Modelling and Verifying Dependability of Hybrid Systems in HCSP

SHULING WANG[1], FLEMMING NIELSON[2], HANNE RIIS NIELSON[2] AND NAIJUN ZHAN[1]

[1] *State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China*
[2] *DTU Compute, Technical University of Denmark, Denmark*
Email: {wangsl,znj}@ios.ac.cn

Hybrid systems are dynamic systems with interacting discrete computation and continuous physical processes. They have become ubiquitous in our daily life, e.g., automotive, aerospace, medical systems and so on, particular, many of them are *safety-critical*. For these safety critical systems, it is demanded to guarantee not only the correctness (safety normally), i.e., its functions satisfying the given requirements, but also the dependability, i.e., the resistance to the unexpected behaviour from its environment, as many of them are deployed in highly uncertain environment, and the unexpected behaviour from the environment may result in a correct system malfunctioning. For example, the interactions between a controller and a physical processes are possibly realised via (wireless) communications. In case that the communications fail, the expected control from the controller may get lost and as a consequence the physical processes cannot behave as expected. In the literature, how to guarantee the correctness of hybrid systems has been extensively investigated, but there is little work on dependability of hybrid systems. To address this issue, this paper proposes a formal framework by extending HCSP, a formal modeling language for hybrid systems, for modelling and verification of hybrid systems in the presence of communication failure. Thus, safety and dependability of hybrid systems can be considered in the unform framework. Furthermore, by leveraging the expressivity and efficiency, we present two inference systems for the extension, and correspondingly implement two theorem provers in Isabelle/HOL. To illustrate our approach, we consider a case study on train control system originating from Chinese Train Control System, for which the two theorem provers are applied separately and the proof results are compared.

Keywords: Hybrid Systems, HCSP, HHL, Dependability, Communication Fault Tolerance, Safety Verification, Inference System

Received ; revised

1. INTRODUCTION

Hybrid systems, also known as cyber-physical systems, are dynamic systems with interacting discrete computation and continuous-time physical processes. Many hybrid systems in real applications, such as avionics, the traffic control systems, are required to conform to a higher safety standard. In hybrid systems, the physical processes evolve continuously with respect to time, and the discrete computers monitor and control the physical processes, to meet the safety requirement. The correct functioning of the control from the controllers is essential to guarantee the safety of hybrid systems. In the literature, this issue has been studied extensively through system verification or controller synthesis. For a hybrid system with a given controller, the verification of hybrid systems, i.e., whether under the given controller the hybrid system can achieve the desired safety requirement, can be done either through model-checking mainly depending on reachability computation [1, 2, 3, 4, 5, 6, 7] or through deductive way mainly depending on invariant generation [8, 9, 10, 11, 12]. As an alternative, given an incomplete hybrid system and a specification, one can synthesize a correct controller which ensures the given specification is satisfied by the system by restricting its behaviour. There are many approaches proposed for controller synthesis of hybrid systems, e.g., [13, 14, 15, 16, 17, 18].

However, a correct implementation of a hybrid system cannot guarantee that its functionality always works well as it may be deployed in a highly uncertainty environment, so it is impossible to predict all possible behaviours of the environment during the design, and some unexpected behaviour of the environment may make the system malfunctioning. Therefore, *dependability*, resistance to the unexpected behaviour of the environment, is another important issue in the design of safety-critical hybrid systems. But people have not paid enough attention on it so far. In addition, most of modern computer controlled systems are remotely controlled via (wireless) communications, thus communication failure is the most common unexpected behaviour of the environment of a safety-critical hybrid system. In this paper, we will try to address this issue by investigating when communication fails and thus the controllers fail to behave as expected, how to still guarantee that the functionality of the system works correctly. In another word, we aim to develop hybrid systems with communication fault tolerance.

A Motivating Example We illustrate our motivation by a train control system that originates from Chinese Train Control System (CTCS) [19]. The system is depicted in Fig. 1. It consists of three inter-communicating components: Train, Driver and on board vital computer (VC). We assume that the train owns arbitrarily long movement authority, within which the train is allowed to move only, and must conform to a safety requirement, i.e. the velocity must be non-negative and cannot exceed a maximum limit. The train acts as a continuous plant, and moves with a given acceleration; both the driver and the VC act as controllers, in such a way that, either of them observes the velocity of the train periodically, and then according to the safety requirement, computes the new acceleration for the train to follow in the next period. According to the specification of the system, the message from the VC always takes high priority over the one from the driver.

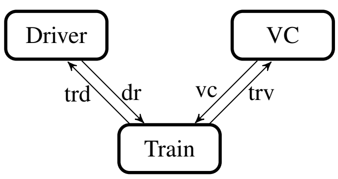

FIGURE 1. The structure of train control example

However, the expected monitoring and control from VC or driver may fail due to communication failure, that may be caused by many reasons, e.g. if the driver falls asleep, or if the VC gets malfunction. As a consequence, the train may get no response from any of them within a duration of time. The safety requirement of the train will then be violated easily. This poses the problem of how to build a safe hybrid system in the presence of losing control due to communication failure.

Comparison with The Conference Paper

In our previous work [20], we proposed a programming notation for formally modeling hybrid systems in the presence of communication failure. Meanwhile, for specifying and verifying such programs, we defined a deductive inference system for reasoning about whether the program satisfies the annotated safety property. In subsequence, an interactive theorem prover is implemented based on the inference system and has been applied to the train control example. As a direct application, a safe system for the example is built such that:

(F1) the error configurations where neither driver nor VC is available are not reachable;
(F2) the velocity of the train keeps always in the safe range, although in the presence of denial of control from the driver or the VC due to communication failure.

However, as seen from the result, the proof of the case study is done manually, rather lengthy about 900 lines of code. In this paper, we extend the previous work [20] in the following aspects:

- By leveraging the expressivity of the inference system and the efficiency of proof, we propose a more lightweight inference system. In [20], we use the interval temporal logic, i.e. duration calculus, to specify the interval property of the system. Here instead, we use first-order logic to specify the invariant property that holds for all reachable states of the system. Different from the interval property, the invariant property is independent on time. Thus we call them *time aware* and *time oblivious* inference systems respectively.
- We add a new general rule (**SHF**) in the time aware inference system, to strengthen the history formula for processes by adding the history for the termination point.
- We prove that the time oblivious inference system is an over-approximation of the time aware inference system: if a specification is proved by the time oblivious inference system, then an equivalent specification can be proved by the time aware inference system. Thus, the time aware inference system is more expressive than the time oblivious one.
- We implement a subsequent theorem prover based on the new inference system.
- We re-investigate the case study on train control system by applying the new prover. The same result stated above is obtained. The proof in the prover based on the time oblivious inference system allows more automation, and the length of the proof in it is reduced to about 300 lines of code. From this point of view, the time oblivious inference system is more efficient than the time aware one.

Although the time oblivious inference system is less expressive, it is more preferred in real applications because of its efficiency. Actually, it can obtain the same results as the time aware one in many cases.

Structure of This Paper

The rest of the paper is organized as follows. The related work is listed at the end of this section. In Section 2 and Section 3, we present the syntax and semantics for the formal modeling language. It is a combination of Hybrid CSP (HCSP) [21, 22], a process algebra based modeling language for describing hybrid systems, and the binders from Quality Calculus [23], a process calculus that allows one to take measures in case of unreliable communications. We call it bHCSP for simplicity. With the introducing of binders, bHCSP is capable of programming a safe system to be executed in an open environment that does not always live up to expectations.

In Section 4, we revisit the time aware inference system presented in [20] for reasoning about bHCSP. For each construct P, the specification is of the form $\{\varphi\}\ P\ \{\psi, HF\}$, where φ and ψ are the precondition and postcondition recording the initial and terminating states of P respectively, and *HF* the interval property held throughout the whole execution history of P. Different from [20], the inference system is strengthened by addition of a general rule, which adds the history of the single termination point into the history formula of a process. In Section 5, we present the time oblivious inference system for reasoning about bHCSP. The specification is of the same form, but the formula *HF* is in the form of first-order logic, specifying the invariant property of P. The comparison between the two inference systems is also given. In Section 6, the theorem provers implemented based on the inference systems are introduced. In Section 7, we apply the two theorem provers separately to the train control case study, and verify the properties (F1) and (F2) respectively. At last, we conclude the paper and address the future work.

Contribution to Development of Cyber-physical Systems

The development of cyber-physical systems is widely recognized as a highly complex and challenging task [24, 25]. To develop complex systems, model-based design is proposed and has been successfully applied in industry, especially for embedded systems and cyber-physical systems [24, 26]. In the model-based design approach, first of all, an abstract model of the system to be developed is built, and then extensive analysis and verification are conducted on the model, so that errors can be detected and corrected at early stages of design of the system. Afterwards, model transformations are applied iteratively to generate more concrete models at different levels of granularity, even to implemental code. This paper aims to study the first topic of model-based design of cyber-physical systems.

The first challenge faced is to have an expressive modelling language which can model all kinds of features of cyber-physical systems such as continuous and discrete dynamics, and the interaction between them. Meanwhile, to realize the correct control to the continuous process, it is extremely important to have a system with communication fault tolerance, i.e. it is still able to behave in a safe manner in case that the interactions between the controller and the plant fail due to communication failure. The bHCSP modelling language proposed in this work meet all the above requirements. Furthermore, the verification of the models is aided by the two inference systems and the corresponding theorem provers in Isabelle/HOL. As a consequence, the correctness of the system can be checked in the early design stage.

Related Work There have been numerous work on formal modeling and verification of hybrid systems. The most popular model for hybrid systems is *hybrid automata* [1, 27, 28]. For automata-based approaches, the verification of hybrid systems is reduced to computing reachable sets, which is conducted either by model checking [1] or by the decision procedure of Tarski algebra [2]. However, the verification based on reachability computation is not scalable and only applicable to some specific linear hybrid systems. For the first approach based on model checking, it requires the decidability of the problem and therefore can only be applied to some simple hybrid systems, e.g. timed automata [29], multirate automata [30], and rectangular automata [31, 32]. The second approach can apply to a wider range of hybrid systems [2], however, it still can not be applied to general linear hybrid systems and nonlinear systems. Applying abstraction or (numeric) approximation [33, 3, 4] can improve the scalability, but as a price we have to sacrifice the precision.

In contrast, deductive methods increasingly attract more attention in the verification of hybrid systems as it can scale up to complex systems. Differential invariant generation for differential equations is at the core of deductive verification of hybrid systems. Invariants for linear hybrid systems are first studied [34, 35, 36]. For polynomial systems, the method based on constraint solving is proposed to generate polynomial invariants [37, 38, 5, 39]. The basic idea of these methods is to reduce the invariant generation problem to a constraint solving problem using techniques from polynomial ideal theory. Another method is based on the SOS-relaxation approach [8, 40], to compute *barrier certificates* for polynomial hybrid systems. The work on generating non-polynomial invariants for polynomial hybrid systems are also studied recently [41, 42]. For elementary hybrid systems, some ideas on generating invariants for them are investigated in [39, 43]. In [44], the author proposed a change-of-bases method to transform elementary hybrid systems to polynomial and linear systems. In [45], the elementary hybrid systems are reduced to polynomial hybrid systems for verification, by replacing all non-polynomial terms with newly introduced variables based on symbolic abstraction.

Based on the differential invariants, the deductive verification method can be extended to hybrid systems. A differential-algebraic dynamic logic for hybrid programs [46] was proposed by extending dynamic logic with continuous statements, and has been applied for safety checking of European Train Control System [47]. The hybrid programs proposed in [46] are a textual encoding of hybrid automata. In [48], Hybrid Event-B is proposed by extending Event-B with continuous behaviors, and furthermore, a suite of proof obligations is defined for semantics and verification of Hybrid Event-B. In [49, 50, 51], the Hoare logic is extended to hybrid systems modeled by Hybrid CSP [21, 22], and then used for safety checking of Chinese Train Control System.

All the work mentioned above focus on safety without considering denial-of-service security attacks from the environment. Quality Calculus [23, 52] for the first time proposed a programming notation for expressing denial-of-service in communication systems, but is currently limited to discrete time world.

2. SYNTAX

We first choose Hybrid CSP (HCSP) [21, 22] as the modelling language for hybrid systems. HCSP inherits from CSP the explicit communication model and concurrency, thus is expressive enough for describing distributed components and the interactions among them. Moreover, it extends CSP with differential equations for representing continuous evolution, and provides several forms of interrupts to continuous evolution for realizing communication-based discrete control. On the other hand, Quality Calculus [23, 52] is recently proposed to programming software components and their interactions in the presence of unreliable communications. With the help of *binders* specifying the success or failure of communications and the communications to be performed before continuing, it becomes natural in Quality Calculus to plan for reasonable behavior in case the ideal behavior fails due to communication failure and thereby to increase the quality of the system.

In our approach, we will extend HCSP with the notion of binders from Quality Calculus, for modelling hybrid systems in the presence of unreliable communications. The overall modelling language, called *bHCSP*, is given by the following syntax:

$$
\begin{array}{lll}
e & ::= & c \mid x \mid f^k(e_1, ..., e_k) \\
b & ::= & ch!e\{u_1\} \mid ch?x\{u_2\} \mid \&_q(b_1, \cdots, b_n) \\
P, Q & ::= & \mathrm{skip} \mid x := e \mid b \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \\
 & & \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq b \to Q \\
 & & \mid P \| Q \mid P; Q \mid \omega \to P \mid P^*
\end{array}
$$

Expressions e are used to construct data elements and consist of constants c, data variables x, and function application $f^k(e_1, ..., e_k)$.

Binders b specify the inputs and outputs to be performed before continuing. The output $ch!e\{u_1\}$ expects to send message e along channel ch, with u_1 being the acknowledgement in case the communication succeeds, and the dual input $ch?x\{u_2\}$ expects to receive a message from ch and assigns it to variable x, with u_2 being the acknowledgement similarly. We call both u_1 and u_2 *acknowledgment variables*, and assume in syntax that for each input or output statement, there exists a unique acknowledgement variable attached to it. In the sequel, we will use $\mathcal{V}$ and $\mathcal{A}$ to represent the set of data variables and acknowledgement variables respectively, and they are disjoint. For the general form $\&_q(b_1, \cdots, b_n)$, the quality predicate q specifies the sufficient communications among $b_1, \cdots, b_n$ for the following process to proceed. In syntax, q is a logical combination of quality predicates corresponding to b_1, $\cdots$, b_n recursively (denoted by $q_1, \cdots, q_n$ respectively below). For example, the quality predicates for $ch!e\{u_1\}$ and $ch?x\{u_2\}$ are boolean formulas $u_1 = 1$ and $u_2 = 1$. There are two special forms of quality predicates, abbreviated as $\forall$ and $\exists$, with the definitions: $\forall(q_1, \cdots, q_n) \stackrel{\text{def}}{=} q_1 \wedge \cdots \wedge q_n$ and $\exists(q_1, \cdots, q_n) \stackrel{\text{def}}{=} q_1 \vee \cdots \vee q_n$. More forms of quality predicates can be found in [23].

EXAMPLE 1. For the train example, let binder b_0 be $\&_\exists(\mathsf{dr}?x_a\{u_a\}, \mathsf{vc}?y_a\{w_a\})$, the quality predicate of which amounts to $u_a = 1 \vee w_a = 1$. It expresses that, the train is waiting for the acceleration from the driver and the VC, via dr and vc respectively, and as soon as one of the communications succeeds (i.e., when the quality predicate becomes true), the following process will be continued without waiting for the other if it is not ready to occur. □

P, Q define processes. The skip and assignment $x := e$ are defined as usual, taking no time to complete. Binders b are explained above. The continuous evolution $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$, where s represents a vector of continuous variables and $\dot{s}$ the corresponding first-order derivative of s, forces s to evolve according to the differential equations $\mathcal{F}$ as long as B, a boolean formula of s that defines the *domain of* s, holds, and terminates when B turns false. Without loss of generality, we assume B is open, e.g. $s < 2$. The communication interrupt $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq b \to Q$ behaves as $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$ first, and if b occurs before the continuous terminates, the continuous will be preempted and Q will be executed instead.

The rest of the constructs define compound processes. The parallel composition $P \| Q$ behaves as if P and Q run independently except that the communications along the common channels connecting P and Q are to be synchronized. In syntax, P and Q in parallel are restricted not to share variables, nor input or output channels. The sequential composition $P; Q$ behaves as P first, and if it terminates, as Q afterwards. The conditional $\omega \to P$ behaves as P if ω is true, otherwise terminates immediately. The condition ω can be used for checking the status of data variables or acknowledgement, thus in syntax, it is a boolean formula on data and acknowledgement variables (while for the above continuous evolution, B is a boolean formula on only data variables). The repetition P^* executes P for arbitrarily finite number of times.

Some constructs of HCSP in [21, 22] are derivable from the above syntax, e.g.,

$$
\begin{array}{lll}
\textbf{wait } d & \stackrel{\text{def}}{=} & t := 0; \langle \dot{t} = 1 \& t < d \rangle \\
\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q & \stackrel{\text{def}}{=} & (t := 0); \\
\multicolumn{3}{r}{\langle (\mathcal{F}(\dot{s}, s) = 0 \wedge \dot{t} = 1) \& (t < d \wedge B) \rangle; (t \geq d \to Q)}
\end{array}
$$

Especially the timeout $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q$ executes according to the continuous evolution $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$ for the first d time units, then Q afterwards. Furthermore, with the addition of binders, it is able to derive a number of other known constructs of process calculi, e.g., internal and external choice [23].

EXAMPLE 2. Following Example 1, the following model

$$t := 0;{}^1\langle \dot{s} = v, \dot{v} = a, \dot{t} = 1 \& t < T \rangle \trianglerighteq b_0{}^2 \rightarrow \\ (w_a = 1^3 \rightarrow a := y_a; w_a = 0 \wedge u_a = 1^4 \rightarrow a := x_a; \\ w_a = 0 \wedge u_a = 0^5 \rightarrow \text{skip})$$

denoted by P_0, expresses that, the train moves with velocity v and acceleration a, and as soon as b_0 occurs within T time units, i.e. the train succeeds to receive a new acceleration from either the driver or the VC, then its acceleration a will be updated correspondingly by case analysis. It can be seen that the acceleration from VC will be used in higher priority. For later reference we have annotated the program with labels (e.g. 1, 2, etc.). □

3. TRANSITION SEMANTICS

We first introduce a variable now to record the global time during process execution, and then define the set $\mathcal{V}^+ = \mathcal{V} \cup \mathcal{A} \cup \{now\}$. A state, ranging over σ, σ', is a mapping from variables in $\mathcal{V}^+$ to their respective values, and we will use Σ to represent the set of states. A flow, ranging over h, h', defined on a closed time interval $[r_1, r_2]$ with $0 \leq r_1 \leq r_2$, or an infinite interval $[r, \infty)$ with some $r \geq 0$, assigns a state in Σ to each point in the interval. Given a state σ, an expression e is evaluated to a value under σ, denoted by $\sigma(e)$.

Each transition relation has the form $(P, \sigma) \xrightarrow{\alpha} (P', \sigma', h)$, where P is a process, or ϵ introduced for representing the terminal process, σ, σ' are states, h is a flow, and α is an event. It represents that starting from initial state σ, P evolves into P' and ends with state σ' and flow h, while performing event α. When the above transition takes no time, it produces a point flow, i.e. $\sigma(now) = \sigma'(now)$ and $h = \{\sigma(now) \mapsto \sigma'\}$, and we will call the transition *discrete* and write $(P, \sigma) \xrightarrow{\alpha} (P', \sigma')$ instead without losing any information. The label α represents events, which are classified into the following cases:

- a discrete internal event, including skip, assignment, evaluation of boolean conditions, or termination of a continuous evolution, and so on, uniformly denoted by τ;
- an external communication, including $ch!c$ or $ch?c$ with $c \in \mathbb{R}$, meaning that an output or an input along ch occurs, to be synchronized with the compatible input or output from the external environment in parallel respectively;
- an internal communication, denoted by $ch\dagger c$, meaning that a synchronized communication occurs along channel ch. More precisely, when both $ch!c$ and $ch?c$ occur, $ch\dagger c$ occurs as a consequence;
- a time delay d for some positive real $d > 0$

We call the events but the time delay *discrete events*, and will use β to range over them. For simplicity, we will use $(P, \sigma) \xrightarrow{\alpha}$ as an abbreviation of the following definition:

$$\exists P', \sigma', h.(P, \sigma) \xrightarrow{\alpha} (P', \sigma', h)$$

meaning that, starting from state σ, P is able to take a transition by performing event α.

The transition relations for binders are defined in Table 1. The input $ch?x\{u\}$ may perform an external communication $ch?c$, and as a result x will be bound to c and u set to 1, or it may keep waiting for d time. For the second case, a flow h_d over $[\sigma(now), \sigma(now) + d]$ is produced, satisfying that for any t in the domain, $h_d(t) = \sigma[now \mapsto t]$, i.e. no variable but the clock now in $\mathcal{V}^+$ is changed during the waiting period. Similarly, there are two rules for output $ch!e\{u\}$. Here $\sigma[now + d]$ is an abbreviation for $\sigma[now \mapsto \sigma(now) + d]$.

Before defining the semantics of general binders, we introduce two auxiliary functions. Assume $(b_1, \cdots, b_n)$ is an intermediate tuple of binders that occurs during execution (thus some of b_is might contain ϵ), q a quality predicate, and σ a state. The function $[\![q]\!](b_1, \cdots, b_n)$ defines the truth value of q under $(b_1, \cdots, b_n)$, which is calculated by replacing each sub-predicate q_i corresponding to b_i in q by $b_i \equiv \epsilon$ respectively (here $\equiv$ represents the structural equality); and function $\langle\!|(b_1, b_2, \cdots, b_n)|\!\rangle\sigma$ returns a state that fully reflects the failure or success of binders $b_1, \cdots, b_n$, and can be constructed from σ by setting the acknowledgement variables corresponding to the failing inputs or outputs among $b_1, \cdots, b_n$ to be 0. Based on these definitions, binder $\&_q(b_1, \cdots, b_n)$ executes according to the following cases: it may keep waiting for d time when q is false under $(b_1, \cdots, b_n)$, or it performs a communication event β that is enabled for some b_i, or it performs a τ transition and terminates, when q turns true under $(b_1, \cdots, b_n)$, and moreover, no communication event is enabled for all b_is. Notice the special case that q turns true, but there is still communication event enabled for some b_i. For this case, the binder will not terminate until all the enabled communication events are taken.

EXAMPLE 3. Starting from σ_0, the execution of b_0 in Example 1 may lead to three possible states at termination:

- $\sigma_0[now + d, x_a \mapsto c_a, u_a \mapsto 1, w_a \mapsto 0]$, indicating that the train succeeds to receive c_a from the driver after d time units have passed, but fails for the VC;
- $\sigma_0[now + d, y_a \mapsto d_a, w_a \mapsto 1, u_a \mapsto 0]$, for the opposite case of the first;
- $\sigma_0[now + d, x_a \mapsto c_a, u_a \mapsto 1, y_a \mapsto d_a, w_a \mapsto 1]$, indicating that the train succeeds to receive messages from the driver as well as the VC after d time. □

The transition relations for other processes are defined in Table 2 and Table 3. The rules for skip and assignment can be defined as usual. The idle rule represents that the process can stay at the terminating state ϵ for arbitrary d time units, with nothing changed but only the clock progress. For continuous evolution, for any $d > 0$, it evolves for d time units according to $\mathcal{F}$ if B evaluates to true within this period (the right end exclusive). A flow $h_{d,s}$ over $[\sigma(now), \sigma(now) + d]$ will then be produced, such that for any o in the domain, $h_{d,s}(o) = \sigma[now \mapsto o, s \mapsto S(o - \sigma(now))]$, where $S(t)$ is the solution as defined in

(Input)

$(ch?x\{u\},\sigma) \xrightarrow{ch?c} (\epsilon, \sigma[x \mapsto c, u \mapsto 1])$

$(ch?x\{u\},\sigma) \xrightarrow{d} (ch?x\{u\}, \sigma[now+d], h_d)$

(Output)

$(ch!e\{u\},\sigma) \xrightarrow{ch!\sigma(e)} (\epsilon, \sigma[u \mapsto 1])$

$(ch!e\{u\},\sigma) \xrightarrow{d} (ch!e\{u\}, \sigma[now+d], h_d)$

(Auxiliary functions)

$[\![q]\!](b_1,\cdots,b_n) = q[(b_1 \equiv \epsilon)/q_1, \cdots, (b_n \equiv \epsilon)/q_n]$

$\langle\!|()|\!\rangle\sigma = \sigma \quad \langle\!|(\epsilon, b_2,\cdots,b_n)|\!\rangle\sigma = \langle\!|(b_2,\cdots,b_n)|\!\rangle\sigma$

$\langle\!|(ch?x\{u\}, b_2,\cdots,b_n)|\!\rangle\sigma = \langle\!|(b_2,\cdots,b_n)|\!\rangle(\sigma[u \mapsto 0])$

$\langle\!|(ch!e\{u\}, b_2,\cdots,b_n)|\!\rangle\sigma = \langle\!|(b_2,\cdots,b_n)|\!\rangle(\sigma[u \mapsto 0])$

$\langle\!|(\&_{q_k}(b_{k1},\cdots,b_{km}), b_2,\cdots,b_n)|\!\rangle\sigma =$
$\langle\!|(b_{k1},\cdots,b_{km}, b_2,\cdots,b_n)|\!\rangle\sigma$

(Binder-1)

$$\frac{[\![q]\!](b_1,\cdots,b_n) = false}{(\&_q(b_1,\cdots,b_n),\sigma) \xrightarrow{d} (\&_q(b_1,\cdots,b_n), \sigma[now+d], h_d)}$$

(Binder-2)

$$\frac{(b_i,\sigma) \xrightarrow{\beta} (b_i',\sigma') \quad \beta \in \{ch?c, ch!c\}}{(\&_q(b_1,\cdots,b_i,\cdots,b_n),\sigma) \xrightarrow{\beta} (\&_q(b_1,\cdots,b_i',\cdots,b_n),\sigma')}$$

(Binder-3)

$$\frac{[\![q]\!](b_1,\cdots,b_n) = true \quad \neg\exists\beta \in \{ch?c, ch!c\}.((b_i,\sigma) \xrightarrow{\beta}) \quad \langle\!|(b_1,\cdots,b_n)|\!\rangle\sigma = \sigma'}{(\&_q(b_1,\cdots,b_n),\sigma) \xrightarrow{\tau} (\epsilon,\sigma')}$$

TABLE 1. The transition relations for binders

(Skip) $(\text{skip},\sigma) \xrightarrow{\tau} (\epsilon,\sigma)$

(Ass) $(x := e,\sigma) \xrightarrow{\tau} (\epsilon, \sigma[x \mapsto \sigma(e)])$

(Idle) $(\epsilon,\sigma) \xrightarrow{d} (\epsilon, \sigma[now+d], h_d)$

(Continuous-1)

$$\frac{\begin{array}{c}\text{For any } d>0,\ S(t) \text{ is a solution of } \mathcal{F}(\dot{s},s)=0 \\ \text{over } [0,d] \text{ satisfying that } S(0)=\sigma(s) \\ \text{and } \forall t \in [0,d).h_{d,s}(t+\sigma(now))(B) = true\end{array}}{\begin{array}{c}(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle,\sigma) \xrightarrow{d} (\langle \mathcal{F}(\dot{s},s)=0\&B\rangle, \\ \sigma[now+d, s \mapsto S(d)], h_{d,s})\end{array}}$$

(Continuous-2)

$$\frac{\sigma(B) = false}{(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle,\sigma) \xrightarrow{\tau} (\epsilon,\sigma)}$$

(Interrupt-1)

$$\frac{\begin{array}{c}(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle,\sigma) \xrightarrow{d} (\langle \mathcal{F}(\dot{s},s)=0\&B\rangle,\sigma',h) \\ (b,\sigma) \xrightarrow{d} (b,\sigma'',h'')\end{array}}{\begin{array}{c}(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b \to Q,\sigma) \xrightarrow{d} \\ (\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b \to Q,\sigma',h)\end{array}}$$

(Interrupt-2)

$$\frac{(b,\sigma) \xrightarrow{\beta} (b',\sigma') \quad b' \neq \epsilon}{\begin{array}{c}(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b \to Q,\sigma) \xrightarrow{\beta} \\ (\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b' \to Q,\sigma')\end{array}}$$

(Interrupt-3)

$$\frac{(b,\sigma) \xrightarrow{\beta} (\epsilon,\sigma')}{(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b \to Q,\sigma) \xrightarrow{\beta} (Q,\sigma')}$$

(Interrupt-4)

$$\frac{\begin{array}{c}(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle,\sigma) \xrightarrow{\tau} (\epsilon,\sigma')\ \neg((b,\sigma) \xrightarrow{\tau} (\epsilon,-)) \\ b \equiv \&_q(b_1,\cdots,b_n) \quad \langle\!|(b_1,\cdots,b_n)|\!\rangle\sigma' = \sigma''\end{array}}{(\langle \mathcal{F}(\dot{s},s)=0\&B\rangle \trianglerighteq b \to Q,\sigma) \xrightarrow{\tau} (\epsilon,\sigma'')}$$

TABLE 2. The transition relations for atomic processes

the rule. Otherwise, the continuous evolution terminates at a point if B evaluates to false at the point.

For communication interrupt, the process may evolve for d time units if both the continuous evolution and the binder can progress for d time units, and then reach the same state and flow as the continuous evolution does. It may perform a discrete event over b, and if the resulting binder b' is not ϵ, then the continuous evolution is kept, otherwise, the continuous evolution will be interrupted and Q will be followed to execute, and for both cases, will reach the same state and flow as the binder does. Finally, it may perform a τ event and terminate immediately, if the continuous evolution terminates with a τ event and b is not able to terminate by taking a τ event. Notice that the final state σ'' needs to be reconstructed from σ' by resetting the acknowledgement variables of those unsuccessful binders occurring in b to be 0.

Before defining the semantics of parallel composition, we need to introduce some notations. Two states σ_1 and σ_2 are *disjoint*, iff $\text{dom}(\sigma_1) \cap \text{dom}(\sigma_2) = \{now\}$ and $\sigma_1(now) = \sigma_2(now)$. For two disjoint states σ_1 and σ_2, $\sigma_1 \uplus \sigma_2$ is defined as a state over $\text{dom}(\sigma_1) \cup \text{dom}(\sigma_2)$, satisfying that $\sigma_1 \uplus \sigma_2(v)$ is $\sigma_1(v)$ if $v \in \text{dom}(\sigma_1)$, otherwise $\sigma_2(v)$ if $v \in \text{dom}(\sigma_2)$. We lift this definition to flows h_1 and h_2 satisfying $\text{dom}(h_1) = \text{dom}(h_2)$, and define $h_1 \uplus h_2$ to be a flow such that $h_1 \uplus h_2(t) = h_1(t) \uplus h_2(t)$. For $P\|Q$, assume σ_1 and σ_2 represent the initial states for P and Q respectively and are disjoint. The process will perform a communication along a common channel of P and Q, if P and Q get ready to synchronize with each other along the channel. Otherwise, it will perform a discrete event, that can be τ, an internal communication of P, or an external communication along some non-common channel of P and Q, if P can progress separately on this event (and the symmetric rule for Q is left out here). When neither internal communication nor τ event is enabled for $P\|Q$, it may evolve for d time units if both P and Q can evolve for d time units. Finally, the process will perform a τ event and terminate as soon as both the components terminate.

At last, the rules for conditional, sequential, and repetition are defined as usual.

EXAMPLE 4. Starting from state σ_0, the execution of P_0

(**Parallel-1**)

$$\frac{(P,\sigma_1) \xrightarrow{ch?c} (P',\sigma_1') \quad (Q,\sigma_2) \xrightarrow{ch!c} (Q',\sigma_2')}{(P \parallel Q, \sigma_1 \uplus \sigma_2) \xrightarrow{ch\dagger c} (P'||Q', \sigma_1' \uplus \sigma_2')}$$

(**Parallel-2**)

$$\frac{\begin{array}{c}(P,\sigma_1) \xrightarrow{\beta} (P',\sigma_1') \quad \beta \in \{\tau, ch\dagger c, ch?c, ch!c \mid \\ ch \notin \mathbf{Chan}(P) \cap \mathbf{Chan}(Q)\} \\ \forall ch, c.(\neg((P,\sigma_1) \xrightarrow{ch?c} \wedge (Q,\sigma_2) \xrightarrow{ch!c}) \\ \wedge\neg((P,\sigma_1) \xrightarrow{ch!c} \wedge (Q,\sigma_2) \xrightarrow{ch?c}))\end{array}}{(P \parallel Q, \sigma_1 \uplus \sigma_2) \xrightarrow{\beta} (P'||Q, \sigma_1' \uplus \sigma_2)}$$

(**Parallel-3**)

$$\frac{\begin{array}{c}(P,\sigma_1) \xrightarrow{d} (P',\sigma_1',h_1) \quad (Q,\sigma_2) \xrightarrow{d} (Q',\sigma_2',h_2) \\ \forall ch, c.\neg((P \parallel Q, \sigma_1 \uplus \sigma_2) \xrightarrow{ch\dagger c}) \\ \neg((P \parallel Q, \sigma_1 \uplus \sigma_2) \xrightarrow{\tau})\end{array}}{(P \parallel Q, \sigma_1 \uplus \sigma_2) \xrightarrow{d} (P'||Q', \sigma_1' \uplus \sigma_2', h_1 \uplus h_2)}$$

(**Parallel-4**)

$$(\epsilon||\epsilon,\sigma) \xrightarrow{\tau} (\epsilon,\sigma)$$

(**Alternation**)

$$\frac{\sigma(\omega) = true\ (P,\sigma) \xrightarrow{\alpha} (P',\sigma',h)}{(\omega \to P,\sigma) \xrightarrow{\alpha} (P',\sigma',h)} \quad \frac{\sigma(\omega) = false}{(\omega \to P,\sigma) \xrightarrow{\tau} (\epsilon,\sigma)}$$

(**Sequential composition**)

$$\frac{(P,\sigma) \xrightarrow{\alpha} (P',\sigma',h)\ P' \neq \epsilon}{(P;Q,\sigma) \xrightarrow{\alpha} (P';Q,\sigma',h)} \quad \frac{(P,\sigma) \xrightarrow{\alpha} (\epsilon,\sigma',h)}{(P;Q,\sigma) \xrightarrow{\alpha} (Q,\sigma',h)}$$

(**Repetition**)

$$\frac{(P,\sigma) \xrightarrow{\alpha} (P',\sigma',h)\ P' \neq \epsilon}{(P^*,\sigma) \xrightarrow{\alpha} (P';P^*,\sigma',h)} \quad \frac{(P,\sigma) \xrightarrow{\alpha} (\epsilon,\sigma',h)}{(P^*,\sigma) \xrightarrow{\alpha} (P^*,\sigma',h)}$$

$$(P^*,\sigma) \xrightarrow{\tau} (\epsilon,\sigma)$$

TABLE 3. The transition relations for composite processes

in Example 2 leads to the following cases (let v_0 denote $\sigma_0(v)$ below):

- P_0 terminates without the occurrence of b_0, the final state is $\sigma_0[now + T, t + T, v + aT, s + v_0T + 0.5aT^2, u_a \mapsto 0, w_a \mapsto 0]$;
- b_0 occurs after d time units for some $d \leq T$, and as a result P_0 executes to location 2, with state $\sigma_0[now + d, t + d, v + ad, s + v_0d + 0.5ad^2, u_a, w_a, x_a, y_a]$, where u_a, w_a, x_a and y_a have 3 possible evaluations as defined in Example 3, and then depending on the values of u_a and w_a, executes to location 3 or 4 respectively, and finally terminates after a corresponding acceleration update. □

Flow of a Process Given two flows h_1 and h_2 defined on $[r_1, r_2]$ and $[r_2, r_3]$ (or $[r_2, \infty)$) respectively, we define the concatenation $h_1^\frown h_2$ as the flow defined on $[r_1, r_3]$ (or $[r_1, \infty)$) such that $h_1^\frown h_2(t)$ is equal to $h_1(t)$ if $t \in [r_1, r_2)$, otherwise $h_2(t)$. Given a process P and an initial state σ, if we have the following sequence of transitions:

$$(P,\sigma) \xrightarrow{\alpha_0} (P_1,\sigma_1,h_1) \quad (P_1,\sigma_1) \xrightarrow{\alpha_1} (P_2,\sigma_2,h_2)$$
$$\ldots \quad (P_{n-1},\sigma_{n-1}) \xrightarrow{\alpha_{n-1}} (P_n,\sigma_n,h_n)$$

then we define $h_1^\frown \ldots ^\frown h_n$ as the *flow* from P to P_n with respect to the initial state σ, and furthermore, write $(P,\sigma) \xrightarrow{\alpha_0 \cdots \alpha_{n-1}} (P_n, \sigma_n, h_1^\frown \ldots ^\frown h_n)$ to represent the whole transition sequence (and for simplicity, the label sequence can be omitted sometimes). When P_n is ϵ, we call $h_1^\frown \ldots ^\frown h_n$ a *complete flow* of P with respect to σ.

4. A TIME AWARE INFERENCE SYSTEM

In this section, we define a time aware inference system for reasoning about both discrete and continuous properties of bHCSP, which are considered for an isolated time point and a time interval respectively.

History Formulas In order to describe the interval-related properties, we introduce history formulas, that are defined by duration calculus (DC) [53, 54]. DC is a first-order interval-based real-time logic with one binary modality known as chop $^\frown$. History formulas HF are defined by the following subset of DC:

$$\begin{array}{rcl} HF & ::= & \ell \circ T \mid \lceil S \rceil^0 \mid \lceil S \rceil \mid HF_1{}^\frown HF_2 \\ & & \mid HF_1 \wedge HF_2 \mid HF_1 \vee HF_2 \end{array}$$

where ℓ is a special temporal variable of DC denoting the length of the considered interval, $\circ \in \{<, >, =\}$ is a relation, T is a non-negative real, and S is a first-order state formula over process variables.

State formulas S can be interpreted over states, and history formulas HF can be interpreted over flows and intervals. We define the judgements $\sigma \models S$ to represent that S holds under state σ, and $h, [a, b] \models HF$ to represent that HF holds under h and $[a, b]$. We have

$h, [a, b] \models \ell \circ T$ iff $(b - a) \circ T$
$h, [a, b] \models \lceil S \rceil^0$ iff $a = b \wedge h(a) \models S$
$h, [a, b] \models \lceil S \rceil$ iff $\forall t \in [a, b).h(t) \models S$
$h, [a, b] \models HF_1{}^\frown HF_2$ iff $\exists c.a \leq c \leq b \wedge h, [a, c] \models HF_1$ $\wedge h, [c, b] \models HF_2$

As defined above, ℓ indicates the length of the considered interval; $\lceil S \rceil^0$ asserts that the interval contains only one point and S holds for the point, and it is called *singleton* formula; $\lceil S \rceil$ asserts that S holds everywhere except for the right endpoint in the considered interval[3]; and $HF_1^\frown HF_2$ asserts that the interval can be divided into two sub-intervals such that HF_1 holds for the first and HF_2 for the second. The first-order connectives $\wedge$ and $\vee$ can be explained as usual.

For the history formulas, all axioms and inference rules for DC presented in [54] can be applied here, such as

$$\text{True} \Leftrightarrow \ell \geq 0 \quad \lceil S \rceil^\frown \lceil S \rceil \Leftrightarrow \lceil S \rceil \quad HF^\frown \ell = 0 \Leftrightarrow HF$$
$$\lceil S_1 \rceil \Rightarrow \lceil S_2 \rceil \text{ if } S_1 \Rightarrow S_2 \text{ is valid in FOL}$$

[3]This is a stronger version of the operator $\lceil S \rceil$ in [20], which requires that S holds almost everywhere, i.e. S can be 0 at at most a finite number of time points in the interval.

Specification The specification for process P takes form $\{\varphi\}\ P\ \{\psi, \mathit{HF}\}$, where the precondition/postcondition φ and ψ, defined by FOL, specify properties of variables that hold at the beginning and termination of the execution of P respectively, and the history formula HF, specifies properties of variables that hold throughout the execution interval of P. The specification of P is defined with no dependence on the behavior of its environment. The specification is *valid*, denoted by $\models \{\varphi\}\ P\ \{\psi, \mathit{HF}\}$, iff for any state σ, if $(P, \sigma) \rightarrow (\epsilon, \sigma', h)$, then $\sigma \models \varphi$ implies $\sigma' \models \psi$ and $h, [\sigma(now), \sigma'(now)] \models \mathit{HF}$.

Acknowledgement of Binders In order to define the inference rules for binders b, we first define an auxiliary typing judgement $\vdash b \blacktriangleright \varphi$, where the first-order formula φ describes the acknowledgement corresponding to successful passing of b, and is defined without dependence on the precondition of b. We say $b \blacktriangleright \varphi$ *valid*, denoted by $\models b \blacktriangleright \varphi$, iff given any state σ, if $(b, \sigma) \rightarrow (\epsilon, \sigma', h)$, then $\sigma' \models \varphi$ holds.

The typing judgement for binders is defined as follows:

$$\vdash ch?x\{u\} \blacktriangleright u = 1 \qquad \vdash ch!e\{u\} \blacktriangleright u = 1$$

$$\frac{\vdash b_1 \blacktriangleright \varphi_1,\ \cdots,\ \vdash b_n \blacktriangleright \varphi_n}{\vdash \&_q(b_1, \cdots, b_n) \blacktriangleright [\![q]\!](\varphi_1, \cdots, \varphi_n)}$$

As indicated above, for input $ch?x\{u\}$, the successful passing of it gives rise to formula $u = 1$, and similarly for output $ch!e\{u\}$; for binder $\&_q(b_1, \cdots, b_n)$, it gives rise to formula $[\![q]\!](\varphi_1, \cdots, \varphi_n)$, which encodes the effect of quality predicate q to the sub-formulas $\varphi_1, \ldots, \varphi_n$ corresponding to $b_1, \ldots, b_n$ respectively.

EXAMPLE 5. For binder b_0 in Example 1, we have $\vdash b_0 \blacktriangleright u_a = 1 \vee w_a = 1$, indicating that, if the location after b_0 is reachable, then at least one of the communications with the driver or the VC succeeds. $\square$

4.1. Inference Rules

Before presenting the inference system, we introduce some notations first. Given a binder b, we define a function $mv(b)$ to return the variables that may be modified by b. It can be defined directly by structural induction on b:

$$mv(b) \stackrel{\text{def}}{=} \begin{cases} \{x, u_1\} & \text{if } b \equiv ch?x\{u_1\} \\ \{u_2\} & \text{if } b \equiv ch!e\{u_2\} \\ \bigcup_{1 \leq i \leq n} mv(b_i) & \text{if } b \equiv \&_q(b_1, \cdots, b_n) \end{cases}$$

Given a history formula HF, we define a function $\mathit{Inr}(HF)$ to return the *internal* of HF, which is same to HF except that each singleton formula of the right endpoint in HF is replaced by $\ell = 0$ if it exists. $\mathit{Inr}(HF)$ is defined by

$$\textbf{(Skip-A)} \quad \{\varphi\}\ \text{skip}\ \{\varphi, \lceil\varphi\rceil^0\}$$

$$\textbf{(Ass-A)} \quad \{\psi[e/x]\}\ x := e\ \{\psi, \lceil\psi\rceil^0\}$$

$$\textbf{(In-A)} \quad \{\varphi\}\ ch?x\{u\}\ \{(\exists x, u.\varphi) \wedge u = 1, \lceil\varphi\rceil\}$$

$$\textbf{(Out-A)} \quad \{\varphi\}\ ch!e\{u\}\ \{(\exists u.\varphi) \wedge u = 1, \lceil\varphi\rceil\}$$

$$\textbf{(Binder-A)} \quad \frac{\vdash \&_q(b_1, \cdots, b_n) \blacktriangleright \alpha}{\{\varphi\}\ \&_q(b_1, \cdots, b_n)\ \left\{ \begin{array}{l} (\exists mv(\&_q(b_1, \cdots, b_n)).\varphi) \wedge \alpha, \\ \lceil\exists mv(\&_q(b_1, \cdots, b_n)).\varphi\rceil \end{array} \right\}}$$

$$\textbf{(Con-A)} \quad \{\varphi\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle\ \left\{ \begin{array}{l} (\exists s.\varphi) \wedge \neg B \wedge Inv, \\ \lceil(\exists s.\varphi) \wedge B \wedge Inv\rceil \end{array} \right\}$$

$$\textbf{(Int-A)} \quad \frac{\begin{array}{c}\vdash \&_q(b_1, \cdots, b_n) \blacktriangleright \alpha \\ \{(\exists mv(b).(\exists s.\varphi) \wedge Inv) \wedge \alpha\}\ Q\ \{\psi_1, \mathit{HF}_1\}\end{array}}{\begin{array}{l}\{\varphi\}\langle\mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq b \rightarrow Q \\ \quad \{(\exists mv(b).(\exists s.\varphi) \wedge \neg B \wedge Inv) \vee \psi_1, \\ \qquad \lceil\exists mv(b).(\exists s.\varphi) \wedge B \wedge Inv\rceil^\frown(\ell = 0 \vee \mathit{HF}_1)\}\end{array}}$$

$$\textbf{(Par-A)} \quad \frac{\{\varphi\}\ P\ \{\psi_1, \mathit{HF}_1\} \quad \{\varphi\}\ Q\ \{\psi_2, \mathit{HF}_2\}}{\{\varphi\}\ P \| Q\ \left\{ \begin{array}{l} \psi_1 \wedge \psi_2, \\ ((\mathit{HF}_1^\frown\lceil\psi_1\rceil) \wedge \mathit{HF}_2) \vee (\mathit{HF}_1 \wedge (\mathit{HF}_2^\frown\lceil\psi_2\rceil)) \end{array} \right\}}$$

$$\textbf{(Seq-A)} \quad \frac{\{\varphi\}\ P\ \{\psi_1, \mathit{HF}_1\} \quad \{\psi_1\}\ Q\ \{\psi_2, \mathit{HF}_2\}}{\{\varphi\}\ P; Q\ \{\psi_2, \mathit{Inr}(\mathit{HF}_1)^\frown\mathit{HF}_2\}}$$

$$\textbf{(Alt-A)} \quad \frac{\{\varphi \wedge \omega\}\ P\ \{\psi_1, \mathit{HF}_1\}}{\{\varphi\}\ \omega \rightarrow P\ \{(\varphi \wedge \neg\omega) \vee \psi_1, \ell = 0 \vee \mathit{HF}_1\}}$$

$$\textbf{(Rep-A)} \quad \frac{\{\varphi\}\ P\ \{\varphi, InvH\} \quad InvH^\frown InvH \Rightarrow InvH}{\{\varphi\}\ P^*\ \{\varphi, InvH \vee \ell = 0\}}$$

$$\textbf{(SHF)} \quad \frac{\{\varphi\}\ P\ \{\psi, \mathit{HF}\}}{\{\varphi\}\ P\ \{\psi, \mathit{HF}^\frown\lceil\psi\rceil^0\}}$$

TABLE 4. A time aware inference system for processes

structural induction on HF as follows:

$$\mathit{Inr}(HF) \stackrel{\text{def}}{=} \begin{cases} \ell = 0 & \text{if } HF \equiv \lceil S\rceil^0 \\ HF_1{}^\frown\mathit{Inr}(HF_2) & \\ \qquad \text{if } HF \equiv HF_1{}^\frown HF_2 \wedge HF_2 \Rightarrow \ell > 0 & \\ \mathit{Inr}(HF_1)^\frown\mathit{Inr}(HF_2) & \\ \qquad \text{if } HF \equiv HF_1{}^\frown HF_2 \wedge HF_2 \Rightarrow \ell = 0 & \\ \mathit{Inr}(HF_1) \vee \mathit{Inr}(HF_2) & \text{if } HF \equiv HF_1 \vee HF_2 \\ \mathit{Inr}(HF_1) \wedge \mathit{Inr}(HF_2) & \text{if } HF \equiv HF_1 \wedge HF_2 \\ HF & \text{otherwise} \end{cases}$$

The internal of history formulas is proposed specially for handling the super-dense computation in sequential composition.

The inference rules for deducing the specifications of all constructs are presented in Table 4. Statements skip and assignment terminate simultaneously, as indicated by the history formula. For each form of the binders b, the postcondition is the conjunction of the quantified precondition φ over variables in $mv(b)$ and the acknowledgement corresponding to the successful passing of b. For both $ch?x\{u\}$

and $ch!e\{u\}$, φ will hold everywhere in the waiting interval, i.e. the execution interval by excluding the right endpoint, at which the communication occurs and variables might be changed correspondingly. For $\&_q(b_1, \cdots, b_n)$, only the quantified φ over variables in $mv(b)$ is guaranteed to hold everywhere throughout the waiting interval, since some binders b_is that make q true might occur at sometime during the interval and as a consequence variables in φ might get changed.

For continuous evolution, the notion of differential invariants is used instead of explicit solutions. A *differential invariant* of $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle$ for given initial values of s is a first-order formula of s, which is satisfied by the initial values and also by all the values reachable by the trajectory of s defined by $\mathcal{F}$ within the domain B. The problem of differential invariant generation for polynomial differential equations has been studied in [10, 11]. Here we assume Inv is a differential invariant with respect to precondition φ for the continuous evolution (more details on using Inv will be shown in the later example proof in Section 7). For the postcondition, the quantified φ over the only modified variables s, $\neg B$, and Inv hold. For the history formula, the quantified φ over s, B and Inv holds everywhere throughout the execution interval except for the right endpoint.

For communication interrupt, if b fails to occur before the continuous evolution terminates, the effect of the whole statement is almost equivalent to the continuous evolution, except that some variables in b may get changed because of occurrences of some communications during the execution of the continuous evolution. Otherwise, if b succeeds within the termination of the continuous evolution, the continuous evolution will be interrupted and Q will start to execute from the interrupting point. At the interrupting point, the acknowledgement of b holds, and moreover, because s and variables in $mv(b)$ may have been modified, $\exists mv(b).((\exists s.\varphi) \wedge Inv)$ holds. For the second case, the postcondition is defined as the one for Q, and the history formula as the chop of the one for the continuous evolution before interruption and the one for Q afterwards. Finally, as indicated by the rule, the postcondition and history formula for the whole statement are defined as the disjunction of the above two cases.

For $P\|Q$, because P and Q do not share variables, the rule is defined by conjunction as usual. The disjunction in the history formula is due to the case that P and Q may terminate at different time. If P terminates before Q, as shown by the first disjunctive clause, the postcondition of P will always holds till the termination of Q; the contrary case is defined by the second clause. For $P;Q$, the history formula is defined by the concatenation of the internal of the history formula of of P and the history formula of Q. As a result, the super-dense computation problem can be well handled: when there are multiple discrete actions occurring at a time point, which is here the termination time of P, also the starting time of Q, only the final state according to the execution order is recorded in the final history formula of the sequential composition. This is consistent with the definition of the concatenation of flows given in Section 3. The rule for $\omega \rightarrow P$ includes two cases depending on whether ω holds or not. For P^*, we need to find the invariants, i.e. φ and $InvH$, for both the postcondition and history formula.

The last defines a general inference rule to strengthen the history formula of a process by adding the postcondition as a singleton formula at the end. We denote it by (**SHF**) for further reference. Other general inference rules that are applicable to all constructs, like monotonicity, case analysis etc., can be defined as usual and are omitted here.

We have proved the following soundness theorem:

THEOREM 4.1. *Given a process P, if $\{\varphi\}\ P\ \{\psi, HF\}$ can be deduced from the inference rules, then $\models \{\varphi\}\ P\ \{\psi, HF\}$.*

Proof. We need to prove that, for any state σ, if $(P, \sigma) \rightarrow (\epsilon, \sigma', h)$, then $\sigma \models \varphi$ implies $\sigma' \models \psi$ and $h, [\sigma(now), \sigma'(now)] \models HF$. The proof is given by structural induction on P as follows.

- The proof for skip and $x := e$ is trivial.
- Cases binders b: For $b \equiv ch?x\{u\}$, according to the transition system, there exist some $d \geq 0$ and c such that $\sigma' = \sigma[\sigma(now) \mapsto \sigma(now) + d][x \mapsto c, u \mapsto 1]$ and h defined on $[\sigma(now), \sigma(now) + d]$ satisfies that $h(t) = \sigma[now \mapsto t]$ for each t in $[\sigma(now), \sigma(now)+d)$ and $h(\sigma(now) + d) = \sigma'$. Thus, from $\sigma \models \varphi$, $\sigma' \models \exists x, u.\varphi$ and $h, [\sigma(now), \sigma'(now)] \models \lceil \varphi \rceil$ must hold (notice that now does not occur in assertions). The case for $b \equiv ch!e\{u\}$ can be proved similarly.
For $b \equiv \&_q(b_1, \cdots, b_n)$, according to the transition system, there must exist some $d \geq 0$ such that $\sigma'(now) = \sigma(now) + d$, and for each b_i evolving to ϵ at termination, there must be $\sigma'(u_i) = 1$, and for any variable x that is not $mv(b)$, for any $t \in [\sigma(now), \sigma(now')]$, $h(t)(x) = \sigma(x)$. Thus $\sigma' \models \exists mv(b).\varphi$ and $h, [\sigma(now), \sigma'(now)] \models \lceil \exists mv(b).\varphi \rceil$ hold. And, from $[\![q]\!](b_1', \cdots, b_n') = true$, where $b_1', \cdots, b_n'$ represent the final form of $b_1, \cdots, b_n$ during the execution of b, we have $\sigma' \models \alpha$ proved.
- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle$: According to the transition system, there must exist $d \geq 0$ such that $\sigma' = \sigma[now \mapsto \sigma(now) + d, s \mapsto S(d)]$ and h defined over $[\sigma(now),$
$\sigma(now) + d]$ satisfies that for any o in the domain, $h(o) = \sigma[now \mapsto o, s \mapsto S(o - \sigma(now))]$, where S is the solution of the continuous with respect to $\sigma(s)$ as defined in the rule. Moreover, for any $o \in [\sigma(now), \sigma(now) + d)$, $h(o) \models B$, and $\sigma' \models \neg B$. Obviously, $\sigma' \models (\exists s.\varphi) \wedge \neg B$. According to the definition of Inv, then for any $o \in [\sigma(now), \sigma(now) + d)$, $h(o) \models Inv$, thus $\sigma' \models Inv$ and $h, [\sigma(now), \sigma'(now)] \models \lceil Inv \rceil$ hold. Plus the fact that $h, [\sigma(now), \sigma'(now)] \models \lceil (\exists s.\varphi) \wedge B \rceil$, the result is proved.
- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle \trianglerighteq b \rightarrow Q$: According to the transition system, there are two cases for termination, by applying the fourth and the third transition rules for it respectively. For the first case, there must exist d such that $\sigma'(now) = \sigma(now) + d$, and for any variable

x except for s and the ones in $mv(b)$, $\sigma'(x) = \sigma(x)$ and for any $o \in [\sigma(now), \sigma(now) + d]$, $h(o)(x) = \sigma(x)$. Plus the semantics of continuous, we have $\sigma' \models \exists mv(b).(\exists s.\varphi) \wedge \neg B \wedge Inv$ and $h, [\sigma(now), \sigma'(now)] \models \lceil \exists mv(b).(\exists s.\varphi) \wedge B \wedge Inv \rceil$ proved. For the second case, there must exist d_1 such that $\sigma''(now) = \sigma(now) + d_1$, and for any variable x except for s and the ones in $mv(b)$, $\sigma''(x) = \sigma(x)$ and for any $o \in [\sigma(now), \sigma(now) + d]$, $h'(o)(x) = \sigma(x)$, and $\sigma'' \models (\exists mv(b).(\exists s.\varphi) \wedge Inv) \wedge \alpha$, and $(Q, \sigma'') \rightarrow (\epsilon, \sigma', h'')$, and $h = h'^\frown h''$. The fact is proved based on the inductive hypothesis on Q.

- Cases $P_1 \| Q_1$, $P_1; Q_1$ and $\omega \rightarrow P_1$: According to the transition system, for $P_1 \| Q_1$, suppose P_1 and Q_1 terminate at the same time, then there must exist σ_1, h_1, and σ_2, h_2 such that $(P_1, \sigma) \rightarrow (\epsilon, \sigma_1, h_1)$, $(Q_1, \sigma) \rightarrow (\epsilon, \sigma_2, h_2)$, $\sigma' = \sigma_1 \uplus \sigma_2$ and $h = h_1 \uplus h_2$. The fact is proved by induction hypothesis on P_1 and Q_1. The other cases can be proved easily.
According to the transition semantics, there must exist σ_1, h_1, and σ_2, h_2 such that $(P_1, \sigma) \rightarrow (\epsilon, \sigma_1, h_1)$, $(Q_1, \sigma_1) \rightarrow (\epsilon, \sigma_2, h_2)$, and $h = h_1{}^\frown h_2$. By induction hypothesis, we have the facts $\sigma_1 \models \psi_1$ and h_1, $[\sigma(now), \sigma_1(now)] \models HF_1$ for P_1, and then $\sigma_2 \models \psi_2$ and h_2, $[\sigma_1(now), \sigma_2(now)] \models HF_2$ for Q_1. According to the definition of $h_1{}^\frown h_2$, h is equal to h_1 on the right open interval $[\sigma(now), \sigma_1(now))$, while not at the point $\sigma_1(now)$. According to the definition of the internal of history formulas, $h, [\sigma(now), \sigma_1(now)] \models Inr(HF_1)$. On the other hand, h is equal to h_2 on the closed interval $[\sigma_1(now), \sigma_2(now)]$, thus $h, [\sigma_1(now), \sigma_2(now)] \models HF_2$. The fact $h, [\sigma(now), \sigma_2(now)] \models Inr(HF_1)^\frown HF_2$ is proved. At the end, the rule for $\omega \rightarrow P_1$ can be proved easily by induction hypothesis, and we omit the details here.
- Case P_1^*: According to the transition system, we have

$$\sigma' = \sigma \quad h = \{\sigma(now) \mapsto \sigma'\}$$

or there exists an integer $k > 0$ such that $\sigma_k = \sigma'$, $h = h_1{}^\frown h_2{}^\frown \cdots {}^\frown h_k$, and a sequence of transitions as follows:

$$\begin{array}{c} (P_1, \sigma) \rightarrow (\epsilon, \sigma_1, h_1) \\ (P_1, \sigma_1) \rightarrow (\epsilon, \sigma_2, h_2) \\ \cdots, (P_1, \sigma_{k-1}) \rightarrow (\epsilon, \sigma_k, h_k) \end{array}$$

For the first case, the fact holds trivially. For the second case, suppose the fact holds when $k < n$ for some $n > 0$, next we prove that the fact holds for $k = n$. According to the transition rule, we have

$$\begin{array}{l} (P, \sigma_{n-1}) \rightarrow (\epsilon, \sigma_n, h_n), \quad \sigma_{n-1} \models \varphi \\ h_1{}^\frown \cdots {}^\frown h_{n-1}, [\sigma(now), \sigma_{n-1}(now)] \models InvH \vee \ell = 0 \end{array}$$

By induction hypothesis on P_1, $\sigma_n \models \varphi$ and h_n, $[\sigma_{n-1}(now), \sigma_n(now)] \models InvH$ must hold. Then $h_1{}^\frown \cdots {}^\frown h_n$, $[\sigma(now), \sigma_n(now)] \models (InvH \vee \ell = 0)^\frown InvH$, plus $InvH^\frown InvH \Rightarrow InvH$, we have $h_1{}^\frown \cdots {}^\frown h_n, [\sigma(now), \sigma_n(now)] \models InvH$ proved.

- Case rule (**SHF**): Suppose $(P, \sigma) \rightarrow (\epsilon, \sigma', h)$ and $\sigma \models \varphi$. By induction hypothesis, $\sigma' \models \psi$ and $h, [\sigma(now), \sigma'(now)] \models HF$ are obtained. Plus $h(\sigma'(now)) = \sigma'$, we have $h, [\sigma'(now), \sigma'(now)] \models \lceil \psi \rceil^0$. Thus $h, [\sigma(now), \sigma'(now)] \models HF^\frown \lceil \psi \rceil^0$ is proved.

□

4.2. Application: Reachability Analysis

The inference system can be applied directly for reachability analysis. Given a labelled process S (a process annotated with integers denoting locations), a precondition φ and a location l in S, by applying the inference system, we can deduce a property ψ such that if S reaches l, ψ must hold at l, denoted by $\vdash S, l, \varphi \blacktriangleright \psi$. In another word, If $\vdash S, l, \varphi \blacktriangleright \psi$ and ψ is not satisfiable, then l will not be reachable in S with respect to φ. We have the following facts based on the structural induction of S:

- for any process P, $\vdash {}^l P, l, \varphi \blacktriangleright \varphi$ and $\vdash P^l, l, \varphi \blacktriangleright \psi$ provided $\{\varphi\}\ P\ \{\psi, -\}$;
- $\vdash \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq {}^l b \rightarrow S', l, \varphi \blacktriangleright \varphi$.
$\vdash \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq b^l \rightarrow S', l, \varphi \blacktriangleright (\exists mv(b).(\exists s.\varphi) \wedge Inv) \wedge \alpha$ (denoted by φ'), if $\vdash b \blacktriangleright \alpha$ holds.
$\vdash \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq b \rightarrow S', l, \varphi \blacktriangleright \psi$ if $l \in S'$ and $\vdash S', l, \varphi' \blacktriangleright \psi$ hold;
- $\vdash S_1; S_2, l, \varphi \blacktriangleright \psi$ if $l \in S_1$ and $\vdash S_1, l, \varphi \blacktriangleright \psi$ hold.
$\vdash S_1; S_2, l, \varphi \blacktriangleright \psi'$ if $l \in S_2$, $\{\varphi\}\ S_1\ \{\psi, -\}$ and $\vdash S_2, l, \psi \blacktriangleright \psi'$ hold;
- $\vdash \omega^l \rightarrow S', l, \varphi \blacktriangleright \varphi \wedge \omega$.
$\vdash \omega \rightarrow S', l, \varphi \blacktriangleright \psi$ if $l \in S'$ and $\vdash S', l, \varphi \wedge \omega \blacktriangleright \psi$;
- $\vdash S'^*, l, \varphi \blacktriangleright \psi$, if $l \in S'$, $\vdash S', l, \varphi \blacktriangleright \psi$ and $\{\varphi\}\ S'\ \{\varphi, -\}$ hold.

Obviously, the monotonicity holds: if $\vdash S, l, \varphi \blacktriangleright \psi$ and $\psi \Rightarrow \psi'$, then $\vdash S, l, \varphi \blacktriangleright \psi'$.

EXAMPLE 6. Consider P_0 in Example 2. Given precondition φ , we have $\vdash P_0, 1, \varphi \blacktriangleright (\exists t.\varphi) \wedge t = 0$, denoted by φ_1. Moreover, $\vdash P_0, 5, \varphi \blacktriangleright (\exists mv(b_0).(\exists s, v, t.\varphi_1) \wedge t \leq T) \wedge (u_a = 1 \vee w_a = 1) \wedge (u_a = 0 \wedge w_a = 0)$, the formula is un-satisfiable, thus location 5 is not reachable. Other locations can be considered similarly.

5. A TIME OBLIVIOUS INFERENCE SYSTEM

In this section, we define a more lightweight inference system for bHCSP. Different from the previous one presented in Sec.4, we characterize the continuous behavior of bHCSP by an invariant defined in FOL, thus FOL will be the only assertion language of the inference system.

Specification The specification for process P takes form $\{\varphi\}\ P\ \{\psi, I\}$, where φ, ψ and I are FOL formulas. In particular, the precondition/postcondition φ and ψ are defined as in the previous inference system, and the invariant I, specifies the property that holds throughout the whole

(Skip-O) $\{\varphi\}\ \text{skip}\ \{\varphi, \varphi\}$

(Ass-O) $\{\psi[e/x]\}\ x := e\ \{\psi, \psi\}$

(In-O) $\{\varphi\}\ ch?x\{u\}\ \{(\exists x, u.\varphi) \wedge u = 1, \exists x, u.\varphi\}$

(Out-O) $\{\varphi\}\ ch!e\{u\}\ \{(\exists u.\varphi) \wedge u = 1, \exists u.\varphi\}$

(Binder-O)

$$\frac{\vdash \&_q(b_1, \cdots, b_n) \blacktriangleright \alpha}{\{\varphi\}\ \&_q(b_1, \cdots, b_n)\ \left\{ \begin{array}{l} (\exists mv(\&_q(b_1, \cdots, b_n)).\varphi) \wedge \alpha, \\ \exists mv(\&_q(b_1, \cdots, b_n)).\varphi \end{array} \right\}}$$

(Con-O)

$$\{\varphi\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle\ \{(\exists s.\varphi) \wedge \neg B \wedge Inv, (\exists s.\varphi) \wedge Inv\}$$

(Int-O)

$$\frac{\vdash \&_q(b_1, \cdots, b_n) \blacktriangleright \alpha \qquad \{(\exists mv(b).(\exists s.\varphi) \wedge Inv) \wedge \alpha\}\ Q\ \{\psi_1, I_1\}}{\begin{array}{l} \{\varphi\}\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq b \to Q \\ \quad \{(\exists mv(b).(\exists s.\varphi) \wedge \neg B \wedge Inv) \vee \psi_1, \\ \quad (\exists mv(b).(\exists s.\varphi) \wedge Inv) \vee I_1\} \end{array}}$$

(Par-O) $$\frac{\{\varphi\}\ P\ \{\psi_1, I_1\} \quad \{\varphi\}\ Q\ \{\psi_2, I_2\}}{\{\varphi\}\ P \| Q\ \{\psi_1 \wedge \psi_2, I_1 \wedge I_2\}}$$

(Seq-O) $$\frac{\{\varphi\}\ P\ \{\psi_1, I_1\} \quad \{\psi_1\}\ Q\ \{\psi_2, I_2\}}{\{\varphi\}\ P; Q\ \{\psi_2, I_1 \vee I_2\}}$$

(Alt-O) $$\frac{\{\varphi \wedge \omega\}\ P\ \{\psi_1, I_1\}}{\{\varphi\}\ \omega \to P\ \{(\varphi \wedge \neg\omega) \vee \psi_1, \varphi \vee I_1\}}$$

(Rep-O) $$\frac{\{\varphi\}\ P\ \{\varphi, I\}}{\{\varphi\}\ P^*\ \{\varphi, \varphi \vee I\}}$$

TABLE 5. A time oblivious inference system for processes

execution interval of P. Formally, given a FOL formula I, a flow h, and two reals $c \leq d$, I is an *invariant* of h throughout the interval $[c, d]$, denoted by $h, [c, d] \models I$, iff for any time point $t \in [c, d]$, $h(t) \models I$ holds. In another word, I holds *everywhere* in the interval $[c, d]$. The specification is *valid*, denoted by $\models \{\varphi\}\ P\ \{\psi, I\}$, iff for any state σ, if $(P, \sigma) \to (\epsilon, \sigma', h)$, then $\sigma \models \varphi$ implies $\sigma' \models \psi$ and $h, [\sigma(now), \sigma'(now)] \models I$.

5.1. Inference Rules

The new inference rules for deducing the specifications of all constructs are presented in Table 5. For each inference rule, the precondition and postcondition are the same as in the previous one in Table 4, and we will only explain the invariant part.

For the discrete statements including skip and $x := e$, they both terminate without taking time, thus the invariant is same as the postcondition. For each case of binders, the invariant is the quantified precondition over the may-modified variables. For instance, for input $ch?x\{u\}$, all the variables are kept unchanged except that x and u may get changed at termination. For continuous evolution, as indicated by the rule, except for the quantified precondition over s, the differential invariant Inv also preserves as invariant during the whole continuous evolution. For communication interrupt, there are two cases depending on whether b occurs or not before the continuous evolution terminates. For the case when b fails, some communications among b may have occurred (although not strong enough to make b occur), thus the invariant is the invariant of the continuous evolution quantified over $mv(b)$. For the other case when b succeeds, Q will be followed to execute, and the invariant is the disjunction of the ones before and after b occurs. By making disjunction of these two cases, the invariant of the communication interrupt is defined.

The invariant of $P \| Q$ is defined as the conjunction of the ones of P and Q. For $P; Q$, the invariant is defined as the disjunction of the ones of P and Q. The invariant of $\omega \to P$ includes two cases depending on whether ω holds or not: for the first case, the precondition φ preserves, and for the second case, the invariant of P holds. At last, for P^*, we need to find the invariants, i.e. φ and I, for both the postcondition and the invariant. Notice that the φ in the invariant indicates the special case that the repetition terminates immediately, i.e. P executes for zero time.

The general inference rules that are applicable to all constructs, like monotonicity, case analysis etc., can be defined as usual and are omitted here.

5.2. Properties

We have proved two theorems below. First, we prove that the new inference system is sound with respect to the operational semantics, stated by the following theorem:

THEOREM 5.1. *Given a process P, if* $\{\varphi\}\ P\ \{\psi, I\}$ *can be deduced from the inference rules, then* $\models \{\varphi\}\ P\ \{\psi, I\}$.

Proof. We need to prove that, for any state σ, if $(P, \sigma) \to (\epsilon, \sigma', h)$, then $\sigma \models \varphi$ implies $\sigma' \models \psi$ and $h, [\sigma(now), \sigma'(now)] \models I$. Consider that the proof for the postcondition $\sigma' \models \psi$ has already been given in Theorem 4.1. We only give the proof for the invariant part here.

The proof is given by structural induction on P as follows.

- The proof for skip and $x := e$ is trivial.
- Cases binders b: For $b \equiv ch?x\{u\}$, according to the transition system, there exist some $d \geq 0$ and c such that $\sigma' = \sigma[now \mapsto \sigma(now) + d][x \mapsto c, u \mapsto 1]$ and h defined on $[\sigma(now), \sigma(now) + d]$ satisfies that $h(t) = \sigma[now \mapsto t]$ for each t in $[\sigma(now), \sigma(now)+d)$ and $h(\sigma(now) + d) = \sigma'$. Thus, from $\sigma \models \varphi$, $h, [\sigma(now), \sigma'(now)] \models (\exists x, u.\varphi)$ must hold (notice that now does not occur in assertions). The case for $b \equiv ch!e\{u\}$ can be proved similarly.
  For $b \equiv \&_q(b_1, \cdots, b_n)$, according to the transition system, there must exist some $d \geq 0$ such that $\sigma'(now) = \sigma(now) + d$, and for each b_i evolving to ϵ at termination, there must be $\sigma'(u_i) = 1$, and for any variable x that is not $mv(b)$, for any $t \in [\sigma(now), \sigma(now')]$, $h(t)(x) = \sigma(x)$. Thus $h, [\sigma(now), \sigma'(now)] \models \exists mv(b).\varphi$ hold.

- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle$: According to the transition system, there must exist $d \geq 0$ such that $\sigma' = \sigma[now \mapsto \sigma(now) + d, s \mapsto S(d)]$ and h defined over $[\sigma(now),$ $\sigma(now) + d]$ satisfies that for any o in the domain, $h(o) = \sigma[now \mapsto o, s \mapsto S(o - \sigma(now))]$, where S is the solution of the continuous with respect to $\sigma(s)$ as defined in the rule. According to the definition of Inv, then for any $o \in [\sigma(now), \sigma(now) + d]$, $h(o) \models Inv$, thus $h, [\sigma(now),\ \sigma'(now)] \models Inv$ hold. Plus the fact that $h, [\sigma(now),\ \sigma'(now)] \models (\exists s.\varphi)$, the result is proved.
- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle \trianglerighteq b \rightarrow Q$: According to the transition system, there are two cases for termination, by applying the fourth and the third transition rules for it respectively. For the first case, there must exist d such that $\sigma'(now) = \sigma(now) + d$, and for any variable x except for s and the ones in $mv(b)$, for any $o \in [\sigma(now), \sigma(now) + d]$, $h(o)(x) = \sigma(x)$. Plus the semantics of continuous, we have $h, [\sigma(now),$ $\sigma'(now)] \models \exists mv(b).(\exists s.\varphi) \wedge Inv$ proved. For the second case, there must exist d_1 such that $\sigma''(now) = \sigma(now) + d_1$, and for any variable x except for s and the ones in $mv(b)$, $\sigma''(x) = \sigma(x)$ and for any $o \in [\sigma(now), \sigma(now) + d]$, $h'(o)(x) = \sigma(x)$, and $\sigma'' \models (\exists mv(b).(\exists s.\varphi) \wedge Inv) \wedge \alpha$, and $(Q, \sigma'') \rightarrow (\epsilon, \sigma', h'')$, and $h = h'^\frown h''$. The fact is proved based on the inductive hypothesis on Q.
- Cases $P\|Q$, $P; Q$ and $\omega \rightarrow P$: According to the transition system, for $P\|Q$, suppose P and Q terminate at the same time, then there must exist σ_1, h_1, and σ_2, h_2 such that $(P, \sigma) \rightarrow (\epsilon, \sigma_1, h_1)$, $(Q, \sigma) \rightarrow (\epsilon, \sigma_2, h_2)$, $\sigma' = \sigma_1 \uplus \sigma_2$ and $h = h_1 \uplus h_2$. The fact is proved by induction hypothesis on P and Q. The other cases can be proved similarly without any essential difficulty.
Similarly, the rules for $P; Q$ and $\omega \rightarrow P$ can be proved by induction hypothesis, and we omit the details here.
- Case P^*: According to the transition system, we have

$$\sigma' = \sigma \quad h = \{\sigma(now) \mapsto \sigma'\}$$

or there exists an integer $k > 0$ such that $\sigma_k = \sigma'$, $h = h_1{}^\frown h_2{}^\frown \cdots {}^\frown h_k$, and a sequence of transitions as follows:

$$(P, \sigma) \rightarrow (\epsilon, \sigma_1, h_1), (P, \sigma_1) \rightarrow (\epsilon, \sigma_2, h_2)$$
$$\cdots, (P, \sigma_{k-1}) \rightarrow (\epsilon, \sigma_k, h_k)$$

For the first case, the fact holds trivially. For the second case, suppose the fact holds when $k < n$ for some $n > 0$, next we prove that the fact holds for $k = n$. According to the transition rule, we have

$$(P, \sigma_{n-1}) \rightarrow (\epsilon, \sigma_n, h_n), \quad \sigma_{n-1} \models \varphi$$
$$h_1{}^\frown \cdots {}^\frown h_{n-1}, [\sigma(now), \sigma_{n-1}(now)] \models I$$

By induction hypothesis on P, $\sigma_n \models \varphi$ and h_n, $[\sigma_{n-1}(now), \sigma_n(now)] \models I$ must hold. Then $h_1{}^\frown \cdots {}^\frown h_n, [\sigma(now), \sigma_n(now)] \models I$ is proved.

□

We then establish the following theorem stating that the time oblivious inference system is an over-approximation of the time aware inference system.

THEOREM 5.2. *Given a process P, if $\{\varphi\}\ P\ \{\psi, I\}$ can be deduced from the time oblivious inference system, then $\{\varphi\}\ P\ \{\psi, \lceil I\rceil^\frown\lceil I\rceil^0\}$ can be deduced from the time aware inference system.*

Proof. The proof is given by induction on the structure of P. The proof for most cases is direct, and below we present the proof for some cases as an illustration.

- Cases skip and $x := e$: The facts can be proved easily from the fact that for any formula φ, $\lceil\varphi\rceil^0 \Rightarrow \lceil\varphi\rceil^\frown\lceil\varphi\rceil^0$ holds.
- Cases binders b: By applying the time oblivious inference system, we have

$$\{\varphi\}\ ch?x\{u\}\ \{(\exists x, u.\varphi) \wedge u = 1, \exists x, u.\varphi\}$$

We need to prove that

$$\{\varphi\}\ ch?x\{u\}\ \{(\exists x, u.\varphi) \wedge u = 1, \lceil\exists x, u.\varphi\rceil^\frown\lceil\exists x, u.\varphi\rceil^0\}$$

can be proved by applying the time aware inference system. The fact is proved from the rule for $ch?x\{u\}$ and rule (**SHF**), and the fact that $\lceil\varphi\rceil^\frown\lceil\exists x, u.\varphi \wedge u = 1\rceil^0 \Rightarrow \lceil\exists x, u.\varphi\rceil^\frown\lceil\exists x, u.\varphi\rceil^0$. The cases for $b \equiv ch!e\{u\}$ and $b \equiv \&_q(b_1, \cdots, b_n)$ can be proved similarly.
- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle$: From the fact that

$$\lceil(\exists s.\varphi) \wedge B \wedge Inv\rceil^\frown\lceil(\exists s.\varphi) \wedge \neg B \wedge Inv\rceil^0$$
$$\Rightarrow \lceil(\exists s.\varphi) \wedge Inv\rceil^\frown\lceil(\exists s.\varphi) \wedge Inv\rceil^0$$

the fact for the continuous evolution is proved.
- Case $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle \trianglerighteq b \rightarrow Q$: By induction hypothesis on Q, we obtain $HF_1{}^\frown\lceil\psi_1\rceil^0 \Rightarrow \lceil I_1\rceil^\frown\lceil I_1\rceil^0$. Thus $HF_1 \Rightarrow \lceil I_1\rceil$ and $\psi_1 \Rightarrow I_1$ hold. Denote the postcondition and the history formula of $\langle \mathcal{F}(\dot{s}, s) = 0\&B\rangle \trianglerighteq b \rightarrow Q$ in Table 4 by ψ_2 and HF_2 respectively. It is easy to prove that $HF_2{}^\frown\lceil\psi_2\rceil^0 \Rightarrow \lceil(\exists mv(b).(\exists s.\varphi) \wedge Inv) \vee I_1\rceil^\frown\lceil(\exists mv(b).(\exists s.\varphi) \wedge Inv) \vee I_1\rceil^0$. The proof is done.
- Case $P\|Q$: By induction hypothesis on P and Q, we obtain the facts $HF_i{}^\frown\lceil\psi_i\rceil^0 \Rightarrow \lceil I_i\rceil^\frown\lceil I_i\rceil^0$ for $i = 1, 2$, thus $HF_i \Rightarrow \lceil I_i\rceil$ and $\psi_i \Rightarrow I_i$ hold. Based on these facts, the following implication is valid:

$$(((HF_1^\frown\lceil\psi_1\rceil) \wedge HF_2) \vee (HF_1 \wedge (HF_2^\frown\lceil\psi_2\rceil))$$
$${}^\frown\lceil\psi_1 \wedge \psi_2\rceil^0) \Rightarrow \lceil I_1 \wedge I_2\rceil^\frown\lceil I_1 \wedge I_2\rceil^0$$

The proof is finished.
- Case $P; Q$: By induction hypothesis on P and Q, we obtain the facts $HF_i{}^\frown\lceil\psi_i\rceil^0 \Rightarrow \lceil I_i\rceil^\frown\lceil I_i\rceil^0$ for $i = 1, 2$, thus $HF_i \Rightarrow \lceil I_i\rceil$ and $\psi_i \Rightarrow I_i$ hold. From the definition of $Inr(\cdot)$, it is easy to prove that $Inr(HF_1) \Rightarrow \lceil I_1\rceil$. The following implication is then valid:

$$Inr(HF_1)^\frown HF_2{}^\frown\lceil\psi_2\rceil^0 \Rightarrow \lceil I_1 \vee I_2\rceil^\frown\lceil I_1 \vee I_2\rceil^0$$

The proof is finished.

□

5.3. Comparison Between The Two Inference Systems

In the communication-based mechanism, the behavior of a process heavily relies on its environment. Especially in the synchronized setting, a communication of a process occurs only when its dual communication event in the environment gets ready, and it needs to exchange messages with the environment for the following process to proceed. The binder mechanism in bHCSP to some extent alleviates the dependence of a process upon its environment. It can be used to construct a system with communication fault tolerance. Using binders, a process can still behave in the correct way when some communication fails under some circumstance. We reflect this idea when defining the specifications of bHCSP, i.e. the specification of a process is defined without dependance on the environment. For instance, as shown by the rule of input $ch?x$ in both inference systems, the new value of x received from the partner is not considered in the postcondition.

In the time aware inference system, the interval property of a process is defined by an interval formula in the form of Duration Calculus. Compared with the time oblivious one, the main difference can be seen from the rule for sequential composition:

$$\frac{\{\varphi\}\ P\ \{\psi_1, \mathit{HF}_1\} \quad \{\psi_1\}\ Q\ \{\psi_2, \mathit{HF}_2\}}{\{\varphi\}\ P;Q\ \{\psi_2, \mathit{Inr}(\mathit{HF}_1)^\frown \mathit{HF}_2\}}$$

where the interval property of $P;Q$ is defined by the concatenation of the ones for (the internal of) P and Q in sequence. While in time oblivious inference system, the interval property of $P;Q$ is defined by a weaker invariant, i.e. the disjunction of the ones for P and Q. As shown by Theorem 5.2, the time aware inference system is more expressive than the time oblivious one: For any process P, if $\{\varphi\}\ P\ \{\psi, I\}$ is proved by the time oblivious inference system, then $\{\varphi\}\ P\ \{\psi, \lceil I\rceil^\frown\lceil I\rceil^0\}$ can be proved by the time aware inference system, while the inverse is not true.

However, in real applications of bHCSP, the time oblivious inference system is more preferred, due to the following two reasons:

- Due to the uncertain environment, the execution time of a bHCSP process with external communications involved is not known. As a result, the concatenation of two interval formulas is without time constraints between them. This makes the concatenation less meaningful in real applications. In many cases, we find that it is good enough to consider the invariant property that holds for all reachable states of a process.
- The last but not the least, for tool support, the FOL-based inference system is much easier to implement. Especially, with the help of existing SMT solvers for solving FOL formulas, more automation can be achieved for the proof. It is easier for the users. We will discuss this in more detail in Sec. 7.

Both of the above points are reflected in the verification of the same train control case study by applying the two inference systems in Sec. 7.

We have considered getting rid of the postcondition, or even both the precondition and the postcondition, with only the invariant left in the specification of bHCSP. But the expressiveness of the inference system will then become inadequate. As seen from the rules given in Table 5, for many constructs, the postcondition strictly implies the invariant. As a special evidence, the safety property of the case study cannot be proved with the simpler definition.

In the differential dynamic logic proposed by Platzer [46], the main specification triple is $A \to [P]B$. It corresponds to a Hoare triple for hybrid systems, stating that if A holds in the initial state, then for all states reachable by following the hybrid program P, B holds. Compared to our work, the B is not global for P and only records the invariant property that holds for the last atomic process in P.

6. BHCSP THEOREM PROVERS

Based on each of the inference systems, we implement a bHCSP theorem prover, which aims to verify whether a bHCSP process conforms to its specification in a machine-checkable way. The implementation of bHCSP theorem prover requires to embed the inference system of bHCSP in Isabelle/HOL. There are two different ways for the embedding: shallow or deep. The shallow embedding defines the assertions of bHCSP (i.e. FOL or DC formulas) by HOL predicates on process states or flows, while in deep embedding, it defines the assertions as new datatypes. We will adopt the approach of shallow embedding, to be able to apply the powerful proof tactics of Isabelle/HOL to conduct the proofs. The shallow embedding of bHCSP inference system includes the following aspects:

- Embedding of bHCSP syntax. We implement the bHCSP processes by a new datatype `bproc`, each constructor of which corresponds to a bHCSP construct;
- Embedding of bHCSP assertions. We define the FOL and DC formulas as predicates on states and flows. As a result, the FOL and DC formulas can be derived as specific Isabelle functions from `state` to *bool*, and from `flow` and timed interval `Intv` to *bool*, respectively;
- Embedding of bHCSP semantics. We define the meaning of `bproc` processes in terms of the operational semantics.
- Embedding of bHCSP inference systems. We define the inference rules as new theorems of Isabelle/HOL for the proofs of bHCSP.

At the end, all the theorems corresponding to the inference rules of bHCSP together constitute a verification condition generator for proving bHCSP specifications. The proof is performed according to the following process: first, by applying the bHCSP theorems, a bHCSP specification is transformed step by step to a set of DC or FOL formulas in the form of HOL predicates, i.e. *verification conditions*;

$$
\begin{aligned}
\mathsf{TR} \;=\;& \mathsf{MV}(t_1,T_1) \trianglerighteq {}^{0}\&_{\exists}(\mathsf{trd}!v\{u_v\}, \mathsf{trv}!v\{w_v\})^{7} \\
& \to (u_v = 1 \wedge w_v = 1 \to \\
& \quad (\mathsf{MV}(t_2,T_2) \trianglerighteq \&_{\exists}(\mathsf{dr}?x_a\{u_a\}, \mathsf{vc}?y_a\{w_a\}) \to \\
& \quad (w_a = 1 \to (VA(v,y_a) \to a := y_a; \\
& \qquad \neg VA(v,y_a) \to \mathsf{SC}); \\
& \quad u_a = 1 \wedge w_a = 0 \to (VA(v,x_a) \to a := x_a; \\
& \qquad \neg VA(v,x_a) \to \mathsf{SC}); \\
& \quad u_a = 0 \wedge w_a = 0 \to {}^{2}\text{skip});\ t_2 \geq T_2 \to \mathsf{SC}; \\
& u_v = 1 \wedge w_v = 0 \to \\
& \quad (\mathsf{MV}(t_2,T_2) \trianglerighteq \&_{\exists}(\mathsf{dr}?x_a\{u_a\}) \to \\
& \quad (u_a = 1 \to (VA(v,x_a) \to a := x_a; \\
& \qquad \neg VA(v,x_a) \to \mathsf{SC}); \\
& \quad u_a = 0 \to {}^{3}\text{skip}); t_2 \geq T_2 \to \mathsf{SC}; \\
& u_v = 0 \wedge w_v = 1 \to \\
& \quad (\mathsf{MV}(t_2,T_2) \trianglerighteq \&_{\exists}(\mathsf{vc}?y_a\{w_a\}) \to \\
& \quad (w_a = 1 \to (VA(v,y_a) \to a := y_a; \\
& \qquad \neg VA(v,y_a) \to \mathsf{SC}); \\
& \quad w_a = 0 \to {}^{4}\text{skip}); t_2 \geq T_2 \to \mathsf{SC}; \\
& u_v = 0 \wedge w_v = 0 \to {}^{1}\text{skip}); t_1 \geq T_1 \to \mathsf{SC}; \\
\mathsf{MV}\ (\mathsf{t},\mathsf{T}) \;=\;& t := 0; \langle \dot{s} = v, \dot{v} = a, \dot{t} = 1 \& t < T \rangle \\
\mathsf{SC} \;=\;& a := -c; \langle \dot{s} = v, \dot{v} = a \& v > 0 \rangle; a := 0
\end{aligned}
$$

TABLE 6. The model of **train**

$$
\begin{aligned}
\mathsf{DR} \;=\;& \text{wait } T_3; {}^{5}\&_{\exists}\mathsf{trd}?v_d\{u_v\};\ {}^{8}u_v = 1 \\
& \to (v_d \geq (v_{max} - cT_1 - cT_2) \\
& \qquad \to [\!]_{l \in [-c,0)} d_a := l; \\
& \quad v_d < (cT_1 + cT_2) \to [\!]_{l \in [0,c]} d_a := l; \\
& \quad v_d \in [cT_1 + cT_2, v_{max} - cT_1 - cT_2) \\
& \qquad \to [\!]_{l \in [-c,c]} d_a := l; \\
& \quad \&_{\exists}(\mathsf{dr}!d_a\{u_a\}, \mathsf{tick}?o\{u_c\}) \to \\
& \qquad {}^{12}(u_a = 1 \wedge u_c = 1 \to \text{skip}; \\
& \qquad u_a = 1 \wedge u_c = 0 \to \mathsf{tick}?o\{u_c\}; \\
& \qquad u_a = 0 \wedge u_c = 1 \to \text{skip}; \\
& \qquad u_a = 0 \wedge u_c = 0 \to \text{skip}) \\
& \quad \| \mathsf{CK}); \\
& u_v = 0 \to \text{skip} \\
\mathsf{CK} \;=\;& \text{wait } T_5; \mathsf{tick}!\checkmark \\
\mathsf{VC} \;=\;& \text{wait } T_4; {}^{6}\&_{\exists}\mathsf{trv}?v_r\{w_v\};\ {}^{9}w_v = 1 \\
& \to (v_r \geq (v_{max} - cT_1 - cT_2) \\
& \qquad \to r_a := -c; \\
& \quad v_r < (cT_1 + cT_2) \to r_a := c; \\
& \quad v_r \in [cT_1 + cT_2, v_{max} - cT_1 - cT_2) \\
& \qquad \to [\!]_{l \in [-c,c]} r_a := l; \\
& \quad \&_{\exists}(\mathsf{vc}!r_a\{w_a\}, \mathsf{tick}?o\{w_c\}) \to \\
& \qquad (w_a = 1 \wedge w_c = 1 \to \text{skip}; \\
& \qquad w_a = 1 \wedge w_c = 0 \to \mathsf{tick}?o\{w_c\}; \\
& \qquad w_a = 0 \wedge w_c = 1 \to \text{skip}; \\
& \qquad w_a = 0 \wedge w_c = 0 \to \text{skip}) \\
& \quad \| \mathsf{CK}); \\
& w_v = 0 \to \text{skip}
\end{aligned}
$$

TABLE 7. The models of **driver** and **VC**

and then, by applying proof tactics and rules of HOL, the validity of verification conditions, that is equivalent to the correctness of the original bHCSP specification, is proved.

From now on, we will call the two theorem provers implemented based on the inference systems in Sec. 4 and Sec. 5 the prover I and II respectively. We will separately apply the two provers to the train control example in next section, and moreover compare the proof results.

7. TRAIN CONTROL EXAMPLE

We apply our approach to the train control system depicted in Fig. 1. In Sec. 7.1, we construct the formal model for the whole system, including the train, the driver and the VC. In Sec. 7.2, we prove that the train is safe against denial-of-service security attack with respect to properties (F1) and (F2), without considering the control from VC and driver. In Sec. 7.3, we investigate the behavior of the whole train control system, especially, how the control parameter from the driver or VC will affect the train behavior.

7.1. Models of the Train Control System

Before giving the models, we introduce some variables and constants. The variables s, v and a represent the distance, the velocity and the acceleration of the train respectively. For the train, we assume that its acceleration a ranges over $[-c, c]$ for some $c > 0$, and the maximum speed is limited to be v_{max}.

The model of the train is given in Table 6. There are two auxiliary processes: MV (t, T) models that the train moves with velocity v and acceleration a for up to T time units, where t is the clock variable recording the moving time and T is the time limit; and SC defines the feedback control of the train when the services from the driver or the VC fail: it performs an emergency brake by setting a to be $-c$, and as soon as v is decreased to 0, resets a to be 0, thus the train keeps still finally. The main process TR models the movement of a train. The train first moves for at most T_1 time units, during which it is always ready to send v to the driver as well as the VC along trd and trv respectively. If neither of them responses within T_1, indicated by $t_1 \geq T_1$, the self control is performed. Otherwise, if at least one communication occurs, the movement is interrupted and a sequence of case analysis is followed to execute.

The first case, indicated by $u_v = 1$ and $w_v = 1$, represents that the driver as well as the VC succeed to receive v simultaneously. The train will wait for at most T_2 time units for receiving the new acceleration from the driver or the VC along dr and vc respectively, and during the waiting time, it continues to move with the original acceleration. It can be easily seen that the maximum time for keeping a same acceleration is $T_1 + T_2$, as a result, the maximum change of velocity is $cT_1 + cT_2$. Thus, in order to keep the velocity always in the safe range $[0, v_{max}]$, the new acceleration received is expected to satisfy the following *boundary condition* $VA(v, a)$:

$$
\begin{aligned}
&(v > v_{max} - cT_1 - cT_2 \Rightarrow -c \leq a < 0) \\
&\wedge (v < cT_1 + cT_2 \Rightarrow c \geq a \geq 0) \\
&\wedge (cT_1 + cT_2 \leq v \leq v_{max} - cT_1 - cT_2) \Rightarrow (-c \leq a \leq c)
\end{aligned}
$$

which implies the boundaries for setting a to be positive or negative. Otherwise, it will be rejected by the train and the self control is performed.

If both the driver and the VC fail to response within T_2, indicated by $t_2 \geq T_2$, the self control is performed. Otherwise, the following case analysis is taken: If the train receives a value (i.e. y_a) from VC, indicated by $w_a = 1$, then sets y_a to be the acceleration if it satisfies VA, otherwise, performs self control; if the train receives a value (i.e. x_a) from the driver but not from the VC, updates the acceleration similarly as above; if the train receives no value from both (in fact never reachable), the skip is performed.

The other three cases, indicated by $u_v = 1 \wedge w_v = 0$, $u_v = 0 \wedge w_v = 1$, and $u_v = 0 \wedge w_v = 0$, can be understood similarly.

Next we present the model of the environment of the train, i.e. the driver and VC. One possible implementation for driver and VC, denoted by DR and VC respectively, is given in Table 7. In process DR, the driver asks the velocity of the train every T_3 time units, and as soon as it receives v_d, indicated by $u_v = 1$, it computes the new acceleration as follows: if v_d is almost reaching v_{max} (by the offset $cT_1 + cT_2$), then chooses a negative in $[-c, 0)$ randomly; if v_d is almost reaching 0, then chooses a non-negative in $[0, c]$ randomly; otherwise, chooses one in $[-c, c]$ randomly. The train then sends the value being chosen (i.e. d_a) to the train, and if it fails to reach the train within T_5 (i.e. the period of the clock), it will give up. The auxiliary process clock is introduced to prevent deadlock caused by the situation when the driver succeeds to receive velocity v_d from the train but fails to send acceleration d_a to the train within a reasonable time (i.e. T_5 here). VC and DR have very similar structure, except that VC has a different period T_4, and it will choose $-c$ or c as the acceleration for the first two critical cases mentioned above.

Finally, the whole train control system can be modeled as the parallel composition: $\mathsf{SYS} = \mathsf{TR}^* \| \mathsf{DR}^* \| \mathsf{VC}^* \| \mathsf{CK}^*$. By using *, each component will be executed repeatedly.

7.2. Verification of The Train

We will prove that the train satisfies the safety properties (F1) and (F2) in an open environment, i.e. no matter whether the VC or the driver behaves in a correct manner or not. First of all, assume that the precondition of the train, denoted by φ_0, is

$$VA(v, a) \wedge 0 \leq v \leq v_{max} \wedge -c \leq a \leq c$$

which indicates that in the initial state, v and a satisfy the boundary condition and are both well-defined.

Secondly, we need to provide the differential invariants for differential equations occurring in TR. Consider the differential equation of $\mathsf{MV}(t_1, T_1)$, the precondition of it with respect to φ_0, denoted by φ_1, can be simply calculated, which is $\varphi_0 \wedge t_1 = 0$. By applying the method proposed in [11], we obtain a candidate for the differential invariant of the differential equation with respect to the initial state φ_1, which is

$$\begin{aligned}&\big(0 \leq t_1 \leq T_1\big)\\ &\bigwedge \big(a < 0 \Rightarrow (v \geq cT_2 + (at_1 + cT_1)) \wedge (v \leq v_{max})\big)\\ &\bigwedge \big(a \geq 0 \Rightarrow (v \leq v_{max} - cT_2 + (at_1 - cT_1)) \wedge (v \geq 0)\big)\end{aligned}$$

denoted by Inv_1. It is a conjunction of three parts, which can be explained intuitively as follows: (1) t_1 is always in the range $[0, T_1]$; (2) if a is negative (thus v is decreasing), then v must be greater or equal than cT_2 plus the maximum possible decrease of the velocity rate in the remaining $T_1 - t_1$ time units, which is $-a(T_1 - t_1) \leq at_1 + cT_1$, and meanwhile $v \leq v_{max}$; and (3) on the contrary, if a is positive (thus v is increasing), then v must be less or equal than $v_{max} - cT_2$ minus the maximum possible increase of the velocity rate in the remaining $T_1 - t_1$ time units, which is $a(T_1 - t_1) \leq cT_1 - at_1$, and meanwhile $v \geq 0$. Obviously, this invariant is strong enough for guaranteeing $cT_2 \leq v \leq v_{max} - cT_2$ after the continuous escapes no matter whether a is in $[-c, c]$. Similarly, we can calculate the differential invariant of the differential equation occurring in $\mathsf{MV}(t_2, T_2)$, which is

$$\begin{aligned}&\big(0 \leq t_2 \leq T_2\big)\\ &\bigwedge \big(a < 0 \Rightarrow (v \geq 0 + (at_2 + cT_2)) \wedge (v \leq v_{max})\big)\\ &\bigwedge \big(a \geq 0 \Rightarrow (v \leq v_{max} + (at_2 - cT_2)) \wedge (v \geq 0)\big)\end{aligned}$$

denoted by Inv_2. This invariant is strong enough for guaranteeing $0 \leq v \leq v_{max}$ after the continuous escapes. Finally, the differential invariant of the differential equation of SC is $0 \leq v \leq v_{max}$, and we denote it by Inv_3.

Next, to prove (F1) and (F2), we can prove the following facts instead:

- Locations 1, 2, 3, 4 are not reachable for TR^*;
- Throughout the execution of TR^*, the invariant $0 \leq v \leq v_{max}$ always holds.

By applying the bHCSP provers I and II respectively, we have proved the following theorems for the train,

$$\{\varphi_0\}\ \mathsf{TR}^*\ \{\varphi_0, \lceil 0 \leq v \leq v_{max}\rceil ^\frown \lceil 0 \leq v \leq v_{max}\rceil^0\}$$

$$\{\varphi_0\}\ \mathsf{TR}^*\ \{\varphi_0, 0 \leq v \leq v_{max}\}$$

indicating that $0 \leq v \leq v_{max}$ always holds for the train. According to the method introduced in Section 4.2, we obtain the following fact for location 1[4],

$$\vdash \mathsf{TR}^*, 1, \varphi_0 \blacktriangleright (\mathbf{u_v} \vee \mathbf{w_v}) \wedge (\neg \mathbf{u_v} \wedge \neg \mathbf{w_v})$$

which is not satisfiable, thus location 1 is never reachable. Similarly, we can deduce that locations 2, 3, 4 are not reachable as well.

Comparison Between Two Inference Systems We have proved the equivalent results for the train by using the two bHCSP provers respectively. The length of the proof in the prover I is about 900 lines of code (loc), while in the prover II about 300 loc. The proof consists of a sequence of

[4]For simplicity, we use the boldface of an acknowledgment variable to represent the corresponding formula, e.g., $\mathbf{u_v}$ for $u_v = 1$.

rule applications, which can be classified to two kinds: the inference rules of bHCSP, and the rules for proving logical formulas. The proof of the second kind is the main reason that leads to the different results of the two provers. The formulas consist of FOL formulas and DC formulas in the prover I, while only FOL formulas in prover II. In prover II, most of the formulas can be proved automatically by calling the proof tactics of Isabelle/HOL. Especially, the tool `sledgehammer`, which is a certified integration of third-party automated theorem provers and SMT solvers including Alt-Ergo, Z3, CVC3, etc, is frequently used to search the proof of formulas.

7.3. Analysis of The Train Control System

We can continue to investigate the behavior of the whole control system SYS, which is a closed system. This needs to take the communications between the components of SYS into account. Consider the first loop of execution of each component, by instantiating the values of the time parameters, e.g. T_1, T_3, T_4 *etc*, in different ways, we obtain different results about the cooperation between the components.

Suppose $T_4 = \frac{1}{2}T_3 < T_1$ and $T_4 = T_5$ hold. TR and VC cooperate to execute, and the following sequence of events will occur:

$$T_4 \cdot \mathsf{trv}\dagger V_1 \cdot \mathsf{vc}\dagger A_1 \cdot \mathsf{tick}\dagger\checkmark$$

in which, the variables V_1 records the value of v at the interrupting point and A_1 the new acceleration provided by VC. In sequence, the communication along trv occurs first at time T_4, with the velocity V_1 being sent from TR to VC; then the communication along vc occurs immediately, with an acceleration A_1 being sent from VC to TR; and at last the communication along tick occurs. The execution of TR$\|$VC terminates and takes T_4 time units to complete.

Suppose $T_4 = T_3 = T_5 < T_1$ holds. TR, VC and DR cooperate to execute, and one possible case for the occurring event sequence is:

$$T_3 \cdot \mathsf{trv}\dagger V_1 \cdot \mathsf{trd}\dagger V_1 \cdot \mathsf{vc}\dagger A_1 \cdot \mathsf{dr}\dagger A_2 \cdot \mathsf{tick}\dagger\checkmark \cdot \mathsf{tick'}\dagger\checkmark$$

in which, the variables V_1 and A_1 are defined as above, and A_2 represents the new acceleration provided by DR. We rename the channel tick from either VC or DR as tick' to avoid the sharing of the same input or output channels in different components. The execution of TR$\|$VC$\|$DR terminates and takes T_3 time units to complete.

We can continue to consider the multiple loops of execution, and obtain some results for the behavior of the whole system SYS.

8. CONCLUSION AND FUTURE WORK

This paper proposes a formal modeling language, that is a combination of hybrid CSP and binders from quality calculus, for expressing hybrid systems with communication fault tolerance. With the linguistic support, it is able to build a safe hybrid system that behaves in a reasonable manner in the presence of communication failure. As a result, when the service from the controllers fails due to communication failure, the physical system itself is able to provide feedback control, to meet the safety requirements.

The paper develops two different inference systems for verifying the safety of such systems, and subsequently implement two theorem provers based on them. In the first approach, the interval property of a bHCSP process is specified by an interval temporal logic formula, which results in an expressive reasoning system but meanwhile brings the big proof burden in the corresponding prover. In the second approach, the interval property is simplified to an invariant property defined by first-order logic, that holds for all reachable states of the process. Although, the expressivity is less than the first one, it enables more automation in the proof thus is more preferred in real applications. Furthermore, as indicated by the application to the train control case study, the second approach can actually achieve the same result as the first one in many cases.

Future Work We will apply the framework based on bHCSP to investigate more practical hybrid systems. In [55], we modelled and verified the moving scenarios of Chinese Train Control System (CTCS) with respect to CTCS requirement specification, and in [56], we applied different formal methods to the verification of a descent guidance control program of a lunar lander. In both work, the systems are assumed to always have well-behaved communications between the continuous plant and the discrete controllers. The assumptions can be loosen in the framework proposed in this paper. On the other hand, as we mentioned at the beginning, we hope eventually to apply this framework to the model-based design of cyber-physical systems. However, this work can only be considered as the first step of the model-based design methodology. We will continue to study model transformations from the abstract bHCSP models to more concrete models, and to the implemental code at the end. In this process, the discretization of continuous evolution, and the extra efforts brought by the complex interactions between continuous plants and discrete computation via communications, are the key problems to be studied.

REFERENCES

[1] Alur, R., Courcoubetis, C., Henzinger, T. A., and Ho, P. (1992) Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid Systems, LNCS 736*, pp. 209–229. Springer Berlin Heidelberg.

[2] Lafferrierre, G., Pappas, G. J., and Yovine, S. (2001) Symbolic reachability computation for families of linear vector fields. *Journal of Symbolic Computation*, **32**, 231–253.

[3] Alur, R., Dang, T., and Ivancic, F. (2006) Predicate abstraction for reachability analysis of hybrid systems. *ACM Trasactions on Embedded Computing Systems*, **5**, 152–199.

[4] Asarin, E., Bournez, O., Dang, T., and Maler, O. (2000) Approximate reachability analysis of piecewise-linear

dynamical systems. *Hybrid Systems: Computation and Control (HSCC), LNCS 1790*, PA, USA, March 23-25, pp. 21–31. Springer Berlin Heidelberg.

[5] Gulwani, S. and Tiwari, A. (2008) Constraint-based approach for analysis of hybrid systems. *Computer Aided Verification (CAV), LNCS 5123*, Princeton, NJ, USA, July 7-14, pp. 190–203. Springer Berlin Heidelberg.

[6] Gan, T., Chen, M., Dai, L., Xia, B., and Zhan, N. (2015) Decidability of the reachability for a family of linear vector fields. *ATVA 2015*, Lecture Notes in Computer Science, **9364**, pp. 482–499.

[7] Gan, T., Chen, M., Li, Y., Xia, B., and Zhan, N. Computing reachable sets of linear vector fields revisited. *ECC 2016*. to appear.

[8] Prajna, S. and Jadbabaie, A. (2004) Safety verification of hybrid systems using barrier certificates. *Hybrid Systems: Computation and Control (HSCC), LNCS 2993*, Philadelphia, PA, USA, March 25-27, pp. 477–492. Springer.

[9] Sankaranarayanan, S., Sipma, H. B., and Manna, Z. (2007) Constructing invariants for hybrid systems. *Formal Methods in System Design*, **32**, 25–55.

[10] Platzer, A. and Clarke, E. M. (2008) Computing differential invariants of hybrid systems as fixedpoints. *Computer Aided Verification (CAV), LNCS 5123*, Princeton, NJ, USA, pp. 176–189. Springer-Verlag Berlin, Heidelberg.

[11] Liu, J., Zhan, N., and Zhao, H. (2011) Computing semi-algebraic invariants for polynomial dynamical systems. *The ninth ACM international conference on Embedded software (EMSOFT)*, Taipei, Taiwan, pp. 97–106. ACM New York, USA.

[12] Dai, L., Gan, T., Xia, B., and Zhan, N. Barrier certificate revisited. *J. of Symbolic Computation* , **?** to appear.

[13] Asarin, E., Bournez, O., Dang, T., Maler, O., and Pnueli, A. (2000) Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, **88**, 1011–1025.

[14] Lygeros, J., Godbole, D. N., and Sastry, S. (2000) A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, **88**, 949–970.

[15] Taly, A., Gulwani, S., and Tiwari, A. (2009) Synthesizing switching logic using constraint solving. *International Journal on Software Tools for Technology Transfer*, **13**, 519–535.

[16] Sturm, T. and Tiwari, A. (2011) Verification and synthesis using real quantifier elimination. *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, San Jose, California, USA, June 8-11, pp. 329–336. ACM.

[17] Zhao, H., Zhan, N., Kapur, D., and Larsen, K. (2012) A "hybrid" approach for synthesizing optimal controllers of hybrid systems: A case study of the oil pump industrial example. *FM 2012*, Lecture Notes in Computer Science, **7436**, pp. 471–485.

[18] Zhao, H., Zhan, N., and Kapur, D. (2013) Synthesizing switching controllers for hybrid systems by generating invariants. *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, Lecture Notes in Computer Science, **8051**, pp. 354–373.

[19] Zhang, S. (2008) *CTCS-3 Technology Specification*. China Railway Publishing House. in Chinese.

[20] Wang, S., Nielson, F., and Nielson, H. R. (2014) Denial-of-service security attack in the continuous-time world. *Formal Techniques for Distributed Objects, Components, and Systems (FORTE), LNCS 8461*, Berlin, Germany, June 3-5, pp. 149–165. Springer Berlin Heidelberg.

[21] He, J. (1994) From CSP to hybrid systems. *A classical mind*, pp. 171–189. Prentice Hall International (UK) Ltd.

[22] Zhou, C., Wang, J., and Ravn, A. P. (1996) A formal description of hybrid systems. *Hybrid systems III : verification and control, LNCS 1066*, pp. 511–530. Springer Berlin Heidelberg.

[23] Nielson, H. R., Nielson, F., and Vigo, R. (2013) A calculus for quality. *Formal Aspects of Component Software (FACS), LNCS 7684*, CA, USA, September 12-14, pp. 188–204. Springer Berlin Heidelberg.

[24] Henzinger, T. and Sifakis, J. (2006) The embedded systems design challenge. *Formal Methods (FM), LNCS 4085*, August 21-27, pp. 1–15. Springer, Hamilton, Canada.

[25] Lee, E. A. (2008) Cyber physical systems: Design challenges. *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, May. Invited Paper.

[26] Sangiovanni-Vincentelli, A. (2007) Quo Vadis SLD: Reasoning about trends and challenges of system-level design. *Proceedings of the IEEE*, **95**, 467–506.

[27] Manna, Z. and Pnueli, A. (1993) Verifying hybrid systems. *Hybrid Systems, LNCS 736*, pp. 4–35. Springer Berlin Heidelberg.

[28] Henzinger, T. A. (1996) The theory of hybrid automata. *Annual IEEE Symposium on Logic in Computer Science (LICS)*, pp. 278–292.

[29] Alur, R. and Dill, D. L. (1994) A theory of timed automata. *Theoretical Computer Science*, **126**, 183–235.

[30] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995) The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, **138**, 3–34.

[31] Prajna, S. and Jadbabaie, A. (1994) Decidability of hybrid systems with rectangular differential inclusions. *Computer Aided Verification (CAV), LNCS 818*, California, USA, June 21-23, pp. 95–104. Springer Berlin Heidelberg.

[32] Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P. (1995) What's decidable about hybrid automata? *Proceedings of the twenty-seventh Annual ACM Symposium on Theory of Computing (STOC)*, Las Vegas, USA, pp. 373–382. ACM, New York, USA.

[33] Clarke, E. M., Fehnker, A., Han, Z., Krogh, B. H., Ouaknine, J., Stursberg, O., and Theobald, M. (2003) Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. J. Found. Comput. Sci.*, **14**, 583–604.

[34] Castelan, E. and Hennet, J. (1993) On invariant polyhedra of continuous-time linear systems. *IEEE Trans. Autom. Control*, **38**, 1680–1685.

[35] Rodríguez-Carbonell, E. and Tiwari, A. (2005) Generating polynomial invariants for hybrid systems. *Hybrid Systems: Computation and Control (HSCC), LNCS 3414*, Zurich, Switzerland, March 9-11, pp. 590–605. Springer Berlin Heidelberg.

[36] Sankaranarayanan, S., Dang, T., and Ivančić, F. (2008) A policy iteration technique for time elapse over template polyhedra. *Hybrid Systems: Computation and Control (HSCC), LNCS 4981*, St. Louis, MO, USA, April 22-24, pp. 654–657. Springer Berlin Heidelberg.

[37] Sankaranarayanan, S., Sipma, H. B., and Manna, Z. (2004) Constructing invariants for hybrid systems. *Formal Methods in System Design*, **32**, 25–55.

[38] Sankaranarayanan, S. (2010) Automatic invariant generation for hybrid systems using ideal fixed points. *Hybrid Systems: Computation and Control (HSCC)*, Stockholm, Sweden, pp. 221–230. ACM, New York, NY, USA.

[39] Platzer, A. and Clarke, E. M. (2009) Computing differential invariants of hybrid systems as fixedpoints. *Formal Methods in System Design*, **35**, 98–120.

[40] Yang, Z., Lin, W., and Wu, M. (2015) Exact safety verification of hybrid systems based on bilinear SOS representation. *ACM Trans. Embed. Comput. Syst.*, **14**, 1–19.

[41] Rebiha, R., Matringe, N., and Moura, A. V. (2012) Transcendental inductive invariants generation for non-linear differential and hybrid systems. *Hybrid Systems: Computation and Control (HSCC)*, Beijing, China, April 17-19, pp. 25–34. ACM, New York, NY, USA.

[42] Goubault, E., Jourdan, J.-H., Putot, S., and Sankaranarayanan, S. (2014) Finding non-polynomial positive invariants and Lyapunov functions for polynomial systems through Darboux polynomials. *2014 American Control Conference (ACC)*, Portland, Oregon, USA, June 4-6, pp. 3571–3578.

[43] Ghorbal, K. and Platzer, A. (2014) Characterizing algebraic invariants by differential radical invariants. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS), LNCS 8413*, Grenoble, France, April 5-13, pp. 279–294. Springer Berlin Heidelberg.

[44] Sankaranarayanan, S. (2016) Change-of-bases abstractions for non-linear hybrid systems. *Nonlinear Analysis: Hybrid Systems*, **19**, 107 – 133.

[45] Liu, J., Zhan, N., Zhao, H., and Zou, L. (2015) Abstraction of elementary hybrid systems by variable transformation. *Formal Methods (FM), LNCS 9109*, Oslo, Norway, June 24-26, pp. 360–377. Springer Berlin Heidelberg.

[46] Platzer, A. (2010) Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. and Comput.*, **20**, 309–352.

[47] Platzer, A. and Quesel, J. (2009) European Train Control System: A case study in formal verification. *Formal Methods and Software Engineering (ICFEM), LNCS 5885*, Shanghai, China, November 28 - December 1, pp. 246–265. Springer Berlin Heidelberg.

[48] Banach, R., Butler, M., Qin, S., Verma, N., and Zhu, H. (2015) Core Hybrid Event-B I: Single Hybrid Event-B machines. *Science of Computer Programming*, **105**, 92 – 123.

[49] Liu, J., Lv, J., Quan, Z., Zhan, N., Zhao, H., Zhou, C., and Zou, L. (2010) A calculus for hybrid CSP. *Programming Languages and Systems (APLAS), LNCS 6461*, Rio de Janeiro, Brazil, December 9-12, pp. 1–15. Springer.

[50] Zhan, N., Wang, S., and Zhao, H. (2013) Formal modelling, analysis and verification of hybrid systems. *Unifying Theories of Programming and Formal Engineering Methods, LNCS 8050*, Shanghai, China, August 26-30, pp. 207–281. Springer Berlin Heidelberg.

[51] Wang, S., Zhan, N., and Guelev, D. (2012) An assume/guarantee based compositional calculus for hybrid CSP. *Theory and Applications of Models of Computation (TAMC), LNCS 7287*, Beijing, China, May 16-21, pp. 72–83. Springer Berlin Heidelberg.

[52] Nielson, H. R. and Nielson, F. (2013) Probabilistic analysis of the quality calculus. *International Conference FMOODS/FORTE 2013, LNCS 7892*, Florence, Italy, June 3-5, pp. 258–272. Springer Berlin Heidelberg.

[53] Zhou, C., Hoare, C., and Ravn, A. P. (1991) A calculus of durations. *Information Processing Letters*, **40**, 269–276.

[54] Zhou, C. and Hansen, M. (2004) *Duration Calculus — A Formal Approach to Real-Time Systems* Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag Berlin Heidelberg.

[55] Zou, L., Lv, J., Wang, S., Zhan, N., Tang, T., Yuan, L., and Liu, Y. (2013) Verifying Chinese train control system under a combined scenario by theorem proving. *VSTTE, LNCS 8164*, Menlo Park, CA, USA, May 17-19, pp. 262–280. Springer Berlin Heidelberg.

[56] Zhao, H., Yang, M., Zhan, N., Gu, B., Zou, L., and Chen, Y. (2014) Formal verification of a descent guidance control program of a lunar lander. *Formal Methods (FM), LNCS 8442*, May 12-16, pp. 733–748. Springer Berlin Heidelberg, Singapore.