Constructive Hybrid Games*

Rose Bohrer[1] and André Platzer[1,2]

[1] Computer Science Department, Carnegie Mellon University
rose.bohrer.cs@gmail.com,aplatzer@cs.cmu.edu
[2] Fakultät für Informatik, Technische Universität München

Abstract. Hybrid games combine discrete, continuous, and adversarial dynamics. Differential game logic (dGL) enables proving (classical) existence of winning strategies. We introduce *constructive differential game logic* (CdGL) for hybrid games, where proofs that a player can win the game correspond to *computable* winning strategies. This constitutes the logical foundation for synthesis of correct control and monitoring code for safety-critical cyber-physical systems. Our contributions include novel semantics as well as soundness and consistency.

Keywords: Game Logic, Constructive Logic, Hybrid Games, Dependent Types

1 Introduction

Differential Game Logic (dGL) provides a calculus for proving the (classical) existence of winning strategies for hybrid games [42], whose mixed discrete, continuous, and adversarial dynamics are compelling models for cyber-physical systems (CPSs). Classical existence does *not* necessarily imply that the resulting winning strategies are computable, however. To overcome this challenge, this paper introduces *Constructive Differential Game Logic* (CdGL) with a Curry-Howard correspondence: constructive proofs for constructive hybrid games correspond to programs implementing their winning strategies. We develop a new type-theoretic semantics which elucidates this correspondence and an operational semantics which describes the execution of strategies. Besides its theoretical appeal, this Curry-Howard interpretation provides the foundation for proof-driven synthesis methods, which excel at synthesizing expressive classes of games for which synthesis and correctness require interactive proof. Hybrid games are a compelling domain for proof-based synthesis both because many CPS applications are safety-critical or even life-critical, such as transportation systems, energy systems, and medical devices and because the combination of discrete, continuous, and adversarial dynamics makes verification and synthesis undecidable in both theory and practice. Our example model and proof, while short, lay the groundwork for future case studies.

* This research was sponsored by the AFOSR under grant number FA9550-16-1-0288 and the Alexander von Humboldt Foundation. The first author was also funded by an NDSEG Fellowship.

Challenges and Contributions. In addition to dGL [42], we build directly on Constructive Game Logic (CGL) [9] for discrete games. Compared to CGL, we target a domain with readily-available practical applications (hybrid games), and introduce new type-theoretic and operational semantics which complement the realizability semantics of CGL while making Curry-Howard particularly clear and providing a simple notion of strategy execution. We overcome the following challenges in the process:

- Our semantics must carefully capture the meaning of constructive hybrid game strategies, including strategies for differential equations (ODEs).
- Soundness must be justified *constructively*. We adapt previous arguments to use constructive analysis [6,12] by appealing to constructive formalizations of ODEs [17,34]. This adaptation to our new semantics makes it possible to simplify statements of some standard lemmas.
- We study 1D driving control as an example, which demonstrates the strengths of both games and constructivity. Games and constructivity both introduce uncertainties: A player is uncertain how their opponent will play, while constructive real-number comparisons are never sure of exact equality. These uncertainties demand more nuanced proof invariants, but these nuances improve our fidelity to real systems.

These contributions are of likely interest to several communities. Other constructive program logics could reuse our semantic approach. Our example uses reach-avoid proofs for hybrid games, a powerful, under-explored [48] approach.

2 Related Work

We discuss related works on games, constructive logic, and hybrid systems.

Games in Logic. Propositional GL was introduced by Parikh [39]. GL is a *program logic* in the spirit of Hoare calculi [26] or especially dynamic logics (DL) [47]: modalities capture the effect of game execution. GLs are unique in their clear delegation of strategy to the *proof* language rather than the *model* language, allowing succinct game specifications with sophisticated winning strategies. Succinct specifications are important: specifications are *trusted* because proving the *wrong theorem* would not ensure correctness. Relatives without this separation include SL [14], ATL [2], CATL [27], SDGL [23], structured strategies [49], DEL [3,5,56], evidence logic [4], and Angelic Hoare Logic [35].

Constructive Modal Logics. We are interested in the semantics of games, thus we review constructive modal semantics generally. This should not be confused with game semantics [1], which give a semantics to programs *in terms of* games. The main semantic approaches for constructive modal logics are intuitionistic Kripke semantics [58] and realizability semantics [38,32]. CGL [9] used a realizability semantics which operate on a state, reminiscent of state in Kripke semantics, whereas we interpret CdGL formulas into type theory.

Modal Curry-Howard is relatively little-studied, and each author has their own emphasis. Explicit proof terms are considered for CGL [9] and a small fragment thereof [30]. Others [59,18,13] focus on intuitionistic semantics for their logics, fragments of CGL. Our semantics should be of interest for these fragments. We omit proof terms for space. CdGL proof terms would extend CGL proof terms [9] with a constructive version of existing classical ODE proof terms [8]. Propositional modal logic [37] has been interpreted as a type system.

Hybrid Systems Synthesis. Hybrid games synthesis is one motivation of this work. Synthesis of hybrid *systems* (1-player games) is an active area. The unique strength of proof-based synthesis is expressiveness: it can synthesize every provable system. CdGL proofs support first-order regular games with first-order (e.g., semi-algebraic) initial and goal regions. While synthesis and proof are both undecidable, interactive proof for undecidable logics is well-understood. The ModelPlex [36] synthesizer for CdGL's classical systems predecessor dL [44] recently added [11] proof-based synthesis to improve expressiveness. CdGL aims to provide a computational foundation for a more systematic proof-based synthesizer in the more general context of games.

Fully automatic synthesis, in contrast, restricts itself to small fragments in order to sidestep undecidability. Studied classes include rectangular hybrid games [25], switching systems [52], linear systems with polyhedral sets [31,52], and discrete abstractions [21,20]. A well-known [55] *systems* synthesis approach translates specifications *into* finite-alternation games. Arbitrary first-order games are our *source* rather than *target* language. Their approach is only known to terminate for simpler classes [50,51].

3 Constructive Hybrid Games

Hybrid games in CdGL are 2-player, zero-sum, and perfect-information, where continuous subgames are ordinary differential equations (ODEs) whose duration is chosen by a player. Hybrid games should not be confused with *differential games* which compete continuously [29,43]. The players considered in this paper are Angel and Demon where the player currently controlling choices is always called Angel, while the player waiting to play is always called Demon. For any game α and formula ϕ, the modal formula $\langle\alpha\rangle\phi$ says Angel can play α to ensure postcondition ϕ, while $[\alpha]\phi$ says Demon can play α to ensure postcondition ϕ. These generalize safety and liveness modalities from DL. Dual games α^d, unique to GLs, take turns by switching the Angel and Demon roles in game α. The Curry-Howard interpretation of a proof of a CdGL modality $\langle\alpha\rangle\phi$ or $[\alpha]\phi$ is a program which performs each player's winning strategy. Games can have several winning strategies, each corresponding to a different proof and a different program.

3.1 Syntax of CdGL

We introduce the language of CdGL with three classes of expressions e: terms f, g, games α, β, and formulas ϕ, ψ. We characterize terms semantically for the sake

of generality: a shallow embedding of CdGL inside a proof assistant might use the host language for terms. For games and formulas, we find it more convenient to explicitly and syntactically define a closed language.

A (scalar) semantic term is a function from states to reals, which are understood constructively *à la* Bishop [6,12]. We use Bishop-style real analysis because it preserves many classical intuitions (e.g., uncountability) about $\mathbb{R}$ while ensuring computability. Type-2 [57] computability requires that all *functions on real numbers* are computable to arbitrary precision if represented as streams of bits, yet computability *does not* require that variables range over *only* computable reals. It is a theorem [57] that all such computable functions are continuous, but not always Lipschitz-continuous nor differentiable.

We introduce commonly used term constructs, which are not exhaustive because the language of terms is open. The simplest terms are *game variables* $x, y \in \mathcal{V}$ where $\mathcal{V}$ is the (at most countable) set of variable identifiers. The game variables, which are mutable, contain the state of the game, which is globally scoped. For every base game variable x there is a primed counterpart x' whose purpose within an ODE is to track the time derivative of x. Real-valued terms f, g are simply type-2 computable functions, usually from states to reals. It is occasionally useful for f to return a tuple of reals, which are computable when every component is computable. Since terms are functions, operators are combinators: $f + g$ is a function which sums the results of f and g.

Definition 1 (Terms). *A* term f, g *is any computable function over the game state. The following constructs appear in this paper:*

$$f, g \ ::= \ \cdots \mid c \mid x \mid f + g \mid f \cdot g \mid f/g \mid \min(f, g) \mid \max(f, g) \mid (f)'$$

where $c \in \mathbb{R}$ is a real literal, x a game variable, $f + g$ a sum, $f \cdot g$ a product, and f/g is real division of f by g. Divisors g are assumed to be nonzero. Minimum and maximum of terms f and g are written $\min(f, g)$ and $\max(f, g)$. Any differentiable term f has a definable (Section 4.2) spatial differential term $(f)'$, which agrees with the time derivative within an ODE.

CdGL is constructive, so Angel strategies make choices computably. Until his turn, Demon just observes Angel's choices, and does not care whether Angel made them computably. We discuss game-playing informally here, then formally in Section 4. In red are the ODE and dual games, which respectively distinguish hybrid games from discrete games and games from systems.

Definition 2 (Games). *The set of* games α, β *is defined recursively as such:*

$$\alpha, \beta \ ::= \ ?\phi \mid x := f \mid x := * \mid x' = f \,\& \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \alpha^d$$

The *test game* $?\phi$, is a no-op if Angel proves ϕ, else Demon wins by default since Angel "broke the rules". A deterministic assignment $x := f$ updates game variable x to the value of term f. Nondeterministic assignments $x := *$ ask Angel to compute the new value of $x : \mathbb{R}$, i.e., Angel's strategy for $x := *$ is a term whose value is assigned to x. The ODE game $x' = f \,\&$ evolves ODE $x' = f$ for duration

$d \geq 0$ chosen by Angel such that Angel proves the domain constraint formula is true throughout. We require that term f is effectively-locally-Lipschitz on domain , meaning that at every state satisfying , a neighborhood and coefficient L can be constructed such that L is a Lipschitz constant of f in the neighborhood. Effective local Lipschitz continuity guarantees unique solutions exist by constructive Picard-Lindelöf [34]. ODEs are explicit-form, so no primed variable y' for $y \in \mathcal{V}$ is mentioned in f or . Systems of ODEs are supported, we present single equations for readability. In the choice game $\alpha \cup \beta$, Angel chooses whether to play game α or game β. In the sequential composition game $\alpha; \beta$, game α is played first, then β from the resulting state. In the repetition game α^*, Angel chooses after each repetition of α whether to continue playing, but must not repeat α infinitely. The exact number of repetitions is not known in advance, because it may depend on Demon's reactions. In the dual game α^d, Angel takes the Demon role and vice-versa while playing α. Demon strategies "wait" until a dual game α^d is encountered, then play an Angelic strategy for α. We parenthesize games with braces $\{\alpha\}$ when necessary.

Definition 3 (CdGL Formulas). *The CdGL* formulas ϕ *(also* ψ*) are:*

$$\phi \ ::= \ \langle\alpha\rangle\phi \mid [\alpha]\phi \mid f \sim g$$

Above, $f \sim g$ is a comparison formula for $\sim \in \{\leq, <, =, \neq, >, \geq\}$. The defining formulas of CdGL (and GL) are the modalities $\langle\alpha\rangle\phi$ and $[\alpha]\phi$. These mean that Angel or Demon respectively have a *constructive* strategy to play hybrid game α and prove postcondition ϕ. We do not develop modalities for existence of classical strategies because those cannot be synthesized to executable code.

Standard connectives are defined from games and comparisons. Verum ($\mathtt{tt}$) is defined $1 > 0$ and falsum ($\mathtt{ff}$) is $0 > 1$. Conjunction $\phi \wedge \psi$ is defined $\langle ?\phi\rangle\psi$, disjunction $\phi \vee \psi$ is defined $\langle ?\phi \cup ?\psi\rangle\mathtt{tt}$, and implication $\phi \rightarrow \psi$ is defined $[?\phi]\psi$. Real quantifiers $\forall x\, \phi$ and $\exists x\, \phi$ are defined $[x := *]\phi$ and $\langle x := *\rangle\phi$, respectively. As usual, equivalence $\phi \leftrightarrow \psi$ reduces to $(\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$, negation $\neg\phi$ is defined as $\phi \rightarrow \mathtt{ff}$, and inequality is defined by $f \neq g \equiv \neg(f = g)$. Semantics and proof rules are needed only for core constructs, but we use derived constructs when they improve readability. Keep these definitions in mind, because the semantics and rules for some game connectives mirror first-order connectives.

For convenience, we also write derived operators where Demon is given control of a single choice before returning control to Angel. The *Demonic choice* $\alpha \cap \beta$, defined $\{\alpha^d \cup \beta^d\}^d$, says Demon chooses which branch to take, but Angel controls the subgames. *Demonic repetition* $\alpha^\times$ is defined likewise by $\{\{\alpha^d\}^*\}^d$.

We write ϕ_x^y (likewise for α and f) for the *renaming* of variable x for y and vice versa in formula ϕ, and write ϕ_x^f for the result of *substitution* of term f for game variable x in ϕ, if the substitution is admissible (Definition 12 on page 14).

3.2 Example Game

We give an example game and theorem statements, proven in [10]. Automotive systems are a major class of CPS. As a simple indicative example we consider

time-triggered 1-dimensional driving with adversarial timing. For maximum time T between control cycles, we let Demon choose any duration in $[0,T]$. When we need to prohibit pathological "Zeno" behaviors while keeping constraints realistic, we can further restrict $t \in [T/2,T]$.

We write x for the current position of the car, v for its velocity, a for the acceleration, $A > 0$ for the maximum positive acceleration, and $B > 0$ for the maximum braking rate. We assume $x = v = 0$ initially to simplify arithmetic. In time-triggered control, the controller runs at least once every $T > 0$ time units. Time and physics are continuous, T gives an upper bound on how often the controller runs. Local clock t marks the current time within the current timestep, then resets at each step. The control game (ctrl) says Angel can pick any acceleration a that is physically achievable ($-B \leq a \leq A$). The clock t is then reinitialized to 0. The plant game (plant) says Demon can evolve physics for duration $t \in [0,T]$ such that $v \geq 0$ throughout, then returns control to Angel.

Typical theorems in DLs and GLs are *safety* and *liveness*: are unsafe states always avoided and are desirable goals eventually reached? Safety and liveness of the 1D *system* has been proved previously: safe driving (safety) never goes past goal g, while live driving eventually reaches g (liveness).

$$\begin{aligned}
\mathsf{pre} &\equiv T > 0 \land A > 0 \land B > 0 \land v = 0 \land x = 0 \quad \mathsf{post} \equiv (g = x \land v = 0)\\
\mathsf{ctrl} &\equiv a := *;\ ?-B \leq a \leq A;\ t := 0\\
\mathsf{plant} &\equiv \{t' = 1, x' = v, v' = a \,\&\, t \leq T \land v \geq 0\}^d\\
\mathsf{safety} &\equiv \mathsf{pre} \to \langle (\mathsf{ctrl};\mathsf{plant})^\times \rangle\, x \leq g\\
\mathsf{liveness} &\equiv \mathsf{pre} \to \langle (\mathsf{ctrl};\mathsf{plant};\{?t \geq T/2\}^d)^* \rangle\, x \geq g
\end{aligned}$$

Liveness theorem liveness requires a lower time bound ($\{?t \geq T/2\}^d$) to rule out Zeno strategies where Demon "cheats" by exponentially decreasing durations to essentially freeze the progress of time. The limit $t \geq T/2$ is chosen for simplicity. Safety theorem safety omits this constraint because even Zeno behaviors are safe.

Safety and liveness theorems, if designed carelessly, have trivial solutions including but not limited to Zeno behaviors. It is safe to remain at $x = 0$ and is live to maintain $a = A$, but not vice-versa. In contrast to DLs, GLs easily express the requirement that *the same* strategy is both safe and live: we must remain safe *while* reaching the goal. We use this *reach-avoid* specification because it is immune to trivial solutions. We give a new reach-avoid result for 1D driving.

Example 4 (Reach-avoid). The following is provable in dGL and CdGL:

$$\mathsf{reachAvoid} \equiv \mathsf{pre} \to \langle \{\mathsf{ctrl};\mathsf{plant};?x \leq g;\{?t \geq T/2\}^d\}^* \rangle \mathsf{post}$$

Angel *reaches* $g = x \land v = 0$ while safely *avoiding* states where $x \leq g$ does not hold. Angel is safe at *every* iteration for *every* time $t \in [0,T]$, thus safe *throughout* the game. The (dual) test $?t \geq T/2$ appears second, allowing Demon to win if Angel violates safety during $t < T/2$.

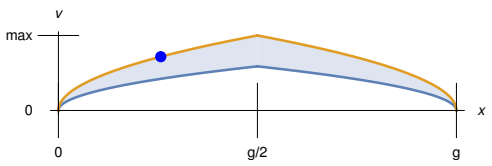

Fig. 1. Safe driving envelope

1D driving is well-studied for classical systems, but the constructive reach-avoid proof [10] is subtle. The proof constructs an envelope of safe upper and live lower bounds on velocity as a function of position (Figure 1). The blue point indicates where Angel must begin to brake to ensure time-triggered safety. It is surprising that Angel can achieve postcondition $g = x \wedge v = 0$, given that trichotomy ($f < g \vee f = g \vee f > g$) is constructively invalid. The key [10] is that comparison *terms* $\min(f, g)$ and $\max(f, g)$ *are* exact in Type 2 computability where bits of min and max may be computed lazily. Our exact result encourages us that constructivity is not overly burdensome in practice. When decidable comparisons ($f < g + \delta \vee f > g$) *are* needed, the alternative is a weaker guarantee $g - \varepsilon \leq x \leq g$ for parameter $\varepsilon > 0$. This relaxation is often enough to make the theorem provable, and reflects the fact that real agents only expect to reach their goal within finite precision.

4 Type-theoretic Semantics

In this section, we define the semantics of hybrid games and game formulas in type theory. We start with assumptions on the underlying type theory.

4.1 Type Theory Assumptions

We assume a Calculus of Inductive and Coinductive Constructions (CIC)-like type theory [15,16,54] with polymorphism and dependency. We write M for terms and $\Delta \vdash M : \tau$ to say M has type τ in CIC context Δ. We assume first-class (indexed [19]) inductive and coinductive types. We write τ for type families and κ for kinds: type families inhabited by other type families. Inductive type families are written $\mu t : \kappa.\, \tau$, which denotes the *smallest* solution $\mathtt{ty}$ of kind κ to the fixed-point equation $\mathtt{ty} = \tau_t^{\mathtt{ty}}$. Coinductive type families are written $\rho t : \kappa.\, \tau$, which denotes the *largest* solution $\mathtt{ty}$ of kind κ to the fixed-point equation $\mathtt{ty} = \tau_t^{\mathtt{ty}}$. Type-expression τ must be monotone in t so smallest and largest solutions exist by Knaster-Tarski [24, Thm. 1.12]. Proof assistants like Coq reject definitions where monotonicity requires nontrivial proof; we did not mechanize our proofs because they use such definitions.

We use one predicative universe which we write $\mathbb{T}$ and Coq writes `Type 0`. Predicativity is an important assumption because our semantic definition is a large elimination, a feature known to interact dangerously with impredicativity. We write $\Pi x : \tau_1.\, \tau_2$ for a dependent function type with argument named x of type τ_1 and where return type τ_2 may mention x. We write $\sum x : \tau_1.\, \tau_2$ for a dependent pair type with left component named x of type τ_1 and right component of type τ_2, possibly mentioning x. These specialize to the simple function $\tau_1 \Rightarrow \tau_2$ and product types $\tau_1 * \tau_2$ respectively when x is not mentioned in τ_2. Lambdas

$(\lambda x : \tau. M)$ inhabit dependent function types. Pairs (M, N) inhabit dependent pair types. Application is $M\ N$. Let-binding unpacks pairs, whose left and right projection are $\pi_L M$ and $\pi_R M$. We write $\tau_1 + \tau_2$ for a disjoint union inhabited by $\ell \cdot M$ and $r \cdot M$, and write $\mathsf{case}\ A\ \mathsf{of}\ p \Rightarrow B \mid q \Rightarrow C$ for its case analysis.

We assume a real number type $\mathbb{R}$ and a Euclidean state type $\mathfrak{S}$. The positive real numbers are written $\mathbb{R}_{>0}$, nonnegative reals $\mathbb{R}_{\geq 0}$. We assume scalar and vector sums, products, inverses, and units. States s, t support operations $s\ x$ and $\mathsf{set}\ s\ x\ v$ which respectively retrieve the value of variable x in $s : \mathfrak{S}$ or update it to v. The usual axioms of setters and getters [22] are satisfied. We write $\mathfrak{s}$ for the distinguished variable of type $\mathfrak{S}$ representing the current state. We will find it useful to consider the semantics of an expression both at current state $\mathfrak{s}$ and at states s, t defined in terms of $\mathfrak{s}$ (e.g., $\mathsf{set}\ \mathfrak{s}\ x\ 5$).

4.2 Semantics of CdGL

Terms f, g are type-theoretic functions of type $\mathfrak{S} \Rightarrow \mathbb{R}$. We will need differential terms $(f)'$, a definable term construct when f is differentiable. Not every term f need be differentiable, so we give a *virtual* definition, defining when $(f)'$ is equal to some term g. If $(f)'$ does not exist, then $(f)' = g$ is not provable. We define the (total) differential as the Euclidean dot product $(\cdot)$ of the gradient (variable name: ∇) with s', which is the vector of values $s\ x'$ assigned to primed variables x'. To show that ∇ is the gradient, we define the gradient as a limit, which we express in (ε, δ) style. In this definition, f and g are scalar-valued, and the minus symbol is used for both scalar and vector difference.

$$\begin{aligned}((f)'\ s = g\ s) \equiv\ & \sum \nabla : \mathbb{R}^{|s'|}. (g\ s = \nabla \cdot s') * \Pi \varepsilon : \mathbb{R}_{>0}. \sum \delta : \mathbb{R}_{>0}. \Pi r : \mathfrak{S}. \\ & (\|r - s\| < \delta) \Rightarrow |f\ r - f\ s - \nabla \cdot (r - s)| \leq \varepsilon \|r - s\|\end{aligned}$$

For practical proofs, a library of standard rules for automatic, syntactic differentiation of common arithmetic operations [7] could be proven.

The interpretation $\ulcorner \phi \urcorner : \mathfrak{S} \Rightarrow \mathbb{T}$ of formula ϕ is a predicate over states. A predicate of kind $\mathfrak{S} \Rightarrow \mathbb{T}$ is also understood as a *region*, e.g., $\ulcorner \phi \urcorner$ is the region containing states where ϕ is provable. A CdGL context Γ is interpreted over a uniform state term $s : \mathfrak{S}$ where $\mathfrak{s} : \mathfrak{S} \vdash s : \mathfrak{S}$, i.e., s usually mentions $\mathfrak{s}$. We define $\ulcorner \Gamma \urcorner(s)$ to be the CIC context containing $\mathfrak{s} : \mathfrak{S}$ and $\ulcorner \phi \urcorner\ s$ for each $\phi \in \Gamma$. The sequent $(\Gamma \vdash \phi)$ is *valid* if there exists M where $\ulcorner \Gamma \urcorner(\mathfrak{s}) \vdash M : (\ulcorner \phi \urcorner\ \mathfrak{s})$. Formula ϕ is *valid* iff sequent $(\cdot \vdash \phi)$ is valid. That is, a valid formula is provable in every state with a common proof term M. The witness may inspect the state, but must do so constructively. Formula semantics employ the Angelic and Demonic semantics of games, which determine how to win a game α whose postcondition is ϕ. We write $\langle\!\langle \alpha \rangle\!\rangle : (\mathfrak{S} \Rightarrow \mathbb{T}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T})$ for the Angelic semantics of α and $[[\alpha]] : (\mathfrak{S} \Rightarrow \mathbb{T}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T})$ for its Demonic semantics.

Definition 5 (Formula semantics). *Angel and Demon strategies for a hybrid game α with goal region P are inhabitants of $\langle\!\langle \alpha \rangle\!\rangle\ P$ and $[[\alpha]]\ P$, respectively.*

$$\ulcorner [\alpha]\phi \urcorner\ s = [[\alpha]]\ \ulcorner \phi \urcorner\ s \quad \ulcorner \langle \alpha \rangle \phi \urcorner\ s = \langle\!\langle \alpha \rangle\!\rangle\ \ulcorner \phi \urcorner\ s \quad \ulcorner f \sim g \urcorner\ s = ((f\ s) \sim (g\ s))$$

Modality $\langle\alpha\rangle\phi$ is provable in s when $\langle\!\langle\alpha\rangle\!\rangle\ \ulcorner\phi\urcorner\ s$ is inhabited so Angel has an α strategy from s to reach region $\ulcorner\phi\urcorner$ on which ϕ is provable. Modality $[\alpha]\phi$ is provable in s when $[[\alpha]]\ \ulcorner\phi\urcorner\ s$ is inhabited so Demon has an α strategy from s to reach region $\ulcorner\phi\urcorner$ on which ϕ is provable. For $\sim \in \{\leq, <, =, \neq, >, \geq\}$, the values of f and g are compared at state s in $f \sim g$. The game and formula semantics are simultaneously inductive. In each case, the connectives which define $\langle\!\langle\alpha\rangle\!\rangle$ and $[[\alpha]]$ are duals, because $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are dual. Below, P refers to the goal region of the game and s to the initial state.

Definition 6 (Angel semantics). *We define* $\langle\!\langle\alpha\rangle\!\rangle : (\mathfrak{S} \Rightarrow \mathbb{T}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T})$ *inductively (by a large elimination) on* α*:*

$$\begin{aligned}
\langle\!\langle ?\psi\rangle\!\rangle\ P\ s &= \ulcorner\psi\urcorner\ s * P\ s \\
\langle\!\langle x := f\rangle\!\rangle\ P\ s &= P\ (\mathsf{set}\ s\ x\ (f\ s)) \\
\langle\!\langle x := *\rangle\!\rangle\ P\ s &= \sum v : \mathbb{R}.\ P\ (\mathsf{set}\ s\ x\ v) \\
\langle\!\langle \alpha \cup \beta\rangle\!\rangle\ P\ s &= \langle\!\langle\alpha\rangle\!\rangle\ P\ s \mathbin{+} \langle\!\langle\beta\rangle\!\rangle\ P\ s \\
\langle\!\langle \alpha;\beta\rangle\!\rangle\ P\ s &= \langle\!\langle\alpha\rangle\!\rangle\ (\langle\!\langle\beta\rangle\!\rangle\ P)\ s \\
\langle\!\langle \alpha^d\rangle\!\rangle\ P\ s &= [[\alpha]]\ P\ s \\
\langle\!\langle x' = f\,\&\,\rangle\!\rangle\ P\ s &= \sum d : \mathbb{R}_{\geq 0}.\ \sum sol : [0,d] \Rightarrow \mathbb{R}. \\
&\quad (sol, s, d \vDash x' = f) \\
&\quad * (\Pi t : [0,d].\ \ulcorner\neg\urcorner\ (\mathsf{set}\ s\ x\ (sol\ t))) \\
&\quad * P\ (\mathsf{set}\ s\ (x, x') \\
&\qquad (sol\ d, f\ (\mathsf{set}\ s\ x\ (sol\ d))))
\end{aligned}$$

$$\langle\!\langle \alpha^*\rangle\!\rangle\ P\ s = \big(\mu\tau' : (\mathfrak{S} \Rightarrow \mathbb{T}).\ \lambda t : \mathfrak{S}.\ (P\ t \Rightarrow \tau'\ t) \mathbin{+} (\langle\!\langle\alpha\rangle\!\rangle\ \tau'\ t \Rightarrow \tau'\ t)\big)\ s$$

Angel wins $\langle ?\psi\rangle P$ by proving both ψ and P at s. Angel wins the deterministic assignment $x := f$ by performing the assignment, then proving P. Angel wins nondeterministic assignment $x := *$ by constructively choosing a value v to assign, then proving P. Angel wins $\alpha \cup \beta$ by choosing between playing α or β, then winning that game. Angel wins $\alpha;\beta$ if she wins α with the postcondition of winning β. Angel wins α^d if she wins α in the Demon role. Angel wins ODE game $x' = f\,\&$ by choosing some solution sol of some duration d which satisfies the ODE and domain constraint throughout and the postcondition ϕ at time d. While top-level postconditions rarely mention x', intermediate invariant steps do, thus x and x' both are updated in the postcondition. The construct $(sol, s, d \vDash x' = f)$, saying sol solves $x' = f$ from state s for time d, is defined:

$$(sol, s, d \vDash x' = f) \equiv ((s\ x = sol\ 0) * \Pi r : [0,d].\ ((sol)'\ r = f\ (\mathsf{set}\ s\ x\ (sol\ r))))$$

Note that variable sol stands for a function of the host theory, all of which are computable and therefore continuous. When $(sol, s, d \vDash x' = f)$ holds, sol is also continuously differentiable. Constructive Picard-Lindelöf [34] constructs a solution for every effectively-locally-Lipschitz ODEs, which need not have a closed form. The proof calculus we introduce in Section 5 includes both solution-based proof rules, which are useful for ODEs with simple closed forms, and invariant-based rules, which enable proof even when closed forms do not exist.

Angel strategies for α^* are inductively defined: either choose to stop the loop and prove P now, else play a round of α before repeating inductively. By Knaster-Tarski [24, Thm. 1.12], this least fixed point exists because the interpretation of a game is monotone in its postcondition (Lemma 7).

Lemma 7 (Monotonicity). *Let* $P, Q : \mathfrak{S} \Rightarrow \mathbb{T}$. *If* $\mathfrak{s} : \mathfrak{S}, P\ s \vdash M : Q\ s$ *then there exists a term* N *such that* $\mathfrak{s} : \mathfrak{S}, [[\alpha]]\ P\ s \vdash N : [[\alpha]]\ Q\ s$

Definition 8 (Demon semantics). *We define* $[[\alpha]] : (\mathfrak{S} \Rightarrow \mathbb{T}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T})$ *inductively (by a large elimination) on* α*:*

$$
\begin{aligned}
[[?\psi]]\ P\ s &= \ulcorner \psi \urcorner\ s \Rightarrow P\ s \\
[[x := f]]\ P\ s &= P\ (\mathsf{set}\ s\ x\ (f\ s)) \\
[[x := *]]\ P\ s &= \Pi v : \mathbb{R}.\ P\ (\mathsf{set}\ s\ x\ v) \\
[[\alpha \cup \beta]]\ P\ s &= [[\alpha]]\ P\ s\ \mathbin{*}\ [[\beta]]\ P\ s \\
[[\alpha; \beta]]\ P\ s &= [[\alpha]]\ ([[\beta]]\ P)\ s
\end{aligned}
\qquad
\begin{aligned}
[[\alpha^d]]\ P\ s &= \langle\!\langle \alpha \rangle\!\rangle\ P\ s \\
[[x' = f\,\&\,]]\ P\ s &= \Pi d : \mathbb{R}_{\geq 0}.\ \Pi sol : [0, d] \Rightarrow \mathbb{R}. \\
&\quad (sol, s, d \vDash x' = f) \\
&\Rightarrow (\Pi t : [0, d].\ \ulcorner \mathsf{set}\ s\ x\ (sol\ t) \urcorner) \\
&\Rightarrow P\ (\mathsf{set}\ s\ (x, x') \\
&\qquad (sol\ d, f\ (\mathsf{set}\ s\ x\ (sol\ d))))
\end{aligned}
$$

$$
[[\alpha^*]]\ P\ s = \big(\rho\tau' : (\mathfrak{S} \Rightarrow \mathbb{T}).\ \lambda t : \mathfrak{S}.\ (\tau'\ t \Rightarrow [[\alpha]]\ \tau'\ t)\ \mathbin{*} (\tau'\ t \Rightarrow P\ t)\big)\ s
$$

Demon wins $[?\psi]P$ by proving P under assumption ψ, which Angel must provide (Section 7). Demon's deterministic assignment is identical to Angel's. Demon wins $x := *$ by proving ψ for *every* choice of x. Demon wins $\alpha \cup \beta$ with a pair of winning strategies. Demon wins $\alpha; \beta$ by winning α with a postcondition of winning β. Demon wins α^d if he can win α after switching roles with Angel. Demon wins $x' = f\,\&$ if for an arbitrary duration and arbitrary solution which satisfy the domain constraint, he can prove the postcondition. Demon wins $[\alpha^*]P$ if he can prove P no matter how many times Angel makes him play α. Demon repetition strategies are coinductive using some invariant τ'. When Angel decides to stop the loop, Demon responds by proving P from τ'. Whenever Angel chooses to continue, Demon proves that τ' is preserved. Greatest fixed points exist by Knaster-Tarski [24, Thm. 1.12] using Lemma 7.

It is worth comparing the Angelic and Demonic semantics of $x := *$. An Angel strategy says how to compute x. A Demon strategy simply accepts $x \in \mathbb{R}$ as its input, even uncomputable numbers. This is because Angel strategies supply a computable real while Demon acts with computable outputs given real *inputs*. In general, each strategy is constructive but permits its opponent to play classically. In the cyber-physical setting, the opponent is indeed rarely a computer.

5 Proof Calculus

To enable direct syntactic proof, we give a natural deduction-style system for CdGL. We write $\Gamma = \psi_1, \ldots, \psi_n$ for a context of formulas and $\Gamma \vdash \phi$ for the natural-deduction sequent with conclusion ϕ and context Γ. We begin with rules shared by CGL [9] and CdGL, then give the ODE rules. We write $\Gamma^{\frac{y}{x}}$ for the renaming of game variable x to y and vice versa in context Γ. Likewise Γ^f_x is the substitution of term f for game variable x. To avoid repetition, we write $\langle\!|\alpha|\!\rangle\phi$ to indicate that the same rule applies for $\langle\alpha\rangle\phi$ and $[\alpha]\phi$. These rules write $[\![\alpha]\!]\phi$ for the dual of $\langle\!|\alpha|\!\rangle\phi$. We write $\mathrm{FV}(e)$, $\mathrm{BV}(\alpha)$, and $\mathrm{MBV}(\alpha)$ for the free variables of expression e, bound variables of game α, and must-bound variables of game α

$$([\cup]\mathrm{I})\ \frac{\Gamma \vdash [\alpha]\phi \quad \Gamma \vdash [\beta]\phi}{\Gamma \vdash [\alpha \cup \beta]\phi} \qquad ([\cup]\mathrm{E1})\ \frac{\Gamma \vdash [\alpha \cup \beta]\phi}{\Gamma \vdash [\alpha]\phi} \qquad ([\cup]\mathrm{E2})\ \frac{\Gamma \vdash [\alpha \cup \beta]\phi}{\Gamma \vdash [\beta]\phi}$$

$$(\langle\cup\rangle\mathrm{I1})\ \frac{\Gamma \vdash \langle\alpha\rangle\phi}{\Gamma \vdash \langle\alpha \cup \beta\rangle\phi} \qquad (\langle\cup\rangle\mathrm{I2})\ \frac{\Gamma \vdash \langle\beta\rangle\phi}{\Gamma \vdash \langle\alpha \cup \beta\rangle\phi} \qquad (\mathrm{hyp})\ \frac{}{\Gamma, \phi \vdash \phi}$$

$$(\langle?\rangle\mathrm{I})\ \frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \langle?\phi\rangle\psi} \qquad (\langle?\rangle\mathrm{E1})\ \frac{\Gamma \vdash \langle?\phi\rangle\psi}{\Gamma \vdash \phi} \qquad (\langle?\rangle\mathrm{E2})\ \frac{\Gamma \vdash \langle?\phi\rangle\psi}{\Gamma \vdash \psi}$$

$$([?]\mathrm{I})\ \frac{\Gamma, \phi \vdash \psi}{\Gamma \vdash [?\phi]\psi} \qquad ([?]\mathrm{E})\ \frac{\Gamma \vdash [?\phi]\psi \quad \Gamma \vdash \phi}{\Gamma \vdash \psi}$$

$$(\langle\cup\rangle\mathrm{E})\ \frac{\Gamma \vdash \langle\alpha \cup \beta\rangle\phi \quad \Gamma, \langle\alpha\rangle\phi \vdash \psi \quad \Gamma, \langle\beta\rangle\phi \vdash \psi}{\Gamma \vdash \psi}$$

Fig. 2. CdGL proof calculus: Propositional game rules

respectively, i.e., variables which *might* influence the meaning of an expression, might be modified during game execution, or are written during *every* execution.

Figure 2 gives the propositional game rules. Rule [?]E is modus ponens and [?]I is implication introduction because $[?\phi]\psi$ is implication. Angelic choices are disjunctions introduced by $\langle\cup\rangle$I1 and $\langle\cup\rangle$I2 and case-analyzed by $\langle\cup\rangle$E. Angelic tests and Demonic choices are conjunctions introduced by $\langle?\rangle$I and $[\cup]$I, eliminated by $\langle?\rangle$E1, $\langle?\rangle$E2, $[\cup]$E1, and $[\cup]$E2. Rule hyp applies an assumption.

$$([:*]\mathrm{I})\ \frac{\Gamma\frac{y}{x} \vdash \phi}{\Gamma \vdash [x := *]\phi} \qquad ([:*]\mathrm{E})\ \frac{\Gamma \vdash [x := *]\phi}{\Gamma \vdash \phi^f_x}$$

$$(\langle:*\rangle\mathrm{I})\ \frac{\Gamma \vdash \langle x := f\rangle\phi}{\Gamma \vdash \langle x := *\rangle\phi} \qquad (\langle:*\rangle\mathrm{E})\ \frac{\Gamma \vdash \langle x := *\rangle\phi \quad \Gamma \vdash \forall x\,(\phi \to \psi)}{\Gamma \vdash \psi}\ (x \notin \mathrm{FV}(\psi))$$

$$(\langle\!\langle;\rangle\!\rangle\mathrm{I})\ \frac{\Gamma \vdash \langle\!\langle\alpha\rangle\!\rangle\langle\!\langle\beta\rangle\!\rangle\phi}{\Gamma \vdash \langle\!\langle\alpha;\beta\rangle\!\rangle\phi} \qquad (\mathrm{M})\ \frac{\Gamma \vdash \langle\!\langle\alpha\rangle\!\rangle\phi \quad \Gamma\frac{\vec{y}}{\mathrm{BV}(\alpha)}, \phi \vdash \psi}{\Gamma \vdash \langle\!\langle\alpha\rangle\!\rangle\psi}$$

$$(\langle\!\langle:=\rangle\!\rangle\mathrm{I})\ \frac{\Gamma\frac{y}{x}, x = f\frac{y}{x} \vdash \phi}{\Gamma \vdash \langle\!\langle x := f\rangle\!\rangle\phi} \qquad (\langle\!\langle^d\rangle\!\rangle\mathrm{I})\ \frac{\Gamma \vdash [\![\alpha]\!]\phi}{\Gamma \vdash \langle\!\langle\alpha^d\rangle\!\rangle\phi}$$

Fig. 3. CdGL proof calculus: First-order games (y fresh, f computable, ϕ^f_x admissible)

Figure 3 covers assignment, choice, sequencing, duals, and monotonicity. Angelic games have injectors ($\langle*\rangle$S,$\langle*\rangle$G) and case analysis ($\langle*\rangle$E). Duality $\langle\!\langle^d\rangle\!\rangle$I switches players by switching modalities. Sequential games ($\langle\!\langle;\rangle\!\rangle$I) are decomposed as nested modalities.

Monotonicity (M) is Lemma 7 in rule form. The second premiss writes $\Gamma\frac{\vec{y}}{\mathrm{BV}(\alpha)}$ to indicate that the bound variables of α must be freshly renamed in Γ for soundness. Rule M is used for generalization because all GLs are subnormal, lacking axiom K (modal modus ponens) and necessitation. Common uses include

concise right-to-left symbolic execution proofs and, in combination with ⟨;⟩I, Hoare-style sequential composition reasoning.

Nondeterministic assignments quantify over real-valued game variables. Assignments ⟨:=⟩I remember the initial value of x in fresh variable y ($\Gamma\frac{y}{x}$) for sake of completeness, then provide an assumption that x has been assigned to f. Skolemization [:*]I bound-renames x to y in Γ, written $\Gamma\frac{y}{x}$. Specialization [:*]E instantiates x to a term f by substituting ϕ^f_x. Existentials are introduced by giving a witness f in ⟨:*⟩I. Herbrandization ⟨:*⟩E unpacks existentials, soundness requires x is not free in ψ.

$$(\langle*\rangle\text{E})\ \frac{\Gamma\vdash\langle\alpha^*\rangle\phi \quad \Gamma,\phi\vdash\psi \quad \Gamma,\langle\alpha\rangle\langle\alpha^*\rangle\phi\vdash\psi}{\Gamma\vdash\psi} \qquad ([*]\text{E})\ \frac{\Gamma\vdash[\alpha^*]\phi}{\Gamma\vdash\phi\wedge[\alpha][\alpha^*]\phi}$$

$$(\langle*\rangle\text{S})\ \frac{\Gamma\vdash\phi}{\Gamma\vdash\langle\alpha^*\rangle\phi} \qquad (\langle*\rangle\text{G})\ \frac{\Gamma\vdash\langle\alpha\rangle\langle\alpha^*\rangle\phi}{\Gamma\vdash\langle\alpha^*\rangle\phi} \qquad (\text{FP})\ \frac{\Gamma\vdash\langle\alpha^*\rangle\phi \quad \phi\vdash\psi \quad \langle\alpha\rangle\psi\vdash\psi}{\Gamma\vdash\psi}$$

$$([*]\text{R})\ \frac{\Gamma\vdash\phi\wedge[\alpha][\alpha^*]\phi}{\Gamma\vdash[\alpha^*]\phi} \qquad (\text{loop})\ \frac{\Gamma\vdash J \quad J\vdash[\alpha]J \quad J\vdash\phi}{\Gamma\vdash[\alpha^*]\phi}$$

$$(\langle*\rangle\text{I})\ \frac{\Gamma\vdash\varphi \quad \varphi,\mathbf{0}\succcurlyeq\mathcal{M}\vdash\phi \quad \varphi,(\mathcal{M}\succ\mathbf{0}\wedge\mathcal{M}_0=\mathcal{M})\vdash\langle\alpha\rangle(\varphi\wedge\mathcal{M}_0\succ\mathcal{M})}{\Gamma\vdash\langle\alpha^*\rangle\phi}$$

Fig. 4. CdGL proof calculus: loops ($\mathcal{M}_0$ fresh)

Figure 4 provides rules for repetitions. In rule ⟨*⟩I, $\mathcal{M}$ indicates an arbitrary termination metric where $\succ$ and $\succcurlyeq$ denote strict and nonstrict comparison in an arbitrary (effectively) well-founded [28] partial order. Metavariable $\mathbf{0}$ represents a terminal value at which iteration stops; we will choose $\mathbf{0}=0$ in our example, but $\mathbf{0}$ need not be 0 in general. $\mathcal{M}_0$ is a fresh variable which remembers $\mathcal{M}$. Angel plays α^* by repeating an α strategy which always decreases the termination metric. Angel maintains a formula φ throughout, and stops once $0\succcurlyeq\mathcal{M}$. The postcondition need only follow from termination condition $0\succcurlyeq\mathcal{M}$ and convergence formula φ. Simple real comparisons $x\geq y$ are not well-founded, but inflated comparisons like $x\geq y+1$ are. Well-founded metrics ensure convergence in finitely (but often unboundedly) many iterations. In the simplest case, $\mathcal{M}$ is a real-valued term. Generalizing $\mathcal{M}$ to tuples enables, e.g., lexicographic termination metrics. For example, the metric in the proof of Example 4 is the distance to the goal, which must decrease by some minimum amount each iteration.

Repetition games can be folded and unfolded ([*]E,[*]R). Rule FP says $\langle\alpha^*\rangle\phi$ is a least pre-fixed-point. It works backwards: first show ψ holds after α^*, then preserve ψ when each iteration is unwound. Rule loop is the repetition invariant rule. Demonic repetition is eliminated by [*]E.

Like any first-order program logic, CdGL proofs contain first-order reasoning at the leaves. Decidability of constructive real arithmetic is an open problem [33], so first-order facts are proven manually in practice. Our semantics embed CdGL into type theory; we defer first-order arithmetic proving to the host theory. Even

effectively-well-founded $\succcurlyeq$ need not have decidable guards ($\mathbf{0} \succ \mathcal{M} \vee \mathcal{M} \succcurlyeq \mathbf{0}$) since exact comparisons are not computable [6]. We may not be able to distinguish $\mathcal{M} = \mathbf{0}$ from very small positive values of $\mathcal{M}$, leading to one unnecessary loop iteration, after which $\mathcal{M}$ is certainly $\mathbf{0}$ and the loop terminates. Comparison up to $\varepsilon > 0$ is decidable [12] ($f > g \vee (f < g + \varepsilon)$).

$$\text{(DC)}\ \frac{\Gamma \vdash [x'{=}f\,\&\,]R \quad \Gamma \vdash [x'{=}f\,\&\,\wedge R]\phi}{\Gamma \vdash [x'{=}f\,\&\,]\phi} \qquad \text{(DI)}\ \frac{\Gamma \vdash \phi \quad \Gamma \vdash \forall x\,(\rightarrow [x' := f](\phi)')}{\Gamma \vdash [x'{=}f\,\&\,]\phi}$$

$$\text{(DG)}\ \frac{\Gamma \vdash \exists y\,[x'{=}f, y' = a(x)y + b(x)\,\&\,]\phi}{\Gamma \vdash [x'{=}f\,\&\,]\phi} \qquad \text{(DW)}\ \frac{\Gamma \vdash \forall x\,\forall x'\,(\rightarrow \phi)}{\Gamma \vdash [x'{=}f\,\&\,]\phi}$$

$$\text{(DV)}\ \frac{\begin{array}{c}\Gamma, h \geq g \vdash \phi \quad \Gamma \vdash d > 0 \wedge \varepsilon > 0 \wedge h - g \geq -d\varepsilon \\ \Gamma \vdash \langle t := 0; \{t'{=}1, x'{=}f\,\&\,\}\rangle t \geq d \quad \Gamma \vdash [x'{=}f]((h)' - (g)') \geq \varepsilon\end{array}}{\Gamma \vdash \langle x'{=}f\,\&\,\rangle\phi}$$

$$\text{(bsolve)}\ \frac{\Gamma \vdash \forall t : \mathbb{R}_{\geq 0}\ ((\forall r : [0,t]\ [t := r; x := sln]\psi) \rightarrow [x := sln; x' := f]\phi)}{\Gamma \vdash [x'{=}f\,\&\,]\phi}$$

$$\text{(dsolve)}\ \frac{\Gamma \vdash \exists t : \mathbb{R}_{\geq 0}\ ((\forall r : [0,t]\ \langle t := r; x := sln\rangle\psi) \wedge \langle x := sln; x' := f\rangle\phi)}{\Gamma \vdash \langle x'{=}f\,\&\,\rangle\phi}$$

Fig. 5. CdGL proof calculus: ODEs. In bsolve and dsolve, sln solves $x' = f$ globally, t and r fresh, $x' \notin \mathrm{FV}(\phi)$

Figure 5 gives the ODE rules, which are a constructive version of those from dGL [42]. For nilpotent ODEs such as the plant of Example 4, reasoning via solutions is possible. Since CdGL supports nonlinear ODEs which often do not have closed-form solutions, we provide invariant-based rules, which are complete [46] for invariants of polynomial ODEs. *Differential induction* DI [41] says ϕ is an invariant of an ODE if it holds initially and if its *differential formula* [41] $(\phi)'$ holds throughout, for example $(f \geq g)' \equiv ((f)' \geq (g)')$. Soundness of DI requires differentiability, and $(\phi)'$ is not provable when ϕ mentions nondifferentiable terms. *Differential cut* DC proves R invariant, then adds it to the domain constraint. *Differential weakening* DW says that if ϕ follows from the domain constraint, it holds throughout the ODE. *Differential ghosts* DG permit us to augment an ODE system with a fresh dimension y, which enables [46] proofs of otherwise unprovable properties. We restrict the right-hand side of y to be linear in y and (uniformly) continuous in x because soundness requires that ghosting y' does not change the duration of an ODE. A linear right-hand side is guaranteed to be Lipschitz on the whole existence interval of equation $x' = f$, thus ensuring an unchanged duration by (constructive) Picard-Lindelöf [34]. *Differential variants* [41,53] DV is an Angelic counterpart to DI. The schema parameters d and ε must not mention x, x', t, t'. To show that f eventually exceeds g, first choose a duration d and a sufficiently high minimum rate ε at which $h - g$ will change. Prove that $h - g$ decreases at rate at least ε and that the ODE has a solution of duration d satisfying constraint . Thus at time d, both $h \geq g$ and its provable

consequents hold. Rules bsolve and dsolve assume as a side condition that sln is the unique solution of $x' = f$ on domain . They are convenient for ODEs with simple solutions, while invariant reasoning supports complicated ODEs.

6 Theory: Soundness

Following constructive counterparts of classical soundness proofs for dGL, we prove that the CdGL proof calculus is sound: provable formulas are true in the CIC semantics. For the sake of space, we give statements and some outlines here, reporting all proofs and lemmas elsewhere [10]. Similar lemmas have been used to prove soundness of dGL [45], but our new semantics lead to simpler statements for Lemmas 10 and 11. The coincidence property for terms is not proved but assumed, since we inherit a semantic treatment of terms from the host theory. Let $s\frac{y}{x}$ be s with the values of x and y swapped. Let s_x^f be $\mathsf{set}\ s\ x\ (f\ s)$. Defined CIC term $s \overset{V}{=} t \leftrightarrow *_{x\in V}(s\ x = t\ x)$ says s and t agree on all $x \in V$.

Lemma 9 (Uniform renaming). *Let $M\frac{y}{x}$ rename x and y in proof term M.*

- *If $\ulcorner\Gamma\urcorner(s) \vdash M : (\ulcorner\phi\urcorner\ s)$ then $\ulcorner\Gamma\frac{y}{x}\urcorner(s\frac{y}{x}) \vdash M\frac{y}{x} : (\ulcorner\phi\frac{y}{x}\urcorner\ s\frac{y}{x})$.*

Lemma 10 (Coincidence). *Assume $s \overset{V}{=} t$ where $V \supseteq FV(\Gamma) \cup FV(\phi)$.*

- *If $\ulcorner\Gamma\urcorner(s) \vdash M : (\ulcorner\phi\urcorner\ s)$ then exists N such that $\ulcorner\Gamma\urcorner(t) \vdash N : (\ulcorner\phi\urcorner\ t)$.*

Lemma 11 (Bound effect). *Let $P : \mathfrak{S} \Rightarrow \mathbb{T}$ and let $V \subseteq BV(\alpha)^{\complement}$, the complement of bound variables of α.*

- *There exists M such that $\ulcorner\Gamma\urcorner(s) \vdash M : (\langle\!\langle\alpha\rangle\!\rangle\ P\ s)$ iff there exists N such that $\ulcorner\Gamma\urcorner(s) \vdash N : (\langle\!\langle\alpha\rangle\!\rangle\ (\lambda t.\ P\ t * s \overset{V}{=} t)\ s)$.*
- *There exists M such that $\ulcorner\Gamma\urcorner(s) \vdash M : ([[\alpha]]\ P\ s)$ iff there exists N such that $\ulcorner\Gamma\urcorner(s) \vdash N : ([[\alpha]]\ (\lambda t.\ P\ t * s \overset{V}{=} t)\ s)$.*

Definition 12 (Term substitution admissibility [40, Def. 6]). *For a formula ϕ, (likewise for context Γ, term f, and game α) we say ϕ_x^f is* admissible *if x never appears free in ϕ under a binder of $\{x\} \cup FV(f)$.*

Lemma 13 (Term substitution). *Let M_x^f substitute f for x in proof term M. Let Γ_x^f and ϕ_x^f be admissible.*

- *If $\ulcorner\Gamma\urcorner(s_x^f) \vdash M : (\ulcorner\phi\urcorner\ s_x^f)$ then $\ulcorner\Gamma_x^f\urcorner(s) \vdash M_x^f : (\ulcorner\phi_x^f\urcorner\ s)$.*

The converse implication also holds, though its witness is not necessarily M.

Soundness of the proof calculus follows from the lemmas, and soundness of the ODE rules employing several known results from constructive analysis.

Theorem 14 (Soundness). *If $\Gamma \vdash M : \phi$ holds, then sequent $(\Gamma \vdash \phi)$ is valid. As a special case, if $\cdot \vdash M : \phi$ holds, then formula ϕ is valid.*

Proof Sketch. By induction on the derivation. The assignment case holds by Lemma 13 and Lemma 9. Lemma 10 and Lemma 11 are applied when maintaining truth of a formula across changing state. The equality and inequality cases of DI and DV employ the constructive mean-value theorem [10, Thm. 21], which has been formalized, e.g., in Coq [17]. Rules DW, bsolve, and dsolve follow from the semantics of ODEs. Rule DC uses the fact that prefixes of solutions are solutions. Rule DG uses constructive Picard-Lindelöf [34], which constitutes an algorithm for arbitrarily approximating the solution of any Lipschitz ODE, with a convergence rate depending on its Lipschitz constant. □

We have shown that every provable formula is true in the type-theoretic semantics. Because the soundness proof is constructive, it amounts to an extraction algorithm from CdGL into type theory: for each CdGL proof, there exists a program in type theory which inhabits the corresponding type of the semantics.

7 Theory: Extraction and Execution

Another perspective on constructivity is that provable properties must have witnesses. We show Existential and Disjunction properties providing witnesses for existentials and disjunctions. For modal formulas $\langle\alpha\rangle\phi$ and $[\alpha]\phi$ we show proofs can be *used as* winning strategies: a big-step operational semantics play allows playing strategies against each other to extract a proof that their goals hold in some final state t. Our presentation is more concise than defining the language, semantics, and properties of strategies, while providing key insights.

Lemma 15 (Existential Property). *Let* $s : \mathfrak{S}$. *If* $\ulcorner\Gamma\urcorner(s) \vdash M : (\ulcorner\exists x\, \phi\urcorner\ s)$ *then there exist terms* $f : \mathbb{R}$ *and* N *such that* $\ulcorner\Gamma\urcorner(s) \vdash N : (\ulcorner\phi^f_x\urcorner\ s)$.

Lemma 16 (Disjunction Property). *If* $\ulcorner\Gamma\urcorner(s) \vdash M : (\ulcorner\phi \vee \psi\urcorner\ s)$ *then there exists a proof term* N *such that* $\ulcorner\Gamma\urcorner(s) \vdash N : (\ulcorner\phi\urcorner\ s)$ *or* $\ulcorner\Gamma\urcorner(s) \vdash N : (\ulcorner\psi\urcorner\ s)$.

The proofs follow their counterparts in type theory. The Disjunction Property considers truth at a *specific state*. Validity of $\phi \vee \psi$ does *not* imply validity of either ϕ or ψ. For example, $x < 1 \vee x > 0$ is valid, but its disjuncts are not.

Function play below gives a big-step semantics: Angel and Demon strategies as and ds for respective goals ϕ and ψ in game α suffice to construct a final state t satisfying both. By parametricity, t was found by playing α, because play cannot inspect P and Q, thus can only prove them via as and ds.

$$\begin{aligned}\mathsf{play} :\ & \Pi\alpha : \mathsf{Game}.\, \Pi P, Q : (\mathfrak{S} \Rightarrow \mathbb{T}).\, \Pi s : \mathfrak{S}.\\ & \big(\langle\!\langle\alpha\rangle\!\rangle\ P\ s \Rightarrow [[\alpha]]\ Q\ s \Rightarrow \sum t : \mathfrak{S}.\, P\ t * Q\ t\big)\end{aligned}$$

Applications of play are written $\mathsf{play}_\alpha\ s\ \mathsf{as}\ \mathsf{ds}$ (P and Q implicit). Game consistency (Corollary 17) is by play and consistency of type theory. Note that α^d is

played by swapping the Angel and Demon strategies in α.

$$
\begin{aligned}
\mathsf{play}_{x:=f}\ s\ \mathsf{as}\ \mathsf{ds} &= (\mathsf{let}\, t\ =\ \mathsf{set}\ s\ x\ (f\ s)\,\mathsf{in}\,(t,(\mathsf{as}\ t,\mathsf{ds}\ t)))\\
\mathsf{play}_{x:=*}\ s\ \mathsf{as}\ \mathsf{ds} &= \mathsf{let}\, t\ =\ \mathsf{set}\ s\ x\ \pi_L\mathsf{as}\,\mathsf{in}\,(t,(\pi_R\mathsf{as},\mathsf{ds}\ \pi_L\mathsf{as}))\\
\mathsf{play}_{x'=f\,\&}\ s\ \mathsf{as}\ \mathsf{ds} &= \mathsf{let}\,(d,\mathrm{sol},\mathrm{solves},c,p)\ =\ \mathsf{as}\ s\,\mathsf{in}\\
&\qquad (\mathsf{set}\ s\ x\ (\mathrm{sol}\ d),(p,\mathsf{ds}\ d\ \mathrm{sol}\ \mathrm{solves}\ c))\\
\mathsf{play}_{?\phi}\ s\ \mathsf{as}\ \mathsf{ds} &= (s,(\pi_R\mathsf{as},\mathsf{ds}\ (\pi_L\mathsf{as})))\\
\mathsf{play}_{\alpha\cup\beta}\ s\ \mathsf{as}\ \mathsf{ds} &= \mathsf{case}\ (\mathsf{as}\ s)\ \mathsf{of}\\
&\qquad \mathsf{as}' \Rightarrow \mathsf{play}_{\alpha}\ s\ \mathsf{as}'\ (\pi_L\mathsf{ds})\\
&\quad \mid \mathsf{as}' \Rightarrow \mathsf{play}_{\beta}\ s\ \mathsf{as}'\ (\pi_R\mathsf{ds})\\
\mathsf{play}_{\alpha;\beta}\ s\ \mathsf{as}\ \mathsf{ds} &= (\mathsf{let}\,(t,(\mathsf{as}',\mathsf{ds}'))\ =\ \mathsf{play}_{\alpha}\ s\ \mathsf{as}\ \mathsf{ds}\,\mathsf{in}\,\mathsf{play}_{\beta}\ t\ \mathsf{as}'\ \mathsf{ds}')\\
\mathsf{play}_{\alpha^*}\ s\ \mathsf{as}\ \mathsf{ds} &= \mathsf{case}\ (\mathsf{as}\ s)\ \mathsf{of}\\
&\qquad \mathsf{as}' \Rightarrow (s,(\mathsf{as}',\pi_L\mathsf{ds}))\\
&\quad \mid \mathsf{as}' \Rightarrow \mathsf{let}\,(t,(\mathsf{as}'',\mathsf{ds}''))\ =\ \mathsf{play}_{\alpha}\ s\ \mathsf{as}'\ (\pi_R\mathsf{ds})\,\mathsf{in}\\
&\qquad\qquad \mathsf{play}_{\alpha^*}\ t\ \mathsf{as}''\ \mathsf{ds}''\\
\mathsf{play}_{\alpha^d}\ s\ \mathsf{as}\ \mathsf{ds} &= \mathsf{play}_{\alpha}\ s\ \mathsf{ds}\ \mathsf{as}
\end{aligned}
$$

Corollary 17 (Consistency). *It is never the case that both* $\ulcorner\langle\alpha\rangle\phi\urcorner\ s$ *and* $\ulcorner[\alpha]\neg\phi\urcorner\ s$ *are inhabited.*

Proof. Suppose $\mathsf{as} : \ulcorner\langle\alpha\rangle\phi\urcorner\ s$ and $\mathsf{ds} : \ulcorner[\alpha]\neg\phi\urcorner\ s$, then $\pi_R(\mathsf{play}_\alpha\ s\ \mathsf{as}\ \mathsf{ds}) : \bot$, contradicting consistency of type theory. □

The play semantics show how strategies can be executed. Consistency is a theorem which ought to hold in any GL and thus helps validate our semantics.

8 Conclusion and Future Work

We extended Constructive Game Logic CGL to CdGL for constructive *hybrid* games. We contributed new semantics. We presented a natural deduction proof calculus for CdGL and used it to prove reach-avoid correctness of 1D driving with adversarial timing. We showed soundness and constructivity results.

The next step is to implement a proof checker, game interpreter, and synthesis tool for CdGL. Function play is the high-level interpreter algorithm, while synthesis would commit to one Angel strategy and allow black-box Demon implementations for an external environment. Angel strategies are positive and are synthesized by extracting witnesses from each introduction rule. Demonic invariants and test conditions describe allowed observable behaviors. Demon strategies are negative and characterized by observable behaviors, so it suffices to monitor their compliance with invariants and test conditions extracted from the proof.

Acknowledgements. We thank Jon Sterling for suggestions regarding our choice of type theory and for references to the literature. We thank the anonymous reviewers for their helpful feedback.

References

1. Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Inf. Comput. **163**(2), 409–470 (2000). https://doi.org/10.1006/inco.2000.2930
2. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002). https://doi.org/10.1145/585265.585270
3. van Benthem, J.: Logic of strategies: What and how? In: van Benthem, J., Ghosh, S., Verbrugge, R. (eds.) Models of Strategic Reasoning - Logics, Games, and Communities, LNCS, vol. 8972, pp. 321–332. Springer (2015), https://doi.org/10.1007/978-3-662-48540-8_10
4. van Benthem, J., Pacuit, E.: Dynamic logics of evidence-based beliefs. Studia Logica **99**(1-3), 61–92 (2011). https://doi.org/10.1007/s11225-011-9347-x
5. van Benthem, J., Pacuit, E., Roy, O.: Toward a theory of play: A logical perspective on games and interaction. Games (2011). https://doi.org/10.3390/g2010052
6. Bishop, E.: Foundations of constructive analysis. McGraw-Hill (1967)
7. Bohrer, R., Fernández, M., Platzer, A.: dL$_\iota$: Definite descriptions in differential dynamic logic. In: Fontaine, P. (ed.) CADE. LNCS, vol. 11716, pp. 94–110. Springer (2019). https://doi.org/10.1007/978-3-030-29436-6_6
8. Bohrer, R., Platzer, A.: Toward structured proofs for dynamic logics. CoRR **abs/1908.05535** (2019)
9. Bohrer, R., Platzer, A.: Constructive game logic. In: Müller, P. (ed.) ESOP. LNCS, vol. 12075. Springer (2020)
10. Bohrer, R., Platzer, A.: Constructive hybrid games. CoRR **abs/2002.02536** (2020), https://arxiv.org/abs/2002.02536
11. Bohrer, R., Tan, Y.K., Mitsch, S., Myreen, M.O., Platzer, A.: VeriPhy: Verified controller executables from verified cyber-physical system models. In: Grossman, D. (ed.) PLDI. pp. 617–630. ACM (2018). https://doi.org/10.1145/3192366.3192406
12. Bridges, D.S., Vita, L.S.: Techniques of constructive analysis. Springer (2007)
13. Celani, S.A.: A fragment of intuitionistic dynamic logic. Fundam. Inform. **46**(3), 187–197 (2001), http://content.iospress.com/articles/fundamenta-informaticae/fi46-3-01
14. Chatterjee, K., Henzinger, T.A., Piterman, N.: Strategy logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR. LNCS, vol. 4703, pp. 59–73. Springer (2007), https://doi.org/10.1007/978-3-540-74407-8_5
15. Coquand, T., Huet, G.P.: The calculus of constructions. Inf. Comput. **76**(2/3), 95–120 (1988). https://doi.org/10.1016/0890-5401(88)90005-3
16. Coquand, T., Paulin, C.: Inductively defined types. In: Martin-Löf, P., Mints, G. (eds.) COLOG. LNCS, vol. 417, pp. 50–66. Springer (1988). https://doi.org/10.1007/3-540-52335-9_47
17. Cruz-Filipe, L., Geuvers, H., Wiedijk, F.: C-CoRN, the constructive Coq repository at Nijmegen. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM. LNCS, vol. 3119. Springer (2004). https://doi.org/10.1007/978-3-540-27818-4_7, accessed: commits 9c44dae and 6411967
18. Degen, J., Werner, J.: Towards intuitionistic dynamic logic. Log. and Log. Philosophy **15**(4), 305–324 (2006). https://doi.org/10.12775/LLP.2006.018

19. Dybjer, P.: Inductive families. Formal Asp. Comput. **6**(4), 440–465 (1994). https://doi.org/10.1007/BF01211308
20. Filippidis, I., Dathathri, S., Livingston, S.C., Ozay, N., Murray, R.M.: Control design for hybrid systems with TuLiP: The temporal logic planning toolbox. In: Conference on Control Applications. pp. 1030–1041. IEEE (2016). https://doi.org/10.1109/CCA.2016.7587949
21. Finucane, C., Jing, G., Kress-Gazit, H.: LTLMoP: Experimenting with language, temporal logic and robot control. In: IROS. pp. 1988–1993. IEEE (2010). https://doi.org/10.1109/IROS.2010.5650371
22. Foster, J.N.: Bidirectional programming languages. Tech. Rep. MS-CIS-10-08, Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA (March 2010)
23. Ghosh, S.: Strategies made explicit in dynamic game logic. Workshop on Logic and Intelligent Interaction at ESSLLI, Hamburg pp. 74 –81 (2008)
24. Harel, D., Kozen, D., Tiuryn, J.: Dynamic logic. MIT Press (2000)
25. Henzinger, T.A., Horowitz, B., Majumdar, R.: Rectangular hybrid games. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR. LNCS, vol. 1664, pp. 320–335. Springer (1999). https://doi.org/10.1007/3-540-48320-9_23
26. Hoare, C.A.R.: An axiomatic basis for computer programming. Commun. ACM **12**(10), 576–580 (1969). https://doi.org/10.1145/363235.363259
27. van der Hoek, W., Jamroga, W., Wooldridge, M.J.: A logic for strategic reasoning. In: Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M.P., Wooldridge, M.J. (eds.) AAMAS. ACM (2005). https://doi.org/10.1145/1082473.1082497
28. Hofmann, M., van Oosten, J., Streicher, T.: Well-foundedness in realizability. Arch. Math. Log. **45**(7), 795–805 (2006), https://doi.org/10.1007/s00153-006-0003-5
29. Isaacs, R.: Differential games: A mathematical theory with applications to warfare and pursuit, control and optimization. Series in Applied Mathematics (SIAM), Wiley, New York (1965)
30. Kamide, N.: Strong normalization of program-indexed lambda calculus. Bull. Sect. Log. Univ. Łódź **39**(1-2), 65–78 (2010)
31. Kloetzer, M., Belta, C.: A fully automated framework for control of linear systems from temporal logic specifications. IEEE Trans. Automat. Contr. **53**(1), 287–297 (2008). https://doi.org/10.1109/TAC.2007.914952
32. Lipton, J.: Constructive Kripke semantics and realizability. In: Moschovakis, Y. (ed.) Logic from Computer Science. pp. 319–357. Springer (1992), https://doi.org/10.1017/978-1-4612-2822-6_13
33. Lombardi, H., Mahboubi, A.: Théories géométriques pour l'algèbre des nombres réels. Contemporary Mathematics **697**, 239–264 (2017)
34. Makarov, E., Spitters, B.: The Picard algorithm for ordinary differential equations in Coq. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) ITP. LNCS, vol. 7998. Springer (2013), https://doi.org/10.1007/978-3-642-39634-2_34
35. Mamouras, K.: Synthesis of strategies using the Hoare logic of angelic and demonic nondeterminism. Log. Methods Comput. Sci. **12**(3), 1–41 (2016). https://doi.org/10.2168/LMCS-12(3:6)2016
36. Mitsch, S., Platzer, A.: ModelPlex: Verified runtime validation of verified cyber-physical system models. Form. Methods Syst. Des. **49**(1), 33–74 (2016). https://doi.org/10.1007/s10703-016-0241-z
37. Murphy VII, T., Crary, K., Harper, R., Pfenning, F.: A symmetric modal lambda calculus for distributed computing. In: LICS. IEEE (2004), https://doi.org/10.1109/LICS.2004.1319623

38. van Oosten, J.: Realizability: A historical essay. Math. Structures Comput. Sci. **12**(3), 239–263 (2002). https://doi.org/10.1017/S0960129502003626
39. Parikh, R.: Propositional game logic. In: FOCS. pp. 195–200. IEEE (1983). https://doi.org/10.1109/SFCS.1983.47
40. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. **41**(2), 143–189 (2008). https://doi.org/10.1007/s10817-008-9103-8
41. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. **20**(1), 309–352 (2010). https://doi.org/10.1093/logcom/exn070
42. Platzer, A.: Differential game logic. ACM Trans. Comput. Log. **17**(1), 1:1–1:51 (2015). https://doi.org/10.1145/2817824
43. Platzer, A.: Differential hybrid games. ACM Trans. Comput. Log. **18**(3), 19:1–19:44 (2017). https://doi.org/10.1145/3091123
44. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer, Switzerland (2018). https://doi.org/10.1007/978-3-319-63588-0
45. Platzer, A.: Uniform substitution for differential game logic. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR. LNCS, vol. 10900, pp. 211–227. Springer (2018). https://doi.org/10.1007/978-3-319-94205-6_15
46. Platzer, A., Tan, Y.K.: Differential equation invariance axiomatization. J. ACM **67**(1) (2020). https://doi.org/10.1145/3380825
47. Pratt, V.R.: Semantical considerations on Floyd-Hoare logic. In: FOCS. pp. 109–121. IEEE (1976). https://doi.org/10.1109/SFCS.1976.27
48. Quesel, J.D., Platzer, A.: Playing hybrid games with keymaera. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR. LNCS, vol. 7364, pp. 439–453. Springer (2012). https://doi.org/10.1007/978-3-642-31365-3_34
49. Ramanujam, R., Simon, S.E.: Dynamic logic on games with structured strategies. In: Brewka, G., Lang, J. (eds.) Knowledge Representation. pp. 49–58. AAAI Press (2008), http://www.aaai.org/Library/KR/2008/kr08-006.php
50. Shakernia, O., Pappas, G.J., Sastry, S.: Semi-decidable synthesis for triangular hybrid systems. In: Benedetto, M.D.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC. LNCS, vol. 2034, pp. 487–500. Springer (2001). https://doi.org/10.1007/3-540-45351-2_39
51. Shakernia, O., Sastry, S., Pappas, G.J.: Decidable controller synthesis for classes of linear systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC. LNCS, vol. 1790, pp. 407–420. Springer (2000). https://doi.org/10.1007/3-540-46430-1_34
52. Taly, A., Tiwari, A.: Switching logic synthesis for reachability. In: Carloni, L.P., Tripakis, S. (eds.) EMSOFT. pp. 19–28. ACM (2010). https://doi.org/10.1145/1879021.1879025
53. Tan, Y.K., Platzer, A.: An axiomatic approach to liveness for differential equations. In: ter Beek, M., McIver, A., Oliviera, J.N. (eds.) FM. LNCS, vol. 11800, pp. 371–388. Springer (2019). https://doi.org/10.1007/978-3-030-30942-8_23
54. The Coq development team: The Coq proof assistant reference manual (2019), https://coq.inria.fr/
55. Tomlin, C.J., Lygeros, J., Sastry, S.S.: A game theoretic approach to controller design for hybrid systems. Proc. IEEE **88**(7), 949–970 (2000)
56. Van Benthem, J.: Games in dynamic-epistemic logic. Bull. Econ. Research **53**(4), 219–248 (2001)
57. Weihrauch, K.: Computable Analysis - An Introduction. Texts in Theoretical Computer Science, Springer (2000). https://doi.org/10.1007/978-3-642-56999-9
58. Wijesekera, D.: Constructive modal logics I. Ann. Pure Appl. Log. **50**(3), 271–301 (1990). https://doi.org/10.1016/0168-0072(90)90059-B

59. Wijesekera, D., Nerode, A.: Tableaux for constructive concurrent dynamic logic. Ann. Pure Appl. Log. **135**(1-3), 1–72 (2005). https://doi.org/10.1016/j.apal.2004.12.001