Operator Bellman Equations Support Compositional & Verifiable Planning in Hierarchical World-models

Anonymous authors
Paper under double-blind review

Abstract

We introduce new reward-free Bellman Equations called *Operator Bellman Equations* which, rather than value functions, produce predictive planning representations called *state-time feasibility functions* (STFFs) which are compositional, factorizable, and interpretable. This means: 1) STFFs can be sequentially composed to compute high-dimensional predictions over long-horizons of sequential Options (policies), 2) high-dimensional STFFs can be represented and computed efficiently in a factorized form, and 3) STFFs record the probabilities of semantically interpretable goal-success and constraint-violation *events*. We discuss how these properties are critical for verifiable planning that can scale to dynamic high-dimensional world-models.

Consider an agent in Fig. 1, minimally constituted as a set of coupled transition systems: a grid-world space $P_x(x'|x,a,t)$, a hydration space $P_y(y'|y,\alpha_y)$ with conditioning variables $\alpha_y \in \mathcal{A}_y = \{\alpha_{hyd}, \alpha_\epsilon\}$ which hydrate or dehydrate the agent, respectively; and a logical space $P_\sigma(\boldsymbol{\sigma}'|\boldsymbol{\sigma},\alpha_\sigma)$, where $\boldsymbol{\sigma}$ is a binary vector, and $\alpha_\sigma \in \mathcal{A}_\sigma = \{\alpha_\epsilon, \alpha_1, \alpha_2, \alpha_3\}$ is a variable that flips bits (α_2 flips bit 2, $(1,0,0) \xrightarrow{\alpha_2} (1,1,0)$, α_ϵ does nothing). The full Cartesian product-space dynamics are defined as:

$$P_s(\boldsymbol{\sigma}', y', x'|\boldsymbol{\sigma}, y, x, a, t) = \sum_{\alpha_\sigma, \alpha_y} P_\sigma(\boldsymbol{\sigma}'|\boldsymbol{\sigma},\alpha_\sigma) P_y(y'|y,\alpha_y) F(\alpha_\sigma,\alpha_y|x,a) P_x(x'|x,a,t), \tag{1}$$

where F is an *affordance function* that specifies a causal coupling between low- and high-level dynamics, e.g. an agent can drive the y-dynamics by taking action a_{drink} at x_{lake} to induce α_{hyd}.

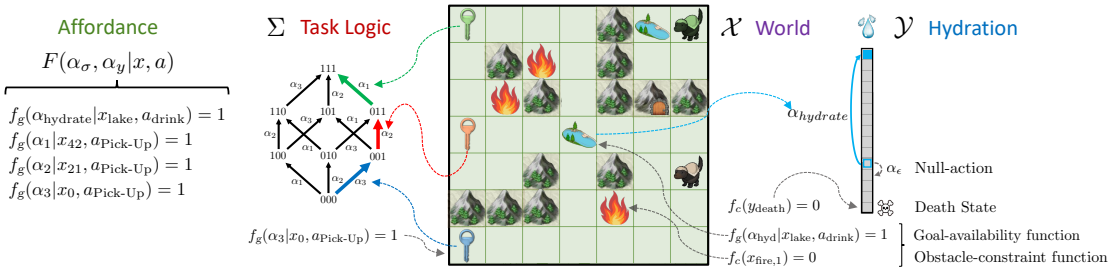

Figure 1: The agent must regulate internal states, find 3 keys to unlock the door, and meet a friend.

When optimizing a policy with P_s, reward functions are a problem because they are difficult to define in high-dimensions and they link an agent's representations to a fixed normative quantity, e.g. value functions on a product-space are brittle if the reward- or world-model changes. However, reality is dynamic, new systems can become known for an agent to control, and different goals and constraints may become relevant. States in an internal *need space* $\mathcal{Y}$ could make logic states in Σ and low-level goals on $\mathcal{X}$ salient. Agents need to flexibly reason about solutions to new complex goals and constraints (i.e. *tasks*) (Lake et al., 2017), and the feasibility that they can be satisfied across high-dimensional world-models. This is especially true if the agent is deployed in a safety-critical context which requires verification that task specifications are met (Dalrymple et al., 2024). Given recent discussion

about the generality of reward-maximization to subserve a theory of general intelligence and express notions of goal and purpose (Sutton & Barto, 1998; Silver et al., 2021; Vamplew et al., 2022; Skalse & Abate, 2023; Bowling et al., 2023; Ringstrom, 2022; Abel et al., 2021), it is important to critically investigate whether reward-maximization actually facilitates the properties of interpretability and compositionality characteristic of general intelligence and essential for verification. In this paper, we advocate for eliminating the reward function because reward maximization *destroys* interpretability. Instead, we let goals, constraints, and the world-model dictate the optimization of flexible predictive planning representations and policies, compatible with the Options framework in RL (Sutton et al., 1999). Expanding on Ringstrom (2023), we develop new Markov Decision Processes (MDPs) and reward-free Bellman Equations (Bertsekas, 2012; Bellman, 1957) that produce *feasibility functions* which propagate information about an Option's goal and constraint satisfaction events across a hierarchical world-model for verifiable, high-dimensional planning. Our work differs from Ringstrom in that we introduce constraint functions, an improved failure STFF, and use Options explicitly.

1 The Task Markov Decision Process

We start by introducing a decision process called the Task MDP, where *tasks* are goals and constraints:

Definition 1.1 (Task MDP (TMDP))**.** *A TMDP is defined as* $\mathscr{M} = (\mathcal{X}, \mathcal{A}, \mathcal{T}, P_x, f_g, f_c)$*, where* $\mathcal{X}$ *is a set of discrete states,* $\mathcal{A}$ *is a set of discrete actions,* $\mathcal{T}$ *is a set of discrete times,* $P_x : (\mathcal{X} \times \mathcal{A} \times \mathcal{T}) \times \mathcal{X} \to [0, 1]$ *is a transition operator,* $f_g : (\mathcal{X} \times \mathcal{A} \times \mathcal{T}) \times \{\alpha_g\} \to [0, 1]$ *is the goal-availability function, and* $f_c : \mathcal{X} \times \mathcal{A} \times \mathcal{T} \to [0, 1]$ *is the obstacle-constraint function.*

Before giving the Bellman equations for the TMDP, we discuss some important functions.

Goal-availability Function: The *goal availability function*, $f_g(\alpha_g|x, a, t)$, specifies the probability that a singleton *goal-action* α_g is available to be *caused* by a low-level state, action, and time (α is *not* a free variable in an optimization like an action "a"). The function f_g is not like an arbitrary reward model and it is not a full probability distribution like F in equation (1), rather, it is directly derived from F, and will take the form $f_g(\alpha_g|x, a, t) = F(\alpha_g|x, a, t)$ for one of N α_g variables in F. Thus, like we could solve for a family of regular MDPs for N reward functions $\{R_{g_1}, ..., R_{g_N}\}$, we can also solve a family of N individual TMDPs $\{f_{g_1}, ..., f_{g_N}\}$ (indexed by goals $\mathcal{G} = \{g_1, ..., g_N\}$) when we create sets of Options. We will often write the function as $f_{g_i}(x, a, t)$, and drop α_g for simplicity.

We call goal-availability functions which are a functions of actions, *action-dependent* ($f_g(\alpha_g|x, a, t), f_g(\alpha_g|x, t)$, and $f_g(x, a, t)$), and ones that are not function of actions are *action-independent* ($f_g(x, t)$). As we will explain in section 2.4, these distinctions are important when sequentially chaining Option-policies together because action-dependent functions commit the agent to an action at the terminal state, requiring a state-update before another Option can be initiated.

Obstacle Constraint Function: We also defined an obstacle constraint function f_c, where a constraint is violated with probability $1 - f_c(x, a, t)$ (zeros encode hard constraints). We will not use aleatoric probabilities for obstacles. We consider *goals* to be achievement conditions, and *tasks* to be goal-achievement conditions and task-constraints, which can be potentially violated (see Fig. 1). We define f_c assuming that the set of goal state-time-actions is disjoint with the set of state-time-action obstacles, so that a task can't be simultaneously achieved and violated.

State-time Feasibility Function: We now discuss the concept of feasibility, which we will be optimizing in our Bellman equations. Let $\mathbf{xt} = ((x_{t_0}, t_0), ..., (x_{T_f}, T_f))$ be a state-time trajectory over $\mathcal{X}$. We can calculate the *feasibility* that a sub-trajectory of $\mathbf{xt}$ satisfies the task by choosing starting state and time $(x_s, t_s) \in \mathbf{xt}$ and final state-time $(x_f, t_f) \in \mathbf{xt}$. Since task-completion is an event, we can introduce event logic for a given sub-trajectory of $\mathbf{xt}$. Consider a Bernoulli R.V. indicating a goal-success event, $S \sim Bern(f_g(x, t))$, and a Bernoulli R.V. for constraint-violation events, $V \sim Bern(1 - f_c(x, t))$. A task is completed if (S^+, V^-) (using capitalized realizations to avoid notational conflict), and is uncompleted if (S^-, V^-). We define a state-time feasibility function (STFF) $\eta^+ : (\mathcal{X} \times \mathcal{T}) \times (\mathcal{X} \times \mathcal{T}) \to [0, 1]$, which outputs the probability that the first task-success event is at state-time (x_f, t_f) without a preceding failure event (S^-, V^+) when starting

from (x_s, t_s): $\eta^+_{\mathbf{xt}}(x_f, t_f | x_{t_s}, t_s) = Pr((S^-_{t_s}, V^-_{t_s}), (S^-_{t_s+1}, V^-_{t_s+1}), ..., (S^-_{t_f-1}, V^-_{t_f-1}), (S^+_{t_f}, V^-_{t_f}))$ This is the logical form of η. We can express this with f_g and f_c, which leads to a recursive form in equation (3). Defining $f_1 = f_g f_c$ and $f_2 = (1 - f_g) f_c$, we have:

$$\begin{aligned}\eta^+_{\mathbf{xt}}(x_f, t_f | x_{t_s}, t_s) &= \left(\prod_{\tau=t_s}^{t_f-1}(1 - f_g(x_\tau, \tau)) f_c(x_\tau, \tau)\right) f_g(x_f, t_f) f_c(x_f, t_f), \\ &= \left(\prod_{\tau=t_s}^{t_f-1} f_2(x_\tau, \tau)\right) f_1(x_f, t_f) = f_2(x_{t_s}, t_s) \left(\prod_{\tau=t_s+1}^{t_f-1} f_2(x_\tau, \tau)\right) f_1(x_f, t_f), \quad (2)\\ \eta^+_{\mathbf{xt}}(x_f, t_f | x_{t_s}, t_s) &= f_2(x_{t_s}, t_s) \eta^+_{\mathbf{xt}}(x_f, t_f | x_{t_s+1}, t_s + 1), \text{ where: } \eta^+_{\mathbf{xt}}(x_f, t_f | x_f, t_f) = f_1(x_f, t_f) \quad (3)\end{aligned}$$

Achievement and Continuation Functions: We combined the goal-availability and obstacle-constraint functions into two different functions: the *Achievement Function,* $f_1(x, a, t) = f_g(x, a, t) f_c(x, a, t)$ and the *Continuation Function,* $f_2(x, a, t) = (1 - f_g(x, a, t)) f_c(x, a, t)$. Here we show this with actions for generality. The achievement function captures both goal-success termination events and the absence of a constraint-violation termination event, whereas the continuation function represents the absence of both goal-success and constraint-violation events (i.e. the agent can *continue* the task). It is important to understand that while goal-success and constraint-violation events will terminate the use of a control policy, so too will the event of the agent entering a state-time from which the goal is known to be infeasible, which is policy-dependent, depending on the feasibility of a goal under a policy (represented by κ, defined in the next subsection). This *third* policy termination condition will be found in the Operator Bellman Equations (equations (9,11)).

Cumulative Feasibility Function: Summing over x_f and t_f gives us the total cumulative probability of achieving the goal while not violating the constraints. This is represented by $\kappa : \mathcal{X} \times \mathcal{T} \to [0, 1]$, the cumulative feasibility function (CFF) (4). We can substitute the R.H.S. of (2) into (4), and with some additional manipulations (not shown) we obtain a recursion for κ in (5):

$$\kappa_{\mathbf{xt}}(x_{t_s}, t_s) = \textstyle\sum_{t_f=t_s}^{T_f} \sum_{x_{t_f}} \eta^+_{\mathbf{xt}}(x_{t_f}, t_f | x_{t_s}, t_s) \quad (4)$$

$$\kappa_{\mathbf{xt}}(x_{t_s}, t_s) = f_1(x_{t_s}, t_s) + f_2(x_{t_s}, t_s) \kappa_{\mathbf{xt}}(x_{t_s+1}, t_s + 1), \quad \text{where: } \kappa_{\mathbf{xt}}(x_{t_f}, t_f) = f_1(x_{t_f}, t_f) \quad (5)$$

We can see that $\kappa_{\mathbf{xt}}$ summarizes all possible events up to T_f that could complete the goal. By having a recursive form of κ and η, we will show next that we can link the recursion to a transition operator P_x to define a Bellman equation. As of now, we computed κ and η on a single trajectory $\mathbf{xt}$, but when we move to a Bellman equation, we will optimize a policy π, and so κ and η will summarize the feasibility over distributions of trajectories induced by a policy. This will be important for stitching together policies, because the STFF distributions for policies will compose with each other.

1.1 Operator Bellman Equations

Now we introduce a transition operator to our goal-availability and constraint functions and compute the policy which maximizes goal-feasibility by maximizing over the actions. These equations are called the Operator Bellman Equations (OBEs), defined with the TMDP $\mathscr{M} = (\mathcal{X}, \mathcal{A}, \mathcal{T}, P_x, f_g, f_c)$:

Policy Optimization: Optimize cumulative goal success and time-minimization,

$$\kappa^*_g(x, t) = \max_a \left[f_1(x, a, t) + f_2(x, a, t) \sum_{x'} P_x(x' | x, a, t) \kappa^*_g(x', t + 1) \right], \quad (6)$$

$$\pi^{**}_g(x, t) = \operatorname*{argmin}_{a \in \mathcal{A}^*_{x,t}} \left[t f_1(x, a, t) + f_2(x, a, t) \mathop{\mathbb{E}}_{x' \sim P^a_{xt}} \sum_{+} t_+ \eta^+_g(\alpha_+, a_+, x_+, t_+ | x', t + 1) \right], \quad (7)$$

Prediction Functions: Record of where and when success and failure events occur,

$$\eta^+_g(\alpha_g, a_+, x_+, t_+ | x, t) = f_2(x, a^\pi_{xt}, t) \sum_{x'} P_x(x' | x, a^\pi_{xt}, t) \eta^+_g(\alpha_g, a_+, x_+, t_+ | x', t + 1), \quad (8)$$

$$\eta^-_g(\alpha_-, a_-, x_-, t_- | x, t) = \mathbb{1}_\kappa(x, t) \sum_{x'} P_x(x' | x, a^\pi_{xt}, t) \eta^-_g(\alpha_-, a_-, x_-, t_- | x', t + 1), \quad (9)$$

where $a^{\pi}_{xt} = \pi^{**}_g(x,t)$, and $\mathbb{1}_\kappa(x,t) = \{1 \text{ if: } \kappa^*(x,t) > 0;\ 0 \text{ if: } \kappa^*(x,t) = 0\}$, is the feasibility indicator function, outputting 1 if the task is feasible from (x,t), 0 if not. Like equation (3), the above η-OBEs are defined for all $t_+, t_- \in [t+1 : T_f]$, and if $t = t_+$ and $t = t_-$, then $\eta^+_{g_i}$ and $\eta^-_{g_i}$ are defined as,

$$\eta^+_g(\alpha_+, a^{\pi}_{xt}, x, t|x,t) = f_g(x, a^{\pi}_{xt}, t), \tag{10}$$

$$\eta^-_g(\alpha_-, a^{\pi}_{xt}, x, t|x,t) = \mathbb{1}_\kappa(x,t)(1 - f_c(x, a^{\pi}_{xt}, t)) + \bar{\mathbb{1}}_\kappa(x,t), \tag{11}$$

where $\bar{\mathbb{1}}_\kappa(x,t) = |1 - \mathbb{1}_\kappa(x,t)|$ is the infeasibility indicator function, outputting 1 if the task is infeasible at (x,t), and 0 if not. Note that $+$ and $-$ are task success and failure tags where tasks are goals *and* constraints, and α_g can be written as α_+, but we use g to emphasize a specific goal-variable we are optimizing for. In equation (9), the term $\mathbb{1}_\kappa(x,t)$ makes the probability of *future* failure state-times (x_-, t_-) equal to 0 if $\kappa^*_g(x,t) = 0$ because the failure occurs now at (x,t), as covered in (11). But, if the goal is currently feasible, $\mathbb{1}_\kappa(x,t) = 1$, and the final failure spatiotemporal distribution for $\eta^-_{g_i}$ backward-propagates to the current time-step through transition dynamics P_x. The functions κ and η in the OBEs are undefined past T_f, and naturally we assume they default to 0 in the OBEs (see Appx.1 for horizon conditions). For the STFF $\eta(\alpha_f, a_f, x_f, t_f|x,t)$ we will often omit the final action a_f because it is directly determined by the policy, state, and time. The OBEs can be solved with a dynamic programming algorithm called feasibility iteration (see algorithm 1).

Equations (6) and (**??**)-(9) are extensions of (5) and (3), respectively, but now with transition operators. In the κ-OBE $\kappa(x,t)$ has the logical interpretation: the probability of completing the goal `AND NOT` violating the obstacle constraint now, `OR` $(+)$, `NOT` completing the goal `AND NOT` violating the constraint now, `AND` completing the goal while `NOT` violating the constraint in the future. η-OBEs preserve *event* probabilities, and are therefore interpretable; the STFF, η^+_g, records when and where goals are satisfied, and η^-_g records when and where constraints are violated—this is *essential* for verification.

In the OBEs, $a^{\pi}_{xt} = \pi^{**}_g(x,t)$ where two stars, $**$, indicate optimality with respect to cumulative feasibility *and* expected time minimization conditioned on optimal cumulative feasibility. The set $\mathcal{A}^*_{x,t}$ under the argmin is the set of maximizing arguments from equation (6) used for time-minimization. Note that the formula inside the min-function of the π-OBE, equation (7), is simply the expectation over final success times t_f in two parts, $t_f = t$ and $t_f \in [t+1 : T_f]$, so optimizing over $\mathcal{A}^*_{x,t}$ minimizes expected time conditioned on maximizing cumulative feasibility.

1.1.1 Relationship between κ and η

The CFF and the partial STFFs are simply related through summation, (proof is given in .2):

$$\kappa^*_g(x,t) = \textstyle\sum_{\alpha_+, x_+, t_+} \eta^+_g(\alpha_+, x_+, t_+|x,t), \qquad 1 - \kappa^*_g(x,t) = \textstyle\sum_{\alpha_-, x_-, t_-} \eta^-_g(\alpha_-, x_-, t_-|x,t).$$

Since η^+_g and η^-_g share the same domain, and because goal-success and goal-failure events are disjoint, these partial STFFs can be combined into the full STFF function, $\eta^{**}_{\pi_g}$,

$$\eta^{**}_{\pi_g}(\alpha_f, x_f, t_f|x,t) = \eta^+_g(\alpha_f, x_f, t_f|x,t) + \eta^-_g(\alpha_f, x_f, t_f|x,t)$$

which sums to 1: $\sum_f \eta^{**}_{\pi_g}(\alpha_f, x_f, t_f|x,t) = \kappa^*_g(x,t) + 1 - \kappa^*_g(x,t) = 1.$. The subscript f tags a *final* variable, which can be g, $+$ or $-$. Technically, η^{**}_π can be thought of as a transition operator with one action, π, which could be written as $\eta^{**}_\pi(\alpha_f, x_f, t_f|x,t,\pi)$, where π is the only possible conditioning policy. As we will see, we can aggregate an ensemble of state-time feasibility functions along with an associated set of policies for decomposing a high-dimensional OBEs with Options. Notice that $\eta^{**}_{\pi_g}$ expresses a distribution over interpretable termination events, which occur at goal-failure events, constraint-violation events, and goal-success events. Because of this, they are compositional functions: $\eta_{\rho 12} = \eta_{\pi_2} \circ \eta_{\pi_1} : \eta_{\rho 12}(x_{f_2}, t_{f_2}|x,t) = \sum_{x_{f_1}, t_{f_1}} \eta_{\pi_2}(x_{f_2}, t_{f_2}|x_{f_1}, t_{f_1}) \eta_{\pi_1}(x_{f_1}, t_{f_1}|x,t)$, and this will be important for creating Goal Operators for mapping from goal to goal in a product-space.

1.1.2 Formalizing a TMDP Solution as an Option

The Options framework in RL is a form of semi-Markov planning (Sutton et al., 1999). An option $o = (\mathcal{I}, \pi, \beta)$ has a set of initiation conditions $\mathcal{I} \subseteq \mathcal{X}$, a policy π which can only be followed from states $x \in \mathcal{I}$ to a termination event, and a termination function $\beta : \mathcal{X} \to [0, 1]$, returning a probability of a termination event at state $x \in \mathcal{X}$. Options are instruction sets for using a policy. In the case of the TMDP, our formalism can be cast in the Options framework, where the initiation conditions are entire state-space, i.e. $\mathcal{I} = \mathcal{X}$. For termination, in equation (11), the goal-failure event of a policy is deterministic at $\kappa_g(x, t) = 0$, the constraint-violation termination condition is given by f_c, and the goal-success event terminates a policy with the probability given by the goal-availability function f_g in equation (10). Thus, a termination function is defined by κ_g, f_g, and f_c as:

$$\beta_{\kappa_g, f_g, f_c}(x, a, t) = \mathbb{1}_{\kappa_g}(x, t)(f_g(x, a, t) + (1 - f_c(x, a, t))) + \bar{\mathbb{1}}_{\kappa_g}(x, t)$$

For a single task with goal-index g_i, a TMDP option o_{g_i} is defined as:

$$o_{g_i} = (\mathcal{X}, \pi^{**}_{g_i}, \beta_{g_i}) \leftarrow \texttt{create\_option}(\mathcal{X}, \pi^{**}_{g_i}, \kappa^*_{g_i}, f_{g_i}, f_c),$$

using β_{g_i} instead of $\beta_{\kappa_{g_i}, f_{g_i}, f_c}$ for simplicity. In this paper, we will be directly optimizing open-loop meta-policies (sequences of option). In section 2.4, we will use many options to plan though goal-space.

2 Compositional Task Markov Decision Processes

We now formalize the construction of modular, composite product-space transition operators, and then use them to define a high-dimensional version of the TMDP and OBEs.

Affordance Function: We can couple transition operators together to form a product-space operator with an *affordance function*. This function communicates the ability of a low-level state-action-time to cause transformations on another state-space through its action space:

Definition 2.1 (Affordance Function)**.** *An affordance function $F : (\mathcal{X} \times \mathcal{A} \times \mathcal{T}) \times \mathcal{A}_r \to [0, 1]$ is a conditional joint distribution, $F(\boldsymbol{\alpha}|x, a, t) = F(\alpha_w, ..., \alpha_z|x, a, t)$, on a set of high-level conditioning "actions" $\mathcal{A}_r = (\mathcal{A}_w \times ... \times \mathcal{A}_z)$, where $\boldsymbol{\alpha} = (\alpha_w, ..., \alpha_z) \in \mathcal{A}_r$ is a high-level action-vector.*

The affordance function outputs vectors $\boldsymbol{\alpha}$ from $\mathcal{A}_r$ with a given probability. Each component of the vector corresponds to an action from a higher-level transition operators. For example the variable α_w in $\boldsymbol{\alpha} = (\alpha_w, \alpha_y, ...)$ is an action in set $\mathcal{A}_w$ for a transition operator P_w. Note that the affordance function links features vectors $\boldsymbol{\alpha}$, so (e.g.) drinking at a lake feature, will induce hydration dynamics in thirst-space. A null-variable (null-action) α_ϵ, represents the dynamics of a space in the *absence* of an action induced from the low-level state-space (e.g. getting thirstier over time).

We can use the affordance function to link low-level and high-level transition operators, $P_r(\mathbf{r}'|\mathbf{r}, \boldsymbol{\alpha}) = P_\sigma(\boldsymbol{\sigma}'|\boldsymbol{\sigma}, \alpha_\sigma)P_y(y'|y, \alpha_y)$ one for $\mathcal{R} = \Sigma \times \mathcal{Y}$ and one for $\mathcal{X}$, to create a composite transition operator on $\mathcal{R} \times \mathcal{X}$. We compose two transition operators with a composition function, λ_P:

Definition 2.2 (Composition Functions)**.** *A composition function, λ_P, takes two transition operators along with an affordance function F and produces a composite product-space operator P_s,*

$$P_s(\mathbf{r}', x'|\mathbf{r}, x, a, t) = \lambda_P(P_r, F, P_x) := \textstyle\sum_{\boldsymbol{\alpha} \in \mathcal{A}_r} P_r(\mathbf{r}'|\mathbf{r}, \boldsymbol{\alpha})F(\boldsymbol{\alpha}|x, a, t)P_x(x'|x, a, t), \quad (\boldsymbol{\sigma}, y) = \mathbf{r} \in \mathcal{R}$$

We can see an illustration of the composition function in figure 2, where the full factorization of P_s is shown. The definition of λ_P we gave is *unidirectional* because $\mathcal{X}$ influences $\mathcal{R}$ through F but $\mathcal{R}$ does not influence $\mathcal{X}$. There can also have a *bidirectional* compositions where both state-spaces influence each other, not found in this paper. We are now ready to introduce a Compositional MDP.

We now define a Compositional Task MDP with the product-space transition operator P_s:

Definition 2.3 (Compositional Task MDP)**.** *A Compositional Task MDP (CTMDP) is a TMDP $\bar{\mathscr{M}} = (\mathcal{X} \times \mathcal{R}, \mathcal{A}_x, P_s, \bar{f}_g, \bar{f}_c)$, where $P_s = \lambda_P(P_r, F, P_x)$ is the product-space operator.*

The problem with the CMDP is that it is of size $\mathcal{X} \times \mathcal{R}$, but we now show how to decompose it into a set of goal-conditioned Options and STFFs which are derived in part from the affordance function F.

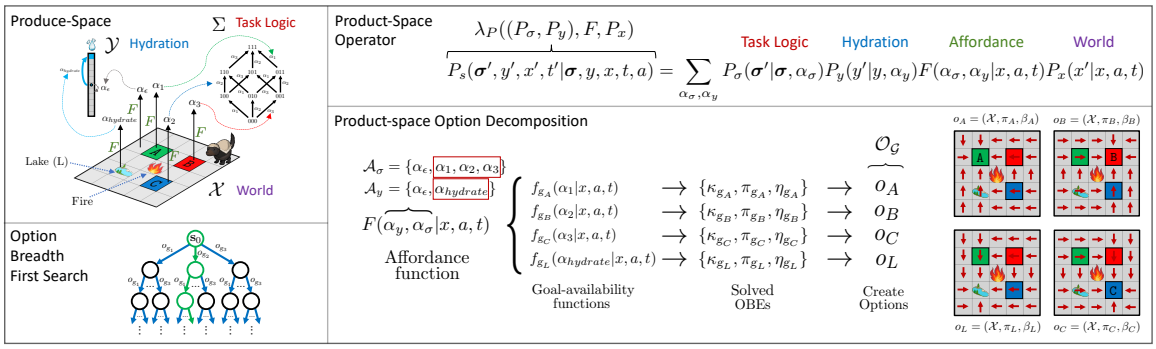

Figure 2: The affordance function F links the base-space transition operators P_x with the logical task-space P_σ and hydration space P_y. The null-action α_ϵ variable causes the agent to become thirstier over time, whereas α_{hyd} makes the agent fully hydrated by drinking at the lake (L). We can decompose F into a set of goal-availability functions f_g. These functions can then be used to solve OBEs and create feasibility functions and policies defining a set of goal-conditioned options: $\mathcal{O}_\mathcal{G} = \{o_A, o_B, o_C, o_L\}$. Breadth first search can solve for optimal sequences of options.

2.1 Time-to-Constraint-Violation

A necessary piece of information an agent needs to plan with *high-level constraints* is the time until a constraint violation event occurs in another space. For instance, our agent might die of dehydration if it tries an ambitions journey. Fortunately, an agent can compute the time-to-constraint-violation (TCV) by using the null-Markov chain (e.g.) $\bar{P}_{y,\epsilon}$ obtained from clamping the action of P_y to α_ϵ where the bar indicates that we delete the rows and columns of the constraint states. We compute $\mathbf{t}_c^y = (I - \bar{P}_{y,\epsilon})^{-1}\mathbf{1}$. The matrix $M = (I - \bar{P}_{y,\epsilon})^{-1}$ is the Fundamental Matrix of an absorbing Markov chain where $M(i,j)$ and represents the expected occupancies of state x_j starting from x_i (under determinism this is exact) (Brémaud, 2013). When rows of M are summed (i.e. $M\mathbf{1}$) we obtain the TCV. The vector $\mathbf{t}_c^y$ is an TCV vector (unknown) where $\mathbf{t}_c^y(i)$ indexes state y_i. We can solve for $\mathbf{t}_c^y$ as a linear system. If an Option were called that had a STFF time duration $t_f - t$ that was greater than the TCV, then the Option would *fail* to complete because the task fails. Thus we can cull the invalid Options from state x_i if the time-inequality $t_f - t \geq \mathbf{t}_c^y(i)$ is true. If the agent has multiple high-level constraints and TCV vectors, $\mathcal{T}_c = \{\mathbf{t}_c^y, ..., \mathbf{t}_c^z\}$, (e.g. starvation, dehydration, etc.), then the smallest TCV determines the (in)validity of an Option, written $t_f - t \geq \min_{\mathbf{t}_c^\ell \in \mathcal{T}_c} \mathbf{t}_c^\ell(x_i)$. We will use this for the STFF decomposition theorem. In Fig. 3 we show how TCV determines the validity of an Option. The red region is where an Option will violate a high-level constraint in a given space. Blue and green regions show the acceptable final states specific to a given space ($\mathcal{W}$ or $\mathcal{Y}$). The most restrictive region (blue) determines the validity of an Option through the inequality.

2.2 STFF Factorization

A key property of the OBE is that it has a factorization for a product-space STFF which allows us to break down the STFF into factors that we can solve individually, thereby avoiding the prohibitive complexity of solving high-dimensional OBEs with dynamic programming in $\mathcal{X} \times \mathcal{R}$.

Let us call an affordance function and constraint function pair $(F, f_c^{g_i})$ *homogeneous* if $f_c^{g_i}$ encodes all other non-null goal-actions $\boldsymbol{\alpha}_{g_j} \in \mathcal{A}_r \setminus \{\boldsymbol{\alpha}_{g_i}\}$ in F as constraints. Homogeneity implies that *all* policies are going to have a guaranteed *equivalent* effect on higher-level state dynamics. This will be relevant in the STFF decomposition because it means that we can make policy-independent predictions of high-level state-dynamics (with ω_r) independent of states in $\mathcal{X}$. Now the theorem:

Theorem 2.1 (Unidirectional STFF Decomposition)**.** *Let* $\bar{\mathscr{M}} = (\mathcal{Y} \times ... \times \mathcal{Z} \times \mathcal{X}, \mathcal{A}_x, P_s, f_{g_i}, f_c^{g_i})$ *be a CTMDP with* $P_s = \lambda_P(P_y, ..., P_z, F, P_x)$*, and* $\Omega = \{\omega_w, ..., \omega_z\}$ *is derived from* $\{P_w, ..., P_z\}$*. If* $(F, f_c^{g_i})$ *is homogeneous,* $f_c^{g_i}$ *is multiplicatively separable,* f_{g_i} *is not a function of high-level states*

$\mathbf{r} = (y, ..., z)$, *and* $t_f - t < \min_{\mathbf{t}_c^\ell \in \mathcal{T}_c} \mathbf{t}_c^\ell(x_i)$, *then the optimal product-space STFF* $\bar{\eta}_\pi^{**}$ *is equivalent to:*

$$\bar{\eta}_\pi^{**}(\alpha_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}, x, t) = \eta_\pi^{**}(\alpha_g, x_f, t_f | x, t) \prod_{\omega_\ell \in \Omega} \omega_\ell(\cdot|\cdot, t_f - t) = \omega_r(\mathbf{r}_f | \mathbf{r}, t_f - t) \eta_\pi^{**}(\alpha_g, x_f, t_f | x, t)$$

where (e.g.) $P_{y,\epsilon}$ *is the null-action Markov chain obtained from clamping the action of* P_y *to* α_ϵ, $\omega_y(y_f | y_i, t_f - t) := P_{y,\epsilon}^{t_f - t}(i, f)$ *is a prediction operator obtained by raising the null-Markov chain to the power of the time-duration, predicting the final state to be* y_f *after time* $t_f - t$ *from* y_i, $\omega_r(\mathbf{r}_f | \mathbf{r}, t_f - t) = \omega_y(y_f | y, t_f - t) ... \omega_z(z_f | z, t_f - t)$, *and* η_π^{**} *is the STFF of TMDP* $\mathscr{M} = (\mathcal{X}, \mathcal{A}, P_x, f_{g_i}, f_c^{g_i})$.

The theorem, proved in A.3 and visualized in figure 3, says if we know the policy has a consistent (homogeneous) effect on the dynamics of other spaces we can use the prediction operators ω of high-level spaces along with a low-level STFF η_π as the full product-space STFF. We can use this factorization for planning.

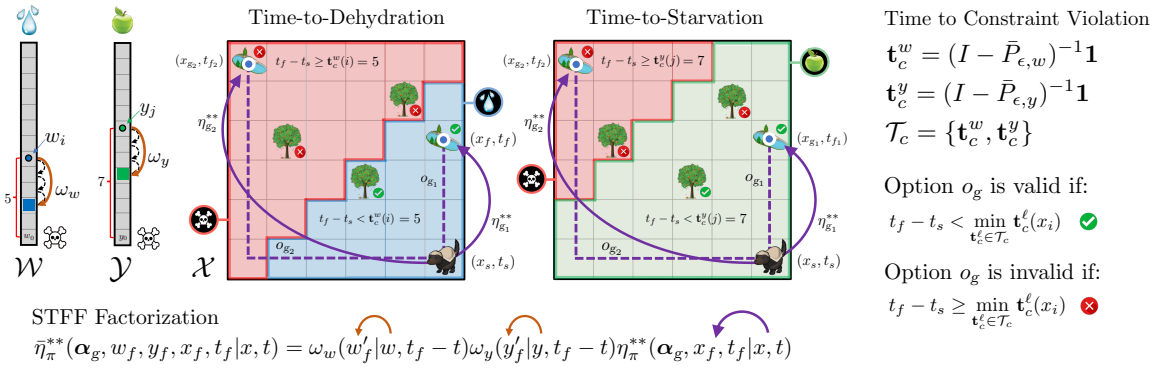

Figure 3: The STFF decomposition. Blue and Green are the acceptable regions of w_i and y_j. Options are valid if they are to states outside the red region for *both* maps, computed with the inequality.

2.3 Options from CMDP Ensembles

Given that we have defined a CMDP $\bar{\mathscr{M}} = (\mathcal{X} \times \mathcal{R}, \mathcal{A}_x, \lambda_P(P_r, F, P_x), f_g, f_c)$, we can solve for a set of options $\mathcal{O}$. To do this we take the operator $P_s = \lambda_P(P_r, F, P_x)$, and, using the STFF factorization, we solve for a set of low-level STFFs for each non-null goal $\boldsymbol{\alpha}_g$ in the affordance function F by converting it into a set $\mathcal{F}_\mathcal{G}$ of goal-availability functions for goal indices $\mathcal{G} = \{g_1, ..., g_N\}$:

$$\mathcal{F}_\mathcal{G} = \{f_{g_1}, f_{g_2}, ..., f_{g_N}\}, \quad \text{where: } f_{g_i}(x, a, t) = F(\boldsymbol{\alpha}_{g_i} | x, a, t), \quad \forall \boldsymbol{\alpha}_{g_i} \in \mathcal{A}_r \setminus \{\boldsymbol{\alpha}_\epsilon\}.$$

Now we solve STFFs $\mathscr{M}_{g_i} = (\mathcal{X}, \mathcal{A}, P_x, f_{g_i}, f_c)$ for each goal-availability function $f_{g_i} \in \mathcal{F}_\mathcal{G}$ along on the low-level operator P_x, which is the obstacle constraint function restricted to variables:

$$\mathcal{K}\Pi\mathcal{H}_\mathcal{G} \leftarrow \texttt{affordance\_ensemble\_solutions}(\bar{\mathscr{M}}), \quad (\kappa_{g_i}, \pi_{g_i}, \eta_{g_i}) \in \mathcal{K}\Pi\mathcal{H}_\mathcal{G}, \ \forall g_i \in \mathcal{G}$$
$$\text{where: } (\kappa_{g_i}, \pi_{g_i}, \eta_{g_i}) \leftarrow \texttt{feasibility\_iteration}(\mathscr{M}_{g_i} = (\mathcal{X}, \mathcal{A}, P_x, f_{g_i}, f_c^{g_i}))$$
$$\mathcal{K}_\mathcal{G} = \{\kappa_{g_1}, \kappa_{g_2}, ..., \kappa_{g_n}\}, \quad \Pi_\mathcal{G} = \{\pi_{g_1}, \pi_{g_2}, ..., \pi_{g_n}\}, \quad \mathcal{H}_\mathcal{G} = \{\eta_{g_1}, \eta_{g_2}, ..., \eta_{g_n}\}.$$

We will use the convention that the function `affordance_ensemble_solutions` returns $\mathcal{K}\Pi\mathcal{H}_\mathcal{G}$, a set of solution tuples, which can be split into individual sets $\mathcal{K}_\mathcal{G}, \Pi_\mathcal{G}$, and $\mathcal{H}_\mathcal{G}$. By combining elements $(\kappa_{g_i}, \pi_{g_i}, f_{g_i})$ from $\mathcal{K}_\mathcal{G}$, $\Pi_\mathcal{G}$, and $\mathcal{F}_\mathcal{G}$, grouped by indices g_i, a set of Options $\mathcal{O}_\mathcal{G}$ is defined as:

$$\mathcal{O}_\mathcal{G} = \{o_{g_1}, o_{g_2}, ..., o_{g_N}\} \leftarrow \texttt{create\_option\_set}(\mathcal{X}, \mathcal{K}_\mathcal{G}, \Pi_\mathcal{G}, \mathcal{F}_\mathcal{G}, f_c^{g_i})$$
$$\text{where: } o_{g_i} = (\mathcal{X}, \pi_{g_i}, \beta_{g_i}) \leftarrow \texttt{create\_option}(\mathcal{X}, \kappa_{g_i}, \pi_{g_i}, f_{g_i}, f_c^{g_i})$$

We now have a set of Options for each high-level goal-action variable in the original CMDP $\bar{\mathscr{M}}$.

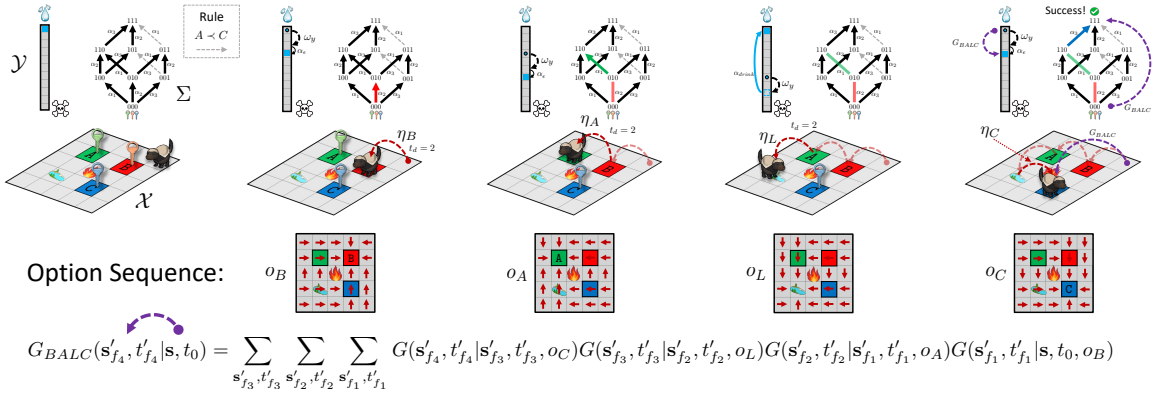

Figure 4: The honey badger completes the Boolean task of reaching $\boldsymbol{\sigma}_{111}$ and not entering the dehydrated state (skull) while respecting the rule $A \prec C$ encoded in P_σ. The solution, computed with BFS, is a sequence of goal-conditioned options $\rho_{BALK} = (o_B, o_A, o_L, o_C)$, each displayed.

2.4 Goal Operators

Having defined options from CMDP ensemble solutions, we will now define the Goal Operator, which is an operator that maps from initial state-time to final state-time under an option as an action. For Goal Operators with options derived from action-dependent goal-availability functions, we have to evolve the dynamics forward one-step from the boundary action of the problem. Using the sets $\mathcal{H}$, $\mathcal{O}$, and relevant operators (P_x, P_r, Ω), the Goal Operator definition is given as:

$$G(\mathbf{r}'_f, x'_f, t'_f|\mathbf{r}, x, t, o_{g_i}) = \sum_{\boldsymbol{\alpha}_f, \mathbf{r}_f, x_f, t_f} P_r(\mathbf{r}'_f|\mathbf{r}_f, \boldsymbol{\alpha}_f) P_x(x'_f|x_f, \pi_{g_i}(x_f, t_f), t_f) \omega_r(\mathbf{r}_f|\mathbf{r}, t_f - t) \eta_{\pi_{g_i}}(\boldsymbol{\alpha}_f, x_f, t_f|x, t)$$

where $t'_f = t_f + 1$, $\mathbf{r} = (\boldsymbol{\sigma}, y)$, $\pi_g \in o_g$, and G is defined for all $o_{g_i} = (\mathcal{X}, \pi_{g_i}, \beta_{g_i}) \in \mathcal{O}_\mathcal{G}$, $\eta_{\pi_{g_i}} \in \mathcal{H}$.

Generally, dynamic programming using G is intractable, but we can optimize sequence of options with breadth first search (BFS, figure 2) by forward-propagating state-vectors through G to predict the resulting state-vectors. Figure 4 shows the optimized solution to BFS where an agent has to finish a logical task (with a precedence rule) in a non-dehydrated state ($y \neq y_0$) while avoiding fire. The forward objective computed with BFS in algorithm (2) is given as:

$$\rho^* = \underset{\rho \in \mathcal{O}^M}{\operatorname{argmax}}\, \kappa_\rho(\mathbf{s}, t, m) = \underset{\rho \in \mathcal{O}^M}{\operatorname{argmax}} \left(f_1(\mathbf{s}, t) + f_2(\mathbf{s}, t) \underset{\mathbf{s}'_f, t'_f \sim G_{\rho(m)}}{\mathbb{E}} \kappa_\rho(\mathbf{s}'_f, t'_f, m+1) \right),$$

where $\mathbf{s} = (\boldsymbol{\sigma}, y, x)$ and m indexes the option sequence ρ in the set $\mathcal{O}^M$ of sequences of length M.

3 Discussion

We presented a new decision processes (the TMDP and CTMDP) and showed how they can be used to define an Operator Bellman Equation which creates factorized predictive planning representations called STFFs. This is possible because the structure of the Bellman equation and its functions (f_g and f_c) are critical to preserving the interpretation goal and constraint *events*. This is in contrast to reward-maximization objectives which aggregate events of obtained reward into the expectation of a running total, thereby losing interpretability of the optimized value function. Consequently, the STFF decomposition allows us to construct factorized Goal Operators to use for verifiable high-dimensional planning. Also, OBE solutions are very risk-sensitive in the canonical form. Creating a risk-calibrated basis of STFFs and Options for multi-goal problems is a very important theoretical problem. We anticipate that further research into this topic will reveal fundamental insights into making risk-calibrated verifiable planning scale to high-dimensional world-models.

References

David Abel, Will Dabney, Anna Harutyunyan, Mark K Ho, Michael Littman, Doina Precup, and Satinder Singh. On the expressivity of markov reward. *Advances in Neural Information Processing Systems*, 34:7799–7812, 2021.

Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pp. 679–684, 1957.

Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

Michael Bowling, John D Martin, David Abel, and Will Dabney. Settling the reward hypothesis. In *International Conference on Machine Learning*, pp. 3003–3020. PMLR, 2023.

Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, et al. Towards guaranteed safe ai: A framework for ensuring robust and reliable ai systems. *arXiv preprint arXiv:2405.06624*, 2024.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

Thomas J. Ringstrom. Reward is not necessary: How to create a modular & compositional self-preserving agent for life-long learning. *arXiv preprint arXiv:2211.10851*, 2022.

Thomas J. Ringstrom. *Reward Is Not Necessary: Foundations for Compositional Non-Stationary Non-Markovian Hierarchical Planning and Intrinsically Motivated Autonomous Agents*. PhD thesis, University of Minnesota, 2023.

David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

Joar Skalse and Alessandro Abate. On the limitations of markovian rewards to express multi-objective, risk-sensitive, and modal tasks. In *Uncertainty in Artificial Intelligence*, pp. 1974–1984. PMLR, 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Peter Vamplew, Benjamin J Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M Roijers, Conor F Hayes, Fredrik Heintz, Patrick Mannion, Pieter JK Libin, et al. Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *Autonomous Agents and Multi-Agent Systems*, 36(2):41, 2022.

Appendix

.1 Boundary conditions of OBEs

Because function in the OBE are undefined past T_f, we assume they are zero. Thus, the boundary conditions for the OBEs defined as:

$$\kappa_g^*(x, T_f) = \max_a [f_g(x, a, T_f)(1 - f_c(x, a, T_f))] = f_g(x, a_{T_f}^*, T_f)(1 - f_c(x, a_{T_f}^*, T_f),$$
$$\pi_g^{**}(x, T_f) = \operatorname*{argmax}_a [f_g(x, a, T_f)(1 - f_c(x, a, T_f))] = a_{T_f}^*,$$
$$\eta_g^+(\alpha_g, a_+, x_+, T_f | x_+, T_f) = f_g(x_+, a_{T_f}^*, T_f),$$
$$\eta_g^-(\alpha_g, a_-, x_-, T_f | x_-, T_f) = \mathbb{1}_\kappa(x_-, T_f)(f_c(x_-, a_{T_f}^*, T_f) + (1 - f_g(x_-, a_{T_f}^*, T_f))) + \bar{\mathbb{1}}_\kappa(x_-, T_f),$$

.2 Derivation of relationship between κ and η

Assume that we compute an optimal κ^*, π^{**}, and η^{**}. Using the $t = t_f$ condition of equation (10), we can substitute η_g^+ in for f_g in the recursion (6) but with the maximization dropped (since we have already optimized the three main equations), resulting in:

$$\kappa_{\pi_g}^*(x_t, t) = \eta_g^+(\alpha, x_t, t | x_t, t) + f_2(x_t, a, t) \sum_{x_{t+1}} P_x(x_{t+1} | x_t, a, t) \kappa_{\pi_g}^*(x_{t+1}, t+1) \tag{12}$$

Focusing on the right side of the addition, we can unroll κ another time-step, again substituting in η in for f_g for the boundary condition:

$$(1 - f_g(x_t, a, t)) \sum_{x'} P_x(x_{t+1} | x_t, a, t) \Bigg[\eta_g^+(\alpha, x_{t+1}, t+1 | x_{t+1}, t+1) + f_2(x_{t+1}, a, t+1) \sum_{x'} P_x(x_{t+2} | x_{t+1}, a, t+1) \kappa_{\pi_g}^*(x_{t+2}, t+2) \Bigg].$$

After distributing we have,

$$f_2(x_t, a, t) \sum_{x_{t+1}} P_x(x_{t+1} | x_t, a, t) \eta_g^+(\alpha_g, x_{t+1}, t+1 | x_{t+1}, t+1) + f_2(x_t, a, t) \mathop{\mathbb{E}}_{x_{t+1} \sim P_x} f_2(x_{t+1}, a, t+1) \mathop{\mathbb{E}}_{x_{t+2} \sim P_x} \kappa_{\pi_g}^*(x_{t+2}, t+2),$$

where the first term is equal to the definition of the feasibility function $\eta_g(\alpha, x_{t+1}, t+1 | x_t, t)$ defined by the OBE equation (8). Substituting, we have:

$$= \eta_g^+(\alpha_g, x_{t+1}, t+1 | x_t, t) + f_2(x_t, a, t) \mathop{\mathbb{E}}_{x_{t+1} \sim P_x} f_2(x_{t+1}, a, t+1) \mathop{\mathbb{E}}_{x_{t+2} \sim P_x} \kappa_{\pi_g}^*(x_{t+2}, t+2). \tag{13}$$

Note that (8) implies that for any start state-time (x, t) and final state-time (x_f, t_f) we can expand η_g^+ into a sequence of expectations of not achieving the goal under the policy, multiplied by the probability of achieving the goal:

$$\eta_g^+(\alpha_g, x_{t_f}, t_f | x_t, t) = f_2(x_t, a_\pi, t) \mathop{\mathbb{E}}_{x_{t+1} \sim P_x} f_2(x_{t+1}, a_\pi, t+1)) \ \ldots$$
$$\ldots \mathop{\mathbb{E}}_{x_{t_f-2} \sim P_x} f_2(x_{t_f-2}, a_\pi, t_f - 2) \mathop{\mathbb{E}}_{x_{t_f-1} \sim P_x} \eta_g^+(\alpha_g, x_{t_f}, t_f | x_{t_f-1}, t_f - 1). \tag{14}$$

Recall that (13) is the right side of the addition in (12), and every time we expand κ and distribute, we get a term which can be written as an extended η_g^+ function, plus the expectation of a future κ

term. Therefore, we can keep unrolling (13) out until T_f, and apply (14) each time, resulting in the sequence:

$$\kappa_g^*(x,t) = \eta_g^+(\alpha_g, x_t, t|x,t) + \eta_g^+(\alpha_g, x_{t+1}, t+1|x,t) + ... + \eta_g^+(\alpha_g, x_{T_f}, T_f|x,t),$$

which can be written as a sum over final state-times, leading to the relationship between κ and η_g^+:

$$\kappa_g^*(x,t) = \sum_{\alpha_+, x_+, t_+} \eta_g^+(\alpha_+, x_+, t_+|x,t).$$

The cumulative feasibility function (which is the cumulative probability of completing the task) is simply the sum of the first goal-satisfaction state-time events of the state-time feasibility function for a policy π_g.

A series of steps similar to those above for η_g^+ can be carried out to prove (not shown):

$$1 - \kappa_g^*(x,t) = \sum_{\alpha_-, x_-, t_-} \eta_g^-(\alpha_-, x_-, t_-|x,t),$$

and so when added together we have:

$$\sum_{\alpha_-, x_-, t_-} \eta_g^-(\alpha_-, x_-, t_-|x,t) + \sum_{\alpha_+, x_+, t_+} \eta_g^+(\alpha_+, x_+, t_+|x,t) = 1 - \kappa_g^*(x,t) + \kappa_g^*(x,t) = 1.$$

This means that when both η_g^{**} and η_g^- are considered as one function, η_π^{**},

$$\eta_\pi^{**}(\alpha_f, x_f, t_f|x,t) = \eta_g^+(\alpha_f, x_f, t_f|x,t) + \eta_g^-(\alpha_f, x_f, t_f|x,t),$$

and it sums to one:

$$\sum_{\alpha_f, x_f, t_f} \eta_\pi^{**}(\alpha_f, x_f, t_f|x,t) = 1.$$

Therefore η_π^{**} is a transition operator over state-action-times.

.3 STFF Decomposition Theorem

.3.1 Preliminaries 1: Definitions and Theory Review

Recall $f_1 = f_g f_c, f_2 = (1 - f_g) f_c$.

Recall, f_c is multiplicatively separable if

$$f_c(w, ..., z, x, a, t) = f_{c,r}(\mathbf{r}) f_{c,x}(x, a, t) = f_{c,w}(w)...f_{c,z}(z) f_{c,x}(x, a, t).$$

This is relevant for computing constraint-violation events efficiently in multiple spaces. We do not care that $f_{c,x}(x, a, t)$ is not separable because x, a, t variables are typically all used in a single "flat OBE" optimization.

Recall $f^{g_i}_{c,x}(x, a, t)$ encodes state-action-times of non-null goals α_g in F that are not α_{g_i} as constraints. The TMDPs with the pair $(F, f^{g_i}_{c,x})$ to be homogeneous, which means any policy derived from this pair implies that a "live" trajectory (not task-invalidating) produced by π induces only null-goal variables α_ϵ through F.

The multiplicatively separable function $f^{g_i}_c(\mathbf{r}, x, a, t) = f_{c,w}(w)...f_{c,z}(z) f^{g_i}_{c,x}(x, a, t)$ will always have $f^{g_i}_{c,x}(x, a, t)$ be the function where the constraints on the non-null goals are encoded.

.3.2 Preliminaries 2: Computing the Time to Constraint Violation

If $P_{y\epsilon}(y'|y)$ is the null Markov chain, then we can compute the first hitting time by calling the set of non-constraint states $\mathcal{N} = \mathcal{Y} \setminus \mathcal{Y}_c$ (i.e. the non-terminal states) and the set of constraint states $\mathcal{T} = \mathcal{Y}_c$ (i.e. terminal states). When we rearrange $P_{y,\epsilon}$ to segregate terminal and non-terminal indices, the block matrix of the controlled Markov dynamics $P_{y,\epsilon}$ is:

$$P_{y,\epsilon} = \left[\begin{array}{c|c} P_{\mathcal{N}\mathcal{N}} & P_{\mathcal{N}\mathcal{T}} \\ \hline P_{\mathcal{T}\mathcal{N}} & P_{\mathcal{T}\mathcal{T}} \end{array}\right],$$

The probability distribution of hitting a mode-inconsistent terminal state $y_j \in \mathcal{T}$ for the first time at time τ when starting from $y_i \in \mathcal{N}$ is:

$$P_{y\tau}(y_j, \tau|y_i) = (P^{(\tau-1)}_{\mathcal{N}\mathcal{N}} P_{\mathcal{N}\mathcal{T}})(i, j),$$

where $P^{(\tau-1)}_{\mathcal{N}\mathcal{N}}$ evolves the chain $\tau - 1$ time-steps to drain out the probability mass that hits a mode-inconsistent state prior to time τ, followed by a one-step evolution by multiplying with $P_{\mathcal{N}\mathcal{T}}$ to capture the probability of being at a mode-inconsistent state $x_j \in \mathcal{T}$.

$P_{y\tau}$ is a PMF over state-times. Let $t_{tcv,y_i} \sim \sum_{y_j} P_{y\tau}(y_j, t|y_i)$ be the TCV from the state y_i. The conditional CDF over TCV times is $S_y(\tau|y_i) = \sum_{t \leq \tau} \sum_{y_j} P_{y\tau}(y_j, t|y_i)$. The probability that a time duration of a goal $t_f - t$ is less than the TCV is obtained from evaluating the CDF:

$$Pr(t_f - t < t_{tcv,y_i}) = S_y(t_f - t|y_i)$$

If we have determinism in $P_{y,\epsilon}$, then the Neumann series $M_{y,\epsilon} = \sum_\tau P^\tau_{y,\epsilon} = (I - P_{y,\epsilon})^{-1}$, where $M_{y,\epsilon}(i, j)$ represents the expected state-occupancies of y_j before hitting the boundary (constraint) states (where the expectation is exact under determinism) Brémaud (2013). We can sum each row by multiplying $M_{y,\epsilon}$ by a column vector of ones $\mathbf{1}$ to obtain the PMF of the first-hit time $P_{y\tau}(\tau|y_i)$ and the CDF $S_y(\tau|y_i)$:

$$\begin{aligned} &\mathbf{t}^y_c = M_{y,\epsilon}\mathbf{1} \\ &P_{y\tau}(\tau|y_i) = \mathbf{t}^y_c(i), \quad \text{where: } P_{y\tau}(\tau|y_j) = 0 \text{ if: } y_j \in \mathcal{T}_y \\ &S_y(\tau|y_i) = \sum_{t \leq \tau} P_{y\tau}(t|y_i) \end{aligned}$$

.3.3 Theorem

To restate the Theorem:

Theorem .1 (Unidirectional STFF Decomposition)**.** *Let $\bar{\mathscr{M}} = (\mathcal{Y} \times ... \times \mathcal{Z} \times \mathcal{X}, \mathcal{A}, P_s, f_{g_i}, f_c^{g_i})$ be a CTMDP where $\mathcal{R} = \mathcal{Y} \times ... \times \mathcal{Z}$, $P_s = \lambda_P(P_r, F, P_x)$, $f_c^{g_i}$ is multiplicatively separable, $(F, f_{c,x}^{g_i})$ is homogeneous, P_r is deterministic, and where $f_g(x, a, t)$ is not a function of $\boldsymbol{r} = (w, ..., z)$. Let the set of TCV be $\mathcal{T}_c = \{\mathbf{t}_c^w, ..., \mathbf{t}_c^z\}$, and prediction operators be $\Omega_r = \{\omega_w, ..., \omega_z\}$.*

If $t_f - t \leq \min_{\mathbf{t}_j \in \mathcal{T}_c} \mathbf{t}_j(\mathbf{r}_j)$, then the optimal STFF $\bar{\eta}^{**}$ is equivalent to:

$$\begin{aligned}\bar{\eta}^{**}(\boldsymbol{\alpha}_g, \boldsymbol{r}_f, x_f, a_f, t_f | \boldsymbol{r}, x, t) &= \eta_\pi^{**}(\boldsymbol{\alpha}_g, a_f, x_f, t_f | x, t) \prod_{\omega_\ell \in \Omega_r} \omega_\ell(\cdot|\cdot, t_f - t) \\ &= \eta_\pi^{**}(\boldsymbol{\alpha}_g, x_f, a_f, t_f | x, t) \omega_r(\boldsymbol{r}_f | \boldsymbol{r}, t_f - t),\end{aligned}$$

where $\omega_r(\mathbf{r}_f | \mathbf{r}, t_d) := \omega_w(w_f | w, t_d) ... \omega_z(z_f | z, t_d)$, and the STFF η_π^{**} is the solution to the TMDP $\mathscr{M} = (\mathcal{X}, \mathcal{A}, P_x, f_g, f_{c,x}^{g_i})$.

Proof sketch (this proof is under review): Remark: notice that this theorem gives an equivalence for the condition if: $t_d \leq \min_{\mathbf{t}_j \in \mathcal{T}_c} \mathbf{t}_j(\mathbf{r}_j)$, which is saying the time that it takes to complete the goal is less than the time that it takes hit a constraint under P_r. In this proof we will point out where we enforce this condition.

For this proof, we will be using the time-variable τ_i where i indexes the number of steps *away* from the final time T_f. So, $\tau_0 = T_f$, $\tau_1 = T_f - 1$ and $\tau_i = T_f - i$, so keep in mind that the subscript on τ is not counting up from $t_0 = 0$ (which will be referred to as $\tau_s = t_0$). Also, all variables with subscript f like x_f are equivalent to x_{τ_0} or x_{T_f}.

Assume we are compute an optimal CFF κ_g^* and policy π_g^{**} at the boundary and step backwards iteratively through time. We will analysis the STFF as we go backwards in time and assume we compute the κ-OBE and π-OBE concurrently.

The horizon is defined as $\bar{\eta}^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}_f, x_f, t_f) = f_g(x, \pi^{**}(x, t), T_f)$. Again, letting $\tau_1 = T_f - 1$, we can start by substituting the product-space $\bar{\eta}$ with the η restricted to $\mathcal{X}$:

$$\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}, x, \tau_1) = f_2(\mathbf{r}, x, a, \tau_1) \sum_{x', \mathbf{r}'} P_s(\mathbf{r}', x' | \mathbf{r}, x, a_\pi^{**}, \tau_1) \bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}', x', \tau_1 + 1) \quad (15)$$

$$\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}, x, \tau_1) = f_2(\mathbf{r}, x, a, \tau_1) \sum_{x', \mathbf{r}'} P_s(\mathbf{r}', x' | \mathbf{r}, x, a_\pi^{**}, \tau_1) \eta_\pi^+(\boldsymbol{\alpha}_g, x_f, t_f | x', \tau_1 + 1),$$

$$\text{where: } \eta_{g_i}^+(\alpha_g, \mathbf{r}, x, \tau | \mathbf{r}, x, \tau_1) = f_g(\mathbf{r}, x, a_{xt}^\pi, \tau_1), \quad \text{when: } \tau = t_f$$

From now on, the last line for when $\tau = t_f$ will always be implicit throughout this derivation, we will not keep re-writing it. Substituting $P_s(\mathbf{r}', x' | \mathbf{r}, x, a, t) := \sum_{\boldsymbol{\alpha}} P_r(\mathbf{r}' | \mathbf{r}, \boldsymbol{\alpha}) F(\boldsymbol{\alpha} | x, a, t) P_x(x' | x, a, t)$ and pulling out the summations over $\boldsymbol{\alpha}$, and by separability of f_2 (and f_c), we can move $f_{2,r}$ out of the parentheses:

$$\begin{aligned}\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f | \mathbf{r}_{\tau_1}, x, \tau_1) = f_{2,r}(\mathbf{r}_{\tau_1}) &\left(\sum_{\mathbf{r}', \boldsymbol{\alpha}} P_r(\mathbf{r}' | \mathbf{r}_{\tau_1}, \boldsymbol{\alpha}) F(\boldsymbol{\alpha} | x, a, \tau_1) \right) \\ &* f_{2,x}^{g_i}(x, a, \tau_1) \sum_{x'} P_x(x' | x, a_\pi^{**}, \tau_1) \eta_\pi^+(\boldsymbol{\alpha}_g, x_f, t_f | x', \tau_1 + 1),\end{aligned}$$

By homogeneity, F will only output $\boldsymbol{\alpha}_\epsilon$ and so it can be dropped and we substitute in $P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_1})$ conditioned only on $\boldsymbol{\alpha}_\epsilon$:

$$\begin{aligned}&\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}_{\tau_1}, x, \tau_1) \qquad (16)\\ &= \sum_{\mathbf{r}'} f_{2,r}(\mathbf{r}') P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_1}) f_{2,r}(\mathbf{r}_{\tau_1}) \underbrace{\left(f_{2,x}^{g_i}(x, a, \tau_1) \sum_{x'} P_x(x'|x, a_\pi^{**}, \tau_1) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x', \tau_1 + 1) \right)}_{= \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_1)}.\end{aligned}$$

The term in the large parentheses of equation (16) is the definition of an STFF $\mathscr{M} = (\mathcal{X}, \mathcal{A}_x, \mathcal{T}, P_x, f_{g_i}, f_{c,x}^{g_i})$ computed only on P_x and $f_{2,x}$. which we call η_g^+ (no bar). Substituting this new STFF in and continuing from τ_1 using the same goal-availability function, we have:

$$\begin{aligned}&\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_1) = f_{2,r}(\mathbf{r}_{\tau_1}) P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_1}) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_1). \qquad (17)\\ &\text{where:}\quad \bar{\eta}_g^+(\alpha_g, \mathbf{r}, x, \tau|\mathbf{r}, x, \tau_1) = f_g(\mathbf{r}, x, a_{xt}^\pi, \tau_1), \quad \text{when: } \tau_1 = t_f\end{aligned}$$

Note that $\tau_1 = T_f - 1$ and so $\mathbf{r}' = \mathbf{r}_f$.

Equation (17) as it currently stands, has two separable parts, we have two terms that only depends on $\mathbf{r}$, and one term that depends on (x, a, t), and so each of these two terms can be computed independently of one another.

Notice that when we arrive at (17), we can substitute (17) for equation (15) but for one time-step backwards in time, $\tau_2 \leftarrow \tau_1 - 1$ (recall $\tau_1 = T_f - 1$ so the subscript is counting backwards in time, i.e. $\tau_2 = T_f - 2$). When we go through (15) to (17) again, starting with the substitution:

$$\begin{aligned}&\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_2)\\ &\quad = f_{2,r}(\mathbf{r}_{\tau_2}) f_{2,x}(x, a, \tau_1) \sum_{x', \mathbf{r}_{\tau_1}} P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_1}) P_s(\mathbf{r}', x'|\mathbf{r}, x, a_\pi^{**}, \tau_2) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_1),\end{aligned}$$

and following the same steps to (17), we end at:

$$\begin{aligned}\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_2) &= \underbrace{\sum_{\mathbf{r}_{\tau_1}, \mathbf{r}_{\tau_2}} P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_1}) f_{2,r}(\mathbf{r}_{\tau_1}) P_{\mathbf{r},\epsilon}(\mathbf{r}_{\tau_1}|\mathbf{r}_{\tau_2}) f_{2,r}(\mathbf{r}_{\tau_2})}_{M_c^n(\mathbf{r}_{\tau 2})} \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_2)\\ &= M_c^n(\mathbf{r}_{\tau 2}) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_2)\end{aligned}$$

This series acts as an absorbing Markov chain. To see how, we can represent the constraint function as a diagonal matrix $D_c = \text{diag}(\text{vec}(f_{c,r}))$, and $P_{r,\epsilon}$ will be the Markov matrix:

$$M_{c,n} = P_{r,\epsilon} D_c \overset{n-2}{\cdots} P_{r,\epsilon} D_c = (P_{r,\epsilon} D_c)^n$$

where $M_{c,n}(i, j)$ is a PDF equivalent to $Pr(\mathbf{r}_j, t|\mathbf{r}_i) = M_{c,t}(i, j)$. We can get a CDF of the probability that the trajectory has been absorbed $S_y(\tau|y_i) = \sum_{t \le \tau} \sum_j M_{c,t}(i, j)$. Furthermore, $1 - M_{c,n}(i, j)$ is the probability that, starting at state $\mathbf{r}_i$, the final state is $\mathbf{r}_j$ without hitting a constraint. Thus, we can see that $1 - M_{c,n}(i, j)$ represents the probability that a constraint has not been violated up to n time-steps (i.e.) the probability that the option or policy is *valid*.

As we keep repeating this process and we obtain the general form:

$$\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_i) = S_{\mathbf{r}}(t_f|y_i) P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_i}, t_f - \tau_i) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_i)$$

where we rename $P_{\mathbf{r},\epsilon}(\mathbf{r}_f|\mathbf{r}_{\tau_i}, t_f - \tau_i) = \omega_r(\mathbf{r}_f|\mathbf{r}, t_f - t)$,

$$\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_i) = S_{\mathbf{r}}(t_f|\mathbf{r}_i) \omega_r(\mathbf{r}_f|\mathbf{r}, t_f - t) \eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_i)$$

The term $S_{\mathbf{r}}(t_f|\mathbf{r}_i)$ acts to validate or invalidate the dynamics operator ω_r, as $S_{\mathbf{r}}(t_f|\mathbf{r}_i)$ is

A couple things to note, since $\mathbf{r} = (w, ..., z)$, and $f_{2,r}(\mathbf{r})$ is multiplicatively separable, we can break ω_r into $\omega_r(\mathbf{r}_f|\mathbf{r}, t_d) = \omega_w(w_f|w, t_d)...\omega_z(z_f|z, t_d)$, and break down $S_{\mathbf{r}}(t_f|\mathbf{r}_i)$ into $S_{\mathbf{r}}(t_f|\mathbf{r}_i) = S_w(t_f|w_i)...S_z(t_f|z_i)$. Furthermore, if the high-level dynamics are deterministic, then the CDF (e.g.) $S_z(t_f|z_i)$ is the same thing as the inequality:

$$t_f - t \leq \mathbf{t}_{c,z}(i), \quad \text{where: } \mathbf{t}_c = (I - P_{z,\epsilon})^{-1}\mathbf{1}.$$

we then have the set $\mathcal{T}_c = \{\mathbf{t}_w, ..., \mathbf{t}_z\}$, and (assuming determinism in $\mathcal{R}$):

$$S_{\mathbf{r}}(t_f - t|\mathbf{r}_i) = \min_{\mathbf{t}_j \in \mathcal{T}_c} \mathbf{t}_j(\mathbf{r}_j)$$

Therefore, we now have the premise, that is if $t_f - t \leq \min_{\mathbf{t}_j \in \mathcal{T}_c} \mathbf{t}_j(\mathbf{r}_j)$,:

$$\bar{\eta}_g^+(\boldsymbol{\alpha}_g, \mathbf{r}_f, x_f, t_f|\mathbf{r}, x, \tau_i) = \omega_r(\mathbf{r}_f|\mathbf{r}, t_f - t)\eta_g^+(\boldsymbol{\alpha}_g, x_f, t_f|x, \tau_i),$$

assuming determinism in P_r.

Since the entire proof can be done with for the goal-success distribution, one can apply the exact same steps to the goal-failure distribution (not shown), resulting in:

$$\bar{\eta}_-(\bar{\boldsymbol{\alpha}}_-, \mathbf{r}_-, x_-, t_-|\mathbf{r}, x, t) = \omega_r(\mathbf{r}_-|\mathbf{r}, t_- - t)\eta(\bar{\boldsymbol{\alpha}}_-, x_-, t_-|x, t).$$

□

A couple things to note: This proof assumes $P_{\mathbf{r}}$ is deterministic, so the above calculation is for the average hitting time, which must also be the exact hitting time. If one were to prove this for stochastic $P_{\mathbf{r}}$, one would have to compute the joint probability that any of the components change the environment variable, which would involve forward-evolving each the Markov chain dynamics. One would have to compute this probability for all combinations of high-level states, which would be possible, but computationally expensive. Furthermore, if ζ were not element-independent, then we would have to compute the hitting time and probabilities on the full Markov chain $\bar{P}_r$:

$$\mathbf{t}_r = (I - \bar{P}_r)^{-1}\mathbf{1},$$

which is computationally prohibitive, unless alternative techniques were possible to be developed. It is also worth noting that this proof assumed that the high-level states were deterministic. A similar proof could potentially be given assuming stochasticity if one computes the probability of arriving at a specific state before a given time under the controlled Markov dynamics, but would likely not have a simple solution (e.g. in the form of a linear system) and would involve forward-evolving the probability distributions under the Markov chain.

.4 Feasibility Iteration

Algorithm 1: Feasibility Iteration

input : Dynamics P_x, goal-availability function f_g, f_c constraint function, final time T_f
output : Cumulative feasibility function κ, Policy π, State-time feasibility function η

define nx as the number of states
define $f_1 = (1 - f_g)(1 - f_c), f_2 = f_g(1 - f_c)$
initialize $\eta_g^+, \eta_g^- \leftarrow zeros([nx * T_f, nx * T_f])$
initialize $\kappa \leftarrow zeros([nx, T_f])$
$\kappa(x_f, T_f) \leftarrow \max_a f_g(x_f, a, T_f)\ \forall x_f \in \mathcal{X}$, #CFF Boundary Conditions
$\pi(x_f, T_f) \leftarrow \text{argmax}_a f_g(x_f, a, T_f)\ \forall x_f \in \mathcal{X}$, #Policy Boundary Conditions
$\eta_g^+(x_f, T_f | x_f, T_f) \leftarrow f_g(x_f, \pi(x_f, T_f), T_f),\ \forall x_f \in \mathcal{X}$ #Success Boundary Conditions
$\eta_g^-(x_f, T_f | x_f, T_f) \leftarrow 1 - f_g(x_f, \pi(x_f, T_f), T_f),\ \forall x_f \in \mathcal{X}$ #Failure Boundary Conditions
for t *from* $T_f - 1$ *to* 0 **do**
  for $x \in \mathcal{X} \setminus \mathcal{X}_c$ **do**
    $(max\_\kappa, \mathcal{A}^*_{x,t}) \leftarrow \text{argmax}_{\mathcal{A}} [f_1(x, a, t) + f_2(x, a, t)\, \mathbb{E}_{x' \sim P_x}\, \kappa(x', t + 1)]$
    $\kappa(x, t) \leftarrow max\_\kappa$
    $\pi(x, t) \leftarrow a^{**}_{xt} \leftarrow \text{argmin}_{a \in \mathcal{A}^*_{x,t}} \left[t f_1(x, a, t) + f_2(x, a, t)\, \mathbb{E}_{P_x}\, \mathbb{E}_{\eta_g^+}\, t_+ \right]$
    $\eta_g^+(x_f, t | x_f, t) \leftarrow f_1(x, a^{**}_{xt}, t),\ \ \forall x_f \in \mathcal{X}$
    $\eta_g^+(x_f, t_f | x, t) \leftarrow f_2(x, a^{**}_{xt}, t))\, \mathbb{E}_{x' \sim P_x}\, \eta_g^+(x_f, t_f | x', t), \forall (x_f, t_f) \in \mathcal{X} \times \mathcal{T}_{>t}$
    #Optional computation of failure probability for stochastic dynamics
    if $\kappa(x, t) > 0$ **then**
      $\eta_g^-(x_f, t_f | x, t) \leftarrow \mathbb{E}_{x' \sim P_x(\cdot | x, a^{**}_{xt}, t)}\, \eta_g^-(x_f, t_f | x', t + 1), \forall (x_f, t_f) \in \mathcal{X} \times \mathcal{T}_{>t}$
    else
      $\eta_g^-(x, t | x, t) \leftarrow 1$
    end
  end
end
$\eta \leftarrow \texttt{Combine}(\eta_g^+, \eta_g^-)$
`return` κ,π,η

.5 Option Sequence Breadth First Search

Algorithm 2: Breadth_First_Plan_Search

input : $\widehat{\eta}_c, \mathcal{H}_c = \{\eta_{e_1,g_1}, \eta_{e_1,g_2}, ...\}, \Omega_r = \{\omega_w, ..., \omega_z\}, \mathcal{P}_r = \{P_w, ..., P_z\}, P_x, P_\sigma,$
$\Pi_{e,g} = \{\pi_{e_1,g_1}, \pi_{e_1,g_2}, ..., \pi_{e_\ell,g_k}\}$, $\mathcal{O} = \{o_{e_1,g_1}, o_{e_1,g_2}, ..., o_{e_\ell,g_k}\}$, max_horizon M, $\bar{f}_1$

output : tree

let $\mathbf{st}_{init} \leftarrow (\boldsymbol{\sigma}_{init}, \mathbf{r}_{init}, x_{init}, e_{init}, t_0)$
let $\kappa_{init} \leftarrow \bar{f}_1(\mathbf{st}_{init})$
let $root \leftarrow Node(\mathbf{st}_{init}, \rho = (), \kappa_\rho = \kappa_{init}, leaf \leftarrow False, parent \leftarrow none)$ be an initial node
define $\omega_r(\mathbf{r}_f|\mathbf{r}, t_d) := \prod_{\omega_\ell \in \Omega} \omega_\ell(\cdot|\cdot, t_d)$
$tree \leftarrow initialize\_tree(root)$
$Queue.push(root)$
while $not\_empty(Queue)$ **do**
  $node \leftarrow Queue.pop()$
  $(\boldsymbol{\sigma}, \mathbf{r}, x, t) \leftarrow node.\mathbf{st}$
  $e \leftarrow \zeta(\mathbf{r}, \boldsymbol{\sigma})$
  for $\pi_{e,g}$ *in* $\Pi_e \subseteq \Pi$ **do**
    if $(\kappa_{\pi_{e,g}}(x,t) > 0)$ **then**
      $(\boldsymbol{\alpha}, x', t') \leftarrow \widehat{\eta}_c(\boldsymbol{\alpha}, x', t'|x, t, \mathbf{r}, \pi)$
      $\mathbf{r}' \leftarrow \omega_r(\cdot|\mathbf{r}, t' - t)$
      $x'' \leftarrow P_x(x''|x', \pi(x', t), t)$
      $\mathbf{r}'' \leftarrow$ `one_step_internal_update`$(\mathbf{r}', \boldsymbol{\alpha}, \mathcal{P}_r)$
      $\boldsymbol{\sigma}'' \leftarrow P_\sigma(\boldsymbol{\sigma}''|\boldsymbol{\sigma}, \boldsymbol{\alpha})$
      $t'' \leftarrow t' + 1$
      $\mathbf{st}'' \leftarrow (\boldsymbol{\sigma}'', \mathbf{r}'', x'', t'')$
      $cur\_feas \leftarrow (node.\kappa_\rho) \times \bar{f}_2(\boldsymbol{\sigma}, \mathbf{r}, x, t) + \bar{f}_1(\boldsymbol{\sigma}'', \mathbf{r}'', x'', t'')$
      if $len(node.\rho) < M$ **then**
        $new\_node \leftarrow Node(\mathbf{st}'', \kappa_\rho \leftarrow cur\_feas, leaf \leftarrow$ `False`$, parent \leftarrow node)$
        $tree \leftarrow add\_to\_tree(tree, new\_node)$
        $Queue.push(new\_node)$
      else
        $new\_node \leftarrow Node(\mathbf{s}'', leaf \leftarrow$ `True`$, parent \leftarrow node)$
        $tree \leftarrow add\_to\_tree(tree, new\_node)$
      end
    end
  end
end
$\mathcal{P}_{f\_max} \leftarrow$ `get_feasibility_maximizing_plans`$(tree.leaves)$
$\mathcal{P}_{f\_max\_t\_min} \leftarrow$ `get_time_minimizing_plans`$(\mathcal{P}_{f\_max})$
`Return`$(\mathcal{P}_{f\_max\_t\_min})$