A Hybrid, Dynamic Logic for Hybrid-Dynamic Information Flow

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

André Platzer
Carnegie Mellon University
Pittsburgh, PA
aplatzer@cs.cmu.edu

Abstract

Information-flow security is important to the safety and privacy of cyber-physical systems (CPSs) across many domains: information leakage can both violate user privacy and reveal vulnerabilities to physical attacks. CPSs face the challenge that information can flow both in discrete cyber channels and in continuous real-valued physical channels ranging from time to motion to electrical currents. We call these *hybrid-dynamic information flows* (HDIFs) and introduce dHL, the first logic for verifying HDIFs in hybrid-dynamical models of CPSs. Our logic extends differential dynamic logic (dL) for hybrid-dynamical systems with hybrid-logical features for explicit program state representation, supporting relational reasoning used for information flow arguments. By verifying HDIFs, we ensure security even under a strong attacker model wherein an attacker can observe time and physical values continuously. We present a Hilbert-style proof calculus for dHL, prove it sound, and compare the expressive power of dHL with dL. We develop a hybrid system model based on the smart electrical grid FREEDM, with which we showcase dHL. We prove that the naïve model has a previously unknown information flow vulnerability, which we verify is resolved in a revised model. This is the first information flow proof both for HDIFs and for a hybrid-dynamical model in general.

CCS Concepts • **Theory of computation → Logic and verification**; *Modal and temporal logics*; • **Security and privacy → Logic and verification**; • **Computer systems organization → Embedded and cyber-physical systems**;

Keywords dynamic logic, hybrid logic, hybrid systems, information flow, cyber-physical systems, formal verification, smart grid

1 Introduction

Cyber-physical systems (CPSs), which feature discrete computer control interacting with a continuous physical environment, are ubiquitous. They include critical infrastructure such as electricity, natural gas, and petroleum transportation grids, medical devices such as pacemakers and insulin pumps, and transportation systems including aircraft, trains, and automobiles. Because of their criticality, it is essential to ensure their safe, correct operation, and formal methods for safety (e.g., collision-freedom) in CPS have had important successes [14, 18, 29].

There is less work on formal methods for CPS (information-flow) security, which is often as critical as physical safety. We focus on *nondeducibility* [4] of information flow, which captures the notion that an attacker cannot infer private information for certain in a system with non-observable nondeterminism. We choose this focus because for many nondeterministic systems, nondeducibility is the strongest property one can hope to achieve. As consumer-facing infrastructure such as electrical and telecommunication networks are increasingly computerized, the risk of leaking confidential customer information increases. Beyond leaking customer information, it has been suggested [3] that information flow leaks have the potential to aid attackers in identifying vulnerable infrastructure targets. Computerized medical devices risk leaking medical records protected by law (e.g., HIPAA), which can enable attacks that have life-threatening results, such as ventricular fibrillation [17]. Information leakage concerns are also significant in the transportation domain, e.g., position-spoofing attacks in aircraft have been proposed that could cause major disruptions to air-traffic control [33].

The common feature is that information flows in both computer communication channels and physical channels such as transmission lines, pipelines, the human body [31], and roadways. Because information flows through both computation and physics, CPSs demand a notion of flow which accounts for both. While information flow in CPS has been explored [1, 3, 16, 20, 36], prior works either model physics discretely or ignore physics altogether [3].

These abstractions constitute a significant *model gap*: Formal analyses of any model can only be trusted to the extent that the model is faithful to reality and to the abilities of attackers. Event-based models such as the prior model of the FREEDM grid [3], for example, assume attackers cannot observe time or exact physical quantities. Our hybrid (i.e., mixed continuous and discrete) dynamics narrow the gap greatly, modeling attackers that observe continuous time and real-valued physical quantities. In doing so, our FREEDM model reveals a leak undetected by the prior model.

A discrete-time model would also leave a smaller gap than and and reveal more bugs than an event model. A key advantage of a hybrid model is that we support continuous time and thus know for certain that we have accounted for the timing abilities of all possible attackers, while a discrete-time model would leave us uncertain whether the model is precise enough to reveal all practical attacks.

We investigate properties of *hybrid-dynamic information flows* (HDIFs), which combine discrete and continuous flows. We provide an approach for verifying HDIF security by introducing the logic dHL. The dHL logic features two forms of hybridness which should not be confused: it extends differential dynamic logic (dL) for reachability of hybrid-dynamical *systems* with first-order hybrid *logic* [7], which provides first-class representation of program states. This combination of hybrid dynamics with first-class program states enables us to verify HDIF security. In capturing physical and temporal phenomena, hybrid dynamics also provide a flexible

LICS '18, July 9–12, 2018, Oxford, United Kingdom

ACM ISBN 978-1-4503-5583-4/18/07…$15.00
https://doi.org/10.1145/3209108.3209151

framework for modeling side-channels and verifying them with the same techniques as other cyber-physical channels.

The distinguishing feature of information flow (vs. safety and liveness properties) is that it is not a trace property, but a 2-trace hyperproperty [12] (i.e., a property of pairs of program traces). This poses a hurdle for program verification calculi: Hoare calculi (and typical dynamic logics [34]), for example, cannot verify hyperproperties without significant source-level transformations such as self-composition [5]. These source transformations have been noted [35] to make verification tasks needlessly difficult in practice by inflating their size. Relational calculi [6] reduce verification complexity, but also generality.

Our novel use of hybrid logic not only reduces complexity but also maintains generality by allowing first-class representation of program states, which makes both direct statements and proofs of information-flow properties straightforward. In the process, we display a novel connection with hybrid logic that extends beyond information flow to general hyperproperties. Beyond its aesthetic appeal, this generality promises to enable verification of numerous related hyperproperties in one common logic, without having to adapt the logic to: i) more notions of security such as noninterference ii) more hyperproperties such as robustness [12]. We introduce a Hilbert-style proof calculus for dHL and prove it sound, then derive high-level rules for bisimulation. We relate the complexity of proofs in dHL vs. dL: a reduction is possible for a significant fragment of dHL, but impractical: i) it has limitations when applied to the general case, interfering with advanced proof techniques such as refinement [21] for modular verification, ii) the reduction causes quadratic formula size blowup in the worst case, and iii) the reduction is surprisingly subtle, suggesting that a proof by reduction to dL would be needlessly long and unintuitive.

As an example application, we give the first hybrid-dynamical model of a smart grid controller with concrete dynamics for distributed energy generation/storage and load-balancing [2] based on published descriptions of the FREEDM grid [19]. This contrasts with prior models [3], which consider only the high-level structure of event-based interactions between components. Our model shows the importance of dynamical-level modeling by revealing an information flow bug uncaught by higher-level models. We then prove a revised model secure even in the presence of HDIFs.

Our model of FREEDM captures its essential hybrid dynamics and our proof demonstrates important features of proofs in dHL: i) The well-understood principle of proof by bisimulation translates naturally to dHL proofs, ii) dHL provides an effective mechanism to tease apart the interactions between discrete transitions and continuous flow, enabling verification to scale to the complex interactions found in CPSs, and iii) typical CPSs have sufficiently complex information flows to warrant the deductive approach.

These traits are typical across different domains of CPS, showing that our approach holds promise for verifying information flow of applications in various domains beyond smart grids.

2 The Logic dHL

We present the complete syntax and semantics of the logic dHL, extending the dynamic logic dL with explicit hybrid-logical representation of program states. Our calculus, as with modern implementations [15] and machine-checked correctness proofs [9] for dL, is based on uniform substitution [11, §35][30]: symbols ranging over predicates, programs, etc. are explicitly represented in the syntax. This improves the ease with which dHL can be implemented and its soundness proof checked mechanically in future work.

The expressions of dHL consist of real-valued terms θ, world-valued terms w, programs α, and formulas ϕ. We write Θ for an arbitrary term θ or w, and write e when an expression can be either a term Θ or a formula ϕ, but not a program α.

Definition 1 (Real-valued terms of dHL)**.**

$$\theta ::= c \mid x \mid f(\vec{\theta}) \mid F \mid \theta + \theta \mid \theta \cdot \theta \mid @_i\theta$$

Here $c \in \mathbb{Q}$ is a literal and x is a real-valued *program variable*, said to be *flexible* because it can be bound in quantifiers. Their *rigid* counterparts are *nullary function symbols* $f(), g()$ that cannot be bound. The meaning of a function symbol $f(\vec{\theta})$ depends on an arbitrary number of real-valued arguments. Functionals F are a generalization of functions whose meaning depends on all flexible symbols. Functions and functionals are used to express axioms in Sec. 5. Terms in dHL extend dL with at-terms $@_w\theta$ denoting the value of term θ in the state denoted by the world-valued term w.

Definition 2 (World-valued terms of dHL)**.**

$$w ::= s \mid \bar{n}$$

The language of world-valued terms w is simple, consisting only of *world variables* s, t and *nominals* $\bar{n}, \bar{m}$, which differ only in that world variables are flexible while nominals are rigid.

Definition 3 (Programs of dHL)**.**

$$\alpha, \beta ::= ?(\phi) \mid x := \theta \mid x := * \mid x' = \theta \,\&\, \psi \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid a$$

The hybrid program constructs of dHL are simply those of dL. Hybrid programs combine discrete programming constructs with differential equations to provide a program representation of hybrid systems. The atomic dL programs are tests $?(\phi)$ that abort execution if formula ϕ is false, assignments $x := \theta$ and $x := *$ which update program variable x to the value of term θ or a nondeterministic value, differential equation evolution $x' = \theta \,\&\, \psi$, and object-level program constants a which range over fixed, arbitrary programs. They should not be confused with the similar-looking program metavariables α used in schemata and theorems. Differential equations are the defining feature of dL; the effect of $x' = \theta \,\&\, \psi$ is to evolve the differential equation $x' = \theta$ nondeterministically for any duration, but only so long as the formula ψ is always true.

They are composed with nondeterministic choice $\alpha \cup \beta$ that runs exactly one of α or β, sequential composition $\alpha; \beta$, and nondeterministic iteration α^* that runs α any finite number of times sequentially. Traditional deterministic programming constructs can be derived from the nondeterministic hybrid program constructs, e.g., `if` $(\phi)\{\alpha\}$ `else` $\{\beta\} \equiv (?(\phi); \alpha) \cup (?(\neg\phi); \beta)$.

Definition 4 (Formulas of dHL)**.**

$$\phi, \psi ::= \phi \wedge \psi \mid \neg\phi \mid \exists x : \mathbb{R}\ \phi \mid \theta_1 \geq \theta_2 \mid \langle\alpha\rangle\phi$$
$$\mid \exists s : \mathcal{W}\ \phi \mid w \mid @_w\phi \mid \downarrow s\ \phi \mid p(\vec{\Theta}) \mid P$$

Formulas $\phi \wedge \psi, \neg\phi, \exists x : \mathbb{R}\ \phi$, and $\theta_1 \geq \theta_2$ are as in first-order logic. As in dL, the diamond modality $\langle\alpha\rangle\phi$ says there exists an execution of the (nondeterministic) program α where formula ϕ holds in the ending state. Its dual, the box modality $[\alpha]\phi$, says *all* end states satisfy ϕ, and is derived: $[\alpha]\phi \equiv \neg\langle\alpha\rangle\neg\phi$.

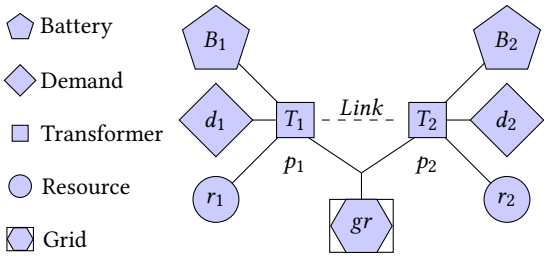

Figure 1. FREEDM load balancing

The following features are added from first-order hybrid logic. The quantifier $\exists s:\mathcal{W}\ \phi$ says there exists a world (program state) s in which ϕ holds (where ϕ can mention the world variable s). We will also use the universal quantifier $\forall s:\mathcal{W}\ \phi$, which is a derived construct by the duality $\forall s:\mathcal{W}\ \phi \equiv \neg\exists s:\mathcal{W}\ \neg\phi$. We support nominal *propositions* w that hold in exactly the one state denoted by the world-valued term w. These allow testing equality of the current state against w. Note the same syntax is used regardless whether w appears as a term or formula; these usages are distinguished by syntactic context. The hybrid *satisfaction modality* $@_w\phi$ says that ϕ is true at the unique state named by w. In addition to the typical existential and universal quantifiers, hybrid logic features the *local* quantifier $\downarrow s\ \phi$ which binds the *current* state to the world variable s within the formula ϕ, whereas the universal quantifier $\forall s:\mathcal{W}\ \phi$ binds an *arbitrary* state to s. The local quantifier $\downarrow s\ \phi$ can be derived as $\downarrow s\ \phi \leftrightarrow \exists s:\mathcal{W}\ (s\wedge\phi)$ or equivalently $\forall s:\mathcal{W}\ (s\rightarrow\phi)$. We present this quantifier in its entirety regardless, because it is important to information-flow applications and may be unfamiliar to the reader. The connectives $\downarrow s\ \phi$ and $@_w\phi$ can be understood computationally as well, as storing or loading the current state to s or from w, respectively.

The *predicate symbols* $p(\vec{\Theta})$ range over both real-valued terms θ and nominal expressions w, and are used in axioms to stand for propositions. Beyond axioms, predicates will be used widely in bisimulation arguments for information-flow: $R(i,j)$ denotes a binary predicate over nominals. Predicationals P simply stand for arbitrary formulas and are used in axioms in Sec. 5.

3 Information-Flow Example: FREEDM Grid

In this section, we introduce two variants of a smart grid model based on NSF FREEDM [19], a *microgrid* which controls a local section of the power grid and interacts with the surrounding *macrogrid*. Our model is the first hybrid-systems model of FREEDM and follows the published algorithm [2], incorporating detailed dynamics not present in prior models [3]. We show how information-flow security properties and their negations are stated in dHL. We prove them in Secs. 8 and 9 once the proof calculus is introduced.

Smart grids like FREEDM use computer control to make electrical grids more robust, efficient, and cost-effective in face of increasingly diverse power loads and supplies. Computer control in grids makes joint cyber-physical security of this critical infrastructure essential. Not only can information flow violations compromise private consumer information, but it has been suggested they can aid attackers [3] in identifying targets for physical attacks.

Scenario. We look at an exchange (depicted in Fig. 1) that migrates power between two neighboring transformers T_1 and T_2 connected to a macrogrid gr over a shared line. Variable names indicate units: energy is uppercase, power (derivative of energy, e.g. $B_i' = b_i$) is lowercase, and migration rates (derivative of power, e.g. $b_i' = bm_i$) end in m. Each transformer T_i carries power p_i and is connected to a renewable energy resource r_i, to a household which demands power d_i, and to a energy storage device B_i. The transformers are connected by a communication *Link*. While real instances of the FREEDM grid have many transformers, each migration involves exactly two transformers, so the two-transformer case provides important insight for the general case.

Each transformer can be in one of three demand states: *Low Demand*, *Normal Demand*, or *High Demand*. The algorithm [2] states:

- Net demand n_i is the difference of gross power demand d_i and the sum of power draw p_i with generation r_i.
- A transformer is in *Low Demand* if it has net demand $n_i < 0$, *High Demand* if net demand exceeds a provided threshold $n_i \geq thresh > 0$ or *Normal* otherwise.
- If any transformer i is *Low* (has excess power) while the other (written $\bar{i}$) is *High*, power migrates at a provided constant rate $m := maxm$ until at least one of them is *Normal*.
- Any excess power supply $-n_i > 0$ not used in migration is accumulated as energy $0 \leq B_i \leq B_{\max}$ in battery i.
- Any excess demand $n_i > 0$ not met by migration is drawn from the battery B_i with power b_i and migration rate bm_i, or sold to the grid if the battery is full ($B_i = B_{\max}$).
- If T_i's corresponding battery B_i is empty, it draws power gr (with migration rate $gr' = grm$) from the macrogrid instead.

The grid is modeled in Fig. 2 as a hybrid program α_F, which contains the controller (ctrl) and physical model (plant). The controller migrates power (migrate) and operates the battery (bat), which has two implementations: a deterministic implementation bat_I of the above algorithm, which we show to be *insecure*, and a nondeterministic version bat_S, which we show to be *secure*. We abbreviate $\alpha_I \equiv \alpha_{F}^{\text{bat}_I}$ and $\alpha_S \equiv \alpha_{F}^{\text{bat}_S}$ for the instantiation of α_F with the insecure or secure battery, respectively.

Our treatment of d_i and r_i is general, assuming only that they are non-negative and can change countably often. Time $t' = 1$ is not used for control, but will factor into our proofs because it is observed by attackers and we prove that, e.g., observing the ODE duration does not leak the continuous variables p_i, B_i, b_i.

Defining Information Flow. We give the general formulation:

Definition 5 (Nondeducibility Information-Flow Security)**.** Let α be a program and let L be the set of publicly observable expressions, e.g., $L = \{gr, t\}$ for FREEDM with publicly-known time t and macrogrid flow gr. At its simplest, binary relation $R(i,j)$ says worlds i and j agree on all public expressions L, and is defined by:

$$R(i,j) \ \equiv\ \left(\bigwedge_{\theta\in L}\left(@_i\theta = @_j\theta\right)\right)\wedge\left(\bigwedge_{\phi\in L}\left(@_i\phi \leftrightarrow @_j\phi\right)\right)$$

In practice, $R(i,j)$ often also states that i and j satisfy any problem-specific constraints, e.g. on the range of system parameters. Now we define nondeducibility information flow:

$$\forall i_1, i_2, o_1 : \mathcal{W}\ \left(@_{i_1}\langle\alpha\rangle o_1 \wedge R(i_1,i_2) \rightarrow\ @_{i_2}\langle\alpha\rangle\downarrow o_2\ R(o_1,o_2)\right)$$

Nondeducibility means an observer can never deduce anything about the input value $@_{i_1}x$ of a private variable $x \notin L$ from the final values of public expressions $@_{o_1}(\theta$ or $\phi)$, when the observer

$$\alpha_F \equiv (\text{ctrl};\text{plant})^* \qquad \text{ctrl} \equiv \text{migrate};\text{bat}$$

$$\begin{aligned}\text{migrate} \equiv \Big\{\ & d_i := *;?(d_i \geq 0);\quad r_i := *;?(r_i \geq 0);\quad n_i := d_i - (r_i + p_i);\\ & \texttt{if } (n_i \geq \mathit{thresh} \wedge n_{\bar{i}} < 0) \quad \{\ m := (-1)^i \cdot \mathit{maxm}\}\\ & \texttt{else} \qquad\qquad\qquad\quad \{\ m := 0\}\Big\}\end{aligned}$$

$$\text{plant} \equiv \{p_i' = -1^i \cdot m, B_i' = b_i, b_i' = bm_i, gr' = grm, t' = 1 \,\&\, B_i \geq 0\}$$

$$\begin{aligned}\text{bat}_I \equiv \Big\{\ & gr := 0;\ bm_i := 0;\ grm := 0;\\ & \texttt{if } ((\mathbf{n_i} \leq \mathbf{0} \wedge B_i < B_{\max}) \vee (n_i > 0 \wedge B_i > 0))\{\\ & \quad \{\ b_i := -n_i;\ bm_i := bm_i + (-1^{i+1}) \cdot m\}\\ & \texttt{else } \{\ b_i := 0;\ gr := gr + n_i;\ grm := grm + (-1)^{i+1} \cdot m\}\Big\}\end{aligned}$$

$$\begin{aligned}\text{bat}_S \equiv \Big\{\ & gr := 0;\ bm_i := 0;\ grm := 0;\\ & \big(?(B_i < B_{\max}) \vee (n_i > 0 \wedge B_i > 0);\\ & \quad b_i := -n_i;\ bm_i := bm_i + (-1)^{i+1} \cdot m\big)\\ & \cup \big(b_i := 0;\ gr := gr + n_i;\ grm := grm + (-1)^{i+1} \cdot m\big)\ \Big\}\end{aligned}$$

Figure 2. FREEDM model with insecure and secure batteries

does not directly observe how nondeterminism is resolved. This is the case when Def. 5 holds because it says *every* input state i_2 that agrees on public terms ($R(i_1,i_2)$) has at least one program path $@_{i_2}\langle\alpha\rangle o_2$ (where o_2 is the final state of α, as bound by the quantifier $\downarrow o_2$) that would explain the final values of public expressions at o_1. Because all inputs would have made the output possible, it is impossible to deduce anything about the input state. As we will see in Sec. 9, the core challenge in proving this property is identifying *which* program path explains any given final public value: for a nondeterministic program α, only by carefully resolving nondeterminism will we find a path that ensures secure flow.

In Sec. 8 we will also prove bat_I is nondeducibility *insecure*, by proving the negation of nondeducibility security, i.e.:

$$\exists i_1,i_2,o_1 : \mathcal{W}\Big(@_{i_1}\langle\alpha\rangle o_1 \wedge R(i_1,i_2) \wedge @_{i_2}[\alpha_I]\downarrow o_2\ \neg R(o_1,o_2)\Big)$$

4 Semantics

We return to developing dHL. Its semantic development begins with our semantic building blocks. The building blocks are *worlds* ω,μ,ν, *galaxies* g,h, and *interpretations* I,J. Interpretations give meaning to *rigid* symbols, whose meanings are fixed throughout a formula, such as functions, predicates, nominal constants, and program constants. The *flexible symbols*, whose meaning can change throughout a program or formula, are real-valued *program variables* x,y and world-valued *world variables* s,t. The program variables receive their values from the active world ω while world variables receive their values from the galaxy g, which contains an infinitude of worlds, one for each world variable. We write ω_x^r for the state that updates ω's value of program variable x to $r \in \mathbb{R}$ and likewise g_s^ω for updated galaxies.

Program and world variables are both drawn from a countable variable set $\mathcal{V}$ isomorphic to $\mathbb{N}$. The set of all worlds $\mathcal{W}$ is isomorphic to $\mathbb{R}^{\mathcal{V}}$. The set of all galaxies $\mathcal{G}$ is isomorphic to $\left(\mathbb{R}^{\mathcal{V}}\right)^{\mathcal{V}}$.

We write $I(f), I(p)$, etc. for the interpretation of a given symbol and $\mathcal{I}$ for the set of all interpretations. The types of each component are as follows, where $\wp(S)$ is the power set of S:

$$\begin{aligned} I(f)&\colon (\mathbb{R} \cup \mathcal{W}) \to \mathbb{R}\\ I(p)&\colon \wp(\mathbb{R} \cup \mathcal{W})\\ I(F)&\colon \mathcal{G} \times \mathcal{W} \to \mathbb{R}\\ I(P)&\colon \mathcal{G} \to \wp(\mathcal{W})\\ I(a)&\colon \mathcal{G} \to \wp(\mathcal{W} \times \mathcal{W})\end{aligned}$$

That is, we treat f and p as untyped and unary, because generalizations to polyadic f,p are straightforward.

Definition 6 (Real term semantics)**.** Let $\omega\in\mathcal{W}, g\in\mathcal{G}, I\in\mathcal{I}$, then:

$$[\![x]\!]Ig\omega = \omega(x) \tag{1}$$
$$[\![c]\!]Ig\omega = c \tag{2}$$
$$[\![\theta_1 + \theta_2]\!]Ig\omega = [\![\theta_1]\!]Ig\omega + [\![\theta_2]\!]Ig\omega \tag{3}$$
$$[\![\theta_1 \cdot \theta_2]\!]Ig\omega = [\![\theta_1]\!]Ig\omega \cdot [\![\theta_2]\!]Ig\omega \tag{4}$$
$$[\![@_w\theta]\!]Ig\omega = [\![\theta]\!]Ig\nu \text{ where } \nu = [\![w]\!]Ig\omega \tag{5}$$
$$[\![f(\theta)]\!]Ig\omega = I(f)([\![\theta]\!]Ig\omega) \tag{6}$$
$$[\![F]\!]Ig\omega = I(F)(g\omega) \tag{7}$$

Equations (1)–(4) describe polynomial terms as in dL. The new term construct of dHL is the at-term $@_w\theta$, whose meaning is the meaning of real term θ at the state denoted by world term w.

The language of world terms is quite simple, containing only rigid nominals $\bar{n}$ and flexible world variables s, deriving their meaning from the interpretation I or galaxy g, respectively. We write $[\![w]\!]Ig\omega$ for symmetry with real terms θ even though ω is unused.

Definition 7 (World term semantics)**.** Let $g \in \mathcal{G}, I \in \mathcal{I}$, then:

$$[\![\bar{n}]\!]Ig\omega = I(n)$$
$$[\![s]\!]Ig\omega = g(s)$$

Definition 8 (Program semantics)**.** As in dL with the addition of galaxies g. Let $\omega,\nu \in \mathcal{W}, g \in \mathcal{W}, I \in \mathcal{I}$, then:

$$(\omega,\omega) \in [\![?\phi]\!]Ig \text{ iff } \omega \in [\![\phi]\!]Ig \tag{8}$$
$$(\omega,\nu) \in [\![x := \theta]\!]Ig \text{ iff } \nu = \omega_x^r \text{ for } r = [\![\theta]\!]Ig\omega \tag{9}$$
$$(\omega,\nu) \in [\![x := *]\!]Ig \text{ iff } \nu = \omega_x^r \text{ for some } r \in \mathbb{R} \tag{10}$$
$$\begin{aligned}(\omega,\nu) \in [\![x' = \theta \,\&\, \psi]\!]Ig \text{ iff } &(\omega,\nu) = (\varphi(0),\varphi(t)) \text{ and } \varphi \text{ solves}\\ &x' = \theta \text{ on } [0,t] \text{ and } \varphi(s) \in [\![\psi]\!]Ig \text{ for all } s \in [0,t]\end{aligned} \tag{11}$$
$$(\omega,\nu) \in [\![\alpha \cup \beta]\!]Ig \text{ iff } (\omega,\nu) \in [\![\alpha]\!]Ig \text{ or } (\omega,\nu) \in [\![\beta]\!]Ig \tag{12}$$
$$(\omega,\nu) \in [\![\alpha;\beta]\!]Ig \text{ iff } (\omega,\nu) \in ([\![\alpha]\!]Ig) \circ ([\![\beta]\!]Ig) \tag{13}$$
$$(\omega,\nu) \in [\![\alpha^*]\!]Ig \text{ iff } (\omega,\nu) \in ([\![\alpha]\!]Ig)^* \tag{14}$$
$$(\omega,\nu) \in [\![a]\!]Ig \text{ iff } (\omega,\nu) \in I(a)(g) \tag{15}$$

Galaxy g is untouched by execution. Equations (8)–(11) are the atomic hybrid programs. Differential equations (11) evolve according to the solution of the ODE for any duration $t \geq 0$, but must stop while the formula ψ still holds. In (13), $\circ$ is composition. In (14), $([\![\alpha]\!]Ig)^*$ is the reflexive, transitive closure of $[\![\alpha]\!]Ig$. Program constants (15) receive their meaning from the interpretation (and galaxy, since formulas inside programs can mention nominals).

Definition 9 (Formula semantics)**.**

$$\omega \in [\![\phi \wedge \psi]\!] Ig \text{ iff } \omega \in [\![\phi]\!] Ig \text{ and } \omega \in [\![\psi]\!] Ig \tag{16}$$

$$\omega \in [\![\neg\phi]\!] Ig \text{ iff } \omega \notin [\![\phi]\!] Ig \tag{17}$$

$$\omega \in [\![\exists x : \mathbb{R}\ \phi]\!] Ig \text{ iff } \omega^r_x \in [\![\phi]\!] Ig \text{ for some } r \in \mathbb{R} \tag{18}$$

$$\omega \in [\![\theta_1 \geq \theta_2]\!] Ig \text{ iff } [\![\theta_1]\!] Ig\omega \ \geq\ [\![\theta_2]\!] Ig\omega \tag{19}$$

$$\omega \in [\![\langle\alpha\rangle\phi]\!] Ig \text{ iff } \nu \in [\![\phi]\!] Ig \text{ for some } \nu \text{ s.t. } (\omega, \nu) \in [\![\alpha]\!] Ig \tag{20}$$

$$\omega \in [\![\exists s : \mathcal{W}\ \phi]\!] Ig \text{ iff } \omega \in [\![\phi]\!] Ig^\nu_s \text{ for some } \nu \in \mathcal{W} \tag{21}$$

$$\omega \in [\![@_w\phi]\!] Ig \text{ iff } \nu \in [\![\phi]\!] Ig \text{ for } \nu = [\![w]\!] Ig \tag{22}$$

$$\omega \in [\![\downarrow s\ \phi]\!] Ig \text{ iff } \omega \in [\![\phi]\!] Ig^\omega_s \tag{23}$$

$$\omega \in [\![w]\!] Ig \text{ iff } [\![w]\!] Ig\omega = \omega \tag{24}$$

$$\omega \in [\![p(\Theta)]\!] Ig \text{ iff } [\![\Theta]\!] Ig\omega \in I(p) \tag{25}$$

$$\omega \in [\![P]\!] Ig \text{ iff } \omega \in I(P)(g) \tag{26}$$

Equations (16)–(19) are a standard definition of first-order logic connectives, wherein $[\![\theta_i]\!] Ig\omega \colon \mathbb{R}$ is the denotation of real-valued term θ_i (Def. 6). Equation (20) defines the diamond modality $\langle\alpha\rangle\phi$: we employ a Kripke semantics where possible worlds are program states (Def. 8). Equations (21)–(24) define the hybrid-logical operators, where $[\![w]\!] Ig \colon \mathcal{W}$ (Def. 7) is the denotation of a world term. Equations (25)–(26) say predicates and predicationals derive their meaning from the interpretation I, with the difference that predicates depend only on their arguments while predicationals depend on all flexible symbols. We say formula ϕ is *valid* when the relation $\omega \in [\![\phi]\!] Ig$ holds for all states ω, galaxies g, and interpretations I.

5 Proof Calculus

We present a sound proof calculus for dHL, which is used for deductive verification. The calculus is given in Hilbert style, i.e., axioms are used wherever possible, with a minimal number of proof rules. Axioms are instantiated with a uniform substitution [30][11, §35] rule: from the validity of ϕ we can conclude validity of $\sigma(\phi)$ where substitution σ specifies concrete replacements for some or all rigid symbols in a formula ϕ:

$$\text{US}\ \frac{\phi}{\sigma(\phi)}$$

The side-conditions determining which substitutions σ are sound are non-trivial, and make up much of the soundness proof in Sec. 6, with the benefit that soundness arguments and implementation for individual axioms are greatly simplified.

Program Axioms. The axioms for programs in diamond modalities in Fig. 3 are as in dL [27]. Axioms for box modalities $[\alpha]\phi$ can be derived by duality (axiom $\langle\cdot\rangle$, Fig. 3). With the exception of the loop axioms, these axioms can be read off directly from the semantics of hybrid programs. The axiom $\langle'\rangle$ replaces a differential equation with a global solution, represented here by the expression[1] $y(t)$. Loops can be finitely unfolded with the axiom $\langle*\rangle$. More often, we reason by induction using axiom I or its derived rules.

Modal Axioms and Hilbert Rules. Generic modal axioms and Hilbert rules are as in dL [27] and are listed in Fig. 3. The axiom $\langle\cdot\rangle$ relates the diamond and box modalities, and is used to derive axioms for box modalities $[\alpha]\phi$. As is typical for Hilbert calculus, axioms are combined with rules G, US, and MP. Axiom V says nullary predicates $p()$ are preserved under program execution because they depend on no program variables.

$\langle;\rangle$ $\langle a;b\rangle P \leftrightarrow \langle a\rangle\langle b\rangle P$

$\langle\cup\rangle$ $\langle a \cup b\rangle P \leftrightarrow (\langle a\rangle P \vee \langle b\rangle P)$

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

$\langle:=\rangle$ $\langle x := f()\rangle p(x) \leftrightarrow p(f())$

$\langle:*\rangle$ $\langle x := *\rangle p(x) \leftrightarrow \exists x : \mathbb{R}\ p(x)$

$\langle'\rangle$ $\langle x' = F\ \&\ q(x)\rangle p(x) \leftrightarrow \exists t{\geq}0(p(y(t)) \wedge \forall 0{\leq}s{\leq}t\ q(y(s)))$

$\langle*\rangle$ $\langle a^*\rangle P \leftrightarrow (P \vee \langle a\rangle\langle a^*\rangle P)$

I $[a^*]P \leftrightarrow (P \wedge [a^*](P \to [a]P))$

G $\dfrac{P}{[a]P}$ M $\dfrac{P \to Q}{\langle a\rangle P \to \langle a\rangle Q}$ $\langle\cdot\rangle$ $\langle a\rangle P \leftrightarrow \neg[a]\neg P$

US $\dfrac{\phi}{\sigma(\phi)}$ MP $\dfrac{P \to Q \quad P}{Q}$ V $p() \to [a]p()$

K $[a](P \to Q) \to [a]P \to [a]Q$

Figure 3. dL axioms and rules

Hybrid rules and axioms. Our hybrid modality and quantifier axioms come from first-order hybrid logic [7] and Combinatory Dynamic Logic [24] and are listed in Fig. 4. Axiom @id says nominal constant formulas $\bar{n}$ are satisfied at the state named by $\bar{n}$. Axiom $\exists W$ introduces a name for the current state. Rule $G_@$ and axiom $K_@$ are the Gödel generalization rule and Kripke axiom for the @ modality. Axioms $\forall I_@$ and $\forall E_@$ are Skolemization and elimination for universal world quantifiers. Axiom @I introduces an $@_{\bar{n}}\phi$ modality when $\bar{n}$ is the current state. Axiom @@ collapses nested @ modalities to the inner modality. Axiom @↔ replaces equal states in context. Axiom $\langle\bar{n}\rangle$ introduces an @ modality for any state $\bar{n}$ reachable by a program a. Axiom ↓ reduces the local quantifier to its definition in terms of the existential operator. Homomorphism axiom family @hom completely captures the meaning of at-terms. Barcan axiom schema BW swaps a quantifier $\exists s : \mathcal{W}$ with a program α where s does not appear *free* (s is never *bound* in *any* program). To make schematic program α into a constant a and make BW a concrete axiom, it would suffice to prohibit nominals inside programs.

6 Theory

We now develop the theory of dHL, showing that the proof calculus is sound and comparing the expressiveness of dHL with other logics. Complete definitions and proofs are in the companion report [8].

6.1 Soundness

We show soundness of dHL by extending the soundness proof for the uniform substitution calculus for dL [30]. Uniform substitution allows for a modular soundness proof: the soundness proof is separated into proving that a finite list of dHL axioms are valid and that uniform substitution and the remaining Hilbert rules preserve validity. We prove that all valid dL formulas are valid dHL formulas, and thus dL axioms are also sound in dHL automatically.

[1] The presentation of this axiom is simplified for clarity. In reality, differential equation solving is implemented with the combination of several axioms [30].

$$\text{@id}\quad @_{\bar{n}}\bar{n}$$

$$\exists W\quad \exists s:\mathcal{W}\ s$$

$$\mathrm{G}_@\quad \frac{P}{@_{\bar{n}}P}$$

$$\forall I_@\quad \frac{q(t)}{\forall s:\mathcal{W}\ q(s)} \qquad \text{(t fresh)}$$

$$@\mathrm{I}\quad (\bar{n}\wedge P)\to @_{\bar{n}}P$$

$$@@\quad @_{\bar{n}}@_{\bar{m}}P\leftrightarrow @_{\bar{m}}P$$

$$@\leftrightarrow\quad @_{\bar{n}}\bar{m}\to(p(\bar{n})\leftrightarrow p(\bar{m}))$$

$$\forall E_@\quad (\forall s:\mathcal{W}\ p(s))\to p(\bar{n})$$

$$\langle\bar{n}\rangle\quad ([a]\downarrow s\ p(s)\wedge\langle a\rangle\bar{n})\to p(\bar{n})$$

$$\downarrow\quad \downarrow s\ p(s)\leftrightarrow\exists s:\mathcal{W}\ \ (s\wedge p(s))$$

$$\mathrm{K}_@\quad @_{\bar{n}}(P\to Q)\to(@_{\bar{n}}P\to @_{\bar{n}}Q)$$

$$@\text{hom}\quad @_{\bar{n}}p(F_1,\dots F_m)\leftrightarrow p(@_{\bar{n}}F_1,\dots,@_{\bar{n}}F_m)$$

$$\mathrm{BW}\quad \langle\alpha\rangle\exists s:\mathcal{W}\ P\leftrightarrow\exists s:\mathcal{W}\ \langle\alpha\rangle P\quad (s\notin \mathrm{FV}(\alpha))$$

Figure 4. Hybrid rules and axioms

Case	Replacement	Admissible when
$\sigma(@_w\theta)$	$= @_{\sigma(w)}\sigma(\theta)$	No new $x\in\mathcal{V}$ in $\mathrm{FV}(\sigma(\theta))$
$\sigma(@_w\phi)$	$= @_{\sigma(w)}\sigma(\phi)$	No new $x\in\mathcal{V}$ in $\mathrm{FV}(\sigma(\phi))$
$\sigma(\forall s:\mathcal{W}\ \phi)$	$= \forall s:\mathcal{W}\ \sigma(\phi)$	No new s in $\mathrm{FV}(\sigma(\phi))$
$\sigma(\exists s:\mathcal{W}\ \phi)$	$= \exists s:\mathcal{W}\ \sigma(\phi)$	No new s in $\mathrm{FV}(\sigma(\phi))$
$\sigma(\downarrow s\ \phi)$	$= \downarrow s\ \sigma(\phi)$	No new s in $\mathrm{FV}(\sigma(\phi))$
$\sigma(\bar{n})$	$= \bar{n}$, if $\bar{n}\notin\sigma$	
$\sigma(\bar{n})$	$= \sigma\bar{n}$, if $\bar{n}\in\sigma$	
$\sigma(s)$	$= s$	

Figure 5. Uniform substitution algorithm (new cases)

Substitution. The US rule in dHL is analogous to that in dL:

$$\mathrm{US}\quad \frac{\phi}{\sigma(\phi)}$$

In dL, the US rule is sound when the substitution σ does not introduce free references to bound variables. Such substitutions are called *admissible*, a condition which can be checked syntactically.

We generalize: in dHL, admissible substitutions do not introduce free references to any bound *flexible symbol*, world variables included. Admissibility conditions are checked recursively by the substitution algorithm (Fig. 5). We give the new cases and their admissibility conditions here. The *free-variable* function $\mathrm{FV}(e)$ recursively computes which flexible symbols might influence e.

Examples. The admissibility conditions in Fig. 5 are instantiations of the principle that substitution should not introduce new free references under a binder. Yet, it can be surprisingly subtle both why these conditions are necessary for soundness and why the resulting calculus is sufficiently complete. To see why they are necessary for soundness, consider for example the instance of axiom @hom for $\sigma_C \overset{\text{def}}{\equiv} \{F_C\mapsto x,\ p(\cdot)\mapsto x=\cdot\}$, a substitution which *clashes* and which, if it were permitted, would result in an invalid formula:

$$@_{\bar{n}}(x=x)\leftrightarrow x=@_{\bar{n}}x$$

where the LHS is a tautology and the RHS is false almost everywhere. The fundamental problem is that the modality $@_{\bar{n}}$ modifies the meaning of x: on the left it refers to $@_{\bar{n}}x$ while on the right it refers to x in the present state. In short, the meaning of σp is *non-uniform*, so substitution with σ would be unsound. The admissibility check for $@_w$ prohibits such substitutions, ensuring uniformity and thus soundness of rule US.

It is equally subtle why rule US allows axiom @hom to be instantiated at all, because the $@_{\bar{n}}$ modality binds the entire state. The key is that the argument $\cdot$ of a predicate does not contribute to the free variables of a substitution. For example, the substitution $\sigma_A \overset{\text{def}}{\equiv} \{F_1\mapsto x^2,\ p(\cdot)\mapsto\cdot\geq 0\}$ is admissible because i) F_1 already depends on all variables, so $\sigma_A(F_1)$ introduces no free variables and ii) the argument $\cdot$ is rigid, so $\sigma_A(p)$ introduces no free variables. Intuitively, σ_A should be admissible for formula $p(F_1)$ because the *argument* F_1 is a functional that can depend on all variables, yet σ_C clashes since $\sigma_C(p)$ does not defer to the argument (written $\cdot$), causing @hom to incorrectly use simply x on both sides.

We proceed to the soundness theorem, following the same structure as previous work [30]. We begin with lemmas on the correctness of free variable and signature computations, where the signature $\Sigma(e)$ is the analog of $\mathrm{FV}(e)$ for *rigid* symbols. The coincidence lemmas say that expressions depend only on their signature and free variables. We extend coincidence for terms and formulas with new cases for hybrid-logical constructs:

Lemma 1 (Coincidence)**.**

1. *If $\omega=\tilde{\omega}$ on $FV(\theta)$, $g=h$ on $FV(\theta)$, and $I=J$ on $\Sigma(\theta)$, then $[\![\theta]\!]Ig\omega=[\![\theta]\!]Jh\tilde{\omega}$.*
2. *If $\omega=\tilde{\omega}$ on $FV(\phi)$, $g=h$ on $FV(\phi)$, and $I=J$ on $\Sigma(\phi)$, then $\omega\in[\![\phi]\!]Ig$ iff $\tilde{\omega}\in[\![\phi]\!]Jh$.*
3. *If $\omega=\tilde{\omega}, g=h$ on $V\supseteq FV(\alpha), I=J$ on $\Sigma(\alpha)$, and $(\omega,\nu)\in[\![\alpha]\!]Ig$, then exists $\tilde{\nu}$ s.t. $(\tilde{\omega},\tilde{\nu})\in[\![\alpha]\!]Ih$ and $\nu=\tilde{\nu}$ on V.*

Axioms of dL need not be reproved because dHL contains dL:

Proposition 2 (dHL contains dL)**.** *If ϕ is a* dL *formula, then validity in* dL *semantics and validity in* dHL *semantics coincide for ϕ.*

Theorem 3 (dHL soundness)**.** *All* dHL *rules are sound and all axioms valid, thus all provable* dHL *formulas are valid.*

Proof Sketch [8]. Soundness of US is proven inductively, appealing to Lem. 1. The dL axioms are valid in dL [30] and (by Prop. 2) dHL, and by US so are their instances, even instances containing hybrid connectives. Validity of the new dHL axioms is by direct proof. □

6.2 Reducibility

We compare the expressive power of dHL to that of dL in order to determine when and in what sense dHL is necessary or especially beneficial compared to dL. The comparison is surprisingly subtle, and finds that while dHL is reducible to dL, its specialized hybrid-logical rules make direct proof in dHL preferable for practical purposes. The core idea is to emulate each world variable from

dHL with a finite number of program variables in dL, resulting in an equivalent, finite dL formula. This approach is subtle primarily because dHL states contain infinitely many program variables:

1. The size of worlds is not a cosmetic decision, but rather affects validity of some formulas. Consider the formula:
$$\phi \equiv \langle x := * \rangle s$$
This is valid iff $\mathcal{V} \equiv \{x\}$, i.e. iff reaching all values of x suffices to reach all states. We consider this behavior too surprising, and eliminate it by requiring infinite states. The formula is then invalid because program $x := *$ cannot, e.g., transition between any ω and ν where $\omega(r) \neq \nu(r)$ for $r \neq x$.
2. Program constants and predicationals depend on every variable, in order to capture the notion that programs and formulas can use arbitrary variable names. Since there are infinitely many variables, they have infinitely many dependencies.

We show that for formulas without program constants and predicationals (called *concrete* formulas), infinite worlds are not an obstacle, while for formulas with program constants or predicationals (called *abstract* formulas), they are. Concrete formulas are reducible: even though each state contains infinitely many variables, it suffices to employ a single *fresh* variable r (consider example above). To make this claim formal, we introduce a notion of finite domains for states, galaxies and interpretations.

Definition 10 (Finite-domain states)**.** State ω has *finite domain* $S \subseteq \mathcal{V}$ if S is finite and $\omega(x) = 0$ for all $x \notin S$. Galaxies and interpretations are analogous. A formula ϕ is *finite-domain valid* for domain S if $\omega \in [\![\phi]\!]Ig$ for all ω, I, g with finite domain S.

We outline the proof here, with full proofs in the report [8]. The proof proceeds by showing that in both dHL and dL, validity and finite-domain agree for concrete formulas, then showing that our reduction preserves finite-domain validity. Thus our reduction preserves validity for concrete formulas. The converse also holds because dL is a fragment of dHL (Prop. 2).

Lemma 4 (Finitization)**.** *Let ϕ be any concrete* dHL *formula. Then let $r \notin FV(\phi) \cup BV(\phi)$ (where $BV(\phi)$ is everything bound in ϕ). Then ϕ is valid iff ϕ is finite-domain valid with domain $\{r\} \cup (FV(\phi) \cup BV(\phi))$.*

Lemma 5 (Finite-domain reducibility)**.** *There exists a computable reduction $T(\phi)$ such every* dHL *formula ϕ is finite-domain valid iff the* dL *formula $T(\phi)$ is finite-domain valid.*

Proof Sketch [8]. We present the translation $T(\phi)$. In this translation we write $\vec{x}$ for the vector of all variables $x_1, \ldots, x_n$ in the domain S, and $e^{\vec{\theta}}_{\vec{x}}$ for the vectorial substitution of all θ_i for the corresponding x_i in e. For each of the finitely-many world terms w in ϕ, let $\vec{x}^w$ be a vector of $|S|$ fresh symbols implementing $@_w\vec{x}$. When convenient, we implicitly assume $S \supset \mathrm{FV}(\phi) \cup \mathrm{BV}(\phi)$.

$$T(@_w\theta) = T(\theta)^{\vec{x}^w}_{\vec{x}} \tag{27}$$
$$T(@_w\phi) = [\vec{x} := \vec{x}^w]T(\phi) \tag{28}$$
$$T(w) = (\vec{x} = \vec{x}^w) \tag{29}$$
$$T(\forall s : \mathcal{W}\ \phi) = \forall \vec{x}^s : \mathbb{R}\ T(\phi) \tag{30}$$
$$T(\exists s : \mathcal{W}\ \phi) = \exists \vec{x}^s : \mathbb{R}\ T(\phi) \tag{31}$$
$$T(\downarrow s\ \phi) = [\vec{x}^s := \vec{x}]T(\phi) \tag{32}$$
$$T(\otimes(e_1, \ldots, e_2)) = \otimes(T(e_1), \ldots, T(e_n)) \tag{33}$$

In Equation (33), the notation $\otimes(e_1, \ldots, e_n)$ stands for any of the other dL connectives. In Equation (29), vector equality $\vec{x} = \vec{y}$ stands for the conjunction $\bigwedge_i x_i = y_i$. The result proves by induction. □

Lemma 6 (De-finitization)**.** *A concrete* dL *formula ϕ is valid iff it is finite-domain valid over domain $FV(\phi) \cup BV(\phi)$.*

Theorem 7 (Concrete reducibility)**.** *Concrete* dHL *(i.e., with no rigid symbols) reduces to concrete* dL*. That is, for all concrete* dHL *formulas, the concrete* dL *formula $T(\phi)$ is valid iff ϕ is.*

Proposition 8 (Complexity of T)**.** *$|T(\phi)| \in \Theta(|\phi|^2)$ for concrete ϕ.*

Proof Sketch [8]. To prove the upper bound, note the function $T(\phi)$ expands ϕ by at most a factor of $|S|$. By Lem. 4, $|S| \in O(|\phi|)$ suffices for finitely-valid ϕ. To prove the lower bound, simply observe there exist formulas where the number of program and world variables are both linear in the size. We give a concrete example:

$$\phi_n \equiv \exists s_1 : \mathcal{W}\ \cdots \exists s_n : \mathcal{W}\ \ (@_{s_1}x_1 > 0 \wedge \cdots \wedge @_{s_n}x_n > 0)$$

Now observe that applying T with $S = \{r\} \cup \mathrm{FV}(\phi_n) \cup \mathrm{BV}(\phi_n)$ results in $|T(\phi_n)| \in \Omega(n^2)$. □

We note these theorems *do not* entail (infinite) validity reduction for abstract dHL formulas. Thm. 5 does however preserve finite validity even in the presence of abstract formulas. In summary, reduction fails if and only if abstract constants are allowed to introduce arbitrary new variables.

Corollary 9 (Relative semi-decidability)**.** *Concrete* dHL *is semi-decidable relative to properties of differential equations.*

Sketch [8]. By relative semi-decidability [27] of dL and Thm. 7. □

We reflect on the practical implications of the reducibility results. The reduction requires a finite variable domain, but the natural domain for abstract formulas is infinite. This means verification by reduction is especially ill-suited for proofs using advanced proof techniques like refinement [21] which rely on abstract formulas. Most dHL axioms are also abstract, and so cannot be translated to concrete dL axioms! Even on concrete formulas, the reduction exhibits quadratic blowup and obscures the more convenient proof techniques available in dHL. Thus, direct proof in dHL is strongly preferable to dL reduction for practical purposes.

7 Derived Rules for Bisimulation

The proof calculus of Sec. 5 provides a hybrid-logical core for hyperproperty verification. We connect nondeducibility to this core by deriving a library of rules for information-flow proofs which, being derived, lie outside the core calculus. Our derived rules show that bisimulation, the core proof technique for information flow, derives from nominals in hybrid logic. Because information flow arguments specifically equate values from initial and ending states, we also derive rules for equalities over at-terms. In Sec. 8 and Sec. 9, we apply our library to our smart grid example and see it raises the level of abstraction. Derivations are given in the report [8].

Bisimulation Rules. In Fig. 6, R refers to a relation over world expressions, i.e., $R(i_1, i_2)$ means that worlds i_1 and i_2 are related in R, and m_1, m_2 refer to any **m**iddle states. The rules proceed by destructing a trace on the left; any nondeterminism is resolved identically on the right. Rule @ind is an auxiliary rule for loop induction with nominals, derived from loop induction axiom I. It is in turn used to

$$\text{@ind}\ \frac{@_{i_1} o_1 \to p(o_1) \quad @_{m_1}\langle\alpha\rangle o_1 \wedge p(m_1) \to p(o_1)}{@_{i_1}\langle\alpha^*\rangle o_1 \to p(o_1)}$$

$$\text{BS}^*\ \frac{@_{i_1}\langle\alpha\rangle o_1 \wedge R(i_1,i_2) \to @_{i_2}\langle\alpha\rangle\downarrow o_2\ R(o_1,o_2)}{@_{i_1}\langle\alpha^*\rangle o_1 \wedge R(i_1,i_2) \to @_{i_2}\langle\alpha^*\rangle\downarrow o_2\ R(o_1,o_2)}$$

$$\text{BS}'\ \frac{@_{i_1}\langle\text{ASGN}\rangle o_1 \wedge R(i_1,i_2) \to @_{i_2}[\text{ASGN}]\downarrow o_2\ R(o_1,o_2)}{@_{i_1}\langle\text{ODE}\rangle o_1 \wedge R(i_1,i_2) \to @_{i_2}\langle\text{ODE}\rangle\downarrow o_2\ R(o_1,o_2)}$$

$$\text{BS;}\ \frac{\begin{array}{c}@_{i_1}\langle\alpha\rangle m_1 \wedge R_i(i_1,i_2) \to @_{i_2}\langle\alpha\rangle\downarrow m_2\ R_m(m_1,m_2)\\ @_{m_1}\langle\alpha\rangle o_1 \wedge R_m(m_1,m_2) \to @_{m_2}\langle\alpha\rangle\downarrow o_2\ R_o(o_1,o_2)\end{array}}{@_{i_1}\langle\alpha;\beta\rangle o_1 \wedge R_i(i_1,i_2) \to @_{i_2}\langle\alpha;\beta\rangle\downarrow o_2\ R_o(o_1,o_2)}$$

$$\text{BS}\cup\ \frac{\begin{array}{c}@_{i_1}\langle\alpha\rangle o_1 \wedge R_i(i_1,i_2) \to @_{i_2}\langle\alpha\rangle\downarrow o_2\ R_o(o_1,o_2)\\ @_{i_1}\langle\beta\rangle o_1 \wedge R_i(i_1,i_2) \to @_{i_2}\langle\beta\rangle\downarrow o_2\ R_o(o_1,o_2)\end{array}}{@_{i_1}\langle\alpha\cup\beta\rangle o_1 \wedge R_i(i_1,i_2) \to @_{i_2}\langle\alpha\cup\beta\rangle\downarrow o_2\ R_o(o_1,o_2)}$$

Figure 6. Bisimulation: Derived rules (m_i fresh)

$$\text{NTV}\quad @_i\langle\alpha\rangle j \to @_i\theta = @_j\theta \qquad (\text{FV}(\theta)\cap\text{BV}(\alpha)=\emptyset)$$

$$\text{NT:=}\quad @_i\langle x := F\rangle j \to @_i F = @_j x$$

$$\text{NT;}\ \frac{@_i\langle\alpha\rangle m \to @_i F = @_m H \quad @_m\langle\beta\rangle j \to @_m H = @_j G}{@_i\langle\alpha;\beta\rangle j \to @_i F = @_j G}$$

$$\text{NT}\cup\ \frac{@_i\langle\alpha\rangle j \to @_i F = @_j G \quad @_i\langle\beta\rangle j \to @_i F = @_j G}{@_i\langle\alpha\cup\beta\rangle j \to @_i F = @_j G}$$

$$\text{NT}'\quad @_i\langle x' = F, t' = 1\,\&\,\psi\rangle j \to @_i y((@_j t) - t) = @_j x$$

$$\text{NT}*\ \frac{@_s\langle\alpha\rangle t \to @_s F = @_t F}{@_i\langle\alpha^*\rangle j \to @_i F = @_j F}$$

Figure 7. At-terms: Derived rules (n,m,s,t fresh)

derive Rule BS*, which says any relation R is a bisimulation for loop α^* any time it is (in every state) a bisimulation for α. Rule BS; is Hoare-style composition reasoning raised to the bisimulation level: we can reason about $\alpha;\beta$ by establishing a relation R_m that holds in the intermediate state. Rule BS∪ says a nondeterministic choice maintains a bisimulation if each branch does. In rule BS′, ODE is a differential equation of form $\{x' = \theta, t' = 1\,\&\,\psi\}$ (any model can be trivially extended to this form by adding a fresh variable t) and ASGN $\equiv$ $(t := @_{o_1} t; x := y(t - @_{i_1} t))$ simplifies ODEs to assignments implementing their solutions, plugging in the same duration $@_{o_1} t - @_{i_1} t$ as in the trace $@_{i_1}\langle\text{ODE}\rangle o_1$.

At-Terms. Fig. 7 derives rules for at-term equalities. Rule NTV says a term θ is unchanged if its variables never appear bound in α. The remaining rules are derived from the program axioms and capture the effect of each program on a term. In rule NT′, $y(t)$ is a global solution to $x' = f(x)$.

8 FREEDM: Proving Vulnerability Existence

We now prove that the naïve deterministic controller bat_I based on the published algorithm for FREEDM [2] is insecure: our quantitative dynamical model reveals a bug obscured by the finite event-based abstraction in previous models [3]. Information leaks because when $gr > 0$ is true we can infer $B_i = B_{\max}$ for some i, meaning we have leaked the information that some battery is at capacity. In principle, this could be useful to an attacker, because high charge is associated with vulnerability to (explosive) thermal runaways [13]!

To prove that a system is nondeducibility-*insecure*, we prove the negation of nondeducibility security, i.e., we prove:

Proposition 10 (bat_I is insecure)**.** *Let α_I be the insecure version of the grid α_F and $R(i,j) \equiv (@_i t = @_j t \wedge @_i gr = @_j gr)$. Then $\exists i_1, i_2, o_1 : \mathcal{W}\Big(R(i_1,i_2) \wedge @_{i_1}\langle\alpha_I\rangle o_1 \wedge @_{i_2}[\alpha_I]\downarrow o_2\ \neg R(o_1,o_2)\Big)$ is valid.*

Proof Sketch [8]. We begin by constructing the states i_1, i_2, o_1. First, let i be an arbitrary state. Then let i_1 be the unique state such that $@_i\langle B_i := B_{\max}; t := 0; gr := 0\rangle i_1$ and i_2 the unique state such that $@_i\langle B_i := \frac{1}{2}B_{\max}; t := 0; gr := 0\rangle i_2$. Let o_1 be any state such that $@_{i_1}\langle\alpha_I\rangle o_1$ and $@_{o_1}(t = 0 \wedge gr > 0)$. We know such a state exists by running α_I for exactly one iteration, setting $r_i = \max(0, -p_i)$ and $d_i = 1 + \max(0, p_i)$ which always results in $N_i = 1$. Thus after evolving the differential equation for time 0 we arrive at $gr > 0$. By induction we show that all traces of α_I maintain the invariant $J \equiv (t{\geq}0 \wedge (t{=}0 \to gr \leq 0 \wedge B_i = \frac{1}{2}B_{\max}))$, which can be proven by mechanically applying program axioms, then checking first-order real arithmetic at the leaves. The result follows from $@_{o_1}(t = 0 \wedge gr > 0) \wedge @_{o_2} J \to \neg R(o_1,o_2)$, which itself follows from a simpler arithmetic argument: $@_{o_1} gr \neq @_{o_2} gr$. □

9 FREEDM: Ensuring and Proving Security

We learned that bat_I leaks $B_i = B_{\max}$, ultimately because it is too deterministic: If $gr > 0$ we learn for a fact some $B_i = B_{\max}$. The simplest solution, as taken in bat_S of Fig. 2, is to add the nondeterministic option to use the macrogrid even when a battery has capacity. Now that the macrogrid is always an option, an attacker who observes $gr > 0$ cannot infer $B_i = B_{\max}$ for certain.

We now prove bat_S nondeducibility secure, i.e., an attacker observing only gr and t deduces nothing else. We instantiate Def. 5 to arrive at the theorem statement:

Proposition 11 (Nondeducibility for FREEDM)**.** *First, we define $R(i,j) \equiv \Big(@_i t = @_j t \wedge @_i gr = @_j gr \wedge pre(i) \wedge pre(j)\Big)$ and define $pre(i) \equiv @_i(maxm > 0 \wedge B_{\max} > 0 \wedge thresh > 0)$. Then formula*

$$\forall i_1, i_2, o_1 : \mathcal{W}\ \Big(@_{i_1}\langle\alpha_S\rangle o_1 \wedge R(i_1,i_2) \to @_{i_2}\langle\alpha_S\rangle\downarrow o_2\ R(o_1,o_2)\Big)$$

is valid, where α_S is the secure version of the grid α_F.

Proof Sketch [8]. Recall from Sec. 3 that the heart of the proof is choosing a trace $@_{i_2}\langle\alpha_S\rangle o_2$ which shows the public outputs $@_{o_1} gr$ and $@_{o_1} t$ of trace

$$@_{i_2}\langle\alpha_S\rangle o_2$$

are possible from all related input states i_2. We apply loop rule BS* with $R(i,j) \equiv @_i t = @_j t \wedge @_i gr = @_j gr \wedge pre(i) \wedge pre(j)$. The key proof observation is that for the final values of gr to agree, it suffices that the values agree for both gr and grm at the start of the ODE. We perform this reasoning formally using the composition rule BS; with $R_P(i,j) \equiv R(i,j) \wedge @_i gr = @_j gr \wedge @_i grm = @_j grm \wedge @_i t = @_j t$. This gives two proof obligations: one for the control and one for the physics. We split into four cases for the controller using Lem. 12, according to whether each transformer chooses to migrate.

Lemma 12 (Controller cases)**.** *The* dHL *formula $@_{i_1}\langle ctrl\rangle m_1 \to @_{m_1}\Big((gr = 0 \wedge grm = 0) \vee (gr = n_1 \wedge grm = m) \vee (grm = n_2 \wedge grm = -m) \vee (gr = n_1 + n_2 \wedge grm = 0)\Big)$ is valid, where* ctrl *is as in Fig. 2.*

The second case is representative, the rest are in the report [8].

Case $@_{m_1}(gr = n_1 \wedge grm = m)$: By inspection, it suffices to set $n_1 = @_{m_1}n_1$ and $n_2 = 0, m = 0$. If $@_{m_1}n_1 - @_{i_2}p_1 \geq 0$ then we set $r_1 = -@_{m_1}n_1 - @_{i_2}p_1$ and $d_1 = 0$ else we set $r_1 = 0$ and $d_1 = @_{i_2}p_1 + @_{m_1}n_1$. By algebra, in each case $@_{m_2}n_1 = @_{m_1}n_1$. Then for $i = 2$ if $@_{i_2}p_2 \geq 0$ then set $r_2 = -@_{i_2}p_2$ and $d_2 = 0$ else set $r_2 = 0, d_2 = @_{i_2}p_2$. By algebra, $@_{m_2}n_2 = 0$. Executing the load balancer, we get $m = 0$ because $n_2 = 0$ (thus T_2 is *Normal*). Each case takes the second branch of the battery controller, getting $gr = n_1 + n_2 = n_1$ and $grm = m \cdot -1^2 + m \cdot -1^1 = 0 = m$ as desired. This completes the proof of Prop. 11. □

We have shown how to use dHL to find and resolve hybrid-dynamic information flow (HDIF) vulnerabilities in CPSs. We reflect on how verified safety of a model can contribute to the safety of real-world implementations as well.

Our first model was insecure because a deterministic branch in the battery controller leaked information. This was fixed by introducing a nondeterministic branch. Implementations can simulate the same effect, e.g., with randomized branching. Our models taught us that the battery controller needs such measures, but the load balancing controller, in contrast, is secure even with deterministic control. This knowledge is helpful in practice because measures like randomization typically reduce operational suitability of a controller, so verifying that a deterministic controller is secure enables using the efficient deterministic controller with confidence.

In comparison with many other formal security models, our approach is especially well-suited to verifying security in the presence of *side-channels*. Many would-be side-channels for cyber systems (time, electrical flow, etc.) are *primary* channels in CPS and, as shown in our models, are modeled naturally as hybrid systems. Once these channels are modeled in our hybrid system, they can be verified with the same techniques as any other HDIF!

10 Related Work

Dynamic Logics and Hybrid Logics. The logic dHL is a hybrid version of the dynamic logic dL, adding the ability to verify hyperproperties in addition to safety and liveness properties. The logic d$\mathcal{L}_h$ [26] is a proposed extension of dL with propositional hybrid connectives, but lacks world quantifiers and at-terms, which are essential for information flow. Dynamic logic and first-order hybrid logic have been combined in Combinatory PDL [23], which extends Propositional Dynamic Logic (PDL) with additional set-theoretic program combinators, but has neither at-terms nor assignments, let alone differential equations as dHL does. Hybrid logic has been used in distributed systems [25] reasoning as well. While many CPSs are distributed systems, distributed systems reasoning alone does not suffice to verify hybrid discrete and continuous dynamics. The logic Qd$\mathcal{L}$ [28] allows verification of distributed hybrid dynamics, but is not a hybrid logic, and faces the same challenges with hyperproperties as dL does. First-order hybrid logic [7] (without dynamic-logical or continuous features) and its proof theory [10] have been studied in detail.

Static Information Flow Security. Logics and type systems for information-flow security have been widely studied for discrete programs. Sebelfeld and Myers [32] provide a survey of language-based security approaches. Approaches can be broadly categorized into automatic vs. interactive (or manual) approaches. Automation increases the potential user base, typically at the cost of greatly reduced completeness. When the proof is done automatically, the simplicity of the proof is of little concern, and self-composition [5] can be used to reduce information-flow proofs to a safety property suitable for Hoare and dynamic logics.

In interactive use, self-composition has been noted [35] to make proofs awkward by reducing locality: bisimulation techniques consider the local effect of each statement α on two traces, but self-composition may move the original statement α far from its copy. For humans, a usable calculus as provided by dHL is far more important. As our smart grid example demonstrates, typical HDIFs rely on fine-grained interactions between discrete and continuous dynamics within system loops. This suggests that automated approaches would struggle and that our approach, which is amenable to interactive proof, is merited. An approach analogous to proof by reduction from dHL to dL has been implemented for the dynamic logic JavaDL in the theorem prover KeY (which supports both automatic and interactive proof, but not nominals). To avoid the awkwardness of the reduction approach, calculi meant for interactive use [6, 22] typically build in special-purpose relations for information flow. The disadvantage of such calculi is that baking in these relations prevents generalizing to other hyperproperties.

We strike a middle ground with dHL: The proof techniques we would expect of a dedicated calculus are easily implemented as derived rules, yet we maintain the generality to express arbitrary safety and liveness properties and hyperproperties as well. Our development hints that the relationship between hybrid logic and hyperproperties is general and deserves further exploration.

While we present the first HDIF result for a CPS, information flow has been verified for discrete models of several different CPSs via model-checking; e.g., Akella [3] has verified process algebra models of FREEDM and Wang [36] has verified a Petri-net model of a pipeline network. The absence of continuous dynamics amounts to a significant *model gap* between these models and reality. HDIFs greatly narrow the model gap: for example, our HDIF analysis of our FREEDM model revealed a vulnerability that was not visible in the discrete model of Akella [3].

11 Conclusion and Future Work

We introduced dHL, a hybrid logic for verifying information-flow security properties of hybrid dynamical systems in order to ensure the security of critical cyber-physical systems (CPS). In contrast with previous approaches, it allows verifying cyber-physical hybrid-dynamic information flows (HDIFs), communicating information through both discrete computation and physical dynamics, so security is ensured even when attackers observe continuously-changing values in continuous time. It achieves this by combining dL, a logic for hybrid dynamical systems, with hybrid-logical features enabling explicit reference to program states. This provides a novel way to verify information flow: information flow properties are hyperproperties, which are expressed naturally in hybrid logic via its ability to refer freely to states from multiple traces simultaneously. The foundation of hybrid logic allows verification (and falsification) of security in a common system at no added complexity, and we expect the same system can support additional notions of information flow such as non-interference, as well as arbitrary hyperproperties. We introduced a calculus for dHL, proved it sound, and derived high-level bisimulation rules for information flow proofs. Our use

of uniform substitution provides modularity: we can instantiate all existing dL axioms with dHL formulas and need not individually reprove that each axiom is a valid formula of dHL. Uniform substitution also provides a clear path for extending the dL theorem-prover KeYmaera X [15] and soundness formalization [9] with dHL.

We showed that dHL is capable of verifying the presence or absence of information-flow vulnerabilities in realistic hybrid models. As an example, we debugged and then verified a hybrid model of the FREEDM [19] smart grid controller based on the published algorithm [2] with load-balancing and distributed energy generation and storage, all important features for practical grids. Moreover, the proof demonstrates both i) the close correspondence between dHL information-flow proofs and natural-language proofs and ii) the non-trivial proof arguments that quickly arise when mixing cyber and physical dynamics.

The main places where dHL proofs require more effort than an informal proof were in introducing names for intermediate states and observing the effect of a program on an individual term. We plan to provide a proof format in the eventual KeYmaera X implementation that allows us to automate the majority of these low-level steps, making proofs efficiently match to human intuition. Our information-flow arguments depend closely on the exact semantics of the program and do not follow from, e.g., simple syntactic checks on variable dependencies, meaning the expressive power provided by dHL's deductive calculus is essential for verifying realistic CPS information flow problems. A major novelty in both the logic dHL and our model of FREEDM is the presence of HDIFs that mix discrete cyber and continuous physical flows. These cyber-physical flows arise naturally in many other critical applications, such as oil and natural gas networks, canals, smart homes, medical devices, and vehicles, which deserve future exploration.

Lastly, we also wish to explore other uses of dHL beyond hyperproperty verification. Derived features such as refinement and universal modalities have been explored in Combinatory Dynamic Logic [23], and may have applications [21] in modular hybrid systems modeling and verification.

Acknowledgments

We thank Hannah Gommerstadt, Yong Kiam Tan and Stefan Mitsch for feedback and discussions on earlier drafts of this paper, and thank the anonymous referees for their detailed comments.

This material is based upon work supported by the National Science Foundation under NSF CAREER Award CNS-1054246 and by the AFOSR under grant number FA9550-16-1-0288. The first author was supported by the Department of Defense through the National Defense Science & Engineering Graduate Fellowship Program.

References

[1] Ravi Akella and Bruce M. McMillin. 2010. Information flow analysis of energy management in a smart grid. In *Computer Safety, Reliability, and Security (LNCS)*, Erwin Schoitsch (Ed.), Vol. 6351. Springer, 263–276.

[2] Ravi Akella, Fanjun Meng, Derek Ditch, Bruce McMillin, and Mariesa Crow. 2010. Distributed power balancing for the FREEDM system. In *SmartGridComm*. IEEE.

[3] Ravi Akella, Han Tang, and Bruce M. McMillin. 2010. Analysis of information flow security in cyber-physical systems. *IJCIP* 3, 4 (2010), 157–173.

[4] P. G. Allen. 1991. A comparison of non-interference and non-deducibility using CSP. In *CSFW*. IEEE, 43–54.

[5] Gilles Barthe, Pedro R. D'Argenio, and Tamara Rezk. 2011. Secure information flow by self-composition. *Mathematical Structures in Computer Science* 21, 6 (2011), 1207–1252.

[6] Nick Benton. 2004. Simple relational correctness proofs for static analyses and program transformations. In *POPL 2004*, Neil D. Jones and Xavier Leroy (Eds.). ACM.

[7] Patrick Blackburn and Jerry Seligman. 1995. Hybrid languages. *Journal of Logic, Language and Information* 4, 3 (1995), 251–272.

[8] Brandon Bohrer and André Platzer. 2018. *A hybrid, dynamic logic for hybrid-dynamic information flow*. Technical Report CMU-CS-18-105. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[9] Brandon Bohrer, Vincent Rahli, Ivana Vukotic, Marcus Völp, and André Platzer. 2017. Formally verified differential dynamic logic. In *CPP 2017*, Yves Bertot and Viktor Vafeiadis (Eds.). ACM.

[10] Torben Braüner. 2011. Hybrid logic and its proof-theory. *Applied Logic Series* 37 (2011).

[11] Alonzo Church. 1956. *Introduction to Mathematical Logic*. Princeton University Press.

[12] Michael R. Clarkson and Fred B. Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210.

[13] Xuning Feng, Minggao Ouyang, Xiang Liu, Languang Lu, Yong Xia, and Xiangming He. 2018. Thermal runaway mechanism of lithium ion battery for electric vehicles: A review. *Energy Storage Materials* 10 (2018), 246–267.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *CAV 2011 (LNCS)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.), Vol. 6806. Springer.

[15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. 2015. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *CADE (LNCS)*, Amy Felty and Aart Middeldorp (Eds.), Vol. 9195. Springer.

[16] Thoshitha T. Gamage, Bruce M. McMillin, and Thomas P. Roth. 2010. Enforcing information flow security properties in cyber-physical systems: A generalized framework based on compensation. In *COMPSAC Workshops 2010*. IEEE.

[17] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. 2008. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *S&P 2008*. IEEE.

[18] Thomas A. Henzinger. 1996. The theory of hybrid automata. In *LICS 1996*. IEEE.

[19] A. Q. Huang. 2009. Renewable energy system research and education at the NSF FREEDM systems center. In *Power & Energy Society General Meeting, 2009*. IEEE.

[20] Ruggero Lanotte, Massimo Merro, Riccardo Muradore, and Luca Viganò. 2017. A formal approach to cyber-physical attacks. In *CSF 2017*. IEEE.

[21] Sarah M. Loos and André Platzer. 2016. Differential refinement logic. In *LICS*, Natarajan Shankar (Ed.). ACM.

[22] Aleksandar Nanevski, Anindya Banerjee, and Deepak Garg. 2011. Verification of information flow and access control policies with dependent types. In *S&P*. IEEE.

[23] Solomon Passay and Tinko Tinchev. 1991. An essay in combinatory dynamic logic. *Inf. Comput.* 93, 2 (1991), 263–332.

[24] Solomon Passy and Tinko Tinchev. 1985. Quantifiers in combinatory PDL: completeness, definability, incompleteness. In *FCT 1985 (LNCS)*, Lothar Budach (Ed.), Vol. 2751. Springer.

[25] Dirk Pattinson and Bernhard Reus. 2005. A complete temporal and spatial logic for distributed systems. In *FroCoS 2005*, Bernhard Gramlich (Ed.). Springer.

[26] André Platzer. 2007. Towards a hybrid dynamic logic for hybrid dynamic systems, In International Workshop on Hybrid Logic, Patrick Blackburn, Thomas Bolander, Torben Braüner, Valeria de Paiva, and Jørgen Villadsen (Eds.). *ENTCS* 174.

[27] André Platzer. 2008. Differential dynamic logic for hybrid systems. *J. Autom. Reas.* 41, 2 (2008), 143–189.

[28] André Platzer. 2012. A complete axiomatization of quantified differential dynamic logic for distributed hybrid systems. *Log. Meth. Comput. Sci.* 8, 4 (2012), 1–44. Special issue for selected papers from CSL'10.

[29] André Platzer. 2016. Logic & proofs for cyber-physical systems. In *IJCAR (LNCS)*, Nicola Olivetti and Ashish Tiwari (Eds.), Vol. 9706. Springer, 15–21.

[30] André Platzer. 2017. A complete uniform substitution calculus for differential dynamic logic. *J. Autom. Reas.* 59, 2 (2017), 219–265.

[31] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. 2013. Heart-to-heart (H2H): authentication for implanted medical devices. In *CCS 2013*, Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung (Eds.). ACM.

[32] Andrei Sabelfeld and Andrew C. Myers. 2003. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications* 21, 1 (2003), 5–19.

[33] Krishna Sampigethaya, Radha Poovendran, Sudhakar Shetty, Terry Davis, and Chuck Royalty. 2011. Future E-enabled aircraft communications and security: The next 20 years and beyond. *Proc. IEEE* 99, 11 (2011), 2040–2055.

[34] Christoph Scheben and Peter H. Schmitt. 2011. Verification of information flow properties of Java programs without approximations. In *FoVeOOS (LNCS)*, Bernhard Beckert, Ferruccio Damiani, and Dilian Gurov (Eds.), Vol. 7421. Springer.

[35] Tachio Terauchi and Alexander Aiken. 2005. Secure information flow as a safety problem. In *SAS (LNCS)*, Chris Hankin and Igor Siveroni (Eds.), Vol. 3672. Springer.

[36] Jingming Wang and Huiqun Yu. 2014. Analysis of the composition of non-deducibility in cyber-physical systems. *Applied Mathematics & Information Sciences* 8, 6 (2014), 3137–3143.