Hybrid Game Control Envelope Synthesis

ADITI KABRA, Carnegie Mellon University, USA
JONATHAN LAURENT, Karlsruhe Institute of Technology, Germany
STEFAN MITSCH, DePaul University, USA
ANDRÉ PLATZER, Karlsruhe Institute of Technology, Germany

Control problems for embedded systems like cars and trains can be modeled by two-player hybrid games. Control envelopes, which are families of safe control solutions, correspond to nondeterministic winning policies of hybrid games, where each deterministic specialization of the policy is a control solution. This paper synthesizes nondeterministic winning policies for hybrid games that are as permissive as possible. It introduces *subvalue maps*, a compositional representation of such policies that enables verification and synthesis along the structure of the game. An inductive logical characterization in differential game logic (dGL) checks whether a subvalue map induces a sound control envelope which always induces a winning play. A policy is said to win if it always achieves the desirable outcome when the player follows it, no matter what actions the opponent plays. The maximal subvalue map, which allows the most action options while still winning, is shown to exist and satisfy a logical characterization. A family of algorithms for nondeterministic policy synthesis can be obtained from the inductive subvalue map soundness characterization. An implementation of these findings is evaluated on examples that use the expressivity of dGL to model a range of diverse control challenges.

Additional Key Words and Phrases: Hybrid games, Program synthesis, Differential game logic

1 Introduction

Imperative program-like games serve as specifications for program synthesis [13, 40], where the game specifies the *shape* of the program, leaving the details as nondeterministic choices for the player to resolve. The synthesis of a single correct solution consists of identifying a single winning policy, indicating how to resolve choices to obtain a deterministic program that meets the specification (wins the game). For some programs, such as embedded system control software, it is useful to know a maximal *set* of control solutions, which can then be specialized for secondary or evolving requirements [3, 16, 24], e.g., train control software that ensures speed limit adherence can be specialized for fuel efficiency depending on current operating conditions. This corresponds to the synthesis of a *set* of winning policies for the specification game, which is to say, a *nondeterministic policy* whose every deterministic specialization corresponds to a winning policy. This paper introduces *subvalue maps* as a compositional representation of nondeterministic policies that can be verified and synthesized recursively along the syntactic structure of the game. Via this representation we develop the theory to compare, verify, and synthesize nondeterministic policies. Our formalization is in differential game logic (dGL) [44], which models two-player games with imperative program constructs like assignments, loops and branching. A central contribution is to develop a representation of nondeterministic game policies whose verification/synthesis composes at the level of such program constructs.

This paper synthesizes nondeterministic policies for *hybrid games*. It is able to solve problems that previous hybrid games nondeterministic policy synthesis work does not because the subvalue map representation lets us leverage program analysis techniques like loop variants, invariants and their continuous analogs. *Hybrid* games [44] extend games [42] with differential equations. Nondeterministic policy synthesis for hybrid games is important because it solves the problem of identifying families of correct control solutions (also called *control envelopes*) for embedded systems

Authors' Contact Information: Aditi Kabra, Carnegie Mellon University, Pittsburgh, PA, USA, akabra@cs.cmu.edu; Jonathan Laurent, Karlsruhe Institute of Technology, Karlsruhe, Germany, jonathan.laurent@kit.edu; Stefan Mitsch, DePaul University, Chicago, IL, USA, smitsch@depaul.edu; André Platzer, Karlsruhe Institute of Technology, Karlsruhe, Germany, platzer@kit.edu.

like cars and trains which have discrete as well as continuous physical behavior. Hybrid games serve as controller synthesis specifications [39, 41, 61] which characterize the control possibilities within a system's existing hardware and environment. One player, canonically called Angel, resolves the nondeterminism that the controller can resolve by making a control decision, while the other player, canonically called Demon, resolves nondeterminism resulting from environment behavior. This results in a two-player, zero-sum *game* where Angel's policy to win, however Demon may resolve non-determinism, is exactly what the controller must do to maintain the desired properties (e.g., safety, liveness, reach-avoid) regardless of environmental conditions. Embedded controllers are safety-critical, ubiquitous, but hard to design correctly, so synthesizing their control envelopes addresses an important challenge.

After defining subvalue maps, this paper introduces the theory to verify that subvalue maps induce only policies that don't lose. To ensure that induced policies additionally never let the player get stuck, always allowing some action, the refined concept of *inductive* subvalue map is introduced. We order subvalue maps such that more permissive maps that permit more control options are greater. We synthesize inductive subvalue maps. First, we show completeness: it is always possible to construct an optimal inductive subvalue map for a given game. Then, since our ultimate goal is to use subvalue maps as control envelopes/nondeterministic policies, we consider a restricted category of inductive subvalue maps where checking if a given controller lies within the envelope is efficient (polynomial time). We present an algorithmic framework to synthesize such subvalue maps. The theoretical development needed is subtle. For example, at loops, a nondeterministic policy should allow *unboundedly* many repetitions so long as there is still a *finite* exit strategy *within* the same nondeterministic policy. The first few sections of the paper build up the theoretical tools that make the theory for subvalue maps possible, and largely internalized within dGL. These include: (1) a dGL analog to Brzozowski derivatives [15] that let us step through a game and characterize gameplay after a given subgame (Section 4), and (2) dGL characterizations checking whether a set of states is reachable by following *some* strategy or *any* strategy in a given subvalue map (Section 5).

We synthesize inductive subvalue maps by recursively computing and composing them while syntactically stepping through a game. In dGL, *subgames* specify game decision points where at most one player makes a play. These subgames compose together per a recursive syntax to form large, arbitrarily complex games. Subvalue maps are analogous to value functions in reinforcement learning [57]. A value function maps states to utility. Once it is known, an agent policy can be derived from it. Similarly, an Angelic subvalue map maps every subgame to a formula denoting a set of states from which Angel can win the rest of the game (*sub*value because some maps map to a set of *known* winning states smaller than the theoretical maximum winning set). A player's policy set can then be reconstructed by tracing through the game: at every subgame where the player must take a decision, it accepts only those decisions that reach the next subgame in a mapped winning state for that subgame per the subvalue map.

This paper solves these challenges for *all* of dGL: Synthesis for safety properties, liveness properties, and any combination of these are all solved by a single, natural framework without special handling of either. The flexibility of our final synthesis framework is demonstrated through examples with diverse control challenges, and performance is evaluated on benchmarks from the literature and a procedurally generated benchmark suite. Our main contributions are to:

(1) Introduce *inductive subvalue maps* as symbolic, composable representations of nondeterministic winning policies for hybrid games (Section 6).
(2) Prove completeness for optimal hybrid game inductive subvalue map synthesis via the construction of a *maximal subvalue map* (Section 7).

(3) Present an algorithmic framework (Section 8) for *synthesis* of inductive subvalue maps where checking that control is within the induced nondeterministic policy is *computationally efficient.*
(4) Evaluate the approach, demonstrating its application to representatives of various control theory problem classes (Section 9).

2 Differential Game Logic

This paper synthesizes policies for games written in differential game logic (dGL), a logic for two-player hybrid games that has a relatively complete axiomatization. A detailed explanation can be found elsewhere [44, 45], but we provide an overview of dGL and set up the notation used in this paper.

A dGL game is played by two adversarial players, canonically called Angel and Demon. Various operators give either player opportunities to make control decisions to try to attain a winning condition. This paper uses the convention that Angel is responsible for the controller's decisions, and wins when she maintains the desired system properties (e.g. safety), and Demon plays for the environment. dGL games are generated by a recursive grammar consisting of the game constructs that follow. In many of the game constructs, either Angel or Demon can make a choice, deciding how the game should proceed, and try to choose in a way that is most favorable to them. These decisions are represented using *actions*, which will later be used to keep track of state change, construct game trees and to represent policies[1]. The game constructs are as follows:

(1) Angelic free assignment $x := *$ models Angel assigning any real value of her choice to variable x, and can be used, e.g., to model a controller's ranged choices. Angel's available actions are of the form $(x := e)$, where e is a (real polynomial) term. Dually, Demonic free assignment $x := \otimes$ models Demon assigning a real value of his choice to variable x instead. It models, e.g., the environment introducing disturbances making control harder. Demon's available actions are of the form $(x := e)^d$, where e is a term.
(2) Continuous evolution $\{x' = f(x) \,\&\, Q\}$ modifies x per the solution of differential equation $x' = f(x)$. How long the ODE runs is determined by Angel, who must also maintain that domain constraint formula Q is true throughout. This can model, for example, a control mode running till the controller interrupts. Angel's available actions are of the form $(x' = f(x) \,\&\, Q \,@\, t)$, where t is a term representing the time Angel chooses to run the ODE for. Dually, in $\{x' = f(x) \,\&\, Q\}^d$, Demon chooses how long to run the ODE while ensuring that Q holds throughout. Demon's available actions are of the form $(x' = f(x) \,\&\, Q \,@\, t)^d$, where t is a term for the time Demon runs the ODE. Demon-controlled ODEs can model, e.g., control loop latency in controllers that periodically poll to take decisions.
(3) Loop α^* runs α as many times as Angel wants. Before starting a fresh iteration of the loop, Angel gets to choose whether to run the loop again or to exit it. This can model, for example, a controller that repeatedly performs a task until a criterion is met. Her available actions are $\mathfrak{g}$ to *go* repeat the loop for one more iteration, or $\mathfrak{s}$ to *stop* and exit the loop. Dually, $\alpha^\times$ runs α as many times as Demon chooses. Demon loop can model a control loop that runs arbitrarily many times, and must stay safe forever. Demon's available actions are $\mathfrak{g}^d$ to repeat the loop for one more iteration, or $\mathfrak{s}^d$ to exit the loop.
(4) Branch choice $\alpha \cup \beta$ lets Angel choose to play either game α or game β. This can model a controller choosing between two modes of operation, for example accelerating or braking. Angel's available actions are $\mathfrak{l}$ to play *left* game α, or $\mathfrak{r}$ to play *right* game β. Dually, $\alpha \cap \beta$

[1]The action notation presented here is a simplified form of the operational semantics of dGL [44][Appendix C]. Sequential compositions are given an action for uniformity but do not have an action in the original operational semantics.

lets Demon choose to play either game α or game β. This can model different modes of environment behavior, such as a car in the environment, that the controller must avoid colliding with, accelerating or braking. Demon's available actions are $\mathfrak{l}^d$ to play game α, or $\mathfrak{r}^d$ to play game β.

(5) Test $?Q$ makes Angel immediately lose the current game if formula Q is false, but has no effect if Q is true. It can be used to limit the actions available to Angel modeling control system constraints. The action representing Angel going through the test (and possibly losing) is written as $(?Q)$. Dually, $!Q$ makes Demon lose the game immediately when Q is false, and has no effect if Q is true. It can model physical environment constraints. Demon's only available action, to take the test, is written as $(!Q)$.

Additionally, some game constructs do not involve Angel or Demon making any decisions but are still important to model overall functioning of the game.

(1) Assignment $x := e$ assigns the expression e to variable x. The corresponding action that Angel performs to change state is written as $(x := e)$.
(2) Sequential composition $\alpha;\beta$ runs game α followed by game β. The corresponding action that Angel performs is $\mathfrak{c}$ to start α.

dGL formula $\langle\alpha\rangle\,\phi$ is true in any state in which player Angel has a winning strategy to play hybrid game α so that regardless of what Demon does, in the end dGL formula ϕ holds true. As usual, state σ is a mapping from variables to real numbers, and $\sigma(x)$ is the real value associated with variable x. Dually, formula $[\alpha]\phi$ means Demon has a winning strategy to play game α so as to reach a state satisfying ϕ in the end. Appendix D recalls the denotational semantics. Eq. (1) shows the grammar of dGL hybrid games.

$$\begin{aligned}\alpha \equiv\ & x := e \mid \alpha;\beta \mid ?Q \mid \{x' = f(x)\&Q\} \mid \alpha^* \mid \alpha \cup \beta \mid x := * \\ & \mid !Q \mid \{x' = f(x)\&Q\}^d \mid \alpha^\times \mid \alpha \cap \beta \mid x := \otimes\end{aligned} \tag{1}$$

The *subgames* of a dGL game α are the games that are recursively composed to construct α, and correspond to the nodes of the abstract syntax tree (AST) induced by the grammar of Eq. (1).

We briefly describe the formal representation of actions and policies. Sections 4 and 5 will revisit these ideas with further detail. In any given initial state, a dGL game generates a game tree (Section 4.2) where each node corresponds to an action. A *play* of a game is a sequence of actions corresponding to a path from a root of the game tree to a leaf. Actions map old state to new state. For example, the action $(x := e)$ maps state σ to $\sigma(x \mapsto e)$, i.e., σ with value e replaced for x. Since agents and their gameplay are symmetric, in this paper we often focus on the Angelic perspective without loss of generality.

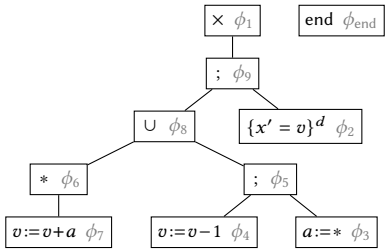
An Angelic (nondeterministic) *policy* is a function that maps every subgame and state to the set of actions that Angel can take at that subgame and state. *Henceforth in this paper, policy will be used to mean nondeterministic policy.* Angel *follows* an Angelic policy if at every subgame she must take a decision, she plays one of the actions permitted by the policy. An Angelic policy *wins* if Angel following it guarantees her reaching the end of the game in her winning region.

This paper synthesizes a winning policy for Angel, which characterizes the space of control actions that ensures her victory, given a dGL game and winning condition. We introduce a way to represent winning policies by capturing the player's *winning regions*, i.e., the set of game states from which the player has a way to win regardless of what the opponent does. As usual, formulas can be interpreted as the set of states in which they are true. Thus, formula $\langle\alpha\rangle\,\phi$ represents the winning region of Angel for game α and winning condition ϕ. Notation $\models \phi$ means formula ϕ is valid (true in all states).

3 Overview

1. Input
Game: $(((v := v + a)^* \cup (v := v - 1\,;\, a := *))\,;\, \{x' = v\}^d)^\times$ **Angel's Goal:** $x > 0$

2. Compute subvalue map



3. Policy for Angel derived from subvalue map

∪	Taking the left (resp. right) branch is allowed if ϕ_6 (resp. ϕ_5) is true.
∗	(Re-)entering the loop is allowed if ϕ_7 is true. Exiting is allowed if ϕ_2 is.
$a := *$	The value assigned to a must satisfy ϕ_2.

Computing the subvalue map:

Deduce $\phi_{\text{end}} \equiv x > 0$	(Goal)
Guess $\phi_1 \equiv \langle(((v := v + a)^* \cup (v := v - 1\,;\, a := *))\,;\, \{x' = v\}^d)^\times\rangle\, x > 0$	(Seek Invariant)
$\Leftarrow \langle((n{:=}*\,;?n{\geq}0\,;v{:=}v{+}an) \cup (v{:=}v{-}1\,;a{:=}*))\,;\{x'{=}v\}^d\rangle\, x{>}0$	(Refinement)
$\Leftrightarrow x > 0 \wedge (v \geq 0 \vee a > 0)$	(Axioms, QE)
Deduce $\phi_2 \equiv \langle\{x' = v\}^d\rangle\,\phi_1 \Leftrightarrow \forall t\,(t \geq 0 \rightarrow \phi_1(x \mapsto x + vt))$	(Solve ODE)
$\Leftrightarrow x > 0 \wedge v \geq 0$	(QE)
Deduce $\phi_3 \equiv \exists a\,\phi_2 \Leftrightarrow x > 0 \wedge v \geq 0, \quad \phi_5 \equiv \phi_4 \equiv \phi_3(v{-}1 \mapsto v) \Leftrightarrow x > 0 \wedge v \geq 1$	(Axioms, QE)
Guess $\phi_6 \equiv \langle(v := v + a)^*\rangle\,\phi_2 \Leftrightarrow \langle n{:=}*\,;?n{\geq}0\,;v{:=}v{+}an\rangle\,\phi_2$	(Invariant)
$\Leftrightarrow x > 0 \wedge (v \geq 0 \vee a > 0)$	(QE)
Deduce $\phi_7 \equiv \phi_6(v \mapsto v{+}a) \Leftrightarrow x > 0 \wedge (v + a \geq 0 \vee a > 0)$	
Check $\phi_6 \rightarrow \langle(?\phi_7\,;\, v := v + a)^*\,;\, ?\phi_2\rangle\phi_2$ valid	(Check Invariant)
Deduce $\phi_9 \equiv \phi_8 \equiv \phi_6 \vee \phi_5 \Leftrightarrow x > 0 \wedge (v \geq 0 \vee a > 0)$	
Check $\phi_1 \rightarrow \phi_9 \wedge x > 0$ valid	(Check Invariant)

Fig. 1. An *inductive subvalue* map for a simple hybrid game and the associated policy. The subvalue formulas are numbered in the order in which they are derived by the algorithm presented in Section 8. We will revisit this example throughout the paper, providing more explanations.

To illustrate our new concept of an *inductive subvalue map* and how it captures a policy (set of control solutions), let us consider the simple hybrid game from Fig. 1. In this game, Angel aims to ensure $x > 0$ after each iteration of a loop ($^\times$) controlled by Demon. At each iteration, Angel must choose ($\cup$) between two alternative paths, after which Demon continuously evolves x at rate v for a duration of his choice ($\{x' = v\}^d$). In the first path, Angel can increment v by an amount of a in a loop (*) as many times as she desires. In the second path, v is decremented by 1 and Angel gets an opportunity to arbitrarily set the value of a ($a := *$).

Our goal is to characterize not *one*, but *all* of the ways Angel can win (or at least *as many* as possible). At every choice point, we must determine which actions are compatible with Angel still winning the game and which actions are not. This way, Angel can pursue additional, unmodeled and possibly changing secondary goals without threatening her mission-critical objective of ensuring $x > 0$. In the traditional setting of control theory [8, 9] and reinforcement learning [56–58], this information is typically presented in the form of a *value function* that splits the state space into winning and losing states. This work proposes a representation of such value functions for hybrid games which is *sound*, *symbolic*, and which respects their *composable* structure.

Subvalue maps (Section 5). A *subvalue map* associates *each* subgame of a hybrid game to a formula that *conservatively* estimates the set of states from which the overall game can be won, *starting from* this specific subgame. In the example from Fig. 1, ϕ_1 provides a sufficient initial condition for winning the game. More interestingly, ϕ_5 provides a sufficient condition for winning the subgame starting from the second Angel branch within the loop. It must respect

$$\vDash\ \phi_5 \ \rightarrow\ \langle v := v-1\,;\, a := *\,;\, \{x' = v\}^d\,;\, \alpha^\times \rangle\, x > 0 \tag{2}$$

where α stands for the full Demon loop body. At every control point for Angel, the subvalue map indicates which actions can be taken without forfeiting winning. For example, when reaching the $\cup$ decision point, Angel can safely take the left branch if ϕ_6 is true. Similarly, she can safely take the right branch if ϕ_5 is. What if none of these hold though? Useful policies permit at least one action to be chosen. For this reason, Section 6 introduces the refined concept of an *inductive* subvalue map, whose local compatibility conditions ensure this property. In our example, these conditions mandate in particular that $\phi_8 \rightarrow \phi_6 \vee \phi_5$ is valid, ensuring that a safe action is always available to Angel at subgame $\cup$.

Internalizing policy reasoning. It is possible to reason about the nondeterministic policies represented by subvalue maps *within* dGL *itself*. In particular, the *universal projection* of a game onto a subvalue maps constructs a dGL game that wins precisely when *all plays that respect the subvalue-map policy* are guaranteed to win. The *existential projection* of a game onto a subvalue map wins when *there exists some policy-compliant play* that wins. These projections let us use dGL to state and prove properties, keeping the theory tidy. The safety of the monitor derived from an inductive subvalue map that keeps the game on track for winning can similarly be expressed in dGL itself. In our example, the following formula is *guaranteed* to be valid:

$$\begin{aligned}\phi_1 \rightarrow \big\langle \big(\big(?(\phi_6 \vee \phi_5)\,;\, \big(!\phi_6\,;\, ?\phi_6\,;\, (!\phi_7\,;\, v := v+a\,;\, ?\phi_6)^\times\,;\, !\phi_2\big) \cap \\ \big(!\phi_5\,;\, v := v-1\,;\, ?(\exists a\, \phi_2)\,;\, a := \otimes\,;\, !\phi_2\big)\big)\,;\, \{x' = v\}^d\big)^\times \big\rangle\, x > 0.\end{aligned} \tag{3}$$

In the game above, every Angel choice from the original game has been transferred to Demon, on the condition that he complies with Angel's subvalue strategy monitor. Compliance is always possible, as ensured by proper Angel tests.

Existence of an optimal subvalue map (Section 7). Subvalue maps can be ordered by precision, and thus by how permissive the policy is. Section 7 formalizes such an ordering, where the main subtlety is about ensuring that monitor guards are compared *in the context* in which they are reachable. An optimal subvalue map *exists* that corresponds to the dominant dGL strategy. In our example, this solution maps ϕ_5 to the right-hand side of the implication in Eq. (2). The optimal subvalue map construction achieves completeness of the optimal subvalue map synthesis problem.

Solving hybrid games via symbolic execution (Section 8). For practical applications, it is useful for subvalue maps to compute the set of available actions at a given state efficiently. An algorithmic framework for computing such subvalue maps of hybrid games is presented in Section 8. Like precondition calculus [21], it is parametrized on an invariant generation procedure. We propose *game rewriting* as a general principle for implementing this procedure (Section 8.1). We illustrate the resulting algorithm in Fig. 1. The algorithm leverages the fact that any dGL formula that is free of loops and whose ODEs have polynomial solutions can be automatically rewritten into an equivalent formula of propositional real arithmetic, by using the axioms of dGL along with quantifier elimination (QE) [60]. From there, it works backwards to bound the subvalue of all subgames. Whenever a loop is hit, a guess is needed for a subvalue of the associated subgame. Such a guess is typically made by using heuristics to conservatively rewrite it into a loop-free

dGL formula that can be symbolically evaluated. The algorithm then proceeds with the loop body, after which the validity of the guess is checked retrospectively. Section 8 provides more details. A subvalue map induces a *lower* bound of a game's value function. An *upper* bound can also be computed by considering the game from the opposing player's perspective. We can compare the dual subvalue maps and thereby symmetrically prove the optimality of the solution derived in Fig. 1.

The proofs in this paper largely follow from structural induction. *Appendix F provides all proofs throughout this paper.*

4 Prefixes and Suffixes

An *Angelic subvalue map* maps every dGL subgame to a formula denoting Angel's known winning subregion for the rest of the game starting at that subgame. As it is not always possible to compute exact winning regions, the *known* winning *sub*regions in a subvalue map are conservative, safe subsets of the actual theoretical winning regions per dGL semantics. Fig. 1 shows an example where each subgame, corresponding to nodes of the syntax tree, is mapped to its corresponding winning subregion ($\phi_1, \cdots \phi_9$). In this example, the winning *sub*regions are maximal, *equal* to the winning regions.

This section develops the formalism necessary for defining subvalue maps. A subvalue map maps each subgame to a formula that must logically imply the winning region for playing the rest of the game starting at that subgame. To characterize the winning region starting from a subgame, we first introduce the *game suffix*, or game remainder of a subgame. The game suffix of a subgame features *all* possible future behaviors of the overall game from *any* point where an instance of the subgame is encountered. Crucially, we characterize all these behaviors with a finite dGL game. Then, the exact winning region starting from a subgame will be defined as the winning region for the game suffix.

4.1 Game Suffix Construction

For disambiguation, each subgame is given a unique label. Notation $_a\alpha$ refers to the *named* game α where each subgame has a unique label, and the entire game α (corresponding to the root node of the AST) is labeled a. Even subgames that are identical in structure but appear at different places in the game are given different labels to distinguish multiple occurrences. subgames($_a\alpha$) refers to the set of all named subgames of $_a\alpha$. Game suffix $_a\alpha_{b:}$ is a game modeling what remains to be played after seeing some occurrence of subgame $_b\beta$ while playing $_a\alpha$ (Def. 4.2)[2]. This is a complex idea: unboundedly many different plays of a game can lead to playing a given subgame b, and even within a single play, b can occur an unbounded number of times. The ability to succinctly characterize all suffix plays (leveraging dGL) is crucial to the definition of subvalue maps.

Example 4.1. Let the overall game of Fig. 1 be $_a\alpha$, and let the label of each subgame be the index i of the winning subregion formula ϕ_i shown on its corresponding node. Consider subgame 5 inside loop α. All execution traces after any instance of subgame 5 have the remainder of the loop body ($\{x' = v\}^d$) followed by potential future iterations of the loop (α). So the game suffix $_a\alpha_{5:}$ of subgame 5 is $\{x' = v\}^d \,;\, \alpha$.

[2]The notation is analogous to Python list slicing. $_a\alpha_{b:}$ represents the part of $_a\alpha$ appearing *after* b. The complementary concept of *game prefix* is written as $_a\alpha_{:b}$, and consists of the part of the game appearing before subgame b.

Definition 4.2 (Game suffix). The game suffix $a{:}\alpha_{b:}$ of subgame $b{:}\beta$ in $\mathsf{subgames}(a{:}\alpha)$ is constructed as follows. If $a{:}\alpha$ is $b{:}\beta$ then $a{:}\alpha_{b:} = a{:}\alpha$. Otherwise, if $a{:}\alpha$ has structure:

$$a{:}(g{:}\gamma)^* \text{ or } a{:}(g{:}\gamma)^\times \text{ then } a{:}\alpha_{b:} = (g{:}\gamma_{b:})\,;\, a{:}\alpha$$

$$a{:}(g{:}\gamma \cup d{:}\delta) \text{ or } a{:}(g{:}\gamma \cap d{:}\delta) \text{ then } \begin{cases} a{:}\alpha_{b:} = g{:}\gamma_{b:} & b \in \mathsf{subgames}(g{:}\gamma) \\ a{:}\alpha_{b:} = d{:}\delta_{b:} & b \in \mathsf{subgames}(d{:}\delta) \end{cases}$$

$$a{:}(g{:}\gamma\,;\, d{:}\delta) \text{ then } \begin{cases} a{:}\alpha_{b:} = (g{:}\gamma_{b:})\,;\, d{:}\delta & b \in \mathsf{subgames}(g{:}\gamma) \\ a{:}\alpha_{b:} = d{:}\delta_{b:} & b \in \mathsf{subgames}(d{:}\delta) \end{cases}$$

In $a{:}(g{:}\gamma\,;\, d{:}\delta)$ and $a{:}(g{:}\gamma \cup d{:}\delta)$ cases, b is either in $\mathsf{subgames}(g{:}\gamma)$ or $\mathsf{subgames}(d{:}\delta)$ because b is a subgame of $a{:}\alpha$ (and not a itself which is already handled initially). Cases where $a{:}\alpha$ is atomic, i.e., $\alpha \in \alpha \in \{x := e, x := *, ?Q, !Q, \{x' = f(x)\ \&\ Q\}, \{x' = f(x)\ \&\ Q\}^d\}$, b must be a, so these cases are already handled initially.

Remark 1. The suffix characterizing behavior of a dGL game after a subgame *itself as a* dGL *game* is analogous to how Brzozowski derivatives reason about the behavior seen in a regular expression after a given string as a new regular expression [15].

4.2 Game Trees and Plays

This section discusses dGL gameplay to precisely connect subgame suffixes to their defining property in dGL operational semantics. However, later in this paper we will rely only on the syntactic suffix definition that has already been introduced. The background on dGL game trees is also useful for understanding *strategies*, which in Section 5 will provide a global view of the gameplay that a policy induces. This section first recounts which plays lie within a game and makes precise which suffix plays occur after a subgame. Then it defines a way to syntactically compare these play suffixes by the impact they have on state, *ignoring minor game structure differences*, to ensure that the *game suffix* has the same effect as these *suffix plays*, establishing that game suffixes reconstruct all gameplay that can occur after a subgame is seen. The definitions of game trees and plays that follow are from the existing operational semantics of dGL [44][Appendix C], extended to account for labels.

A labeled game $a{:}\alpha$, at a given initial state σ corresponds to a *labeled game tree* $\mathcal{T}(a{:}\alpha, \sigma)$ where each node is associated with a subgame label and an action. Using the conventions of descriptive set theory, the tree is represented as a prefix-closed set of the paths originating at the root (Fig. 2).

Example 4.3. In the game of Fig. 1 let the label of each subgame be the index i of the winning subregion formula ϕ_i shown on its corresponding node. Consider subgame 6, the inner loop. The game tree for this subgame has the paths $6 \cdot \mathfrak{s}$ for 0 iterations, $6 \cdot \mathfrak{g} \cdot 7 \cdot (v := v + a) \cdot 6 \cdot \mathfrak{s}$ for one iteration, and so on. Operator $\cdot$ concatenates actions/sequences of actions. This tree is visually represented in Fig. 2.

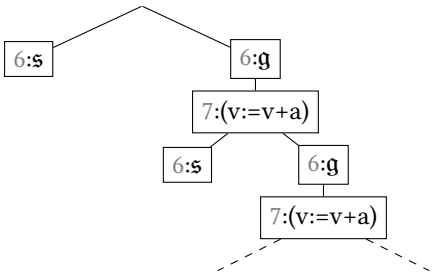
Def. G.1 in Appendix G.1 shows the full construction of trees[3]. A path in a tree can be represented as the sequence of labels and actions encountered while traversing the path, where the subgame label appears immediately before the action that subgame corresponds to. Because of loops, subgame labels can repeat even within a single path. For example, in Fig. 2 because of the Angelic loop, labels 6 and 7 repeat.

Tree nodes have actions, which correspond to transformations from old game state before the action to new state. Eq. (4) shows the transformations each action corresponds to, where notation

[3]This is similar to the standard dGL operational semantics [44, Appendix C], but with the addition of labels to keep track of subgames and some notation changes.

$$
\begin{aligned}
\text{Paths} \equiv \{ & \\
& (6\colon \mathfrak{s}), (6\colon \mathfrak{g}), \\
& (6\colon \mathfrak{g} \cdot 7\colon v \coloneqq v{+}a), \\
& (6\colon \mathfrak{g} \cdot 7\colon v \coloneqq v{+}a \cdot 6\colon \mathfrak{s}), \\
& (6\colon \mathfrak{g} \cdot 7\colon v \coloneqq v{+}a \cdot 6\colon \mathfrak{g} \cdot 7\colon v \coloneqq v{+}a), \\
& \cdots \}
\end{aligned}
$$

(a) Representation of game tree as (prefix-closed) set of paths through the tree.



(b) Visualization of the labeled game tree

Fig. 2. The game tree for subgame 6 of the game in Fig. 1, i.e., $6\colon(7\colon v := v + a)^*$

$\lfloor \mathfrak{a} \rfloor_\sigma$ is the state reached by running action $\mathfrak{a}$ starting at state σ. Labels have no impact on state and correspond to the identity function. For tests and ODEs, it is possible for the transition to be undefined. In this case, the player responsible for the action loses.

$$
\begin{aligned}
&\lfloor x := e \rfloor_\sigma = \sigma(x \mapsto e) \qquad \lfloor \mathfrak{a} \rfloor_\sigma = \sigma \text{ for } \mathfrak{a} \in \{\mathfrak{g}, \mathfrak{s}, \mathfrak{l}, \mathfrak{r}, \mathfrak{c}, \mathfrak{g}^d, \mathfrak{s}^d, \mathfrak{l}^d, \mathfrak{r}^d, \text{label}\} \\
&\lfloor ?Q \rfloor_\sigma = \begin{cases} \sigma & \text{if } \sigma \in [\![Q]\!] \\ \text{not defined} & \text{otherwise} \end{cases} \\
&\lfloor \{x' = f(x) \& Q @ t\} \rfloor_\sigma = \varphi(t) \text{ for the unique (differentiable) } \varphi : [0, t] \to \text{States}, \varphi(0) = \sigma, \\
&\text{for all } s \in [0, t] \left(\frac{\mathrm{d}\varphi(r)(x)}{\mathrm{d}r}(s) = f(\varphi(s)(x)) \wedge \varphi(s) \models Q \right). \text{ Not defined if no such } \varphi \text{ exists.}
\end{aligned}
\tag{4}
$$

Some transformations have no impact on state. These transformations correspond either to labels or to actions whose only role is to restrict tree structure. This set of these actions is $\mathfrak{S} = \{\mathfrak{g}, \mathfrak{s}, \mathfrak{l}, \mathfrak{r}, \mathfrak{c}, \mathfrak{g}^d, \mathfrak{s}^d, \mathfrak{l}^d, \mathfrak{r}^d\}$. A path is executed by inductively applying the actions: $\lfloor a \cdot t \rfloor_\sigma = \lfloor t \rfloor_{\lfloor a \rfloor_\sigma}$.

We syntactically compare paths and trees per the reachable states resulting from gameplay, ignoring structural differences, *by erasing the nodes that have no impact on state*. This is useful for characterizing subgame suffixes because they do not preserve structure, only the reachable states. The action projection operator $\pi_{\neg\mathfrak{S}}$ returns the path with all labels and actions belonging to $\mathfrak{S}$ removed (full construction in Appendix G.5, Def. G.7). For paths p_1 and p_2, if $\pi_{\neg\mathfrak{S}}(p_1) = \pi_{\neg\mathfrak{S}}(p_2)$, then $\lfloor p_1 \rfloor_\sigma = \lfloor p_2 \rfloor_\sigma = \lfloor \pi_{\neg\mathfrak{S}}(p_1) \rfloor_\sigma$.

$\pi_{\neg\mathfrak{S}}$ is overloaded to apply similarly to game trees. For game tree t, $\pi_{\neg\mathfrak{S}}(t)$ is a game tree where all paths have only actions that can impact state, produced by applying $\pi_{\neg\mathfrak{S}}$ to every path in t. Two trees result in the same reachable states (even if their structure differs) if their projections are the same. We are now ready to show the defining property of game suffixes.

Lemma 4.4 (Game Suffixes are Operational Gameplay Suffixes). *For subgame $b\colon\beta$ in overall game $a\colon\alpha$, let the game suffix be $a\colon\alpha_{b:}$. Let σ be some state. The states reachable from playing $a\colon\alpha$ such that σ is reached at subgame $b\colon\beta$ at some point during the gameplay are the same as the states reachable from playing $a\colon\alpha_{b:}$ starting in σ. That is,*

$$\{\pi_{\neg\mathfrak{S}}(s) \ : \ p \cdot b \cdot s \in \mathcal{T}(a\colon\alpha, \sigma'), \lfloor p \rfloor_{\sigma'} = \sigma \text{ and } \pi_{\neg\mathfrak{S}}(s) \neq ()\} = \pi_{\neg\mathfrak{S}}(\mathcal{T}(a\colon\alpha_{b:}, \sigma)),$$

where $()$ is the empty path.

In Lemma 4.4, the quantity on the left-hand side considers all paths in the original game that contain b. These must be of the form $p \cdot b \cdot s$ where p is a prefix of b and s is a suffix. The suffix set is built by collecting the projection of all such suffixes (ignoring sequences that are empty after applying $\pi_{\neg\mathfrak{S}}$). The second quantity on the right-hand side is the projection of the game tree of the subgame suffix at state σ. Although suffix $a{:}\alpha_{b:}$ may change game structure, ignoring $\mathfrak{S}$, it captures all remaining plays occurring after any occurrence of $b{:}\beta$ while playing $a{:}\alpha$.

4.3 Game Prefix

We also define the complementary concept of the *game prefix*, which will be useful to compute in *what contexts* a subgame can be seen. In Section 5, for example, the prefix of a subgame will help show that while following an Angelic subvalue map, Angel only reaches the subgame within the region from which she can win the rest of the game. The game prefix of subgame b in the game $a{:}\alpha$, written $a{:}\alpha_{:b}$, features *all* possible behaviors of game α that can happen *before* an occurrence of subgame b within $a{:}\alpha$. It is constructed analogously to the game suffix (Appendix G.7, Def. G.9) and satisfies an analogous property (Lemma 4.5).

Lemma 4.5 (Execution Prefixes are Original Gameplay Prefixes). *For subgame $b{:}\beta$ in $a{:}\alpha$, for any initial state σ, the states reachable at $b{:}\beta$ while playing $a{:}\alpha$ are equal to those reachable by playing game prefix $a{:}\alpha_{:b}$. That is, prefix set $\{\pi_{\neg\mathfrak{S}}(p) \,:\, p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma)\}$ is equal, under prefix closure, to prefix game tree $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha_{:b}, \sigma))$*[4].

5 Subvalue Maps

This section defines subvalue maps, shows their interpretation as policies, and discusses the properties of these policies.

5.1 Subvalue Map Definition

Def. 5.1 characterizes a subvalue map. Each subgame maps to a formula that must logically imply the winning region for playing the rest of the game starting at that subgame. The exact winning region starting from a subgame can be defined as the winning region for the game suffix. A special end subgame additionally stores the terminal winning subregion of the overall game that the policy induced by the subvalue map tries to reach. Mathematically, when Angel is playing the overall game $a{:}\alpha$ with winning condition $S(\mathsf{end})$, her winning region at the point when she is about to play subgame $b{:}\beta$ to achieve $S(\mathsf{end})$ overall is $\langle a{:}\alpha_{b:}\rangle S(\mathsf{end})$.

Definition 5.1 (Subvalue maps). A map S from the subgames of game $a{:}\alpha$ and special subgame end to formulas is an *Angelic subvalue map* of $a{:}\alpha$ when $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\mathsf{end})$ for every subgame $b{:}\beta$ in $\mathsf{subgames}(a{:}\alpha)$. Dually, S is a *Demonic subvalue map* for game $a{:}\alpha$ when $\models S(\beta) \to [a{:}\alpha_{b:}]S(\mathsf{end})$ for every subgame $b{:}\beta$ in $\mathsf{subgames}(a{:}\alpha)$.

Interpretation as policy. Angelic subvalue maps effectively provide Angel with a (nondeterministic) policy that describes how to play the game. For example, the map in Fig. 1 says that Angel at her choice $\cup$ should not transition to $(v := v+a)^*$ when ϕ_6 is false, and not transition to $v := v-1\,;\ a := *$ when ϕ_5 is false. The intuition is, at any given subgame where Angel has a choice (e.g. Fig. 1 at $\cup$), the *next subgame* that she chooses to run (e.g., left branch $(v := v + a)^*$) should be such that the current state σ satisfies the winning subregion for that subsequent subgame (for the example,

[4]The prefix closure is required to recover tree structure. Unlike with the suffix construction, the prefix set builder collects only *parent* subgames of $b{:}\beta$ and *not all predecessors*. Expanding the prefix closure results in the following condition:

$$\{\pi_{\neg\mathfrak{S}}(t) \,:\, p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma) \text{ and } p = t \cdot u\} = \pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha_{:b}, \sigma))$$

$\sigma \models \phi_6$). This next subgame to run after the current one ends is called the *successor*, identified as Def. 5.2 shows [5].

Definition 5.2 (Successor). The successor of subgame $b{:}\beta$ in overall game $a{:}\alpha$, written $\text{succ}(b, a{:}\alpha)$ is defined as follows. If $a{:}\alpha$ is $b{:}\beta$ then $\text{succ}(b, a{:}\alpha)$ is the special subgame end. Otherwise, if $a{:}\alpha$ has structure:

$$a{:}(g{:}\gamma\,;\, d{:}\delta) \text{ then } \begin{cases} \text{succ}(b, a{:}\alpha) = d{:}\delta & b \in \text{subgames}(g{:}\gamma) \text{ and } \text{succ}(b, g{:}\gamma) = \text{end} \\ \text{succ}(b, a{:}\alpha) = \text{succ}(b, g{:}\gamma) & \text{otherwise if } b \in \text{subgames}(g{:}\gamma) \\ \text{succ}(b, a{:}\alpha) = \text{succ}(b, d{:}\delta) & \text{otherwise} \end{cases}$$

$$a{:}(g{:}\gamma)^* \text{ or } a{:}(g{:}\gamma)^\times \text{ then } \begin{cases} \text{succ}(b, a{:}\alpha) = a & b \in \text{subgames}(g{:}\gamma) \text{ and } \text{succ}(b, g{:}\gamma) = \text{end} \\ \text{succ}(b, a{:}\alpha) = \text{succ}(b, g{:}\gamma) & \text{otherwise if } b \in \text{subgames}(g{:}\gamma) \end{cases}$$

$$a{:}(g{:}\gamma \cup d{:}\delta) \text{ or } a{:}(g{:}\gamma \cap d{:}\delta) \text{ then } \begin{cases} \text{succ}(b, a{:}\alpha) = \text{succ}(b, g{:}\gamma) & b \in \text{subgames}(g{:}\gamma) \\ \text{succ}(b, a{:}\alpha) = \text{succ}(b, d{:}\delta) & \text{otherwise} \end{cases}$$

Using the successor definition, Def. 5.3 shows how in general, Angel can interpret the subvalue map as a (nondeterministic) policy indicating what control decisions are safe to take to eventually win based on the current state.

Definition 5.3 (Policy Interpretation). Policy interpretation function $\mathcal{P}_{a:\alpha}(S)$ transforms an Angelic subvalue map S for game $a{:}\alpha$ to a nondeterministic policy function that maps current state σ and subgame $b{:}\beta$ to the set of acceptable control actions Angel can take. $\mathcal{P}_{a:\alpha}(S)(b, \sigma)$ is defined as below. When the current subgame $b{:}\beta$ has structure:

$$b{:}(d{:}\delta \cup g{:}\gamma) \text{ then } \begin{cases} \{\mathfrak{l}, \mathfrak{r}\} & \sigma \models S(d) \wedge S(g) \\ \{\mathfrak{l}\} & \text{else if } \sigma \models S(d) \\ \{\mathfrak{r}\} & \text{else if } \sigma \models S(g) \\ \emptyset & \text{otherwise} \end{cases}$$

$$b{:}(g{:}\gamma)^* \text{ then } \begin{cases} \{\mathfrak{s}, \mathfrak{g}\} & \sigma \models S(g) \wedge S(\text{succ}(b, a{:}\alpha)) \\ \{\mathfrak{g}\} & \text{else if } \sigma \models S(g) \\ \{\mathfrak{s}\} & \text{else if } \sigma \models S(\text{succ}(b, a{:}\alpha)) \\ \emptyset & \text{otherwise} \end{cases}$$

$$b{:}x := * \text{ then } \{(x := e) \mid \sigma(x \mapsto e) \models S(\text{succ}(b, a{:}\alpha))\}$$

$$b{:}\{x' = f(x)\,\&\,Q\} \text{ then } \{(x' = f(x)\,\&\,Q\,@\,t) \;:\; t \geq 0, \forall s \in [0,t], \varphi(s) \models S(b) \text{ and } \varphi(t) \models S(\text{succ}(b, a{:}\alpha)) \text{ where } \varphi : [0,t] \to \mathcal{S} \text{ is differentiable}, \varphi(0) = \sigma, \text{ and } \forall s \in [0,t],\ \varphi(s) \models x' = f(x) \wedge Q\}$$

Relationship with strategies. We have focused on *policies* which provide local answers for what action to play next. *Strategies*, on the other hand, are global gameplay solutions. The Angelic strategy at a given initial state corresponds to a game tree at that state that has been pruned so that at every subgame where Angel has to take a decision, she takes exactly one decision. Thus, a strategy plans out all the future moves of its player for all potential actions of the opponent for all

[5]The successor is not just the next label to occur in the game suffix. This is because the successor should not be a child of the current subgame. It should only start when the current subgame ends.

parts of the game at once. Viewing policies from the lens of strategies is useful for understanding the global behaviors that a policy induces, and will be relevant in Section 5.2 and Section 5.3. The set of strategies that Angel can play for the game $a\alpha$ while following the policy of a subvalue map S starting in a state σ is written as $\mathcal{S}_{a\alpha}(S)(\sigma)$. The construction is shown in Appendix G.5, Def. G.6. The intuition is to generate the game tree while restricting Angel to play per the policy. When the policy has multiple options at a given Angelic decision point, $\mathcal{S}_{a\alpha}(S)(\sigma)$ has different strategies for each way to proceed. When the policy is deterministic, the set contains a single strategy.

Winning games. Subvalue maps induce policies that are suitable to play games with *compatible* winning conditions, i.e., winning conditions that contain the terminal subregion of the policy that end maps to (Def. 5.4).

Definition 5.4 (Compatible winning conditions). An Angelic subvalue map S for game $a\alpha$ is said to be *compatible with Angel winning condition* ϕ when $\models S(\mathsf{end}) \rightarrow \phi$. Dually, a Demonic subvalue map S for game $a\alpha$ is compatible with Demon winning condition ϕ when $\models S(\mathsf{end}) \rightarrow \phi$.

If Angel plays a game for a compatible winning condition per a policy induced by an Angelic subvalue map, she always remains in a state where there *exists* a way for her to win the remaining game (Theorem 5.9 at the end of this section). However, the policy she needs to play to win *might not lie within the subvalue map.*

To see what this can look like, for the example of Fig. 1 at Angel choice $\cup$, consider the subvalue map in the figure except that ϕ_6 is $\bot$ instead of $x > 0 \land (v \geq 0 \lor a > 0)$. This is still an Angelic subvalue map. If we arrive at ϕ_8 in state $\sigma \equiv \{x = 0, v = 0, a = 0\}$, then Angel can win by choosing the left branch guarded by ϕ_6, but will lose if she chooses the right branch. The policy induced by the map at this point does not let Angel into the losing play of choosing the right branch. However it does not let her choose the left branch either since $\sigma \not\models \bot$. Its returned action set is $\emptyset$. Angel is stuck. This motivates the definition of an *inductive* subvalue map (Section 6), where not only does the policy keep Angel in a state where she has a way to win, but this victory can be achieved by continuing to follow the subvalue map policy.

We next develop the formalism to say *for all ways to play the policy of the subvalue map* (Section 5.3) up to any subgame, *there exists a way to continue to play within the subvalue map* (Section 5.2). We formulate this quantification over the ways to play a subvalue map within dGL.

5.2 Existential Subvalue Map Projection

The *existential projection* of game $a\alpha$ onto Angelic subvalue map S, written $a\alpha \Supset S$, is the game that can be won exactly when there *exists* a way for Angel to win $a\alpha$ *while restricted to stay within the (nondeterministic) policy of* S. Projection changes the game so that Angel is put on rails, limited to the policy induced by S while taking every decision, and is forced to manifest what she must do to win while playing within the policy.

Lemma 5.5 (Existential projection correspondence). *For any Angelic subvalue map S for game $a\alpha$ and compatible winning condition ϕ, in initial state $\sigma \models \langle\alpha\rangle\,\phi$, Angel has a winning strategy for the game $a\alpha \Supset S$ if and only if Angel can win $\langle\alpha\rangle\,\phi$ while following the policy of S. That is, $\sigma \models \langle a\alpha \Supset S\rangle\,\phi$ iff $\mathcal{S}_{a\alpha}(S)(\sigma) \neq \emptyset$.*

Def. 5.6 defines existential projection by inserting Angelic tests to guard *every* Angelic decision that would make Angel lose if her choice would result in playing into the subsequent subgame in a state outside that subgame's winning subregion. For example, in Fig. 1, the subvalue map says that ϕ_2 is the winning subregion of the subgame that is played after subgame $3a := *$. The projection of the subvalue map onto subgame 3 is $a := *;\ ?\phi_2$, thus *locally* ensuring that Angel makes a correct

assignment to a that will let Angel win the rest of the game. When Angel has no decisions to take, e.g., $g{:}\gamma \cap d{:}\delta$ where *Demon* chooses between $g{:}\gamma$ and $d{:}\delta$, projection does not interfere, only recursively adding guards for any Angelic decisions within subgames γ and δ. During recursive calls, we sometimes have to update the end subvalue to reflect the appropriate terminal condition for the subgame under consideration. Notation $S(\text{end} \mapsto Q)$ refers to S with the subvalue of end updated to Q.

Definition 5.6 (Existential projection). The *existential projection* of dGL game $a{:}\alpha$ onto Angelic subvalue map S, written $a{:}\alpha \exists\!\downarrow S$, is generated recursively from the structure of $a{:}\alpha$ as follows. If $a{:}\alpha$ has structure:

$$a{:}x := * \text{ then } x := *\,;\ ?S(\text{end}). \quad a{:}\{x' = f(x)\ \&\ Q\} \text{ then } \{x' = f(x)\ \&\ Q\}\,;\ ?S(\text{end})$$

$$a{:}(g{:}\gamma \cup d{:}\delta) \text{ then } (?S(g)\,;\ g{:}\gamma \exists\!\downarrow S) \cup (?S(d)\,;\ d{:}\delta \exists\!\downarrow S)$$

$$a{:}(g{:}\gamma)^* \text{ then } (?S(g)\,;\ (g{:}\gamma \exists\!\downarrow (S(\text{end} \mapsto S(a))))^*\,;?S(\text{end})$$

$$a{:}(g{:}\gamma; d{:}\delta) \text{ then } (g{:}\gamma \exists\!\downarrow (S(\text{end} \mapsto S(d))))\ ;\ (d{:}\delta \exists\!\downarrow S)$$

$$a{:}(g{:}\gamma \cap d{:}\delta) \text{ then } g{:}\gamma \exists\!\downarrow S \cap d{:}\delta \exists\!\downarrow S \quad a{:}(g{:}\gamma)^\times \text{ then } \left(g{:}\gamma \exists\!\downarrow (S(\text{end} \mapsto S(a)))\right)^\times$$

atomic and not controlled by Angel, i.e.,

$$\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}, \text{ then } \alpha$$

where game labels are preserved and fresh labels are used for newly introduced subgames. The existential projection of a Demonic subvalue map is symmetric (Appendix G.3), with the addition of Demonic assertions after subgames controlled by Demon.

Remark 2. $a{:}\alpha \exists\!\downarrow S$ is an *Angelic refinement* of $a{:}\alpha$, i.e., for any winning condition ψ, $\models \langle a{:}\alpha \exists\!\downarrow S\rangle\, \psi \rightarrow \langle\alpha\rangle\, \psi$ (Lemma F.3). Intuitively, if Angel has a winning strategy within S, she can win with the same strategy in the unconstrained game.

5.3 Universal Subvalue Map Projection

The *universal projection*, written $a{:}\alpha \forall\!\downarrow S$, explores a subvalue map maximally flexibly by making Demon play Angel's policy in Angelic subvalue map S to look for loopholes. It changes the game to put *Demon* on rails forcing him to play within Angel's S, as he looks for *any* strategy in the policy that falsifies Angel's winning condition.

Lemma 5.7 (Universal projection correspondence). *For any Angelic subvalue map S for game $a{:}\alpha$, in initial state $\sigma \models \langle\alpha\rangle\, a$, for compatible winning condition ϕ, Angel has a winning strategy for the game $a{:}\alpha \forall\!\downarrow S$ if and only if all ways to pursue the policy of S for game $a{:}\alpha$ that complete the game end in Angel's winning region. That is, $\sigma \models \langle a{:}\alpha \forall\!\downarrow S\rangle\, \phi$ iff all elements of set $\mathcal{S}_{a:\alpha}(S)(\sigma)$ are Angel winning strategies.*

Def. 5.8 defines universal projection by replacing all Angelic games $(\cup, x := *, \alpha^*, \{x' = f(x)\&Q\})$ by their dual Demonic games and inserting Demonic tests to ensure Demon plays these decisions within Angel's control envelope. Like in projection, control choices that were already Demon's remain unguarded. Eq. (3) shows universal projection for the example subvalue map and game in Fig. 1.

Definition 5.8 (Universal projection). The *universal projection* of Angelic subvalue map S onto dGL game $a{:}\alpha$, written $a{:}\alpha \forall\!\downarrow S$, is generated recursively per the structure of $a{:}\alpha$ as follows. If $a{:}\alpha$ has structure:

$$a{:}x := *, \text{ then } x := \otimes\,;\ !S(\text{end}). \quad a{:}(g{:}\gamma; d{:}\delta), \text{ then } (g{:}\gamma \forall\!\downarrow (S(\text{end} \mapsto S(d))))\ ;\ (d{:}\delta \forall\!\downarrow S)\,.$$

$a{:}\{x' = f(x)\ \&\ Q\}$, then $\{x' = f(x)\ \&\ Q\}^d$; $!S(\text{end})$.

$a{:}(g{:}\gamma \cup d{:}\delta)$, then $(!S(g)\,;\, g{:}\gamma \between S) \cap (!S(d)\,;\, d{:}\delta \between S)$.

$a{:}(g{:}\gamma)^*$, then $(!S(g)\,;\, (g{:}\gamma \between (S(\text{end} \mapsto S(a)))))^\times\,;\, !S(\text{end})$.

$a{:}(g{:}\gamma \cap d{:}\delta)$, then $(g{:}\gamma \between S) \cap (d{:}\delta \between S)$. $a{:}(g{:}\gamma)^\times$, then $(g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times$.

atomic and not controlled by Angel, i.e.,

$\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then α.

where game labels are preserved (including for Angel-controlled subgames transformed to their Demon-controlled equivalent) and fresh labels are used for newly introduced subgames. The universal projection of a Demonic subvalue map is symmetric (see Appendix G.4).

We can now show the defining property of subvalue maps.

Theorem 5.9 (Subvalue Map stays in Winning Region). *Suppose S is an Angelic subvalue map for game $a{:}\alpha$ compatible with winning condition ϕ, i.e., for every subgame $b \in \text{subgames}(a{:}\alpha)$, $\models S(b) \rightarrow \langle a{:}\alpha_{b:}\rangle\, S(\text{end})$ and $\models S(\text{end}) \rightarrow \phi$. Upon starting in any state $\sigma \in [\![S(a)]\!]$, and reaching subgame b by following any strategy induced by the subvalue map S, there exists a winning strategy for Angel to win the remainder of the game. That is, $S(a) \models \langle (a{:}\alpha \between S)_{:b}\rangle\, \langle a{:}\alpha_{b:}\rangle\, \phi$.*

As discussed in Section 5.1, the property of Theorem 5.9 is critical but not alone strong enough, since it does not guarantee that following the policy will ensure Angel wins. The property it maintains is only that Angel always continues to have some winning strategy ($\langle a{:}\alpha_{b:}\rangle\, \phi$), not that such a strategy is within the policy ($\langle a{:}\alpha \between S_{b:}\rangle\, \phi$). Angel might get stuck without a valid choice permitted by the map S. The *inductive* subvalue map introduced next prevents this situation, disqualifying subvalue maps where Angel gets stuck.

6 Inductive Subvalue Maps

To correct for the shortcoming of subvalue maps by preventing players from getting stuck, we introduce *inductive* subvalue maps. Notation $a{:}\alpha \models S$ indicates S is an *inductive* subvalue map for game $a{:}\alpha$. Theorem 6.1 shows the characteristic property of an inductive subvalue map: its policy always keeps the player in a region where the player can win the rest of the game *by following the subvalue map policy.*

Theorem 6.1 (Inductive subvalue map ensures winning actions). *Suppose S is an inductive Angelic subvalue map for game $a{:}\alpha$ and compatible with winning condition ϕ, i.e., $a{:}\alpha \models S$ and $S(\text{end}) \models \phi$. Upon starting in any state $\sigma \models S(a)$, and reaching subgame b by following any strategy induced by the subvalue map S, there is a way for Angel to win by continuing to follow S. That is, $S(a) \models \langle (a{:}\alpha \between S)_{:b}\rangle\, \langle (a{:}\alpha \between S)_{b:}\rangle\, \phi$.*

It is possible to characterize an inductive subvalue map using recursive conditions expressed via dGL formulas (Def. 6.2).

Definition 6.2 (Inductive subvalue maps). Let S be a map from the subgames of $a{:}\alpha$ to winning subregions. S is an *inductive Angelic subvalue map* for game $a{:}\alpha$, written $a{:}\alpha \models S$, when the following holds. If $a{:}\alpha$ has structure:

atomic, i.e., $\alpha \in \{x := e, x := *, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}, \{x' = f(x)\ \&\ Q\}^d\}$ then $\models S(a) \rightarrow \langle\alpha\rangle\, S(\text{end})$.

$a{:}(g{:}\gamma \cup d{:}\delta)$ then $\models S(a) \rightarrow S(g) \vee S(d)$ and $g{:}\gamma \models S$ and $d{:}\delta \models S$.

$a{:}(g{:}\gamma \cap d{:}\delta)$ then $\models S(a) \rightarrow S(g) \wedge S(d)$ and $g{:}\gamma \models S$ and $d{:}\delta \models S$.

$$a{:}(g{:}\gamma\,;\, d{:}\delta) \text{ then } \models S(a) \to S(g) \text{ and } g{:}\gamma \models S(\mathsf{end} \mapsto S(d)) \text{ and } d{:}\delta \models S.$$
$$a{:}(g{:}\gamma)^* \text{ then } \models S(a) \to \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\, S(\mathsf{end}) \text{ and } g{:}\gamma \models S(\mathsf{end} \mapsto S(a)).$$
$$a{:}(g{:}\gamma)^\times \text{ then } \models S(a) \to S(g) \wedge S(\mathsf{end}) \text{ and } g{:}\gamma \models S(\mathsf{end} \mapsto S(a)).$$

Symmetric conditions characterize when S is an *inductive Demonic subvalue map* for game $a{:}\alpha$ (Appendix G.6).

Example 6.3. Consider again the Angelic choice in Fig. 1. To establish $S \models 8 : (6{:}(v := a)^* \cup 5{:}(v := v - 1\,;\, a := *))$, Def. 6.2 first checks that if $S(8)$ (i.e., ϕ_8) holds then either Angel can win by choosing to run subgame 6 (when ϕ_6 holds) or subgame 5 (when ϕ_5 holds). It then also recursively ensures that after going to subgame 5 or 6, S continues to provide a valid Angelic strategy ($6{:}(v := v + a)^* \models S$ and $5{:}(v := v - 1\,;\, a := *) \models S$).

We discuss the subtle loop cases of Def. 6.2. For Demonic loop validity $a{:}(g{:}\gamma)^\times \models S$, the following conditions should hold.

(1) $\models S(a) \to S(\mathsf{end})$, and $\models S(a) \to \langle\gamma\rangle S(a)$, i.e. S(a) is an invariant that holds inductively and implies the postcondition. Inductiveness condition $\models S(a) \to \langle\gamma\rangle S(a)$ does not appear in Def. 6.2 because it is already implied by later conditions $\models S(a) \to S(g)$ and $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))$.

(2) $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))$, i.e. the subvalue map for the loop body ensures finishing within the safe invariant. Using instead the more liberal condition $g{:}\gamma \models S$ to finish the loop body within the postcondition is unsound: Demon can choose to repeat the loop indefinitely, so Angel must play to remain within the inductive region. Alternative condition $g{:}\gamma \models S(\mathsf{end} \mapsto S(g))$ is also unsound since then $S(g)$ is merely inductive and does not have to imply reaching desired postcondition $S(\mathsf{end})$.

(3) $\models S(a) \to S(g)$. In case Demon chooses to run the loop, Angel is guaranteed to not get stuck while playing the loop body in winning subregion $S(g)$, so overall loop winning subregion $S(a)$ should not exceed $S(g)$. On the other hand, $S(g)$ can be weaker than $S(a)$ as it can be one loop iteration away from implying $S(\mathsf{end})$, while $S(a)$ must already imply $S(\mathsf{end})$.

For Angelic loop validity $a{:}(g{:}\gamma)^* \models S$, the following conditions must hold.

(1) $\models S(a) \to \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\, S(\mathsf{end})$, i.e., Angelic loop invariant $S(a)$ ensures that there is a *well-founded* strategy to exit while staying within the control envelope. Angel should not get stuck in a state where her only option is playing the loop forever. Using instead the more liberal condition $\models S(a) \to \langle\alpha\rangle\, S(\mathsf{end})$ is not enough, since Angel can get stuck in a state where there is a way to exit the loop but only when going outside the subvalue map (see Appendix B.1).

(2) $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))$, i.e., the subvalue map for the loop body ensures that after playing the loop body, there is still a well-founded exit strategy within the subvalue map ($S(a)$). Using condition $g{:}\gamma \models S(\mathsf{end} \mapsto S(g) \vee S(\mathsf{end}))$ instead is unsound as then $S(g)$ is merely inductive and does not have to imply an exit strategy to reach postcondition $S(\mathsf{end})$ (see Appendix B.2).

Theorem 6.4 below proves that *all* inductive subvalue maps are subvalue maps, indicating that the validity conditions compose to produce correct Angel and Demon winning subregions. Consequently, for suitable initial states, while following an inductive subvalue map, the player will never reach a state from which it cannot win (Theorem 5.9). Additionally, Theorem 6.1 at the start of this section shows that following the policy induced by an inductive subvalue map ensures that the agent will never reach a state where it is stuck. In combination, these mean that starting in a suitable initial state, while following an inductive subvalue map, a player cannot lose.

Theorem 6.4 (Inductive subvalue maps are subvalue maps). *For* dGL *game* $a{:}\alpha$*, if* S *is an inductive Angelic subvalue map (*$a{:}\alpha \models S$*), then it is also an Angelic subvalue map for* $a{:}\alpha$*, i.e., for every subgame* $b{:}\beta$ *in* subgames($a{:}\alpha$)*,* $\models S(b) \rightarrow \langle a{:}\alpha_{b:}\rangle S(\mathsf{end})$*. Dually, if* S *is an inductive Demonic subvalue map for* $a{:}\alpha$*, then* S *is a Demonic subvalue map for* $a{:}\alpha$*.*

Def. 6.5 shows how the dominant strategy to win a dGL game can be captured as an inductive subvalue map, indicating that Def. 6.2 is sufficiently liberal. Angel's *dominant* strategy for a game is one that wins in the *largest* possible set of states. Def. 6.5 constructs the inductive subvalue map inducing a dominant strategy indicating subvalues via modal dGL formulas. The winning subregion for the overall game is maximal.

Definition 6.5 (Model predictive subvalue map). For the game $a{:}\alpha$ and win condition ϕ, the *model predictive Angelic subvalue map* $\{\mathsf{end} \mapsto \phi\} \cup \bigcup_{b\in\mathsf{subgames}(a:\alpha)} \{b \mapsto \langle a{:}\alpha_{b:}\rangle \phi\}$ maps every subgame b of $a{:}\alpha$ to the optimal winning regions of its game suffix. Dually, the *model predictive Demonic subvalue map* is $\{\mathsf{end} \mapsto \phi\} \cup \bigcup_{b\in\mathsf{subgames}(a:\alpha)} \{b \mapsto [a{:}\alpha_{b:}]\phi\}$.

Example 6.6. For the game of Fig. 1, the model predictive Angelic subvalue map maps subgame 5 to $\langle \{x' = v\}^d ; \alpha\rangle\, x > 0$, which per the semantics of dGL, precisely means the condition under which Angel can win by reaching her assigned winning conditions after playing the remainder of the game $\{x' = v\}^d ; \alpha$.

Remark 3. Model Predictive Control (MPC) [18, 26, 51] is a well-known controller design principle that uses a mathematical model to predict controller behavior and back-computes the control conditions that respect the desired system constraints. Def. 6.5 extends this construction to hybrid games logic. game suffix $\langle a{:}\alpha_{b:}\rangle\phi$ serves as an exact model of future game behavior after subgame b[6].

Remark 4. A runtime monitor (constructed similar to existing work on monitoring dL [38] and constructive dGL [14]) raises an alarm or intervenes to enforce a correct fallback when gameplay deviates from the control envelope. It is possible to model the use of a subvalue map as a *runtime monitor* in a single dGL formula. For an Angelic subvalue map, this *monitored game* is constructed by (1) giving Demon control over Angelic decisions while restricting him to play per the subvalue map (like in universal projection) to mimic an unverified controller that plays unreliably while being forced to stay within the control envelope by a runtime monitor and (2) ensuring before each such monitored subgame that the subvalue map allows *some* control action using Angelic tests. Eq. (3) shows the monitored game for the running example. The *monitored game* for dGL game $a{:}\alpha$ and Angelic subvalue map S, written $a{:}\alpha\lightning S$, is generated recursively per the structure of $a{:}\alpha$ as follows. If $a{:}\alpha$ has structure:

$a{:}x := *$, then $?(\exists x\, S(\mathsf{end})) ; x := \otimes ; !S(\mathsf{end})$. $a{:}(g{:}\gamma; d{:}\delta)$, then $g{:}\gamma\lightning S(\mathsf{end} \mapsto S(d)) ; d{:}\delta\lightning S$.

$a{:}\{x' = f(x)\ \&\ Q\}$, then $?\langle\alpha\rangle\, S(\mathsf{end}) ; \{x' = f(x)\ \&\ Q\}^d ; !S(\mathsf{end})$.

$a{:}(g{:}\gamma \cup d{:}\delta)$, then $?(S(g) \vee S(d)) ; ((!S(g) ; g{:}\gamma\lightning S) \cap (!S(d) ; d{:}\delta\lightning S))$.

$a{:}(g{:}\gamma)^*$, then $?S(a) ; (!S(g) ; g{:}\gamma\lightning S(\mathsf{end} \mapsto S(a)) ; ?S(a))^\times ; !S(\mathsf{end})$.

$a{:}(g{:}\gamma \cap d{:}\delta)$, then $(g{:}\gamma\lightning S) \cap (d{:}\delta\lightning S)$. $a{:}(g{:}\gamma)^\times$, then $(g{:}\gamma\lightning S(\mathsf{end} \mapsto S(a)))^\times$.

atomic and not controlled by Demon, i.e.,

$\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then α.

[6] Though unlike standard MPC, as we are computing maximal control sets, there is no optimization step to identify the best control solution within the set.

Monitoring a Demonic subvalue map is symmetric (see Appendix G.4). Every inductive subvalue map S is a sound runtime monitor, i.e., $\models S(a) \rightarrow \langle a{:}\alpha \lightning S\rangle\, S(\text{end})$ (Theorem F.10).

In subsequent sections, Def. 7.1 provides a partial ordering on inductive subvalue maps where more permissive maps are better, and Theorem 7.2 shows that the dominant dGL strategy corresponds to the maximally permissive inductive subvalue map. Finally, Section 8 uses Def. 6.2 to define a natural, symbolic execution based algorithm that synthesizes subvalue maps that generate policy decisions in polynomial time.

7 Maximal Solution

For a given game and winning condition, there are many possible correct inductive subvalue maps. Most useful are the *more permissive* subvalue maps that allow as many control actions as possible. More permissive subvalue maps leave more control options open, permitting more aggressive, efficient control strategies. We formalize subvalue map permissiveness with a partial ordering where more permissive subvalue maps are greater than less permissive subvalue maps. A natural candidate for ordering the subvalue maps of game $a{:}\alpha$ would be to say that S is at least as permissive as S' ($S \sqsupseteq S'$) when for each subgame b in $\text{subgames}(a{:}\alpha)$, *every state in subregion* $S'(b)$ *is also within subregion* $S(b)$, i.e., $\models S'(b) \rightarrow S(b)$, so that $S(b)$ locally *permits* the agent to play into subgame b in all the states that $S'(b)$ does. However, this ordering fails to order cases like the following. Let S' set the winning region for the overall game to be empty ($S'(a) = \bot$), meaning that Angel is never allowed to play at all. For *any* other inductive subvalue map S, a good ordering should indicate that $S \sqsupseteq S'$ since S cannot possibly be less permissive, but the proposed naïve ordering does not. The winning subregion for some later subgame $b \neq a$ can be such that the check $\models S'(b) \rightarrow S(b)$ fails (see Appendix A for details).

An improved ordering makes the following modified check for every subgame b. $S \sqsupseteq S'$ when, *amongst the states that are reachable at subgame* b *while playing per subvalue map* S', every state that is in subregion $S'(b)$ should also be within subregion $S(b)$. To express this logically, we first characterize the states that are reachable at subgame b while playing per subvalue map S'.

The game prefix (Def. G.9) of subgame b in the game $a{:}\alpha$, written $a{:}\alpha_{:b}$, features *all* possible behaviors of game α that can happen *before* an occurrence of subgame b within $a{:}\alpha$. To characterize all the states reachable at subgame b while following the policy induced by S', we use the universal projection of S' onto the game prefix $(a{:}\alpha_{:b}) \Upsilon S'$. Def. 7.1 then compares the winning subregions of S and S' after running the projected game to filter out unreachable states.

Definition 7.1 (Permissiveness ordering). For two inductive Angelic subvalue maps S and S' for the game $a{:}\alpha$, S is at least as good as S', written $S \sqsupseteq S'$, iff for each subgame b of $a{:}\alpha$, $\models \langle a{:}\alpha_{:b} \Upsilon S'\rangle (S'(b) \rightarrow S(b))$ and $\models \langle a{:}\alpha \Upsilon S'\rangle (S'(\text{end}) \rightarrow S(\text{end}))$. Dually, for inductive Demonic subvalue maps, $S \sqsupseteq S'$ iff for subgame b of α, $\models [(a{:}\alpha_{:b})[]\Upsilon S'](S'(b) \rightarrow S(b))$ and $\models [a{:}\alpha[]\Upsilon S'](S'(\text{end}) \rightarrow S(\text{end}))$.

Remark 5. For every game and winning condition combination, the model predictive subvalue map (Def. 6.5) is a maximally permissive solution (Theorem 7.2). As Def. 6.5 always constructs an optimal subvalue map, it optimally solves the dGL control envelope synthesis problem, achieving completeness for symbolic hybrid games synthesis.

Theorem 7.2 (Maximal inductive subvalue map). *Amongst the inductive Angelic subvalue maps for game* $a{:}\alpha$ *compatible with winning condition* ϕ*, the model predictive Angelic map given by Def. 6.5 is maximal under the ordering of Def. 7.1. That is, for all* S *such that* $a{:}\alpha \models S$*, model predictive Angelic subvalue map* S' *satisfies* $S' \sqsupseteq S$*. Dually, the model predictive Demonic subvalue map (Def. 6.5) is maximal amongst inductive Demonic subvalue maps per the ordering of Def. 7.1.*

8 Synthesis Framework for Inductive Subvalue Map with Efficient Checks

To check whether a given control decision lies within the policy of a subvalue map, Def. 5.3 checks whether the resulting state is within a subvalue (e.g., for $b:(d:\delta \cup g:\gamma)$, the policy checks $\sigma \models S(d)$ and $\sigma \models S(g)$). This section synthesizes inductive subvalue maps where state checks are efficient, for quick responses in real systems. In the maximal solution from the previous section (Def. 6.5), such checking is not efficient because of subvalues that are dGL formulas, for which checking is undecidable [44]. In this section we restrict subvalues to formulas consisting of equations/inequalities of real polynomials along with propositional logic connectives ($\wedge, \vee, \neg, etc.$) but no quantifiers or modalities, henceforth, *propositional real arithmetic* formulas ($\mathcal{P}_{\mathbb{R}}$). For these, checking the truth value at a state takes polynomial time. *Given a* dGL *game and winning condition in* $\mathcal{P}_{\mathbb{R}}$*, this section synthesizes inductive subvalue maps where subvalues are in* $\mathcal{P}_{\mathbb{R}}$.

Synthesizing such a map is challenging because the subvalues must remain *mutually compatible*, indicating only the winning regions achievable while following the other subvalues. That is, they must follow the inductive subvalue map conditions Def. 6.2, where there are circular dependencies between subvalues. Consider, for example, the case of Demonic loop $a:(g:\gamma)^\times$. Subvalue $S(a)$ must imply $S(g)$, but $S(g)$ depends on $S(a)$ via condition $g:\gamma \models S(\text{end} \mapsto S(a))$. In particular, starting from the optimal subvalue map construction (Def. 6.5) and changing individual dGL formulas to under-approximations in $\mathcal{P}_{\mathbb{R}}$ would not alone work because cyclically interdependent soundness conditions would need to be reestablished. (1) We address the challenge of interdependent conditions by isolating their effect with a hybrid games analog of loop invariants. (2) To soundly lower formulas to $\mathcal{P}_{\mathbb{R}}$ while maintaining permissiveness, we use a *predicate transformer* based approach that given a subgame and winning condition in $\mathcal{P}_{\mathbb{R}}$ computes subvalues in $\mathcal{P}_{\mathbb{R}}$. This lets us present a backwards symbolic execution [7, 21] based *algorithmic framework* to compute inductive subvalue maps with subvalues in $\mathcal{P}_{\mathbb{R}}$.

Remark 6. Weakest precondition calculi [21] generate weak *preconditions*. To reason about loops, they generally *check* loop invariants, and rely on external loop invariant generation. Analogously, our framework generates permissive *subvalues*. To reason about loops, it *provides checks for invariants of Angel and Demon loop subvalue maps*. It is *parametric* in the generation of such invariants.

We define predicate transformers over subgames such that *given a postcondition in* $\mathcal{P}_{\mathbb{R}}$, they *output a subvalue in* $\mathcal{P}_{\mathbb{R}}$ to recursively produce an entire subvalue map with subvalues in $\mathcal{P}_{\mathbb{R}}$. The transformers for the loop-free fragment of dGL are derived from existing work [30], which we discuss later in this section. For loops, our framework uses the approach of heuristically generating *invariant candidates* and checking their correctness, which has worked well for hybrid systems [46, 53]. We characterize the necessary and sufficient conditions for a sound invariant (for Angel loops, $\models \mathsf{Inv} \rightarrow \langle a:\alpha^{\exists}_{\downarrow} S\rangle\phi$, for Demon loops, $\models \mathsf{Inv} \rightarrow S(g) \wedge \phi$). The framework generates invariant candidate Inv (details in (Section 8.1)), computes the subvalue map of the loop body, and then checks whether the invariant and loop body map are compatible[7]. Proposed invariants must be in $\mathcal{P}_{\mathbb{R}}$.

Algorithm 1 puts together the predicate transformers to define a recursive solving function $\langle\rangle\mathsf{map}(a:\alpha, \phi)$ that computes an inductive Angelic subvalue map with winning regions in $\mathcal{P}_{\mathbb{R}}$ for game $a:\alpha$ that is compatible with winning condition $\phi \in \mathcal{P}_{\mathbb{R}}$. Operator $\uplus$ computes the disjoint union of subvalue maps. $S \setminus \text{end}$ indicates the subvalue map S with the mapping for end removed. Such deletion is sometimes necessary to ensure that two maps to which we apply the disjoint union operator are genuinely disjoint. The algorithm is parametrized on *invariant candidate* generation function genInv. $[]\mathsf{map}$ similarly computes an inductive Demonic subvalue map (Appendix G.8).

[7] Checking Angel loop invariants ($\models \mathsf{Inv} \rightarrow \langle a:\alpha^{\exists}_{\downarrow} S\rangle\phi$) can be difficult to automate using traditional variant generation techniques. We use refinement based proofs in practice (Appendix E.5).

Algorithm 1 Inductive subvalue map synthesis framework

function $\Diamond\mathsf{map}(a{:}\alpha, \phi)$ ▷ Parametric Input: genInv
  $S_e \leftarrow \{\mathsf{end} \mapsto \phi\}$;
  if $\alpha \in \{x := e, x := *, x := \otimes, ?Q, !Q, \{x' = f(x)\&Q\}, \{x' = f(x)\&Q\}^d\}$ **then**
    return $\{a \mapsto \mathsf{simpl}(\langle\alpha\rangle\phi)\} \uplus S_e$
  else if $\alpha = a{:}(g{:}\gamma \cup d{:}\delta)$ **then**
    $S_1 \leftarrow \Diamond\mathsf{map}(g{:}\gamma, \phi)$; $S_2 \leftarrow \Diamond\mathsf{map}(d{:}\delta, \phi)$; **return** $S_1 \uplus (S_2 \setminus \mathsf{end}) \uplus \{a \mapsto S_1(g) \vee S_2(d)\}$
  else if $\alpha = a{:}(g{:}\gamma \cap d{:}\delta)$ **then**
    $S_1 \leftarrow \Diamond\mathsf{map}(g{:}\gamma, \phi)$; $S_2 \leftarrow \Diamond\mathsf{map}(d{:}\delta, \phi)$; **return** $S_1 \uplus (S_2 \setminus \mathsf{end}) \uplus \{a \mapsto S_1(g) \wedge S_2(d)\}$
  else if $\alpha = a{:}(g{:}\gamma; d{:}\delta)$ **then**
    $S_1 \leftarrow \Diamond\mathsf{map}(d{:}\delta, \phi)$; $S_2 \leftarrow \Diamond\mathsf{map}(g{:}\gamma, S_1(d))$; **return** $S_1 \uplus (S_2 \setminus \mathsf{end}) \uplus \{a \mapsto S_2(g)\}$
  else if $\alpha = a{:}(g{:}\gamma)^*$ **then**
    while true **do**
      $\mathsf{Inv} \leftarrow \mathsf{genInv}(\langle(g{:}\gamma)^*\rangle\phi)$; $S \leftarrow \Diamond\mathsf{map}(g{:}\gamma, \mathsf{Inv} \vee \phi)$
      if $\models \mathsf{Inv} \rightarrow \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\phi$ **then return** $(S \setminus \mathsf{end}) \uplus \{a \mapsto \mathsf{Inv}\} \uplus S_e$
  else $\alpha = a{:}(g{:}\gamma)^\times$
    while true **do**
      $\mathsf{Inv} \leftarrow \mathsf{genInv}(\langle(g{:}\gamma)^\times\rangle\phi)$; $S \leftarrow \Diamond\mathsf{map}(g{:}\gamma, \mathsf{Inv})$
      if $\models \mathsf{Inv} \rightarrow S(g) \wedge \phi$ **then return** $(S \setminus \mathsf{end}) \uplus \{a \mapsto \mathsf{Inv}\} \uplus S_e$

$\Diamond$map recursively computes the subvalue for each subgame. The terminal winning subregion $S(\mathsf{end})$ is always set to the target winning condition ϕ. In the atomic cases, there is only one subgame, and thus only one additional subvalue to compute. Here, the predicate transformer sets the subvalue per dGL winning region semantics (e.g., for $a{:}x := e$, the winning subregion is $\langle x := e\rangle\phi$). But $\langle\alpha\rangle\phi$ is a modal dGL formula which needs to be further simplified to a $\mathcal{P}_{\mathbb{R}}$ formula (in this case, to $\phi(x \mapsto e)$, i.e., the formula $\phi \in \mathcal{P}_{\mathbb{R}}$ with every occurrence of variable x substituted by e). This simplification is written as $\mathsf{simpl}(\langle\alpha\rangle\phi)$. Existing work [30] shows how to implement simpl[8]. While $\mathsf{simpl}(\langle\alpha\rangle\phi)$ usually computes an exact expression in $\mathcal{P}_{\mathbb{R}}$, this is sometimes not possible for subgames with complicated ODEs. In such situations, a conservative, sound approximation is made such that $\models \mathsf{simpl}(\langle\alpha\rangle\phi) \rightarrow \langle\alpha\rangle\phi$, i.e. the simplified expression is true only in the states where the original expression was true. Theorem 8.1 shows that the solving framework is robust to such conservative approximations, producing inductive subvalue maps despite them. In the compositional cases, e.g., $a{:}(g{:}\gamma \cup d{:}\delta)$, $\Diamond$map first computes the subvalues of the composing subgames (in this case S_1 for $g{:}\gamma$ and S_2 for $d{:}\delta$), and then the subvalue for the outer subgame (in the case, $S(g) \vee S(d)$), following the rules of simpl [30]. For the loop cases, $\Diamond$map generates invariants until it finds one that passes the check.

Fig. 1 shows an example of computing a subvalue map using $\Diamond$map. As before, label i denotes the subgame with winning subregion ϕ_i. At the top level, subgame 1 is a Demon loop. For this structure, after setting ϕ_{end}, in second step **Guess** ϕ_1, $\Diamond$map guesses invariant Inv using a *refinement* heuristic (discussed in Section 8.1), computing answer $x > 0 \wedge (v \geq 0 \vee a > 0)$. This invariant is used to recursively compute an inductive Angelic subvalue map for the loop body consisting of subgame 9. A final **Check** step ensures that $\models \psi \rightarrow \phi_9 \wedge x > 0$, i.e., the invariant is compatible with the inner subvalue map so Angel won't get stuck regardless of whether Demon runs the loop or exits.

[8]The function $\mathsf{reduce}(\langle\alpha\rangle\phi, A)$ [30] produces a $\mathcal{P}_{\mathbb{R}}$ formula that is equivalent to $\langle\alpha\rangle\phi$ under assumption formula A. $\mathsf{simpl}(\langle\alpha\rangle\phi)$ is $\mathsf{reduce}(\langle\alpha\rangle\phi, \top)$.

Theorem 8.1 (Sound solving). *$S := \langle\rangle\mathsf{map}(a\alpha, \phi)$ is an inductive Angelic subvalue map for game $a\alpha$ ($a\alpha \models S$) with all subvalues in $\mathcal{P}_{\mathbb{R}}$, compatible with Angel winning condition $\phi \in \mathcal{P}_{\mathbb{R}}$. Dually, $S := []\mathsf{map}(a\alpha, \phi)$ is an inductive Demonic subvalue map with all subvalues in $\mathcal{P}_{\mathbb{R}}$, compatible with Demon winning condition ϕ.*

8.1 Invariant Generation

The subvalue map synthesis framework of Algorithm 1 provides the conditions to check if a candidate formula is a sound invariant. Any combination of invariant generation heuristics [30, 46, 53] can be used to generate candidate invariants. To solve complicated games where no techniques apply, we propose a *game rewriting* framework. This framework uses a library of case-based syntactic transformations to rewrite the loop game α^* or $\alpha^\times$ as a new simpler game β that emulates aspects of its behavior. Then, if β is loop-free, $\mathsf{simpl}(\langle\beta\rangle\phi)$ becomes an invariant guess, otherwise other invariant generation heuristics are tried on β. Checks in Algorithm 1, Line 14 and Line 18 recover soundness regardless. For example, the step **Guess** ϕ_1 in Fig. 1 shows a rewrite when computing an invariant of the outermost loop (ϕ_1). The rewritten game (on the second line) emulates inner loop $(v := v + a)^*$ by letting Angel decide in advance how many loop iterations to run and assign her decision freely to new variable n. Then, setting $v := v + an$ conservatively estimates the effect of running the loop $\lceil n \rceil$ times (when n is fractional, v ends up smaller, and thus conservatively less favorable to Angel than after $\lceil n \rceil$ loop iterations). The rewrite also eliminates the outer loop using a generalized form of one-shot refinement [30] where running the ODE emulates the worst case behavior of multiple loop iterations. In general, different rewrite rules are tried heuristically based on the shape of a problem until an answer is found. New rewrite rules can be added to the library as needed. Appendix E.4 shows the rewrite rules that solve the evaluation problems (Section 9).

8.2 Recovering Solution Optimality

Algorithm 1 independently constructs strategies for Angel and Demon. For the same game $a\alpha$, let $S_{\langle\rangle}$ be an Angelic subvalue map for winning condition ϕ and $S_{[]}$ be a Demonic subvalue map for the opposite objective $\neg\phi$. Cross-checking these subvalue maps can recover whether they are optimal, capturing all possible ways for their players to win. If for each subgame b, $S_{[]}(b) \vee S_{\langle\rangle}(b)$ is valid, then intuitively, $S_{\langle\rangle}(b)$ and $S_{[]}(b)$ together cover every possible state, identifying it either as one where Angel has a winning strategy or where Demon does. Further weakening $S_{\langle\rangle}(b)$ would violate consistency (i.e., it is never possible for both Angel and Demon to win)[9]. So $S_{\langle\rangle}(b)$ must be a maximally permissive, optimal Angel winning region, and $S_{\langle\rangle}$, an optimal Angelic subvalue map.

9 Evaluation

To test the generality and flexibility of our approach, we demonstrate it on examples from classic control theory problem classes (event-triggered control, reach-avoid problems, nonlinear control, infinite horizon switching). The existing tool addressing a problem space closest to ours is CESAR [30], which synthesizes symbolic control envelopes as well, but for hybrid *systems*. NYCS [11] is a tool that can synthesize control envelopes for hybrid games (expressed via hybrid automata), but does not solve parametrically for symbolic constraints[10]. We compare performance.

[9] This is analogous to how in α-β pruning [31], if MIN's action results in a position where MAX wins, then that action cannot be part of MIN's strategy.

[10] Our problems have symbolic parameters that can represent any real number, e.g., in event-triggered ETCS (Model 1), A, B and T. We generate an envelope that is parametric in these symbols, equivalent to making uncountably infinitely many calls to NYCS corresponding to every possible parameter value assignment. Such parametricity is crucial in some applications (e.g., when parameters are unknown statically and only identified at runtime [25]).

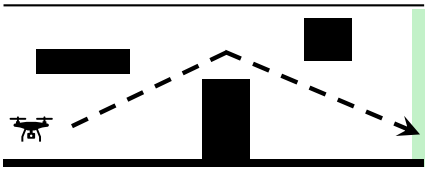
Benchmark	Ours	CESAR	NYCS	Problem Type
Event ETCS	23s	∞	∞	Event-triggered control
Infinite Track	233s	∞	∞	Infinite horizon switching
Surgical Robot [33]	168s	∞	∞	Nonlinear control
Highway Driving [35]	91s	∞	∞	Time-triggered control with adversarial agent
Reach-avoid Robot	21s	∞	∞	Reach-avoid problem

(a) Summary of new examples. Appendix E discusses them.

Benchmark	Ours	CESAR	NYCS
ETCS Train	12s	15s	∞
Sled	64s	64s	∞
Intersection	104s	104s	∞
Curvebot	111s	112s	∞
Parachute	113s	123s	∞
Corridor	40s	40s	∞
Power Station	56s	56s	∞
Coolant	311s	312s	∞

(b) Running time comparison on the CESAR benchmark suite [30]

Tool	Success	Failure
Ours	17	8
NYCS	25	0
CESAR	0	25



(c) Setup and outcome of the quadcopter reach-avoid problem with procedurally generated obstacle placement.

CESAR and NYCS cannot solve the new examples, each of which displays different problem features, demonstrating the greater generality of our approach, which solves them within a few minutes (Fig. 3a). We also evaluate our approach on CESAR's control envelope synthesis benchmark suite [30]. Our implementation's performance is similar to CESAR despite the significantly greater generality (Fig. 3b). To compare to NYCS, we procedurally generate 25 instantiations of a reach-avoid problem. The implementation finds control envelopes for 17 of them, timing out for the rest, while NYCS solves all of them, suggesting potential for future optimizations for this problem class.

The experiments were run on a 32GB RAM M2 MacBook Pro. The implementation uses Pegasus [53], CESAR [30], and additional rewriting heuristics for invariant generation. It uses Mathematica for simplification and quantifier elimination. Our examples favor interesting control challenges over dynamic complexity to avoid the computer algebra bottleneck, which is an avenue for future research. dGL formulas for all the new examples are provided in Appendix E.3.

Event-triggered ETCS (Model 1) is an example of an *event-triggered* control problem, in contrast with the time-triggered control handled by existing tools [30]. In time-triggered control, the controller repeatedly polls with some maximum time latency. In event-triggered control, the controller is instead triggered at *events*, when specific conditions are met. Identifying what these events are that should trigger controller intervention is a part of the synthesis challenge. Our example problem (Model 1) translates the ETCS Train benchmark of the CESAR benchmark suite from the time-triggered to the event-triggered paradigm. It describes a train controller that must stop the train by point e, and has the choice to either accelerate or to brake. Line 3 shows the plant which Angel (emulating the controller) chooses how long to run. The synthesized subvalue map indicates when Angel must stop continuous evolution to enforce braking, characterizing the event at which to trigger the controller. Modeling an event-triggered system as a game is subtle: the

Model 1 Event-triggered ETCS

$$\begin{array}{rll} \text{assumptions} & 1 & A > 0 \land B > 0 \land T > 0 \rightarrow \langle \\ \text{choice} & 2 & \Big(t := 0\,;\ \big((a := A \cup a := -B)\ ; \\ \text{plant} & 3 & \{p' = v, v' = a, t' = 1 \ \&\ e - p > 0\}\big)^*\ ;\ ?t \geq 1\Big)^{\times} \rangle \mathsf{True} \end{array}$$

controller has the ability to choose how long to run the plant, and should not be permitted to win by interrupting infinitely often and inducing Zeno behavior. dGL's flexibility allows a correct encoding using nested loops and time progress assertions.

Surgical Robot shows an example of nonlinear control (Model 2). It expresses the core control challenge in a case study from the literature [33] modeling force feedback in surgical robots. The objective is to dampen the force applied by the surgeon to ensure that a surgical tool stays within bounding planes called *virtual fixtures*. The damping factor g (Line 3) that we synthesize for can be *any* nonnegative real number, implying infinitely many control possibilities.

Model 2 Surgical Robot example based on [33]

$$\begin{array}{rll} \text{assumptions} & 1 & T > 0 \land K > 0 \land n_x^2 + n_y^2 = 1 \rightarrow \langle \\ \text{input} & 2 & \big(f_{xp} := \otimes\,;\ f_{yp} := \otimes\,; \\ \text{damping} & 3 & g := *; ?g \geq 0\,; \\ \text{plant} & 4 & t := 0; \{q_x' = Kf_x g, q_y' = Kf_y g, f_x' = f_{xp}, f_y' = f_{yp}, t' = 1 \ \&\ t \leq T\}^d \\ \text{safe} & 5 & \big)^{\times}\rangle\big((q_x - p_x)n_x + (q_y - p_y)n_y \geq 0\big) \end{array}$$

Highway Driving (Model 6) expresses the core control challenge of a case study [35] where on a highway, a controlled car must follow the car ahead at a safe distance. The car ahead can behave adversarially by braking suddenly. *Reach-avoid Robot* (Model 5) demonstrates an example of envelope synthesis for reach-avoid problems, where an agent must *reach* an objective while *avoiding* unsafe situations. *Infinite Track* (Model 4) is an example of infinite horizon switching. To remain safe for arbitrarily long time, the controlled vehicle must keep switching directions at the right moments, and make *infinitely* many switches. Appendix E.1 discusses these examples and their synthesized control envelopes further, and Appendix E.3 lists the models.

To compare the algorithm to NYCS, we procedurally generate instantiations of *quadcopter*, replacing symbolic parameters with concrete numbers, to create a reach-avoid challenge. Each instantiation describes a scenario like in Fig. 3c, where a quadcopter must reach a target safe area while avoiding obstacles placed randomly in the environment. The quadcopter constantly moves forward and can choose to either move upwards or downwards, but cannot revise its decision, once made, for one second. The objective is to find a control envelope showing from which points the quadcopter has a way to reach the target safe zone while avoiding the obstacles. The implementation finds control envelopes for 17 problems. The other 8 problems time out after 20 minutes. CESAR cannot solve any of these problems because they do not fit in its template. NYCS, with its polytope representation and optimizations for this problem class, solves all of the problems. Appendix E.2 provides further details.

10 Related Work

Subvalue maps represent nondeterministic policies for hybrid games in a way that is compositional at the level of imperative program constructs such as loops and branches. They can be verified/synthesized leveraging program analysis techniques like loop invariants, making it possible to answer hybrid game control envelope synthesis questions that no previous work solves.

- *Program logics* like game logic and differential game logic express games as imperative programs with Angelic and Demonic nondeterminism [42–44]. Their semantics compositionally define when *there exists* a way to win a game, but not when a given policy wins.
- Refinement [13, 40], Hoare logic [36], and constructive proof [14] based approaches soundly construct deterministic policies (i.e., programs with Angelic nondeterminism fully resolved) but do not study *maximal sets of solutions*. They do not have a direct representation of nondeterministic policies from which to derive *sets* of permissible control actions, or a formal comparison of permissiveness of such policies, both of which are crucial parts of the permissive policy synthesis problem that we solve.
- A line of research solves games specified by *hybrid automata* with objectives expressed in logics like LTL [11, 22, 29, 62, 64]. Our approach targets specifications at a higher level of abstraction, where the game has imperative programming constructs like loops and branches. Compositionality is also at the level of these constructs, and the theoretical development required is different (e.g., the subtle Angel loop case in Def. 6.2). This higher level of abstraction permits synthesis to use program analysis techniques like loop invariants and their continuous analog, differential invariants. The latter has the practical advantage of being complete for reasoning about the very general class of dynamics whose solutions are Noetherian functions [48], thus making it possible to solve problems that these previous techniques do not.
- Traditional *game theory* [52, 63] and *reinforcement learning* [57] do have recursive solving techniques and representations of nondeterministic policies (value function) [56, 58]. But these works also operate at a different level of abstraction than our imperative program-like hybrid games. Also, unlike these works, the subvalue maps of this paper are logically formalized, suitable for computerized verification and formally justified synthesis.
- *Compositional game theory* [6, 27] also has a compositional representation of nondeterministic policies but also operates at a different level of abstraction, solving not hybrid games but abstract games with sequential and parallel composition.
- There is also an important distinction between this work and *controller synthesis* [10, 34, 59], for which approaches include the use of games [41, 61] and CEGIS-based [54] guidance from counterexamples [1, 20, 50]. We tackle control *envelope* synthesis, where the goal is to characterize *all* the correct control strategies, requiring an extra universal quantifier compared to controller synthesis. Controller synthesis techniques do not solve control envelope synthesis. For example, dGL control envelope synthesis does not fit the CEGIS quantifier pattern of $\exists\forall$.
- *Safety shields* computed by numerical methods [2, 23, 32] share the objective of characterizing the space of safe control. They can handle dynamical systems that are hard to analyze symbolically, but do not support unbounded state spaces, require a hopeless increase in dimensionality to computing shields parameterized on the model's parameters, and can lose rigorous formal safety guarantees when they discretize continuous systems.
- Compared to safe set and *barrier function* approaches [4, 5, 17, 19, 37, 49, 55], we again solve for the higher level of abstraction of dGL, with program structures such as loops, as well as the adversarial dynamics of games.

- *Differential Floyd-Hoare logic* [28] identifies the relationship between preconditions and runtime monitoring, but deals with hybrid *systems*, not games, which lack the complexity of adversarial dynamics. Additionally it uses preconditions to synthesize *deterministic* policies (proper responses), different from our nondeterministic policy question.
- The *CESAR algorithm* [30] also synthesizes symbolic control envelopes for hybrid *systems*, for a *fixed, time-triggered template*. However, it does not provide a general, compositional representation for nondeterministic policies, and does not deal with the complexities brought by the adversarial dynamics of full dGL.

11 Conclusion and Future Work

We introduce a generalized symbolic control envelope synthesis approach for hybrid games. Such envelopes identify what control action is safe to take when. We represent the envelopes using inductive subvalue maps and provide an algorithmic framework to generate them. Our approach allows the expression and synthesis of a variety of control problems. The approach should generalize for non-dGL Markovian games, where the policy depends only on state and not the history of previous moves, which is an avenue for future research. Another direction for future work is to integrate other tools, e.g., NYCS [12] as invariant generators to improve performance on the problem classes that they solve.

12 Acknowledgments

This work was partially funded by the National Science Foundation (NSF) under Award Number 2220311, and by an Alexander von Humboldt Professorship. We thank Ruben Martins and Stephanie Balzer for helpful feedback on drafts of this paper.

References

[1] Alessandro Abate, Iury Bessa, Lucas C. Cordeiro, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. 2020. Automated formal synthesis of provably safe digital controllers for continuous plants. *Acta Informatica* 57, 1-2 (2020), 223–244. doi:10.1007/s00236-019-00359-1

[2] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu. 2018. Safe reinforcement learning via shielding. *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (2018). Issue 1. doi:10.1609/aaai.v32i1.11797

[3] Rajeev Alur, Salar Moarref, and Ufuk Topcu. 2016. Compositional Synthesis with Parametric Reactive Controllers. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control* (Vienna, Austria) *(HSCC '16)*. Association for Computing Machinery, New York, NY, USA, 215–224. doi:10.1145/2883817.2883842

[4] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. Control Barrier Functions: Theory and Applications. In *17th European Control Conference, ECC 2019, Naples, Italy, June 25-28, 2019*. IEEE, 3420–3431. doi:10.23919/ECC.2019.8796030

[5] Zvi Artstein. 1983. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications* 7, 11 (1983), 1163–1173.

[6] Robert Atkey, Bruno Gavranović, Neil Ghani, Clemens Kupke, Jérémy Ledent, and Fredrik Nordvall Forsberg. 2020. Compositional Game Theory, Compositionally. In *3rd international Applied Category Theory Conference*, Vol. 333. Online, United States, 198 – 214. doi:10.4204/eptcs.333.14

[7] Robert J. Aumann. 1995. Backward induction and common knowledge of rationality. *Games and Economic Behavior* 8, 1 (1995), 6–19. doi:10.1016/S0899-8256(05)80015-6

[8] Richard Bellman. 1957. *Dynamic Programming*. Dover Publications.

[9] Richard E. Bellman and Stuart E Dreyfus. 1962. *Applied Dynamic Programming*. Princeton University Press, Princeton. doi:10.1515/9781400874651

[10] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. 2017. *Formal Methods for Discrete-Time Dynamical Systems*. Springer Cham.

[11] Massimo Benerecetti and Marco Faella. 2017. Automatic Synthesis of Switching Controllers for Linear Hybrid Systems: Reachability Control. *ACM Trans. Embed. Comput. Syst.* 16, 4, Article 104 (May 2017), 27 pages. doi:10.1145/3047500

[12] Massimo Benerecetti, Marco Faella, and Stefano Minopoli. 2012. Reachability games for linear hybrid systems. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control* (Beijing, China)

(HSCC '12). Association for Computing Machinery, New York, NY, USA, 65–74. doi:10.1145/2185632.2185645

[13] Rastislav Bodik, Satish Chandra, Joel Galenson, Doug Kimelman, Nicholas Tung, Shaon Barman, and Casey Rodarmor. 2010. Programming with angelic nondeterminism. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Madrid, Spain) *(POPL '10)*. Association for Computing Machinery, New York, NY, USA, 339–352. doi:10.1145/1706299.1706339

[14] Rose Bohrer and André Platzer. 2020. Constructive Hybrid Games. In *ITCAR* (Paris, France). Springer, Berlin, Heidelberg, 454–473. doi:10.1007/978-3-030-51074-9_26

[15] Janusz A. Brzozowski. 1964. Derivatives of Regular Expressions. *J. ACM* 11, 4 (Oct. 1964), 481–494. doi:10.1145/321239.321249

[16] Mo Chen, Qizhan Tam, Scott C. Livingston, and Marco Pavone. 2020. Signal Temporal Logic Meets Reachability: Connections and Applications. In *Algorithmic Foundations of Robotics XIII*, Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson (Eds.). Springer International Publishing, Cham, 581–601.

[17] Max H. Cohen and Calin Belta. 2020. Approximate Optimal Control for Safety-Critical Systems with Control Barrier Functions. In *2020 59th IEEE Conference on Decision and Control (CDC)*. 2062–2067. doi:10.1109/CDC42340.2020.9303896

[18] Charles R. Cutler and Brian L. Ramaker. 1979. Dynamic matrix control—A computer control algorithm. *IEEE Trans. Automat. Control* 17 (1979), 72. doi:10.1109/JACC.1980.4232009

[19] J. Daafouz, P. Riedinger, and C. Iung. 2002. Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach. *IEEE Trans. Automat. Control* 47, 11 (2002), 1883–1887. doi:10.1109/TAC.2002.804474

[20] Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. 2020. Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems. *2020 59th IEEE Conference on Decision and Control (CDC)* (2020), 1274–1281. doi:10.1109/CDC42340.2020.9304201

[21] Edsger W. Dijkstra. 1975. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM* 18, 8 (Aug. 1975), 453–457. doi:10.1145/360933.360975

[22] Catalin Dima, Mariem Hammami, Youssouf Oualhadj, and Régine Laleau. 2024. Deciding the Synthesis Problem for Hybrid Games Through Bisimulation. In *Formal Methods and Software Engineering*, Kazuhiro Ogata, Dominique Mery, Meng Sun, and Shaoying Liu (Eds.). Springer Nature Singapore, Singapore, 181–198. doi:10.1007/978-981-96-0617-7_11

[23] J. Fisac, A. Akametalu, M. Zeilinger, S. Kaynama, J. Gillula, and C. Tomlin. 2019. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Trans. Automat. Control* 64 (2019), 2737–2752. Issue 7. doi:10.1109/tac.2018.2876389

[24] Nathan Fulton and André Platzer. 2018. Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, February 2-7, 2018, New Orleans, Louisiana, USA.*, Sheila McIlraith and Kilian Weinberger (Eds.). AAAI Press, 6485–6492. doi:10.1609/aaai.v32i1.12107

[25] Nathan Fulton and André Platzer. 2019. Verifiably Safe Off-Model Reinforcement Learning. In *TACAS*, Tomáš Vojnar and Lijun Zhang (Eds.). Springer International Publishing, Cham, 413–430. doi:10.1007/978-3-030-17462-0_28

[26] Carlos E. García, David M. Prett, and Manfred Morari. 1989. Model predictive control: Theory and practice—A survey. *Automatica* 25, 3 (1989), 335–348. doi:10.1016/0005-1098(89)90002-2

[27] Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. 2018. Compositional Game Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science* (Oxford, United Kingdom) *(LICS '18)*. Association for Computing Machinery, New York, NY, USA, 472–481. doi:10.1145/3209108.3209165

[28] Ichiro Hasuo, Clovis Eberhart, James Haydon, Jérémy Dubut, Rose Bohrer, Tsutomu Kobayashi, Sasinee Pruekprasert, Xiao-Yi Zhang, Erik André Pallas, Akihisa Yamada, Kohei Suenaga, Fuyuki Ishikawa, Kenji Kamijo, Yoshiyuki Shinya, and Takamasa Suetomi. 2023. Goal-Aware RSS for Complex Scenarios via Program Logic. *IEEE Transactions on Intelligent Vehicles* 8, 4 (2023), 3040–3072. doi:10.1109/TIV.2022.3169762

[29] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. 1999. Rectangular Hybrid Games. In *CONCUR'99 Concurrency Theory*, Jos C. M. Baeten and Sjouke Mauw (Eds.). Springer, Berlin, Heidelberg, 320–335. doi:10.1007/3-540-48320-9_23

[30] Aditi Kabra, Jonathan Laurent, Stefan Mitsch, and André Platzer. 2024. CESAR: Control Envelope Synthesis via Angelic Refinements. In *Tools and Algorithms for the Construction and Analysis of Systems (LNCS, Vol. 14570)*, Bernd Finkbeiner and Laura Kovács (Eds.). Springer, 144–164. doi:10.1007/978-3-031-57246-3_9

[31] Donald E. Knuth and Ronald W. Moore. 1975. An analysis of alpha-beta pruning. *Artificial Intelligence* 6, 4 (1975), 293–326. doi:10.1016/0004-3702(75)90019-3

[32] Mykel J Kochenderfer, Jessica E Holland, and James P Chryssanthacopoulos. 2012. Next generation airborne collision avoidance system. *Lincoln Laboratory Journal* 19, 1 (2012), 17–33.

[33] Yanni Kouskoulas, David W. Renshaw, André Platzer, and Peter Kazanzides. 2013. Certifying the Safe Design of a Virtual Fixture Control Algorithm for a Surgical Robot. In *Hybrid Systems: Computation and Control, HSCC, Philadelphia, PA, USA*, Calin Belta and Franjo Ivancic (Eds.). ACM, 263–272. doi:10.1145/2461328.2461369

[34] Siyuan Liu, Ashutosh Trivedi, Xiang Yin, and Majid Zamani. 2022. Secure-by-construction synthesis of cyber-physical systems. *Annual Reviews in Control* 53 (2022), 30–50. doi:10.1016/j.arcontrol.2022.03.004

[35] Sarah M. Loos, André Platzer, and Ligia Nistor. 2011. Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified. In *FM (LNCS, Vol. 6664)*, Michael Butler and Wolfram Schulte (Eds.). Springer, 42–56. doi:10.1007/978-3-642-21437-0_6

[36] Konstantinos Mamouras. 2015. Synthesis of Strategies and the Hoare Logic of Angelic Nondeterminism. In *Foundations of Software Science and Computation Structures*, Andrew Pitts (Ed.). Springer, Berlin, Heidelberg, 25–40. doi:10.1007/978-3-662-46678-0_2

[37] Ian M. Mitchell and Jeremy A. Templeton. 2005. A Toolbox of Hamilton-Jacobi Solvers for Analysis of Nondeterministic Continuous and Hybrid Systems. In *Hybrid Systems: Computation and Control*, Manfred Morari and Lothar Thiele (Eds.). Springer, Berlin, Heidelberg, 480–494. doi:10.1007/978-3-540-31954-2_31

[38] Stefan Mitsch and André Platzer. 2016. ModelPlex: verified runtime validation of verified cyber-physical system models. *Formal Methods Syst. Des.* 49, 1-2 (2016), 33–74. doi:10.1007/s10703-016-0241-z

[39] Thomas Moor and Jennifer M. Davoren. 2001. Robust Controller Synthesis for Hybrid Systems Using Modal Logic. In *HSCC (LNCS, Vol. 2034)*, Maria Domenica Di Benedetto and Alberto L. Sangiovanni-Vincentelli (Eds.). Springer, 433–446. doi:10.1007/3-540-45351-2_35

[40] Carroll Morgan. 1994. *Programming from specifications (2nd ed.)*. Prentice Hall International (UK) Ltd., GBR.

[41] A. Nerode and A. Yakhnis. 1992. Modelling hybrid systems as games. In *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*. 2947–2952 vol.3. doi:10.1109/CDC.1992.371272

[42] Rohit Parikh. 1983. Propositional game logic. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*. 195–200. doi:10.1109/SFCS.1983.47

[43] Marc Pauly. 2002. A Modal Logic for Coalitional Power in Games. *Journal of Logic and Computation* 12, 1 (02 2002), 149–166. doi:10.1093/logcom/12.1.149

[44] André Platzer. 2015. Differential Game Logic. *ACM Trans. Comput. Log.* 17, 1 (2015), 1:1–1:51. doi:10.1145/2817824

[45] André Platzer. 2018. *Logical Foundations of Cyber-Physical Systems*. Springer, Cham. doi:10.1007/978-3-319-63588-0

[46] André Platzer and Edmund M. Clarke. 2009. Computing Differential Invariants of Hybrid Systems as Fixedpoints. *Form. Methods Syst. Des.* 35, 1 (2009), 98–120. doi:10.1007/s10703-009-0079-8

[47] André Platzer and Jan-David Quesel. 2009. European Train Control System: A Case Study in Formal Verification. In *Formal Methods and Software Engineering, ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings*. 246–265. doi:10.1007/978-3-642-10373-5_13

[48] André Platzer and Yong Kiam Tan. 2020. Differential equation invariance axiomatization. *Journal of the ACM (JACM)* 67, 1 (2020), 1–66.

[49] Stephen Prajna and Ali Jadbabaie. 2004. Safety Verification of Hybrid Systems Using Barrier Certificates. In *Hybrid Systems: Computation and Control*, Rajeev Alur and George J. Pappas (Eds.). Springer, Berlin, Heidelberg, 477–492. doi:10.1007/978-3-540-24743-2_32

[50] Hadi Ravanbakhsh and Sriram Sankaranarayanan. 2016. Robust controller synthesis of switched systems using counterexample guided framework. In *2016 International Conference on Embedded Software, EMSOFT 2016, Pittsburgh, Pennsylvania, USA, October 1-7, 2016*. 8:1–8:10. doi:10.1145/2968478.2968485

[51] Jacques Richalet, André Rault, JL Testud, and J Papon. 1978. Model predictive heuristic control. *Automatica* 14, 5 (1978), 413–428. doi:10.1016/0005-1098(78)90001-8

[52] Claude E. Shannon. 1950. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41, 314 (1950), 256–275. doi:10.1080/14786445008521796

[53] Andrew Sogokon, Stefan Mitsch, Yong Kiam Tan, Katherine Cordwell, and André Platzer. 2022. Pegasus: Sound Continuous Invariant Generation. *Form. Methods Syst. Des.* 58, 1 (2022), 5–41. doi:10.1007/s10703-020-00355-z Special issue for selected papers from FM'19.

[54] Armando Solar-Lezama. 2013. Program sketching. *STTT* 15, 5-6 (2013), 475–495. doi:10.1007/s10009-012-0249-7

[55] Eduardo D. Sontag. 1983. A Lyapunov-Like Characterization of Asymptotic Controllability. *SIAM Journal on Control and Optimization* 21, 3 (1983), 462–471. doi:10.1137/0321028

[56] Richard S. Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine Learning* 3 (1988), 9–44. doi:10.1007/BF00115009

[57] R. S. Sutton and A. Barto. 1998. Reinforcement learning: an introduction. *IEEE Transactions on Neural Networks* 9 (1998), 1054–1054. Issue 5. doi:10.1109/tnn.1998.712192

[58] C. Szepesvári and M. Littman. 1999. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural Computation* 11 (1999), 2017–2060. Issue 8. doi:10.1162/089976699300016070

[59] Paulo Tabuada. 2009. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, Berlin. doi:10.1007/978-1-4419-0224-5

[60] Alfred Tarski. 1998. A Decision Method for Elementary Algebra and Geometry. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, Bob F. Caviness and Jeremy R. Johnson (Eds.). Springer, Vienna, 24–84. doi:10.1007/978-3-7091-9459-1_3

[61] Claire J. Tomlin, John Lygeros, and Shankar Sastry. 2000. A Game Theoretic Approach to Controller Design for Hybrid Systems. *Proc. IEEE* 88, 7 (2000), 949–970. doi:10.1109/5.871303

[62] Alan C. van Hulst, Michel A. Reniers, and Wan J. Fokkink. 2015. Maximally Permissive Controlled System Synthesis for Modal Logic. In *SOFSEM 2015: Theory and Practice of Computer Science*, Giuseppe F. Italiano, Tiziana Margaria-Steffen, Jaroslav Pokorný, Jean-Jacques Quisquater, and Roger Wattenhofer (Eds.). Springer, Berlin, Heidelberg, 230–241. doi:10.1007/978-3-662-46078-8_19

[63] John von Neumann. 1928. Zur Theorie der Gesellschaftsspiele. *Math. Ann.* 100 (1928), 295–320. doi:10.1007/BF01448847

[64] Shufang Zhu and Giuseppe De Giacomo. 2022. Synthesis of Maximally Permissive Strategies for LTLf Specifications. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2783–2789. doi:10.24963/ijcai.2022/386

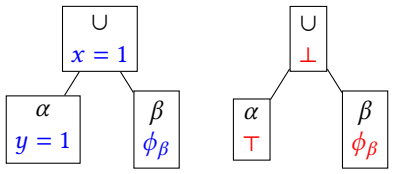


Fig. 4. Inductive subvalue maps S and S'. Locally at subgame α, S' seems more permissive, but global analysis reveals S is more permissive.

A Naïve Ordering Counterexample

For game $a{:}\alpha$, let S' set the winning region for the overall game to be empty ($S'(a) = \bot$), meaning that Angel is never allowed to play at all. Now the winning subregions for later subgames are irrelevant because Angel cannot reach them, but S' nevertheless sets them to the theoretical optimal (for every subgame $b \neq a$, $S(b) \mapsto \langle a{:}\alpha_{b:}\rangle\phi$). For *any* other inductive subvalue map S, a good ordering should indicate that $S \sqsupseteq S'$ since S cannot possibly be less permissive, but the proposed naïve ordering does not. When S sets the winning subregion for any of the later subgames $b \neq a$ to anything less than the optimal solution, the check $\models S'(b) \rightarrow S(b)$ fails.

For a concrete visualization, consider the two inductive subvalue maps shown in Fig. 4: S on the left and S' on the right. Because S' immediately blocks off *all* strategies at the root, a good ordering should indicate $S \sqsupseteq S'$, but the proposed naïve ordering does not because at the subgame α, S has a strictly stronger formula ($y = 1 \rightarrow \top$ is false). The problem in the naïve, point wise definition (Def. F.1) is it does not recognize that the $\top$ at subgame α is vacuous: S' permits no traces to reach subgame α in the first place, so the permissiveness of $\top$ is unimpressive.

B Inductive Subvalue Map Loop Condition Discussion

We discuss the details of the definition of an inductive subvalue maps (Def. 6.2) for the case $a{:}(g{:}\gamma)^* \models S$, and correspondingly the synthesis framework $\Diamond$map (Algorithm 1) for the case of $\Diamond\mathsf{map}(a{:}(g{:}\gamma)^*, \phi)$.

B.1 Counterexample for Projection Condition Weakening

Def. 6.2 and Algorithm 1 for the case $(g{:}\gamma)^*$ use the check $\models S(a) \rightarrow \langle a{:}\alpha \exists\downarrow S\rangle\phi$, i.e., winning subregion $S(a)$ should ensure that there is a *finite* strategy to exit while staying within the control envelope. The subvalue map should ensure that Angel does not get stuck in a state where her only option is playing the loop forever. It might seem that the more liberal, simpler condition $\models S(a) \rightarrow \langle\alpha\rangle\phi$ also ensures Angel has a finite exit strategy, but this condition is subtly incorrect. Angel can get stuck in a state where there is theoretically a way to exit the loop but this requires going outside the control envelope. Eq. (5) shows a counterexample.

$$\begin{aligned} &\langle a{:}(b{:}(c{:}x := x + 1 \cup d{:}x := x - 1))*\rangle x \geq 0 \\ &S := \{a \mapsto \top, b \mapsto \top, c \mapsto \bot, d \mapsto \top\} \end{aligned} \tag{5}$$

For this game, if Def. 6.2 were modified such that $\langle a{:}\alpha \exists\downarrow S\rangle\phi$ is replaced by $\langle\alpha\rangle\phi$, the subvalue map S would be inductive. However, it would also let Angel get stuck iterating the loop forever. Consider starting state $\{x \mapsto -1\}$. This state belongs to the winning subregion $S(a)$ of the overall game, so Angel should be able to win. And indeed she can win the game by running one iteration of the loop where she increments x. The new state would be $\{x \mapsto 0\}$, so Angel would exit the loop and win since winning condition $x \geq 0$ holds. However, if Angel is forced to actually respect the subvalue

map (modeled by projection of S onto the game as shown in Eq. (6)), she cannot win.

$$\langle ?\top\,;\,(?\top\,;\,((?\bot\,;\,x := x+1) \cup (?\top\,;\,x := x-1)))^*\,;\,?x \geq 0\rangle x \geq 0 \tag{6}$$

Starting at $\{x \mapsto -1\}$, Angel has to repeat the loop since she has not reached the winning region $x \geq 0$. But inside the loop, the subvalue map says she can never choose to increment x (subgame c). So she is forced to decrement x, reaching new state $\{x \mapsto -2\}$. In principle, if she were allowed to ignore the subvalue map, she could still exit with finite iterations by incrementing x twice. But with subgame c blocked off, she is only ever allowed to decrement x and is stuck in the loop forever. To prevent such subvalue maps from counting as inductive, the check $\models S(a) \rightarrow \langle a:\alpha \overset{\exists}{\downarrow} S\rangle\phi$ is necessary.

B.2 Counterexample for Recursive Condition Weakening

Def. 6.2 and Algorithm 1 for the case $(g:\gamma)^*$ also use the check $g:\gamma \models S$, i.e., the subvalue map for the loop body ensures that after playing the loop body, either there is still a finite exit strategy or Angel can already exit the loop. Eq. (7) shows a counterexample where using $g:\gamma \models S$ instead is not sound, since the inner subvalue map permit reaching a state without a finite exit strategy.

$$\begin{aligned} &\langle a:(b:(c:x := x+1 \cup d:x := x-1))*\rangle x \geq 0 \\ &\quad S := \{a \mapsto x \geq -1, b \mapsto x \geq -1, c \mapsto x \geq -1, d \mapsto \top\} \end{aligned} \tag{7}$$

Starting at $\{x \mapsto -1\}$, it is possible for Angel to choose subgame d and decrement x. The new state is $\{x \mapsto -2\}$, from which there is now no finite exit strategy because to increment x, per the guard of subgame c, x must be at least -1. Angel is now stuck decrementing x forever.

C Example of Subvalue Map Shortfall for Runtime Monitoring

We show how for a (non-inductive) subvalue map, the property $\models S(a) \rightarrow \langle a:\alpha \not\downarrow S\phi\rangle\phi$ does not always hold. Consider the subvalue map of Fig. 1, but with ϕ_6 set to the empty subregion $\bot$ instead of $x > 0 \land (v \geq 0 \lor a > 0)$. As before, let the overall game be $a:\alpha$, and subgame labels be the numbers indexing the winning subregions. The new map is still a subvalue map since $\models \bot \rightarrow \langle a:\alpha_{5:}\rangle x > 0$. The untrusted controller monitoring correctness property is Eq. (3) with ϕ_6 replaced by $\bot$. The result has shape

$$x > 0 \land (v \geq 0 \lor a > 0) \rightarrow \left\langle \left(\left(?(\bot \lor (x > 0 \land v \geq 1))\,;\cdots \cap \cdots \right);\cdots \right)^\times \right\rangle x > 0.$$

This formula is not valid. Consider the starting state $\{x \mapsto 1, v \mapsto 0\}$. Demon first chooses to run the loop, then Angel must pass the test $x > 0 \land v \geq 1$, which fails. The *inductive* subvalue map prevents this situation by requiring that ϕ_8 implies $\phi_6 \lor \phi_5$, thus disqualifying this counterexample where Angel gets stuck.

D Axiomatization of dGL

Fig. 5 summarizes the axioms and proof rules of dGL. A full explanation is in [45]. Note that the dual operator α^d switches the role of players in α. For example, $(\alpha \cup \beta)^d$ is the same as $\alpha^d \cap \beta^d$. The semantics [45] of dGL is as follows.

*Definition D.1 (*dGL *semantics).* The *semantics of a* dGL *formula* ϕ is the subset $[[\phi]] \subseteq \mathcal{S}$ of states in which ϕ is true. It is defined inductively as follows

(1) $[[p(\theta_1,\ldots,\theta_k)]] = \{\omega \in \mathcal{S} : (\omega[[\theta_1]],\ldots,\omega[[\theta_k]]) \in (p)\}$
(2) $[[\theta_1 \sim \theta_2]] = \{\omega \in \mathcal{S} : \omega[[\theta_1]] \sim \omega[[\theta_2]]\}$ where $\sim \in \{<, \leq, =, \geq, >\}$
(3) $[[\neg\phi]] = ([[\phi]])^\complement$
(4) $[[\phi \land \psi]] = [[\phi]] \cap [[\psi]]$
(5) $[[\exists x\, \phi]] = \{\omega \in \mathcal{S} : \omega^r_x \in [[\phi]] \text{ for some } r \in \mathbb{R}\}$
(6) $[[\langle\alpha\rangle\,\phi]] = \varsigma_\alpha([[\phi]])$

(7) $[[[\alpha]\,\phi]] = \delta_\alpha([[\phi]])$

A dGL formula ϕ is *valid*, written $\vDash \phi$, iff it is true in all states, i.e. $[[\phi]] = \mathcal{S}$.

Definition D.2 (Semantics of hybrid games). The *semantics of a hybrid game* α is a function $\varsigma_\alpha(\cdot)$ that, for each set of Angel's winning states $X \subseteq \mathcal{S}$, gives the *winning region*, i.e. the set of states $\varsigma_\alpha(X)$ from which Angel has a winning strategy to achieve X in α (whatever strategy Demon chooses). It is defined inductively as follows

(1) $\varsigma_{x:=\theta}(X) = \{\omega \in \mathcal{S} : \omega_x^{\omega[[\theta]]} \in X\}$
(2) $\varsigma_{x'=f(x)\,\&\,Q}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X \text{ for some } r \in \mathbb{R}_{\geq 0} \text{ and (differentiable) } \varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [[Q]]$ and $\frac{d\varphi(t)(x)}{dt}(\zeta) = \varphi(\zeta)[[f(x)]]$ for all $0 \leq \zeta \leq r\}$
(3) $\varsigma_{?Q}(X) = [[Q]] \cap X$
(4) $\varsigma_{\alpha\cup\beta}(X) = \varsigma_\alpha(X) \cup \varsigma_\beta(X)$
(5) $\varsigma_{\alpha;\beta}(X) = \varsigma_\alpha(\varsigma_\beta(X))$
(6) $\varsigma_{\alpha^*}(X) = \bigcap\{Z \subseteq \mathcal{S} : X \cup \varsigma_\alpha(Z) \subseteq Z\}$
(7) $\varsigma_{\alpha^d}(X) = (\varsigma_\alpha(X^{\complement}))^{\complement}$

The *winning region* of Demon, i.e. the set of states $\delta_\alpha(X)$ from which Demon has a winning strategy to achieve X in α (whatever strategy Angel chooses) is defined inductively as follows

(1) $\delta_{x:=\theta}(X) = \{\omega \in \mathcal{S} : \omega_x^{\omega[[\theta]]} \in X\}$
(2) $\delta_{x'=f(x)\,\&\,Q}(X) = \{\varphi(0) \in \mathcal{S} : \varphi(r) \in X \text{ for all } r \in \mathbb{R}_{\geq 0} \text{ and (differentiable) } \varphi : [0,r] \to \mathcal{S}$ such that $\varphi(\zeta) \in [[Q]]$ and $\frac{d\varphi(t)(x)}{dt}(\zeta) = \varphi(\zeta)[[\theta]]$ for all $0 \leq \zeta \leq r\}$
(3) $\delta_{?Q}(X) = ([[Q]])^{\complement} \cup X$
(4) $\delta_{\alpha\cup\beta}(X) = \delta_\alpha(X) \cap \delta_\beta(X)$
(5) $\delta_{\alpha;\beta}(X) = \delta_\alpha(\delta_\beta(X))$
(6) $\delta_{\alpha^*}(X) = \bigcup\{Z \subseteq \mathcal{S} : Z \subseteq X \cap \delta_\alpha(Z)\}$
(7) $\delta_{\alpha^d}(X) = (\delta_\alpha(X^{\complement}))^{\complement}$

E Benchmarks

This appendix discusses the benchmarks, provides their full listings, and discusses the rewrite heuristics used to solve them.

E.1 New Examples Discussion

We discuss the examples representing diverse control challenges introduced in this paper to demonstrate the flexibility of our dGL control envelope synthesis approach.

Event-triggered ETCS (Model 1) models a train from a case study modeling the European Train Control System (ETCS) [47], but with a key modification: the train is modeled event triggered instead of time triggered. The control envelope computed by our implementation finds the winning subregion for the overall loop to be $p < e \land (2Ae + v^2 < 2Ap \lor v \leq 0) \land (A \leq 0 \lor v > 0) \lor p < e \land (2(-B)e + v^2 < 2(-B)p \lor v \leq 0) \land (-B \leq 0 \lor v > 0)$. After simplification, the key term ends up being $e - p + v^2/2B > 0$, i.e., there is still enough time for the train to come to a stop before the end of motion authority e if it starts braking now. Hence, the control envelope has a strategy to keep the train safe whenever it is possible for the train to be safe by braking.

Surgical Robot (Model 2) is based on a case study from the literature [33] modeling force feedback in surgical robots. The control envelope computed by our implementation finds the winning subregion for the overall loop to be $(q_x - p_x)n_x + (q_y - p_y)n_y >= 0$, i.e., so long as the robot is not already outside the fixture, the subvalue map has a strategy for safe control.

$$([\cdot]) \quad [\alpha]\, P \leftrightarrow \neg\langle\alpha\rangle\, \neg P$$

$$(\langle:=\rangle) \quad \langle x := e\rangle\, p(x) \leftrightarrow p(e)$$

$$(\langle'\rangle) \quad \langle x' = f(x)\rangle\, p(x) \leftrightarrow \exists t{\geq}0\, \langle x := y(t)\rangle\, p(x) \qquad (y'(t) = f(y))$$

$$(\langle?\rangle) \quad \langle ?Q\rangle\, P \leftrightarrow Q \wedge P$$

$$(\langle\cup\rangle) \quad \langle \alpha \cup \beta\rangle\, P \leftrightarrow \langle\alpha\rangle\, P \vee \langle\beta\rangle\, P$$

$$(\langle;\rangle) \quad \langle \alpha;\beta\rangle\, P \leftrightarrow \langle\alpha\rangle\, \langle\beta\rangle\, P$$

$$(\langle^*\rangle) \quad \langle \alpha^*\rangle\, P \leftrightarrow P \vee \langle\alpha\rangle\, \langle\alpha^*\rangle\, P$$

$$(\langle^d\rangle) \quad \langle \alpha^d\rangle\, P \leftrightarrow \neg\langle\alpha\rangle\, \neg P$$

$$(\langle\cap\rangle) \quad \langle \alpha \cap \beta\rangle\, P \leftrightarrow \langle\alpha\rangle\, P \wedge \langle\beta\rangle\, P$$

$$(\langle^\times\rangle) \quad \langle \alpha^\times\rangle\, P \leftrightarrow P \wedge \langle\alpha\rangle\, \langle\alpha^\times\rangle\, P$$

$$([:=]) \quad [x := e]\, p(x) \leftrightarrow p(e)$$

$$([']) \quad [x' = f(x)]\, p(x) \leftrightarrow \forall t{\geq}0\, [x := y(t)]\, p(x) \qquad (y'(t) = f(y))$$

$$([?]) \quad [?Q]\, P \leftrightarrow (Q \rightarrow P)$$

$$([\cup]) \quad [\alpha \cup \beta]\, P \leftrightarrow [\alpha]\, P \wedge [\beta]\, P$$

$$([;]) \quad [\alpha;\beta]\, P \leftrightarrow [\alpha]\, [\beta]\, P$$

$$([^*]) \quad [\alpha^*]\, P \leftrightarrow P \wedge [\alpha]\, [\alpha^*]\, P$$

$$([^d]) \quad [\alpha^d]\, P \leftrightarrow \neg[\alpha]\, \neg P$$

$$([\cap]) \quad [\alpha \cap \beta]\, P \leftrightarrow [\alpha]\, P \vee [\beta]\, P$$

$$([^\times]) \quad [\alpha^\times]\, P \leftrightarrow P \vee [\alpha]\, [\alpha^\times]\, P$$

$$\text{(loop)}\ \frac{\Gamma \vdash J, \Delta \quad J \vdash [\alpha]\, J \quad J \vdash P}{\Gamma \vdash [\alpha^*]\, P, \Delta} \qquad \text{(ind)}\ \frac{P \rightarrow [\alpha]\, P}{P \rightarrow [\alpha^*]\, P}$$

$$\text{(M)}\ \frac{P \rightarrow Q}{\langle\alpha\rangle\, P \rightarrow \langle\alpha\rangle\, Q} \qquad \text{(FP)}\ \frac{P \vee \langle\alpha\rangle\, Q \rightarrow Q}{\langle\alpha^*\rangle\, P \rightarrow Q}$$

$$\text{(M}[\cdot]\text{)}\ \frac{P \rightarrow Q}{[\alpha]\, P \rightarrow [\alpha]\, Q} \qquad \text{(FP}^\times\text{)}\ \frac{P \vee [\alpha]\, Q \rightarrow Q}{[\alpha^\times]\, P \rightarrow Q}$$

Fig. 5. dGL axiomatization and derived axioms and rules

Infinite Track (Model 4) is an example of infinite horizon switching. A vehicle in a looping track can choose to move in any of the four coordinate directions, but must remain moving at all times. To remain safe for arbitrarily long time, this vehicle must keep switching directions at the right moments, and make *infinitely* many switches. Our approach synthesizes a control envelope that

indicates when it is safe to switch to any given direction. CESAR does not solve this problem as it does not handle infinite control strategies. The control envelope computed by our implementation finds the winning subregion for the overall loop to be $2R > x \land x > -2R \land 2R > y \land y > -2R \land (x > R \lor x < -R \lor y > R \lor y < -R)$, i.e., the subvalue map has a safe strategy for the vehicle at every point in the track.

Reach-avoid Robot (Model 5) demonstrates an example of envelope synthesis for reach-avoid problems, where an agent must *reach* an objective while *avoiding* unsafe situations. Such problems can model the simultaneous requirement of safety and liveness. In this example, the robot is safe in the square $[-2R, 2R] \times [-2R, 2R]$ (enforced in domain constraint of Line 3) and must reach the target region $[R, 2R] \times [R, 2R]$ (on Line 4). The robot can either travel upward ($v_x = 0, v_y = V$) or leftward ($v_x = -V, v_y = 0$). The synthesized subvalue map identifies that it is safely possible to reach the target region starting in $[R, 2R] \times [-2R, 2R]$.

Highway Driving (Model 6) is a time-triggered problem based on the core control challenge from a case study on highway driving [35]. Two cars, are driving on a highway. The lead car, with velocity v_l and acceleration a_l is driven by some external agent with the physical limitation that the car's accelerates is bounded above by A and its braking is bounded below by $-B$ (Line 2). The following car, with velocity v_f and acceleration a_f, is driven by a controller that must follow the lead car while avoiding collisions (Line 3). The controllers revise decisions in a time triggered fashion, with maximum latency T. Regardless of the lead car's behavior, the following car must always be able to stop in time to avoid collision (Line 5). This problem does not fit CESAR's template, which does not support an adversarial agent like the lead car. In the computed control envelope, the winning region for the overall game is $p_f < p_l \land v_f \leq v_l$, i.e. the subvalue map has a strategy to avoid the collisions whenever that velocity of the controlled car is less than that of the lead car and hasn't already collided with it.

The descriptions of the CESAR benchmarks can be found in [30]. The envelopes that we synthesize for these benchmarks are the same as those computed by CESAR.

E.2 Quadcopter

The procedurally generated Quadcopter benchmark suite is based on the following template with the addition of randomly generated values for velocity and obstacles.

Model 3 Quadcopter Suite Template

$$
\begin{array}{rll}
\text{assumptions} & 1 & x > 0 \land V = \texttt{<velocity>} \rightarrow \langle \\
\text{direction} & 2 & \Big(\big(v_y := 1\big) \cup \big(v_y := -1\big) \Big); \\
\text{plant} & 3 & t := 0; \big\{x' = V, y' = v_y, t' = 1 \;\&\; \texttt{<avoid obstacles>} \land y \geq 0\big\}; \\
\text{safe} & 4 & ?t \geq 1\Big)^* \rangle\, x > 20
\end{array}
$$

The Quadcopter chooses whether to go up or down and runs continuous dynamics. The catch is that the quadcopter has a slow processor and cannot make a new choice in less than 1 second (Line 4). However, it can choose to run the dynamics for any time of its choice that is greater than 1 second (Line 3 has an Angel ODE). It can repeat this process of choosing and then sticking with a choice any number of times (Angel loop in Line 4). While it flies, it must avoid obstacles at all times. It must also not crash into the floor at $y > 0$. It must eventually reach the target region $x > 20$. CESAR cannot solve this problem since it lies outside the template. NYCS solves all instances.

An example of the formula that can be generated for <avoid obstacles> obstacles is $(x < 2 \lor 2 < x \lor y < 0 \lor 3 < y) \land (x < 5 \lor 6 < x \lor y < 0 \lor 3 < y) \land (x < 5 \lor 8 < x \lor y < 9 \lor 9 < y) \land (x < 8 \lor 10 < x \lor y < 7 \lor 7 < y)$. This formula has four obstacles. The first one is a line segment from $(2, 0)$ to $(2, 3)$. The second is a rectangle from $(5, 0)$ to $(6, 3)$. The third is a rectangle from $(5, 9)$ to $(8, 9)$. The fourth is a rectangle from $(8, 7)$ to $(10, 7)$. An example of a generated number that fills <velocity> is 1. The procedural generation parameters allow up to 5 obstacles with widths and heights of at most 3 distributed in the region $[0, 10] \times [0, 10]$. Velocity is an integer between 1 and 5.

As expected, the implementation solves problems with fewer obstacles quickly, with the fastest solutions for one obstacle completing in a couple of seconds. However, the time taken increases with the number of obstacles, and more importantly, when obstacles are placed in such a way that they interact to make the solution space more complex. Table 1 summarizes the generated problems and their solving outcomes.

As a demonstrative exmaple, we discuss the control envelope computed for problem the 23 in the benchmark suite. Fig. 6 shows a Mathematica plot of the winning subregion for the overall loop of this example. The problem has three obstacles, two of which overlap. It is naturally unsafe to start at an obstacle. But it is further unsafe to start right before an obstacle because collision with the obstacle becomes unavoidable. Notice a triangle of unsafe space right after the lowest obstacle. This unsafe tringle is an example how obstacles can interact to create unsafe regions. In this triangle, it is neither safe to go upwards because of collision with the upper obstacle, nor is it safe to go downwards because of collision with the floor. This envelope is computed in 68 seconds.

E.3 Full Benchmark Listings

We list the dGL formulas for all the new examples in Fig. 3a.

Model 4 Infinite Track

assumptions	1	$T > 0 \land V > 0 \land R > 0 \land 2VT < R \rightarrow \langle$
east/west	2	$\Big(\big((v_x := V; v_y := 0) \cup (v_x := -V; v_y := 0) \cup$
north/south	3	$(v_x := 0; v_y := -V) \cup (v_x := 0; v_y := V) \big)$;
plant	4	$t := 0; \{x' = v_x, y' = v_y, t' = 1 \;\&\; t \leq T\}^d \Big)^\times$
safe	5	$\rangle (2R > x \land x > -2R \land 2R > y \land y > -2R \land$
	6	$(x > R \lor x < -R \lor y > R \lor y < -R))$

Model 5 Reach-avoid Robot

assumptions	1	$V > 0 \land R > 0 \rightarrow \langle$
direction	2	$\Big(\big((v_x := -V; v_y := 0) \cup (v_x := 0; v_y := V) \big)$;
plant	3	$\{x' = v_x, y' = v_y \& (2R \geq x \land x \geq -2R \land 2R \geq y \land y \geq -2R)\} \Big)^* \rangle$
safe	4	$R \leq x \land x \leq 2R \land R \leq y \land y \leq 2R$

Table 1. Quadcopter benchmark suite generated parameters and solving outcomes. In the result column, ✓ indicates that a nonempty control envelope was computed, while ✗ indicates that algorithm timed out after 20 minutes, failing to compute an envelope.

No.	Velocity	Obstacles	Result
1	5	$x < 9 \vee 11 < x \vee y < 1 \vee 3 < y$	✓
2	4	$x < 9 \vee 9 < x \vee y < 3 \vee 3 < y$	✓
3	1	$(x < 2 \vee 2 < x \vee y < 0 \vee 3 < y) \wedge (x < 5 \vee 6 < x \vee y < 0 \vee 3 < y) \wedge (x < 5 \vee 8 < x \vee y < 9 \vee 9 < y) \wedge (x < 8 \vee 10 < x \vee y < 7 \vee 7 < y)$	✗
4	1	$x < 3 \vee 6 < x \vee y < 9 \vee 10 < y$	✓
5	2	$x < 0 \vee 1 < x \vee y < 2 \vee 4 < y$	✓
6	2	$(x < 7 \vee 10 < x \vee y < 8 \vee 8 < y) \wedge (x < 3 \vee 5 < x \vee y < 8 \vee 9 < y)$	✓
7	4	$(x < 7 \vee 8 < x \vee y < 1 \vee 3 < y) \wedge (x < 5 \vee 5 < x \vee y < 6 \vee 8 < y)$	✗
8	1	$(x < 3 \vee 3 < x \vee y < 8 \vee 11 < y) \wedge (x < 0 \vee 0 < x \vee y < 1 \vee 1 < y)$	✓
9	5	$(x < 1 \vee 3 < x \vee y < 7 \vee 9 < y) \wedge (x < 4 \vee 4 < x \vee y < 0 \vee 3 < y)$	✗
10	1	$\top$	✓
11	2	$x < 3 \vee 4 < x \vee y < 6 \vee 8 < y$	✓
12	2	$(x < 10 \vee 13 < x \vee y < 10 \vee 11 < y) \wedge (x < 5 \vee 5 < x \vee y < 7 \vee 8 < y) \wedge (x < 7 \vee 10 < x \vee y < 10 \vee 12 < y) \wedge (x < 3 \vee 4 < x \vee y < 7 \vee 8 < y) \wedge (x < 2 \vee 2 < x \vee y < 2 \vee 4 < y)$	✗
13	3	$(x < 2 \vee 2 < x \vee y < 7 \vee 7 < y) \wedge (x < 8 \vee 11 < x \vee y < 7 \vee 8 < y) \wedge (x < 1 \vee 2 < x \vee y < 9 \vee 12 < y)$	✗
14	4	$x < 9 \vee 11 < x \vee y < 4 \vee 7 < y \wedge x < 3 \vee 3 < x \vee y < 8 \vee 8 < y \wedge x < 9 \vee 11 < x \vee y < 8 \vee 11 < y \wedge x < 8 \vee 11 < x \vee y < 4 \vee 7 < y \wedge x < 7 \vee 8 < x \vee y < 9 \vee 10 < y$	✓
15	3	$(x < 1 \vee 4 < x \vee y < 9 \vee 10 < y) \wedge (x < 7 \vee 10 < x \vee y < 0 \vee 3 < y) \wedge (x < 8 \vee 10 < x \vee y < 8 \vee 11 < y) \wedge (x < 3 \vee 6 < x \vee y < 3 \vee 4 < y)$	✗
16	4	$(x < 1 \vee 1 < x \vee y < 8 \vee 11 < y) \wedge (x < 8 \vee 10 < x \vee y < 6 \vee 9 < y) \wedge (x < 5 \vee 8 < x \vee y < 8 \vee 11 < y) \wedge (x < 6 \vee 6 < x \vee y < 3 \vee 4 < y) \wedge (x < 10 \vee 13 < x \vee y < 10 \vee 13 < y)$	✗
17	4	$\top$	✓
18	3	$(x < 5 \vee 5 < x \vee y < 4 \vee 4 < y) \wedge (x < 1 \vee 2 < x \vee y < 5 \vee 8 < y) \wedge (x < 8 \vee 11 < x \vee y < 9 \vee 12 < y) \wedge (x < 2 \vee 2 < x \vee y < 1 \vee 3 < y) \wedge (x < 6 \vee 7 < x \vee y < 1 \vee 2 < y)$	✗
19	3	$(x < 4 \vee 6 < x \vee y < 0 \vee 0 < y) \wedge (x < 8 \vee 8 < x \vee y < 9 \vee 10 < y) \wedge (x < 2 \vee 2 < x \vee y < 2 \vee 2 < y) \wedge (x < 3 \vee 3 < x \vee y < 0 \vee 2 < y) \wedge (x < 6 \vee 9 < x \vee y < 8 \vee 11 < y)$	✓
20	2	$(x < 2 \vee 3 < x \vee y < 9 \vee 11 < y) \wedge (x < 10 \vee 13 < x \vee y < 4 \vee 5 < y) \wedge (x < 9 \vee 11 < x \vee y < 8 \vee 11 < y)$	✓
21	3	$(x < 4 \vee 4 < x \vee y < 5 \vee 5 < y) \wedge (x < 5 \vee 6 < x \vee y < 7 \vee 9 < y) \wedge (x < 10 \vee 13 < x \vee y < 3 \vee 4 < y)$	✓
22	4	$(x < 6 \vee 8 < x \vee y < 3 \vee 4 < y)$	✓
23	3	$(x < 5 \vee 8 < x \vee y < 8 \vee 9 < y) \wedge (x < 0 \vee 3 < x \vee y < 3 \vee 6 < y) \wedge (x < 6 \vee 8 < x \vee y < 6 \vee 9 < y)$	✓
24	4	$(x < 3 \vee 3 < x \vee y < 7 \vee 10 < y) \wedge (x < 9 \vee 9 < x \vee y < 1 \vee 4 < y) \wedge (x < 3 \vee 5 < x \vee y < 10 \vee 12 < y) \wedge (x < 4 \vee 4 < x \vee y < 6 \vee 9 < y)$	✓
25	1	$(x < 6 \vee 6 < x \vee y < 9 \vee 10 < y)$	✓

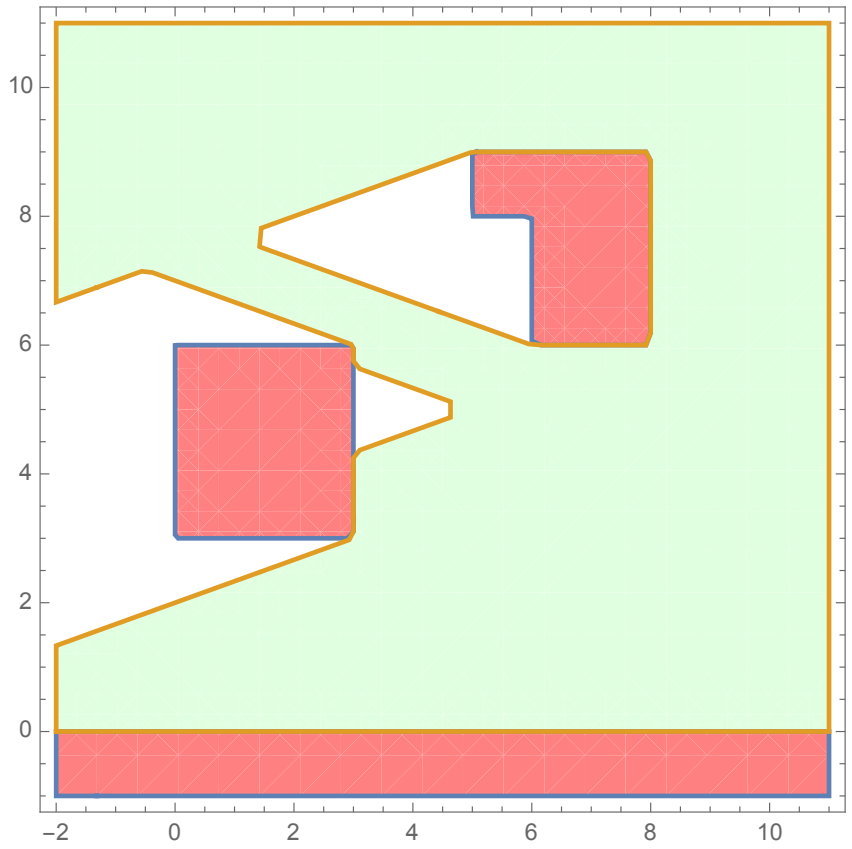


Fig. 6. Computed control envelope for quadcopter problem 23. Red regions are unsafe (obstacles or floor). Green regions are safe starting points per the computed control envelope. The x and y coordinates represent the quadcopter's position.

Model 6 Highway Driving

assumptions	1	$A > 0 \wedge B > 0 \wedge T > 0 \wedge v_f > 0 \wedge v_l > 0 \rightarrow \langle$
lead car	2	$\left(a_l := \otimes; !{-B} < a_l \wedge a_l < A;\right.$
controlled car	3	$a_f := *; ? -B < a_f \wedge a_f < A;$
plant	4	$t := 0; \left\{p_f' = v_f, v_f' = a_f, p_l' = v_l, v_l' = a_l, t' = 1 \;\&\; t \leq T\right\}^d \left.\right)^\times \rangle$
safe	5	$p_f < p_l$

Fig. 3b compares the performance of our approach to CESAR on the CESAR benchmark suite. The performance is similar at most benchmarks with only a few seconds of overhead. An exception is Intersection, where CESAR is significantly more expensive. The observed differences are likely a consequence of different simplification heuristics rather than any fundamental algorithmic

Table 2. Rewrite heuristics used in the evaluation problems.

Heuristc	Problems
Extremal assignment rewrite	Surgical robot, Highway driving
Adversarial one-shot rewrite	Event-triggered ETCS
Loop unrolling	Reach-avoid robot, Quadcopter
CESAR rewrite	Highway driving, CESAR benchmarks

reasons. Simplification is itself an expensive operation, but if done at the right juncture, can make the even more expensive quantifier eliminations that occur later in symbolic execution cheaper. Thus simplification has a large influence on performance. Further code tuning and heuristics optimizations can likely reduce the overheads. LLM assistance was used to generate some of the boilerplate code of the implementation.

E.4 Rewrite Heuristics

Our implementation uses rewriting heuristics to simplify problems into shapes where they can either be symbolically executed or solved using other invariant generation heuristics. Table 2 lists the rewrite heuristics and which evaluation problem each is used in.

We discuss how the rewrite heuristics work.

- *Extremal assignment rewrite.* This rewrite replaces a guarded assignment (e.g., $x := \otimes; ?x > 0$) with an assignment to an extremal value identified from the guard (e.g., $x := 0$). The rewritten program is now much easier to reason about, since a free assignment otherwise requiring a quantifier to reason about has been replaced by a constant expression.
- *Adversarial one-shot rewrite.* This rewrite is similar in spirit to the *one-shot unrolling* of CESAR [30]. It takes an Angel loop ending with an Angel ODE (e.g., the inner loop of Model 1) and replaces it with a single iteration of the loop but with the ODE changed to a Demon ODE guarded at the end by an Angel test for the domain constraint of the original ODE. The intuition is that the rewritten program emulates running any number of iterations of the original loop with the ODE run for any amount of time, so a precondition for the harder rewritten program is likely to also be a precondition for the original program.
- *Loop unrolling.* This heuristic unrolls loops a fixed number of times. For example, $\langle\alpha^*\rangle\phi$ is rewritten as $\langle\alpha\rangle\phi$ or $\langle\alpha\rangle\langle\alpha\rangle\phi$ or $\langle\alpha\rangle\langle\alpha\rangle\langle\alpha\rangle\phi$ and so on.
- *CESAR rewrite.* This is the one-shot refinement and multi-shot unrolling form CESAR [30] where a time-triggered control loop is replaced by running the ODE for an unbounded amount of time. Because of the of our framework's retrospective checks, the requirement that the original program should have an idempotence property ("action permanence") which was essential to the soundness of CESAR is relaxed.

E.5 Refinements

While synthesizing subvalue maps for loops, Algorithm 1 uses checks to test invariant candidates before accepting them. It is possible to use *refinements* to perform these checks. A game β is an *Angelic refinement* of game α when for any formula ψ, $\models \langle\beta\rangle\psi \to \langle\alpha\rangle\psi$. Some rewrites result in refinements, making the invariant candidate generated correct by construction. For instance, for the Angelic loop in Reach-avoid robot (Model 5), let α be the loop body (Line 2 and Line 3), and ϕ the postcondition (Line 4). The inductiveness check is $\mathsf{Inv} \to \langle(a :\alpha^*)^{\exists}_{\dagger} S\rangle\phi$. If Inv is computed by rewriting loop $\langle\alpha^*\rangle\phi$ as $\langle\alpha\rangle\phi$ (loop unrolling rewrite) and then setting $\mathsf{Inv} = \mathsf{simpl}(\langle\alpha\rangle\phi)$ while writing all the intermediate symbolic execution steps of simpl into the subvalue map S, then Inv

will pass the check by construction. The reason is that $\langle a{:}\alpha \Cup S\rangle\phi$ is a refinement of $\langle (a{:}\alpha)^* \Cup S\rangle\phi$. Since $\models \mathsf{Inv} \rightarrow \langle a{:}\alpha \Cup S\rangle\phi$ by construction, it follows that $\models \mathsf{Inv} \rightarrow \langle (a{:}\alpha)^* \Cup S\rangle\phi$. We exploit such refinement properties in our implementation to soundly reduce the number of expensive checks performed.

F Proofs

We prove the correctness of the theorems in the paper. Notation $S|_U$ is the restriction of the subvalue map S to the set of subgames $U \cup \{\mathsf{end}\}$.

Definition F.1 (Weak ordering). We define a weaker ordering relation $\gtrsim$ on inductive subvalue maps that compares formulas at each subgame separately. For two inductive subvalue maps S_1 and S_2 for the game $a{:}\alpha$, S_1 is at least as good as S_2, written $S_1 \gtrsim S_2$, iff for each subgame b in $\mathsf{subgames}(a{:}\alpha)$, $\models S_2(b) \rightarrow S_1(b)$ and $\models S_2(\mathsf{end}) \rightarrow S_1(\mathsf{end})$.

Lemma F.2 (Ordering subsumption). *If $S \gtrsim S'$, then $S \supseteq S'$.*

Proof. By definition of $\gtrsim$, for each subgame $b \in \mathsf{subgames}(\alpha) \cup \{\mathsf{end}\}$, $\models S'(b) \rightarrow S(b)$. Using the dGL Gödel generalization rule G, we infer $\models \langle (a{:}\alpha_{:b}) \Cup S'\rangle(S(b) \rightarrow S'(b))$ showing that $S \supseteq S'$ when interpreted as an Angelic strategy. When interpreted as a Demonic strategy, the same argument holds: using the dGL Gödel generalization rule G we can infer that $\models [(a{:}\alpha_{:b})]\Cap S'](S'(b) \rightarrow S(b))$. □

Lemma F.3 (Subvalue Map Projection is a Refinement). *The projection of an Angelic subvalue map onto a game α is an Angelic refinement of the game, i.e.,*

$$\models \langle a{:}\alpha \Cup S\rangle\psi \rightarrow \langle\alpha\rangle\psi.$$

Dually, the projection of a Demonic subvalue map is a Demonic refinement.

Proof. We show this for Angelic subvalue maps, the proof for Demonic subvalue maps is symmetric. Intuitively this lemma is true because the projection of an Angelic subvalue map only adds Angelic tests, which make the game monotonically harder. The proof uses structural induction based on the shape of $a{:}\alpha$. When $a{:}\alpha$ is:

- Atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := *, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, the result is immediate since $a{:}\alpha \Cup S = \alpha$.
- Atomic and controlled by Angel, i.e., $\alpha \in \{x := \otimes, \{x' = f(x)\ \&\ Q\}\}$, then $a{:}\alpha \Cup S = \alpha\,;\, ?S(\mathsf{end})$. By the dGL axioms $\langle ?\rangle$ and $\langle;\rangle$, $\models \langle\alpha\,;\, ?S(\mathsf{end})\rangle\psi \leftrightarrow \langle\alpha\rangle(\psi \wedge S(\mathsf{end}))$. By the dGL axiom M, $\models \langle\alpha\rangle(\psi \wedge S(\mathsf{end})) \rightarrow \langle\alpha\rangle\psi$. By transitivity of implication, $\models \langle a{:}\alpha \Cup S\rangle\psi \rightarrow \langle\alpha\rangle\psi$.
- $g{:}\gamma \cup d{:}\delta$, then $a{:}\alpha \Cup S = (?S(g)\,;\, g{:}\gamma \Cup S) \cup (?S(d)\,;\, d{:}\delta \Cup S)$. By the dGL axiom $\langle\cup\rangle$, $\models \langle(?S(g)\,;\, g{:}\gamma \Cup S) \cup (?S(d)\,;\, \langle ?S(g)\,;\, g{:}\gamma \Cup S\rangle\psi \vee \langle ?S(d)\,;\, d{:}\delta \Cup S\rangle\psi$. By the dGL axioms $\langle;\rangle$ and $\langle\cup\rangle$, $\models \langle ?S(g)\,;\, g{:}\gamma \Cup S\rangle\psi \vee \langle ?S(d)\,;\, d{:}\delta \Cup S\rangle\psi \leftrightarrow (S(g) \wedge \langle g{:}\gamma \Cup S\rangle\psi) \vee (S(d) \wedge \langle d{:}\delta \Cup S\rangle\psi)$. By conjunction elimination, $\models (S(g) \wedge \langle g{:}\gamma \Cup S\rangle\psi) \vee (S(d) \wedge \langle d{:}\delta \Cup S\rangle\psi) \rightarrow \langle a{:}\gamma \Cup S\rangle\psi \vee \langle d{:}\delta \Cup S\rangle\psi$. By the inductive hypothesis, $\models \langle g{:}\gamma \Cup S\rangle\psi \rightarrow \langle\gamma\rangle\psi$ and $\models \langle d{:}\delta \Cup S\rangle\psi \rightarrow \langle\delta\rangle\psi$. By the transitivity of implication, $\models \langle g{:}\gamma \Cup S\rangle\psi \vee \langle d{:}\delta \Cup S\rangle\psi \rightarrow \langle\gamma\rangle\psi \vee \langle\delta\rangle\psi$. By the dGL axiom $\langle\cup\rangle$, $\models \langle\gamma\rangle\psi \vee \langle\delta\rangle\psi \leftrightarrow \langle\gamma \cup \delta\rangle\psi$.
- $g{:}\gamma \cap d{:}\delta$, then $a{:}\alpha \Cup S = g{:}\gamma \Cup S \cap d{:}\delta \Cup S$. By the dGL axiom $[\cup]$, $\models \langle g{:}\gamma \Cup S \cap d{:}\delta \Cup S\rangle\psi \leftrightarrow \langle g{:}\gamma \Cup S\rangle\psi \wedge \langle d{:}\delta \Cup S\rangle\psi$. By the inductive hypothesis, $\models \langle g{:}\gamma \Cup S\rangle\psi \rightarrow \langle\gamma\rangle\psi$ and $\models \langle d{:}\delta \Cup S\rangle\psi \rightarrow \langle\delta\rangle\psi$. Thus, $\models \langle g{:}\gamma \Cup S\rangle\psi \wedge \langle d{:}\delta \Cup S\rangle\psi \rightarrow \langle\gamma\rangle\psi \wedge \langle\delta\rangle\psi$. By the dGL axiom $[\cup]$, $\models \langle\gamma\rangle\psi \wedge \langle\delta\rangle\psi \leftrightarrow \langle\gamma \cap \delta\rangle\psi$.
- $g{:}\gamma; d{:}\delta$, then $a{:}\alpha \Cup S = g{:}\gamma \Cup S(\mathsf{end} \mapsto S(d))\,;\, d{:}\delta \Cup S$.
  Using the dGL axiom $\langle;\rangle$,
  $\models \langle g{:}\gamma \Cup S(\mathsf{end} \mapsto S(d))\,;\, d{:}\delta \Cup S\rangle\psi \leftrightarrow \langle g{:}\gamma \Cup S(\mathsf{end} \mapsto S(d))\rangle\langle d{:}\delta \Cup S\rangle\psi$. By the inductive hypothesis, $\models \langle d{:}\delta \Cup S\rangle\psi \rightarrow \langle\delta\rangle\psi$.

Thus, $\models \langle g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(d))\rangle\langle d{:}\delta \mathbin{\dot{\cup}} S\rangle\psi \rightarrow \langle g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(d))\rangle \langle\delta\rangle\psi$. Further, by the inductive hypothesis, $\models \langle g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(d))\rangle\langle\delta\rangle\psi \rightarrow \langle\gamma\rangle\langle\delta\rangle\psi$. By the dGL axiom $\langle;\rangle$, $\models \langle\gamma\rangle\langle\delta\rangle\psi \leftrightarrow \langle\gamma;\delta\rangle\psi$.

- $a{:}(g{:}\gamma)^*$, then $a{:}\alpha \mathbin{\dot{\cup}} S = (?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(s)))^*; ?S(\text{end})$. By dGL axioms $\langle;\rangle$, $\langle?\rangle$, and M, $\models \langle a{:}\alpha \mathbin{\dot{\cup}} S\rangle\psi \rightarrow \langle(?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(a)))^*\rangle\psi$. By the dGL proof rule FP,
$$\frac{\psi \vee \langle(?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(a)))\rangle\langle\gamma^*\rangle\psi \rightarrow \langle\gamma^*\rangle\psi}{\langle(?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(a)))^*\rangle\psi \rightarrow \langle\gamma^*\rangle\psi}\ .$$
Now, $\langle(?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S(\text{end} \mapsto S(a)))\rangle\langle\gamma^*\rangle\psi$, per the inductive hypothesis, implies $\langle\gamma\rangle\langle\gamma^*\rangle\psi$. Per dGL axiom $\langle^*\rangle$, $\langle\gamma\rangle\langle\gamma^*\rangle\psi \leftrightarrow \langle\gamma^*\rangle\psi$. This allows us to show the premise, since $\models \psi \vee \langle\gamma^*\rangle\psi \rightarrow \langle\gamma^*\rangle\psi$.
- $a{:}(g{:}\gamma)^\times$, then $a{:}\alpha \mathbin{\dot{\cup}} S = \left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times$. By the dGL proof rule loop,
$$\frac{\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \vdash J \quad J \vdash \langle\gamma\rangle J \quad J \vdash \psi}{\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \vdash \langle\gamma^\times\rangle\psi}\ .$$
Setting invariant J to $\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi$, the first premise is immediate.
The second premise is now $\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \vdash \langle\gamma\rangle\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi$. By $\langle^\times\rangle$, $\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \rightarrow \langle g{:}\gamma \mathbin{\dot{\cup}} S\rangle\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi$.
By the inductive hypothesis, $\models \langle g{:}\gamma \mathbin{\dot{\cup}} S\rangle\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \rightarrow \langle\gamma\rangle\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi$, thus proving the second premise.
The third premise is proved as follows.

$$\cfrac{\cfrac{\cfrac{*}{\psi \vdash \psi}\,\text{id}}{\psi \wedge \langle g{:}\gamma \mathbin{\dot{\cup}} S\rangle\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \vdash \psi}\,\text{WL}}{\langle\left(g{:}\gamma \mathbin{\dot{\cup}} S\right)^\times\rangle\psi \vdash \psi}\,\langle^\times\rangle$$

□

Lemma F.4 (MPC Subvalue Map Projection Equivalence). *The projection game $\langle a{:}\alpha \mathbin{\dot{\cup}} S\rangle\phi$ of the MPC Angelic subvalue map (Def. 6.5) is equivalent to the original game $\langle\alpha\rangle\phi$. Dually, the projection game $[a{:}\alpha[]\mathbin{\dot{\cup}} S]\phi$ of the MPC Demonic subvalue map (Def. 6.5) is equivalent to the original game $[\alpha]\phi$.*

Proof. We show this for Angelic subvalue maps, the proof for Demonic subvalue maps is symmetric. Intuitively this lemma is true because the projection of an Angelic subvalue map only adds Angelic tests, which make the game monotonically harder. The proof uses structural induction based on the shape of $a{:}\alpha$. Note that for the MPC solution, end is ϕ. When $a{:}\alpha$ is:

- Atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := *, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, the result is immediate since $a{:}\alpha \mathbin{\dot{\cup}} S = \alpha$.
- Atomic and controlled by Angel, i.e., $\alpha \in \{x := \otimes, \{x' = f(x)\ \&\ Q\}\}$, then $a{:}\alpha \mathbin{\dot{\cup}} S = \alpha ; ?\phi$. By the dGL axioms $\langle?\rangle$ and $\langle;\rangle$, $\models \langle\alpha ; ?\phi\rangle\phi \leftrightarrow \langle\alpha\rangle(\phi \wedge \phi) \leftrightarrow \langle\alpha\rangle\phi$.
- $g{:}\gamma \cup d{:}\delta$, then $a{:}\alpha \mathbin{\dot{\cup}} S = (?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S) \cup (?S(d) ; d{:}\delta \mathbin{\dot{\cup}} S)$.
  $\langle(?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S) \cup (?S(d) ; d{:}\delta \mathbin{\dot{\cup}} S)\rangle\phi$
  $\leftrightarrow \langle?S(g) ; g{:}\gamma \mathbin{\dot{\cup}} S\rangle\phi \vee \langle?S(d) ; d{:}\delta \mathbin{\dot{\cup}} S\rangle\phi$ (dGL axiom $\langle\cup\rangle$)
  $\leftrightarrow (S(g) \wedge \langle g{:}\gamma \mathbin{\dot{\cup}} S\rangle\phi) \vee (S(d) \wedge \langle d{:}\delta \mathbin{\dot{\cup}} S\rangle\phi)$ (dGL axioms $\langle;\rangle$, $\langle\cup\rangle$)
  $\leftrightarrow (S(g) \wedge \langle\gamma\rangle\phi) \vee (S(d) \wedge \langle\delta\rangle\phi)$ (inductive hypothesis)
  $\leftrightarrow (\langle\gamma\rangle\phi \wedge \langle\gamma\rangle\phi) \vee (\langle\delta\rangle\phi \wedge \langle\delta\rangle\phi)$ (MPC definition)
  $\leftrightarrow \langle\gamma\rangle\phi \vee \langle\delta\rangle\phi$.
- $g{:}\gamma \cap d{:}\delta$, then $a{:}\alpha \mathbin{\dot{\cup}} S = g{:}\gamma \mathbin{\dot{\cup}} S \cap d{:}\delta \mathbin{\dot{\cup}} S$.

$\langle g;\gamma \mathbin{?}_{\downarrow} S \cap d;\delta \mathbin{?}_{\downarrow} S\rangle\phi$
$\leftrightarrow \langle g;\gamma \mathbin{?}_{\downarrow} S\rangle\phi \wedge \langle d;\delta \mathbin{?}_{\downarrow} S\rangle\phi$ (dGL axiom $[\cup]$)
$\leftrightarrow \langle\gamma\rangle\phi \wedge \langle\delta\rangle\phi$ (inductive hypothesis)
$\leftrightarrow \langle\gamma \cap \delta\rangle\phi$ (dGL axiom $[\cup]$).

- $g;\gamma; d;\delta$, then $a;\alpha \mathbin{?}_{\downarrow} S = g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(d))\,;\, d;\delta \mathbin{?}_{\downarrow} S$.
  $\langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(d))\,;\, d;\delta \mathbin{?}_{\downarrow} S\rangle\phi$
  $\leftrightarrow \langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(d))\rangle\langle d;\delta \mathbin{?}_{\downarrow} S\rangle\phi$ (dGL axiom $[;]$)
  $\leftrightarrow \langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(d))\rangle\langle\delta\rangle\phi$ (inductive hypothesis)
  $\leftrightarrow \langle\gamma\rangle\langle\delta\rangle\phi$ (inductive hypothesis)
  $\leftrightarrow \langle\gamma;\delta\rangle\phi$ (dGL axiom $[;]$).
- $a;(g;\gamma)^*$. By Lemma F.3, $\models \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi \rightarrow \langle\gamma^*\rangle\,\phi$.
  We now show the opposite direction, i.e., $\models \langle\gamma^*\rangle\,\phi \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$.
  Since $S(a) = \langle\gamma^*\rangle\,\phi$, we have $\models \langle\gamma^*\rangle\,\phi \rightarrow S(a)$. From $a;\alpha \models S$, we get $\models S(a) \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$.
  Thus, by transitivity of implication, $\models \langle\gamma^*\rangle\,\phi \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$.
- $a;(g;\gamma)^\times$
  By Lemma F.3, $\models \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi \rightarrow \langle\gamma^\times\rangle\,\phi$. We now show the opposite direction, i.e., $\models \langle\gamma^\times\rangle\,\phi \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$.
  Since $S(a) = \langle\gamma^\times\rangle\,\phi$, we have $\models \langle\gamma^\times\rangle\,\phi \rightarrow S(a)$. We now show that $\models S(a) \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$, so that by transitivity of implication, $\models \langle\gamma^\times\rangle\,\phi \rightarrow \langle a;\alpha \mathbin{?}_{\downarrow} S\rangle\,\phi$.
  By the dGL proof rule loop,
  $$\frac{S(a) \vdash J \quad J \vdash \langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, J \quad J \vdash \phi}{S(a) \vdash \langle \big(g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(a))\big)^\times\rangle\,\phi}\,.$$
  Setting invariant J to $S(a)$, the first premise is immediate. The last premise is $S(a) \vdash \phi$ which follows from $S(a) \leftrightarrow \phi \wedge \langle\gamma\rangle\,\langle\gamma^\times\rangle\,\phi$ per dGL axiom $\langle^\times\rangle$. Only the middle premise remains. It is $S(a) \vdash \langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, S(a)$. Per the inductive hypothesis, $\models \langle g;\gamma \mathbin{?}_{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, S(a) \leftrightarrow \langle\gamma\rangle S(a)$. Thus, the middle premise can be rewritten as $S(a) \vdash \langle\gamma\rangle S(a)$. Recall that $S(a) = \langle\gamma^\times\rangle\,\phi$, so that the middle premise is $\langle\gamma^\times\rangle\,\phi \vdash \langle\gamma\rangle\langle\gamma^\times\rangle\,\phi$. By the dGL axiom $\langle^\times\rangle$, $\langle\gamma^\times\rangle\,\phi \rightarrow \langle\gamma\rangle\langle\gamma^\times\rangle\,\phi$, thus proving the middle premise.

□

Lemma F.5 (MPC Valid). *The MPC solution (Def. 6.5) S for a game $a;\alpha$ for Angel winning condition ϕ is an inductive subvalue map compatible with ϕ: $a;\alpha \models S$. Dually, the MPC solution S for a game $a;\alpha$ for Demon winning condition ϕ is an inductive subvalue map: $a;\alpha[]\models S$.*

Proof. Proved by induction on the structure of the game $a;\alpha$. We show this for the Angel MPC solution; the proof for the Demonic MPC solution is analogous. Firstly, for the MPC solution, $S(\mathsf{end}) = \phi$, so ϕ is a compatible winning condition. The base case is when $a;\alpha$ is atomic, i.e., $\alpha \in \{x := e, x := *, ?f, !f, \{x' = f(x)\ \&\ Q\}, \{x' = f(x)\ \&\ Q\}^d\}$. In this case the MPC subvalue map consists of the mapping from a to $\langle\alpha\rangle\phi$ (along with $\mathsf{end} \mapsto \phi$), which matches the corresponding condition for $a;\alpha \models S$ in Def. 6.2. In the recursive case, if $a;\alpha$ has the structure:

(1) $g;\gamma \cup d;\delta$: by the inductive hypothesis, as $S|_{\mathsf{subgames}(g;\gamma)}$ is the MPC solution for game γ and $S|_{\mathsf{subgames}(d;\delta)}$ is the MPC solution for game δ, $g;\gamma \models S|_{\mathsf{subgames}(g;\gamma)}$ and $d;\delta \models S|_{\mathsf{subgames}(d;\delta)}$. So, $g;\gamma \models S$ and $d;\delta \models S$. It remains to show that $S(a) \rightarrow S(g) \vee S(d)$. By the definition of the MPC solution, $S(a) = \langle g;\gamma \cup d;\delta\rangle\phi$, $S(g) = \langle g;\gamma\rangle\phi$ and $S(d) = \langle d;\delta\rangle\phi$. By the dGL rule $\langle\cup\rangle$, $\langle g;\gamma \cup d;\delta\rangle\phi = \langle g;\gamma\rangle\phi \vee \langle d;\delta\rangle\phi$. Thus, $S(a) \rightarrow S(g) \vee S(d)$.

(2) $g;\gamma \cap d;\delta$: by the inductive hypothesis, as $S|_{\mathsf{subgames}(g;\gamma)}$ is the MPC solution for game γ and $S|_{\mathsf{subgames}(d;\delta)}$ is the MPC solution for game δ, $g;\gamma \models S|_{\mathsf{subgames}(g;\gamma)}$ and $d;\delta \models S|_{\mathsf{subgames}(d;\delta)}$. So, $g;\gamma \models S$ and $d;\delta \models S$. It remains to show that $S(a) \rightarrow S(g) \wedge S(d)$. By the definition of

the MPC solution, $S(a) = \langle g{:}\gamma \cap d{:}\delta\rangle\phi$, $S(g) = \langle g{:}\gamma\rangle\phi$ and $S(d) = \langle d{:}\delta\rangle\phi$. By the dGL rule $\langle\cap\rangle$, $\langle g{:}\gamma \cap d{:}\delta\rangle\phi = \langle g{:}\gamma\rangle\phi \wedge \langle d{:}\delta\rangle\phi$. Thus, $S(a) \to S(g) \wedge S(d)$.

(3) $g{:}\gamma\,;\, d{:}\delta$: by the inductive hypothesis, as $S|_{\mathsf{subgames}(d{:}\delta)}$ is the MPC solution for game δ and Angel winning condition ϕ, $d{:}\delta \models S|_{\mathsf{subgames}(d{:}\delta)}$. So $d{:}\delta \models S$. Further, as $S|_{\mathsf{subgames}(g{:}\gamma)}$ is the MPC solution for game γ and Angel winning condition $\langle d{:}\delta\rangle\phi = S(\delta)$, by the inductive hypothesis, $g{:}\gamma \models S(\mathsf{end} \mapsto S(\delta))|_{\mathsf{subgames}(g{:}\gamma)}$. So $g{:}\gamma \models S(\mathsf{end} \mapsto S(\delta))$. It remains to show that $S(a) \to S(g)$ which is immediate from the definition of the MPC solution since $a{:}\alpha_{a:} = a{:}\alpha_{g:}$ in this case.

(4) $g{:}\gamma^*$: observe that $S|_{\mathsf{subgames}(g{:}\gamma)}$ is the MPC solution for game γ and Angel winning condition $\langle\gamma^*\rangle\phi$. $S(a) = \langle\gamma^*\rangle\phi$ and $S(a) \to \phi$, so $\langle\gamma^*\rangle\phi = S(a) \vee \phi$, so $S|_{\mathsf{subgames}(g{:}\gamma)}$ is the MPC solution for game γ and Angel winning condition $S(a) \vee \phi$. Then, by the inductive hypothesis, $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))|_{\mathsf{subgames}(g{:}\gamma)}$. So $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))$. The remaining condition is $S(a) \to \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\phi$. For MPC solution S, $\langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\,\phi = \langle\gamma^*\rangle\phi$ (Lemma F.4). Since $S(a) = \langle\gamma^*\rangle\phi$, the condition is satisfied.

(5) $g{:}\gamma^\times$: observe that $S|_{\mathsf{subgames}(g{:}\gamma)}$ is the MPC solution for game γ and Angel winning condition $\langle\gamma^\times\rangle\phi$. By definition, $S(a) = \langle\gamma^\times\rangle\phi$. So, by the inductive hypothesis, $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))|_{\mathsf{subgames}(g{:}\gamma)}$. Thus $g{:}\gamma \models S(\mathsf{end} \mapsto S(a))$. Next we show that $\models S(a) \to S(g) \wedge \phi$. Using the dGL axiom $\langle^\times\rangle$, $\langle\gamma^\times\rangle\phi = \langle\gamma\rangle\langle\gamma^\times\rangle\phi \wedge \phi$. Substituting back the MPC definitions of $S(a) = \langle\gamma^\times\rangle\phi$ and $S(g) = \langle\gamma\rangle S(a)$, this means $S(a) = S(g) \wedge \phi$. So, $\models S(a) \to S(g) \wedge \phi$.

□

Lemma F.6 (Subvalue maps win). *For any game $a{:}\alpha$, Angelic subvalue map S, and compatible Angel winning condition ϕ, $\langle\alpha\rangle\,\phi \models \langle a{:}\alpha \Downarrow S\rangle\,\phi$.*

Proof. The proof follows from structural induction along with the application of the usual dGL axioms and proof rules.

- If $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then $\langle\alpha\rangle\,\phi = \langle a{:}\alpha \Downarrow S\rangle\,\phi$ and the result is immediate.
- If $a{:}\alpha = x := *$, then $a{:}\alpha \Downarrow S$ is $x := \otimes\,;\, !S(\mathsf{end})$. Thus we must show that

$$\langle\alpha\rangle\,\phi \models \langle x := \otimes\,;\, !S(\mathsf{end})\rangle\,\phi.$$

By the dGL axioms $\langle;\rangle$ and $[?]$, this is

$$\langle\alpha\rangle\,\phi \models \langle x := \otimes\rangle\,(S(\mathsf{end}) \to \phi).$$

Since ϕ is a compatible Angel winning condition, $S(\mathsf{end}) \to \phi$ holds. Thus, we need to show

$$\langle\alpha\rangle\,\phi \models \langle x := \otimes\rangle\,\top.$$

By the semantics of nondeterministic assignments in dGL, the succedent is $\top$, completing the proof.

- If $a{:}\alpha$ is $\{x' = f(x)\ \&\ Q\}$, then the argument is similar to the previous case. $a{:}\alpha \Downarrow S$ is $\{x' = f(x)\ \&\ Q\}^d\,;\, !S(\mathsf{end})$. Thus we must show that

$$\langle\alpha\rangle\,\phi \models \langle\{x' = f(x)\ \&\ Q\}^d\,;\, !S(\mathsf{end})\rangle\,\phi.$$

Applying the dGL axioms $\langle;\rangle$, $\langle^d\rangle$, and $[?]$, this is

$$\langle\alpha\rangle\,\phi \models \langle\{x' = f(x)\ \&\ Q\}^d\rangle\,(S(\mathsf{end}) \to \phi).$$

Since ϕ is a compatible Angel winning condition, $S(\mathsf{end}) \to \phi$ holds. Thus, we need to show

$$\langle\alpha\rangle\,\phi \models \langle\{x' = f(x)\ \&\ Q\}^d\rangle\,\top.$$

By the semantics of differential games in dGL, the succedent is $\top$, completing the proof.

- If $a{:}\alpha = g{:}\gamma \cup d{:}\delta$, then $a{:}\alpha \between S$ is $(!S(g)\,;\, g{:}\gamma \between S) \cap (!S(d)\,;\, d{:}\delta \between S)$. Thus we must show that

$$\langle\alpha\rangle\,\phi \models \langle(!S(g)\,;\, g{:}\gamma \between S) \cap (!S(d)\,;\, d{:}\delta \between S)\rangle\,\phi.$$

  By the dGL axioms $\langle;\rangle$ and $[\cup]$, this is

$$\langle\alpha\rangle\,\phi \models (S(g) \to \langle g{:}\gamma \between S\rangle\,\phi) \wedge (S(d) \to \langle d{:}\delta \between S\rangle\,\phi).$$

  But by the inductive hypothesis, we have $\langle\gamma\rangle\,\phi \models \langle g{:}\gamma \between S\rangle\,\phi$ and $\langle\delta\rangle\,\phi \models \langle d{:}\delta \between S\rangle\,\phi$. $S(g) \models \langle\gamma\rangle\,\phi$ and $S(d) \models \langle\delta\rangle\,\phi$ by the definition of a subvalue map. By the proof rules $\to$R and WL, the following two formulas hold: $S(g) \to \langle g{:}\gamma \between S\rangle\,\phi$ and $S(d) \to \langle d{:}\delta \between S\rangle\,\phi$. Thus, we have $\langle\alpha\rangle\,\phi \models (S(g) \to \langle g{:}\gamma \between S\rangle\,\phi) \wedge (S(d) \to \langle d{:}\delta \between S\rangle\,\phi)$, completing the proof for this case.
- If $a{:}\alpha = g{:}\gamma^*$, then $\langle a{:}\alpha \between S\rangle\,\phi = \langle(!S(g)\,;\, g{:}\gamma \between S(\mathsf{end} \mapsto S(a)))^\times\,;\, !S(\mathsf{end})\rangle\,\phi$.
  Applying axioms $\langle;\rangle$ and $[?]$, this is

$$\langle(!S(g)\,;\, g{:}\gamma \between S(\mathsf{end} \mapsto S(a)))^\times\rangle\,(S(\mathsf{end}) \to \phi)$$

  .
  Since ϕ is a compatible Angel winning condition, $\models S(\mathsf{end}) \to \phi$.
  Thus, we need to show

$$\langle\alpha\rangle\,\phi \models \langle(!S(g)\,;\, g{:}\gamma \between S(\mathsf{end} \mapsto S(a)))^\times\rangle\,(\top)$$

  This we do using the loop rule with invariant $\langle\alpha\rangle\,\phi$. The inductive stop is shown using the inductive hypothesis, i.e., $\langle\gamma\rangle\,\phi \models \langle g{:}\gamma \between S(\mathsf{end} \mapsto S(a))\rangle\,\langle\alpha\rangle\,\phi$. Here, $\langle\alpha\rangle\,\phi$ is a compatible winning condition because $S(a) \to \langle\alpha\rangle\,\phi$ holds by the definition of a subvalue map.
- If $a{:}\alpha = g{:}\gamma \cap d{:}\delta$, then $a{:}\alpha \between S$ is $(g{:}\gamma \between S) \cap (d{:}\delta \between S)$. We must show that

$$\langle\gamma \cap \delta\rangle\,\phi \models \langle(g{:}\gamma \between S) \cap (d{:}\delta \between S)\rangle\,\phi.$$

  By the dGL axiom $\langle\cup\rangle$, this is

$$\langle\gamma\rangle\,\phi \wedge \langle\delta\rangle\,\phi \models \langle g{:}\gamma \between S\rangle\,\phi \wedge \langle d{:}\delta \between S\rangle\,\phi.$$

  Applying the proof rules $\wedge$R and WL, the goals to show are

$$\langle\gamma\rangle\,\phi \models \langle g{:}\gamma \between S\rangle\,\phi \text{ and } \langle\delta\rangle\,\phi \models \langle d{:}\delta \between S\rangle\,\phi.$$

  Both of these follow from the inductive hypotheses.
- If $a{:}\alpha = g{:}\gamma\,;\, d{:}\delta$, then $a{:}\alpha \between S$ is $(g{:}\gamma \between S(\mathsf{end} \mapsto S(d)))\,;\, (d{:}\delta \between S)$. We must show that

$$\langle\gamma\,;\,\delta\rangle\,\phi \models \langle(g{:}\gamma \between S(\mathsf{end} \mapsto S(d)))\,;\, (d{:}\delta \between S)\rangle\,\phi.$$

  By the dGL axiom $[;]$, this is

$$\langle\gamma\rangle\,\langle\delta\rangle\,\phi \models \langle g{:}\gamma \between S(\mathsf{end} \mapsto S(d))\rangle\,\langle d{:}\delta \between S\rangle\,\phi.$$

  By the inductive hypothesis, we know that $\langle\gamma\rangle\,\langle\delta\rangle\,\phi \to \langle g{:}\gamma \between S(\mathsf{end} \mapsto S(d))\rangle\,S(d)$ holds because $\langle\delta\rangle\,\phi$ is compatible with $S(\mathsf{end} \mapsto S(d))$, since by the definition of a subvalue map, we have $S(d) \to \langle\delta\rangle\,\phi$. Per inductive hypothesis, we also know that $\langle\delta\rangle\,\phi \models \langle d{:}\delta \between S\rangle\,\phi$. Thus by the transitivity of implication along with the monotonicity rule, we complete the proof.
- If $a{:}\alpha = g{:}\gamma^\times$, then $a{:}\alpha \between S$ is $(g{:}\gamma \between S(\mathsf{end} \mapsto S(a))^\times)$. We must show that

$$\langle\gamma^\times\rangle\,\phi \models \langle(g{:}\gamma \between S(\mathsf{end} \mapsto S(a)))^\times\rangle\,\phi.$$

  We show this by using the dGL loop rule loop with invariant $\langle\gamma^\times\rangle\,\phi$. The inductive stop is shown using the inductive hypothesis, i.e.,

$$\langle\gamma\rangle\,\langle\gamma^\times\rangle\,\phi \models \langle g{:}\gamma \between S(\mathsf{end} \mapsto S(a))\rangle\,\langle\gamma^\times\rangle\,\phi,$$

where $\langle \gamma^\times \rangle \phi$ is a compatible winning condition because $S(a) \rightarrow \langle \gamma^\times \rangle \phi$ holds by the definition of a subvalue map. The invariant holds initially by id and implies the postcondition by the dGL axiom $\langle^\times\rangle$.

□

Lemma F.7 (Inductive subvalue maps have some strategy). *For any game $a{:}\alpha$, Angelic inductive subvalue map S, and compatible Angel winning condition ϕ, $S(a) \models \langle a{:}\alpha \fatsemi S \rangle \phi$.*

Proof. This follows from structural induction, unwrapping the inductive subvalue map definition, and applying the usual dGL axioms and proof rules.

- If $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x) \,\&\, Q\}^d\}$, then $a{:}\alpha \fatsemi S = a{:}\alpha$. In these cases, by the definition of a inductive subvalue map, $\models S(a) \rightarrow \langle a{:}\alpha \rangle \mathsf{end}$. Further since ϕ is a compatible winning condition, $\models S(\mathsf{end}) \rightarrow \phi$. By the monotonicity rule, $S(a) \models \langle a{:}\alpha \rangle \phi$. This allows us to conclude that $S(a) \models \langle a{:}\alpha \fatsemi S \rangle \phi$.
- If $a{:}\alpha = a{:}x := *$, then $a{:}\alpha \fatsemi S = x := * \,;\, ?S(\mathsf{end})$. Thus we must show that

  $$S(a) \models \langle x := * \,;\, ?S(\mathsf{end}) \rangle \phi.$$

  By the dGL axioms $\langle;\rangle$ and $\langle?\rangle$, this is

  $$S(a) \models \langle x := * \rangle S(\mathsf{end}) \wedge \phi.$$

  Since ϕ is a compatible Angel winning condition, $S(\mathsf{end}) \rightarrow \phi$ holds. Applying the monotonicity rule, we must show that

  $$S(a) \models \langle x := * \rangle S(\mathsf{end}).$$

  By the definition of an inductive subvalue map for Angelic free assignments along with the $\rightarrow$R rule, we conclude that $S(a) \models \langle x := * \rangle S(\mathsf{end})$.
- If $a{:}\alpha = a{:}\{x' = f(x) \,\&\, Q\}$, the proof follows exactly the same steps as the previous case.
- If $a{:}\alpha = g{:}\gamma \cup d{:}\delta$, then $a{:}\alpha \fatsemi S = (?S(g) \,;\, g{:}\gamma \fatsemi S) \cup (?S(d) \,;\, d{:}\delta \fatsemi S)$. Thus we must show that

  $$S(a) \models \langle (?S(g) \,;\, g{:}\gamma \fatsemi S) \cup (?S(d) \,;\, d{:}\delta \fatsemi S) \rangle \phi.$$

  By the dGL axioms $\langle;\rangle$ and $\langle\cup\rangle$, this is

  $$S(a) \models (S(g) \rightarrow \langle g{:}\gamma \fatsemi S \rangle \phi) \vee (S(d) \rightarrow \langle d{:}\delta \fatsemi S \rangle \phi).$$

  By the definition of an inductive subvalue map, we have $\models S(a) \rightarrow S(g) \vee S(d)$. Thus by proof rules cut, $\vee$L, $\vee$R, and WR it suffices to show the following two goals:

  $$S(g) \models S(g) \rightarrow \langle g{:}\gamma \fatsemi S \rangle \phi \text{ and } S(d) \models S(d) \rightarrow \langle d{:}\delta \fatsemi S \rangle \phi.$$

  Applying the rule $\rightarrow$R, the remaining goals hold per the inductive hypotheses, $S(g) \models \langle g{:}\gamma \fatsemi S \rangle \phi$ and $S(d) \models \langle d{:}\delta \fatsemi S \rangle \phi$.
- We show the case of Angel loop, which follows directly from the definition of an inductive subvalue map. If $a{:}\alpha = g{:}\gamma^*$, then by the definition of an inductive subvalue map, $S(a) \models \langle a{:}\alpha \fatsemi S \rangle S(\mathsf{end})$. Since ϕ a compatible winning condition, we have $S(\mathsf{end}) \models \phi$. Using dGL rule M, we can conclude that $S(a) \models \langle a{:}\alpha \fatsemi S \rangle \phi$.
- If $a{:}\alpha = g{:}\gamma \,;\, d{:}\delta$, then $a{:}\alpha \fatsemi S = g{:}\gamma \fatsemi S(\mathsf{end} \mapsto S(d)) \,;\, d{:}\delta \fatsemi S$. Thus we must show that

  $$S(a) \models \langle g{:}\gamma \fatsemi S(\mathsf{end} \mapsto S(d)) \,;\, d{:}\delta \fatsemi S \rangle \phi.$$

  By the dGL axioms $\langle;\rangle$, this is

  $$S(a) \models \langle g{:}\gamma \fatsemi S(\mathsf{end} \mapsto S(d)) \rangle \langle d{:}\delta \fatsemi S \rangle \phi.$$

By the inductive hypothesis along with the fact that $\models S(a) \rightarrow S(g)$ from the definition of inductive subvalue maps,

$$S(a) \models \langle g{:}\gamma \supseteq S(\mathsf{end} \mapsto S(d))\rangle\, S(d).$$

Additionally, by the inductive hypothesis,

$$S(d) \models \langle d{:}\delta \supseteq S\rangle\, \phi.$$

Applying the M rule, we can conclude that

$$S(a) \models \langle g{:}\gamma \supseteq S(\mathsf{end} \mapsto S(d))\rangle\, \langle d{:}\delta \supseteq S\rangle\, \phi,$$

completing the proof for this case.

- If $a{:}\alpha = g{:}\gamma \cap d{:}\delta$, then $a{:}\alpha \supseteq S = g{:}\gamma \supseteq S \cap d{:}\delta \supseteq S$. Thus we must show that

$$S(a) \models \langle g{:}\gamma \supseteq S \cap d{:}\delta \supseteq S\rangle\, \phi.$$

By the dGL axiom $\langle\cap\rangle$, this is

$$S(a) \models \langle g{:}\gamma \supseteq S\rangle\, \phi \wedge \langle d{:}\delta \supseteq S\rangle\, \phi.$$

By the definition of an inductive subvalue map, we have $S(a) \models S(g) \wedge S(d)$. Applying the proof rules $\wedge$R, cut, $\wedge$L and WL, the goals to show are

$$S(g) \models \langle g{:}\gamma \supseteq S\rangle\, \phi \text{ and } S(d) \models \langle d{:}\delta \supseteq S\rangle\, \phi.$$

Both of these follow from the inductive hypotheses, completing the proof for this case.

- If $a{:}\alpha = g{:}\gamma^\times$, then $a{:}\alpha \supseteq S = (g{:}\gamma \supseteq (S(\mathsf{end} \mapsto S(a))))^\times$. Thus we must show that

$$S(a) \models \langle (g{:}\gamma \supseteq (S(\mathsf{end} \mapsto S(a))))^\times\rangle\, S(\mathsf{end}).$$

We use the loop rule with invariant $S(a)$. $S(a)$ holds initially by assumption. We show that is holds inductively, i.e., that

$$S(a) \models \langle g{:}\gamma \supseteq (S(\mathsf{end} \mapsto S(a)))\rangle\, S(a).$$

By the definition of an inductive subvalue map, $\models S(a) \rightarrow S(g)$. Thus, after applying cut, we must show

$$S(g) \models \langle g{:}\gamma \supseteq (S(\mathsf{end} \mapsto S(a)))\rangle\, S(a).$$

This holds by the inductive hypothesis. Finally, the invariant implies the postcondition by the definition of an inductive subvalue map, which requires that $\models S(a) \rightarrow S(\mathsf{end})$.

□

Lemma F.8 (Inductive subvalue maps win). *For any game $a{:}\alpha$, Angelic inductive subvalue map S, and compatible Angel winning condition ϕ, $S(a) \models \langle a{:}\alpha \between S\rangle\, \phi$.*

Proof. Follows from Lemma F.6 because an inductive subvalue map is a subvalue map (Theorem 6.4) and $S(a) \rightarrow \langle\alpha\rangle\, \phi$ holds per the definition of a subvalue map. □

Lemma 4.4 (Game Suffixes are Operational Gameplay Suffixes). *For subgame $b{:}\beta$ in overall game $a{:}\alpha$, let the game suffix be $a{:}\alpha_{b:}$. Let σ be some state. The states reachable from playing $a{:}\alpha$ such that σ is reached at subgame $b{:}\beta$ at some point during the gameplay are the same as the states reachable from playing $a{:}\alpha_{b:}$ starting in σ. That is,*

$$\{\pi_{\neg\mathfrak{S}}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma'), \lfloor p \rfloor_{\sigma'} = \sigma \text{ and } \pi_{\neg\mathfrak{S}}(s) \neq ()\} = \pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha_{b:}, \sigma)),$$

where $()$ is the empty path.

Proof. We prove

$$\{\pi_{\neg\mathfrak{S}}(s) \;:\; p\cdot b\cdot s\in\mathcal{T}(a{:}\alpha,\sigma'),\lfloor p\rfloor_{\sigma'}=\sigma \text{ and } \pi_{\neg\mathfrak{S}}(s)\neq()\}=\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha_{b:},\sigma))$$

by induction on the structure of $a{:}\alpha$, and case analysis.

- When $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha\in\{x:=e,x:=\otimes,?Q,!Q,\{x'=f(x)\;\&\;Q\}^d\}$, then the only subgame b in subgames$(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b=a$. $a{:}\alpha_{a:}$ is the game $a{:}\alpha$. $\mathcal{T}(a{:}\alpha_{a:},\sigma)$ is then $\mathcal{T}(a{:}\alpha,\sigma)$.
  The left-hand side, $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$, is $\mathcal{T}(\alpha,\sigma)$, the label-free game tree of the original game (see Def. G.3). Per the definition of labeled game trees for any atomic subgame α (Def. G.1), $\mathcal{T}(a{:}\alpha,\sigma)$ is $a\cdot t\;:\;t\in\mathcal{T}(\alpha,\sigma)$. Thus, the right-hand side is also $\mathcal{T}(\alpha,\sigma)$.
- Extending the previous case to compositional subgames, when $b=a$ and $a{:}\alpha$ is not a loop, then a similar argument holds.
  The *left-hand side* is $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$.
  The *right-hand side* is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot a\cdot s\in\mathcal{T}(a{:}\alpha,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$. Per the definition of (non-loop) labeled game trees, the only occurrence of label a is at the beginning of all plays and thus the right-hand side is equal to $\{\pi_{\neg\mathfrak{S}}(s)\;:\;a\cdot s\in\mathcal{T}(a{:}\alpha,\sigma),\pi_{\neg\mathfrak{S}}(s)\neq()\}$. This is just $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$.
- When $a{:}\alpha$ is a loop, i.e., $a{:}(g{:}\gamma)^*$ or $a{:}(g{:}\gamma)^\times$, and $b\neq a$, then $a{:}\alpha_{b:}=(g{:}\gamma_{b:})\,;\,a{:}\alpha$.
  The *left-hand side* is then $\pi_{\neg\mathfrak{S}}(\mathcal{T}(g{:}\gamma_{b:}\,;\,a{:}\alpha,\sigma))$.
  Per the inductive hypothesis, $\pi_{\neg\mathfrak{S}}(\mathcal{T}(g{:}\gamma_{b:},\sigma))$ is equal to $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p{\cdot}b{\cdot}s\in\mathcal{T}(g{:}\gamma,\sigma')\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$. Thus, per the definition of labeled game trees for sequential composition games, the left-hand side is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot b\cdot s\in\mathcal{T}(g{:}\gamma,\sigma')\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}\cup\{\pi_{\neg\mathfrak{S}}(s\cdot t)\;:\;p\cdot b\cdot s\in\text{leaf}(\mathcal{T}(g{:}\gamma,\sigma')),\lfloor p\rfloor_{\sigma'}=\sigma,\lfloor s\rfloor_{\sigma}=\sigma''\text{ and }t\in\mathcal{T}(a{:}\alpha,\sigma'')\}$, where leaf returns the set of leaf nodes in a tree as discussed in Appendix G.1.
  The *right-hand side* is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot b\cdot s\in\mathcal{T}(a{:}\alpha,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$. Per the definition of labeled game trees for loops, we can break this down by unrolling the loop: $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p{\cdot}b{\cdot}s\in\mathcal{T}(g{:}\gamma,\sigma')\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}\cup\{\pi_{\neg\mathfrak{S}}(s{\cdot}t)\;:\;p{\cdot}b{\cdot}s\in\text{leaf}(\mathcal{T}(g{:}\gamma,\sigma')),\lfloor p\rfloor_{\sigma'}=\sigma,\lfloor s\rfloor_{\sigma}=\sigma''\text{ and }t\in\mathcal{T}(a{:}\alpha,\sigma'')\}$.
- Otherwise, when $b=a$ and $a{:}\alpha$ is a loop, i.e., $a{:}(g{:}\gamma)^*$ or $a{:}(g{:}\gamma)^\times$, then $a{:}\alpha_{a:}=a{:}\alpha$.
  The *left-hand side* is $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$.
  The *right-hand side* is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot a\cdot s\in\mathcal{T}(a{:}\alpha,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$. By the choice of $\sigma'=\sigma$, $p=()$ and the definition of labeled game trees, we can show that this is a superset of $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$. Moreover, by the definition of labeled game trees for loops, we know that any suffix s that follows label a must be generated by the loop fixed point function f from Def. G.1 for initial state σ. So, $a\cdot s$ must be within the game tree $\mathcal{T}(a{:}\alpha,\sigma)$. Therefore, the right-hand side is also a subset of $\pi_{\neg\mathfrak{S}}(\mathcal{T}(a{:}\alpha,\sigma))$. Thus, the left-hand side and right-hand side must be equal.
- When $a{:}\alpha$ is a choice, i.e., $a{:}(g{:}\gamma\cup d{:}\delta)$ or $a{:}(g{:}\gamma\cap d{:}\delta)$, and $b\neq a$, then there are two cases.
  (1) If $b\in$ subgames$(g{:}\gamma)$, then $a{:}\alpha_{b:}=g{:}\gamma_{b:}$.
  The *left-hand side* is $\pi_{\neg\mathfrak{S}}(\mathcal{T}(g{:}\gamma_{b:},\sigma))$. Per the inductive hypothesis, this is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot b\cdot s\in\mathcal{T}(g{:}\gamma,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$.
  The *right-hand side* is $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot b\cdot s\in\mathcal{T}(a{:}\alpha,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$. Per the definition of labeled game trees for choices, $\mathcal{T}(a{:}\alpha,\sigma')$ is the union of the game trees of $g{:}\gamma$ and $d{:}\delta$. However, b does not appear in $d{:}\delta$. Thus the right-hand side can be simplified to $\{\pi_{\neg\mathfrak{S}}(s)\;:\;p\cdot b\cdot s\in\mathcal{T}(g{:}\gamma,\sigma'),\pi_{\neg\mathfrak{S}}(s)\neq()\text{ and }\lfloor p\rfloor_{\sigma'}=\sigma\}$.
  (2) If $b\in$ subgames$(d{:}\delta)$, then $a{:}\alpha_{b:}=d{:}\delta_{b:}$, and a symmetric argument applies, with $g{:}\gamma$ and $d{:}\delta$ swapped.
- When $a{:}\alpha$ is a sequential composition, i.e., $a{:}(g{:}\gamma\,;\,d{:}\delta)$, and $b\neq a$, then there are two cases.

(1) If $b \in \text{subgames}(g{:}\gamma)$, then $a{:}\alpha_{b:} = (g{:}\gamma_{b:})\,;\, d{:}\delta$. The argument is similar to the loop case.
The *left-hand side* is $\pi_{\neg\Subset}(\mathcal{T}((g{:}\gamma_{b:})\,;\, d{:}\delta, \sigma))$. As per the inductive hypothesis, $\pi_{\neg\Subset}(\mathcal{T}(g{:}\gamma_{b:}, \sigma))$ is equal to $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma'), \pi_{\neg\Subset}(s) \neq () \text{ and } \lfloor p \rfloor_{\sigma'} = \sigma\}$. Thus, per the definition of labeled game trees for sequential composition games, the left-hand side is $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma'), \lfloor p \rfloor_{\sigma'} = \sigma, \pi_{\neg\Subset}(s) \neq ()\} \cup \{\pi_{\neg\Subset}(s \cdot t) \;:\; p \cdot b \cdot s \in \text{leaf}(\mathcal{T}(g{:}\gamma, \sigma')), t \in \mathcal{T}(d{:}\delta, \sigma''), \lfloor p \rfloor_{\sigma'} = \sigma, \pi_{\neg\Subset}(s \cdot t) \neq () \text{ and } \lfloor s \rfloor_{\sigma} = \sigma''\}$.
The *right-hand side* is $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma'), \pi_{\neg\Subset}(s) \neq () \text{ and } \lfloor p \rfloor_{\sigma'} = \sigma\}$. Per the definition of labeled game trees for sequential composition, since $b \notin \text{subgames}(d{:}\delta)$, this is: $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma'), \lfloor p \rfloor_{\sigma'} = \sigma, \pi_{\neg\Subset}(s) \neq ()\} \cup \{\pi_{\neg\Subset}(s \cdot t) \;:\; p \cdot b \cdot s \in \text{leaf}(\mathcal{T}(g{:}\gamma, \sigma')), t \in \mathcal{T}(d{:}\delta, \sigma''), \lfloor p \rfloor_{\sigma'} = \sigma, \pi_{\neg\Subset}(s \cdot t) \neq () \text{ and } \lfloor s \rfloor_{\sigma} = \sigma''\}$.

(2) If $b \in \text{subgames}(d{:}\delta)$, then $a{:}\alpha_{b:} = d{:}\delta_{b:}$.
The *left-hand side* is $\pi_{\neg\Subset}(\mathcal{T}(d{:}\delta_{b:}, \sigma))$. Per the inductive hypothesis, this is $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(d{:}\delta, \sigma'), \pi_{\neg\Subset}(s) \neq () \text{ and } \lfloor p \rfloor_{\sigma'} = \sigma\}$.
The *right-hand side* is $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma'), \pi_{\neg\Subset}(s) \neq () \text{ and } \lfloor p \rfloor_{\sigma'} = \sigma\}$. Per the definition of labeled game trees for sequential composition, using the fact that $b \notin \text{subgames}(g{:}\gamma)$, this can be simplified to $\{\pi_{\neg\Subset}(s) \;:\; p \cdot b \cdot s \in \mathcal{T}(d{:}\delta, \sigma') \text{ and } \lfloor p \rfloor_{\sigma'} = \sigma\}$.

□

Lemma 4.5 (Execution Prefixes are Original Gameplay Prefixes). *For subgame $b{:}\beta$ in $a{:}\alpha$, for any initial state σ, the states reachable at $b{:}\beta$ while playing $a{:}\alpha$ are equal to those reachable by playing game prefix $a{:}\alpha_{:b}$. That is, prefix set $\{\pi_{\neg\Subset}(p) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma)\}$ is equal, under prefix closure, to prefix game tree $\pi_{\neg\Subset}(\mathcal{T}(a{:}\alpha_{:b}, \sigma))$*[11].

Proof. We show that

$$\pi_{\neg\Subset}(\mathcal{T}(a{:}\alpha_{:b}, \sigma)) = \{\pi_{\neg\Subset}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma) \text{ and } p = t \cdot u\}$$

Similar to Lemma 4.4, we perform induction on the structure of $a{:}\alpha$ and case analysis.

- When $b = a$ and $a{:}\alpha$ is not a loop, then $a{:}\alpha_{:a}$ is the empty game skip.
The *left-hand side* $\pi_{\neg\Subset}(\mathcal{T}(a{:}\alpha_{:a}, \sigma))$ is then $\emptyset$.
Indeed, on the *right-hand side*, all plays in $\mathcal{T}(a{:}\alpha, \sigma)$ are of the form $a \cdot p$ where p does not contain a, so the condition of the theorem holds.
- When $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then the only subgame b in $\text{subgames}(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b = a$, handled already by the first case.
- When $a{:}\alpha$ is a choice, i.e., $a{:}(g{:}\gamma \cup d{:}\delta)$ or $a{:}(g{:}\gamma \cap d{:}\delta)$, then we analyze cases depending on the value of $b \neq a$.
  (1) If b is in $g{:}\gamma$, then $a{:}\alpha_{:b} = g{:}\gamma_{:b}$.
  The *left-hand side* is $\pi_{\neg\Subset}(\mathcal{T}(g{:}\gamma_{:b}, \sigma))$. Per the inductive hypothesis, this consists of
  $$\{\pi_{\neg\Subset}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma), p = t \cdot u, \pi_{\neg\Subset}(t) \neq ()\}.$$
  The *right-hand side* is $\{\pi_{\neg\Subset}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Subset}(t) \neq ()\}$. Per the definition of game trees for choices, since $\mathcal{T}(a{:}\alpha, \sigma)$ is the union of the game trees for $g{:}\gamma$ and $d{:}\delta$, and b does not appear in $d{:}\delta$, this simplifies to
  $$\{\pi_{\neg\Subset}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma), p = t \cdot u, \pi_{\neg\Subset}(t) \neq ()\}.$$

[11] The prefix closure is required to recover tree structure. Unlike with the suffix construction, the prefix set builder collects only *parent* subgames of $b{:}\beta$ and *not all predecessors*. Expanding the prefix closure results in the following condition:

$$\{\pi_{\neg\Subset}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma) \text{ and } p = t \cdot u\} = \pi_{\neg\Subset}(\mathcal{T}(a{:}\alpha_{:b}, \sigma))$$

(2) If b is in $d{:}\delta$, then the argument is symmetric to the previous case, but $g{:}\gamma$ swapped with $d{:}\delta$.

- When $a{:}\alpha$ is a sequential composition, i.e., $a{:}(g{:}\gamma\,;\,d{:}\delta)$, then we analyze cases depending on the value of $b \neq a$.

(1) If b is in $g{:}\gamma$, then $a{:}\alpha_{:b} = g{:}\gamma_{:b}$.
The *left-hand side* is $\pi_{\neg\Leftarrow}(\mathcal{T}(g{:}\gamma_{:b}, \sigma))$. Per the inductive hypothesis, this is

$$\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}.$$

The *right-hand side* is $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$. Since $b \notin \text{subgames}(d{:}\delta)$, using the definition of labeled game trees for sequential composition, this can be simplified to

$$\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}.$$

(2) If b is in $d{:}\delta$, then $a{:}\alpha_{:b} = g{:}\gamma\,;\,d{:}\delta_{:b}$.
The *left-hand side* is $\pi_{\neg\Leftarrow}(\mathcal{T}(g{:}\gamma\,;\,d{:}\delta_{:b}, \sigma))$. As per the inductive hypothesis, $\pi_{\neg\Leftarrow}(\mathcal{T}(d{:}\delta_{:b}, \sigma))$ is equal to $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(d{:}\delta, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$. Thus, per the definition of labeled game trees for sequential composition games, the left-hand side is

$$\pi_{\neg\Leftarrow}(\mathcal{T}(g{:}\gamma, \sigma)) \cup \{\pi_{\neg\Leftarrow}(q \cdot t) \;:\; q \in \mathsf{leaf}(\mathcal{T}(g{:}\gamma, \sigma)), \lfloor q \rfloor_\sigma = \sigma', p \cdot b \cdot s \in \mathcal{T}(d{:}\delta, \sigma'), p = t \cdot u\}$$

where leaf indicates a leaf node a game tree, as indicated in Appendix G.1.
The *right-hand side* is $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma)\}, p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()$. Per the definition of labeled game trees for sequential composition, since $b \notin \text{subgames}(g{:}\gamma)$, this can be written as

$$\pi_{\neg\Leftarrow}(\mathcal{T}(g{:}\gamma, \sigma)) \cup \{\pi_{\neg\Leftarrow}(q \cdot t) \;:\; q \in \mathsf{leaf}(\mathcal{T}(g{:}\gamma, \sigma)), \lfloor q \rfloor_\sigma = \sigma', p \cdot b \cdot s \in \mathcal{T}(d{:}\delta, \sigma'), p = t \cdot u\}.$$

- When $a{:}\alpha$ is a loop, i.e., $a{:}(g{:}\gamma)^*$ or $a{:}(g{:}\gamma)^\times$, then we analyze cases depending on the value of b.

(1) If $b = a$, then $a{:}\alpha_{:b} = a{:}\alpha$.
The *left-hand side* is then $\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha, \sigma))$.
The *right-hand side* is $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot a \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$. By the definition of labeled game trees (Def. G.1), all plays in $\mathcal{T}(a{:}\alpha, \sigma)$ end with $a \cdot \mathfrak{s}^d$ or $a \cdot \mathfrak{s}$. Thus, from the elements where play s is $\mathfrak{s}$ or $\mathfrak{s}^d$, $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot a \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$ is a superset of $\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha, \sigma))$. Moreover, as labeled game trees are prefix-closed, $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot a \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$ is also a subset of $\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha, \sigma))$. Thus, the left-hand side and right-hand side must be equal.

(2) If $b \neq a$, then $a{:}\alpha_{:b} = (a{:}\alpha\,;\,g{:}\gamma_{:b})$.
The *left-hand side* is $\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha\,;\,(g{:}\gamma_{:b}), \sigma))$. As per the inductive hypothesis, $\pi_{\neg\Leftarrow}(\mathcal{T}(g{:}\gamma_{:b}, \sigma))$ is equal to $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$. Thus, per the definition of labeled game trees for sequential composition games, the left-hand side is

$$\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha, \sigma)) \cup \{\pi_{\neg\Leftarrow}(q \cdot t) \;:\; q \in \mathsf{leaf}(\mathcal{T}(a{:}\alpha, \sigma)), \lfloor q \rfloor_\sigma = \sigma', p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma'), p = t \cdot u\}.$$

The *right-hand side* is $\{\pi_{\neg\Leftarrow}(t) \;:\; p \cdot b \cdot s \in \mathcal{T}(a{:}\alpha, \sigma), p = t \cdot u, \pi_{\neg\Leftarrow}(t) \neq ()\}$. This is a subset of $\pi_{\neg\Leftarrow}(\mathcal{T}(a{:}\alpha, \sigma))$, and therefore the left-hand side, because game trees are prefix-closed. Next we show it is also a superset. Consider any play in the left-hand side set. For Angelic loops, it must be a prefix of some play of the form $q \cdot p$ where $q \in \mathsf{leaf}(f^n(a\hat{}\mathfrak{s}, a\hat{}\mathfrak{g}))$ for some n, $\lfloor q \rfloor_\sigma = \sigma'$, and $p \cdot b \cdot s \in \mathcal{T}(g{:}\gamma, \sigma')$ where f and $\hat{}$ are from Def. G.1. Such a play and all its prefixes feature in the right-hand side set because they are in $f^{n+1}(a\hat{}\mathfrak{s}, a\hat{}\mathfrak{g})$ which is in $\mathcal{T}(a{:}\alpha, \sigma)$. For Demonic loops, the same argument applies but for $f^n(a\hat{}\mathfrak{s}^d, a\hat{}\mathfrak{g}^d)$ as defined in Def. G.1.

□

Lemma 5.5 (Existential projection correspondence). *For any Angelic subvalue map S for game $a{:}\alpha$ and compatible winning condition ϕ, in initial state $\sigma \models \langle\alpha\rangle\,\phi$, Angel has a winning strategy for the game $a{:}\alpha\exists\!\downarrow\!S$ if and only if Angel can win $\langle\alpha\rangle\,\phi$ while following the policy of S. That is, $\sigma \models \langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ iff $\mathcal{S}_{a:\alpha}(S)(\sigma) \neq \emptyset$.*

Proof. We use structural induction on the game $a{:}\alpha$. The cases for $a{:}\alpha$ are as follows.

- When α is atomic and not controlled by Angel, that is, $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then:
  $\langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ is
  $\langle\alpha\rangle\,\phi$ because $a{:}\alpha\exists\!\downarrow\!S = \alpha$,
  which holds because by assumption $\sigma \models \langle\alpha\rangle\,\phi$.
  The corresponding strategy sets are all nonempty and always contain exactly one strategy per Def. G.6 as Angel has no decisions to make and the Angelic strategy is just the full game tree.
- If $a{:}\alpha$ is a non-deterministic Angel assignment $a{:}x := *$, then $\mathcal{S}_{a:\alpha}(S)(\sigma)$ has the strategy set $\{\{(x := e) \mid \sigma(x \mapsto e) \models S(\text{end})\}\}$.
  $\sigma \models \langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ iff
  $\sigma \models \langle x := *\,;\, ?S(\text{end})\rangle\,\phi$ because $a{:}\alpha\exists\!\downarrow\!S$ is $x := *\,;\, ?S(\text{end})$
  iff $\sigma \models \langle x := *\rangle\,(S(\text{end}) \wedge \phi)$ applying axiom $\langle?\rangle$
  iff $\sigma \models \langle x := *\rangle\,(S(\text{end}))$ $S(\text{end}) \wedge \phi \leftrightarrow S(\text{end})$ as ϕ is a compatible condition
  iff $\exists e\ \sigma(x \mapsto e) \models S(\text{end})$ by semantics of dGL
  which is exactly the condition for which the strategy set is nonempty.
- If $a{:}\alpha$ is an Angelic ODE $a{:}\{x' = f(x)\ \&\ Q\}$, then $\mathcal{S}_{a:\alpha}(S)(\sigma)$ has the strategy set $\{\{(x' = f(x)\ \&\ Q\ @\ t)\}\ :\ t \geq 0, \forall s \in [0,t], \varphi(s) \models S(b)$ and
  $\varphi(t) \models S(\text{succ}(b, a{:}\alpha))$ where $\varphi : [0,t] \to \mathcal{S}$ is differentiable,
  $\varphi(0) = \sigma$, and $\forall s \in [0,t]$, $\varphi(s) \models x' = f(x) \wedge Q\}$.
  $\sigma \models \langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ iff
  $\sigma \models \langle\{x' = f(x)\ \&\ Q\}\,;\, ?S(\text{end})\rangle\,\phi$ because $a{:}\alpha\exists\!\downarrow\!S$ is $\{x' = f(x)\ \&\ Q\}\,;\, ?S(\text{end})$, iff
  $\sigma \models \langle\{x' = f(x)\ \&\ Q\}\rangle\,(S(\text{end}) \wedge \phi)$ applying axiom $\langle?\rangle$
  iff $\sigma \models \langle\{x' = f(x)\ \&\ Q\}\rangle\,(S(\text{end}))$ $S(\text{end}) \wedge \phi \leftrightarrow S(\text{end})$ as ϕ is a compatible winning condition
  Which per the semantics of dGL is exactly when the strategy set is nonempty.
- If $a{:}\alpha$ is an Angelic choice $a{:}(g{:}\gamma \cup d{:}\delta)$, then $\mathcal{S}_{a:\alpha}(S)(\sigma)$ there are four possibilities.
  (1) If $\sigma \not\models S(g)$ and $\sigma \not\models S(d)$, then the strategy set is empty.
  Accordingly, Angel cannot win the existential projection game: $\sigma \models \langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ iff
  $\sigma \models \langle(?S(g)\,;\, g{:}\gamma) \cup (?S(d)\,;\, d{:}\delta)\rangle\,\phi$ because $a{:}\alpha\exists\!\downarrow\!S$ is $(?S(g)\,;\, g{:}\gamma) \cup (?S(d)\,;\, d{:}\delta)$
  iff $\sigma \models \langle(?S(g)\,;\, g{:}\gamma\exists\!\downarrow\!S)\rangle\,\phi \vee \langle(?S(d)\,;\, d{:}\delta\exists\!\downarrow\!S)\rangle\,\phi$ by axiom $\langle\cup\rangle$
  iff $\sigma \models (S(g) \wedge \langle g{:}\gamma\exists\!\downarrow\!S\rangle\,\phi) \vee (S(d) \wedge \langle d{:}\delta\exists\!\downarrow\!S\rangle\,\phi)$ by axiom $\langle?\rangle$, $\langle;\rangle$
  Which is false by assumption $\sigma \not\models S(g)$ and $\sigma \not\models S(d)$.
  (2) If $\sigma \models S(g)$ and $\sigma \not\models S(g)$, then the strategy set is $\{\{\mathrm{I}\hat{}t\}\ :\ t \in \mathcal{S}_{g:\gamma}(S)(\sigma)\}$, which is nonempty when $\mathcal{S}_{g:\gamma}(S)(\sigma)$ is nonempty. By the inductive hypothesis, $\mathcal{S}_{g:\gamma}(S)(\sigma)$ is nonempty when $\sigma \models \langle g{:}\gamma\exists\!\downarrow\!S\rangle\,\phi$.
  $\sigma \models \langle a{:}\alpha\exists\!\downarrow\!S\rangle\,\phi$ iff
  $\sigma \models \langle((?S(g)\,;\, g{:}\gamma) \cup (?S(d)\,;\, d{:}\delta))\rangle\,\phi$ because $a{:}\alpha\exists\!\downarrow\!S$ is $(?S(g)\,;\, g{:}\gamma) \cup (?S(d)\,;\, d{:}\delta)$
  iff $\sigma \models \langle(?S(g)\,;\, g{:}\gamma\exists\!\downarrow\!S)\rangle\,\phi \vee \langle(?S(d)\,;\, d{:}\delta\exists\!\downarrow\!S)\rangle\,\phi$ by axiom $\langle\cup\rangle$
  iff $\sigma \models \langle g{:}\gamma\exists\!\downarrow\!S\rangle\,\phi \vee (S(d) \wedge \langle d{:}\delta\exists\!\downarrow\!S\rangle\,\phi)$ $\sigma \models S(g)$, applying axioms $\langle?\rangle$, $\langle;\rangle$

iff $\sigma \models \langle g\gamma \overset{\exists}{\downarrow} S\rangle\, \phi$ $\quad \sigma \not\models S(g)$ by assumption
which is exactly the condition for which the strategy set is nonempty.

(3) If $\sigma \models S(d)$ and $\sigma \not\models S(g)$, the argument is symmetric but with the roles of γ and δ swapped.

(4) If $\sigma \models S(g)$ and $\sigma \models S(d)$, then the strategy set is $\{\{\mathfrak{l}\hat{}t\} : t \in \mathcal{S}_{g\gamma}(S)(\sigma)\} \cup \{\{\mathfrak{r}\hat{}t\} : t \in \mathcal{S}_{d\delta}(S)(\sigma)\}$. This set is nonempty when either $\mathcal{S}_{g\gamma}(S)(\sigma)$ or $\mathcal{S}_{d\delta}(S)(\sigma)$ is nonempty. By the inductive hypothesis, this is when $\sigma \models \langle g\gamma \overset{\exists}{\downarrow} S\rangle\, \phi$ or $\sigma \models \langle d\delta \overset{\exists}{\downarrow} S\rangle\, \phi$.
$\sigma \models \langle a\alpha \overset{\exists}{\downarrow} S\rangle\, \phi$ iff
$\sigma \models \langle ((?S(g)\,;\, g\gamma) \cup (?S(d)\,;\, d\delta))\rangle\, \phi$ $\quad$ because $a\alpha \overset{\exists}{\downarrow} S$ is $(?S(g)\,;\, g\gamma) \cup (?S(d)\,;\, d\delta)$
iff $\sigma \models \langle (?S(g)\,;\, g\gamma \overset{\exists}{\downarrow} S)\rangle\, \phi \vee \langle (?S(d)\,;\, d\delta \overset{\exists}{\downarrow} S)\rangle\, \phi$ $\quad$ by axiom $\langle\cup\rangle$
iff $\sigma \models \langle g\gamma \overset{\exists}{\downarrow} S\rangle\, \phi \vee \langle d\delta \overset{\exists}{\downarrow} S\rangle\, \phi$ $\quad \sigma \models S(g), \sigma \models S(d)$ which is exactly the condition for which the strategy set is nonempty.

- If $a\alpha$ is an Angelic loop $a(g\gamma)^*$. By the inductive hypothesis, $\mathcal{S}_{g\gamma}(S)(\sigma)$ is nonempty when $\sigma \models \langle g\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, S(a)$.
$\sigma \models \langle a\alpha \overset{\exists}{\downarrow} S\rangle\, \phi$ iff
$\sigma \models \langle a\alpha \overset{\exists}{\downarrow} S\rangle\, S(\mathsf{end})$ $\quad \phi$ is a compatible winning condition

We show that if there is a strategy in the subvalue map strategy set then there is a strategy to win $\langle (?S(g)\,;\, g\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a)))^*\rangle\, (S(\mathsf{end}))$ from σ and vice versa by induction over iterations.
First we build up a translation function i_Z inductively builds up translations for sequences (elements) in strategy set Z. It maps elements of a strategy for $\langle (?S(g)\,;\, g\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a)))^*\rangle\, (S(\mathsf{end}))$ to elements of a strategy in the subvalue map strategy set. For example, $i_{\{\mathfrak{g}\}}(\mathfrak{g}) = \mathfrak{g}$. i_Z is overloaded to also apply to strategies by applying to every element of the strategy.

$$
\begin{aligned}
i_{f(Z)} \equiv\ & \{\mathfrak{s}\hat{}(?S(\mathsf{end}))\} \mapsto \{\mathfrak{s}\}; \qquad \{\mathfrak{g}\} \mapsto \{\mathfrak{g}\};\\
& \text{for } t\cdot\mathfrak{g} \in \mathsf{leaf}(Z),\ (t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t) \mapsto (i_Z(t\hat{}\mathfrak{g})\hat{}h_{g\gamma}(u_t))\\
& \quad (t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t\hat{}\mathfrak{g}) \mapsto (i_Z(t\hat{}\mathfrak{g})\hat{}h_{g\gamma}(u_t)\hat{}\mathfrak{g})\\
& \quad (t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t\hat{}\mathfrak{s}\hat{}(?S(\mathsf{end}))) \mapsto (i_Z(t\hat{}\mathfrak{g})\hat{}h_{g\gamma}(u_t)\hat{}\mathfrak{s})\\
& \quad \text{where } u_t \text{ is a winning strategy for game } \langle g\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, S(a) \text{ and}\\
& \quad \text{corresponding translation function } h_{g\gamma}(u_t) \text{ for strategies of game}\\
& \quad \langle g\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a))\rangle\, S(a) \text{ to strategies induced by subvalue map}\\
& \quad S(\mathsf{end} \mapsto S(a)).
\end{aligned}
$$

Above, where for readability, the map implicitly respects prefix closure over *caret*, i.e., if $a\hat{}b \mapsto c\hat{}d$ then $a \mapsto c$ and $a\cdot b \mapsto c\cdot d$. The exception is for the extra $?S(g)$ and $?S(\mathsf{end})$ actions which have no corresponding mapped action. The mapping function $h_{g\gamma}(u_t)$ can be recursively defined to map a strategy winning the existential projection of a game onto a subvalue map to a strategy from the strategy set generated by that subvalue map where both strategies result in the same final state, though we do not show the full construction here.
Now we argue that the translated functions are amongst the subvalue map's induced strategies.

- As base case, if $\{\mathfrak{s}, \mathfrak{s}\cdot(?S(\mathsf{end}))\}$ wins the projection game from initial state σ then $\{\mathfrak{s}\}$ is in $\mathcal{S}_{a\alpha}(S)(\sigma)$. Otherwise if $\mathfrak{g}$ is the first step in the projection game's strategy, then $\sigma \models S(g)$ of the Angelic test that follows, $\{\mathfrak{g}\}$ can start strategies in $\mathcal{S}_{a\alpha}(S)(\sigma)$.

– For the recursive case, suppose that $Z \cup (\bigcup_{t\cdot\mathfrak{g}\in\mathsf{leaf}(Z)} t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t\hat{}\mathfrak{s}\hat{}(?S(\mathsf{end})))$ is a strategy that wins for the projection game.
Then $(i_Z(t\hat{}\mathfrak{g})\hat{}h_{g;\gamma}(u_t)\hat{}\mathfrak{s})$ should be in $\mathcal{S}_{a;\alpha}(S)(\sigma)$ for the original game. $i_Z(t\hat{}\mathfrak{g})$ can begin a strategy in $\mathcal{S}_{a;\alpha}(S)(\sigma)$ because of the inductive hypothesis of the inner induction on iterations. $h_{g;\gamma}(u_t)$ is some strategy for game $\langle g;\gamma\rangle^{\exists}_{\downarrow} S(\mathsf{end}\mapsto S(a))\rangle\, S(a)$ which must exist because of the inductive hypothesis from the outer proof. $\mathfrak{s}$ appears in $\mathcal{P}_{a;\alpha}(S)(\lfloor\sigma\rfloor_{t\hat{}\mathfrak{g}\hat{}h_{g;\gamma}(u_t)\hat{}\mathfrak{s}})$, appearing as a final action during strategy generation because in the original projection game, because in the existential projection it was in a state where the test $(?S(\mathsf{end}))$ is satisfied.
Otherwise, if $(Z\cup\bigcup_{t\cdot\mathfrak{g}\in\mathsf{leaf}(Z)} t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t)$ or $(Z\cup\bigcup_{t\cdot\mathfrak{g}\in\mathsf{leaf}(Z)} t\hat{}\mathfrak{g}\hat{}(?S(g))\hat{}u_t)$ appear in a winning strategy of the projection game and do not yet lose, then their translations can generate strategies in $\mathcal{S}_{a;\alpha}(S)(\sigma)$ as well because of a similar argument.

This establishes that if there is a way to win the projection game, then the subvalue map strategy set is nonempty.
For the other direction, a similar translation function can be written by inverting the construction above. A similar argument can be made to show that the translated strategies are winning for the projection game.

- If $a;\alpha$ is a Demonic loop $a;(g;\gamma)^*$, we use an approach similar to the Angelic loop case. By the inductive hypothesis, $\mathcal{S}_{g;\gamma}(S)(\sigma)$ is nonempty when $\sigma \models \langle g;\gamma\rangle^{\exists}_{\downarrow} S\rangle\, S(a)$.
$\sigma \models \langle a;\alpha\rangle^{\exists}_{\downarrow} S\rangle\, \phi$ iff
$\sigma \models \langle a;\alpha\rangle^{\exists}_{\downarrow} S\rangle\, S(\mathsf{end})$ $\quad\phi$ is a compatible winning condition

  We show that if there is a strategy in the set then there is a strategy to win $\sigma\langle(?S(g)\,;\, g;\gamma\rangle^{\exists}_{\downarrow} S(\mathsf{end}\mapsto S(a)))$ and vice versa by induction over iterations.
  We again build up a translation function i_Z.It inductively builds up translations for sequences (elements) in strategy set Z. It is similar to the previous case but with dual actions and without the added Angelic tests.

$$
\begin{aligned}
i_{f(Z)} \;\equiv\; & \{\mathfrak{s}^d\} \mapsto \{\mathfrak{s}^d\}; \qquad \{\mathfrak{g}^d\} \mapsto \{\mathfrak{g}^d\};\\
& \text{for } t\cdot\mathfrak{g}^d \in \mathsf{leaf}(Z),\ (t\hat{}\mathfrak{g}^d\hat{}u_t) \mapsto (i_Z(t\hat{}\mathfrak{g}^d)\hat{}h_{g;\gamma}(u_t))\\
& \quad (t\hat{}\mathfrak{g}^d\hat{}u_t\hat{}\mathfrak{g}^d) \mapsto (i_Z(t\hat{}\mathfrak{g}^d)\hat{}h_{g;\gamma}(u_t)\hat{}(?S(a))\hat{}\mathfrak{g}^d)\\
& \quad (t\hat{}\mathfrak{g}^d\hat{}u_t\hat{}\mathfrak{s}^d\hat{}(?S(\mathsf{end}))) \mapsto (i_Z(t\hat{}\mathfrak{g}^d)\hat{}h_{g;\gamma}(u_t)\hat{}\mathfrak{s}^d)\\
& \qquad \text{where } u_t \text{ is a winning strategy for game } \langle g;\gamma\rangle^{\exists}_{\downarrow} S(\mathsf{end}\mapsto S(a))\rangle\, S(a) \text{ and}\\
& \qquad \text{corresponding } h_{g;\gamma}(u_t) \text{ for game } \langle\gamma\rangle\, S(a) \text{ must exist per inductive hypothesis.}
\end{aligned}
$$

  Above, where for readability, the map implicitly respects prefix closure over *caret*. A full strategy is translated by translating each sequence.
  The argument that the translated function is in the strategy set is also similar to the previous case. As with the previous case, an inverse translation function shows the opposite direction.
- If $a;\alpha$ is a Demonic choice $a;(g;\gamma \cap d;\delta)$, then $\mathcal{S}_{a;\alpha}(S)(\sigma)$ is $\{\mathfrak{l}^d\hat{}t_l \cup \mathfrak{r}^d\hat{}t_r \;:\; t_l \in \mathcal{S}_{g;\gamma}(S)(g,\sigma) \text{ and } t_r \in \mathcal{S}_{d;\delta}(S)(d,\sigma)\}$, which is nonempty when both $\mathcal{S}_{g;\gamma}(S)(\sigma)$ and $\mathcal{S}_{d;\delta}(S)(\sigma)$ are nonempty. By the inductive hypothesis, this is when $\sigma \models \langle g;\gamma\rangle^{\exists}_{\downarrow} S\rangle\, \phi$ and $\sigma \models \langle d;\delta\rangle^{\exists}_{\downarrow} S\rangle\, \phi$.
$\sigma \models \langle a;\alpha\rangle^{\exists}_{\downarrow} S\rangle\, \phi$ iff
$\sigma \models \langle (g;\gamma\rangle^{\exists}_{\downarrow} S) \cap (d;\delta\rangle^{\exists}_{\downarrow} S)\rangle\, \phi$ $\quad$ by definition of $a;\alpha\rangle^{\exists}_{\downarrow} S$

iff $\sigma \models \langle g;\gamma \mathbin{\dot\cup} S\rangle\,\phi \wedge \langle d;\delta \mathbin{\dot\cup} S\rangle\,\phi$ by axiom $\langle\cap\rangle$ which is exactly the condition for which the strategy set is nonempty.

- If $a;\alpha$ is a sequence $a;(g;\gamma; d;\delta)$, then $\mathcal{S}_{a;\alpha}(S)(\sigma)$ is $\{t \cup \bigcup_{v\in\mathsf{leaf}(t)}(v\hat{}u) \;:\; t \in \mathcal{S}_{g;\gamma}(S(\mathsf{end} \mapsto S(d)))(g,\sigma)$ and $u \in \mathcal{S}_{d;\delta}(S)(d)(\lfloor v\rfloor_\sigma)\}$, which is nonempty when both $\mathcal{S}_{g;\gamma}(S)(\sigma)$ is nonempty and for some choice of v, $\mathcal{S}_{d;\delta}(S)(\lfloor v\rfloor_\sigma)$ is nonempty. By the inductive hypothesis, this is when $\sigma \models \langle g;\gamma \mathbin{\dot\cup} S(\mathsf{end} \mapsto S(d))\rangle\, S(d)$ and for some $v \in \mathcal{S}_{g;\gamma}(S)(\sigma)$, $\lfloor v\rfloor_\sigma \models \langle d;\delta \mathbin{\dot\cup} S\rangle\,\phi$.
  $\sigma \models \langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff
  $\sigma \models \langle (g;\gamma \mathbin{\dot\cup} S(\mathsf{end} \mapsto S(d)))\,;\,(d;\delta \mathbin{\dot\cup} S)\rangle\,\phi$ by definition of $a;\alpha \mathbin{\dot\cup} S$
  $\sigma \models \langle (g;\gamma \mathbin{\dot\cup} S(\mathsf{end} \mapsto S(d)))\rangle\,\langle (d;\delta \mathbin{\dot\cup} S)\rangle\,\phi$ by axiom $\langle;\rangle$
  which is matches the condition for which the strategy set is nonempty.

□

Lemma 5.7 (Universal projection correspondence). *For any Angelic subvalue map S for game $a;\alpha$, in initial state $\sigma \models \langle\alpha\rangle\, a$, for compatible winning condition ϕ, Angel has a winning strategy for the game $a;\alpha \mathbin{\dot\cup} S$ if and only if all ways to pursue the policy of S for game $a;\alpha$ that complete the game end in Angel's winning region. That is, $\sigma \models \langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff all elements of set $\mathcal{S}_{a;\alpha}(S)(\sigma)$ are Angel winning strategies.*

Proof. We show the proof by induction over the structure of $a;\alpha$.

- If $a;\alpha$ is atomic, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then $a;\alpha \mathbin{\dot\cup} S$ is α. $\sigma \models \langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff $\sigma \models \langle\alpha\rangle\,\phi$ by definition of $a;\alpha \mathbin{\dot\cup} S$. This is true because $\sigma \models \langle\alpha\rangle\,\phi$.
  The strategy set consists of a single strategy with no Angelic decisions, that is just the entire game tree. This strategy is a winning strategy because we know there is one winning strategy because by assumption $\sigma \models \langle\alpha\rangle\,\phi$, and this strategy is the only candidate.
- If $a;\alpha$ is a non-deterministic Angel assignment $a;x := *$, then $\mathcal{S}_{a;\alpha}(S)(\sigma)$ has the strategy set $\{\{(x := e) \mid \sigma(x \mapsto e) \models S(\mathsf{end})\}\}$, where all strategies are winning strategies when $\sigma(x \mapsto e) \models \phi \mid \sigma(x \mapsto e) \models S(\mathsf{end})$, which is always, since ϕ is a compatible winning condition so $S(\mathsf{end}) \to \phi$.
  $\sigma \models \langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff
  $\sigma \models \langle x := \otimes\,;\,!S(\mathsf{end})\rangle\,\phi$ by definition of $a;\alpha \mathbin{\dot\cup} S$
  iff $\sigma \models \langle x := \otimes\rangle\,(S(\mathsf{end}) \to \phi)$ by axioms $\langle;\rangle$, $[?]$
  iff $\sigma \models \langle x := \otimes\rangle\,\top$ $S(\mathsf{end}) \to \phi$ because ϕ is a compatible winning condition
  But this always holds by the semantics of dGL.
- If $a;\alpha$ is an Angelic ODE $a;\{x' = f(x)\ \&\ Q\}$, then $\mathcal{S}_{a;\alpha}(S)(\sigma)$ has the strategy set $\{\{(x' = f(x)\ \&\ Q\ @\ t)\} \;:\; t \geq 0, \forall s \in [0,t], \varphi(s) \models S(b)$ and $\varphi(t) \models S(\mathsf{succ}(b, a;\alpha))$ where $\varphi : [0,t] \to \mathcal{S}$ is differentiable, $\varphi(0) = \sigma$, and $\forall s \in [0,t]$, $\varphi(s) \models x' = f(x) \wedge Q\}$. All strategies are always winning strategies by a similar argument to for the previous case, since $S(\mathsf{end}) \to \phi$.
  $\sigma \models \langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff
  $\sigma \models \langle \{x' = f(x)\ \&\ Q\}^d\,;\,!S(\mathsf{end})\rangle\,\phi$ by definition of $a;\alpha \mathbin{\dot\cup} S$
  iff $\sigma \models \langle \{x' = f(x)\ \&\ Q\}^d\rangle\,(S(\mathsf{end}) \to \phi)$ by axioms $\langle;\rangle$, $[?]$
  iff $\sigma \models \langle \{x' = f(x)\ \&\ Q\}^d\rangle\,\top$ $S(\mathsf{end}) \to \phi$
  But this always holds by the semantics of dGL.
- If $a;\alpha$ is an Angelic choice $a;(g;\gamma \cup d;\delta)$, then
  $\langle a;\alpha \mathbin{\dot\cup} S\rangle\,\phi$ iff
  $\langle (!S(g)\,;\,g;\gamma \mathbin{\dot\cup} S) \cap (!S(d)\,;\,d;\delta \mathbin{\dot\cup} S)\rangle\,\phi$ by definition of $a;\alpha \mathbin{\dot\cup} S$
  iff $\sigma \models \langle !S(g)\,;\,g;\gamma \mathbin{\dot\cup} S\rangle\,\phi \wedge \langle !S(d)\,;\,d;\delta \mathbin{\dot\cup} S\rangle\,\phi$ by axiom $\langle\cap\rangle$

iff $\sigma \models (S(g) \rightarrow \langle g;\gamma \gtrdot S\rangle\, \phi) \wedge (S(d) \rightarrow \langle d;\delta \gtrdot S\rangle\, \phi)$ by axiom [?], $\langle;\rangle$

There are four cases:

- $\sigma \not\models S(g)$ and $\sigma \not\models S(d)$: then the Angel wins the projection game because of the vacuous premises of the implications, and the strategy set has only winning strategies vacuously since it has no strategies.
- $\sigma \models S(g)$ and $\sigma \not\models S(d)$: then the Angel wins the projection game iff
  $\sigma \models (S(g) \rightarrow \langle g;\gamma \gtrdot S\rangle\, \phi)$ second conjunct true as $\sigma \not\models S(d)$
  iff $\sigma \models \langle g;\gamma \gtrdot S\rangle\, \phi$ as $\sigma \models S(g)$

  By the inductive hypothesis, this is exactly when the strategy set $\mathcal{S}_{g;\gamma}(S)(\sigma)$ has only winning strategies for Angel in game $\langle\gamma\rangle\, \phi$.
  The strategy set for the overall game is $\{\{\Lsh \hat{t}\} : t \in \mathcal{S}_{g;\gamma}(S)(g,\sigma)\}$ which consists of winning strategies exactly when $\mathcal{S}_{g;\gamma}(S)(\sigma)$ has only winning strategies for Angel in game $\langle\gamma\rangle\, \phi$.
- $\sigma \not\models S(g)$ and $\sigma \models S(d)$, the argument is symmetric to the previous case, with the roles of g and d swapped.
- $\sigma \models S(g)$ and $\sigma \models S(d)$: then the Angel wins the projection game iff
  $\sigma \models \langle g;\gamma \gtrdot S\rangle\, \phi \wedge \langle d;\delta \gtrdot S\rangle\, \phi$ as $\sigma \models S(g)$ and $\sigma \models S(d)$
  By the inductive hypothesis, this is exactly when both strategy sets $\mathcal{S}_{g;\gamma}(S)(\sigma)$ and $\mathcal{S}_{d;\delta}(S)(\sigma)$ have only winning strategies for Angel in games $\langle\gamma\rangle\, \phi$ and $\langle\delta\rangle\, \phi$ respectively.
  The strategy set for the overall game is $\{\{\Lsh \hat{t}\} : t \in \mathcal{S}_{g;\gamma}(S)(g,\sigma)\} \cup \{\{\mathfrak{r} \hat{t}\} : t \in \mathcal{S}_{d;\sigma\delta}(S)(d)\}$.
  This consists of winning strategies exactly when both $\mathcal{S}_{g;\gamma}(S)(\sigma)$ and $\mathcal{S}_{d;\delta}(S)(\sigma)$ have only winning strategies for Angel in games $\langle\gamma\rangle\, \phi$ and $\langle\delta\rangle\, \phi$ respectively.

- If $a;\alpha$ is an Angelic loop $a;(g;\gamma)^*$:
  $\sigma \models \langle a;\alpha \gtrdot S\rangle\, \phi$ iff
  $\sigma \models \langle (!S(g)\,;\, (g;\gamma \gtrdot (S(\mathsf{end} \mapsto S(a)))))^{\times}\,;\, !S(\mathsf{end})\rangle\, \phi$ by definition of $a;\alpha \gtrdot S$
  iff $\sigma \models \langle (!S(g)\,;\, (g;\gamma \gtrdot (S(\mathsf{end} \mapsto S(a)))))^{\times}\rangle\, S(\mathsf{end}) \rightarrow \phi$ by axioms $\langle;\rangle$, [?]
  iff $\sigma \models \langle (!S(g)\,;\, (g;\gamma \gtrdot (S(\mathsf{end} \mapsto S(a)))))^{\times}\rangle\, \top \quad S(\mathsf{end}) \rightarrow \phi$
  But this always holds applying the loop rule with invariant $\langle\alpha\rangle\, \phi$ and using the inductive hypothesis on the subgame $\langle\gamma\rangle\, S(\mathsf{end} \mapsto S(a))$ and Theorem 5.9.
  The strategy set also consists of only winning strategies. Consider any leaf t in any strategy tree in the set. If t ends with $\mathfrak{s}$ then by construction (Def. G.6), $\lfloor t \rfloor_\sigma \models S(\mathsf{end})$ and t is a winning play for Angel. Otherwise, if t does not end in $\mathfrak{s}$, then it must have ended while playing loop body γ. Let $t = u \cdot v$ where v plays the final (incomplete) iteration of the loop body. v ends with Demon getting stuck because it is a winning strategy for γ by the inductive hypothesis in combination with Theorem 5.9. Thus, t is also a winning strategy for Angel in the larger game.
- If $a;\alpha$ is a Demonic loop $a;(g;\gamma)^{\times}$, then
  $\sigma \models \langle a;\alpha \gtrdot S\rangle\, \phi$ iff
  $\sigma \models \langle (g;\gamma \gtrdot S(\mathsf{end} \mapsto S(a)))^{\times}\rangle\, \phi$
  This always holds: we can apply the loop invariant rule with invariant $\langle\alpha\rangle\, \phi$ and use Theorem 5.9 on the subgame $\langle\gamma\rangle\, S(\mathsf{end} \mapsto S(a))$.
  The argument that the strategy set has only winning strategies is similar to the previous case. Consider any leaf in any strategy tree in the set. If if ends with $\mathfrak{s}$, it is a winning play for Angel because $\lfloor t \rfloor_\sigma \models S(\mathsf{end})$. Otherwise, if it does not end in $\mathfrak{s}$, then it must have ended while playing loop body γ. It must end with Demon getting stuck because by the inductive

hypothesis in combination with Theorem 5.9, and so must be a winning strategy for Angel in the larger game.

- If $a{:}\alpha$ is a Demonic choice $a{:}(g{:}\gamma \cap d{:}\delta)$, then $\mathcal{S}_{a:\alpha}(S)(\sigma)$ has the strategy set $\{\mathrm{I}^{d\hat{}}t_l \cup \mathfrak{r}^{d\hat{}}t_r \,:\, t_l \in \mathcal{S}_{g:\gamma}(S)(g,\sigma)$ and $t_r \in \mathcal{S}_{d:\delta}(S)(d,\sigma)\}$, which contains only winning strategies exactly when $\mathcal{S}_{g:\gamma}(S)(\sigma)$ and $\mathcal{S}_{d:\delta}(S)(\sigma)$ have only winning strategies for Angel in games $\langle\gamma\rangle\,\phi$ and $\langle\delta\rangle\,\phi$ respectively.
  $\sigma \models \langle a{:}\alpha \between S\rangle\,\phi$ iff
  $\sigma \models \langle (g{:}\gamma \between S) \cap (d{:}\delta \between S)\rangle\,\phi$ by definition of $a{:}\alpha \between S$
  iff $\sigma \models \langle g{:}\gamma \between S\rangle\,\phi \wedge \langle d{:}\delta \between S\rangle\,\phi$ by axiom $\langle\cap\rangle$
  which by the inductive hypothesis applied for $g{:}\gamma$ and $d{:}\delta$ is exactly when the overall strategy set has only winning Angel strategies.
- If $a{:}\alpha$ is sequential composition $a{:}(g{:}\gamma; d{:}\delta)$, then $\mathcal{S}_{a:\alpha}(S)(\sigma)$ has the strategy set $\{t \cup \bigcup_{v\in\mathrm{leaf}(t)}(v\hat{}u) \,:\, t \in \mathcal{S}_{g:\gamma}(S(\mathsf{end} \mapsto S(d)))(g,\sigma)$ and $u \in \mathcal{S}_{d:\delta}(S)(d)(\lfloor v\rfloor_\sigma)\}$. These are always winning strategies: $\mathcal{S}_{g:\gamma}(S(\mathsf{end} \mapsto S(d)))(g,\sigma)$ and $\mathcal{S}_{d:\delta}(S)(d)(\lfloor v\rfloor_\sigma)$ have only winning strategies using the inductive hypothesis along with Theorem 5.9. All plays that get stuck win for Angel, and all plays that run till the end result in a state $\sigma' \in S(\mathsf{end})$, and so win for Angel because $S(\mathsf{end}) \to \phi$.
  $\sigma \models \langle a{:}\alpha \between S\rangle\,\phi$ always holds by Theorem 5.9.

□

Theorem 5.9 (Subvalue Map stays in Winning Region). *Suppose S is an Angelic subvalue map for game $a{:}\alpha$ compatible with winning condition ϕ, i.e., for every subgame $b \in \mathsf{subgames}(a{:}\alpha)$, $\models S(b) \to \langle a{:}\alpha_{b:}\rangle\, S(\mathsf{end})$ and $\models S(\mathsf{end}) \to \phi$. Upon starting in any state $\sigma \in [\![S(a)]\!]$, and reaching subgame b by following any strategy induced by the subvalue map S, there exists a winning strategy for Angel to win the remainder of the game. That is, $S(a) \models \langle (a{:}\alpha \between S)_{:b}\rangle\, \langle a{:}\alpha_{b:}\rangle\,\phi$.*

Proof. We perform induction on the structure of $a{:}\alpha$.

- When $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then The only subgame b in $\mathsf{subgames}(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b = a$. $a{:}\alpha_{:a}$ is the empty game skip, and $a{:}\alpha_{a:}$ is the game $a{:}\alpha$, and $a{:}\alpha \between S$ is just $a{:}\alpha$. Thus, $\langle a{:}\alpha_{:a} \between S\rangle\, \langle a{:}\alpha_{a:}\rangle\,\phi$ is equivalent to $\langle\mathsf{skip}\rangle\, \langle\alpha\rangle\,\phi$. We need to show that $S(a) \models \langle a{:}\alpha\rangle\,\phi$ which holds because by definition of an Angelic subvalue map, $\models S(a) \to \langle\alpha\rangle\,\phi$.
- When $a{:}\alpha$ is an Angelic nondeterministic assignment, i.e., $a{:}x := *$, then again the only subgame b in $\mathsf{subgames}(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b = a$. $a{:}\alpha_{:a}$ is the empty game skip, and $a{:}\alpha \between S$ is just $a{:}\alpha\,;\, ?S(\mathsf{end})$, and $a{:}\alpha\,;\, ?S(\mathsf{end})_{a:}$ is the game $a{:}\alpha\,;\, ?S(\mathsf{end})$. Thus, $\langle a{:}\alpha_{:a} \between S\rangle\, \langle a{:}\alpha_{a:}\rangle\,\phi$ is equivalent to $\langle\mathsf{skip}\rangle\, \langle\alpha\,;\, ?S(\mathsf{end})\rangle\,\phi$. We need to show that $S(a) \models \langle\alpha\rangle\, (?S(\mathsf{end}) \wedge \phi)$. Since ϕ is a compatible winning condition, this is $S(a) \models \langle\alpha\rangle\, ?S(\mathsf{end})$ which holds by the definition of an inductive Angelic subvalue map.
- When $a{:}\alpha$ is an Angelic ODE, i.e., $a{:}\{x' = f(x)\ \&\ Q\}$, then the argument is similar to the previous case. As before the only possibility is $b = a$, and we need to show that $S(a) \models \langle\alpha\rangle\, ?S(\mathsf{end}) \wedge \phi$. Since ϕ is a compatible winning condition, this is $S(a) \models \langle a{:}\alpha\rangle\, ?S(\mathsf{end})$ which holds by the definition of an inductive Angelic subvalue map.
- When $a{:}\alpha$ is an Angelic choice, $a{:}(g{:}\gamma \cup d{:}\delta)$, then we analyze cases depending on the value of b.
  (1) If $b = a$, then $a{:}\alpha_{:a} \between S_{:b} = \mathsf{skip}$, and $a{:}\alpha_{b:} = a{:}\alpha$. Thus, $\langle a{:}\alpha_{:b} \between S\rangle\, \langle a{:}\alpha_{b:}\rangle\,\phi$ is equivalent to $\langle a{:}\alpha\rangle\,\phi$. $S(a) \models \langle a{:}\alpha\rangle\,\phi$ This can be proved by applying the axioms $\to$R along with $S(a) \to \langle a{:}\alpha\rangle\,\phi$ from the definition of an Angelic subvalue map.

(2) Otherwise, if $b \in \text{subgames}(g{:}\gamma)$, then $a{:}\alpha_{:a} \between S_{:b} = !S(g)\,;\, g{:}\gamma \between S_{:b}$ and $a{:}\alpha_{b:} = g{:}\gamma_{b:}$. We can show that $S(a) \models \langle !S(g)\,;\, g{:}\gamma \between S_{:b}\rangle \langle g{:}\gamma_{b:}\rangle \phi$ by first applying axioms $\langle;\rangle$, $[?]$, $[\cdot]$, $\langle^d\rangle$, $\to$R to get the goal $S(a), S(g) \models \langle g{:}\gamma \between S_{:b}\rangle \langle g{:}\gamma_{b:}\rangle \phi$ Notice that S is also an inductive Angelic subvalue map for $g{:}\gamma$ and winning condition ϕ. Thus by the inductive hypothesis, $S(g) \models \langle g{:}\gamma_{:b} \between S\rangle \langle g{:}\gamma_{b:}\rangle \phi$ completing the proof.

(3) Otherwise, if $b \in \text{subgames}(d{:}\delta)$, then the proof proceeds similarly to the previous case, but with $d{:}\delta$ replacing $g{:}\gamma$.

- When $a{:}\alpha$ is an Angelic loop, i.e., $a{:}(g{:}\gamma)^*$, then we analyze cases depending on the value of b.

(1) If $b = a$, then $a{:}\alpha_{b:} = a{:}\alpha$ and $a{:}\alpha \between S_{:b} = (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(\text{end})$. Thus,

$$\langle a{:}\alpha \between S_{:b}\rangle \langle a{:}\alpha_{b:}\rangle \phi$$

is by definition

$$\langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(\text{end})\rangle \langle a{:}\alpha\rangle \phi.$$

By the inductive hypothesis, $S(g) \models \langle g{:}\gamma \between S(\text{end} \mapsto S(a))\rangle S(a)$. Then applying loop with invariant $S(a)$ we have

$$S(a) \models \langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\rangle S(a).$$

By the definition of an Angelic subvalue map, $\models S(a) \to \langle a{:}\alpha\rangle \phi$. Thus by M, $\to$R, and $\langle?\rangle$, we prove the desired goal

$$S(a) \models \langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(\text{end})\rangle \langle a{:}\alpha\rangle \phi.$$

(2) Otherwise, if $b \in \text{subgames}(g{:}\gamma)$, then
$a{:}\alpha \between S_{:b} = (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(g)\,;\, (g{:}\gamma \between S(\text{end} \mapsto S(a)))_{:b}$,
$a{:}\alpha_{b:} = g{:}\gamma_{b:}\,;\, a{:}\alpha$.
Want to show:

$$S(a) \quad \models \quad \langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(g)\,;\, (g{:}\gamma \between S(\text{end} \mapsto S(a)))_{:b}\rangle \langle (g{:}\gamma)_{b:}\,;\, a{:}\alpha\rangle \phi.$$

First, by Lemma F.6, we have $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\,;\, !S(\text{end})\rangle \top$, so by axioms $[?]$ and $\langle;\rangle$, $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times\rangle \top$.
Next, by the inductive hypothesis and applications of axioms $\langle;\rangle$, $\langle?\rangle$ and $\to$R, we have

$$\top \models \langle !S(g)\,;\, (g{:}\gamma \between S(\text{end} \mapsto S(a)))_{:b}\rangle \langle g{:}\gamma_{b:}\rangle S(a)$$

Finally, since S is an Angelic subvalue map, we have $S(a) \models \langle a{:}\alpha\rangle \phi$. Thus by M and $\langle;\rangle$ we can conclude the desired result.

- When $a{:}\alpha$ is a Demon loop, i.e., $a{:}(g{:}\gamma)^\times$, then we analyze cases depending on the value of b.

(1) If $b = a$, then $(a{:}\alpha \between S)_{:b} = a{:}\alpha \between S$ and $(a{:}\alpha)_{b:} = a{:}\alpha$.
Thus we must show that $S(a) \models \langle a{:}\alpha \between S\rangle \langle a{:}\alpha\rangle \phi$.
$S(a) \models \langle a{:}\alpha \between S\rangle S(a)$ because $S(a)$ is a loop invariant of $(g{:}\gamma \between S(\text{end} \mapsto S(a)))^\times$ by Lemma F.6.
$S(a) \models \langle a{:}\alpha\rangle \phi$ by definition of a subvalue map. Thus by M, we can conclude the desired result.

(2) Otherwise, if $b \in \text{subgames}(g{:}\gamma)$, then:
$(a{:}\alpha \between S)_{:b} = a{:}\alpha \between S\,;\, (g{:}\gamma \between S(\text{end} \mapsto S(a)))_{:b}$,
$(a{:}\alpha)_{b:} = (g{:}\gamma)_{b:}\,;\, a{:}\alpha$.
Thus we must show that

$$S(a) \models \langle a{:}\alpha \between S\,;\, (g{:}\gamma \between S(\text{end} \mapsto S(a)))_{:b}\rangle \langle (g{:}\gamma)_{b:}\,;\, a{:}\alpha\rangle \phi$$

First, we have $S(a) \models \langle a{:}\alpha \pmb{\wr} S\rangle\, S(a)$ because $S(a)$ is a loop invariant of $(g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(a)))^\times$ by Lemma F.6.
Next, by the inductive hypothesis, we have

$$S(a) \models \langle (g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(a)))_{:b}\,;\, (g{:}\gamma)_{b:}\rangle\, S(a).$$

By definition of a subvalue map, we have $S(a) \models \langle a{:}\alpha\rangle\, \phi$.
Thus by M and $\langle;\rangle$, we can conclude the desired result.

- When $a{:}\alpha$ is sequential composition, i.e., $a{:}(g{:}\gamma\,;\, d{:}\delta)$, then we analyze cases depending on the value of b.
  (1) If $b = a$, then $(a{:}\alpha \pmb{\wr} S)_{:b} = \text{skip}$ and $a{:}\alpha_{b:} = a{:}\alpha$. Thus we must show that $S(a) \models \langle\text{skip}\rangle\, \langle a{:}\alpha\rangle\, \phi$, i.e., $S(a) \models \langle a{:}\alpha\rangle\, \phi$, which hold by definition of a subvalue map.
  (2) Otherwise, if $b \in \text{subgames}(g{:}\gamma)$, then $(a{:}\alpha \pmb{\wr} S)_{:b} = (g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d)))_{:b}$ and $a{:}\alpha_{b:} = (g{:}\gamma)_{b:}\,;\, d{:}\delta$.
  Thus we must show that $S(a) \models \langle g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d))_{:b}\rangle\, \langle (g{:}\gamma)_{b:}\,;\, d{:}\delta\rangle\, \phi$.
  This is $S(a) \models \langle (g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d)))_{:b}\rangle\, \langle (g{:}\gamma)_{b:}\rangle\, \langle d{:}\delta\rangle\, \phi$ applying $\langle;\rangle$.
  This is $S(a) \models \langle (g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d)))_{:b}\,;\, (g{:}\gamma)_{b:}\rangle\, \langle d{:}\delta\rangle\, \phi$ again applying $\langle;\rangle$.
  By the inductive hypothesis, $S(a) \models \langle (g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d)))_{:b}\rangle\, S(d)$ since $S(a) \to S(g)$ in an inductive subvalue map.
  By definition of an Angelic subvalue map, $S(d) \models \langle d{:}\delta\rangle\, \phi$.
  By rule M, we can conclude the desired result.
  (3) Otherwise, if $b \in \text{subgames}(d{:}\delta)$, then a similar argument applies. $a{:}\alpha \pmb{\wr} S_{:b} = \{g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d))\,;$ and $a{:}\alpha_{b:} = d{:}\delta_{b:}$.
  Thus we must show that $S(a) \models \langle g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d))\,;\, (d{:}\delta \pmb{\wr} S)_{:b}\rangle\, \langle d{:}\delta_{b:}\rangle\, \phi$.
  By the $\langle;\rangle$ axiom, this is $S(a) \models \langle g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d))\rangle\, \langle (d{:}\delta \pmb{\wr} S)_{:b}\,;\, (d{:}\delta)_{b:}\rangle\, \phi$.
  By Lemma F.6, and since in an inductive subvalue map, $\models S(a) \to S(g)$, $S(a) \models \langle g{:}\gamma \pmb{\wr} S(\text{end} \mapsto S(d))\rangle\, S(d)$.
  By the inductive hypothesis, $S(d) \models \langle (d{:}\delta \pmb{\wr} S)_{:b}\,;\, (d{:}\delta)_{b:}\rangle\, \phi$.
  By rule M, we can conclude the desired result.

□

Theorem 6.1 (Inductive subvalue map ensures winning actions). *Suppose S is an inductive Angelic subvalue map for game $a{:}\alpha$ and compatible with winning condition ϕ, i.e., $a{:}\alpha \models S$ and $S(\text{end}) \models \phi$. Upon starting in any state $\sigma \models S(a)$, and reaching subgame b by following any strategy induced by the subvalue map S, there is a way for Angel to win by continuing to follow S. That is, $S(a) \models \langle (a{:}\alpha \pmb{\wr} S)_{:b}\rangle\, \langle (a{:}\alpha \pmb{\wr}^{\exists} S)_{b:}\rangle\, \phi$.*

Proof. We perform induction on the structure of $a{:}\alpha$.

- When $a{:}\alpha$ is atomic and not controlled by Angel, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then The only subgame b in subgames$(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b = a$. $a{:}\alpha_{:a}$ is the empty game skip, and $a{:}\alpha_{a:}$ is the game $a{:}\alpha$, and $a{:}\alpha \pmb{\wr} S$ is just $a{:}\alpha$. Thus, $\langle (a{:}\alpha \pmb{\wr} S)_{:a}\rangle\, \langle (a{:}\alpha \pmb{\wr}^{\exists} S)_{a:}\rangle\, \phi$ is equivalent to $\langle\text{skip}\rangle\, \langle\alpha\rangle\, \phi$. We need to show that $S(a) \models \langle a{:}\alpha \pmb{\wr}^{\exists} S\rangle\, \phi$ which holds because by definition of an inductive Angelic subvalue map along with the fact that $S(\text{end}) \to \phi$, $S(a) \to \langle a{:}\alpha\rangle\, \phi$.
- When $a{:}\alpha$ is an Angelic nondeterministic assignment, i.e., $a{:}x := *$, then again the only subgame b in subgames$(a{:}\alpha)$ is $a{:}\alpha$ itself. Thus the only possibility is $b = a$. $a{:}\alpha_{:a}$ is the empty game skip, and $a{:}\alpha \pmb{\wr} S$ is just $a{:}\alpha\,;\, ?S(\text{end})$, and $a{:}\alpha\,;\, ?S(\text{end})_{a:}$ is the game $a{:}\alpha\,;\, ?S(\text{end})$. Thus, $\langle a{:}\alpha_{:a} \pmb{\wr} S\rangle\, \langle a{:}\alpha_{a:} \pmb{\wr}^{\exists} S\rangle\, \phi$ is equivalent to $\langle\text{skip}\rangle\, \langle\alpha\,;\, ?S(\text{end})\rangle\, \phi$. We need to show that $S(a) \models \langle\alpha\rangle\, (?S(\text{end}) \wedge \phi)$. Since ϕ is a compatible winning condition, this is $S(a) \models \langle\alpha\rangle\, ?S(\text{end})$ which holds by the definition of an inductive Angelic subvalue map.

- When $a{:}\alpha$ is an Angelic ODE, i.e., $a{:}\{x' = f(x)\ \&\ Q\}$, then the argument is similar to the previous case. As before the only possibility is $b = a$, and we need to show that $S(a) \models \langle \alpha \rangle\, ?S(\mathsf{end}) \wedge \phi$. Since ϕ is a compatible winning condition, this is $S(a) \models \langle a{:}\alpha \dagger S \rangle\, ?S(\mathsf{end})$ which holds by the definition of an inductive Angelic subvalue map.
- When $a{:}\alpha$ is an Angelic choice, i.e., $a{:}(g{:}\gamma \cup d{:}\delta)$, then we analyze cases depending on the value of b.
  (1) If $b = a$, then $(a{:}\alpha \curlyvee S)_{:b} = \mathsf{skip}$ and $a{:}\alpha \dagger S_{b:} = a{:}\alpha \dagger S$. Thus we must show that $S(a) \models \langle \mathsf{skip} \rangle \langle a{:}\alpha \dagger S \rangle \phi$, i.e., $S(a) \models \langle a{:}\alpha \dagger S \rangle \phi$, which hold by Lemma F.7.
  (2) Otherwise, if $b \in \mathsf{subgames}(g{:}\gamma)$, then $(a{:}\alpha \curlyvee S)_{:b} = !S(g)\,;\, g{:}\gamma \curlyvee S_{:b}$ and $a{:}\alpha \dagger S_{b:} = g{:}\gamma_{b:}$. Thus we must show that $S(a) \models \langle !S(g)\,;\, g{:}\gamma \curlyvee S_{:b} \rangle \langle g{:}\gamma_{b:} \rangle \phi$.
  This is $S(a) \wedge S(g) \models \langle g{:}\gamma \curlyvee S_{:b} \rangle \langle g{:}\gamma_{b:} \rangle \phi$ using axioms $\langle;\rangle$, $[?]$. This holds by the inductive hypothesis.
  (3) Otherwise, if $b \in \mathsf{subgames}(d{:}\delta)$, then the proof proceeds similarly to the previous case, but with $d{:}\delta$ replacing $g{:}\gamma$.
- When $a{:}\alpha$ is a Demonic choice, i.e., $a{:}(g{:}\gamma \cap d{:}\delta)$, then the argument is similar to Angelic choice, except that we get assumptions $S(g)$ or $S(d)$ not from the execution prefix but from $S(a)$ because in an inductive subvalue map, $\models S(a) \to S(g) \wedge S(d)$. We analyze cases depending on the value of b.
  (1) If $b = a$, then $(a{:}\alpha \curlyvee S)_{:b} = \mathsf{skip}$ and $a{:}\alpha \dagger S_{b:} = a{:}\alpha \dagger S$. Thus we must show that $S(a) \models \langle \mathsf{skip} \rangle \langle a{:}\alpha \dagger S \rangle \phi$, i.e., $S(a) \models \langle a{:}\alpha \dagger S \rangle \phi$, which hold by Lemma F.7.
  (2) Otherwise, if $b \in \mathsf{subgames}(g{:}\gamma)$, then $(a{:}\alpha \curlyvee S)_{:b} = g{:}\gamma \curlyvee S_{:b}$ and $a{:}\alpha \dagger S_{b:} = g{:}\gamma_{b:}$. Thus we must show that $S(a) \models \langle g{:}\gamma \curlyvee S_{:b} \rangle \langle g{:}\gamma_{b:} \rangle \phi$. This holds by the inductive hypothesis since as discussed, $S(a) \to S(g)$.
  (3) Otherwise, if $b \in \mathsf{subgames}(d{:}\delta)$, then the proof proceeds similarly to the previous case, but with $d{:}\delta$ replacing $g{:}\gamma$.
- When $a{:}\alpha$ is sequential composition, i.e., $a{:}(g{:}\gamma\,;\, d{:}\delta)$, then we analyze cases depending on the value of b.
  (1) If $b = a$, then $(a{:}\alpha \curlyvee S)_{:b} = \mathsf{skip}$ and $a{:}\alpha \dagger S_{b:} = a{:}\alpha \dagger S$. Thus we must show that $S(a) \models \langle \mathsf{skip} \rangle \langle a{:}\alpha \dagger S \rangle \phi$, i.e., $S(a) \models \langle a{:}\alpha \dagger S \rangle \phi$, which hold by Lemma F.7.
  (2) Otherwise, if $b \in \mathsf{subgames}(g{:}\gamma)$, then $(a{:}\alpha \curlyvee S)_{:b} = (g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)))_{:b}$ and $a{:}\alpha \dagger S_{b:} = (g{:}\gamma \dagger S(\mathsf{end} \mapsto S(d)))_{b:}\,;\, d{:}\delta \dagger S$. Thus we must show that $S(a) \models \langle g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)))_{:b} \rangle \langle (g{:}\gamma \dagger S(\mathsf{en}$
  This is $S(a) \models \langle (g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)))_{:b} \rangle \langle (g{:}\gamma \dagger S(\mathsf{end} \mapsto S(d)))_{b:} \rangle \langle d{:}\delta \dagger S \rangle \phi$ applying $\langle;\rangle$.
  This is $S(a) \models \langle (g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)))_{:b}\,;\, (g{:}\gamma \dagger S(\mathsf{end} \mapsto S(d)))_{b:} \rangle \langle d{:}\delta \dagger S \rangle \phi$ again applying $\langle;\rangle$.
  By the inductive hypothesis, $S(a) \models \langle (g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)))_{:b} \rangle S(d)$ since $S(a) \to S(g)$ in an inductive subvalue map.
  By Lemma F.7, $S(d) \models \langle d{:}\delta \dagger S \rangle \phi$.
  By rule M, we can conclude the desired result.
  (3) Otherwise, if $b \in \mathsf{subgames}(d{:}\delta)$, then a similar argument applies. $a{:}\alpha \curlyvee S_{:b} = \{g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d))\,;$ and $a{:}\alpha \dagger S_{b:} = d{:}\delta \dagger S_{b:}$.
  Thus we must show that $S(a) \models \langle g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d))\,;\, (d{:}\delta \curlyvee S)_{:b} \rangle \langle d{:}\delta \dagger S_{b:} \rangle \phi$.
  By the $\langle;\rangle$ axiom, this is $S(a) \models \langle g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)) \rangle \langle (d{:}\delta \curlyvee S)_{:b}\,;\, (d{:}\delta \dagger S)_{b:} \rangle \phi$.
  By Lemma F.8, and since in an inductive subvalue map, $\models S(a) \to S(g)$, $S(a) \models \langle g{:}\gamma \curlyvee S(\mathsf{end} \mapsto S(d)) \rangle S(d)$.
  By the inductive hypothesis, $S(d) \models \langle (d{:}\delta \curlyvee S)_{:b}\,;\, (d{:}\delta \dagger S)_{b:} \rangle \phi$.
  By rule M, we can conclude the desired result.
- When $a{:}\alpha$ is an Angelic loop, i.e., $a{:}(g{:}\gamma)^*$, then we analyze cases depending on the value of b.

(1) If $b = a$ then $(a{:}\alpha \cup S)_{:b} = (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times$ (with the terminal test missing compared to $a{:}\alpha \cup S$) and $a{:}\alpha \overset{\exists}{\downarrow} S_{b:} = a{:}\alpha \overset{\exists}{\downarrow} S$.
We need to show that $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times \rangle \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$.
We first show that $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times \rangle S(a)$. This follows from applying the loop rule with invariant $S(a)$ and then using the fact hat per Lemma F.8, $S(g) \models \langle g{:}\gamma \cup S(\mathsf{end} \mapsto S(a))\rangle S(a)$.
Then by Lemma F.7, $S(a) \models \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$. Thus by M, we can conclude the desired result.

(2) Otherwise, if $b \neq a$ then it is in subgames$(g{:}\gamma)$.
$(a{:}\alpha \cup S)_{:b} = (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a))_{:b})^\times\,;\, !S(g)\,;\, (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}$.
$a{:}\alpha \overset{\exists}{\downarrow} S_{b:} = (g{:}\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a))_{b:}\,;\, a{:}\alpha \overset{\exists}{\downarrow} S$.
Want to show:

$$S(a) \models \langle (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times\,;\, !S(g)\,;\, (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}\rangle \langle (g{:}\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a)))_{b:}\,;\, a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi.$$

First, by Lemma F.8, we have $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times\,;\, !S(\mathsf{end})\rangle \top$, so by axioms [?] and $\langle;\rangle$, $S(a) \models \langle (!S(g)\,;\, g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times\rangle \top$.
Next, by the inductive hypothesis and applications of axioms $\langle;\rangle$, $\langle ?\rangle$ and $\to$R, we have

$$\top \models \langle !S(g)\,;\, (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}\rangle \langle g{:}\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a))_{b:}\rangle S(a)$$

Finally, by Lemma F.7, we have $S(a) \models \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$. Thus by M and $\langle;\rangle$ we can conclude the desired result.

- When $a{:}\alpha$ is a Demonic loop, i.e., $a{:}(g{:}\gamma)^\times$, then we analyze cases depending on the value of b.

(1) If $b = a$, then $(a{:}\alpha \cup S)_{:b} = a{:}\alpha \cup S$ and $(a{:}\alpha \overset{\exists}{\downarrow} S)_{b:} = a{:}\alpha \overset{\exists}{\downarrow} S$.
Thus we must show that $S(a) \models \langle a{:}\alpha \cup S\rangle \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$.
$S(a) \models \langle a{:}\alpha \cup S\rangle S(a)$ because $S(a)$ is a loop invariant of $(g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times$ by Lemma F.8.
$S(a) \models \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$ by Lemma F.7. Thus by M, we can conclude the desired result.

(2) Otherwise, if $b \in$ subgames$(g{:}\gamma)$, then:
$(a{:}\alpha \cup S)_{:b} = a{:}\alpha \cup S\,;\, (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}$,
$(a{:}\alpha \overset{\exists}{\downarrow} S)_{b:} = (g{:}\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a)))_{b:}\,;\, a{:}\alpha \overset{\exists}{\downarrow} S$.
Thus we must show that

$$S(a) \models \langle a{:}\alpha \cup S\,;\, (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}\rangle \langle (g{:}\gamma \overset{\exists}{\downarrow} S(\mathsf{end} \mapsto S(a)))_{b:}\,;\, a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$$

First, we have $S(a) \models \langle a{:}\alpha \cup S\rangle S(a)$ because $S(a)$ is a loop invariant of $(g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))^\times$ by Lemma F.8.
Next, by the inductive hypothesis, we have $S(a) \models \langle (g{:}\gamma \cup S(\mathsf{end} \mapsto S(a)))_{:b}\rangle S(a)$.
By Lemma F.7, we have $S(a) \models \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle \phi$.
Thus by M and $\langle;\rangle$, we can conclude the desired result.

□

Theorem 6.4 (Inductive subvalue maps are subvalue maps). *For* dGL *game* $a{:}\alpha$*, if* S *is an inductive Angelic subvalue map (*$a{:}\alpha \models S$*), then it is also an Angelic subvalue map for* $a{:}\alpha$*, i.e., for every subgame* $b{:}\beta$ *in* subgames$(a{:}\alpha)$*,* $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\mathsf{end})$*. Dually, if* S *is an inductive Demonic subvalue map for* $a{:}\alpha$*, then* S *is a Demonic subvalue map for* $a{:}\alpha$*.*

Proof. The proof uses structural induction. When $a{:}\alpha$ is atomic, i.e. $\alpha \in \{x := e, x := *, ?Q, !Q, \{x' = f(x)\ \&\ Q\}, \{x' = f(x)\ \&\ Q\}^d\}$, according to Def. 6.2, $a{:}\alpha \models S$ exactly when $\models S(a) \to \langle\alpha\rangle S(\mathsf{end})$. Since $a{:}\alpha_{a:} = a{:}\alpha$ and a is the only subgame in subgames$(a{:}\alpha)$, we can conclude that S is a subvalue map. For the recursive cases, if $a{:}\alpha$ has the structure:

(1) $g{:}\gamma \cup d{:}\delta$, for any subgame b in subgames($g{:}\gamma$), $a{:}\alpha_{b:} = g{:}\gamma_{b:}$. Since the per inductive hypothesis, $\models S(b) \to \langle g{:}\gamma_{b:}\rangle S(\text{end})$, we can conclude that $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$. The same argument applies to all subgames in subgames($d{:}\delta$). It remains to show that for the overall game a, $\models S(a) \to \langle a{:}\alpha_{a:}\rangle S(\text{end})$. By the inductive hypothesis, because $g{:}\gamma \models S$, so $\models S(g) \to \langle\gamma\rangle S(\text{end})$ and because $d{:}\delta \models S$, so $\models S(d) \to \langle\delta\rangle S(\text{end})$. Using the disjunction of implications, $\models S(g) \vee S(d) \to \langle\gamma\rangle S(\text{end}) \vee \langle\delta\rangle S(\text{end})$. But according to dGL axiom $\langle\cup\rangle$, $\models \langle\gamma\rangle S(\text{end}) \vee \langle\delta\rangle S(\text{end}) \leftrightarrow \langle\gamma \cup \delta\rangle S(\text{end})$, so $\models S(g) \vee S(d) \to \langle\gamma \cup \delta\rangle S(\text{end})$. Because S is valid, $\models S(a) \to S(g) \vee S(d)$. By transitivity of implication, $\models S(a) \to \langle\gamma \cup \delta\rangle S(\text{end})$, proving the desired result.

(2) $g{:}\gamma \cap d{:}\delta$, for any subgame b in subgames($g{:}\gamma$), $a{:}\alpha_{b:} = g{:}\gamma_{b:}$. Since per the inductive hypothesis, $\models S(b) \to \langle g{:}\gamma_{b:}\rangle S(\text{end})$, we can conclude that $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$. The same argument applies to all subgames in subgames($d{:}\delta$). It remains to show that for the overall game a, $\models S(a) \to \langle a{:}\alpha_{a:}\rangle S(\text{end})$. By the inductive hypothesis, because $g{:}\gamma \models S$, so $\models S(g) \to \langle\gamma\rangle S(\text{end})$ and because $d{:}\delta \models S$, so $\models S(d) \to \langle\delta\rangle S(\text{end})$. Using the conjunction of implications, $\models S(g) \wedge S(d) \to \langle\gamma\rangle S(\text{end}) \wedge \langle\delta\rangle S(\text{end})$. But according to dGL axiom $\langle\cap\rangle$, $\models \langle\gamma\rangle S(\text{end}) \wedge \langle\delta\rangle S(\text{end}) \leftrightarrow \langle\gamma \cap \delta\rangle S(\text{end})$, so $\models S(g) \wedge S(d) \to \langle\gamma \cap \delta\rangle S(\text{end})$. Because S is valid, $\models S(a) \to S(g) \wedge S(d)$. By transitivity of implication, $\models S(a) \to \langle\gamma \cap \delta\rangle S(\text{end})$, proving the desired result.

(3) $g{:}\gamma; d{:}\delta$, for any subgame b in subgames($d{:}\delta$), $d{:}\delta_{b:} = a{:}\alpha_{b:}$. Since per the inductive hypothesis, $\models S(b) \to \langle d{:}\delta_{b:}\rangle S(\text{end})$, we can conclude that $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$.
Consider now any subgame b in subgames($g{:}\gamma$). Because $g{:}\gamma \models S$, $\models S(b) \to \langle g{:}\gamma_{b:}\rangle S(d)$. Now, because $d{:}\delta \models S$, $\models S(d) \to \langle\delta\rangle S(\text{end})$. According to the dGL monotonicity rule M, this means that $\models S(b) \to \langle g{:}\gamma_{b:}\rangle\langle\delta\rangle S(\text{end})$. According to the dGL axiom $\langle;\rangle$, $\models S(b) \to \langle g{:}\gamma_{b:}; \delta\rangle S(\text{end})$. Since $a{:}\alpha_{b:} = g{:}\gamma_{b:}; d{:}\delta$, we can conclude that $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$.
It remains to show that for the overall game a, $\models S(a) \to \langle a{:}\alpha_{a:}\rangle S(\text{end})$. By the inductive hypothesis, because $g{:}\gamma \models S$, so $\models S(g) \to \langle\gamma\rangle S(\delta)$ and because $d{:}\delta \models S$, so $\models S(d) \to \langle\delta\rangle S(\text{end})$. According to the dGL monotonicity rule M, this means that $\models S(g) \to \langle\gamma\rangle\langle\delta\rangle S(\text{end})$. According to the dGL axiom $\langle;\rangle$, $\models \langle\gamma\rangle\langle\delta\rangle S(\text{end}) \leftrightarrow \langle\gamma;\delta\rangle S(\text{end})$. Thus, $\models S(g) \to \langle\gamma;\delta\rangle S(\text{end})$. Because S is valid, $\models S(a) \to S(g)$. By transitivity of implication, $\models S(a) \to \langle\gamma;\delta\rangle S(\text{end})$, proving the desired result.

(4) $g{:}\gamma^*$, then because $a{:}(g{:}\gamma)^* \models S$, then immediately as the first validity condition $\models S(a) \to \langle a{:}\alpha \exists\!\downarrow S\rangle S(\text{end})$. Per Lemma F.3, $\models \langle a{:}\alpha \exists\!\downarrow S\rangle S(\text{end}) \to \langle\alpha\rangle S(\text{end})$. By the transitivity of implication, $\models S(a) \to \langle\alpha\rangle S(\text{end})$.

(5) $g{:}\gamma^\times$, then because $a{:}(g{:}\gamma^\times) \models S$, $\models S(a) \to S(g)$ and $\models S(a) \to S(\text{end})$. Additionally, by the inductive hypothesis, because $g{:}\gamma \models S$, $\models S(g) \to \langle\gamma\rangle S(a)$. By the transitivity of implication, $\models S(a) \to \langle\gamma\rangle S(a)$. Applying the ind rule of dGL, this permits us to infer $\models S(a) \to \langle\gamma^\times\rangle S(a)$. Recall that because $a{:}(g{:}\gamma^\times) \models S$, $\models S(a) \to S(\text{end})$. By the monotonicity rule M, $\models S(a) \to \langle\gamma^\times\rangle S(\text{end})$, proving that $\models S(a) \to a{:}\alpha_{a:}$.
It remains to show that for any subgame b in subgames($g{:}\gamma$), $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$. Because $g{:}\gamma \models S$, by the inductive hypothesis $\models S(b) \to \langle g{:}\gamma_{b:}\rangle S(a)$. Since $\models S(a) \to \langle\gamma^\times\rangle S(\text{end})$, by the monotonicity rule M, $\models S(b) \to \langle g{:}\gamma_{b:}\rangle\langle\gamma^\times\rangle S(\text{end})$. By dGL axiom $\langle;\rangle$, $\models S(b) \to \langle g{:}\gamma_{b:}; \gamma^\times\rangle S(\text{end})$. But, $a{:}\alpha_{b:} = g{:}\gamma_{b:}; (\gamma^\times)$. Thus, $\models S(b) \to \langle a{:}\alpha_{b:}\rangle S(\text{end})$.

□

Corollary F.9 (Inductive subvalue map correctness). *For* dGL *game* $a{:}\alpha$ *and inductive Angelic subvalue map* S*, if* $a{:}\alpha \models S$*, then* $\models S(a) \to \langle\alpha\rangle S(\text{end})$*. Dually, if* $a{:}\alpha[]\models S$*, then* $\models S(a) \to [\alpha]S(\text{end})$.

Proof. Corollary of Theorem 6.4, which shows that inductive subvalue maps are subvalue maps. Def. 5.1 says that the winning subregion for every subgame in the subvalue map implies the winning region for starting at that subgame and running the rest of the game. For overall game a, this means $\models S(a) \to \langle\alpha\rangle S(\text{end})$. □

Theorem F.10 (Subvalue maps are sound monitors). *For* dGL *game* $a{:}\alpha$ *and subvalue map* S*, if* $a{:}\alpha \models S$*, then* $\models S(a) \to \langle a{:}\alpha \between S\phi\rangle\phi$*. Dually, if* $a{:}\alpha[] \models S$ *then* $\models S(a) \to [a{:}\alpha \between S\phi]\phi$.

Proof. The proof proceeds by induction on the structure of the game $a{:}\alpha$. We show this for the $a{:}\alpha \between S$ case; the dual case is analogous. If $a{:}\alpha$ has structure:

- atomic and Angel does not take any decision, i.e., $\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\}$, then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle\alpha\rangle\phi$. Because $a{:}\alpha \models S$, we know that $\models S(a) \to \langle\alpha\rangle\phi$.
- $x := *$, then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle ?\langle\alpha\rangle\phi\,;\, x := \otimes\,;\, !\phi\rangle\phi$. By the $\langle;\rangle$, $[?]$ and $\langle ?\rangle$ axioms, this is equivalent to $\langle\alpha\rangle\phi \land \langle x := \otimes\rangle(\phi \to \phi)$. Because $a{:}\alpha \models S$, we know that $\models S(a) \to \langle\alpha\rangle\phi$. By $\langle:=\rangle$, $\langle x := \otimes\rangle\top$ is $\forall x \top$ which is $\top$.
- $\{x' = f(x)\ \&\ Q\}$, then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle ?\langle\alpha\rangle\phi\,;\, \{x' = f(x)\ \&\ Q\}^d\,;\, !\phi\rangle\phi$. By the $\langle:=\rangle$, $\langle ?\rangle$ and $[?]$ axioms, this is equivalent to $\langle\alpha\rangle\phi \land \langle\{x' = f(x)\ \&\ Q\}^d\rangle(\phi \to \phi)$. Because $a{:}\alpha \models S$, we know that $\models S(a) \to \langle\alpha\rangle\phi$. By $[']$ and $\langle^d\rangle$, $\langle\{x' = f(x)\ \&\ Q\}^d\rangle\top$ is $\forall t \geq 0 [x := y(t)]\top$ where $y'(t) = f(y)$. By G, $[x := y(t)]\top$ is $\top$, so $\forall t \geq 0 \top$ is $\top$.
- $a{:}(g{:}\gamma \cup d{:}\delta)$ then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle ?S(g) \lor S(d)\,;\, \big((g{:}\gamma \between S) \cap (d{:}\delta \between S)\big)\rangle\phi$. By the $\langle ?\rangle$, $\langle\cup\rangle$, $[?]$ and $\langle;\rangle$ axioms, this is equivalent to $(S(g) \lor S(d)) \land (S(g) \to \langle g{:}\gamma \between S\rangle\phi) \land (S(d) \to \langle d{:}\delta \between S\rangle\phi)$. Because $a{:}\alpha \models S$, we know that $\models S(a) \to S(g) \lor S(d)$, and further, $g{:}\gamma \models S$ and $d{:}\delta \models S$. By the inductive hypothesis, $\models S(g) \to \langle g{:}\gamma \between S\rangle\phi$ and $\models S(d) \to \langle d{:}\delta \between S\rangle\phi$. Thus, $\models S(a) \to \langle a{:}\alpha \between S\rangle\phi$.
- $a{:}(g{:}\gamma \cap d{:}\delta)$ then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle\big((g{:}\gamma \between S) \cap (d{:}\delta \between S)\big)\rangle\phi$. We want to show $S(a) \to \langle(g{:}\gamma \between S) \cap (d{:}\delta \between S)\rangle\phi$. By the inductive hypothesis, $\models S(g) \to \langle g{:}\gamma \between S\rangle\phi$ and $\models S(d) \to \langle d{:}\delta \between S\rangle\phi$. By $\langle\cup\rangle$ and M, then, $\models (S(g) \land S(d)) \to \langle(g{:}\gamma \between S) \cap (d{:}\delta \between S)\rangle\phi$. By the definition of a valid solution, $S(a) \to S(g) \land S(d)$. By transitivity of implication, $\models S(a) \to \langle(g{:}\gamma \between S) \cap (d{:}\delta \between S)\rangle\top$.
- $a{:}(g{:}\gamma\,;\, d{:}\delta)$ then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle(g{:}\gamma \between S(\text{end} \mapsto S(d(\text{end}))))\,;\, (d{:}\delta \between S)\rangle\phi$. After applying the $\langle;\rangle$ axiom, we want to show $S(a) \to \langle g{:}\gamma \between S(\text{end} \mapsto S(d(\text{end})))\rangle\langle d{:}\delta \between S\rangle\phi$. By the inductive hypothesis, $\models S(g) \to \langle g{:}\gamma \between S(\text{end} \mapsto S(d(\text{end})))\rangle S(d)$ and $\models S(d) \to \langle d{:}\delta \between S\rangle\phi$. Since $\models S(d) \to \langle d{:}\delta \between S\rangle\phi$, by M, it remains to show that $\models S(a) \to \langle g{:}\gamma \between S(\text{end} \mapsto S(d(\text{end})))\rangle S(d)$. Since $a{:}\alpha \models S$, we know that $\models S(a) \to S(g)$. Thus, by transitivity of implication, $\models S(a) \to \langle g{:}\gamma \between S(\text{end} \mapsto S(d(\text{end})))\rangle S(d)$, completing the proof.
- $a{:}(g{:}\gamma)^*$ then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle ?(S(a))\,;\, (!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a))^\times\,;\, !S(\text{end})\rangle\phi$. By the $\langle ?\rangle$, $[?]$ and $\langle;\rangle$ axioms, this is equivalent to
  $(S(a)) \land \langle(!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a))^\times\rangle(\phi \to \phi)$.
  Thus, we want to show $\models S(a) \to (S(a)) \land \langle(!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a))^\times\rangle\top$.
  We want to show the that $S(a) \to \langle(!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a))^\times\rangle\top$ holds. We prove this using the loop rule. $\top$ serves as an invariant of this loop game since it holds initially ($\models \top$), implies the postcondition ($\models \top \to \top$), and is inductive ($\langle(!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a)\rangle\top$) as argued next. Since $a{:}\alpha \models S$, so $g{:}\gamma \models S(\text{end} \mapsto S(a(\text{end})))$, By the inductive hypothesis, $\models S(g) \to \langle g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\rangle(S(a))$. Applying the axioms $[?]$, $\langle;\rangle$ and $\langle ?\rangle$, this is equivalent to
  $\models \langle!S(g)\,;\, g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end})))\,;\, ?S(a)\rangle\top$
  showing that $\top$ is an inductive invariant.
- $a{:}(g{:}\gamma)^\times$ then $\langle a{:}\alpha \between S\rangle\phi$ is $\langle(g{:}\gamma \between S)^\times\rangle\phi$.
  We want to show $\models S(a) \to \langle(g{:}\gamma \between S(\text{end} \mapsto S(a(\text{end}))))^\times\rangle\phi$.

Since $a{:}\alpha \models S$, it must be the case that $\models S(a) \to (S(g) \wedge \phi)$ and $g{:}\gamma \models S$. By the inductive hypothesis, $\models S(g) \to \langle g{:}\gamma \not\downarrow S(\mathsf{end} \mapsto S(a))\rangle S(a)$. We apply the loop rule to prove the conclusion $\models S(a) \to \langle (g{:}\gamma \not\downarrow S)^\times\rangle \phi$. $S(a)$ serves as an invariant of this loop. It holds initially. It is inductive per the inductive hypothesis (using weakening and $\models S(a) \to S(g)$), and implies the postcondition ($\models S(a) \to \phi$).

□

Theorem 7.2 (Maximal inductive subvalue map). *Amongst the inductive Angelic subvalue maps for game $a{:}\alpha$ compatible with winning condition ϕ, the model predictive Angelic map given by Def. 6.5 is maximal under the ordering of Def. 7.1. That is, for all S such that $a{:}\alpha \models S$, model predictive Angelic subvalue map S' satisfies $S' \sqsupseteq S$. Dually, the model predictive Demonic subvalue map (Def. 6.5) is maximal amongst inductive Demonic subvalue maps per the ordering of Def. 7.1.*

Proof. Lemma F.5 shows that the MPC solution is valid. Next we show that it is maximal. We first show that $S' \gtrsim S$ for all S such that $a{:}\alpha \models S$ under the weaker ordering of Def. F.1. A symmetric proof applies when $a{:}\alpha \models S$ is replaced by $a{:}\alpha[]\models S$. The proof uses structural induction.

For end, if $a{:}\alpha \models S$ then $\models S(\mathsf{end}) \to \phi$. But for the MPC solution, $S'(\mathsf{end})$ is ϕ, so $\models S(a) \to S'(a)$.

When $a{:}\alpha$ is atomic, i.e. $\alpha \in \{x := e, x := *, ?f, !f, \{x' = f(x) \;\&\; Q\}, \{x' = f(x) \;\&\; Q\}^d\}$, the result is immediate: according to Def. 6.2, if $a{:}\alpha \models S$ then $\models S(a) \to \langle\alpha\rangle\phi$. But per the definition of the MPC solution, $S'(a)$ is $\langle\alpha\rangle\phi$. Thus, $\models S(a) \to S'(a)$.

For the recursive cases, if $a{:}\alpha$ has the structure:

(1) $g{:}\gamma \cup d{:}\delta$, then observe that restrictions $S'|_{\mathsf{subgames}(\gamma)}$ and $S'|_{\mathsf{subgames}(\delta)}$ of S' are the MPC solutions for Angel win condition ϕ for games $g{:}\gamma$ and $d{:}\delta$ respectively. By the inductive hypothesis, for all subgames $b \in \mathsf{subgames}(g{:}\gamma)$, $S(b) \to S'(b)$. Similarly, for all subgames $b \in \mathsf{subgames}(d{:}\delta)$, $S(b) \to S'(b)$. The only remaining subgame in $\mathsf{subgames}(\alpha)$ is a. Per the inductive hypothesis, $\models S(g) \to S'(g)$ and $\models S(d) \to S'(g)$. Taking the disjunction of these implications, we get $\models S(g) \vee S(d) \to S'(g) \vee S'(d)$. Because S is valid, $\models S(a) \to S(g) \vee S(d)$. By transitivity of implication, $\models S(a) \to S'(g) \vee S'(d)$. By definition, $S'(a)$ is $S'(g) \vee S'(d)$. Thus we can conclude that $\models S(a) \to S'(a)$.

(2) $g{:}\gamma \cap d{:}\delta$, then observe that restrictions $S'|_{\mathsf{subgames}(\gamma)}$ and $S'|_{\mathsf{subgames}(\delta)}$ of S' are the MPC solutions for Angel win condition ϕ for games $g{:}\gamma$ and $d{:}\delta$ respectively. By the inductive hypothesis, for all nodes $b \in \mathsf{subgames}(g{:}\gamma)$, $S(b) \to S'(b)$. Similarly, for all subgames $b \in \mathsf{subgames}(d{:}\delta)$, $S(b) \to S'(b)$. The only remaining subgame in $\mathsf{subgames}(\alpha)$ is a. Per the inductive hypothesis, $\models S(g) \to S'(g)$ and $\models S(d) \to S'(g)$. Taking the conjunction of these implications, we get $\models S(g) \wedge S(d) \to S'(g) \wedge S'(d)$. Because S is valid, $\models S(a) \to S(g) \wedge S(d)$. By transitivity of implication, $\models S(a) \to S'(g) \wedge S'(d)$. By definition, $S'(a)$ is $S'(g) \wedge S'(d)$. Thus we can conclude that $\models S(a) \to S'(a)$.

(3) $g{:}\gamma; d{:}\delta$, then observe that $S'|_{\mathsf{subgames}(\delta)}$ is the MPC solutions for Angel win condition ϕ for game $d{:}\delta$. By the inductive hypothesis, for all subgames $b \in \mathsf{subgames}(d{:}\delta)$, $\models S(b) \to S'(b)$. Now, let S'' be the MPC solution for Angel win condition $S(d)$ for game $g{:}\gamma$. By the inductive hypothesis, for all subgames $b \in \mathsf{subgames}(g{:}\gamma)$, $\models S(b) \to S''(b)$. By definition, for any subgame $b \in \mathsf{subgames}(g{:}\gamma)$, $S''(b)$ is $\langle g{:}\gamma_{b:}\rangle S(d)$ while $S'(b)$ is $\langle g{:}\gamma_{b:}\rangle\phi$. As argued already, by the inductive hypothesis applied to game δ, $\models S(d) \to S'(d)$. By the dGL monotonicity rule M, $\models \langle g{:}\gamma_{b:}\rangle S(d) \to \langle g{:}\gamma_{b:}\rangle S'(d)$. Thus, $\models S''(b) \to S'(b)$. By the transitivity of implication, $\models S(b) \to S'(b)$. We have shown the correct implication for all the nodes in $\mathsf{subgames}(g{:}\gamma)$ and $\mathsf{subgames}(d{:}\delta)$. The only subgame remaining is a. Because $a{:}\alpha \models S$, $\models S(a) \to S(g)$. We have shown that $\models S(g) \to S'(g)$. By the definition of the MPC solution, $S'(a)$ is $S'(g)$. Thus, $\models S(a) \to S'(a)$.

(4) $g{:}\gamma^*$, then because $a{:}(\gamma^*) \models S$, $\models S(a) \to \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\phi$. Further, by Lemma F.3, $\models S(a) \to \langle\gamma^*\rangle\phi$. But by the definition of the MPC solution, $S'(a)$ is $\langle\gamma^*\rangle\phi$. By transitivity of implication, $\models S(a) \to S'(a)$. The subgames that remain besides a all belong to γ. Consider S'', the MPC solution for Angel win condition $S(a)$ for game $g{:}\gamma$. By the inductive hypothesis, for every subgame b in γ, $\models S(b) \to S''(b)$. Now, we argue that $\models S''(b) \to S'(b)$, so that by transitivity of implication, $\models S(b) \to S'(b)$ completing the proof. Per the definition of the MPC solution, $S''(b)$ is $\langle g{:}\gamma_{b:}\rangle S(a)$ while $S'(b)$ is $\langle g{:}\gamma_{b:}\rangle\phi$. By the monotonicity rule M, since $\models S(a) \to \phi$, we have $\models \langle g{:}\gamma_{b:}\rangle S(a) \to \langle g{:}\gamma_{b:}\rangle\phi$. Thus, $\models S''(b) \to S'(b)$.

(5) $g{:}\gamma^\times$, then because $a{:}(\gamma^\times) \models S$, per Theorem 6.4, $\models S(a) \to \langle\gamma^\times\rangle\phi$. By the definition of the MPC solution, $S'(a)$ is $\langle\gamma^\times\rangle\phi$. So, $\models S(a) \to S'(a)$. The nodes that remain besides a all belong to subgames$(g{:}\gamma)$. Consider S'', the MPC solution for Angel win condition $S(a)$ for game $g{:}\gamma$. By the inductive hypothesis, for every subgame b in γ, $\models S(b) \to S''(b)$. Now, we argue that $\models S''(b) \to S'(b)$, so that by transitivity of implication, $\models S(b) \to S'(b)$, completing the proof. Per the definition of the MPC solution, $S''(b)$ is $\langle g{:}\gamma_{b:}\rangle S(a)$ while $S'(b)$ is $\langle g{:}\gamma_{b:}\rangle S'(a)$. Since $\models S(a) \to S'(a)$, by the dGL monotonicity rule M, $\models \langle g{:}\gamma_{b:}\rangle S(a) \to \langle g{:}\gamma_{b:}\rangle S'(a)$. Thus, $\models S''(b) \to S'(b)$.

We have shown that $S' \gtrsim S$ for all S such that $a{:}\alpha \models S$ under the weaker ordering of Def. F.1. From Lemma F.2, it also follows that $S' \sqsupseteq S$ under the stronger ordering of Def. 7.1. Thus, S' is maximal. □

Theorem 8.1 (Sound solving). *$S := \langle\rangle\mathsf{map}(a{:}\alpha, \phi)$ is an inductive Angelic subvalue map for game $a{:}\alpha$ ($a{:}\alpha \models S$) with all subvalues in $\mathcal{P}_\mathbb{R}$, compatible with Angel winning condition $\phi \in \mathcal{P}_\mathbb{R}$. Dually, $S := []\mathsf{map}(a{:}\alpha, \phi)$ is an inductive Demonic subvalue map with all subvalues in $\mathcal{P}_\mathbb{R}$, compatible with Demon winning condition ϕ.*

Proof. The proof is by structural induction on $a{:}\alpha$. We show the proof for $\langle\rangle$map, the proof for $[]$map is symmetric. $S(\mathsf{end})$ is compatible with ϕ because $S(\mathsf{end})$ is set to ϕ. In the case where $a{:}\alpha$ is atomic, i.e., when $\alpha \in \{x := e, x := *, ?f, !f, \{x' = f(x)\&Q\}, \{x' = f(x)\&Q\}^d\}$, the result is immediate. According to Def. 6.2, $a{:}\alpha \models S$ exactly when $\models S(a) \to \langle\alpha\rangle\phi$. But per the definition of $\langle\rangle$map, $S(a)$ is $\mathsf{simpl}(\langle\alpha\rangle\phi)$ which by definition implies $\langle\alpha\rangle\phi$.

In the compositional cases, we use structural induction. If $a{:}\alpha$ has the structure:

- $a{:}(g{:}\gamma \cup d{:}\delta)$, then let $S_1 := \langle\rangle\mathsf{map}(g{:}\gamma, \phi)$ and $S_2 := \langle\rangle\mathsf{map}(d{:}\delta, \phi)$. S is the disjoint union of S_1 and S_2 extended by mapping a to $S_1(g) \vee S_2(d)$. We must show that $\models S(a) \to S(g) \vee S(d)$ and $g{:}\gamma \models S$ and $d{:}\delta \models S$. The first implication is immediate as $S(g) = S_1(g)$ and $S(d) = S_2(d)$ by the construction of S. The second and third formulas hold by the inductive hypothesis.
- $a{:}(g{:}\gamma \cap d{:}\delta)$, then let $S_1 := \langle\rangle\mathsf{map}(g{:}\gamma, \phi)$ and $S_2 := \langle\rangle\mathsf{map}(d{:}\delta, \phi)$. S is the disjoint union of S_1 and S_2 extended by mapping a to $S_1(g) \wedge S_2(d)$. We must show that $\models S(a) \to S(g) \wedge S(d)$ and $g{:}\gamma \models S$ and $d{:}\delta \models S$. The first implication is immediate as $S(g) = S_1(g)$ and $S(d) = S_2(d)$ by the construction of S. The second and third formulas hold by the inductive hypothesis.
- $a{:}(g{:}\gamma\,;\, d{:}\delta)$, then let $S_1 := \langle\rangle\mathsf{map}(d{:}\delta, \phi)$ and $S_2 := \langle\rangle\mathsf{map}(g{:}\gamma, S_1(d))$. S is the disjoint union of S_1 and S_2 extended by mapping a to $S_2(g)$. We must show that $\models S(a) \to S(g)$ and $g{:}\gamma \models S$ and $d{:}\delta \models S$. The first implication is immediate as $S(g) = S_2(g)$ by the construction of S. The second and third formulas hold by the inductive hypothesis.
- $a{:}(g{:}\gamma)^*$, the let $S' := \langle\rangle\mathsf{map}(g{:}\gamma, \mathsf{Inv} \vee \phi)$. S is constructed by extending S' by mapping a to Inv. Additionally, per the side condition $\models \mathsf{Inv} \to \langle a{:}\alpha \overset{\exists}{\downarrow} S'\rangle\phi$. We must show $S(a) \to \langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\phi$ and $g{:}\gamma \models S$. The first implication follows from the side condition along with the fact that

$S(g) = S'(g)$ by the construction of S. The second formula holds by the inductive hypothesis as it applies to S'.

- $a;(g;\gamma)^\times$, then let $S' := \langle\rangle \mathsf{map}(g;\gamma, \mathsf{Inv})$. S is constructed by extending S' by mapping a to Inv. Additionally, per the side condition $\models \mathsf{Inv} \rightarrow S(g) \wedge \phi$. We must show that $S(a) \rightarrow \phi \wedge S(g)$, and $g;\gamma \models S$. The first implication follows from the side condition along with the fact that $S(g) = S'(g)$ by the construction of S. The second formula holds by the inductive hypothesis as it applies to S'.

□

G Additional Definitions

This appendix provides additional definitions and constructions including the Demonic version for definitions whose Angelic subvalue map version is already defined in the main text, and dGL operational semantics concepts.

G.1 Labeled Game Trees

We construct the labeled game tree, extending [44][Appendix C] with labels. The operator $\hat{}$ denotes concatenation under prefix closure. That is:

(1) For two actions/labels $\mathfrak{a}_1$ and $\mathfrak{a}_2$, $\{\mathfrak{a}_1\hat{}\mathfrak{a}_2\}$ is the set $\{\mathfrak{a}_1, \mathfrak{a}_1 \cdot \mathfrak{a}_2\}$.
(2) For an action $\mathfrak{a}$ and a sequence of actions b, $\{\mathfrak{a}\hat{}b\}$ is the set $\{\mathfrak{a}, \mathfrak{a} \cdot b\}$.
(3) For two actions/labels $\mathfrak{a}_1$ and $\mathfrak{a}_2$, and sequence b, $\{\mathfrak{a}_1\hat{}\mathfrak{a}_2\hat{}b\}$ is the set $\{\mathfrak{a}_1, \mathfrak{a}_1 \cdot \mathfrak{a}_2, \mathfrak{a}_1 \cdot \mathfrak{a}_2 \cdot b\}$.
(4) For an action $\mathfrak{a}$ and *a set of sequences* $\{a, b\}$, $\{\mathfrak{a}\hat{}\{a, b\}\}$ is the set $\{\mathfrak{a}, \mathfrak{a} \cdot a, \mathfrak{a} \cdot b\}$.

Thus, $\hat{}$ is overloaded to denote the concatentation of actions, sequences, and sets of sequences under prefix closure.

Definition G.1 (Labeled Semantics). For any game $a;\alpha$ and state σ, the labeled semantics $g(a;\alpha)(\sigma)$ is defined as follows, where $\hat{}$ denotes concatenation under prefix closure (Appendix G.1).

$$
\begin{aligned}
\mathcal{T}(x := \theta, \sigma) &\equiv \{a\hat{}x := \theta\} \\
\mathcal{T}(x := \otimes, \sigma) &\equiv \{a\hat{}x := \theta^d \ : \ \theta \in \mathbb{R}\} \\
\mathcal{T}(x := *, \sigma) &\equiv \{a\hat{}x := \theta \ : \ \theta \in \mathbb{R}\} \\
\mathcal{T}(\{x' = f(x) \ \&\ Q\}, \sigma) &\equiv \{a\hat{}(x' = f(x)\&Q@r) \ : \ r \in \mathbb{R}, r \geq 0, \varphi(0) = \sigma \text{ for some} \\
&\qquad \text{(differentiable) } \varphi : [0, r] \to \mathcal{S} \text{ such that } \frac{d\varphi(t)(x)}{dt}(\zeta) = [\![\theta]\!]_{\varphi(\zeta)} \\
&\qquad \text{and } \varphi(\zeta) \in [\![Q]\!]^I \text{ for all } \zeta \leq r\} \\
\mathcal{T}(?Q, \sigma) &\equiv \{a\hat{}?Q\} \\
\mathcal{T}(!Q, \sigma) &\equiv \{a\hat{}!Q\} \\
\mathcal{T}(g;\gamma \cup d;\delta, s) &\equiv \{a\hat{}\mathfrak{l}\hat{}b \ : \ b \in g(g;\gamma)(\sigma)\} \cup \\
&\qquad \{a\hat{}\mathfrak{r}\hat{}b \ : \ b \in g(d;\delta)(\sigma)\} \\
\mathcal{T}(g;\gamma \cap d;\delta, \sigma) &\equiv \{a\hat{}\mathfrak{l}^d\hat{}b \ : \ b \in g(g;\gamma)(\sigma)\} \cup \\
&\qquad \{a\hat{}\mathfrak{r}^d\hat{}b \ : \ b \in g(d;\delta)(\sigma)\} \\
\mathcal{T}(g;\gamma \,;\, d;\delta, \sigma) &\equiv a\hat{}\mathfrak{c}\hat{}\mathcal{T}(\gamma, g) \cup \bigcup_{t \in \mathsf{leaf}(a\hat{}\mathcal{T}(\gamma,g))} \mathcal{T}(d;\delta, \lfloor t \rfloor_\sigma) \\
\mathcal{T}((g;\gamma)^*, \sigma) &\equiv \bigcup_{n<\omega} f^n(\{(a\hat{}\mathfrak{s}), (a\hat{}\mathfrak{g})\})
\end{aligned}
$$

where f^n is the n-fold composition of the function

$$f(Z) \overset{\text{def}}{=} Z \cup \bigcup_{t\hat{}\mathfrak{g}\in\text{leaf}(Z)} t\hat{}\mathfrak{g}\hat{}\mathcal{T}(\alpha, \lfloor t\hat{}\mathfrak{g}\rfloor_\sigma)\hat{}\{(a\hat{}\mathfrak{s}), (a\hat{}\mathfrak{g})\}$$

$$\mathcal{T}((g{:}\gamma)^\times, \sigma) \equiv \bigcup_{n<\omega} f^n(\{(a\hat{}\mathfrak{s}^d), (a\hat{}\mathfrak{g}^d)\})$$

where f^n is the n-fold composition of the function

$$f(Z) \overset{\text{def}}{=} Z \cup \bigcup_{t\hat{}\mathfrak{g}^d\in\text{leaf}(Z)} t\hat{}\mathfrak{g}^d\hat{}\mathcal{T}(\alpha, \lfloor t\hat{}\mathfrak{g}^d\rfloor_\sigma)$$

Leaves are the nodes of a tree with no children, i.e., any sequence l such that there is no sequence $l \cdot s$ where s is a single action/label in the tree.

G.2 Label-free Game Trees

To get back unlabeled dGL games operational semantics, we erase labels from the labeled semantics.

Definition G.2 (Erasing Labels). The function $\pi_\mathfrak{a}$ takes in a node or game tree and returns it with all subgame labels removed. First we define erasing labels for a single node. Here, $\pi_\mathfrak{a}$ accepts a node s (which is a sequence) and returns a label-free node (the sequence with labels dropped).

$$\pi_\mathfrak{a}(s) \equiv (a_i \ : \ a_i \in s,\ a_i \notin \text{subgame labels})$$

Erasing labels for a game tree t proceeds by erasing labels for every node, producing a label-free tree.

$$\pi_\mathfrak{a}(t) \equiv \{\pi_\mathfrak{a}(s) \ : \ s \in t \text{ and } \pi_\mathfrak{a}(s) \neq ()\}$$

If erasing labels results in an empty sequence (), it is discarded.

Definition G.3 (Label-free Semantics). The label-free game tree of a dGL game $a{:}\alpha$ results from erasing labels in the labeled sematics, i.e., $\pi_\mathfrak{a}(\mathcal{T}(a{:}\alpha, \sigma))$.

G.3 Demonic Subvalue Projection

Definition G.4 (Demonic subvalue projection). The *projection* of Demonic subvalue map S onto dGL game $a{:}\alpha$ with Demon winning condition ϕ, written $a{:}\alpha[]^{\exists}_{\downarrow} S$, is generated recursively per the structure of $a{:}\alpha$ as follows. If $a{:}\alpha$ has structure:

$$\begin{aligned}
a{:}x := \times \text{ then } x := \times\,;\ !\phi.\\
a{:}\{x' = f(x)\ \&\ Q\}^d \text{ then } \{x' = f(x)\ \&\ Q\}^d\,;\ !\phi\\
a{:}(g{:}\gamma \cap d{:}\delta) \text{ then } (!S(g)\,;\ g{:}\gamma[]^{\exists}_{\downarrow} S) \cap (!S(d)\,;\ d{:}\delta[]^{\exists}_{\downarrow} S)\\
a{:}(g{:}\gamma)^\times \text{ then } (!S(g)\,;\ g{:}\gamma[]^{\exists}_{\downarrow} S)^\times; !\phi\\
a{:}(g{:}\gamma; d{:}\delta) \text{ then } g{:}\gamma[]^{\exists}_{\downarrow} S\,;\ d{:}\delta[]^{\exists}_{\downarrow} S\\
a{:}(g{:}\gamma \cup d{:}\delta) \text{ then } g{:}\gamma[]^{\exists}_{\downarrow} S \cup d{:}\delta[]^{\exists}_{\downarrow} S \quad a{:}(g{:}\gamma)^* \text{ then } g{:}\gamma[]^{\exists}_{\downarrow} S^*\\
\text{atomic and not controlled by Demon, i.e.,}\\
\alpha \in \{x := e, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}^d\},\ \text{then } \alpha
\end{aligned}$$

G.4 Universal Projection of Demonic Subvalue Maps

Definition G.5 (Universal Projection). The *universal projection* of Demonic subvalue map S onto dGL game $a{:}\alpha$ with Demon winning condition ϕ, written $a{:}\alpha[]^{\forall}_{\downarrow} S$, is generated recursively per the

structure of $a{:}\alpha$ as follows. If $a{:}\alpha$ has structure:

$a{:}x := \otimes$ then $!(\exists x\, \phi)$; $x := *$; $?\phi$.

$a{:}\{x' = f(x)\ \&\ Q\}^d$, then $!\langle\alpha\rangle\phi$; $\{x' = f(x)\ \&\ Q\}$; $?\phi$.

$a{:}(g{:}\gamma \cap d{:}\delta)$, then $!(S(g) \vee S(d))$; $((?S(g)$; $g{:}\gamma[]\!\!\downarrow S) \cup (?S(d)$; $d{:}\delta[]\!\!\downarrow S))$.

$a{:}(g{:}\gamma)^\times$, then $!(S(g) \vee \phi)$; $(?S(g)$; $[]{:}\gamma\!\!\downarrow S$; $?S(a) \vee \phi)^*$; $?\phi$.

$a{:}(g{:}\gamma; d{:}\delta)$, then $g{:}\gamma[]\!\!\downarrow S$; $d{:}\delta[]\!\!\downarrow S$.

$a{:}(g{:}\gamma \cap d{:}\delta)$, then $g{:}\gamma[]\!\!\downarrow S \cap d{:}\delta[]\!\!\downarrow S$.

$a{:}(g{:}\gamma)^\times$, then $g{:}\gamma[]\!\!\downarrow S^\times$.

atomic and not controlled by Demon, i.e.,

$\alpha \in \{x := e, x := *, ?Q, !Q, \{x' = f(x)\ \&\ Q\}\}$, then α.

G.5 Strategy Set Generation

Def. 5.3 shows how an agent's Subvalue map lets it decide what actions to take at a given state. This section shows how to generate a set of strategies (as defined in dGL operational semantics) given a subvalue map by tracing through the game while keeping track of which strategy leads to what state, and using the policy to predict what actions to take at decision points.

Definition G.6 (Transformation to Strategy Set). The function $\mathcal{S}_{a:\alpha}(S)(b,\sigma)$ produces the set of Angel strategies to play game $a{:}\alpha$ following subvalue map S starting at subgame b, where σ represents the current state, where $\hat{}$ is prefix-closed concatenation as shown in Appendix G.1. If the structure of b is:

$b{:}x := e$ then $\{\{(x := e)\}\}$

$b{:}x := *$ then $\{p \ :\ p \in \mathcal{P}_{a:\alpha}(S)(b,\sigma)\}$

$b{:}x := \otimes$ then $\{\{(x := e^d) \ :\ e \in \mathbb{R}\}\}$

$b{:}\{x' = f(x)\ \&\ Q\}$ then $\{\{p\} \ :\ \mathcal{P}_{a:\alpha}(S)(b,\sigma)\}$

$b{:}\{x' = f(x)\ \&\ Q\}^d$ then

$$\{\{(x' = f(x) \& Q^d\ @\ t) \ :\ t \in \mathbb{R} \text{ where } \varphi(0) = \sigma(x) \text{ and } \forall s \in [0,t]\ \varphi'(s) = f(\varphi(s))\}\}$$

$b{:}?Q$ then $\{\{(?Q)\}\}$

$b{:}!Q$ then $\{\{(!Q)\}\}$

$b{:}(g{:}\gamma \cup d{:}\delta)$ then

$$\begin{cases} \{\} & \text{if } \mathcal{P}_{a:\alpha}(S)(b) = \{\} \\ \{\{\mathfrak{l}\hat{}t\} \ :\ t \in \mathcal{S}_{g:\gamma}(S)(g,\sigma)\} & \text{if } \mathcal{P}_{a:\alpha}(S)(b) = \{\mathfrak{l}\} \\ \{\{\mathfrak{r}\hat{}t\} \ :\ t \in \mathcal{S}_{d:\delta}(S)(d,\sigma)\} & \text{if } \mathcal{P}_{a:\alpha}(S)(b) = \{\mathfrak{r}\} \\ \{\{\mathfrak{l}\hat{}t\} \ :\ t \in \mathcal{S}_{g:\gamma}(S)(g,\sigma)\} \cup & \text{otherwise if } \mathcal{P}_{a:\alpha}(S)(b) = \{\mathfrak{l},\mathfrak{r}\} \\ \quad \{\{\mathfrak{r}\hat{}t\} \ :\ t \in \mathcal{S}_{d,\sigma:\delta}(S)(d)\} & \end{cases}$$

$b{:}(g{:}\gamma \cap d{:}\delta)$ then $\{\mathfrak{l}^d\hat{}t_l \cup \mathfrak{r}^d\hat{}t_r \ :\ t_l \in \mathcal{S}_{g:\gamma}(S)(g,\sigma) \text{ and } t_r \in \mathcal{S}_{d:\delta}(S)(d,\sigma)\}$

$b{:}(g{:}\gamma\,;\, d{:}\delta)$ then $\{t \cup \bigcup_{v \in \text{leaf}(t)} (v\hat{}u) \ :\ t \in \mathcal{S}_{g:\gamma}(S(\text{end} \mapsto S(d)))(g,\sigma)$ and $u \in \mathcal{S}_{d:\delta}(S)(d)(\lfloor v \rfloor_\sigma)\}$

$b{:}(g{;}\gamma)^*$ then $\bigcup_{n<\omega}\{t \;:\; t \in f^n(\mathcal{P}_{a{:}\alpha}(S)(b,\sigma)) \text{ and } \nexists u(u\hat{}\mathfrak{g} \in \text{leaf}(t))\}$

where f^n is n-fold composition of the function :

$$f(Z) = \bigcup_{t\in Z}\{t \cup \bigcup_{u\hat{}\mathfrak{g}\in\text{leaf}(t)} \left\{ \begin{array}{ll} \{u\hat{}\mathfrak{g}\hat{}v_u\} & \text{if } \lfloor u\hat{}\mathfrak{g}\hat{}v_u\rfloor_\sigma = \text{undefined} \\ \{u\hat{}\mathfrak{g}\hat{}v_u\hat{}o_{uv}\} & \text{otherwise} \end{array} \right\} :$$

$v_u \in \mathcal{S}_{g;\gamma}(S(\text{end} \mapsto S(a)))(g)(\lfloor u\rfloor_\sigma)$ and

$o_{uv} \in \mathcal{P}_{a{:}\alpha}(S)(b, \lfloor u\hat{}\mathfrak{g}\hat{}v_u\rfloor_\sigma)\}$

$b{:}(g{;}\gamma)^\times$ then $\bigcup_{n<\omega} f^n(\{\{\mathfrak{s}^d, \mathfrak{g}^d\}\})$

where f^n is n-fold composition of the function :

$$f(Z) = \bigcup_{t\in Z}\{t \cup \bigcup_{u\hat{}\mathfrak{g}^d\in\text{leaf}(t)} \{u\hat{}\mathfrak{g}^d\hat{}v\hat{}\{\mathfrak{s}^d, \mathfrak{g}^d\} \;:\; v \in \mathcal{S}_{a{:}\alpha}(S)(g)(\lfloor u\rfloor_\sigma)\}\}$$

with implicit closure under prefixes.

To reason only about the states reachable by playing the game trees, we should introduce a version of the game trees free of actions that do nothing.

Definition G.7 (Erasing skip Actions). Let the set of skip actions $\mathfrak{S}$ be

$$\mathfrak{S} = \{\mathfrak{g}, \mathfrak{s}, \mathfrak{l}, \mathfrak{r}, \mathfrak{c}, \mathfrak{g}^d, \mathfrak{s}^d, \mathfrak{l}^d, \mathfrak{r}^d\}$$

. The function $\pi_{\neg\mathfrak{S}}$ takes in a node or game tree and returns it with all labels and skip actions removed. First we define $\pi_{\neg\mathfrak{S}}$ for a single node. Here, $\pi_{\neg\mathfrak{S}}$ accepts a node s (which is a sequence) and returns a node (the sequence with labels dropped).

$$\pi_{\neg\mathfrak{S}}(s) \;\equiv\; (a_i \;:\; a_i \in s,\ a_i \notin \text{subgame labels and } a_i \notin \mathfrak{S})$$

Erasing labels and skip actions for a game tree t proceeds by erasing these for every node, producing a new tree.

$$\pi_{\neg\mathfrak{S}}(t) \;\equiv\; \{\pi_{\neg\mathfrak{S}}(s) \;:\; s \in t \text{ and } \pi_{\neg\mathfrak{S}}(s) \neq ()\}.$$

G.6 Inductive Demonic Subvalue Maps

Definition G.8 (Inductive Demonic subvalue maps). Let S be a map from the subgames of $a{:}\alpha$ to winning subregions. S is an *inductive Demonic subvalue map* for game $a{:}\alpha$, written $a{:}\alpha[]\models S$, when the following holds. If $a{:}\alpha$ has structure:

atomic, i.e., $\alpha \in \{x := e, x := *, x := \otimes, ?Q, !Q, \{x' = f(x)\ \&\ Q\}, \{x' = f(x)\ \&\ Q\}^d\}$ then $\models S(a) \to [\alpha]S(\text{end})$.

$a{:}(g{;}\gamma \cup d{:}\delta)$ then $\models S(a) \to S(g) \wedge S(d)$ and $g{;}\gamma[]\models S$ and $d{:}\delta[]\models S$.

$a{:}(g{;}\gamma \cap d{:}\delta)$ then $\models S(a) \to S(g) \vee S(d)$ and $g{;}\gamma[]\models S$ and $d{:}\delta[]\models S$.

$a{:}(g{;}\gamma\,;\, d{:}\delta)$ then $\models S(a) \to S(g)$ and $g{;}\gamma[]\models S(\text{end} \mapsto S(d))$ and $d{:}\delta[]\models S$.

$a{:}(g{;}\gamma)^*$ then $\models S(a) \to S(\text{end}) \wedge S(g)$ and $g{;}\gamma[]\models S(\text{end} \mapsto S(a))$.

$a{:}(g{;}\gamma)^\times$ then $\models S(a) \to [a{:}\alpha[]^{\exists}_{\downarrow}S]S(\text{end})$ and $g{;}\gamma[]\models S(\text{end} \mapsto S(a))$.

G.7 Game Prefix

This definition uses *the empty game* skip which can be treated like the game $?true$ in terms of effect on plays, except with no corresponding action in the operational semantics.

Definition G.9 (Game Prefix). The game suffix $a{:}\alpha_{:b}$ of subgame $b{:}\beta$ in subgames$(a{:}\alpha)$ is constructed as below. If $a{:}\alpha$ is $b{:}\beta$ and α is not a loop, then $a{:}\alpha_{:b}$ is the empty game, skip. If $b = a$ and $a{:}\alpha$ is a loop, then $a{:}\alpha_{:b}$ is $a{:}\alpha$. If b is end then $a{:}\alpha_{:b}$ is the entire game $a{:}\alpha$. Otherwise, if $a{:}\alpha$ has structure:

$$a{:}(g{:}\gamma)^* \text{ or } a{:}(g{:}\gamma)^\times \text{ then } a{:}\alpha_{:b} = a{:}\alpha\,;\,(g{:}\gamma_{:b})$$

$$a{:}(g{:}\gamma \cup d{:}\delta) \text{ or } a{:}(g{:}\gamma \cap d{:}\delta) \text{ then } \begin{cases} a{:}\alpha_{:b} = g{:}\gamma_{:b} & b \in \text{subgames}(g{:}\gamma) \\ a{:}\alpha_{:b} = d{:}\delta_{:b} & \text{otherwise} \end{cases}$$

$$a{:}(g{:}\gamma\,;\,d{:}\delta) \text{ then } \begin{cases} a{:}\alpha_{:b} = g{:}\gamma_{:b} & b \in \text{subgames}(g{:}\gamma) \\ a{:}\alpha_{:b} = g{:}\gamma\,;\,d{:}\delta_{:b} & \text{otherwise} \end{cases}$$

G.8 Solving Function for Inductive Demonic Subvalue Maps

$[]\text{map}(a{:}\alpha, \phi)$, defined below, computes an inductive Demonic subvalue map for game $a{:}\alpha$ and Demon winning condition ϕ. Operator $\uplus$ denotes the disjoint union of two subvalue maps.

$[]\text{map}(a{:}\alpha, \phi) := \{a \mapsto \text{simpl}([\alpha]\phi), \text{end} \mapsto \phi\}$ where $\alpha \in \{x := e, x := *, x := \otimes,$
$?Q, !Q, \{x' = f(x)\&Q\}, \{x' = f(x)\&Q\}^d\}$

$[]\text{map}(a{:}(g{:}\gamma \cap d{:}\delta), \phi) := S_1 \uplus S_2 \uplus \{a \mapsto S_1(g) \vee S_2(d)\}$
where $S_1 := []\text{map}(g{:}\gamma, \phi)$, $S_2 := []\text{map}(d{:}\delta, \phi) \setminus \text{end}$

$[]\text{map}(a{:}(g{:}\gamma \cup d{:}\delta), \phi) := S_1 \uplus S_2 \uplus \{a \mapsto S_1(g) \wedge S_2(d)\}$
where $S_1 := []\text{map}(g{:}\gamma, \phi)$, $S_2 := []\text{map}(d{:}\delta, \phi) \setminus \text{end}$

$[]\text{map}(a{:}(g{:}\gamma; d{:}\delta), \phi) := S_1 \uplus S_2 \uplus \{a \mapsto S_2(g)\}$
where $S_1 := []\text{map}(d{:}\delta, \phi)$, $S_2 := []\text{map}(g{:}\gamma, S_1(d)) \setminus \text{end}$

$[]\text{map}(a{:}(g{:}\gamma)^\times, \phi) := S \uplus \{a \mapsto \text{Inv}, \text{end} \mapsto \phi\}$ if $\langle a{:}\alpha \overset{\exists}{\downarrow} S\rangle\phi$
where $S := []\text{map}(g{:}\gamma, \text{Inv} \vee \phi) \setminus \text{end}$

$[]\text{map}(a{:}(g{:}\gamma)^*, \phi) := S \uplus \{a \mapsto \text{Inv}, \text{end} \mapsto \phi\}$ if $\models \text{Inv} \rightarrow S(g) \wedge \phi$
where $S := []\text{map}(g{:}\gamma, \text{Inv}) \setminus \text{end}$