exist by monotonicity (Lemma 26) and fairness of the construction. Again by monotonicity

conclude $\sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \sum_{p\in\mathcal{P}_K} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$ ($> \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$), a plain contradiction.

Now we turn to the last equality $\lim_{i\to\infty} \sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) = \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p)$. With (1) and (3) established, choose $\varepsilon > 0$ arbitrarily and let $\{p_1, \ldots, p_k\} \subset Paths(\mathbf{F}\,G)$ be any set of goal paths such that $\sum_{i=1}^{k} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p_i) > \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) - \varepsilon$. As in (2) above, there is a $K \geq 1$ such that $\{p_1, \ldots, p_k\} \subseteq \mathcal{P}_K$. With monotonicity (Lemma 26) it follows, for all $i \geq K$, $\sum_{p\in\mathcal{P}_i} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) > \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) - \varepsilon$. □

Lemma 29 $\displaystyle\sum_{\langle \boldsymbol{m}, s\rangle \in G^\times} \sigma(\mathrm{in}(\boldsymbol{m}, s))\, T(s) = \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpT}_{\mathcal{S}^\pi}(p).$

Proof. Compute

$$
\begin{aligned}
&\sum_{\langle \boldsymbol{m}, s\rangle \in G^\times} \sigma(\mathrm{in}(\boldsymbol{m}, s))\, T(s) \\
&= \sum_{s\in G} \sum_{\boldsymbol{m}\in M} \sigma(\mathrm{in}(\boldsymbol{m}, s))\, T(s) \\
&= \sum_{s\in G} \Pr_{\mathcal{S}^\pi}(\mathbf{F}\,s)\, T(s) && \text{(by Lemma 23-(1))} \\
&= \sum_{p\in Paths(\mathbf{F}\,s) \text{ s.th. } s\in G} \Pr_{\mathcal{S}^\pi}(p)\, T(s) && \text{(Def. of } \Pr_{\mathcal{S}^\pi}(\mathbf{F}\,s)\text{)} \\
&= \sum_{p\in Paths(\mathbf{F}\,G)} \Pr_{\mathcal{S}^\pi}(p)\, T(s) \\
&= \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpT}_{\mathcal{S}^\pi}(p)
\end{aligned}
$$

Finally:

Proof. (of Lemma 24)

$$
\begin{aligned}
&\sum_{\langle \boldsymbol{m}, s\rangle \in S^\times, \alpha\in A(s)} \sigma(x_{\boldsymbol{m},s,\alpha})\, C(\alpha) \;+\; \sum_{\langle \boldsymbol{m}, s\rangle \in G^\times} \sigma(\mathrm{in}(\boldsymbol{m}, s))\, T(s) \\
&= \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpTot}_{\mathcal{S}^\pi}(p) \;+\; \sum_{p\in Paths(\mathbf{F}\,G)} \mathrm{ExpT}_{\mathcal{S}^\pi}(p) && \text{(by Lemma 28 and Lemma 29)} \\
&= \sum_{s_1\cdots s_n\in Paths(\mathbf{F}\,G)} \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} C(q)\, P(q|\pi) \;+\; \sum_{s_1\cdots s_n\in Paths(\mathbf{F}\,G)} \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} T(s_n)\, P(q|\pi) \\
&\qquad \text{where } q = s_1 \xrightarrow{\alpha_1} s_2 \cdots s_{n-1} \xrightarrow{\alpha_{n-1}} s_n \\
&= \sum_{s_1\cdots s_n\in Paths(\mathbf{F}\,G)} \sum_{\alpha_1\in A(s_1),\ldots,\alpha_{n-1}\in A(s_{n-1})} (C(q) + T(s_n))\, P(q|\pi) \\
&= \sum_{r\in GRuns(s_{\mathrm{init}},\pi)} (C(r) + T(\mathrm{last}(r)))\, P(r|\pi) \\
&= V^\pi(s_{\mathrm{init}}) = V^\pi
\end{aligned}
$$

□

A.7.3 Completeness Theorem

Theorem 6 (Completeness) *If π is an unrestricted proper policy for $\mathcal{S}$ such that $\mathcal{S}, \pi \models \varphi$ then π^{prog} exists and is a proper policy such that $\pi^{\mathrm{prog}} \models \varphi$ and $V^{\pi^{\mathrm{prog}}} \leq V^\pi$.*

Proof. Suppose an unrestricted proper policy π for $\mathcal{S}$ such that $\mathcal{S}, \pi \models \varphi$. Among all such policies there is also an optimal one, π^*, which we use in the following.

Lemma 23 applied to π^* gives us a solution σ of $\mathrm{LP}^-_{C^\times}$, i.e., the constraints (C1)-(C3) of $\mathrm{LP}_{C^\times}$. First we show that σ also satisfies (C4) and (C5alt).

Regarding (C4) we have

$$\begin{aligned}
1 &= \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, G) && \text{(by Lemma 19)}\\
&= \sum_{s \in G} \Pr_{\mathcal{S}^\pi}(\mathbf{F}\, s)\\
&= \sum_{s \in G} \sum_{\boldsymbol{m} \in \boldsymbol{M}} \sigma(\mathrm{in}(\boldsymbol{m}, s)) && \text{(by Lemma 23-(1))}\\
&= \sum_{\langle \boldsymbol{m}, s\rangle \in G^\times} \sigma(\mathrm{in}(\boldsymbol{m}, s))
\end{aligned}$$

This is just (C4) (with the solution σ made explicit).

Regarding (C5alt) recall the objective formula is written as $\varphi = \bigwedge_{i=1}^{k} \mathbf{P}_{\in z_i}\, \psi_i$. Choose $i \in 1..k$ arbitrarily. By Lemma 23-(2) then

$$\Pr_{\mathcal{S}^\pi}(\psi_i) = \sum_{\langle \boldsymbol{m}, s\rangle \in \mathrm{Accept}_i} \sigma(\mathrm{in}(\boldsymbol{m}, s)) \ . \tag{4}$$

From $\mathcal{S}, \pi \models \varphi$ it follows $\mathcal{S}, \pi \models \mathbf{P}_{\in z_i}\, \psi_i$, equivalently $\Pr_{\mathcal{S}^\pi}(\psi_i) \in z_i$. With (4) it follows immediately that σ satisfies (C5alt). Altogether, hence, σ is a solution of (C1)-(C5alt), equivalently, of (C1)-(C5).

By Lemma 24

$$V^{\pi^*} = \sum_{\langle \boldsymbol{m}, s\rangle \in S^\times, \alpha \in A(s)} \sigma(x_{\boldsymbol{m}, s, \alpha})\, C(\alpha) \ + \sum_{\langle \boldsymbol{m}, s\rangle \in G^\times} \sigma(\mathrm{in}(\boldsymbol{m}, s))\, T(s) = \sigma(e)$$

where e is the expression to be minimized in the constraint (LP1).

The next step is to show that σ indeed minimizes e. Suppose, by way of contradiction, this is not the case. Then there is a different solution σ' of $\mathrm{LP}_{C^\times}$ with costs $\sigma'(e) < \sigma(e)$. With the corresponence between $\mathrm{LP}_{C^\times}$ and $C^\times$ (Theorem 4) and the same arguments as in the beginning of the soundess theorem (Theorem 5, the part that does note refer to optimality) that solution σ' defines an optimal policy π' for $\mathcal{S}$ with $V^{\pi'} = \sigma'(e) < \sigma(e) = V^{\pi^*}$. This contradits optimality of π^*.

Alltogether, as σ minimizes e and satisfies (C1)-(C5), σ is a solution of $\mathrm{LP}_{C^\times}$.

With this result established, we can again exploit the correspondence between $\mathrm{LP}_{C^\times}$ and $C^\times$. It follows that $\pi^*_{C^\times}$ is a proper (because of constraint (C4)) and optimal policy solution of $C^\times$.

It follows that $\pi^{\mathrm{prog}} = (\boldsymbol{M}^{\mathrm{prog}}, \mathbf{start}^{\mathrm{prog}}, \mathbf{mod}^{\mathrm{prog}}, \pi^*_{C^\times})$ exists. Moreover, with $\pi^*_{C^\times}$ being proper and optimal, so is π^{prog} (π^{prog} is just a reformulation of $\pi_{C^\times}$ as a finite-memory policy for $\mathcal{S}$). Because π^{prog} is optimal it follows $V^{\pi^{\mathrm{prog}}} \leq V^\pi$, the last open claim. □

A.8 Admissibility of the NBA Projection Heuristic

Theorem 31 *$h^{BA}_{\psi_i}$ is an admissible heuristic for ψ_i.*

Proof. By definition of admissible heuristic, $h^{\mathrm{BA}}_{\psi_i}$ is admissible if $h^{\mathrm{BA}}_{\psi_i}(\chi_{\mathrm{start}}) \geq \max_{\langle s, \chi_{\mathrm{start}}\rangle \in S^\times, \pi \in \Pi^*} V^\pi_i(\langle s, \chi_{\mathrm{start}}\rangle)$. We prove this by showing that the runs in the relaxed SSP $\mathcal{S}^{\psi_i}$ contains all the words accepted by the NBA $\mathcal{B}_i$.

Suppose that $s_1 s_2 \cdots s_g s_g s_g \cdots$ is accepted by the NBA $\mathcal{B}_i$ using χ_{start} as initial state. Then there exists an exhaustive run of $s_1 \xrightarrow{\alpha_1} s_2 \xrightarrow{\alpha_2} s_3 \cdots s_g$ of $\mathcal{S}$ and an infinite path $q_{\mathrm{start}} \xrightarrow{s_1} q_1 \xrightarrow{s_2} q_2 \cdots$ of

$\mathcal{B}_i$ that visits an accepting state $q_f \in F_i$ infinitely often for $q_{\text{start}} \in \chi_{\text{start}}$. By definition of exhaustive run, we have that $\alpha_j \in A(s_j)$ and $P(s_{j+1}|s_j, \alpha_j) > 0$ for $j \geq 1$. Thus, $pre(\alpha_j) \subseteq s_j$ and there exists $e_j \in \mathit{eff}(\alpha_j)$ s.t. $res(s_j, e_j) = s_{j+1}$. Therefore s_j is consistent with e_j, $q_{j+1} \in \text{Comp}(e_j, q_j)$, and there exists $\beta_j \in B^{\psi_i}$ s.t. $P^{\psi_i}(q_{j+1}|q_i, \beta_i) > 0$ for $j > 0$. Moreover, $q_{\text{start}} \xrightarrow{\beta_1} q_1 \xrightarrow{\beta_2} q_2 \cdots$ is a run of the relaxed SSP $\mathcal{S}^{\psi_i}$. Since, the state $q_f \in F_i$ is visited infinitely often (Büchi condition) and F_i is also the goal set of $\mathcal{S}^{\psi_i}$, we have that $q_{\text{start}} \xrightarrow{\beta_1} q_1 \xrightarrow{\beta_2} q_2 \cdots q_f$ is an exhaustive run and reaches the goal of $\mathcal{S}^{\psi_i}$, therefore the flow from χ_{start} reaches F_i representing that the word $s_1 s_2 \cdots s_g s_g s_g \cdots$ is accepted by $h^{\text{BA}}_{\psi_i}$.

References

[Alt99] Eitan Altman. *Constrained Markov Decision Processes*, volume 7. CRC Press, 1999.

[AW13] Noran Azmy and Christoph Weidenbach. Computing Tiny Clause Normal Forms. In *Proc. Automated Deduction - CADE-24*, volume 7898 of *LNCS*. Springer, 2013.

[Bac92] Christer Backström. Equivalence and Tractability Results for SAS$^+$ Planning. In *Proc. Int. Conf. on Principles of Knowledge Representation and Reasoning*, 1992.

[BBS95] Andrew Barto, Steven Bradtke, and Satinder Singh. Learning to Act Using Real-Time Dynamic Programming. *Artif. Intell.*, 72(1-2):81–138, 1995.

[BG03] Blai Bonet and Hector Geffner. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2003.

[BG05] Blai Bonet and Hector Geffner. mGPT: A Probabilistic Planner Based on Heuristic Search. *J. Artif. Intell. Res.*, 24:933–944, 2005.

[BG12] Blai Bonet and Hector Geffner. Action Selection for MDPs: Anytime AO* Versus UCT. In *Proc. AAAI Conf. on Artificial Intelligence*, 2012.

[BH10] Andreas Bauer and Patrik Haslum. LTL Goal Specifications Revisited. In *Proc. Int. Conf. on Artificial Intelligence*, 2010.

[BK98] Fahiem Bacchus and Froduald Kabanza. Planning for Temporally Extended Goals. *Ann. Math. Artif. Intell.*, 22(1-2):5–27, 1998.

[BK08] C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.

[BM06] Jorge A. Baier and Sheila A. McIlraith. Planning with First-Order Temporally Extended Goals using Heuristic Search. In *Proc. AAAI Conf. on Artificial Intelligence*, 2006.

[CTM$^+$17] Alberto Camacho, Eleni Triantafillou, Christian J. Muise, Jorge A. Baier, and Sheila A. McIlraith. Non-Deterministic Planning with Temporally Extended Goals: LTL over Finite and Infinite Traces. In *Proc. AAAI Conf. on Artificial Intelligence*, 2017.

[CY95] C. Courcoubetis and M. Yannakakis. The Complexity of Probabilistic Verification. *J. ACM*, 42(4):857–907, 1995.

[DD05] Dmitri A. Dolgov and Edmund H. Durfee. Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005.

[D'E63] F. D'Epenoux. A probabilistic production and inventory problem. *Management Science*, 10:98–108, 1963.

[DSBR14] X. C. Ding, S. Smith, C. Belta, and D. Rus. Optimal Control of Markov Decision Processes With Linear Temporal Logic Constraints. *IEEE Trans. Automat. Contr.*, 59(5):1244–1257, 2014.

[DV13] Giuseppe De Giacomo and Moshe Y. Vardi. Linear Temporal Logic and Linear Dynamic Logic on Finite Traces. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2013.

[DV15] Giuseppe De Giacomo and Moshe Y. Vardi. Synthesis for LTL and LDL on Finite Traces. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2015.

[Ede06] Stefan Edelkamp. On the Compilation of Plan Constraints and Preferences. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2006.

[EKVY08] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-Objective Model Checking of Markov Decision Processes. *Logical Methods in Computer Science*, 4(4), 2008.

[FKNP11] V. Forejt, M. Z. Kwiatkowska, G. Norman, and D. Parker. Automated Verification Techniques for Probabilistic Systems. In *Proc. Int. School on Formal Methods for the Design of Computer*, 2011.

[HZ01] Eric A Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1):35–62, 2001.

[KE12] Thomas Keller and Patrick Eyerich. PROST: Probabilistic Planning Based on UCT. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2012.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Proc. Int. Conf. on Computer Aided Verification*, 2011.

[KP13] M. Z. Kwiatkowska and D. Parker. Automated Verification and Strategy Synthesis for Probabilistic Systems. In *Proc. Int. Symp. on Automated Technology for Verification and Analysis*, 2013.

[LPH15] Bruno Lacerda, David Parker, and Nick Hawes. Optimal Policy Generation for Partially Satisfiable Co-Safe LTL Specifications. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2015.

[LPH17] Bruno Lacerda, David Parker, and Nick Hawes. Multi-Objective Policy Generation for Mobile Robots under Probabilistic Time-Bounded Guarantees. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2017.

[SKT14] J. Sprauel, A. Kolobov, and F. Teichteil-Königsbuch. Saturated Path-Constrained MDP: Planning under Uncertainty and Deterministic Model-Checking Constraints. In *Proc. AAAI Conf. on Artificial Intelligence*, 2014.

[TB15] Jorge Torres and Jorge A. Baier. Polynomial-Time Reformulations of LTL Temporally Extended Goals into Final-State Goals. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2015.

[TGS+06] Sylvie Thiébaux, Charles Gretton, John K. Slaney, David Price, and Froduald Kabanza. Decision-Theoretic Planning with non-Markovian Rewards. *J. Artif. Intell. Res.*, 25:17–74, 2006.

[TKVI11] F. Teichteil-Königsbuch, V. Vidal, and G. Infantes. Extending Classical Planning Heuristics to Probabilistic Planning with Dead-Ends. In *Proc. AAAI Conf. on Artificial Intelligence*, 2011.

[Tse68] G.S. Tseitin. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic*, 8:115–125, 1968.

[TTH17] F. Trevizan, S. Thiébaux, and P. Haslum. Occupation Measure Heuristics for Probabilistic Planning. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2017.

[TTKT17] Felipe W. Trevizan, Florent Teichteil-Königsbuch, and Sylvie Thiébaux. Efficient solutions for Stochastic Shortest Path Problems with Dead Ends. In *Proc. Conf. on Uncertainty in Artificial Intelligence*, August 2017.

[TTSW16] Felipe W. Trevizan, Sylvie Thiébaux, Pedro Henrique Santana, and Brian C. Williams. Heuristic Search in Dual Space for Constrained Stochastic Shortest Path Problems. In *Proc. Int. Conf. on Automated Planning and Scheduling*, 2016.