Uniform Substitution for Dynamic Logic with Communicating Hybrid Programs

Marvin Brieger[1(✉)], Stefan Mitsch[2], and André Platzer[2,3]

[1] LMU Munich, Munich, Germany
marvin.brieger@sosy.ifi.lmu.de
[2] Carnegie Mellon University, Pittsburgh, USA
smitsch@cs.cmu.edu, platzer@kit.edu
[3] Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. This paper introduces a uniform substitution calculus for dL_{CHP}, the dynamic logic of communicating hybrid programs. Uniform substitution enables parsimonious prover kernels by using axioms instead of axiom schemata. Instantiations can be recovered from a single proof rule responsible for soundness-critical instantiation checks rather than being spread across axiom schemata in side conditions. Even though communication and parallelism reasoning are notorious for necessitating subtle soundness-critical side conditions, uniform substitution when generalized to dL_{CHP} manages to limit and isolate their conceptual overhead. Since uniform substitution has proven to simplify the implementation of hybrid systems provers substantially, uniform substitution for dL_{CHP} paves the way for a parsimonious implementation of theorem provers for hybrid systems with communication and parallelism.

Keywords: Uniform substitution · Parallel programs · Differential dynamic logic · Assumption-commitment reasoning · CSP

1 Introduction

Hybrid systems and parallel systems are notoriously subtle to analyze. Combining both not only culminates these subtleties but is further complicated because parallel hybrid systems are interlocked by synchronization in a shared global time. The *dynamic logic of communicating hybrid programs* dL_{CHP} [6] tames the complexity of parallel hybrid systems providing a compositional proof calculus that disentangles reasoning into purely discrete, continuous, and communication pieces. However, the calculus is subject to schematic side conditions whose implementation is generally error-prone causing large soundness-critical code bases [30]. In particular, compositional reasoning about parallelism as in the idealized proof rule in Fig. 1 holds the challenge to exhaustively characterize *all* side conditions required to make *all* instances of this proof rule sound. Proof

$$\frac{[\alpha]\varphi \qquad [\beta]\psi}{[\alpha \parallel \beta](\varphi \wedge \psi)}\ (\star\star)$$

Fig. 1. The proof rule is only sound under subtle side conditions $(\star\star)$.

B. Pientka and C. Tinelli (Eds.): CADE 2023, LNAI 14132, pp. 96–115, 2023.
https://doi.org/10.1007/978-3-031-38499-8_6

systems for discrete parallelism [1,19,27,35,44,46] already have complicated side conditions, but complexity only increases with continuous interactions in shared global time.

In order to compositionally support compositional reasoning for parallel hybrid systems, this paper generalizes Church's uniform substitution [8] and develops a uniform substitution calculus [30–32] for $\mathsf{dL}_{\mathrm{CHP}}$. Uniform substitution modularizes the calculus itself enabling its parsimonious implementation. Although applicable to discrete parallelism, the $\mathsf{dL}_{\mathrm{CHP}}$ development resolves the inherent challenge that parallel hybrid systems always synchronize in time.

Uniform substitution adopts a finite list of concrete formulas as axioms instead of an infinite set of formulas via axiom schemata with side conditions. This enables theorem provers without the extensive algorithmic checks otherwise required for each schema to sort out unsound instances. Thanks to the proof rule US for uniform substitution, only sound instances derive from the axioms such that the parallel composition rule in $\mathsf{dL}_{\mathrm{CHP}}$ could be adopted almost literally as above, but with all the soundness-critical checking encapsulated solely in rule US. Thanks to US's checking, parallel systems reasoning even reduces to a single parallel injection axiom $[\alpha]\psi \to [\alpha \parallel \beta]\psi$ that merely describes the preservation of property ψ of one parallel component α in the parallel system $\alpha \parallel \beta$. Proofs about $\alpha \parallel \beta$ reduce to a sequence of property embeddings with this axiom from local abstractions of the subcomponents, which combine soundly due to US.

Soundness checks in uniform substitution are ultimately determined by the binding structures as identified in the static semantics. The development of uniform substitution for $\mathsf{dL}_{\mathrm{CHP}}$ is, therefore, grounded in the following key observation: Communication and parallelism both cause additional binding structure that needs attention in the substitution process performed by rule US:

(B I) Expressions depend on communication along (co)finite channel sets (besides finitely many free variables), which, by the core substitution principle [8], must not be introduced free into contexts where they are written.

(B II) Subprograms in a parallel context need to be restricted in the variables and channels written as compositional proof rules for parallelism require local abstractions of subprograms not depending on the internals of the context [35].

Grounded in the need for abstraction (B II), $[\alpha]\psi \to [\alpha \parallel \beta]\psi$ can only be adopted as a sound axiom schema if α and β do not share state, and if program β does not interfere with the contract ψ, i.e., (i) ψ has no free variables bound by β (with exceptions), and (ii) ψ does not depend on communication channels written by β (except for channels joint with α). This extensive side condition would need nontrivial soundness-critical implementations of $\mathsf{dL}_{\mathrm{CHP}}$ axiom schemata. Still, uniform substitution can be lifted with only small changes locally checking for clashes with written channels, and prohibited variables or channels.

The modularity of uniform substitution is the key to the parsimonious implementation [23] of the theorem prover KeYmaera X [11] for differential dynamic logic dL and differential game logic dGL [29], thus paving the way for a straightforward theorem prover implementation of $\mathsf{dL}_{\mathrm{CHP}}$. Since $\mathsf{dL}_{\mathrm{CHP}}$ conservatively

generalizes dL [6], its uniform substitution calculus inherits the complete [33] axiomatic treatment of differential equation invariants [30]. All proofs are in [7].

2 Dynamic Logic of Communicating Hybrid Programs

This section briefly recaps dL_{CHP} [6], the dynamic logic of communicating hybrid programs (CHPs). It combines hybrid programs [28] with CSP-style communication and parallelism [15]. By assumption-commitment (ac) reasoning [22,46,47], dL_{CHP} allows compositional verification of parallelism in dL. For uniform substitution, function and predicate symbols, and program constants are added.

2.1 Syntax

The set of variables $V = V_{\mathbb{R}} \cup V_{\mathbb{N}} \cup V_{\mathcal{T}}$ has real ($V_{\mathbb{R}}$), integer ($V_{\mathbb{N}}$), and trace ($V_{\mathcal{T}}$) variables. For each $x \in V_{\mathbb{R}}$, the differential symbol x' is in $V_{\mathbb{R}}$, too. The designated variable $\mu \in V_{\mathbb{R}}$ represents the shared global time. The set of channel names is Ω. By convention $x, y \in V_{\mathbb{R}}$, $n \in V_{\mathbb{N}}$, $h \in V_{\mathcal{T}}$, $\text{ch} \in \Omega$, and $z \in V$. Channel set $Y \subseteq \Omega$ is (co)finite. Vectorial expressions are denoted $\bar{e}$. Moreover, $f^{\mathbb{M}}$, $g^{\mathbb{M}}$ are $\mathbb{M}$-valued function symbols and p, q, r are predicate symbols, where argument sorts are annotated by $\_ : \mathbb{M}_1, \ldots, \mathbb{M}_k$. Finally, a, b are program constants.

Definition 1 (Terms). *Terms consist of real* ($\text{Trm}_{\mathbb{R}}$)*, integer* ($\text{Trm}_{\mathbb{N}}$)*, channel* (Trm_{Ω})*, and trace* ($\text{Trm}_{\mathcal{T}}$) *terms, and are defined by the grammar below, where* $\theta, \theta_1, \theta_2 \in \mathbb{Q}[V_{\mathbb{R}}] \subset \text{Trm}_{\mathbb{R}}$ *are polynomials in* $V_{\mathbb{R}}$*:*

$$
\begin{aligned}
\text{Trm}_{\mathbb{R}} :&\quad \eta_1, \eta_2 ::= x \mid f^{\mathbb{R}}(Y, \bar{e}) \mid \eta_1 + \eta_2 \mid \eta_1 \cdot \eta_2 \mid (\theta)' \mid \texttt{val}(te) \mid \texttt{time}(te)\\
\text{Trm}_{\mathbb{N}} :&\quad ie_1, ie_2 ::= n \mid f^{\mathbb{N}}(Y, \bar{e}) \mid ie_1 + ie_2 \mid |te|\\
\text{Trm}_{\Omega} :&\quad ce_1, ce_2 ::= f^{\Omega}(Y, \bar{e}) \mid \texttt{chan}(te)\\
\text{Trm}_{\mathcal{T}} :&\quad te_1, te_2 ::= h \mid f^{\mathcal{T}}(Y, \bar{e}) \mid \langle ch, \theta_1, \theta_2 \rangle \mid te_1 \cdot te_2 \mid te \downarrow Y \mid te[ie]
\end{aligned}
$$

Real terms are polynomials in $V_{\mathbb{R}}$ enriched with function symbols $f^{\mathbb{R}}(Y, \bar{e})$ (including constants $c \in \mathbb{Q}$) only depending on communication along channels Y and terms $\bar{e}$, differential terms $(\theta)'$, and $\texttt{val}(te)$ and $\texttt{time}(te)$, which access the value and the timestamp of the last communication in te, respectively. By convention, $\theta \in \mathbb{Q}[V_{\mathbb{R}}]$ denotes a pure polynomial in $V_{\mathbb{R}}$ without $(\cdot)'$, $\texttt{val}(\cdot)$, and $\texttt{time}(\cdot)$ as they occur in programs. For simplicity, we do not define $\mathbb{Q}[V_{\mathbb{R}}] \subset \text{Trm}_{\mathbb{R}}$ as a fifth term sort but use the convention that function symbols $g^{\mathbb{R}}$ can only be replaced with $\mathbb{Q}[V_{\mathbb{R}}]$-terms. Integer terms are variables n, function symbols $f^{\mathbb{N}}(Y, \bar{e})$ (including constants 0, 1), addition, and length $|te|$ of trace term te.[1] The function symbol $f^{\Omega}(Y, \bar{e})$ includes constants $\text{ch} \in \Omega$, and $\texttt{chan}(te)$ is channel access. Trace terms record the communication history of programs. They encompass variables h, function symbols $f^{\mathcal{T}}(Y, \bar{e})$ (including the empty trace ϵ), communication items $\langle \text{ch}, \theta_1, \theta_2 \rangle$ with value θ_1 and timestamp θ_2, projection

[1] Omitting multiplication results in decidable Presburger arithmetic [34].

$te \downarrow Y$ onto channels Y, and access $te[ie]$ of the ie-th item in te. Where useful, $\mathtt{op}(\bar{e})$ denotes built-in function symbols of fixed interpretation, e.g., $\cdot + \cdot$.

$\mathsf{dL}_{\mathrm{CHP}}$'s context-sensitive program and formula syntax presumes notions of free and bound variables (Sect. 2.3) defined on the context-free syntax:

Definition 2 (Programs). Communicating hybrid programs *are defined by the following grammar, where* $\theta \in \mathbb{Q}[V_{\mathbb{R}}]$ *is a polynomial in* $V_{\mathbb{R}}$ *and* $\chi \in \mathrm{FOL}_{\mathbb{R}}$ *is a formula of first-order real-arithmetic. In* $\alpha \parallel \beta$*, the subprograms must not share state but can share time and history, i.e.,* $\mathsf{BV}(\alpha) \cap \mathsf{BV}(\beta) \subseteq \{\mu, \mu'\} \cup V_{\mathcal{T}}$.[2]

$$\alpha, \beta ::= a(\!|Y, \bar{z}|\!) \mid x := \theta \mid x := * \mid ?\chi \mid \{x' = \theta \,\&\, \chi\} \mid \alpha; \beta \mid \alpha \cup \beta \mid \alpha^* \mid ch(h)!\theta \mid ch(h)?x \mid \alpha \parallel \beta$$

The program constant $a(\!|Y, \bar{z}|\!)$ restricts the written channels to $Y \subseteq \Omega$ and the bound variables to $\bar{z} \subseteq V_{\mathbb{R}} \cup V_{\mathcal{T}}$, where Y and $\bar{z}$ are (co)finite. Instead of $a(\!|Y, \bar{z}|\!)$, write a if Y and $\bar{z}$ can be arbitrary. Assignment $x := \theta$ updates x to θ, nondeterministic assignment $x := *$ assigns an arbitrary real value to x, and the test $?\chi$ does nothing if χ holds and aborts the computation otherwise. The continuous evolution $\{x' = \theta \& \chi\}$ follows the ODE $x' = \theta$ for any duration as long as formula χ is not violated. The global time μ evolves with every continuous evolution according to ODE $\mu' = 1$. Sequential composition $\alpha; \beta$ executes β after α, choice $\alpha \cup \beta$ executes α or β nondeterministically, α^* repeats α zero or more times, $\mathrm{ch}(h)!\theta$ sends θ along channel ch, and $\mathrm{ch}(h)?x$ receives a value into variable x along channel ch. The trace variable h records communication. Finally, $\alpha \parallel \beta$ executes α and β in parallel synchronized in global time μ.

Example 3. The program $\mathtt{ct}^* \parallel \mathtt{ve}^*$ models a simplified cruise control [24]. The vehicle $\mathtt{ve}$ repeatedly receives a target velocity $v_{\mathtt{ve}}^{\mathrm{tr}}$ from the controller $\mathtt{ct}$ along channel tar. The target $v_{\mathtt{ct}}^{\mathrm{tr}}$ sent by $\mathtt{ct}$ is in range $[0, V]$. Hence, $\mathtt{ve}$'s velocity $v_{\mathtt{ve}}$ stays in range $[0, V]$ within the $\epsilon > 0$ time units till the next communication if $v_{\mathtt{ve}} \in [0, V]$ held initially. The evolution $\{t' = 1\}$ allows passage of time in $\mathtt{ct}$.

$$\mathtt{ct} \equiv v_{\mathtt{ct}}^{\mathrm{tr}} := *; ?(0 \le v_{\mathtt{ct}}^{\mathrm{tr}} \le V); \mathrm{tar}(h)!v_{\mathtt{ct}}^{\mathrm{tr}}; \{t' = 1\}$$

$$\mathtt{ve} \equiv \mathrm{tar}(h)?v_{\mathtt{ve}}^{\mathrm{tr}}; a_{\mathtt{ve}} := \frac{v_{\mathtt{ve}}^{\mathrm{tr}} - v_{\mathtt{ve}}}{\epsilon}; t_0 := \mu; \{v_{\mathtt{ve}}' = a_{\mathtt{ve}} \,\&\, \mu - t_0 \le \epsilon\}$$

Definition 4 (Formulas). *Formulas are defined by the grammar below for relations* $\sim$*, terms* $e_1, e_2 \in \mathrm{Trm}$ *of equal sort, and* $z \in V$*. Moreover, the ac-formulas are unaffected by state change in* α*, i.e.,* $(\mathsf{FV}(\mathsf{A}) \cup \mathsf{FV}(\mathsf{C})) \cap \mathsf{BV}(\alpha) \subseteq V_{\mathcal{T}}$.

$$\varphi, \psi, \mathsf{A}, \mathsf{C} ::= e_1 \sim e_2 \mid p(Y, \bar{e}) \mid \neg\varphi \mid \varphi \wedge \psi \mid \forall z\, \varphi \mid [\alpha]\psi \mid [\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi$$

The formulas combine first-order dynamic logic with ac-reasoning. Predicate symbols $p(Y, \bar{e})$ depend on channels Y and terms $\bar{e}$. The ac-box $[\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi$

[2] Previous work [6] disallows reading of variables bound in parallel as their change is not observable. This restriction is conceptually desirable but not soundness-critical. Here we drop it for simplicity, but it could be maintained by US as well.

expresses that C holds after each communication event and ψ in the final state, for all runs of α whose incoming communication satisfies A. Other connectives $\vee$, $\rightarrow$, $\leftrightarrow$ and quantifiers $\exists z\, \varphi \equiv \neg\forall z\, \neg\varphi$ can be derived. The relations $\sim$ include $=$ for all term sorts, $\geq$ on real and integer terms, and prefixing $\preceq$ on trace terms.

By convention, the predicate symbol $q_{\mathbb{R}}$ can only be replaced with formulas of first-order real arithmetic. It serves as placeholder for tests χ in CHPs.

Example 5. The cruise control from Example 3 is safe if its velocity stays in range $[0, V]$. This can be expressed with the formula $\varphi \rightarrow [\mathtt{ct}^* \parallel \mathtt{ve}^*]\psi_{\text{safe}}$, where $\psi_{\text{safe}} \equiv 0 \leq v_{\mathtt{ve}} \leq V$ and $\varphi \equiv \psi_{\text{safe}} \wedge \epsilon > 0 \wedge V > 0$.

2.2 Semantics

A *trace* $\tau = (\tau_1, ..., \tau_k)$ is a finite chronological sequence of communication events $\tau_i = \langle \mathrm{ch}_i, d_i, s_i \rangle$, where $\mathrm{ch}_i \in \Omega$, and $d_i \in \mathbb{R}$ is the communicated value, and $s_i \in \mathbb{R}$ is a timestamp such that $s_i \leq s_j$ for $1 \leq i < j \leq k$. A *recorded trace* $\tau = (\tau_1, ..., \tau_k)$ additionally carries a trace variable $h_i \in V_{\mathcal{T}}$ with each event, i.e., $\tau_i = \langle h_i, \mathrm{ch}_i, d_i, s_i \rangle$. For variable $z \in V_{\mathbb{M}}$ and $\mathbb{M} \in \{\mathbb{R}, \mathbb{N}, \mathcal{T}\}$, let $type(z) = \mathbb{M}$. A *state* v maps each $z \in V$ to a value $v(z) \in type(z)$. The sets of traces, recorded traces, and states are denoted $\mathcal{T}$, $\mathcal{T}_{\text{rec}}$, and $\mathcal{S}$, respectively.

For $d \in type(z)$, the state v_z^d is the modification of v at z to d. For $\tau \in \mathcal{T}_{\text{rec}}$, the trace $\tau(h) \in \mathcal{T}$ is obtained from the subsequence of τ carrying $h \in V_{\mathcal{T}}$ by removing the carried variable. *State-trace concatenation* $v \cdot \tau \in \mathcal{S}$ for $\tau \in \mathcal{T}_{\text{rec}}$, appends $\tau(h)$ to v at h for all $h \in V_{\mathcal{T}}$. The *projection* $\tau \downarrow Y$ of (recorded) trace τ is the subsequence of all communication events in τ whose channel is in $Y \subseteq \Omega$. The *state projection* $v \downarrow Y \in \mathcal{S}$ modifies v at h to $v(h) \downarrow Y$ for all $h \in V_{\mathcal{T}}$.

An *interpretation* I assigns a function $I(f^{\mathbb{M}} : \mathbb{M}_1, \ldots, \mathbb{M}_k) : \bigtimes_{i=1}^{k} \mathbb{M}_i \rightarrow \mathbb{M}$ to each function symbol $f^{\mathbb{M}}$ that is smooth in all real-valued arguments if $\mathbb{M} = \mathbb{R}$, and a relation $I(p : \mathbb{M}_1, \ldots, \mathbb{M}_k) \subseteq \bigtimes_{i=1}^{k} \mathbb{M}_i$ to each k-ary predicate symbol p.

Definition 6 (Term Semantics). *The* valuation $Iv[\![e]\!] \in \mathbb{R} \cup \mathbb{N} \cup \Omega \cup \mathcal{T}$ *of term* e *in interpretation* I *and state* v *is defined as follows:*

$$
\begin{aligned}
Iv[\![z]\!] &= v(z) \\
Iv[\![f(Y, e_1, ..., e_k)]\!] &= I(f)(I\tilde{v}[\![e_1]\!], ..., I\tilde{v}[\![e_k]\!]) \quad \text{where } \tilde{v} = v \downarrow Y \\
Iv[\![\mathtt{op}(e_1, \ldots, e_k)]\!] &= \mathtt{op}(Iv[\![e_1]\!], \ldots, Iv[\![e_k]\!]) \quad \text{for builtin } \mathtt{op} \in \{\cdot + \cdot, \cdot \downarrow Y, \ldots\} \\
Iv[\![(\theta)']\!] &= \sum_{x \in V_{\mathbb{R}}} v(x') \frac{\partial Iv[\![\theta]\!]}{\partial x}
\end{aligned}
$$

The projection $\tilde{v} = v \downarrow Y$ ensures that $f(Y, \bar{e})$ only depends on Y, i.e., the communication in v along channels $Y^{\complement}$ does not matter. The differentials $(\theta)'$ have a semantics describing the local rate of change of θ [30].

The denotational semantics of CHPs [6] combines dL's Kripke semantics [30] with a linear history semantics [47] and a global notion of time. Denotations are subsets of $\mathcal{D} = \mathcal{S} \times \mathcal{T}_{\text{rec}} \times \mathcal{S}_\perp$ with $\mathcal{S}_\perp = \mathcal{S} \cup \{\perp\}$. Final state $\perp$ marks an unfinished computation, i.e., it still can be continued or was aborted due to a

failing test. If ($w' = \bot$ and $\tau' \preceq \tau$), where $\preceq$ is the prefix relation on traces, or $(\tau', w') = (\tau, w)$, then (τ', w') is a prefix of (τ, w) written $(\tau', w') \preceq (\tau, w)$. Since (even empty) communication of unfinished computations is still observable, denotations $D \subseteq \mathcal{D}$ of CHPs are prefix-closed and total, i.e., $(v, \tau, w) \in D$ and $(\tau', w') \preceq (\tau, w)$ implies $(v, \tau', w') \in D$, and $\bot_{\mathcal{D}} \subseteq D$ with $\bot_{\mathcal{D}} = \mathcal{S} \times \{\epsilon\} \times \{\bot\}$. Moreover, all $(v, \tau, w) \in D$ are chronological, i.e., $v(\mu) \leq w(\mu)$ and when $\tau = (\tau_1, \ldots, \tau_k) \neq \epsilon$ and let $\tau_i(\mu) = (\langle h_i, \mathrm{ch}_i, d_i, s_i\rangle)(\mu) = s_i$, then $v(\mu) \leq \tau_1(\mu)$ and if $w \neq \bot$, then $\tau_k(\mu) \leq w(\mu)$. Note that τ is chronological as all traces are.

The interpretation $I(a(\!|Y, \bar{z}|\!)) \subseteq \mathcal{D}$ of a program constant $a(\!|Y, \bar{z}|\!)$ is a prefix-closed and total set of chronological computations that (i) only communicate along (write) channels Y and (ii) only bind variables $\bar{z}$. More precisely, for all $(v, \tau, w) \in I(a(\!|Y, \bar{z}|\!))$, we have (i) $\tau \downarrow Y^{\complement} = \epsilon$, and (ii) $v = w$ on $V_{\mathcal{T}}$ and $w \cdot \tau = v$ on $\bar{z}^{\complement}$. For $D, M \subseteq \mathcal{D}$, we define $D_{\bot} = \{(v, \tau, \bot) \mid (v, \tau, w) \in D\}$, and $(v, \tau, w) \in D \triangleright M$ if $(v, \tau_1, u) \in D$ and $(u, \tau_2, w) \in M$ exist with $\tau = \tau_1 \cdot \tau_2$. For states w_α, w_β, the merged state $w_\alpha \oplus w_\beta$ is $\bot$ if one of the substates w_α or w_β is $\bot$. Otherwise, $w_\alpha \oplus w_\beta = w_\alpha$ on $\mathsf{BV}(\alpha)$ and $w_\alpha \oplus w_\beta = w_\beta$ on $\mathsf{BV}(\alpha)^{\complement}$ (or, equivalently by syntactic well-formedness, on $\mathsf{BV}(\beta)^{\complement}$ and $\mathsf{BV}(\beta)$, respectively). If Y is the set of all channel names occurring in α, we write $\tau \downarrow \alpha$ for $\tau \downarrow Y$.

Definition 7 (Program semantics). *Given an interpretation I, the* semantics *$I[\![\alpha]\!] \subseteq \mathcal{D}$ of a CHP α is defined as follows, where $\bot_{\mathcal{D}} = \mathcal{S} \times \{\epsilon\} \times \{\bot\}$ and $\vDash$ denotes the satisfaction relation (Definition 8):*

$$I[\![a(\!|Y, \bar{z}|\!)]\!] = I(a(\!|Y, \bar{z}|\!))$$

$$I[\![x := \theta]\!] = \bot_{\mathcal{D}} \cup \{(v, \epsilon, w) \mid w = v_x^d \text{ where } d = Iv[\![\theta]\!]\}$$

$$I[\![x := *]\!] = \bot_{\mathcal{D}} \cup \{(v, \epsilon, w) \mid w = v_x^d \text{ where } d \in \mathbb{R}\}$$

$$I[\![?\chi]\!] = \bot_{\mathcal{D}} \cup \{(v, \epsilon, v) \mid Iv \vDash \chi\}$$

$$I[\![\{x' = \theta \,\&\, \chi\}]\!] = \bot_{\mathcal{D}} \cup \big\{(v, \epsilon, \varphi(s)) \mid v = \varphi(0) \text{ on } \{\mu', x'\}^{\complement}, \text{ and } \varphi(\zeta) = \varphi(0) \text{ on } \{x, x', \mu, \mu'\}^{\complement}, \text{ and } I\varphi(\zeta) \vDash \mu' = 1 \wedge x' = \theta \wedge \chi \text{ for all } \zeta \in [0, s] \text{ and a solution } \varphi : [0, s] \to \mathcal{S} \text{ with } \varphi(\zeta)(z') = \frac{d\varphi(t)(z)}{dt}(\zeta) \text{ for } z \in \{x, \mu\}\big\}$$

$$I[\![ch(h)!\theta]\!] = \{(v, \tau, w) \mid (\tau, w) \preceq (\langle h, ch, d, v(\mu)\rangle, v) \text{ where } d = Iv[\![\theta]\!]\}$$

$$I[\![ch(h)?x]\!] = \{(v, \tau, w) \mid (\tau, w) \preceq (\langle h, ch, d, v(\mu)\rangle, v_x^d) \text{ where } d \in \mathbb{R}\}$$

$$I[\![\alpha \cup \beta]\!] = I[\![\alpha]\!] \cup I[\![\beta]\!]$$

$$I[\![\alpha; \beta]\!] = I[\![\alpha]\!] \mathbin{\hat{\circ}} I[\![\beta]\!] \stackrel{\text{def}}{=} (I[\![\alpha]\!])_{\bot} \cup (I[\![\alpha]\!] \triangleright I[\![\beta]\!])$$

$$I[\![\alpha^*]\!] = \bigcup_{n \in \mathbb{N}} (I[\![\alpha]\!])^n = \bigcup_{n \in \mathbb{N}} I[\![\alpha^n]\!] \quad \text{where } \alpha^0 \equiv ?\mathsf{T} \text{ and } \alpha^{n+1} = \alpha; \alpha^n$$

$$I[\![\alpha_1 \parallel \alpha_2]\!] = \left\{ (v, \tau, w_{\alpha_1} \oplus w_{\alpha_2}) \;\middle|\; \begin{array}{l} (v, \tau \downarrow \alpha_j, w_{\alpha_j}) \in I[\![\alpha_j]\!] \text{ for } j = 1, 2, \text{ and} \\ w_{\alpha_1} = w_{\alpha_2} \text{ on } \{\mu, \mu'\}, \text{ and } \tau = \tau \downarrow (\alpha_1 \| \alpha_2) \end{array} \right\}$$

The semantics is indeed constructed prefix-closed, total, and chronological. Communication τ of $\alpha_1 \parallel \alpha_2$ is implicitly characterized via its subsequences for

the subprograms. By $\tau = \tau \downarrow (\alpha_1 \parallel \alpha_2)$, there is no non-causal communication. Joint communication and the whole computation are synchronized in global time by the projections and by $w_{\alpha_1} = w_{\alpha_2}$ on $\{\mu, \mu'\}$, respectively. Likewise, by projection, communication is synchronously recorded by trace variables.

Definition 8 (Formula semantics). *The* satisfaction $Iv \vDash \phi$ *of a* $\mathsf{dL}_{\mathrm{CHP}}$ *formula* ϕ *in interpretation* I *and state* v *is inductively defined as follows:*

1. $Iv \vDash e_1 \sim e_2$ *if* $Iv[\![e_1]\!] \sim Iv[\![e_2]\!]$ where $\sim$ is any relation symbol
2. $Iv \vDash p(Y, e_1, \ldots, e_k)$ *if* $(I\tilde{v}[\![e_1]\!], \ldots, I\tilde{v}[\![e_k]\!]) \in I(p)$ where $\tilde{v} = v \downarrow Y$
3. $Iv \vDash \varphi \wedge \psi$ *if* $Iv \vDash \varphi$ *and* $Iv \vDash \psi$
4. $Iv \vDash \neg\varphi$ *if* $Iv \nvDash \varphi$, *i.e., it is not the case that* $Iv \vDash \varphi$
5. $Iv \vDash \forall z\, \varphi$ *if* $Iv_z^d \vDash \varphi$ *for all* $d \in type(z)$
6. $Iv \vDash [\alpha]\psi$ *if* $Iw \cdot \tau \vDash \psi$ *for all* $(v, \tau, w) \in I[\![\alpha]\!]$ *with* $w \neq \bot$
7. $Iv \vDash [\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi$ *if for all* $(v, \tau, w) \in I[\![\alpha]\!]$ *the following conditions hold:*

$$\{Iv \cdot \tau' \mid \tau' \prec \tau\} \vDash \mathsf{A} \text{ implies } Iv \cdot \tau \vDash \mathsf{C} \qquad \text{(commit)}$$

$$\big(\{Iv \cdot \tau' \mid \tau' \preceq \tau\} \vDash \mathsf{A} \text{ and } w \neq \bot\big) \text{ implies } Iw \cdot \tau \vDash \psi \qquad \text{(post)}$$

Where $U \vDash \varphi$ *for a set of interpretation-state pairs* U *and any formula* φ *if* $Iv \vDash \varphi$ *for all* $Iv \in U$. *In particular,* $\emptyset \vDash \varphi$.

In item 6 and 7, reachable worlds are built from states v and w, and communication τ, as change of state *and* communication are observable. The strict prefix $\prec$ for the assumption in case (commit) in item 6 excludes (when $\mathsf{A} \equiv \mathsf{C}$) the circularity that commitment C can be shown in states where it is assumed.

2.3 Static Semantics

In the uniform substitution process, checks of free and bound variables, as well as accessed and written channels, separate sound from unsound axiom instantiations. As parallelism requires fine-grained control over channels, the static semantics for dL [30] is lifted to a communication-aware static semantics for $\mathsf{dL}_{\mathrm{CHP}}$. It uses accessed channels to characterize the subsequence of a communication trace influencing truth of a formula even more precisely than free variables.

To precisely grasp free and bound variables, and accessed and written channels, Definition 9 gives a semantic characterization. In this section, formulas are considered truth-valued, i.e., $Iv[\![\phi]\!] = \mathbf{tt}$ if $Iv \vDash \phi$ and $Iv[\![\phi]\!] = \mathbf{ff}$ if $Iv \nvDash \phi$.

Definition 9 (Static semantics). *For term or formula* e, *and program* α, *free variables* $\mathsf{FV}(e)$ *and* $\mathsf{FV}(\alpha)$, *bound variables* $\mathsf{BV}(\alpha)$, *accessed channels* $\mathsf{CN}(e)$, *and written channels* $\mathsf{CN}(\alpha)$ *form the static semantics.*

$$\mathsf{FV}(e) = \{z \in V \mid \exists I, v, \tilde{v} \text{ such that } v = \tilde{v} \text{ on } \{z\}^{\mathsf{C}} \text{ and } Iv[\![e]\!] \neq I\tilde{v}[\![e]\!]\}$$

$$\mathsf{CN}(e) = \{ch \in \Omega \mid \exists I, v, \tilde{v} \text{ such that } v \downarrow \{ch\}^{\mathsf{C}} = \tilde{v} \downarrow \{ch\}^{\mathsf{C}} \text{ and } Iv[\![e]\!] \neq I\tilde{v}[\![e]\!]\}$$

$$\mathsf{FV}(\alpha) = \{z \in V \mid \exists I, v, \tilde{v}, \tau, w \text{ such that } v = \tilde{v} \text{ on } \{z\}^{\mathsf{C}} \text{ and } (v, \tau, w) \in I[\![\alpha]\!], \text{ and there is no } (\tilde{v}, \tilde{\tau}, \tilde{w}) \in I[\![\alpha]\!] \text{ such that } \tilde{\tau} = \tau \text{ and } w = \tilde{w} \text{ on } \{z\}^{\mathsf{C}}\}$$

$$\mathsf{BV}(\alpha) = \{z \in V \mid \exists I, (v, \tau, w) \in I[\![\alpha]\!] \text{ such that } w \neq \bot \text{ and } (w \cdot \tau)(z) \neq v(z)\}$$

$$\mathsf{CN}(\alpha) = \{ch \in \Omega \mid \exists I, (v, \tau, w) \in I[\![\alpha]\!] \text{ such that } \tau \downarrow \{ch\} \neq \epsilon\}$$

The already subtle static semantics of hybrid systems [30] becomes even more subtle with communication and parallelism. For example, CHPs (silently) synchronize with the global time μ, which is free and bound in ODEs, and the differential μ' is bound, i.e.,$\mu \in \mathsf{FV}(\{x' = \theta \,\&\, \chi\})$ and $\mu, \mu' \in \mathsf{BV}(\{x' = \theta \,\&\, \chi\})$ if the evolution has a run of non-zero duration, regardless of whether μ occurs in x. Since reachable worlds of CHPs consist of communication *and* state, bound variables $\mathsf{BV}(\alpha)$ of program α compare v with the state-trace concatenation $w \cdot \tau$ instead of missing τ. Consequently, $h \in \mathsf{BV}(\mathrm{ch}(h)!\theta) \subseteq \mathsf{FV}(\mathrm{ch}(h)!\theta)$, which also reflects that the initial communication never gets lost.

Lemma 10 (Bound effect property). *The sets* $\mathsf{BV}(\alpha)$ *and* $\mathsf{CN}(\alpha)$ *are the smallest sets with the* bound effect property for program α. *That is,* $v = w$ *on* $V_{\mathcal{T}}$ *and* $v = w \cdot \tau$ *on* $\mathsf{BV}(\alpha)^{\complement}$ *if* $w \neq \bot$, *and* $\tau \downarrow \mathsf{CN}(\alpha)^{\complement} = \epsilon$ *for all* $(v, \tau, w) \in I[\![\alpha]\!]$.

By the following *communication-aware* coincidence property, terms and formulas only depend on their free variables, which for trace variables can be further refined to the subtraces whose channels are accessed. This subtrace-level precision is crucial in the soundness proof of the parallel injection axiom as it allows to drop β from $[\alpha \parallel \beta]\psi$ only if β does not write channels of ψ that are not also written by α. The signature $\Sigma(\cdot)$ of an expression denotes all occurring symbols.

Lemma 11 (Coincidence for terms and formulas). *The sets* $\mathsf{FV}(e)$ *and* $\mathsf{CN}(e)$ *are the smallest sets with the* communication-aware coincidence property for term or formula e. *That is, if* $v \downarrow \mathsf{CN}(e) = \tilde{v} \downarrow \mathsf{CN}(e)$ *on* $\mathsf{FV}(e)$ *and* $I = J$ *on* $\Sigma(e)$, *then* $Iv[\![e]\!] = J\tilde{v}[\![e]\!]$. *In particular, for formula* ϕ: $Iv \vDash \phi$ *iff* $J\tilde{v} \vDash \phi$.

Programs communicate but do *not* depend on the recorded history, thus the coincidence property for programs is not communication-aware. However, programs can produce the same communication starting from coinciding states.

Lemma 12 (Coincidence for programs). *The set* $\mathsf{FV}(\alpha)$ *is the smallest set with the* coincidence property for program α. *That is, if* $v = \tilde{v}$ *on* $X \supseteq \mathsf{FV}(\alpha)$, *and* $I = J$ *on* $\Sigma(\alpha)$, *and* $(v, \tau, w) \in I[\![\alpha]\!]$, *then* $(\tilde{v}, \tilde{\tau}, \tilde{w}) \in J[\![\alpha]\!]$ *exists such that* $w = \tilde{w}$ *on* X, *and* $\tau = \tilde{\tau}$, *and* $(w = \bot$ *iff* $\tilde{w} = \bot)$.

3 Uniform Substitution for $\mathsf{dL}_{\mathrm{CHP}}$

In $\mathsf{dL}_{\mathrm{CHP}}$, a uniform substitution [30] σ maps function and predicate symbols to terms (of equal sort) and formulas, respectively, while substituting the arguments of the symbol for their placeholders in the replacement, and program constants are mapped to CHPs. For example, $\sigma = \{f(\cdot) \mapsto \cdot + 1, a \mapsto \mathrm{ch}(h)?v; \{x' = v\}\}$ replaces all occurrences of function symbol f with $\cdot + 1$ while the reserved 0-ary function symbol $\cdot$ marks the positions for the parameter of f in the replacement. Moreover, σ replaces the program constant a with the program $\mathrm{ch}(h)?v; \{x' = v\}$.

The key to sound uniform substitution is that new free variables must not be introduced into a context where they are bound [8]. In the presence of communication, likewise, *new channel access must not be introduced into contexts*

where the channel is written (B I). For parallelism, substitution *must not reveal internals* of the parallel context to the local abstraction of a subprogram (B II), and must not violate state disjointness. The one-pass approach [32] used for dL_{CHP} postpones these checks *and* simply applies the substitution recursively while collecting written variables and channels as taboo set (Fig. 2), thus operates linearly in the input. Clashes between the taboo, and new free variables and channel access are only checked locally at the replacement site. Likewise, clashes between the permitted channels and variables of a program constant, and its replacement program are checked locally.

The substitution operator $\sigma_Z^{U,W}(\alpha)$ for program α takes an input taboo $U \subseteq V \cup \Omega$ and a parallel context $W \subseteq V$, and returns, if defined, the substitution result and a set of output taboos $Z \subseteq V \cup \Omega$. For terms and formulas, the substitution operator σ^U only takes a taboo $U \subseteq V \cup \Omega$ as input. The substitution process clashes, i.e., prevents unsound instantiation, if it were to introduce a free variable or accessed channel into a context where it is bound (B I) *or* if it were to write variables and channels violating abstraction (B II). Moreover, substitution preserves well-formedness of programs and formulas, i.e., substitution clashes if replacements were to violate well-formedness.

$$
\begin{array}{rll}
\sigma^U(z) \equiv z & & \text{for } z \in V \\
\sigma^U(f(Y,e)) \equiv \{\cdot \mapsto \sigma^U(e \downarrow Y)\}^{\emptyset}(\sigma f(\cdot)) & & \text{if } (\mathsf{FV}(\sigma f(\cdot)) \cup \mathsf{CN}(\sigma f(\cdot))) \cap U = \emptyset \\
\sigma^U(\mathsf{op}(e_1,\ldots,e_k)) \equiv \mathsf{op}(\sigma^U(e_1),\ldots,\sigma^U(e_k)) & & \text{for built-in } \mathsf{op} \in \{\cdot+\cdot, \cdot \downarrow Y, \ldots\} \\
\sigma^U((\theta)') \equiv (\sigma^{V\cup\Omega}(\theta))' & &
\end{array}
$$

$$
\begin{array}{rll}
\sigma^U(e_1 \sim e_2) \equiv \sigma^U(e_1) \sim \sigma^U(e_2) & & \\
\sigma^U(p(Y,e)) \equiv \{\cdot \mapsto \sigma^U(e \downarrow Y)\}^{\emptyset}(\sigma p(\cdot)) & & \text{if } (\mathsf{FV}(\sigma p(\cdot)) \cup \mathsf{CN}(\sigma p(\cdot))) \cap U = \emptyset \\
\sigma^U(\neg\varphi) \equiv \neg\sigma^U(\varphi) & & \\
\sigma^U(\varphi \wedge \psi) \equiv \sigma^U(\varphi) \wedge \sigma^U(\psi) & & \\
\sigma^U(\forall z\, \varphi) \equiv \forall z\, \sigma^{U\cup\{z\}}(\varphi) & & \\
\sigma^U([\alpha]\psi) \equiv [\sigma_Z^{U,\emptyset}(\alpha)]\sigma^Z(\psi) & & \\
\sigma^U([\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi) \equiv [\sigma_Z^{U,\emptyset}(\alpha)]_{\{\sigma^Z(\mathsf{A}),\sigma^Z(\mathsf{C})\}}\sigma^Z(\psi) & &
\end{array}
$$

$$
\begin{array}{rll}
\sigma^{U,W}_{U\cup\mathsf{BV}(\sigma a)\cup\mathsf{CN}(\sigma a)}(a(\!|Y,\bar{z}|\!)) \equiv \sigma a & & \text{if } \mathsf{BV}(\sigma a) \subseteq \bar{z} \text{ and } \mathsf{CN}(\sigma a) = Y \\
\sigma^{U,W}_{U\cup\{x\}}(x := \theta) \equiv x := \sigma^{U\cup W}(\theta) & & \\
\sigma^{U,W}_{U\cup\{x\}}(x := *) \equiv x := * & & \\
\sigma^{U,W}_{U}(?\chi) \equiv\ ?\sigma^{U\cup W}(\chi) & & \\
\sigma^{U,W}_{Z}(\{x' = \theta \,\&\, \chi\}) \equiv \{x' = \sigma^{U\cup W}(\theta) \,\&\, \sigma^{U\cup W}(\chi)\} & & \text{with } Z = U \cup \{x, x', \mu, \mu'\} \\
\sigma^{U,W}_{U\cup\{\mathrm{ch},h\}}(\mathrm{ch}(h)!\theta) \equiv \mathrm{ch}(h)!\sigma^{U\cup W}(\theta) & & \\
\sigma^{U,W}_{U\cup\{\mathrm{ch},h,x\}}(\mathrm{ch}(h)?x) \equiv \mathrm{ch}(h)?x & & \\
\sigma^{U,W}_{Z_1\cup Z_2}(\alpha \cup \beta) \equiv \sigma^{U,W}_{Z_1}(\alpha) \cup \sigma^{U,W}_{Z_2}(\beta) & & \\
\sigma^{U,W}_{Z_2}(\alpha;\beta) \equiv \sigma^{U,W}_{Z_1}(\alpha); \sigma^{Z_1,W}_{Z_2}(\beta) & & \\
\sigma^{U,W}_{Z}(\alpha^*) \equiv (\sigma^{Z,W}_{Z}(\alpha))^* & & \text{when } \sigma^{U,W}_{Z}(\alpha) \text{ is defined} \\
\sigma^{U,W}_{Z_1\cup Z_2}(\alpha \parallel \beta) \equiv \sigma^{U,W_{U,\beta}}_{Z_1}(\alpha) \parallel \sigma^{U,W_{U,\alpha}}_{Z_2}(\beta) & &
\end{array}
$$

Fig. 2. Application of uniform substitution for taboo U and parallel context W, where $W_{U,\gamma} \equiv W \cup (\mathsf{BV}(\sigma^{U,W}(\gamma) \setminus (\{\mu,\mu'\} \cup V_{\mathcal{T}}))$ for any program γ, and $e \downarrow Y$ for term e is recursive push down of projection $\downarrow Y$, where $p(Y_0, e) \downarrow Y \equiv p(Y_0 \cap Y, e)$.

The side condition $(\mathsf{FV}(\sigma f(\cdot)) \cup \mathsf{CN}(\sigma f(\cdot))) \cap U = \emptyset$ implements locally that the replacement for f must not introduce free parameters that are tabooed by U (B I). The substitution $\{\cdot \mapsto \sigma^U(e \downarrow Y)\}^{\emptyset}$ is responsible for the argument e,[3] where $\emptyset$ suffices as the taboo U is already checked on $e \downarrow Y$. By the projection, $e\downarrow Y$ only depends on channels Y. Quantification $\forall z$ taboos the bound variable z. Program α in a box or ac-box has an empty parallel context $\emptyset$.

The substitution $\sigma_Z^{U,W}(\alpha)$ computes the output taboo Z by adding the written variables and channels of program α to U, e.g., real variable x for assignment $x := \theta$ and for receiving $\mathrm{ch}(h)?x$ additionally channel ch and trace variable h. The output taboo Z is passed to ac-formulas and postconditions of boxes and ac-boxes for recursive checks for clashes w.r.t. (B I). Crucially for soundness, Lemma 13 below proves that $\sigma_Z^{U,W}(\cdot)$ correctly computes the output taboo Z.

The taboo $U \cup W$ passed to nested expressions contains the parallel context W to prevent free variables in replacements of function and predicate symbols that are bound in parallel. This prepares the substitution process to preserve the syntax restrictions for parallel composition from previous work [6].[4] Substitution for evolution $\{x' = \theta \,\&\, \chi\}$ considers that the global time μ, μ' is always implicitly bound regardless of whether it occurs in x, x'. The fixpoint notation $\sigma_Z^{Z,W}(\alpha)$ for the replacement of repetition α^* ensures that the output taboo of the first iteration is tabooed in the subsequent iterations [32]. Computing the parallel context of α and β in case $\alpha \parallel \beta$ requires one additional pass for both subprograms because what they potentially bind after substitution adds to the parallel context of the respective other subprogram.

Lemma 13 (Correct output taboo). *Application $\sigma_Z^{U,W}(\alpha)$ of uniform substitution retains input taboo U and correctly adds the bound variables and written channels of program α, i.e., $Z \supseteq U \cup \mathsf{BV}(\sigma_Z^{U,W}(\alpha)) \cup \mathsf{CN}(\sigma_Z^{U,W}(\alpha))$.*

The side condition of $\sigma_Z^{U,W}(a(\!|Y, \bar{z}|\!))$ maintains local abstraction of subprograms (B II) because the replacement cannot bind more than $a(\!|Y, \bar{z}|\!)$, thus cannot bind variables and channels of an abstraction that is independent of $a(\!|Y, \bar{z}|\!)$. This also preserves state-disjointness (well-formedness) of parallel programs.

3.1 Semantic Effect of Uniform Substitution

The key ingredients for proving soundness of uniform substitution are Lemma 16 and 17 below. They prove that the effect of the syntactic transformation applied by uniform substitution can be equally mimicked by semantically modifying the interpretation of function and predicate symbols, and program constants. This adjoint interpretation $\sigma_w^* I$ for interpretation I and state w changes how symbols are interpreted according to their syntactic replacements in the substitution σ.

[3] Extension to vectorial arguments is straightforward.

[4] For $\alpha \parallel \beta$, the restriction is $(\mathsf{V}(\alpha) \cap \mathsf{BV}(\beta)) \cup (\mathsf{V}(\beta) \cap \mathsf{BV}(\alpha)) \subseteq \{\mu, \mu'\} \cup V_{\mathcal{T}}$ [6]. However, in this paper, programs obey a less restrictive syntax for simplicity.

Definition 14 (Adjoint substitution). *For interpretation I and state w, the* adjoint interpretation $\sigma^*_w I$ *changes the meaning of function and predicate symbols, and program constants according to the substitution σ evaluated in state w:*

$$\sigma^*_w I(f^{\mathbb{M}} : \mathbb{M}_{\text{arg}}) : \mathbb{M}_{\text{arg}} \to \mathbb{M}; d \mapsto I^d_{\cdot} w[\![\sigma f(\cdot)]\!] \quad \text{where } \mathbb{M}, \mathbb{M}_{\text{arg}} \in \{\mathbb{R}, \mathbb{N}, \Omega, \mathcal{T}\}$$
$$\sigma^*_w I(p : \mathbb{M}_{\text{arg}}) = \{d \in \mathbb{M}_{\text{arg}} \mid I^d_{\cdot} w \vDash \sigma p(\cdot)\} \quad \text{where } \mathbb{M}_{\text{arg}} \in \{\mathbb{R}, \mathbb{N}, \Omega, \mathcal{T}\}$$
$$\sigma^*_w I(a(\!|Y, \bar{z}|\!)) = I[\![\sigma a]\!]$$

We follow the observation for dGL [32] that the more liberal one-pass substitution requires stronger coincidence between the substitution and the adjoint on neighborhoods of the original state. Where the dGL soundness proof has succeeded by a neighborhood semantics of state on taboos, the dL_{CHP} proof succeeds with a generalization to a neighborhood semantics of state and communication on taboos. The neighborhood of a state consists of its variations:

Definition 15 (Variation). *For a set $U \subseteq V \cup \Omega$, a state v is a U-*variation of state w *if v and w only differ on variables or projections onto channels in U, i.e., $v \downarrow (U^{\complement} \cap \Omega) = w \downarrow (U^{\complement} \cap \Omega)$ on $U^{\complement} \cap V$.*

The proofs of Lemma 16 and 17 follow a lexicographic induction on the structure of substitution, and term, formula, or program. In Lemma 17, the induction is mutual for formulas and programs.

Lemma 16 (Semantic uniform substitution). *The term e evaluates equally over U-variations under uniform substitution σ^U and adjoint interpretation $\sigma^*_w I$, i.e., $Iv[\![\sigma^U(e)]\!] = \sigma^*_w Iv[\![e]\!]$ for all U-variations v of w.*

Lemma 17 (Semantic uniform substitution). *The formula ϕ and the program α have equal truth value and semantics, respectively, over U-variations under uniform substitution σ^U and adjoint interpretation $\sigma^*_w I$, i.e.,*

1. *for all U-variations v of w: $Iv \vDash \sigma^U(\phi)$ iff $\sigma^*_w Iv \vDash \phi$*
2. *for all $(U \cup W)$-variations v of w: $(v, \tau, o) \in I[\![\sigma^{U,W}_Z(\alpha)]\!]$ iff $(v, \tau, o) \in \sigma^*_w I[\![\alpha]\!]$*

3.2 Uniform Substitution Proof Rule

The proof rule US for uniform substitution is the single point of truth for the sound instantiation of axioms (plus renaming of bound variables [30] and written channels, e.g., $[x := \theta]\psi(x)$ to $[y := \theta]\psi(y)$ and $[\text{ch}(h)?x]\psi(\text{ch})$ to $[\text{dh}(h)?x]\psi(\text{dh})$. Soundness of the rule, i.e., that validity of its premise implies validity of the conclusion, immediately follows from Lemma 17. Since the substitution process starts with no taboos, $\sigma(\phi)$ is short for $\sigma^{\emptyset}(\phi)$. If the substitution clashes, i.e., $\sigma^{\emptyset}(\phi)$ is not defined, then rule US is not applicable.

Theorem 18 (US is sound). *The proof rule US is sound.*

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

Unlike dL [30] and dGL [32], dL_{CHP} has a context-sensitive syntax for programs and formulas (see Definition 2 and Definition 4). By Proposition 19, uniform substitution, however, preserves syntactic well-formedness. Since all axioms in Sect. 4 will be well-formed, only well-formed formulas can be derived in dL_{CHP}.

Proposition 19 (US preserves well-formedness). *The result $\sigma^U(\phi)$ (if defined) of applying uniform substitution to a well-formed formula ϕ is well-formed.*

4 Axiomatic Proof Calculus

Figure 3 presents a sound proof calculus for dL_{CHP}. The significant difference to dL_{CHP}'s schematic calculus [6] is that it completely abandons soundness-critical side conditions, internalizing them syntactically in the axioms. Only axiom $[]_{\text{WA}}$ was adjusted to obtain a symbolic representation and an ac-version K_{AC} of modal modus ponens is included. Now, distribution of ac-boxes over conjuncts $[]_{\text{AC}}\wedge$ and ac-monotonicity $\text{M}[\cdot]_{\text{AC}}$ derive from K_{AC}, thus are dropped. Except for the small changes soundness is inherited from the schematic axioms [6].

Algebraic laws for reasoning about traces [6] can be easily adapted to uniform substitution as well [7]. Decidable first-order real arithmetic [41] and Presburger arithmetic [34] have corresponding oracle proof rules [6].

Remark 20. To obtain a truly finite list of axioms from Fig. 3, symbolic (co)finite sets can be finitely axiomatized as a boolean algebra together with extensionality, which can be unrolled to a finite disjunction for (co)finite sets [7].

Parallel Composition. The parallel injection axiom $[\|\_]_{\text{AC}}$ in Fig. 3 decomposes parallel CHPs by local abstraction (B II). Unlike dL_{CHP}'s [6] and Hoare-style [46,47] schematic calculi for ac-reasoning, axiom $[\|\,\_]_{\text{AC}}$ internalizes the noninterference property [6, Def. 7] that determines valid instances of formula

$$[\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi \rightarrow [\alpha \parallel \beta]_{\{\mathsf{A},\mathsf{C}\}}\psi \tag{1}$$

purely syntactically. To focus on noninterference, $a(\!|Y_a, \bar{z}_a|\!) \parallel_{\text{wf}} b(\!|Y_b, \bar{z}_b|\!)$ abbreviates well-formed parallel composition $a(\!|Y_a, \bar{z}_a|\!) \parallel b(\!|Y_b, (\bar{z}_b \cap \bar{z}_a^{\mathsf{C}}) \cup \{\mu, \mu'\} \cup V_{\mathcal{T}}|\!)$ using operator $\|_{\text{wf}}$ for program constants $a(\!|Y_a, \bar{z}_a|\!)$, $b(\!|Y_b, \bar{z}_b|\!)$. This notation ensures disjoint parallel state except for the global time μ, μ' and recorder variables $V_{\mathcal{T}}$.

Intuitively, axiom $[\|\_]_{\text{AC}}$ restricts β in Eq. (1) such that α overapproximates the behavior of $\alpha \parallel \beta$ influencing A, C, or ψ. For this purpose, noninterference internalized in $b(\!|Y_b \cap (Y^{\mathsf{C}} \cup Y_a), \bar{z}^{\mathsf{C}}|\!)$ forbids b to bind variables $\bar{z}$ that are free in the postcondition $p(Y, \bar{z})$, and Y^{C} forbids b to bind channels Y (except for channels Y_a written by a because joint parallel communication can already be observed from a, too). Moreover, parallel programs always agree on the global time μ, μ' and the communication recorded by trace variables $V_{\mathcal{T}}$. Therefore, the operator $\|_{\text{wf}}$ explicitly allows their sharing even if $\bar{z}^{\mathsf{C}}$ disallows it. Note that Y_a and Y, and $\bar{z}_a$ and $\bar{z}$ may overlap but can also be disjoint.

$[:=]$ $[x := g^{\mathbb{R}}]p(x) \leftrightarrow p(g^{\mathbb{R}})^a$ $[;]_{\mathsf{AC}}$ $[a;b]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow [a]_{\{\mathrm{R},\mathrm{Q}\}}[b]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P}$

$[:*]$ $[x := *]p(x) \leftrightarrow \forall x\, p(x)$ $[\cup]_{\mathsf{AC}}$ $[a \cup b]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow [a]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \wedge [b]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P}$

$[?]$ $[?q_{\mathbb{R}}]p \leftrightarrow (q_{\mathbb{R}} \rightarrow p)^a$ $[^*]_{\mathsf{AC}}$ $[a^*]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow [a^0]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \wedge [a]_{\{\mathrm{R},\mathrm{Q}\}}[a^*]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P}^b$

$[]_{\top,\top}$ $[a]\mathrm{P} \leftrightarrow [a]_{\{\mathrm{T},\mathrm{T}\}}\mathrm{P}$ $[]_{\mathsf{WA}}$ $[a]_{\{\mathrm{T},\mathrm{W_A}\}}\mathsf{T} \wedge [a]_{\{\mathrm{R}_1\wedge\mathrm{R}_2,\mathrm{Q}_1\wedge\mathrm{Q}_2\}}\mathrm{P} \rightarrow [a]_{\{\mathrm{R},\mathrm{Q}_1\wedge\mathrm{Q}_2\}}\mathrm{P}^c$

$[\|\_]_{\mathsf{AC}}$ $[a(\!|Y_a, \bar{z}_a|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z}) \rightarrow [a(\!|Y_a, \bar{z}_a|\!)\ \|_{\mathrm{wf}}\ b(\!|Y_b \cap (Y^{\complement} \cup Y_a), \bar{z}^{\complement}|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z})^d$

$[\mu]$ $[\{\bar{x}' = g^{\mathbb{R}}(\bar{x},\mu)\ \&\ q_{\mathbb{R}}(\bar{x},\mu)\}]p(\bar{x},\mu) \leftrightarrow [\{\mu' = 1, \bar{x}' = g^{\mathbb{R}}(\bar{x},\mu)\ \&\ q_{\mathbb{R}}(\bar{x},\mu)\}]p(\bar{x},\mu)^a$

$[\mathrm{ch}!]$ $[\mathrm{ch}(h)!g^{\mathbb{R}}]p(\mathrm{ch},h) \leftrightarrow \forall h_0\, \big(h_0 = h \cdot \langle \mathrm{ch}, g^{\mathbb{R}}, \mu\rangle \rightarrow p(\mathrm{ch},h_0)\big)$

$[\mathrm{ch}!]_{\mathsf{AC}}$ $[\mathrm{ch}(h)!g^{\mathbb{R}}]_{\{\widehat{r},\widehat{q}\}}\widehat{p} \leftrightarrow \widehat{q} \wedge \big(\widehat{r} \rightarrow [\mathrm{ch}(h)!g^{\mathbb{R}}]\big(\widehat{q} \wedge (\widehat{r} \rightarrow \widehat{p})\big)\big)$

$[\mathrm{ch}?]_{\mathsf{AC}}$ $[\mathrm{ch}(h)?x]_{\{\widehat{r},\widehat{q}\}}p(\mathrm{ch},h,x) \leftrightarrow [x := *][\mathrm{ch}(h)!x]_{\{\widehat{r},\widehat{q}\}}p(\mathrm{ch},h,x)$

$[\epsilon]_{\mathsf{AC}}$ $[a(\!|\emptyset, V_{\mathbb{R}}|\!)]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow \mathrm{Q} \wedge (\mathrm{R} \rightarrow [a(\!|\emptyset, V_{\mathbb{R}}|\!)]\mathrm{P})$

$\mathrm{W}[]_{\mathsf{AC}}$ $[a]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow \mathrm{Q} \wedge [a]_{\{\mathrm{R},\mathrm{Q}\}}(\mathrm{Q} \wedge (\mathrm{R} \rightarrow \mathrm{P}))$

I_{AC} $[a^*]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow [a^0]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \wedge [a^*]_{\{\mathrm{R},\mathrm{T}\}}(\mathrm{P} \rightarrow [a]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P})$

K_{AC} $[a]_{\{\mathrm{R},\mathrm{Q}_1\rightarrow\mathrm{Q}_2\}}(\mathrm{P}_1 \rightarrow \mathrm{P}_2) \rightarrow ([a]_{\{\mathrm{R},\mathrm{Q}_1\}}\mathrm{P}_1 \rightarrow [a]_{\{\mathrm{R},\mathrm{Q}_2\}}\mathrm{P}_2)$

$\mathrm{MP}\ \dfrac{p \rightarrow q \quad p}{q}$

$\mathrm{G}_{\mathsf{AC}}\ \dfrac{\mathrm{Q} \wedge \mathrm{P}}{[a]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P}}$

$\forall\ \dfrac{p(x)}{\forall x\, p(x)}$

$\mathrm{CE}\ \dfrac{\mathrm{P}_1 \leftrightarrow \mathrm{P}_2}{C(\mathrm{P}_1) \leftrightarrow C(\mathrm{P}_2)}$

$\mathrm{P}_j \equiv p_j(Y,\bar{z})$, and $\mathrm{R}_j \equiv r_j(Y,\bar{h})$, and $\mathrm{Q}_j \equiv q_j(Y,\bar{h})$, and $\widehat{\chi} \equiv \chi(\mathrm{ch},h)$, where j may be blank, and $Y \subseteq \Omega$, $\bar{z} \subseteq V_{\mathbb{R}} \cup V_{\mathcal{T}}$, and $\bar{h} \subseteq V_{\mathcal{T}}$ are (co)finite.

[a] Replacements for function symbol $g^{\mathbb{R}}$ and predicate symbol $q_{\mathbb{R}}$ are restricted to polynomials in $V_{\mathbb{R}}$ and first-order real arithmetic, respectively.
[b] Recall that $[\alpha^0]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \leftrightarrow \mathrm{Q} \wedge (\mathrm{R} \rightarrow \mathrm{P})$ by $[\epsilon]_{\mathsf{AC}}$ and $[?]$ since $\alpha^0 \equiv\ ?\mathsf{T}$.
[c] $\mathrm{W_A}$ is the compositionality condition $(\mathrm{R} \wedge \mathrm{Q}_1 \rightarrow \mathrm{R}_2) \wedge (\mathrm{R} \wedge \mathrm{Q}_2 \rightarrow \mathrm{R}_1)$.
[d] The operator $\|_{\mathrm{wf}}$ abbreviates well-formed parallel composition (see above).

Fig. 3. $\mathsf{dL}_{\mathrm{CHP}}$ proof calculus

Despite its asymmetric shape, axiom $[\|\_]_{\mathsf{AC}}$ decomposes $[\alpha\|\beta](\phi\wedge\psi)$ into $[\alpha]\phi$ and $[\beta]\psi$ (if they mutually do not interfere) via independent proofs for $[\alpha\|\beta]\phi$ and $[\alpha\|\beta]\psi$, which drop either α or β by $[\|\_]_{\mathsf{AC}}$ modulo commutativity.

Axiom System. For each program statement, there is either a dynamic or an ac-axiom because the respective other version derives by axiom $[]_{\top,\top}$ or $[\epsilon]_{\mathsf{AC}}$. Axioms $[:=]$, $[:*]$, and $[?]$ are as in dL [30]. Axioms $[;]_{\mathsf{AC}}$, $[\cup]_{\mathsf{AC}}$, and $[^*]_{\mathsf{AC}}$ for decomposition, and I_{AC} for induction carefully generalize their versions in differential [30] dynamic [14] logic to ac-reasoning. Sending is handled step-wise via flattening the assumption-commitments by axiom $[\mathrm{ch}!]_{\mathsf{AC}}$ and axiom $[\mathrm{ch}!]$ that executes the effect onto the recorder h. The duality $[\mathrm{ch}?]_{\mathsf{AC}}$ turns receiving into arbitrary sending, which only synchronizes if it agrees with the parallel context on the value. Usage of axiom $\mathrm{W}[]_{\mathsf{AC}}$ is for convenience. Axiom $[\mu]$ materializes the flow of global time μ such that dL's axiomatization of continuous evolution [30] gets applicable, which requires ODE shape $\bar{x}' = f^{\mathbb{R}}(\bar{x})$. The axiomatic proof

rules G_{AC}, MP, $\forall$, and CE are an ac-version of Gödels generalization rule, modus ponens, quantifier elimination, and contextual equivalence, respectively.

The axiom $[]_{\mathrm{W}\mathsf{A}}$ can weaken assumptions. Its slight change compared to $\mathsf{dL}_{\mathrm{CHP}}$'s schematic calculus [6] exploits that the compositionality condition W_{A} is only required for a's reachable worlds. Interestingly, $\mathsf{dL}_{\mathrm{CHP}}$'s monotonicity rule $\mathrm{M}[\cdot]_{\mathsf{AC}}$ [6] does not derive from modal modus ponens K_{AC} and Gödel generalization G_{AC} in analogy to dL [30] but needs $\mathrm{W}[]_{\mathsf{AC}}$ handling monotonicity of assumptions, which does not fit into G_{AC} because necessitating the assumption in G_{AC} would render the derivation of $[\alpha]_{\{\bot,\mathsf{T}\}}\mathsf{T}$ by G_{AC} impossible.

Axioms using postcondition $\mathrm{P} \equiv p(Y,\bar{z})$, e.g., in $[;]_{\mathsf{AC}}$, allow any replacement of P since accessed channels $Y \subseteq \Omega$ and free variables $\bar{z} \subseteq V_{\mathbb{R}} \cup V_{\mathcal{T}}$ can be arbitrary. Replacements of assumptions $\mathrm{R} \equiv r(Y,\bar{h})$ and commitments $\mathrm{Q} \equiv q(Y,\bar{h})$ can instead only mention trace variables $\bar{h} \subseteq V_{\mathcal{T}}$ bound in their context. This reflects that trace variables are the only interface between the program α and the ac-formulas A and C in an ac-box $[\alpha]_{\{\mathsf{A},\mathsf{C}\}}\psi$ (well-formedness).

Theorem 21 (Soundness). *The proof calculus for* $\mathsf{dL}_{\mathrm{CHP}}$ *presented in Fig. 3 is sound as an instantiation of the schematic calculus [6].*

Clashes. Clashes sort out unsound instantiations of axioms. Unlike in dL and dGL [30,32] whose clashes are solely due to tabooed variables in terms and formulas, clashes in $\mathsf{dL}_{\mathrm{CHP}}$ can also be due to tabooed channels, and even due to taboos in programs. For example, the substitution $\sigma = \{a \mapsto \mathrm{gh}(h)!1, b \mapsto \mathrm{ch}(h)!2, p \mapsto \psi, r \mapsto \mathsf{T}, q \mapsto \mathsf{T}\}$ with $\psi \equiv |h \downarrow \mathrm{ch}| > 0 \wedge |h \downarrow \mathrm{dh}| > 0 \wedge y < 0$ clashes below, where $Y = \{\mathrm{ch}, \mathrm{dh}\}$, and $\bar{z} \equiv h, y$, and $\mathrm{R} \equiv r(Y)$, and $\mathrm{Q} \equiv q(Y)$. Writing channel ch in the replacement for b would break the local abstraction of a as ch is accessed in ψ but not written in the replacement for a, thus the clash indeed sorts out an unsound instantiation.

$$\frac{[a(\!|\{\mathrm{gh}\},h|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z}) \to [a(\!|\{\mathrm{gh}\},h|\!) \parallel_{\mathrm{wf}} b(\!|\{\mathrm{ch}\}\cap(Y^{\complement}\cup\{\mathrm{gh}\}),\bar{z}^{\complement}|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z})}{[\mathrm{gh}(h)!1]_{\{\mathsf{T},\mathsf{T}\}}\psi \to [\mathrm{gh}(h)!1 \parallel \mathrm{ch}(h)!2]_{\{\mathsf{T},\mathsf{T}\}}\psi}\ \lightning\ \text{clash}$$

In contrast, $\sigma = \{a \mapsto \mathrm{ch}(h)?x; \mathrm{gh}(h)!1, b \mapsto \mathrm{ch}(h)!2, p \mapsto \psi, r \mapsto \mathsf{T}, q \mapsto \mathsf{T}\}$ does not clash below, where $Y = \{\mathrm{ch}, \mathrm{dh}\}$, and $Y_a = \{\mathrm{ch}, \mathrm{gh}\}$, and other abbreviations are as above, because $\mathrm{ch} \in Y^{\complement} \cup Y_a = \{\mathrm{dh}\}^{\complement}$. Intuitively, the ch-communication of b remains observable after dropping b from the parallel composition as it is joint with a.

$$\dfrac{\dfrac{*}{[a(\!|Y_a,h,x|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z}) \to [a(\!|Y_a,h,x|\!) \parallel_{\mathrm{wf}} b(\!|\{\mathrm{ch}\}\cap(Y^{\complement}\cup Y_a),\bar{z}^{\complement}|\!)]_{\{\mathrm{R},\mathrm{Q}\}}p(Y,\bar{z})}\ [\|\_]_{\mathsf{AC}}}{[\mathrm{ch}(h)?x;\mathrm{gh}(h)!1]_{\{\mathsf{T},\mathsf{T}\}}\psi \to [(\mathrm{ch}(h)?x;\mathrm{gh}(h)!1) \parallel \mathrm{ch}(h)!2]_{\{\mathsf{T},\mathsf{T}\}}\psi}\ \mathrm{US}$$

Also note that by the operator $\parallel_{\mathrm{wf}}$ for well-formed parallel composition, the recorder variable h can be shared without causing a clash above. However, clashes prevent instantiation that would violate syntactic well-formedness of programs (Definition 2) by binding the same state variable in parallel:

$$\frac{[a(\!|\emptyset, x|\!)]_{\{r,q\}}p(x,y) \to [a(\!|\emptyset, x|\!) \parallel_{\text{wf}} b(\!|\emptyset, \{x,y\}^{\complement}|\!)]_{\{r,q\}}p(x,y)}{[x := y]_{\{\mathsf{T},\mathsf{T}\}}y = x \to [x := y \parallel x := 0]_{\{\mathsf{T},\mathsf{T}\}}y = x}\ \lightning\,\text{clash}$$

Well-formedness of programs and formulas is ensured in the axioms by well-formed parallel composition $\parallel_{\text{wf}}$ and limitation to trace variables $\bar{h}$ in $\mathrm{R}_j \equiv r_j(Y, \bar{h})$ and $\mathrm{Q}_j \equiv q_j(Y, \bar{h})$ in ac-boxes $[\alpha]_{\{\mathrm{R}_j,\mathrm{Q}_j\}}\psi$ in Fig. 3, respectively. By Proposition 19, uniform substitution always preserves well-formedness.

Example 22. The proof tree below decomposes safety (Example 5) of cruise control (Example 3) into safety ① of controller ct and branch ② to be continued to safety of the vehicle ve. The lower subproof introduces the ac-formulas

$$\mathsf{A} \equiv \mathsf{C} \equiv \big(|h \downarrow \mathrm{tar}| > 0 \to 0 \le \mathtt{val}(h \downarrow \mathrm{tar}) \le V\big)$$

using axiom $[]_{\text{WA}}$ to abstract from the communication between ct and ve. The upper subproof uses the parallel injection axiom $[\|\,\_]_{\text{AC}}$ to drop ve. Uniform substitution US does not clash as the commitment C only refers to joint communication of ct and ve. Other applications of US (e.g., for $[]_{\text{WA}}$) are omitted. Rule Prop denotes propositional reasoning. Abbreviations are as follows: $\alpha \equiv a(\!|\mathrm{tar}, v^{\text{tr}}_{\text{ct}}, t, t', \mu, \mu', h|\!)$, $\mathrm{R} \equiv r(\mathrm{tar}, h)$, $\mathrm{Q} \equiv q(\mathrm{tar}, h)$, $\mathrm{P} \equiv p(\mathrm{tar})$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \to [\mathsf{ct}^*]_{\{\mathsf{T},\mathsf{C}\}}\mathsf{T}}\,①}{\varphi \to [\mathsf{ct}^*]_{\{\mathsf{T},\mathsf{C}\}}\mathsf{T}}\quad \dfrac{\dfrac{*}{[\alpha]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P} \to [\alpha \parallel_{\text{wf}} b(\!|\mathrm{tar}, v^{\text{tr}}_{\text{ve}}, a_{\text{ve}}, t_0, v_{\text{ve}}, v'_{\text{ve}}|\!)]_{\{\mathrm{R},\mathrm{Q}\}}\mathrm{P}}\,[\|\_]_{\text{AC}}}{[\mathsf{ct}^*]_{\{\mathsf{T},\mathsf{C}\}}\mathsf{T} \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T},\mathsf{C}\}}\mathsf{T}}\,\text{US}}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T},\mathsf{C}\}}\mathsf{T}}\,\text{MP, CE}}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T}\wedge\mathsf{A},\mathsf{C}\}}\mathsf{T}}\,\text{M}[\cdot]_{\text{AC}}\quad \dfrac{}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T}\wedge\mathsf{A},\mathsf{T}\}}\psi_{\text{safe}}}\,②}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T}\wedge\mathsf{A},\mathsf{C}\}}\mathsf{T} \wedge [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T}\wedge\mathsf{A},\mathsf{T}\}}\psi_{\text{safe}}}\,\wedge\text{R}\quad \dfrac{\dfrac{*}{(\mathsf{C} \to \mathsf{A}) \wedge \mathsf{T}}\,\text{Prop}}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T},\mathsf{C}\to\mathsf{A}\}}\mathsf{T}}\,\text{G}_{\text{AC}}}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T},\mathsf{C}\to\mathsf{A}\}}\mathsf{T} \wedge [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T}\wedge\mathsf{A},\mathsf{C}\wedge\mathsf{T}\}}(\mathsf{T} \wedge \psi_{\text{safe}})}\,[]_{\text{AC}}\wedge}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]_{\{\mathsf{T},\mathsf{C}\wedge\mathsf{T}\}}(\mathsf{T} \wedge \psi_{\text{safe}})}\,\wedge\text{R}}{\varphi \to [\mathsf{ct}^* \| \mathsf{ve}^*]\psi_{\text{safe}}}\,[]_{\text{WA}}\quad []_{\mathsf{T},\mathsf{T}},\ \text{M}[\cdot]_{\text{AC}}$$

5 Related Work

Uniform substitution for differential dynamic logic dL [30] generalizes Church's uniform substitution for first-order logic [8, §35, 40]. Unlike the lifting from dL to differential game logic dGL [31], dL_{CHP} generalizes into the complementary direction of communication and parallelism. Unlike schematic calculi [2,19,27,44,46], whose treacherous schematic simplicity relies on encoding all subtlety of parallel systems in significant soundness-critical side conditions, our development builds upon a minimalistic non-schematic parallel injection axiom *and* sound instantiation encapsulated in uniform substitution. This provides a new, more

atomic and more modular understanding of parallel systems overcoming the root cause for large soundness-critical prover kernels [5,9,12,16,18,36]. Usage of uniform substitution reduced the kernel of the theorem prover KeYmaera from 105 kLOC to 2 kLOC in KeYmaera X [23]. We expect dL_{CHP}'s integration into KeYmaera X to stay in the same order of magnitude.

To the best of our knowledge, assumption-commitment reasoning [22,46][5] has no tool support, which might be due to vast implementation effort. The latter can be underpinned by analogy with tools [5,9,16,18,36] for verification of shared-variables concurrency, some of which use rely-guarantee reasoning [36,39]. Unlike uniform substitution for dL_{CHP} that enables a straightforward implementation of a small prover kernel, they all rely on large soundness-critical code bases. Unlike refinement checking for CSP [12] and discrete-time CSP [4], dL_{CHP} supports safety properties of dense-time hybrid systems. Contrary to our goal of small prover kernels, implementations of model checkers [12] are inherently large.

Beyond embeddings of concurrency reasoning for discrete systems into proof assistants [3,25,26,38], dL_{CHP} can verify parallel hybrid systems synchronizing in shared global time. The latter imposes even more complicated binding structures than parallel or hybrid systems alone but dL_{CHP}'s uniform substitution calculus continues to manage them in a modular way.

The recent tool HHLPy [37] for hybrid CSP (HCSP) [17] is limited to the sequential fragment. Unlike extending HHLPy to parallelism, which would require extensive soundness-critical side conditions and a treatment of the duration calculus, integrating dL_{CHP} into KeYmaera X [11] boils down to adding a finite list of concrete object level formulas as axioms and only small changes to the uniform substitution process. In contrast to dL_{CHP}'s compositional parallel systems calculus [6], HCSP calculi [13,20,42] are non-compositional [6] as they either unroll exponentially many interleavings from the operational semantics [13,42] or can only decompose independent parallel components [20] causing limited ability to reason about complex systems. Former HCSP tools [43,45] only implement a non-compositional calculus [20] reinforcing the significance of our approach for managing parallel hybrid systems reasoning. Other hybrid process algebras defer to model checkers for reasoning [10,21,40]. Further discussion of dL_{CHP} is in [6].

6 Conclusion

This paper introduced a sound one-pass uniform substitution calculus for the dynamic logic of communicating hybrid programs dL_{CHP} thereby mastering the significant challenge of developing simple sound proof calculi for parallel hybrid systems with communication. Uniform substitution can separate even notoriously complicated binding structures from parallelism with communication in multi-dynamical logics into axioms and their instantiation. In the case of dL_{CHP},

[5] Assumption-commitment and rely-guarantee reasoning are specific patterns for message-passing and shared variables concurrency, respectively. The broader assume-guarantee principle has been used across diverse areas for various purposes.

this applies to channel access in predicates and the need for local abstraction of subprograms in parallel statements, and it even turns out that uniform substitution can maintain a context-sensitive syntax along the way. Thanks to uniform substitution, parallel systems reasoning reduces to multiple uses of an asymmetric parallel injection axiom.

Now, with uniform substitution a straightforward implementation of dL_{CHP} in KeYmaera X is only one step away.

Acknowledgments. This project was funded in part by the Deutsche Forschungsgemeinschaft (DFG) – 378803395 (ConVeY), an Alexander von Humboldt Professorship, and by the AFOSR under grant number FA9550-16-1-0288.

References

1. Apt, K.R., de Boer, F.S., Olderog, E.R.: Verification of Sequential and Concurrent Programs, 3rd edn. Springer, Heidelberg (2010). https://doi.org/10.1007/978-1-84882-745-5
2. Apt, K.R., Francez, N., de Roever, W.P.: A proof system for communicating sequential processes. ACM Trans. Program. Lang. Syst. **2**(3), 359–385 (1980). https://doi.org/10.1145/357103.357110
3. Armstrong, A., Gomes, V.B.F., Struth, G.: Algebras for program correctness in Isabelle/HOL. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M.E. (eds.) RAMICS 2014. LNCS, vol. 8428, pp. 49–64. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06251-8_4
4. Armstrong, P.J., Lowe, G., Ouaknine, J., Roscoe, B.: Model checking timed CSP. In: Voronkov, A., Korovina, M.V. (eds.) HOWARD-60: A Festschrift on the Occasion of Howard Barringer's 60th Birthday, EPiC Series in Computing, vol. 42, pp. 13–33. EasyChair (2014). https://doi.org/10.29007/6fqk
5. Blom, S., Darabi, S., Huisman, M., Oortwijn, W.: The VerCors tool set: verification of parallel and concurrent software. In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. LNCS, vol. 10510, pp. 102–110. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_7
6. Brieger, M., Mitsch, S., Platzer, A.: Dynamic logic of communicating hybrid programs. CoRR abs/2302.14546 (2023). https://doi.org/10.48550/arXiv.2302.14546
7. Brieger, M., Mitsch, S., Platzer, A.: Uniform substitution for dynamic logic with communicating hybrid programs. CoRR abs/2303.17333 (2023). https://doi.org/10.48550/arXiv.2303.17333
8. Church, A.: Introduction to Mathematical Logic. Princeton University Press, Princeton (1956)
9. Cohen, E., et al.: VCC: a practical system for verifying concurrent C. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) TPHOLs 2009. LNCS, vol. 5674, pp. 23–42. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03359-9_2
10. Cong, X., Yu, H., Xu, X.: Verification of hybrid chi model for cyber-physical systems using PHAVer. In: Barolli, L., You, I., Xhafa, F., Leu, F., Chen, H. (eds.) Proceedings of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 122–128. IEEE Computer Society (2013). https://doi.org/10.1109/IMIS.2013.29

11. Fulton, N., Mitsch, S., Quesel, J.-D., Völp, M., Platzer, A.: KeYmaera X: an axiomatic tactical theorem prover for hybrid systems. In: Felty, A.P., Middeldorp, A. (eds.) CADE 2015. LNCS (LNAI), vol. 9195, pp. 527–538. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21401-6_36
12. Gibson-Robinson, T., Armstrong, P., Boulgakov, A., Roscoe, A.W.: FDR3—a modern refinement checker for CSP. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 187–201. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_13
13. Guelev, D.P., Wang, S., Zhan, N.: Compositional Hoare-style reasoning about hybrid CSP in the duration calculus. In: Larsen, K.G., Sokolsky, O., Wang, J. (eds.) SETTA 2017. LNCS, vol. 10606, pp. 110–127. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69483-2_7
14. Harel, D. (ed.): First-Order Dynamic Logic. LNCS, vol. 68. Springer, Heidelberg (1979). https://doi.org/10.1007/3-540-09237-4
15. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**(8), 666–677 (1978). https://doi.org/10.1145/359576.359585
16. Jacobs, B., Smans, J., Philippaerts, P., Vogels, F., Penninckx, W., Piessens, F.: VeriFast: a powerful, sound, predictable, fast verifier for C and Java. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 41–55. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_4
17. Jifeng, H.: From CSP to hybrid systems In: A Classical Mind: Essays in Honour of C. A. R. Hoare, pp. 171–189. Prentice Hall International (1994)
18. Kirchner, F., Kosmatov, N., Prevosto, V., Signoles, J., Yakobowski, B.: Frama-C: a software analysis perspective. Formal Aspects Comput. **27**(3), 573–609 (2015). https://doi.org/10.1007/s00165-014-0326-7
19. Levin, G., Gries, D.: A proof technique for communicating sequential processes. Acta Informatica **15**(3), 281–302 (1981). https://doi.org/10.1007/BF00289266
20. Liu, J., et al.: A calculus for hybrid CSP. In: Ueda, K. (ed.) APLAS 2010. LNCS, vol. 6461, pp. 1–15. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17164-2_1
21. Man, K.L., Reniers, M.A., Cuijpers, P.J.L.: Case studies in the hybrid process algebra HyPA. Int. J. Softw. Eng. Knowl. Eng. **15**(2), 299–306 (2005). https://doi.org/10.1142/S0218194005002385
22. Misra, J., Chandy, K.M.: Proofs of networks of processes. IEEE Trans. Softw. Eng. **7**(4), 417–426 (1981). https://doi.org/10.1109/TSE.1981.230844
23. Mitsch, S., Platzer, A.: A retrospective on developing hybrid system provers in the KeYmaera family. In: Ahrendt, W., Beckert, B., Bubel, R., Hähnle, R., Ulbrich, M. (eds.) Deductive Software Verification: Future Perspectives. LNCS, vol. 12345, pp. 21–64. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64354-6_2
24. Müller, A., Mitsch, S., Retschitzegger, W., Schwinger, W., Platzer, A.: A component-based approach to hybrid systems safety verification. In: Ábrahám, E., Huisman, M. (eds.) IFM 2016. LNCS, vol. 9681, pp. 441–456. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33693-0_28
25. Nieto, L.P.: Verification of parallel programs with the Owicki-Gries and Rely-Guarantee methods in Isabelle/HOL. Ph.D. thesis, Technical University Munich, Germany (2002). http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/prensa_nieto.html
26. Nipkow, T., Nieto, L.P.: Owicki/Gries in Isabelle/HOL. In: Finance, J.-P. (ed.) FASE 1999. LNCS, vol. 1577, pp. 188–203. Springer, Heidelberg (1999). https://doi.org/10.1007/978-3-540-49020-3_13

27. Owicki, S.S., Gries, D.: An axiomatic proof technique for parallel programs I. Acta Informatica **6**, 319–340 (1976). https://doi.org/10.1007/BF00268134
28. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reas. **41**(2), 143–189 (2008). https://doi.org/10.1007/s10817-008-9103-8
29. Platzer, A.: Differential game logic. ACM Trans. Comput. Log. **17**(1), 1:1–1:51 (2015). https://doi.org/10.1145/2817824
30. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. J. Autom. Reas. **59**(2), 219–265 (2017). https://doi.org/10.1007/s10817-016-9385-1
31. Platzer, A.: Uniform substitution for differential game logic. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 211–227. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_15
32. Platzer, A.: Uniform substitution at one fell swoop. In: Fontaine, P. (ed.) CADE 2019. LNCS (LNAI), vol. 11716, pp. 425–441. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29436-6_25
33. Platzer, A., Tan, Y.K.: Differential equation invariance axiomatization. J. ACM **67**(1), 6:1–6:66 (2020). https://doi.org/10.1145/3380825
34. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du I congrès de Mathématiciens des Pays Slaves (1931)
35. de Roever, W.P., et al.: Concurrency Verification: Introduction to Compositional and Noncompositional Methods. Cambridge Tracts in Theoretical Computer Science, vol. 54. Cambridge University Press (2001)
36. Schellhorn, G., Bodenmüller, S., Bitterlich, M., Reif, W.: Software & system verification with KIV. In: Ahrendt, W., Beckert, B., Bubel, R., Johnsen, E.B. (eds.) The Logic of Software. A Tasting Menu of Formal Methods. LNCS, vol. 13360, pp. 408–436. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08166-8_20
37. Sheng, H., Bentkamp, A., Zhan, B.: HHLPy: practical verification of hybrid systems using Hoare logic. In: Chechik, M., Katoen, J., Leucker, M. (eds.) FM 2023. LNCS, vol. 14000, pp. 160–178. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-27481-7_11
38. Shi, L., Zhao, Y., Liu, Y., Sun, J., Dong, J.S., Qin, S.: A UTP semantics for communicating processes with shared variables and its formal encoding in PVS. Formal Aspects Comput. **30**(3–4), 351–380 (2018). https://doi.org/10.1007/s00165-018-0453-7
39. Smans, J., Vanoverberghe, D., Devriese, D., Jacobs, B., Piessens, F.: Shared boxes: rely-guarantee reasoning in VeriFast. Technical report, Katholieke Universiteit Leuven, Netherlands (2014). https://lirias.kuleuven.be/handle/123456789/456819
40. Song, H., Compton, K.J., Rounds, W.C.: SPHIN: a model checker for reconfigurable hybrid systems based on SPIN. In: Lazic, R., Nagarajan, R. (eds.) Proceedings of the 5th International Workshop Automated Verification of Critical Systems (AVoCS). ENTCS, vol. 145, pp. 167–183. Elsevier (2005). https://doi.org/10.1016/j.entcs.2005.10.011
41. Tarski, A.: A Decision Method for Elementary Algebra and Geometry, 2nd edn. University of California Press, Berkeley (1951). https://doi.org/10.1525/9780520348097
42. Wang, S., Zhan, N., Guelev, D.: An assume/guarantee based compositional calculus for hybrid CSP. In: Agrawal, M., Cooper, S.B., Li, A. (eds.) TAMC 2012. LNCS, vol. 7287, pp. 72–83. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29952-0_13

43. Wang, S., Zhan, N., Zou, L.: An improved HHL prover: an interactive theorem prover for hybrid systems. In: Butler, M., Conchon, S., Zaïdi, F. (eds.) ICFEM 2015. LNCS, vol. 9407, pp. 382–399. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25423-4_25
44. Xu, Q., de Roever, W.P., He, J.: The rely-guarantee method for verifying shared variable concurrent programs. Formal Aspects Comput. **9**(2), 149–174 (1997). https://doi.org/10.1007/BF01211617
45. Zou, L., et al.: Verifying Chinese train control system under a combined scenario by theorem proving. In: Cohen, E., Rybalchenko, A. (eds.) VSTTE 2013. LNCS, vol. 8164, pp. 262–280. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54108-7_14
46. Zwiers, J., de Bruin, A., de Roever, W.P.: A proof system for partial correctness of dynamic networks of processes. In: Clarke, E., Kozen, D. (eds.) Logic of Programs 1983. LNCS, vol. 164, pp. 513–527. Springer, Heidelberg (1984). https://doi.org/10.1007/3-540-12896-4_384
47. Zwiers, J., de Roever, W.P., van Emde Boas, P.: Compositionality and concurrent networks: soundness and completeness of a proofsystem. In: Brauer, W. (ed.) ICALP 1985. LNCS, vol. 194, pp. 509–519. Springer, Heidelberg (1985). https://doi.org/10.1007/BFb0015776