# Refining Constructive Hybrid Games

Brandon Bohrer
Carnegie Mellon University, USA
bbohrer@cs.cmu.edu

André Platzer
Carnegie Mellon University, USA
Technische Universität München, Germany
aplatzer@cs.cmu.edu

## Abstract

We extend the constructive differential game logic (CdGL) of hybrid games with a refinement connective that relates two hybrid games. In addition to CdGL's ability to prove the existence of winning strategies for specific postconditions of hybrid games, game refinements relate two games to one another. That makes it possible to prove that *any* winning strategy for *any* postcondition of one game carries over to a winning strategy for the other. Since CdGL is constructive, a computable winning strategy can be extracted from a proof that a player wins a game. A folk theorem says that any such winning strategy for a hybrid game gives rise to a corresponding hybrid system satisfying the same property. We make this precise using CdGL's game refinements and prove correct the construction of hybrid systems from winning strategies of hybrid games.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Type theory

**Keywords and phrases** Hybrid Games, Constructive Logic, Refinement, Game Logic

**Digital Object Identifier** 10.4230/LIPIcs.FSCD.2020.13

**Related Version** A companion report with proofs is available [12].

**Funding** This research was sponsored by the AFOSR under grant number FA9550-16-1-0288. The first author was also funded by the NDSEG Fellowship.

**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.

## 1 Introduction

Cyber-physical systems (CPSs) such as transportation systems, medical devices, and power systems are often modeled with hybrid systems and hybrid games. Hybrid systems combine discrete computation with continuous differential equations (ODEs), to which hybrid games add adversarial dynamics. *Differential Game Logic* (dGL) [44] and its systems fragment dL [46] provide formal proofs of correctness properties such as safety and liveness for hybrid games and systems. Theorems of dGL answer: does a winning strategy exist for a given player to achieve a given postcondition in a given game? Because safety-critical CPSs must remain reliable even in adversarial environments, these rigorous correctness guarantees for adversarial models are essential. Despite the importance of games, verification and synthesis technology for hybrid games are less mature than for hybrid systems. For example, the end-to-end verified monitor synthesizer VeriPhy [13] only supports systems.

This paper studies the gap between hybrid games and systems and proposes a reduction, which in principle enables hybrid game synthesis via existing hybrid system tools. To study this gap, we use a program refinement calculus, which is the fundamental tool for comparing programs. Refinement and equivalence reasoning have repeatedly proven fruitful for programs generally and for CPS models specifically:

- Equivalences of programs are the fundamental building block for KAT [34].
- Differential refinement logic dRL [36] has reduced the human labor required for verification of classical hybrid systems by relating one hybrid system to another.
- Differential game refinements in dGL [45] provide a relational reasoning technique for *differential* games (as opposed to hybrid games).

Because we are motivated by code synthesis, we develop our refinement calculus for Constructive Differential Game Logic (CdGL) [11], which ensures winning strategies are *computable*. Having defined refinements for CdGL, we bridge the gap between hybrid games and systems by defining an operation we call *reification*. The reification operation takes as its input a hybrid game $\alpha$, its correctness condition $\phi$, and a CdGL proof that $\alpha$ satisfies $\phi$. The reified output is a hybrid *system* which implements the strategy expressed in the correctness proof. Refinement allows us to prove the relationship between a game and its reified system: the output system refines the input game so that every safety theorem of the output is a theorem of the input. Conversely, the output also satisfies $\phi$. To our knowledge, prior works [5] only feature ad-hoc discussions of reification; we give the first rigorous algorithm and correctness theorems. It may be surprising that game strategies can be reduced to systems, because games are known [44] to be more expressive than systems. Our result does not contradict this fact: only once a winning strategy is known can we bridge this expressiveness gap. Reification makes several practical applications possible:

i) End-to-end correctness for hybrid game synthesis could be implemented by reifying winning strategies as a preprocessing step to VeriPhy [13].
ii) Interactive proof languages are better understood for systems than for games. A proof language for CdGL could be developed which combines systems reasoning with refinement. Experience with dRL [36] suggests refinement-based proof may be more productive.
iii) Reification and refinement give an intensional view of strategy equality: two strategies are "the same" if their reification produces equivalent systems.
iv) Refinement may enable comparing the efficacy of two controllers: does one controller always achieve its goal faster?

In Section 2, we discuss additional related work. In Section 3, we recall the syntax of CdGL, demonstrate the syntax with a toy example, and add a refinement connective. In Section 4, we recall the semantics of CdGL, generalizing them to support refinement. In Section 5, we give a calculus for CdGL refinements. In Section 6, we discuss theoretical results about soundness and reification. The paper concludes with Section 7.

## 2 Related Work

Our most closely related works are refinement logics. Other related works include constructive modal logics, synthesis, and games in logic.

### Refinement

Refinement calculi have been studied extensively, and previous studies [5] have given examples of how programs can be refined from games, but have not given an explicit reification algorithm let alone its correctness theorems. We give an explicit *reification* algorithm and prove that it captures the winning strategy of a game in a system. Our proofs are not formalized because they rely on features which are unsupported or experimental in prominent proof assistants such as Coq. Specifically, we construct inductive families whose well-foundedness requires inductive proof and we make significant use of universe polymorphism.

We build directly on classical refinement reasoning for hybrid systems from d$\mathcal{RL}$ [36] and also on (propositional discrete) game algebra [28]. Game algebra equivalences are not contextual because contextual reasoning is uniquely challenging for games, which are subnormal. d$\mathcal{RL}$ supports contextual reasoning but not games. Our refinement calculus subsumes both game algebra and d$\mathcal{RL}$ by mixing games rules with contextual rules for systems. Even for rules which look the same as prior work, our constructive semantics demand novel soundness proofs. Event-B [2] uses refinement for practical verification by verifying simpler models, then refining them to more complex models. Extensions of Event-B have been proposed for hybrid systems [6] but not hybrid games.

## Games in Logic

Propositional GL was introduced by Parikh [41]. The first-order GL of hybrid games is dGL [44]. We build on CdGL [10, 11], the constructive dialect of dGL. GL formulas have been reduced to $\mu$-calculus [41, 33] and game algebras have been reduced [28] to propositional modal logic. In contrast, we translate game logic *proofs*, which lets us translate CdGL into a *less expressive* logical fragment.

GLs are unique in their clear delegation of strategy to the *proof* language rather than the *model* language, allowing succinct, trustworthy game specifications with sophisticated winning strategies. Relatives without this separation of concerns include Constructive Concurrent Dynamic Logic [59], SL [16], ATL [4], CATL [55], SDGL [27], structured strategies [48], DEL [52, 54, 51], evidence logic [53], and Angelic Hoare Logic [38].

Completeness of game logics is a notoriously difficult problem, which has recently been shown in the propositional case [22]. dGL is undecidable, but is *relatively* complete [44]. Game logics can be expressed in game refinement logics, so game refinement completeness is at least as difficult. Game algebra is complete for games containing only choices, sequencing, and duality [28]. This paper does not pursue a completeness theorem, but we subsume game algebra, incorporate dGL-like rules, and exercise a broad range of refinement reasoning in our theorems on reification. These facts bode well for the expressiveness of our calculus. As with other refinement calculi, we expect that limitations arise when reasoning with ghost variables or reasoning about two games whose structures are entirely different.

## Constructive Modal Logics

The task of assigning a semantics to games 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 [56, 35]. We follow the type theoretic semantics which were introduced for CdGL [11]. A related approach to our type-theoretic semantics is Hoare Type Theory [40], which provides a type-theoretic connective for Hoare triples, but does not consider games or refinement.

Constructive (modal) program logics are less studied than classical ones. A few authors [10, 31] develop a Curry-Howard correspondence with proof terms, the latter for a simple fragment of dynamic logic. Other works [59, 20, 15] address only fragments and do not explore Curry-Howard in the same depth. In contrast to these, we support constructive refinement, which is also of interest for constructive program logics generally. We do not discuss proof terms here for the sake of space. Our treatment of constructive real arithmetic follows CdGL, which follows Bishop [8, 14] using constructive formalizations [19, 37].

### Hybrid Systems Synthesis

Synthesis for hybrid systems is an active research area. Fully automated synthesis relies on restrictions such as simple fragments [32, 49] or discrete abstractions [25, 24]. ModelPlex [39] exploits interactive safety proofs in dL [46], the systems fragment of dGL, for monitor synthesis. Not only can proof-based synthesis synthesize *every* provable model, but it gives the user more control: to generate a less restrictive monitor, simply revise the proof to use less restrictive assumptions. ModelPlex supports an especially rigorous end-to-end verification approach [13]. We aim to provide a reduction through which ModelPlex could support games. Synthesis of high-level plans is also studied [7, 23].

## 3 Constructive Differential Game Logic

We recall the language of CdGL [11], introduce refinement formulas, and give an example.

### 3.1 Syntax

The language of CdGL consists of terms $f, g$, games $\alpha, \beta$, and formulas $\phi, \psi, \varphi$. Games are perfect-information, zero-sum, and with two players. Take note of our terminology for players, which is particularly subtle for constructive games. We use the name *Angel* for the player whose choices are quantified existentially ("us") and *Demon* for the player whose choices are quantified universally ("them"). The players alternate turns, and at any moment one player is *active* (making decisions) while the opponent is *dormant* (waiting for their turn). In an unfortunate subtlety, a formula, proof, refinement, etc. is called Angelic whenever it is existential and Demonic whenever it is universal, regardless of which player is active. The simplest terms are (game) *variables* $x, y \in \mathcal{V}$ where $\mathcal{V}$ is the set of variable identifiers. All variables are mutable and globally scoped. Their values correspond to the state of the game. For every base game variable $x$ there is a primed counterpart $x'$ whose purpose within an ODE is to track the time-derivative of $x$. The state consists of reals, which are uncountable, but the value of a term is computable as a function of the state.

▶ **Definition 1** (Terms). *We define scalar terms $f, g$ inductively, where $c \in \mathbb{R}$ is a real literal, $x$ a game variable, $f + g$ a sum, and $f \cdot g$ a product:*

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

In a practical implementation within a theorem prover, one might prefer to reuse terms from the metalogic. It suffices that the interpretation of every term is a (Type-2 [57]) computable function. Type-2 computability means the interpretation of $f$ must be computable to arbitrary precision when the values of variables are represented as streams of bits. We occasionally use terms which return tuples of reals, which are computable when every component is computable. The total spatial differential of term $f$ is written $(f)'$ and agrees with the time derivative of $f$ during an ODE.

Because CdGL is constructive, strategies must represent Angel's choices computably. While Demon is playing, Angel simply monitors whether Demon's choices obey the rules of the game, and does not care whether choices were computable. We informally discuss how a game is played here, then give full winning conditions in Section 4. The definitions of games and formulas are simultaneously inductive.

▶ **Definition 2** (Games). *The language of* games $\alpha, \beta$ *is defined recursively as such:*

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

The *test game* $?\phi$, is a no-op if the active player can present a proof of $\phi$, else the dormant player wins by default since the active player "broke the rules". A deterministic assignment $x := f$ updates variable $x$ to the value of term $f$. Nondeterministic assignments $x := *$ ask the active player to compute the new value of $x : \mathbb{R}$, i.e. they are witnessed by a term that computes a new value of $x$. The ODE game $x' = f \,\&\, \psi$ evolves the ODE $x' = f$ for some duration $d \geq 0$ chosen by the active player such that the active player proves $\psi$ throughout.

All terms $f$ are effectively locally Lipschitz continuous, meaning that a neighborhood $N$ and real $L$ for each state can be constructed such that $L$ is a Lipschitz constant of $f$ on $N$. Effective local Lipschitz continuity ensures that constructive Picard-Lindelöf [37] can construct the unique solution of each ODE, which need not have a closed form. ODEs are explicit-form, meaning that $f$ and $\psi$ do not mention any primed variables $y'$. Except when otherwise stated, we present ODEs with a single equation $x' = f$ for the sake of readability. In the choice game $\alpha \cup \beta$, the active player chooses whether to play game $\alpha$ or game $\beta$. In the sequential game $\alpha ; \beta$, game $\alpha$ is played first, then $\beta$ from the resulting state (unless a player broke the rules during $\alpha$). In the repetition game $\alpha^*$, the active player chooses after each repetition of $\alpha$ whether to continue playing, but repetitions must be well-founded and thus terminating. The exact number of iterations does not need to be computed in advance but can depend on the opponent's moves. The dual game $\alpha^d$ plays $\alpha$ with the active and dormant roles reversed. We parenthesize games with braces $\{\alpha\}$ when necessary.

▶ **Definition 3** (CdGL Formulas). *The language of CdGL* formulas $\phi, \psi, \varphi$ *is given recursively by the following grammar, where* $\sim \in \{\leq, <, =, \neq, >, \geq\}$ *are comparison predicates:*

$$\phi \ ::= \ \langle \alpha \rangle \phi \mid [\alpha]\phi \mid f \sim g \mid \alpha \leq^i_{[]} \beta$$

Modalities $\langle \alpha \rangle \phi$ and $[\alpha]\phi$ say Angel wins $\alpha$ with postcondition $\phi$, starting as the active or dormant player respectively. Modality $\langle \alpha \rangle \phi$ is Angelic in the sense that decisions are resolved Angelically: Angel is the one currently making choices. Modality $[\alpha]\phi$ is Demonic in the sense that decisions are resolved Demonically: Angel has no control until a dual operator is encountered. We will deal mainly in box modalities $[\alpha]\phi$, with *Angel*'s moves appearing inside dualities $\alpha^d$ and *Demon*'s moves outside dualities.

To define refinements, we introduce the rank $\mathfrak{R}(\alpha \text{ or } \phi)$ of a game or formula, a technical device which represents the smallest predicative universe in which $\alpha$ has a semantics, see Section 4. Game refinements come in two standard [28] kinds: Angelic and Demonic. *Demonic refinement* $\alpha \leq^i_{[]} \beta$ of *rank* $i$ holds if for every $\phi$ with $\mathfrak{R}(\phi) \leq i$, dormant winning strategies of $[\alpha]\phi$ can be mapped constructively into wiig strategies of $[\beta]\phi$. *Angelic refinement* $\alpha \leq^i_{\langle\rangle} \beta$ maps active winning strategies of $\langle \alpha \rangle \phi$ constructively into winning strategies of $\langle \beta \rangle \phi$. Note this difference carefully: Angelic refinement may be more familiar to the reader, but we take the Demonic presentation as primary, in large part because the theorems we wish to prove are Demonic. Angelic and Demonic refinement are interdefinable: $\alpha \leq^i_{\langle\rangle} \beta \leftrightarrow \alpha^d \leq^i_{[]} \beta^d$ and vice versa. You may wish to ignore rank on the first reading: it can be inferred automatically, and we write $\alpha \leq_{[]} \beta$ when rank is unimportant.

The standard connectives of first-order constructive logic are definable from games and comparisons. Verum (`tt`) is defined $1 > 0$ and falsum (`ff`) is $0 > 1$. Conjunction $\phi \wedge \psi$ is defined $\langle ?\phi\rangle\psi$, disjunction $\phi \vee \psi$ is $\langle ?\phi \cup ?\psi\rangle\mathtt{tt}$, implication $\phi \to \psi$ is $[?\phi]\psi$, universal quantification $\forall x\, \phi$ is defined $[x := *]\phi$, and existential quantification $\exists x\, \phi$ is $\langle x := *\rangle\phi$. Equivalence $\phi \leftrightarrow \psi$ is $(\phi \to \psi) \wedge (\psi \to \phi)$. As usual in constructive logics, negation $\neg\phi$ is defined $\phi \to \mathtt{ff}$, and inequality is defined by $f \neq g \equiv \neg(f = g)$. The defined game $\mathsf{skip}$ is the trivial test $?\mathtt{tt}$. While these constructs are derivable, and thus it suffices to provide semantics and proof rules for the core constructs, we find it useful to consider the core and derived forms syntactically distinct. It will also aid in understanding of the semantics to keep the definitions above in mind, because the semantics for many first-order programs mirror those from their counterpart in first-order constructive logic.

## 3.2 Example Game

As a simple example, consider a *push-pull cart* [46] on a 1 dimensional playing field with boundaries $x_l \leq x \leq x_r$ where $x$ is the position of the cart and $x_l < x_r$ strictly. The initial position is written $x_0$. Thes preconditions are in formula $\mathsf{pre}$. Demon is at the left of the cart and Angel at its right. Each player chooses to pull or push the cart, then the (oversimplified) physics say velocity is proportional to the sum of forces. Physics can evolve so long as the boundary $x_l \leq x \leq x_r$ is respected, with duration chosen by Demon.

$$\mathsf{pre} \equiv x_l < x_r \wedge x_l \leq x_0 = x \leq x_r$$
$$\mathsf{PP} \equiv \{\{L := -1 \cup L := 1\}; \{R := -1 \cup R := 1\}^d; \{x' = L + R \,\&\, x_l \leq x \leq x_r\}\}^*$$

A simple safety theorem for the push-pull game says that Angel has a strategy to ensure position $x$ remains constant ($x = x_0$) no matter how Demon plays:

$$\mathsf{pre} \to [\mathsf{PP}]x = x_0 \tag{1}$$

The winning strategy that proves (1) is a simple mirroring strategy: Angel observes Demon's choice of $L$ and plays the opposite value of $R$ so that $L + R = 0$. Because $L + R = 0$, the ODE simplifies to $x' = 0 \,\&\, x_l \leq x \leq x_r$, which has the trivial solution $x(t) = x(0)$ for all times $t \in \mathbb{R}_{\geq 0}$. Angel shows the safety theorem by replacing the ODE with its solution and observing that $x = x_0$ holds for all possible durations.

In addition to solution reasoning, CdGL supports *differential invariant* [46] reasoning which appeals to the derivative of a term and *differential ghost* [46] reasoning which augments an ODE with a new continuous variable. Solution reasoning suffices for this toy example, but invariant reasoning is essential for CdGL games whose ODEs have non-polynomial, even non-elementary solutions. Ghost reasoning allows proving differential invariants which are not inductive [47], which cannot be proved otherwise [43]. For these reasons, our proof calculus (Section 5) includes solution, invariant, and ghost rules.

In contrast to a safety theorem, a liveness theorem would be shown by a progress argument. Suppose that Angel could set $L = 2$ but Demon can only choose $R \in \{-1, 1\}$. Then Angel's liveness theorem might say she can achieve $x = x_r$ if she as allowed to choose ODE duration, because the choice $L = 2$ ensures at least 1 unit of progress in $x$ for each unit of time.

## 4 Type-theoretic Semantics

We generalize the type-theoretic semantics of CdGL [11]. We define the semantics of the new refinement formulas $\alpha \leq^i_{[]} \beta$ and employ an infinite tower of type universes, in support of refinements. We first give our assumptions on the underlying type theory.

## 4.1 Type Theory Assumptions

We assume a Calculus of Inductive and Coinductive Constructions (CIC)-like type theory [17, 18, 50] with dependency and an infinite tower of cumulative predicative universes. Predicativity is essential because our semantics are a large elimination, which would interact dangerously with impredicative quantification. We assume first-class anonymous constructors for (indexed [21]) inductive and coinductive types. We write $M, N$ for type-theoretic terms, $\tau$ for type families, and $\kappa$ for kinds (those type families inhabited by other type families).

We write $\Pi x : \tau_1 .\, \tau_2$ for a dependent function type with argument named $x$ of type $\tau_1$ where return type $\tau_2$ may mention $x$. We write $\Sigma x : \tau_1 .\, \tau_2$ for a dependent pair type with left component named $x$ of type $\tau_1$ and right component of type $\tau_2$, possibly mentioning $x$. These specialize to the simple types $\tau_1 \Rightarrow \tau_2$ and $\tau_1 * \tau_2$ respectively when $x$ is not mentioned in $\tau_2$. Lambdas $(\lambda x : \tau.\ M)$ inhabit dependent function types. Pairs $(M, N)$ inhabit dependent pair types. Let-binding unpacks pairs and $\pi_L M$ and $\pi_R M$ are left and right projection. We write $\tau_1 + \tau_2$ for disjoint unions inhabited by $\ell \cdot M$ and $r \cdot M$, and write $\texttt{case}\ A\ \texttt{of}\ \ell \Rightarrow B \mid r \Rightarrow C$ for case analysis for $\tau_1 + \tau_2$, where $\ell$ and $r$ are variables over proofs.

We assume a type $\mathbb{R}$ for real numbers and type $\mathfrak{S}$ for Euclidean state vectors supporting scalar and vector sums, products, scalar inverses, and units. States $s, \hat{s} : \mathfrak{S}$ assign values to every variable $x \in \mathcal{V}$ and support the operations $s\ x$ for *retrieving* the value of $x$ and $\texttt{set}\ s\ x\ v$ for *updating* the value of $x$ to $v$. Likewise, $\texttt{set}\ s\ (x, y)\ (v, w)$ sets both $x$ and $y$ to $v$ and $w$, respectively. The usual axioms of setters and getters [26] are satisfied.

We write $\mathbb{T}_i$ for the $i$'th predicative universe. We use $P, Q : \mathfrak{S} \Rightarrow \mathbb{T}_i$ for variables over *regions*, e.g., the interpretations of formulas. Inductive type families are written $\mu P : \kappa .\, \tau$, which denotes the *smallest* solution $\texttt{ty}$ of kind $\kappa$ to the fixed-point equation $\texttt{ty} = [\texttt{ty}/P]\tau$. Coinductive type families are written $\rho P : \kappa .\, \tau$, which denotes the *largest* solution $\texttt{ty}$ of kind $\kappa$ to the fixed-point equation $\texttt{ty} = [\texttt{ty}/P]\tau$. The type expression $\tau$ must be monotone in $P$ to ensure that smallest and largest solutions exist, per Knaster-Tarski [29, Thm. 1.12]. Monotonicity of $\tau$ will require inductive proof in our case.

## 4.2 Semantics of CdGL

The interpretation of terms $f, g$ as functions of type $\mathfrak{S} \Rightarrow \mathbb{R}$ is standard. Games $\alpha$ and formulas $\phi$ require a notion of *rank* $\mathfrak{R}(\alpha \text{ or } \phi)$ indicating the smallest universe where $\alpha$ or $\phi$ has semantics. Universes are cumulative, so the semantics also belong to all universes $\mathbb{T}_i$ such that $i \geq \mathfrak{R}(\alpha)$. Refinement quantifies over types of a lower universe, which is predicative. A refinement formula's rank is given by its annotation: $\mathfrak{R}(\alpha \leq^i_{[]} \beta) = 1 + i$, requiring $\mathfrak{R}(\alpha), \mathfrak{R}(\beta) \leq i$. In all other cases, the rank is the maximum of ranks of subexpressions.

Formulas $\phi$ are interpreted as predicates over states, i.e., type families $\ulcorner\phi\urcorner : \mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\phi)}$. We say the formula $\phi$ is *valid* if there exists a CIC term $M : (\Pi s : \mathfrak{S}. \ulcorner\phi\urcorner\ s)$. CIC term $M$ is allowed to inspect state $s$, but only using computable operations. A natural deduction sequent $(\Gamma \vdash \phi)$ is valid iff implication formula $\bigwedge\Gamma \to \phi$ with conjunction $\bigwedge\Gamma$ is valid. The formula semantics are defined in terms of the active and dormant semantics of games, which determine how Angel wins a game $\alpha$ whose postcondition is a formula $\phi$ whose semantics are the goal region $\ulcorner\phi\urcorner$ (variable $P$ in Definition 5). We write $\langle\!\langle\alpha\rangle\!\rangle : (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)})$ for the active semantics of $\alpha$ and $[[\alpha]] : (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)})$ for its dormant semantics, which capture Angel's winning strategies when Angel is active or dormant, respectively. In contrast to classical game logics, the diamond and box modalities are *not* interdefinable constructively. The rank of an expression is only relevant in the refinement cases.

▶ **Definition 4** (Formula semantics). *Interpretation* $\ulcorner \phi \urcorner : \mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\phi)}$ *is defined by*

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

$$\ulcorner \alpha \leq^{i}_{[]} \beta \urcorner\ s = \big(\Pi P : (\mathfrak{S} \Rightarrow \mathbb{T}_i).\ \big([[\alpha]]\ P\ s \Rightarrow [[\beta]]\ P\ s\big)\big)$$

The modality $\langle\alpha\rangle\phi$ is true in state $s$ when active Angel has a strategy $\langle\langle\alpha\rangle\rangle\ \ulcorner\phi\urcorner\ s$ for game $\alpha$ from state $s$ to reach the region $\ulcorner\phi\urcorner$ on which $\phi$ has a proof. The modality $[\alpha]\phi$ is true in state $s$ when dormant Angel has a strategy $[[\alpha]]$ for game $\alpha$ from state $s$ to reach the region $\ulcorner\phi\urcorner$ on which $\phi$ has a proof. For comparison operators $\sim \in \{\leq, <, =, \neq, >, \geq\}$, the values of $f$ and $g$ are compared at state $s$. Game $\alpha$ demonically refines $\beta$ ($\alpha \leq_{[]} \beta$) from a state $s$ if for *all* goal regions $P$ there exists an effective mapping from dormant strategies $[[\alpha]]\ P\ s$ to dormant strategies $[[\beta]]\ P\ s$. The (defined) meaning of angelic refinement $\alpha \leq_{\langle\rangle} \beta$ is symmetric using diamond semantics $\langle\langle\alpha\rangle\rangle$. That is, refinements may depend on the state (they are *local* or *contextual*), but must hold for all goal regions $P$, as refinements consider the general game form itself, not a game fixed to a particular postcondition. Because refinement formulas are first-class, quantifiers may appear nested and in arbitrary positions, not necessarily prenex form. We ensure predicativity by requiring that refinements quantify only over postconditions of lower rank. Rank can be inferred in practice by inspecting a proof: each rank annotation need only be as large as the rank of every postcondition in every application of rules R$\langle\cdot\rangle$ and R$[\cdot]$ from Section 5.

The semantics of games are simultaneously inductive with those for formulas and with one another. In each case, the connectives which define $[[\alpha]]$ and $\langle\langle\alpha\rangle\rangle$ are duals, because $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are dual. Below, $P$ is the goal region and $s$ is the initial state.

▶ **Definition 5** (Active semantics). *The active interpretation* $\langle\langle\alpha\rangle\rangle$ *of the hybrid game* $\alpha$ *has kind* $(\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)})$ *and is defined by*

$$\begin{aligned}
\langle\langle ?\psi \rangle\rangle\ P\ s &= \ulcorner\psi\urcorner\ s * P\ s \\
\langle\langle x := f \rangle\rangle\ P\ s &= P\ (\mathsf{set}\ s\ x\ (f\ s)) \\
\langle\langle x := * \rangle\rangle\ P\ s &= \Sigma v : \mathbb{R}.\ (P\ (\mathsf{set}\ s\ x\ v)) \\
\langle\langle \alpha \cup \beta \rangle\rangle\ P\ s &= \langle\langle\alpha\rangle\rangle\ P\ s + \langle\langle\beta\rangle\rangle\ P\ s \\
\langle\langle \alpha;\beta \rangle\rangle\ P\ s &= \langle\langle\alpha\rangle\rangle\ (\langle\langle\beta\rangle\rangle\ P)\ s
\end{aligned}$$

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

$$\langle\langle \alpha^* \rangle\rangle\ P\ s = \big(\mu Q : (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}).\ \lambda\hat{s} : \mathfrak{S}.\ (P\ \hat{s} \Rightarrow Q\ \hat{s}) + (\langle\langle\alpha\rangle\rangle\ Q\ \hat{s} \Rightarrow Q\ \hat{s})\big)\ s$$

Angel wins $?\psi$ by proving both $\psi$ and $P$ at $s$. Angel wins the deterministic assignment $x := f$ by executing it, then proving $P$. Angel wins nondeterministic assignment $x := *$ by choosing a new value $v$, then proving $P$. Angel wins $\alpha \cup \beta$ by choosing to play game $\alpha$ or $\beta$, then winning it. Angel wins $\alpha;\beta$ by winning $\alpha$ with the postcondition of winning $\beta$. Angel wins $\alpha^d$ if she wins $\alpha$ in the dormant role. Angel wins ODE game $x' = f \,\&\, \psi$ by choosing some solution $y$ of some duration $d$ for which she proves domain constraint $\psi$ throughout and the goal region $P$ at time $d$. While top-level postconditions rarely mention $x'$, intermediate proof steps do, thus $x$ and $x'$ are both updated in the postcondition. The construct $(sol, s, d \vDash x' = f)$ says $sol$ solves $x' = f$ from state $s$ for time $d$ [12, App. A]. Active Angel strategies for $\alpha^*$ are inductively defined: either stop the loop and prove $P$ now, else play a round of $\alpha$ and repeat inductively. By Knaster-Tarski [29, Thm. 1.12], this least fixed point exists since games' semantics are monotone in the postcondition [11, Lem. 7].

▶ **Definition 6** (Dormant semantics). *The dormant interpretation* $[[\alpha]]$ *of the hybrid game* $\alpha$ *has kind* $(\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}) \Rightarrow (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)})$ *and is defined by*

$$\begin{aligned}
[[?\psi]]\ P\ s &= \ulcorner\psi\urcorner\ s \Rightarrow P\ s \\
[[x := f]]\ P\ s &= P\ (\textsf{set}\ s\ x\ (f\ s)) \\
[[x := *]]\ P\ s &= \Pi v : \mathbb{R}.\,(P\ (\textsf{set}\ s\ x\ v)) \\
[[\alpha \cup \beta]]\ P\ s &= [[\alpha]]\ P\ s\ \ast [[\beta]]\ P\ s \\
[[\alpha ; \beta]]\ P\ s &= [[\alpha]]\ ([[\beta]]\ P)\ s
\end{aligned}$$

$$\begin{aligned}
[[\alpha^d]]\ P\ s &= \langle\langle\alpha\rangle\rangle\ P\ s \\
[[x' = f \,\&\, \psi]]\ P\ s &= \Pi d : \mathbb{R}_{\geq 0}.\,\Pi sol : ([0,d] \Rightarrow \mathbb{R}). \\
&\quad (sol, s, d \vDash x' = f) \\
&\quad \Rightarrow \big(\Pi t : [0,d].\ulcorner\psi\urcorner\ (\textsf{set}\ s\ x\ (sol\ t))\big) \\
&\quad \Rightarrow P\ \big(\textsf{set}\ s\ (x, x') \\
&\qquad\qquad (sol\ d, f\ (\textsf{set}\ s\ x\ (sol\ d)))\big)
\end{aligned}$$

$$[[\alpha^*]]\ P\ s = \big(\rho Q : (\mathfrak{S} \Rightarrow \mathbb{T}_{\mathfrak{R}(\alpha)}).\,\lambda\hat{s} : \mathfrak{S}.\ \ (Q\ \hat{s} \Rightarrow [[\alpha]]\ Q\ \hat{s})\ \ast (Q\ \hat{s} \Rightarrow P\ \hat{s})\big)\ s$$

Angel wins $?\psi$ by proving $P$ under assumption $\psi$, which Demon must provide. Deterministic assignment is unchanged. Angel wins $x := *$ by proving $P$ for *every* choice of $x$. Angel wins $\alpha \cup \beta$ with a pair of winning strategies, since Demon chooses whether to play $\alpha$ or $\beta$. Angel wins $\alpha ; \beta$ by winning $\alpha$ with a postcondition of winning $\beta$. Angel wins $\alpha^d$ if she can win $\alpha$ actively. Angel wins $x' = f \,\&\, \psi$ if for an arbitrary duration and arbitrary solution which satisfy the domain constraint, Angel can prove the postcondition. Dormant repetition strategies are coinductive using some invariant region $Q$. When Demon decides to stop the loop, Angel responds by proving $P$ from $Q$. Whenever Demon chooses to continue, Angel proves that $Q$ is preserved. Greatest fixed points exist by Knaster-Tarski [29, Thm. 1.12] using monotonicity [11, Lem. 7].

In general, Angel strategies are constructive but permit Demon to play classically. In the cyber-physical setting, Demon is indeed rarely a computer.

## 5 Refinement Proof Calculus

We give a natural deduction calculus for hybrid game refinements. Refinement is relative to a context $\Gamma$ of CdGL formulas, which may include refinements. All rules are expressed as Demonic refinements $\alpha \leq^i_{[]} \beta$, but an Angelic refinement $\alpha \leq^i_{\langle\rangle} \beta$ is supported by refining the duals $\alpha^d \leq^i_{[]} \beta^d$. Remember that in a Demonic refinement, the Angelic (existential) connectives appear under dualities $\alpha^d$. We write $\alpha \cong \beta$ for $\alpha \leq_{[]} \beta \wedge \beta \leq_{[]} \alpha$. Recall that hybrid *systems* are hybrid games which do not contain the dual operator $\alpha^d$.

The refinement elimination rules R$\langle\cdot\rangle$ and R$[\cdot]$ say every true postcondition $\phi$ of a game $\alpha$ is a true postcondition of every $\beta$ which $\alpha$ refines. The side condition for R$\langle\cdot\rangle$ and R$[\cdot]$ is that $\mathfrak{R}(\phi) \leq i$ where $i$ is the rank annotation of the refinement. These are the only rules which care about rank, so ranks can be inferred from proofs by inspecting the uses of these rules. While rank is of little practical import, it ensures a predicative formal foundation.

Figure 1 gives the refinement rules for discrete connectives. Soundness of game refinement rules is subtle because games are subnormal. Sound game refinement rules can be divided into two classes: either refine games globally by requiring an empty context (;G), or restrict some subgames to be systems (;S). The formal approach follows game algebra [28] and is required when comparing two games. The latter approach generalizes dR$\mathcal{L}$ [36] and is necessary when reifying sequential games. By combining both approaches, our calculus is strictly more complete than both game algebra and dR$\mathcal{L}$. Unlike dR$\mathcal{L}$ [36], we face the challenge that game logics are subnormal and subregular [30]: For a game $\alpha$, formula $[\alpha](\phi \wedge \psi)$ need not hold when both $[\alpha]\phi$ and $[\alpha]\psi$ do.

$$\mathrm{R}\langle\cdot\rangle\ \frac{\Gamma\vdash\langle\alpha\rangle\phi\quad\Gamma\vdash\alpha\leq^{i}_{\langle\rangle}\beta}{\Gamma\vdash\langle\beta\rangle\phi}\,{}^{1}\qquad\qquad \mathrm{R}[\cdot]\ \frac{\Gamma\vdash[\alpha]\phi\quad\Gamma\vdash\alpha\leq^{i}_{[]}\beta}{\Gamma\vdash[\beta]\phi}\,{}^{2}$$

$$;\mathrm{S}\ \frac{\Gamma\vdash\boldsymbol{\alpha}_1\leq_{[]}\alpha_2\quad\Gamma\vdash[\boldsymbol{\alpha}_1]\beta_1\leq_{[]}\beta_2}{\Gamma\vdash\boldsymbol{\alpha}_1;\beta_1\leq_{[]}\alpha_2;\beta_2}\,{}^{2}\qquad\qquad \langle?\rangle\ \frac{\Gamma\vdash\phi\to\psi}{\Gamma\vdash ?\phi^{d}\leq_{[]}?\psi^{d}}$$

$$;\mathrm{G}\ \frac{\Gamma\vdash\alpha_1\leq_{[]}\alpha_2\quad\cdot\vdash\beta_1\leq_{[]}\beta_2}{\Gamma\vdash\alpha_1;\beta_1\leq_{[]}\alpha_2;\beta_2}\qquad\qquad [?]\ \frac{\Gamma\vdash\psi\to\phi}{\Gamma\vdash ?\phi\leq_{[]}?\psi}$$

$$[\cup]\mathrm{L1}\ \Gamma\vdash\alpha\cup\beta\leq_{[]}\alpha\qquad\qquad \langle\cup\rangle\mathrm{R1}\ \Gamma\vdash\alpha^{d}\leq_{[]}\{\alpha\cup\beta\}^{d}$$

$$[\cup]\mathrm{L2}\ \Gamma\vdash\alpha\cup\beta\leq_{[]}\beta\qquad\qquad \langle\cup\rangle\mathrm{R2}\ \Gamma\vdash\beta^{d}\leq_{[]}\{\alpha\cup\beta\}^{d}$$

$$[\cup]\mathrm{R}\ \frac{\Gamma\vdash\alpha\leq_{[]}\beta\quad\Gamma\vdash\alpha\leq_{[]}\gamma}{\Gamma\vdash\alpha\leq_{[]}\beta\cup\gamma}\qquad\qquad \langle\cup\rangle\mathrm{L}\ \frac{\Gamma\vdash\alpha^{d}\leq_{[]}\gamma\quad\Gamma\vdash\beta^{d}\leq_{[]}\gamma}{\Gamma\vdash\{\alpha\cup\beta\}^{d}\leq_{[]}\gamma}$$

$$\langle{:}*\rangle\ \Gamma\vdash x:=f^{d}\leq_{[]}x:=*^{d}\qquad\qquad \mathrm{un}*\ \frac{\Gamma\vdash[\boldsymbol{\alpha}^{*}](\boldsymbol{\alpha}\leq_{[]}\beta)}{\Gamma\vdash\boldsymbol{\alpha}^{*}\leq_{[]}\beta^{*}}\,{}^{2}$$

$$[{:}*]\ \Gamma\vdash x:=*\leq_{[]}x:=f\qquad\qquad \mathrm{roll}_l\ \Gamma\vdash\mathsf{skip}\cup\{\alpha;\alpha^{*}\}\cong\alpha^{*}$$

$$\mathrm{skip}^{d}\ \mathsf{skip}^{d}\cong\mathsf{skip}\qquad ;^{d}\ \{\alpha;\beta\}^{d}\cong\alpha^{d};\beta^{d}\qquad :=^{d}\ x:=f^{d}\cong x:=f\qquad \mathrm{DDE}\ \{\alpha^{d}\}^{d}\cong\alpha$$

1 assuming $\mathfrak{R}(\phi)\leq i$
2 $\boldsymbol{\alpha}_1$ respectively $\boldsymbol{\alpha}$ is a hybrid system

**Figure 1** Refinement of discrete connectives

Bold variables only range over systems, e.g., $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}_1$ in rules ;S and un$*$. One sequence refines another piecewise in the ;S rule, which is *contextual*: refinement of the second component exploits the fact that the first component has been executed. Rule ;G is a variant of ;S which says $\alpha_1$ can be an arbitrary game, but only if $\beta_1\leq_{[]}\beta_2$ holds in the empty context. System $\boldsymbol{\alpha}_1$ in the second premiss of ;S could soundly be $\alpha_2$, but in practical proofs it is often more convenient to work with $[\boldsymbol{\alpha}_1]$ because it is a *system* modality, which is normal. Rules $\langle?\rangle$ and $[?]$ refine tests by weakening or strengthening test conditions. The left and right rules for choices are dual. Rules $\langle\cup\rangle$R1 and $\langle\cup\rangle$R2 say each branch refines an Angelic choice, while $[\cup]$R says a Demonic choice is refined by refining both branches. Rules $\langle{:}*\rangle$ and $[{:}*]$ say that deterministic assignments refine nondeterministic ones. Rule un$*$ compares loops by comparing their bodies and roll$_l$ allows unrolling a loop before refining. Rules skip$^d$, $:=^d$, and $;^d$ say $\mathsf{skip}$ and $x:=f$ are self-dual and the dual of a sequence is a sequence of duals. Double duals cancel by DDE.

The rules in Figure 2 are selected algebraic properties which will be used in the proof of Theorem 10. These rules generalize known game equalities [28] to refinement. Some rules of dR$\mathcal{L}$ [36] are reused here, but other rules of dR$\mathcal{L}$, such as those for repetitions $\alpha^*$ are not sound for arbitrary games. Rules refl and trans say refinement is a partial order. Sequential composition has identities (;id$_l$ and ;id$_r$). Rule :=:= deduplicates a double assignment if the first assignment does not influence the second: FV$(f)$ are the *free variables* mentioned in $f$. Choice ($\cup$A) and sequence (;A) are associative, and choice is commutative ($\cup$c) and idempotent ($\cup$idem), while sequence is right-distributive (;d$_r$). Impossible tests can annihilate any following program (annih$_l$). Assigning a variable to itself is a no-op (:=nop).

Figure 3 gives the ODE refinement rules. *Differential cut* DC says the domain constraints $\phi$ and $\phi\wedge\psi$ are equivalent if $\psi$ holds as a postcondition under domain constraint $\phi$. *Differential*

$$\text{trans}\ \frac{\Gamma \vdash \alpha \leq_{[]} \beta \quad \Gamma \vdash \beta \leq_{[]} \gamma}{\Gamma \vdash \alpha \leq_{[]} \gamma} \qquad ;\mathrm{d}_r\ \ \Gamma \vdash \{\alpha \cup \beta\}; \gamma \cong \{\alpha; \gamma\} \cup \{\beta; \gamma\}$$

$$\text{refl}\ \ \Gamma \vdash \alpha \leq_{[]} \alpha \qquad ;\mathrm{A}\ \ \Gamma \vdash \{\alpha; \beta\}; \gamma \cong \alpha; \{\beta; \gamma\}$$

$$;\mathrm{id}_l\ \ \Gamma \vdash \{\mathsf{skip}; \alpha\} \cong \alpha \qquad :=:=\ \ \Gamma \vdash x := f; x := g \cong x := g\ ^{1}$$

$$;\mathrm{id}_r\ \ \Gamma \vdash \{\alpha; \mathsf{skip}\} \cong \alpha \qquad \cup\mathrm{A}\ \ \Gamma \vdash \{\alpha \cup \beta\} \cup \gamma \cong \alpha \cup \{\beta \cup \gamma\}$$

$$\mathrm{annih}_l\ \ \Gamma \vdash ?\mathtt{ff}; \alpha \cong ?\mathtt{ff} \qquad \cup\mathrm{c}\ \ \Gamma \vdash \alpha \cup \beta \cong \beta \cup \alpha$$

$$:=\mathrm{nop}\ \ \Gamma \vdash \{x := x\} \cong \mathsf{skip} \qquad \cup\mathrm{idem}\ \ \Gamma \vdash \alpha \cup \alpha \cong \alpha$$

[1] for $x \notin \mathrm{FV}(g)$

**Figure 2** Algebraic rules (selected)

$$\text{DC}\ \frac{\Gamma \vdash [x' = f \,\&\, \phi]\psi}{\Gamma \vdash \{x' = f \,\&\, \phi\} \cong \{x' = f \,\&\, \phi \land \psi\}} \qquad \text{DW}\ \ \Gamma \vdash \{x := *; x' := f; ?\psi\} \leq_{[]} \{x' = f \,\&\, \psi\}$$

$$\text{solve}\ \frac{\Gamma \vdash [t := *; ?0 \leq t \leq d; x := sln]\psi}{\Gamma, t = 0, d \geq 0 \vdash \{t := d; x := sln; t' := 1; x' := f\} \leq_{[]} \{t' = 1, x' = f \,\&\, \psi\}^d}\ ^{1}$$

$$\text{DG}\ \ \Gamma \vdash \{y := f_0; x' = f, y' = a(x)y + b(x) \,\&\, \psi\} \leq_{[]} \{x' = f \,\&\, \psi; \{y := *; y' := *\}^d\}$$

[1] $sln$ solves ODE, $\{t, t', x, x'\} \cap FV(d) = \emptyset$

**Figure 3** Differential equation refinements

*weakening* DW says an ODE is overapproximated by the program which assumes only the domain constraint. *Differential solution* solve says that a solvable Angelic ODE $x' = f \,\&\, \psi$ with syntactic solution term $sln$ is refined by a deterministic program which assigns the solution to $x$ after through which the domain constraint holds, specified by a term $d$ which is constant throughout the ODE. Here $sln = (\lambda s : \mathfrak{S}.\ (sol\ (s\ t)))$ is the term corresponding to the semantic solution $sol$ at time $t$. *Differential ghosts* DG soundly augments an ODE with a fresh dimension $y$ so long as the solution for $y$ exists as long as that of $x$, and is known [47] to enable proofs of otherwise unprovable [43] properties. The right-hand side for $y$ is required to be linear in $y$ because this suffices to ensure sufficient duration. Axiom DG is not an equivalence because linear ODEs do not suffice to reach every of the nondeterministically assigned final values for $y$ and $y'$ [42].

## 6 Theory

We develop theoretical results about CdGL refinements: soundness and the relationship between games and systems. Proofs are in a companion report [12].

### 6.1 Soundness

The *sine qua non* condition of any logic is soundness. We show that every formula provable in the CdGL refinement calculus is true in the type-theoretic semantics.

▶ **Theorem 7** (Soundness). *If $\Gamma \vdash \phi$ is provable then the sequent $(\Gamma \vdash \phi)$ is valid.*

## 6.2 Reification

A game $\alpha$ describes what actions are allowed for each player but not how Angel selects among them given an adversarial Demon. Every game modality proof, whether of $[\alpha]\phi$ or $\langle\alpha\rangle\phi$, lets Demon make arbitrary (universally-quantified) moves within the confines of the game, and describes Angel's strategy to achieve a given postcondition $\phi$. Whereas a given game can contain both Angelic and Demonic choices, a *system* can only contain one or the other: modality $[\boldsymbol{\alpha}]\phi$ treats a system $\boldsymbol{\alpha}$ as Demonic while $\langle\boldsymbol{\alpha}\rangle\phi$ treats a system as Angelic.

A folklore theorem describes the relation between hybrid games and hybrid systems: given a proof (winning strategy) for a hybrid game, one can *reify* Angel's strategy to produce a hybrid *system* which implements that strategy. The constructivity of CdGL ensures that Angel's choices are implementable by computable functions. Since Demonic choices survive reification, it is simplest to work with Demonic game modalities $[\alpha]\phi$ here, but every Angelic game modality $\langle\alpha\rangle\phi$ could equivalently be expressed as $[\alpha^d]\phi$. In this section, we formally define the reification operation and prove its relation to the source game using refinements and a derivation $\mathcal{A}$ in the CdGL (non-refinement) proof calculus. For the sake of space, we present CdGL rules informally as we define reification. Our presentation differs in insignificant ways from the full calculus from prior work [11]; it is convenient for our purposes that premisses eliminate as many connectives as possible, as is common in natural-deduction style. Let $\mathcal{A}$ be a CdGL proof of some CdGL formula $[\alpha]\phi$ in context $\Gamma$, i.e., let $\Gamma \vdash \mathcal{A} \colon [\alpha]\phi$. We then write $\mathcal{A} \leadsto \boldsymbol{\alpha}$ to say the (unique) result of reifying the strategy given by $\mathcal{A}$ into hybrid game $\alpha$ is the hybrid *system* $\boldsymbol{\alpha}$. The system $\boldsymbol{\alpha}$ needs to commit to Angel's strategy according to $\mathcal{A}$ while retaining all available choices of Demon. What properties ought $\boldsymbol{\alpha}$ satisfy?

Committing to a safe Angel strategy should never make the system less safe. The safety postcondition $\phi$ should *transfer* to $\boldsymbol{\alpha}$, i.e., the following property should hold:

If $\Gamma \vdash \mathcal{A} \colon [\alpha]\phi$ and $\mathcal{A} \leadsto \boldsymbol{\alpha}$ then $(\Gamma \vdash [\boldsymbol{\alpha}]\phi)$ is provable.

Transfer alone does not capture reification, e.g., defining $\boldsymbol{\alpha} = ?\mathtt{ff}$ for all $\alpha$ and $\mathcal{A}$ would vacuously satisfy the transfer property but certainly not capture the meaning of strategy $\mathcal{A}$.

We, thus, guarantee a converse direction. The reified hybrid system $\boldsymbol{\alpha}$ is a *safety refinement* of hybrid game $\alpha$, so *every* postcondition $\psi$ satisfying $[\boldsymbol{\alpha}]\psi$ also satisfies $[\alpha]\psi$:

If $\Gamma \vdash \mathcal{A} \colon [\alpha]\phi$ and $\mathcal{A} \leadsto \boldsymbol{\alpha}$ then $(\Gamma \vdash \boldsymbol{\alpha} \leq_{[\,]} \alpha)$ is provable.

Intuitively, $[\boldsymbol{\alpha}]\psi$ says postcondition $\psi$ holds for every Demon behavior of $\boldsymbol{\alpha}$, while $[\boldsymbol{\alpha}]\psi$ holds if there *exists* an Angel strategy that ensures $\psi$ for every Demon behavior of $\alpha$. Since derivation $\mathcal{A}$ is designed to satisfy $\psi$, there certainly exists a strategy that satisfies $\psi$. Refinement captures the notion that Angelic choices in $\boldsymbol{\alpha}$ are made more strictly than in $\alpha$, while Demonic choices are only made more loosely.

Even transfer *and* refinement do not fully validate the reification operation, since defining $\boldsymbol{\alpha} = \alpha$ suffices to ensure both. This leads to a third, most obvious property: $\boldsymbol{\alpha}$ must be a system when $\alpha$ is a game. Not only are systemhood, transfer, and refinement all desirable properties for reification, but their combination is an appealing specification because there is no trivial operation which satisfies all three. If the above three properties hold, they also imply a sound version of the normal modal logic axiom K that is elusive in games: If $\Gamma \vdash \mathcal{A} : [\alpha]\phi$ and $\Gamma \vdash [\boldsymbol{\alpha}]\psi$ is provable for $\mathcal{A} \leadsto \boldsymbol{\alpha}$, then $\Gamma \vdash [\alpha](\phi \land \psi)$ is provable. Additionally, transfer and systemhood suggest that game synthesis can "export" a game proof to a systems proof, for which synthesis tools already exist [39, 13]. We discuss some technicalities first.

## Technicalities

Reification accepts a CdGL proof and returns a system. In each case of its inductive definition, we write $\mathcal{A}, \mathcal{B}, \mathcal{C}$ for the proofs of each premiss and $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ for corresponding output systems. For simplicity, we reify *nested* modalities: reifying a proof of $[\alpha_1][\alpha_2]\phi$ results in a system $\boldsymbol{\alpha}$ which refines $\alpha_1; \alpha_2$. Angelic programs are represented by duality $\alpha^d$ and the reification of first-order $\phi$ is a no-op skip. This style is interchangeable with normal-form CdGL proofs; we elide the duality ($[\alpha^d]\phi \leftrightarrow \langle\alpha\rangle\phi$) and skip ($[\mathsf{skip}]\phi \leftrightarrow \phi$) steps which convert between the two. Prior work [10] shows case-analysis, which is not *canonical*, is sometimes *normal* because state-dependent cases are decided only at runtime. Normal case analyses are analogous to case-tree normal forms in lambda calculi with coproducts [3]. Normal forms of (classical) ODE proofs have been characterized [9]. We call a game *system-test* if all its tests and domain constraints are system-test formulas. A formula is *system-test* if all modalities it mentions are box system modalities, while a proof is *system-test* if every context in its proof tree contains only system-test formulas. For the sake of defining system-test, the first-order propositional connectives are considered distinct from game modalities, rather than defined, i.e., first-order arithmetic expressions are permissible in the system-test fragment. Restricting reification to the system-test fragment ensures the reification of the hypothesis rule hyp is a system. System-test is stronger than *weak-test* (no modalities in tests) but weaker than *strong-test* (arbitrary modalities in tests). In the proof rules that follow, $\cdot\frac{y}{x}$ is the renaming of variable $x$ to (usuallly fresh) variable $y$ in a term, formula, game, or context.

## Definitions

We define reification. Reification $\mathcal{A} \leadsto \boldsymbol{\alpha}$ is defined inductively on box CdGL proofs of system-test games. We first give the reification of case-analysis and hypothesis proofs, the only two normal proofs which are not introduction forms. In $\vee$E, $\mathcal{A}$ proves some first-order disjunction $\phi \vee \psi$, since proper choice game modalities $\langle\alpha \cup \beta\rangle$ are not permitted in system-test, normal-form proofs. In $\vee$E, the reified systems for the second and third premiss are $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, likewise in every rule.

$$\text{hyp}\,\frac{(\text{if } [\alpha]\phi \in \Gamma)}{\Gamma \vdash [\alpha]\phi} \leadsto \alpha \qquad\qquad \vee\text{E}\,\frac{\Gamma \vdash \phi \vee \psi \quad \Gamma, \phi \vdash \varphi \quad \Gamma, \psi \vdash \varphi}{\Gamma \vdash \varphi} \leadsto \{?\phi; \boldsymbol{\beta}\} \cup \{?\psi; \boldsymbol{\gamma}\}$$

Hypothesis proofs do not give a concrete strategy for $\alpha$ and thus trivially refine $\alpha$ to itself. Case analysis allows Demon to choose either branch, so long as it is provable. The output is nondeterministic if $\phi$ and $\psi$ are not mutually exclusive. Both $\phi$ and $\psi$ are game-free in the system-test fragment and, in practical proofs, even quantifier-free first-order arithmetic. We first give the discrete *Angelic* cases, which plug in the specific Angel strategy from proof $\mathcal{A}$.

$$\langle\!\!\langle{:=}\rangle\!\!\rangle\text{I}\,\frac{\Gamma\frac{y}{x}, x = f\frac{y}{x} \vdash \phi}{\Gamma \vdash [\{x := f\}^d]\phi} \leadsto x := f; \boldsymbol{\alpha} \quad \langle{:*}\rangle\text{I}\,\frac{\Gamma\frac{y}{x}, x = f\frac{y}{x} \vdash \phi}{\Gamma \vdash [\{x := *\}^d]\phi} \leadsto x := f; \boldsymbol{\alpha} \quad \langle ? \rangle\text{I}\,\frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash [\{?\phi\}^d]\psi} \leadsto \boldsymbol{\beta}$$

$$\langle\!\!\langle ; \rangle\!\!\rangle\text{I}\,\frac{\Gamma \vdash [\alpha^d][\beta^d]\phi}{\Gamma \vdash [\{\alpha; \beta\}^d]\phi} \leadsto \boldsymbol{\alpha} \qquad \langle\cup\rangle\text{IL}\,\frac{\Gamma \vdash [\alpha^d]\phi}{\Gamma \vdash [\{\alpha \cup \beta\}^d]\phi} \leadsto \boldsymbol{\alpha} \qquad \langle\cup\rangle\text{IR}\,\frac{\Gamma \vdash [\beta^d]\phi}{\Gamma \vdash [\{\alpha \cup \beta\}^d]\phi} \leadsto \boldsymbol{\alpha}$$

$$\langle * \rangle\text{I}\,\frac{\begin{array}{c}\Gamma \vdash \varphi \\ \varphi, \mathcal{M} \succ \mathbf{0} \wedge \mathcal{M}_0 = \mathcal{M} \vdash [\alpha^d](\varphi \wedge \mathcal{M}_0 \succ \mathcal{M}) \quad \varphi, \mathbf{0} \succcurlyeq \mathcal{M} \vdash \phi\end{array}}{\Gamma \vdash [\{\alpha^*\}^d]\phi} \leadsto \{?\mathcal{M} \succ \mathbf{0}; \boldsymbol{\beta}\}^*; ?\mathbf{0} \succcurlyeq \mathcal{M}; \boldsymbol{\gamma}$$

Discrete assignments remain in the output. Nondeterministic assignments are proved by providing a witness term $f$ (rule $\langle{:*}\rangle$), which is preserved by the inductive call $\boldsymbol{\alpha}$. Subtly, Angelic tests can be eliminated by $\langle ? \rangle$ because they are *proven to succeed* and because we wish only to keep tests which *Demon* is required to pass. A normal-form proof for a sequential composition $\alpha;\beta$ proves $\alpha$ with $\beta$ in the postcondition. Normal Angelic choice proofs are injections, so Angelic proofs reify by $\langle\cup\rangle$R1 or $\langle\cup\rangle$R2 according to one branch or the other. Normal Angelic repetition proofs are by convergence: some metric $\mathcal{M}$ decreases to terminal value $\mathbf{0}$ while maintaining invariant formula $\varphi$. Variable $\mathcal{M}_0$ remembers the value of $\mathcal{M}$ at the start of each loop iteration for comparison purposes. Hybrid systems loops are nondeterministic, so Demon chooses the loop duration, but the Demonic test $\mathcal{M} \succ \mathbf{0}$ must pass at each repetition and $\mathbf{0} \not\succ \mathcal{M}$ must pass at the end, determinizing the loop duration.

To reify a discrete *Demonic* connective, we do not restrict Demon's capabilities, but recursively traverse the proof so that Angelic proofs can be reified.

$$\langle\!\langle{:=}\rangle\!\rangle\text{I}\,\frac{\Gamma\frac{y}{x}, x = f\frac{y}{x} \vdash \phi}{\Gamma \vdash [x := f]\phi} \rightsquigarrow x := f; \boldsymbol{\alpha} \qquad [{:}*]\frac{\Gamma\frac{y}{x} \vdash \phi}{\Gamma \vdash [x := *]\phi} \rightsquigarrow x := *; \boldsymbol{\alpha} \qquad \langle\!\langle ; \rangle\!\rangle\text{I}\,\frac{\Gamma \vdash [\alpha][\beta]\phi}{\Gamma \vdash [\alpha;\beta]\phi} \rightsquigarrow \boldsymbol{\alpha}$$

$$[?]\text{I}\,\frac{\Gamma, \psi \vdash \phi}{\Gamma \vdash [?\psi]\phi} \rightsquigarrow ?\psi; \boldsymbol{\alpha} \quad [\cup]\text{I}\,\frac{\Gamma \vdash [\alpha]\phi \quad \Gamma \vdash [\beta]\phi}{\Gamma \vdash [\alpha \cup \beta]\phi} \rightsquigarrow \boldsymbol{\alpha} \cup \boldsymbol{\beta} \quad [*]\text{I}\,\frac{\Gamma \vdash \psi \quad \psi \vdash [\alpha]\psi \quad \psi \vdash \phi}{\Gamma \vdash [\alpha^*]\phi} \rightsquigarrow \boldsymbol{\beta}^*; \boldsymbol{\gamma}$$

Nondeterministic Demonic assignments, unlike Angelic ones, are not modified during reification, because Demon retains the power to choose any value. Demonic tests introduce assumptions, and must continue to do so in the reification system to avoid changing the acceptable behavior. Demonic sequential compositions are like Angelic ones. Demonic choices refine each branch. Note that reification of games with form $\{\alpha \cup \beta\}; \gamma$ follows distributive normal forms $\{\alpha;\gamma\} \cup \{\beta;\gamma\}$, which are equivalent by ;d$_r$. Demonic repetitions keep the loop, recalling that the coinductive loop invariant $\psi$ justifies the postcondition by premiss $\mathcal{C}$.

We give the reification cases for ODEs. The reification of an invariant-based Demonic proof (dc and dw) is a *relaxation* of the ODE: the reified system need not follow the precise behavior of the ODE so long as all invariants required for the proof are obeyed. Indeed, this is where proof-based synthesis in ModelPlex [39] gains much of its power: real implementations never follow an ODE with perfect precision, but usually do follow its invariant-based relaxation.

$$\langle ' \rangle\,\frac{\begin{array}{c}\Gamma \vdash d \geq 0 \quad \Gamma\frac{y}{x}, 0 \leq t \leq d, x = sln\frac{y}{x}, x' = f \vdash \psi \\ \Gamma\frac{y}{x}, 0 \leq t = d, x = sln\frac{y}{x}, x' = f \vdash \phi\end{array}}{\Gamma \vdash [t := 0; \{t' = 1, x' = f \,\&\, \psi\}^d]\phi} \rightsquigarrow t := d; x := sln; x' := f; \boldsymbol{\gamma}$$

$$[']\,\frac{\Gamma\frac{y}{x}, t \geq 0, \hat{\psi}, x = sln\frac{y}{x}, x' = f \vdash \phi}{\Gamma \vdash [t := 0; \{t' = 1, x' = f \,\&\, \psi\}]\phi} \rightsquigarrow t := 0; \{t' = 1, x' = f \,\&\, \psi\}; \boldsymbol{\alpha}$$

$$\text{dw}\,\frac{\Gamma\frac{y}{x}, \psi \vdash \phi}{\Gamma \vdash [x' = f \,\&\, \psi]\phi} \rightsquigarrow x := *; x' := f; ?\psi; \boldsymbol{\alpha}$$

$$\text{dc}\,\frac{\Gamma \vdash [x' = f \,\&\, \psi]\varphi \quad \Gamma \vdash [x' = f \,\&\, \psi \wedge \varphi]\phi}{\Gamma \vdash [x' = f \,\&\, \psi]\phi} \rightsquigarrow \boldsymbol{\beta}$$

$$\text{dg}\,\frac{\Gamma, y = f_0 \vdash [x' = f, y' = a(x)y + b(x) \,\&\, \psi]\phi}{\Gamma \vdash [x' = f \,\&\, \psi; \{y := *; y' := *\}^d]\phi} \rightsquigarrow y := f_0; \boldsymbol{\alpha}$$

Variable $y$ is fresh in $\langle ' \rangle$, $[']$, and dg. In $\langle ' \rangle$, the side condition requires that $y$ is fresh, term *sln* is the unique solution of the ODE, and chosen duration term $d$ is constant throughout the ODE. The Angelic domain constraint is comparable to an Angelic test: it is soundly omitted in the reification because it is proven to pass. In Demonic ODE solutions ($[']$), the duration and domain constraint are *assumptions*, and formula $\hat{\psi} \equiv \forall 0 \leq s \leq t\, [t := s; x := sln]\psi$ says the domain constraint $\psi$ holds through time $t$ where $s$ is fresh. Since our ODEs are Lipschitz, they have unique solutions and Demon could reify the unique solution of the ODE, as does case ($\langle ' \rangle$). There is no obvious benefit to doing so, except that the reified system would fall within *discrete* dynamic logic. Differential Cut (dc) reification introduces an assumption in the domain constraint, and is sound by DC. By itself, dc *strengthens* a program, but in combination with dw enables relaxation of ODEs. Differential Weakening (dw) relaxes an ODE by allowing $x$ and $x'$ to change *arbitrarily* so long as the domain constraint $\psi$ (and thus invariants introduced by dc) remain true. Differential Ghost (dg) introduces a dimension to the ODE and reifies according to the recursive call. The introduced dimension is linear in order to soundly preserve the duration of the ODE. Assignment $y := f_0$ sets the initial value of the ghost variable to a chosen term.

### Reification Example

Recall example PP and its safety property (1). Let $\mathcal{A}_{\mathsf{PP}}$ be the proof of (1) with a mirroring strategy described in Section 3.2. Then the reified result $\boldsymbol{\alpha}_{\mathsf{PP}}$ is

$$\begin{aligned}\boldsymbol{\alpha}_{\mathsf{PP}} = \big\{&\{L := -1; R := 1; x' = L + R \,\&\, x_l \le x \le x_r\}\\ \cup&\{L := 1; R := -1; x' = L + R \,\&\, x_l \le x \le x_r\}\big\}^*\end{aligned}$$

which we discuss step-by-step. Demonic repetition reification just repeats the body. Reifying a Demonic choice follows the structure of the proof, not the source program, hence the ODE occurs for each branch. Each branch commits to a choice of $L$, and each branch of $\mathcal{A}_{\mathsf{PP}}$ resolves the Angelic choice $R$ to balance out $L$. When reifying an Angelic choice, only the branch taken is emitted. In $\boldsymbol{\alpha}_{\mathsf{PP}}$, we assume that $\mathcal{A}_{\mathsf{PP}}$ proves the ODE $x' = L + R \,\&\, x_l \le x \le x_r$ by replacing it with its solution, which is why the ODE appears verbatim in the refined system. A differential invariant proof could also be used with a differential cut (DC) of $x = x_0$, in which case physics are represented by the program $x := *; x' := *; ?x_l \le x \le x_r \wedge x = x_0$ in the result of reification. Different proofs generally give rise to different systems, some of which are less restrictive than others. Differential invariants, especially inequational invariants, ($x \ge x_0$ vs. $x = x_0$) can be more easily monitored with finite-precision numbers.

Note that the system $\boldsymbol{\alpha}_{\mathsf{PP}}$ is a refinement of PP and satisfies the same safety theorem $\mathsf{pre} \to [\boldsymbol{\alpha}_{\mathsf{PP}}]x = x_0$. Next, we show that this is the case for all reified strategies.

### Metatheoretic Results

We state theorems (proven in our report [12]) showing how the reification of a game $\alpha$ refines $\alpha$. Recall that $\Gamma, \alpha, \phi$, and $\mathcal{A}$ are in the *system-test* fragment of CdGL.

▶ **Theorem 8** (Systemhood). *If* $\Gamma \vdash \mathcal{A} : [\alpha]\phi$ *for system-test* $\Gamma, \mathcal{A}$, *and hybrid game* $\alpha$ *and* $\mathcal{A} \rightsquigarrow \boldsymbol{\alpha}$ *then* $\boldsymbol{\alpha}$ *is a system, i.e., it does not contain dualities.*

▶ **Theorem 9** (Reification transfer). *If* $\Gamma \vdash \mathcal{A} : [\alpha]\phi$ *for system-test* $\Gamma, \mathcal{A}$, *and hybrid game* $\alpha$ *and* $\mathcal{A} \rightsquigarrow \boldsymbol{\alpha}$ *then* $\Gamma \vdash [\boldsymbol{\alpha}]\phi$ *is provable in CdGL.*

▶ **Theorem 10** (Reification refinement). *If* $\Gamma \vdash \mathcal{A} : [\alpha]\phi$ *for system-test* $\Gamma, \mathcal{A}$, *and hybrid game* $\alpha$ *and* $\mathcal{A} \rightsquigarrow \boldsymbol{\alpha}$ *then* $\Gamma \vdash \boldsymbol{\alpha} \le_{[]} \alpha$ *is provable in CdGL.*

Theorem 8 is proven by trivial induction on $\mathcal{A}$. Theorem 9 is proven by inducting on $\mathcal{A}$, reusing its contents in a proof for $\boldsymbol{\alpha}$. Theorem 10 inducts on $\mathcal{A}$ and in each case appeals to the corresponding refinement rule. The fact that Theorem 10 could be proved validates the strength of CdGL's refinement rules.

## 7 Conclusion

We developed a refinement calculus for Constructive Differential Game Logic (CdGL). Technical challenges in this development included the facts that game logic is subnormal and that the *constructive* box and diamond modalities $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are not interdefinable. We introduced a new constructive semantics for refinement and proved soundness. We formalized a reification operation and folklore theorem which reduce verified hybrid games to hybrid systems by specializing a game to the commitments made by its winning strategy. The immediate applications are synthesis tools and refinement-based proof tools for hybrid games. Theorem 8 and Theorem 9 support synthesis by ensuring that the reified system is a *system* which satisfies the *same* safety condition as the input game, which are respectively required in order to use existing synthesis tools and to ensure an end-to-end safety guarantee. Theorem 10 supports refinement-based proof technology: because the reified system refines the input game, game safety can be shown by choosing a strategy and showing the strategy safe. Once these tools are implemented, there are a wide array of applications studied in the hybrid systems and hybrid games literature which would benefit from the modeling power and synthesis guarantees that are possible with CdGL.

Our refinement calculus is of theoretical and practical interest beyond reducing games to systems. We expect that refinements can be used to provide shorter proofs, to compare the efficacy (dominance) of two strategies for the same game, and to determine when two strategies or programs should be considered "the same". These questions are worth pursuing both for hybrid games and for games in general.

## References

1 Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000. doi:10.1006/inco.2000.2930.

2 Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering.* Cambridge University Press, 2010. doi:10.1017/CBO9781139195881.

3 Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip J. Scott. Normalization by evaluation for typed lambda calculus with coproducts. In *LICS*, pages 303–310. IEEE, 2001. doi:10.1109/LICS.2001.932506.

4 Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. Alternating-time temporal logic. *J. ACM*, 49(5):672–713, 2002. doi:10.1145/585265.585270.

5 Ralph-Johan Back and Joakim von Wright. *Refinement Calculus - A Systematic Introduction.* Graduate Texts in Computer Science. Springer, 1998. doi:10.1007/978-1-4612-1674-2.

6 Richard Banach, Michael J. Butler, Shengchao Qin, Nitika Verma, and Huibiao Zhu. Core Hybrid Event-B I: Single Hybrid Event-B machines. *Sci. Comput. Program.*, 105:92–123, 2015. doi:10.1016/j.scico.2015.02.003.

7 Amit Bhatia, Lydia E. Kavraki, and Moshe Y. Vardi. Motion planning with hybrid dynamics and temporal goals. In *Conference on Decision and Control*, pages 1108–1115. IEEE, 2010. doi:10.1109/CDC.2010.5717440.

8 Errett Bishop. *Foundations of constructive analysis.* McGraw-Hill, 1967.

9 Brandon Bohrer and André Platzer. Toward structured proofs for dynamic logics. *CoRR*, abs/1908.05535, 2019. arXiv:1908.05535.

10 Brandon Bohrer and André Platzer. Constructive game logic. In Peter Müller, editor, *ESOP*, volume 12075 of *LNCS*. Springer, 2020.

11 Brandon Bohrer and André Platzer. Constructive hybrid games. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *IJCAR*, LNCS. Springer, 2020.

12 Brandon Bohrer and André Platzer. Refining constructive hybrid games. *CoRR*, abs/2002.02576, 2020. URL: https://arxiv.org/abs/2002.02576, arXiv:2002.02576.

**13** Brandon Bohrer, Yong Kiam Tan, Stefan Mitsch, Magnus O. Myreen, and André Platzer. VeriPhy: Verified controller executables from verified cyber-physical system models. In Dan Grossman, editor, *PLDI*, pages 617–630. ACM, 2018. `doi:10.1145/3192366.3192406`.

**14** Douglas S Bridges and Luminita Simona Vita. *Techniques of constructive analysis*. Springer, 2007.

**15** Sergio A. Celani. A fragment of intuitionistic dynamic logic. *Fundam. Inform.*, 46(3):187–197, 2001. URL: `http://content.iospress.com/articles/fundamenta-informaticae/fi46-3-01`.

**16** Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *LNCS*, pages 59–73. Springer, 2007. `doi:10.1007/978-3-540-74407-8_5`.

**17** Thierry Coquand and Gérard P. Huet. The calculus of constructions. *Inf. Comput.*, 76(2/3):95–120, 1988. `doi:10.1016/0890-5401(88)90005-3`.

**18** Thierry Coquand and Christine Paulin. Inductively defined types. In Per Martin-Löf and Grigori Mints, editors, *COLOG*, volume 417 of *LNCS*, pages 50–66. Springer, 1988. `doi:10.1007/3-540-52335-9_47`.

**19** Luís Cruz-Filipe, Herman Geuvers, and Freek Wiedijk. C-CoRN, the constructive Coq repository at Nijmegen. In Andrea Asperti, Grzegorz Bancerek, and Andrzej Trybulec, editors, *MKM*, volume 3119 of *LNCS*. Springer, 2004. Accessed: commits 9c44dae and 6411967. `doi:10.1007/978-3-540-27818-4_7`.

**20** JW Degen and JM Werner. Towards intuitionistic dynamic logic. *Log. and Log. Philosophy*, 15(4):305–324, 2006. `doi:10.12775/LLP.2006.018`.

**21** Peter Dybjer. Inductive families. *Formal Asp. Comput.*, 6(4):440–465, 1994. `doi:10.1007/BF01211308`.

**22** Sebastian Enqvist, Helle Hvid Hansen, Clemens Kupke, Johannes Marti, and Yde Venema. Completeness for game logic. In *LICS*, pages 1–13. IEEE, 2019. `doi:10.1109/LICS.2019.8785676`.

**23** Georgios E. Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009. `doi:10.1016/j.automatica.2008.08.008`.

**24** Ioannis Filippidis, Sumanth Dathathri, Scott C. Livingston, Necmiye Ozay, and Richard M. Murray. Control design for hybrid systems with TuLiP: The temporal logic planning toolbox. In *Conference on Control Applications*, pages 1030–1041. IEEE, 2016. `doi:10.1109/CCA.2016.7587949`.

**25** Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. LTLMoP: Experimenting with language, temporal logic and robot control. In *IROS*, pages 1988–1993. IEEE, 2010. `doi:10.1109/IROS.2010.5650371`.

**26** John Nathan Foster. Bidirectional programming languages. Technical Report MS-CIS-10-08, Department of Computer & Information Science, University of Pennsylvania, Philadelphia, PA, March 2010.

**27** Sujata Ghosh. Strategies made explicit in dynamic game logic. *Workshop on Logic and Intelligent Interaction at ESSLLI, Hamburg*, pages 74 –81, 2008.

**28** Valentin Goranko. The basic algebra of game equivalences. *Studia Logica*, 75(2):221–238, 2003. `doi:10.1023/A:1027311011342`.

**29** David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic logic*. MIT Press, 2000.

**30** George Edward Hughes and Max Cresswell. *A new introduction to modal logic*. Routledge, 1996.

**31** Norihiro Kamide. Strong normalization of program-indexed lambda calculus. *Bull. Sect. Log. Univ. Łódź*, 39(1-2):65–78, 2010.

**32** Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Automat. Contr.*, 53(1):287–297, 2008. `doi:10.1109/TAC.2007.914952`.

33 Dexter Kozen. Results on the propositional mu-calculus. *Theor. Comput. Sci.*, 27:333–354, 1983. doi:10.1016/0304-3975(82)90125-6.

34 Dexter Kozen. Kleene algebra with tests. *ACM Trans. Program. Lang. Syst.*, 19(3):427–443, 1997.

35 James Lipton. Constructive Kripke semantics and realizability. In Y. N. Moschovakis, editor, *Logic from Computer Science*, pages 319–357. Springer, 1992. doi:10.1017/978-1-4612-2822-6_13.

36 Sarah M. Loos and André Platzer. Differential refinement logic. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *LICS*, pages 505–514. ACM, 2016. doi:10.1145/2933575.2934555.

37 Evgeny Makarov and Bas Spitters. The Picard algorithm for ordinary differential equations in Coq. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *ITP*, volume 7998 of *LNCS*. Springer, 2013. doi:10.1007/978-3-642-39634-2_34.

38 Konstantinos Mamouras. Synthesis of strategies using the Hoare logic of angelic and demonic nondeterminism. *Log. Methods Comput. Sci.*, 12(3):1–41, 2016. doi:10.2168/LMCS-12(3:6)2016.

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

40 Aleksandar Nanevski, J. Gregory Morrisett, and Lars Birkedal. Hoare type theory, polymorphism and separation. *J. Funct. Program.*, 18(5-6):865–911, 2008. doi:10.1017/S0956796808006953.

41 Rohit Parikh. Propositional game logic. In *FOCS*, pages 195–200. IEEE, 1983. doi:10.1109/SFCS.1983.47.

42 André Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008. doi:10.1007/s10817-008-9103-8.

43 André Platzer. The structure of differential invariants and differential cut elimination. *Log. Methods Comput. Sci.*, 8(4):1–38, 2012. doi:10.2168/LMCS-8(4:16)2012.

44 André Platzer. Differential game logic. *ACM Trans. Comput. Log.*, 17(1):1:1–1:51, 2015. doi:10.1145/2817824.

45 André Platzer. Differential hybrid games. *ACM Trans. Comput. Log.*, 18(3):19:1–19:44, 2017. doi:10.1145/3091123.

46 André Platzer. *Logical Foundations of Cyber-Physical Systems*. Springer, Switzerland, 2018. doi:10.1007/978-3-319-63588-0.

47 André Platzer and Yong Kiam Tan. Differential equation invariance axiomatization. *J. ACM*, 67(1), 2020. doi:10.1145/3380825.

48 Ramaswamy Ramanujam and Sunil Easaw Simon. Dynamic logic on games with structured strategies. In Gerhard Brewka and Jérôme Lang, editors, *Knowledge Representation*, pages 49–58. AAAI Press, 2008. URL: http://www.aaai.org/Library/KR/2008/kr08-006.php.

49 Ankur Taly and Ashish Tiwari. Switching logic synthesis for reachability. In Luca P. Carloni and Stavros Tripakis, editors, *EMSOFT*, pages 19–28. ACM, 2010. doi:10.1145/1879021.1879025.

50 The Coq development team. The Coq proof assistant reference manual, 2019. URL: https://coq.inria.fr/.

51 Johan Van Benthem. Games in dynamic-epistemic logic. *Bull. Econ. Research*, 53(4):219–248, 2001.

52 Johan van Benthem. Logic of strategies: What and how? In Johan van Benthem, Sujata Ghosh, and Rineke Verbrugge, editors, *Models of Strategic Reasoning - Logics, Games, and Communities*, volume 8972 of *LNCS*, pages 321–332. Springer, 2015. doi:10.1007/978-3-662-48540-8_10.

53 Johan van Benthem and Eric Pacuit. Dynamic logics of evidence-based beliefs. *Studia Logica*, 99(1-3):61–92, 2011. doi:10.1007/s11225-011-9347-x.

**54** Johan van Benthem, Eric Pacuit, and Olivier Roy. Toward a theory of play: A logical perspective on games and interaction. *Games*, 2011. doi:10.3390/g2010052.

**55** Wiebe van der Hoek, Wojciech Jamroga, and Michael J. Wooldridge. A logic for strategic reasoning. In Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael J. Wooldridge, editors, *AAMAS*. ACM, 2005. doi:10.1145/1082473.1082497.

**56** Jaap van Oosten. Realizability: A historical essay. *Math. Structures Comput. Sci.*, 12(3):239–263, 2002. doi:10.1017/S0960129502003626.

**57** Klaus Weihrauch. *Computable Analysis - An Introduction*. Texts in Theoretical Computer Science. Springer, 2000. doi:10.1007/978-3-642-56999-9.

**58** Duminda Wijesekera. Constructive modal logics I. *Ann. Pure Appl. Log.*, 50(3):271–301, 1990. doi:10.1016/0168-0072(90)90059-B.

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