IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 14, 2021, accepted June 28, 2021, date of publication July 8, 2021, date of current version July 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095768

Hybrid Modeling and Model Transformation of AADL for Verifying the Properties of CPS Space-Time Compositions

XIAOYING CHEN[1], YI ZHU[1,2], (Member, IEEE), YU ZHAO[1], JINYONG WANG[2], AND ANARBEKOV ALTYNBEK[1]
[1]College of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China
[2]Key Laboratory of Safety-Critical Software, Ministry of Industry and Information Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

Corresponding author: Yi Zhu (zhuy@jsnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62077029, in part by the Applied Basic Research Program of Xuzhou under Grant KC19004, in part by the Graduate Science Research Innovation Program of Jiangsu Province under Grant KYCX20_2380, in part by the Graduate Science Research Innovation Program of Jiangsu Province under Grant KYCX20_2384, and in part by the Nanjing University of Aeronautics and Astronautics for Innovation Foundation of Fundamental Research Funds under Grant NJ2020022.

ABSTRACT The wide application of Cyber Physical System (CPS) makes the security of CPS more and more concerned. As the key factors affecting the safety of CPS, space and time have also become the current research hotspot. The space and time safety of CPS requires that CPS arrives at the specified place at the specified time, time and space should meet the safety requirements of the CPS in the current CPS environment. We call the behavior space-time compositions. In order to solve the problem that CPS lacks the method of modeling and verification of space-time compositions, a hybrid Architecture Analysis & Design Language (AADL) modeling and model transformation method for CPS space-time compositions verification is proposed. Firstly, space-time description capability is extended in the AADL behavior annex and Hybrid AADL (HAADL) is proposed. Secondly, differential equations and space-time compositions vector are introduced in Process Algebra to propose Hybrid Space-Time Communication Sequential Processes (HS-TCSP). Furthermore, the Hybrid AADL is transformed to HS-TCSP. Finally, an example of an aircraft collision avoidance system is used to verify the effectiveness of the method.

INDEX TERMS AADL, cyber physical system, formalization, process algebra.

I. INTRODUCTION

In recent years, with the rapid development of computer and network technology, the application of Cyber Physical System (CPS) has covered all aspects of our production and life. The issue of CPS verification has gradually become a research hot spot. The CPS verification process can both ensure that the system meets customer's requirements and verify whether the system meets certain specifications [1]. In many CPS scenarios with high safety factors, such as aircraft flight control systems, car control systems, railway control systems, etc., the correctness and safety verification of the system is particularly important. In 2011, a bullet train D301 from Beijing South Railway Station to Fuzhou and D3115 from Hangzhou Station to Fuzhou South Railway Station rear-ended on the Ningbo-Wenzhou Line in Wenzhou. It has been confirmed that a total of six carriages derailed in the accident. It killed 40 people, injured 172 and caused great damage. The reason is that due to the serious defect of the equipment of the train control center, the equipment of the train control center still carries out control output according to the status of no vehicle occupation when there are actual vehicles occupied in the subsequent periods. It causes the interval signal machine controlled by the equipment of the train control center of Wenzhou South Railway Station to be upgraded wrongly and remain green [2]. The train control system is a typical CPS. The train control system violates the space-time compositions and causes serious consequences. In addition, in 2015, passengers were sued for compensation after a bullet train broke down nine hours late. In 2016,

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

19 high-speed trains on the Wuhan section of the Beijing-Guangzhou high-speed railway were delayed due to equipment failure, resulting in a total of 19 high-speed trains being delayed. In 2021, NASA's helicopter on Mars, the Wit, has suffered a malfunction during its flight, and images showed it bumping and bumping throughout the flight. The initial flight was smooth, but at 54 seconds, there was a malfunction. Wit relied on images taken by a camera on its belly to navigate. The camera transmitted the images to flight controllers, but at 54 seconds there was a glitch in the camera's transmission, missing an image and causing every image transmitted after that to be incorrectly timed. Although the flight was successful in the end, the security of space-time compositions has a certain impact on the flight. None of these accidents satisfy the space-time compositions constraint. The security problem of CPS will have serious consequences. Therefore, it is necessary to study the security of space-time compositions for CPS.

AADL is a SAE standard [3], AADL has received widespread attention since its introduction. The language is based on a hierarchical structure design method of system components, which can abstractly model hardware and software in a unified manner, and supports the development of highly adaptive systems. At this stage, the use of AADL for CPS modeling has also achieved some results. Cloud cyber physical system was modeled and analyzed based on AADL in [4]. The method of integrating model checking into the design process and generating a timed automata model according to the advanced specifications in AADL was mentioned in [5], thereby integrating model checking technology into the CPS design process. However, AADL does not have enough ability to describe the heterogeneous characteristics and concurrence of CPS. Therefore, [6] proposed a method for constructing a new AADL sublanguage AADL+ for modeling the continuous behavior of CPS and the interaction between network components and physical components. Reference [7] proposed a CPS integrated modeling framework description method based on extended AADL to achieve a unified description of computing entities, physical entities and interactive entities. However, these methods focus on the continuous and interactive behavior of CPS and do not focus on the space-time compositions of CPS. In addition, AADL is semi-formalized, so model checking or theorem proving can't prove it directly. References [8] and [9] combined AADL with Z language to verify its reliability. An AADL model checking method based on time abstract state machine was proposed in [10]. The physical system modeling and cyber system modeling were separated, and the Modelica-AADL interface was designed to combine the two, and finally converted into a probabilistic hybrid automaton in[11].

Communicating Sequential Process (CSP) is a formal method suitable for distributed concurrent software specifications and design, established by Hoare [12]. In 1986, CSP in real time was extended and Timed Communicating Sequential Process (TCSP) was proposed [13]. Process Algebra is a formal method to solve the communication problem of concurrent systems, which can describe the concurrency, synchronization and asynchrony of events in CPS. Variables can be divided into continuous variables and discrete variables according to whether their values are continuous. Variables that can be arbitrarily valued within a certain interval are called continuous variables, and their values are continuous. Conversely, those whose values can only be calculated in natural numbers or integer units are discrete variables. Continuous variables need to be described in the CPS. However, TCSP is a discrete model and has only the ability to describe time performance. Therefore, the TCSP need to be extended to describe space-time compositions behavior in CPS.

Based on this, in order to describe the CPS (as shown in Figure 1), space and time factors are extended in the AADL Behavior Annex and HAADL is proposed. Based on the TCSP, the HS-TCSP is proposed by extending the differential equation and space-time compositions vector. Furthermore, the extended space-time compositions parts of HS-TCSP are analyzed to verify the security of space-time compositions property in CPS environment. Finally, the transformation rules are defined to transform the HAADL into HS-TCSP, thereby verifying the correctness of the extended HAADL. An aircraft collision avoidance system is used to illustrate the effectiveness of this method as an example.

II. RELATED WORK

CPS is a multi-dimensional complex system that integrates computing, network and physical environments. The modeling of CPS should consider space-time compositions. Assume that time and space are consistent, the space-time compositions is that whether the current space-time behavior meets the security requirements of the CPS environment. In other words, whether the current time and space make the system safe in the current CPS scenario. If the time and space security of CPS can't be guaranteed, and it will bring serious consequences. There have been relevant studies on the modeling and verification of CPS space-time compositions property.

Reference [14] combined the domain environment model and the runtime verification process by defining a series of rules. This method studied the real-time impact of environmental changes on system parameters, but did not take the position of the system into account. References [15] and [16] conducted modeling research on how to interact various parts of the CPS under time constraints. HPCCS was proposed on Calculus of Communication System(CCS) and AADL behavior annex was extended to describe random actions in [17]. Most of these traditional CPS modeling and verification methods were limited to time-domain analysis and fail to consider the influence of unified changes in space and time on CPS, thus causing some space-time security problems in CPS. Reference [18] focused on the modeling and verification of CPS processes based on physical and temporal properties, which doesn't take space into account. A framework of space-time event model based on physical state was proposed in [19] which reflected the space-time

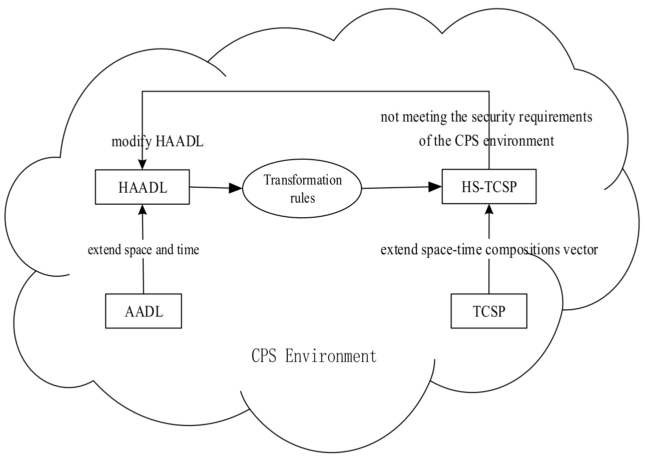
TABLE 1. Comparison of related works.

Reference	DB	CB	T	P	VE	STCV	FD
Ref[15]	√	√	√	×	√	×	Process Calculus
Ref[16]	√	×	√	×	√	×	Time Automaton
Ref[17]	√	√	√	×	×	×	CCS
Ref[18]	√	×	√	×	√	×	BPMN4CPS
Ref[19]	√	√	√	√	×	×	CPS Event Model
Ref[20]	√	√	√	√	×	×	Petri Net
Ref[22]	√	√	√	√	×	×	QCSP
our work	√	√	√	√	√	√	HS-TCSP

Note: √ :support; ×:no support; DB:discrete behavior; CB:continuous behavior; T:time; P:position; STCV:space-time compositions verification; VE:verification; FD: formal description;

features of CPS and the dynamic changes of environment. But it did not take into account the security of space-time compositions. Reference [21] focused on how the probability of data collisions occurrence through a shared constrained network may be reduced. Reference [22] proposed a process algebra called Communicating Sequential Process with Qualitative calculus (QCSP) as a formal language for modeling the spatio-temporal behavior. The key feature of this algebra was that it can reason about the space relations between agents by a finite set of binary relations that obey certain mathematical conditions. While it focused on spatio-temporal modeling and the spatio-temporal behavior at a point in time, we pay attention to both the point in time and the continuity with a whole period of time. CPS can be detected in real time during working to ensure the safety of CPS. Our work is the follow-up work of the above work, which has a certain significance to ensure the space-time security of CPS.

There are also a lot of works related to modeling and verification of CPS using AADL. However, AADL lacks the ability to model continuous behavior in CPS. Besides, AADL also lacks formal semantics. Reference [23] introduced the Hybrid Annex for continuous-time modeling, fulfilling the need for integrated modeling of the computing system along with its physical environment in their respective domains. Reference [24] proposed an approach to build formal semantics to AADL's software component models. A hierarchical and compositional modeling approach based on AADL was proposed to solve the tight coupling between physical and cyber world in [25]. For the problem that core AADL lacked of a mechanism for modeling continuous evolution of physical processes [26] presented formal semantics of the synchronous subset of AADL models annotated with Hybrid Annex specifications using HCSP and verified correctness of AADL models using Hybrid Hoare Logic (HHL) prover. Besides [27] investigated behavior modeling and formal verification of Chinese Train Control System Level 3 (CTCS-3) using AADL.



modify HAADL
not meeting the security requirements of the CPS environment
HAADL
Transformation rules
HS-TCSP
extend space and time
extend space-time compositions vector
AADL
TCSP
CPS Environment

FIGURE 1. Technology roadmap.

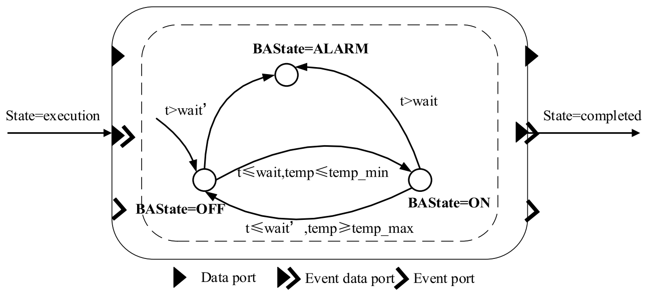


FIGURE 2. Graphical AADL model for automatic temperature control system.

In addition, an encounter model is formalised to identify all of the potential collision scenarios. The collision scenarios could be induced by a resolution advisory without considering the downstream consequences in the surrounding traffic in [28]. In [29], a causal encounter model is proposed to extend the TCAS logic considering the horizontal resolution manoeuvres. Based on the generated state space, the model developed in the graphical modeling and analysis software, not only provides a better comprehension of the potential collision occurrences for risk assessment by representing the cause-effect relationship of each action, but also aids the pilots in the involved aircraft to make a cooperative and optimal option.

Compared with the above research work, this paper focuses on the space-time property of CPS, especially the relationship between space and time compositions, so as to ensure the space and time safety of CPS. The space-time property is extended on the semi-formal AADL. Besides, the space-time compositions vector and differential equations are extended on TCSP to ensure the space-time compositions security of CPS.

III. HYBRID AADL

In this section, the precise definition of HAADL is shown. space and time factors are extended in the AADL Behavior Annex. In addition to the extension to AADL Behavior Annex, AADL Physical Annex is also briefly mentioned.

A. AADL BASICS

The semantics are defined in AADL to describe the components of the system's software and hardware architecture and the properties of the system. There are two kinds of extensions of AADL, annex and property set [30]. AADL Behavior Annex is a state transition system, including state and state transition. Behavior Annex is used to describe the behaviors of a component such as a thread, which includes the execution behaviors of the component, dispatch behavior of input/output port and so on. We extend the behavior annex. The extended behavior annex is compatible with other parts of AADL. At present, AADL modeling tools support the description of behavior annex. For different application levels of AADL, various simulation tools include: integrated development tool OSATE, schedulable analysis tool Cheddar and automatic code generation tool Ocarina etc.

B. BEHAVIOR ANNEX

The behavior annex mainly includes three parts: *variables*, *states* and *transitions*. All local variables used in the current behavior annex are declared in *variables*. Local variables can be used to save intermediate results within the scope of the current behavior annex. State transition includes original state, destination state, transition conditions and execution actions [31]. State transition defines trigger conditions and execution actions after transition. Conditions and actions mainly include receiving/sending data, calling subprogram, asynchronous access, execution time, delay time, etc. In addition, the transition supports more complex behavior descriptions through hierarchies and concurrent states.

Abstract description of AADL Behavior Annex:

The AADL Behavior Annex can be defined as a triple

$BA = (Vars, States, Transition)$

Vars is a set of variables, *States* is a set of states, and *Transition* is a set of state transitions. The state transitions include: *Source*, *Destination*, *Guard*, *Action*.

$Transition = Source\ [<guard>] \rightarrow Destination\ [\{<action>\}].$

Source indicates the state before the transition. *Destination* is the state describing the state after the state is executed. *<guard>* is the condition for the state transition. *<action>* is the action completed after the transition.

C. THE EXTENDED BEHAVIOR ANNEX

Space and time are extended based on the original Behavior Annex. In order to more concisely describe the processing sequence of unresponsive events in CPS, *trans_sequence* is proposed in the behavior annex of AADL execution model. *trans_sequence* combines with transitions actually.

In addition, three operators in the extension are introduced:

vara? varb: Assign the value of variable *varb* to variable *vara*. For example, *speed?v* assigns the value of variable *v* to variable *speed*.

vara! varb: Take the value of variable *vara* and put it in *varb*. For example, *v!speed* takes the value of variable *v* and assigns it to variable *speed*.

|: This symbol indicates that two assignment actions do not affect each other and perform simultaneously.

Definition 1: Behavior Annex that contains space and time:

$ExAction = (Vars, States, trans\_sequence, times, position)$

$times=\{delay, worst\_time, currdelay\_time, currexe\_time\}$

<delay> is the accept delay time of the current transition. *<worst_time>* is the worst execution time which is the total time of the behavior annex performing. *<delay>* and *<worst_time>* are received from the execution model. *<currdelay_time>* is the current delay time for the transition. *<currexe_time>* is the current execution time for the transition.

$<position>=[x][y][z],\ x{\sim}c|y{\sim}c|z{\sim}c,\ c$ is a number, $\sim \in \{\leq, \geq, <, >, =\}$

The three-dimensional coordinates follow the differential equation $\varphi(x)$, $\varphi(y)$, $\varphi(z)$.

"*action*" is the action involved in the current process, or "*act*" for short. We use "*act*" instead.

$trans\_sequence=Source[<pre\_act><guard>]$
$\rightarrow\{DesS1[\{<act1>\}], DesS2[\{<act2>\}, \ldots, error]\}$

$\{DesS1\ [\{<act1>\ \}], DesS2\ [\{<act2>\}, \ldots, error]\}$ is a transition sequence. When *DesS1* does not respond within space-time constraint, *DesS2* is executed. If *DesS2* does not response, *DesS3* is executed, and so on. Until the last error state is executed, the error processing is performed.

"pre_act"$::= act \mid initial\ position? \mid delay? \mid worst\_time?$

pre_act should get initial position, obtain *delay* and *worst_time* from the execution model etc.

$$\text{“}act_i\text{”}::=act \mid currdelay\_time? \mid position?$$

"act_i" executes transition. In addition, it records the current delay time before current state transition is performed. $act_i \in \{act1, act2, \ldots\}$

$guard::= con \wedge (currdelay\_time{\sim}delay),$
$\sim \in \{<, >, \leq, \geq, =\}$

"*guard*" is the condition of this transition. If the current transition is not performed within the delay time, it transforms into the next state of the transition sequence until the error state handles the error.

$error::= reset \mid call\ for\ worker$

"*error*" is an error state and performs operation to deal with the error.

D. THE HYBRID PHYSICAL ANNEX

The physical component of AADL is extended to model the physical behavior in CPS. The hybrid annex obtained by this extension can be used as annotations for device components, modeling continuous behavior of sensors and actuators, or implemented as abstract components.

Definition 2: The hybrid annex is defined as the following:

$PY = (sysvar, position, channels, pro)$

"*sysvar*" is a set of system variables.

"*position*" is the three-dimensional coordinates of the position. *Position*:: $=[x][y][z]$, $x\sim c|y\sim c|z\sim c$, c is a number, $\sim\in\{\leq, \geq, <, >, =\}$.

The three-dimensional coordinates follow the differential equation $\varphi(x)$, $\varphi(y)$, $\varphi(z)$.

"*channels*" is the set of data ports.

"*pro*" is a set of the physical device processes. It is the process in HS-TCSP which includes space-time property.

action:: = *interface? v*| *interface!* v

The direct information exchange of the physical device mainly lies in the exchange of variable values. The variable v of the device is input into the *interface* through *interface*?*v*. Other devices obtain the value of the variable v through *interface*!*v.*

E. EXAMPLE OF AUTOMATIC TEMPERATURE CONTROL SYSTEM

The distribution mechanism of the thread execution model writes data to the input data port; The behavior annex reads the data, makes a logical judgment, calculates it, and writes the result to the output data port when the calculation is complete. The behavior annex can receive events and data through the input event port and the input event data port, such as receiving the interrupt request, refining the dispatch condition of the non-periodic thread, etc. It sends events through the output event port, output event data port, such as sending interrupt request, subroutine call, mode change request, etc. Next, an example of automatic temperature control system shows below.

Automatic temperature control system is a typical CPS (as shown in Figure 2). The variable *temp* is the temperature value representing the real-time changes of the system, and the time t changes according to $t = t + 1$. When the temperature reaches *temp_max*, the heater is *off* and the system state is *OFF*. The temperature changes according to the differential equation of *temp* =-0.*1temp*. If the heater is not turned off by time *wait'*, *ALARM* is issued. When the temperature reaches *temp_min*, the heater will be turned on and the state is *ON*. The temperature will rise according to *temp* =5-0.1 *temp*. If the heater is not turned on within time *wait*, *ALARM* will be issued. The initial state is *OFF*.

The rounded rectangular part of the dotted line is the HAADL behavior annex model. The HAADL behavior annex reads and execution model data to make logical judgment and calculation. After the calculation is completed, the result is written to the output data port.

The HAADL behavior annex is described as:

thread implementation example.impl
annex behavior_specification{**
variables
temp:*Base_Types*::*float*;
times
t:*Base_Types*::*Integer*;
wait:*Base_Types*::*Integer*;
wait′:*Base_Types*::*Integer*;
position
temp_max:*Base_Types*::*float*;
temp_min:*Base_Types*::*float*;
states
OFF:*initial state*;
ON: *state*;
ALARM:*complete state*;
transitions
T_0:
$ON[\langle y = temp\_min\rangle\langle(t \leq wait)\rangle]\rightarrow\{OFF[\{\mathrm{t}= t{+}1 \wedge temp = \text{-}0.1temp\}], ALARM]\}$;
T_1:
$OFF[\langle y = temp\_max\rangle\langle(t \leq wait')\rangle]\rightarrow\{ON[\{\mathrm{t} = t{+}\ 1 \wedge temp = 5\text{-}0.1\ temp\}], ALARM]\}$;
**};
end *example.impl*;

IV. HS-TCSP(HYBRID SPACE-TIME COMMUNICATION SEQUENTIAL PROCESSES)

In this section, the HS-TCSP is proposed by extending the differential equation and space-time compositions vector. Besides, the extended space-time part of HS-TCSP is analyzed to verify the space-time compositions property of HS-TCSP according to the algorithm for space-time compositions satisfiability checking. The abnormal nodes are recorded in order to deal with them.

A. SYNTAX

Definition 3: For a system S, the process variable set

$PV = \{X, Y, P, Q \ldots\}$ represents the process in HS-TCSP, and the discrete variable set $DV = \{m, n, \ldots\}$ contains system variables. The discrete time variable set $TV = \{d, t \ldots\}$, and coordinate variables set $CV = \{x, y, z\}$(By default, x, y and z represent three-dimensional coordinate variables, and cannot be used for other variables).

The HS-TCSP can be defined as:

$P ::= STOP|SKIP|WAITt|a \rightarrow Q|P;Q|P\Box Q|P\sqcap Q| P \overset{d}{\triangleright} Q|PSQ|P\Delta Q|P[R]|P\backslash A|f(\mathrm{P})|P_A||_B Q|P|||Q|\mu X\cdot f(X)|$

$Con\gg P(Con::=VPf\&\&<cposition,ctime>|true)$

$|P\blacktriangleright Fin\_Con(Fin\_Con::=Fcon|true)\vdash Q$

STOP is a process which will never engage in external communication, and it makes the process terminate.

SKIP is a process which does nothing except terminate, and is ready to terminate immediately.

WAIT t is a delay for Skip. It does nothing, but is ready to terminate successfully after t time units.

$a \rightarrow Q$ is initially prepared to engage in synchronization a. If this event occurs, it immediately begins to behave as Q. The prefix operator $\rightarrow$ allows us to add communication events to a process description.

In the process P; Q, control is passed from process P to process Q if and when P performs the termination event. This event is not visible to the environment, and occurs as soon as P is ready to perform it. The sequential composition operator transfers control upon termination.

$P\Box Q$ is an external choice between process P and Q. If the environment is prepared to cooperate with P but not Q, then the choice is resolved in favor of P.

$P\sqcap Q$ is an internal choice between P and Q, and the outcome of this choice is nondeterministic.

$P\overset{d}{\triangleright}Q$ represents timeout. If no communication occurs between the two processes within d, it is considered timeout and control is passed from P to Q;

PSQ is time interruption. No matter whether P elapsed or not, the interruption is switched to Q;

$P\triangle Q$ represents interruption, any event execution of Q can cause interruption of P;

$P[R]$ has the same structure as process P, except that the events in P are mapped to another name through the relationship R. For example, the following process P continuously executes event a. Process Q is equal to all occurrences of a in process P. Replace with b, $P = a \to P$, $Q = P[a \to b]$;

$P\backslash A$ indicates that any events belonging to A in process P are not displayed;

The relabeled process $f(P)$ has a similar control structure to P, with observable events renamed according to function f.

In the hybrid parallel program $P_A||_BQ$, components P and Q must synchronize according to events from set $A \cap B$, and they interleave on all other events.

In an asynchronous parallel combination $P|||Q$, both subprograms evolve concurrently without interacting, though they must agree on termination. If both subprograms are well capable of performing the same event a, then a degree of nondeterminism may be introduced.

$\mu X \cdot f(X)$: X is a process variable, $A = \alpha X$, a recursively defined process must immediately unwind before it is able to perform any visible action.

$Con \gg P$ $(Con::=VPf\&\&<cposition,ctime>\ |true)$:

It is called a space-time conditional execution operator. When Con is satisfied, the execution process continues. $VPf::=Pf(v)|v{\sim}c|VPf1\wedge VPf2|true$, c is a number, $\sim\in\{\leq,\geq,<,>,=\}$. The condition variable v of Con changes continuously according to the differential equation Pf. The space-time vector is described as $<cposition,ctime>$. $cposition$ is the differential equation $\varphi(x)$, $\varphi(y)$, $\varphi(z)$ that the three-dimensional coordinates of the position variable follow. $cposition::=\varphi(x)\ |\varphi(y)\ |\varphi(z)\ |x{\sim}c\ |y{\sim}c\ |z{\sim}c\ |\ cposition1\ \wedge\ cposition2\ |true.$ $ctime$ is the condition that the time variable satisfies. The variable tv satisfies the predicate formula $\psi(tv)$. $ctime::=\psi(tv)\ |tv{\sim}c\ |ctime1\wedge\ ctime2\ |true$. When there is no need for a space or time constraint in the condition, the condition defaults to be true and always holds. For example $(k = 6n + 2\ \&\&<x = x{+}1,\ t = t{+}1\wedge\ t{<}delay>) \gg P$ means that k satisfies the differential equation $6n + 2$. Coordinate of the position x is changed according to $x + 1$. Time variable is changed according to $t + 1$. When the variable $t < delay$, process P is executed, otherwise P is not executed.

$P \blacktriangleright Fin\_Con\ (Fin\_Con::=Fcon|true) \vdash Q$:

It is called space-time conditional interruption operator. $Fcon$ is a predicate formula, which is the condition of the variables involved. The Fin_Con is $false$ and can be omitted by default. Use $\wedge$ if the conditions are met at the same time, and use $\vee$ if at least one of the conditions is met. If the condition Fin_Con is satisfied, the process P is interrupted and the process Q is executed. $P \blacktriangleright (x < 1) \vdash Q$ means that when the process P is executed, if the x coordinate is less than 1, the P process will be terminated and the Q process is executed.

The basic operation of TCSP $a?x$ means that the channel a receives the message x. Channels in TCSP represent a set of events, such as channel a: Int, which means that channel a can communicate with any event with integer data. Event a.2 is an element declared by channel a. $a!x$ means getting message from channel a and sending it to x.

B. HS-TCSP HYBRID SPACE-TIME TRANSITION SYSTEM

The semantics of Hybrid Space-Timed Transition System (HS-TTS) for HS-TCSP are discussed as follows. Reference [32] defines the semantics of the TCSP as a time transition system. The execution of an event in a process is performed during time transition. It can be seen as a transition in the system. The concept of hybrid space-time transition system is introduced for studying HS-TCSP.

Definition 4: The semantics of TCSP are a time transition system TTS_{TCSP}=<$NODES$, $\sum_T$, $\to$>, $NODES$ is a set of nodes, representing each process. $\sum_T$ is a set of events with a delay time, $\sum_T=\{(t_0,a_0), (t_1,a_1)\ldots(t_n,a_n)\}$, $\to$ is a transition relationship, $\to\subseteq NODES \times \sum_T \times NODES$, $N_1 \xrightarrow{(t,a)} N_2$, this represents the execution time of N_1, which is delayed by t time units and becomes the process represented by N_2.

Definition 5: A hybrid space-time transition system $TTHS - TCSP ::=< NODES, \sum_C, \to >$.

$NODES$ is a set of nodes, representing each process. $\sum_C$ is the set of events with delay time, conditional execution, and position assignment, $\sum_C = \{(c_0,a_0),(c_1,a_1)\ldots(c_n,a_n)\}$. $c = (p, t, fm)$. Condition c includes three parts: p, t and fm the predicate formula the variable satisfies. So, $\sum_T \subseteq \sum_C$, when $p_n = \varepsilon \wedge fm_n = true$, $\sum_T = \sum_C$. $\to$ is a transition relationship, $N_1 \xrightarrow{(c,a)} N_2$. It means that the process executed by N_1 performs the position assignment operation of p, executes event a, delays t time units, and becomes the process represented by N_2.

Definition 6: A HS-TCSP can be described as a hybrid space-time transition system $TTHS - TCSP =< NODES, \sum_C, \to >$

C. OPERATIONAL SEMANTICS OF HS-TCSP

The semantics of process calculus are generally given in three ways: operational semantics, denotational semantics and axiom semantics.

1) Operational semantics is that process calculus is regarded as execution on abstract machines, and transition

systems are generally used as abstract machines. By defining the state transition process of different operators, the semantics of process calculus are given;

2) Denotational semantics treats process expressions as functions, that is, the mapping relationship between input and output;

3) Axiom semantics prove the correctness of a program by proving program assertions.

Now we discuss the operational semantics of the hybrid space-time transition system for HS-TCSP.

Definition 7: Operational semantics of HS-TCSP

$Con \gg P\ (Con ::= VPf \&\& {<}cposition, ctime{>} | true)$

$$\frac{(VPf = false) \vee (cposition = false) \vee (ctime = false)}{s \xrightarrow{(c,a)} s} \tag{1}$$

$$\frac{(VPf = true) \wedge (cposition = true) \wedge (ctime = true)}{s \xrightarrow{(c,a)} s'} \tag{2}$$

If v changes according to the differential equation $pf(v)$, the variables of the position change according to $\varphi(x)$, $\varphi(y)$, $\varphi(z)$, the conditions of three-dimensional coordinates and time are *true*. The state s becomes s'(*formula* 2). Otherwise no change occurs(*formula* 2).

(2) $P \blacktriangleright Fin\_Con\ (Fin\_Con ::= Fcon \mid true) \vdash Q$

$$\frac{Fin\_Con = false}{s \xrightarrow{(c,a)} s} \tag{3}$$

$$\frac{Fin\_Con = true}{s \xrightarrow{(c,a)} s'} \tag{4}$$

If *Fin_Con* is *false*, continue to execute *P*(*formula* 3). If *Fin_Con* is *true*, *P* is terminated and the state in *Q* is executed(*formula* 4).

D. REFINEMENT

HS-TCSP is extended from TCSP, so HS-TCSP should include TCSP in syntax and semantics. All model-related function tests on HS-TCSP can be completed by the model test tool of TCSP. HS-TCSP only needs to check the space-time part which is extended.

The following proves that the semantic model of TCSP is a sub-semantic model of HS-TCSP. First, the sub-semantic model is defined, and then the theorem proof that HS-TCSP contains TCSP semantics is given. It is proved that HS-TCSP is an extension of TCSP.

Definition 8: A CSP P_A which belongs to Class A with semantic M_A is able to obtain another CSP P_B which belongs to Class B with semantic M_B. Then M_B semantics is sub-semantic model of M_A semantics. The refined model is as follows:

$$\frac{P_B\ sat\ S_B\ in\ M_B}{P_A\ sat\ S_A\ in\ M_A}(P_B \sqsubseteq P_A)$$

S_A is the semantics of P_A and S_B is the semantics of P_B.

Definition 9: All acceptable languages for HS-TCSP are a set of hybrid space-time transition sequences.

Theorem 1: The semantic model of TCSP is a sub-semantic model of HS-TCSP.

Proof: P_{TCSP} is a TCSP. By definition 8, TCSP is a time transition system $TTS_{TCSP} =< NODES, \sum_T, \rightarrow>$. The language it accepts is L, and HS-TCSP is constructed according to the TCSP. $P_{HS-TCSP}$ is $TTHS - TCSP =< NODES, \sum_C, \rightarrow>, c = (p, t, fm)$. The space-time language accepted by $P_{HS-TCSP}$ is L'. At this time, any time transition sequence in L is taken $R =< (\mathrm{t}_0, a_0), (\mathrm{t}_1, a_1) \ldots (\mathrm{t}_{n-1}, a_{n-1}) >$. In L', the only $R' =< (\mathrm{c}_0, a_0), (\mathrm{c}_1, a_1) \ldots (\mathrm{c}_n, a_n) >, <\mathrm{c}_0 = (p_0, t_0, fm_0), \mathrm{c}_1 = (p_1, t_1, fm_1) \ldots \mathrm{c}_n = (p_n, t_n, fm_n) >$ corresponds to it. $\sum_T \subseteq \sum_C$, if $p_n = \varepsilon \wedge c_n = true$, $\sum_T = \sum_C$. It is the refined relationship of $P_{HS-TCSP}$ to P_{TCSP}.

According to Theorem 1, all model-related function tests on HS-TCSP can be completed by the model test tool of TCSP.

E. ANALYSIS OF SPACE-TIME PROPERTY

HS-TCSP is extended from TCSP, so HS-TCSP should include TCSP in syntax and semantics. All model-related function tests on HS-TCSP can be completed by the model test tool of TCSP. HS-TCSP only needs to check the space-time part which is extended.

We can design a depth-first algorithm to determine whether the reachable graph G becomes a standard reachable graph. By traversing each edge in the reachable graph G and checking its position and time, then comparing the position and time of each edge with the security requirements of space and time constraint values, it can be judged whether the reachable graph G is standard reachable graph. The space-time compositions satisfiability checking algorithm can be used for the current CPS space and time satisfiability checking in CPS environment(as shown in Figure 3). If the current space or time exceeds the position or time constraints, an exception will occur in the system, which is shown as the exception node in the graph.

The input of the algorithm is the reachable graph G, the current space-time constraint array *TARRAY* and total time constraint values *worst_execute*. *worst_execute* is the worst execute time which can be received from execution model. The output results are whether G satisfies space-time compositions. And the set of *abnormal* records the abnormal nodes.

In the algorithm shown above, the current traversal path to graph G is saved in *cur_path*;

"*abnormal*" is used to store the abnormal nodes found in graph G. When the abnormal nodes do not exist in the graph G, the *abnormal* is empty, which means the graph G is a standard reachable graph. Otherwise, the reachable graph G is not a standard reachable graph.

Finally, the abnormal node is found and saved in the *abnormal*. The abnormal nodes can be handled according to the *abnormal* until the graph is space-time compositions satisfiability.

```
total_maxt:=worst_execute;
starray_cps:=TARRAY;//The requirements under current CPS environment.
abnormal:=Φ; cur_path:={$N_0$};
repeat
    totalt:= 0;//total time is 0
    ln:=last node in cur_path;
    if successor nodes of ln have been visited//delete the visited node
        then delete last node of cur_path;
    else
        begin bn:=take a unvisited successor node of ln;
            satisfied:=true;
            totalt:=totalt+ t[ln][bn];
            if totalt > total_maxt
                then satisfied :=false; //If the current totalt > total_maxt, this path does not meet the worst time and the path cannot be executed.
                    abnormal :=abnormal∪ {bn};
            else if totalt ≠ starray_cps[bn];
                then satisfied:=false;//If the current space or time does not satisfy the security requirements of CPS environment, it is false.The current state space contains the abnormal execution path
                    abnormal :=abnormal∪ {bn};
            cur_path:=cur_path∪ {bn};
        end
    until cur_path=Ø;
return satisfied;
```

FIGURE 3. The algorithm for space-time compositions satisfiability checking in CPS environment.

F. EXAMPLE OF INDUCTION LAMP

An induction lamp control system is used to apply HS-TCSP modeling to illustrate the description capabilities of HS-TCSP.

The induction lamp controller controls whether the induction lamp is lit by two variables, if the sound or the light is weak. The induction lamp is *ON* only when the light is dim and someone passes by. The vertical direction of the induction lamp is z coordinate. When a person walks within a range of 2 meters with the z axis as the center within the cylindrical range as shown in Figure 4, the upper and lower floors of the position will be triggered. (Assuming the height between each floor is 6 meters). It will turn off automatically after working longer than 2 minutes. If the induction lamp does not light up within 10 seconds, the standby induction lamp is enabled. The execution process of the standby lamp is the same as that of the induction lamp, except that if the induction lamp does not light up within 10 seconds, an error is reported.

Set the sound variable to sv and the light variable to lv. When $sv = 1$, there is sound, and when $lv = 1$, it is dark. Accordingly, $sv = 0$ is silent, and $lv = 0$ is bright. Only when $sv = 1 \wedge lv = 1$, the sensor light will be on. The remaining $sv = 1 \wedge lv = 0$, $sv = 0 \wedge lv = 1$, $sv = 0 \wedge lv = 0$, the sensor

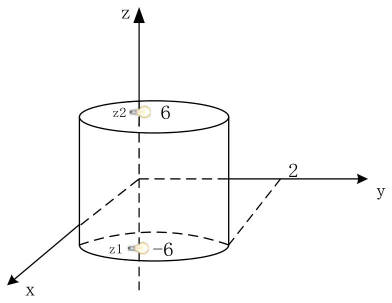


FIGURE 4. Example of induction lamp.

light will not work. That is, no operation will be performed. The unit of time is seconds, and the time changes according to $t = t + 1$. The coordinates of the upper and lower layers of lights are *z1* and *z2* of the z-axis. The induction lamp system includes a sound control system *SOCON*, a photo sensitive system *LICON*, and a human body induction system *PER*. The entire system process *SYSAGENT* is composed of *SENSE*, *STANDBY* and *ERROR* processes.

$PV=\{SOCON, LICON, PER, SENSE, STANDBY, ERROR, SYSAGENT\}$

$DV=\{sv, lv, x, y, z\}$

$CHANNEL = \{sound, light, cx, cy, cz\}$

$SOCON = sound?sv \rightarrow STOP$

$LICON = light?lv \rightarrow STOP$

$PER = cx?x \rightarrow STOP || cy?y \rightarrow STOP || cz?z \rightarrow STOP$

$\mathrm{ON} = turnon \rightarrow STOP$

$OFF = turnoff \rightarrow STOP$

$SENSE =$
$sound!sv \rightarrow STOP || light!lv \rightarrow STOP$
$|| cx!x \rightarrow STOP || cy!y \rightarrow STOP$
$|| cz!z \rightarrow STOP; (sv=1) \wedge (lv=1) \&\& (<Place[x, y, z],$
$(t: =0) \wedge (t=t+1) \wedge\ t \leq 120>)\ \gg ON \blacktriangleright (t\ >120) \vdash\ OFF;$
$SENSE$

$Place[x, y, z]=(x2+y2 \leq 4) \wedge (z - 6 > z_1) \wedge (z + 6 < z_2)$

$STANDBY =$
$sound!sv \rightarrow STOP || light!lv \rightarrow STOP || cx!x \rightarrow STOP ||$
$cy!y \rightarrow STOP || cz!z \rightarrow STOP;\ (sv=1) \wedge (lv=1) \&\&$
$(<Place[x, y, z], (t: = 0) \wedge (t=t+1) \wedge\ t \leq 120>) \gg ON \blacktriangleright$
$(t > 120) \vdash\ OFF;\ STANDBY$

$Place[x, y, z] = (x^2 + y^2 \leq 4) \wedge (z - 6 > z_1) \wedge (z + 6 < z_2)$

$ERROR = call \rightarrow STOP$

$SYSAGENT = SOCON || LICON || PER || SENSE \overset{d}{\triangleright} STANDBY \overset{d}{\triangleright}$
$ERROR\ (d = 10s)$

V. HAADL TO HS-TCSP TRANSFORMATION RULES

This section discusses the transformation rules from HAADL to HS-TCSP and the hierarchical structure based on ATL.

A formal method HS-TCSP is used in order to verify the space-time property satisfiability of HAADL. Next, we define the transformation rules from HAADL to HS-TCSP.

A. TRANSFORMATION RULES FROM HAADL TO HS-TCSP

(1) The variables in the behavior annex are mapped to the discrete variable set *DV* in HS-TCSP;

(2) The states in the behavior annex are mapped to the process variable set *PV* in HS-TCSP;

(3) The time variable set "*times*" in the behavior annex is mapped to the time variable set *TV* in HS-TCSP;

(4) The position variables in the behavior annex are mapped to the position variable set *CV* in HS-TCSP;

(5) The state transition in the behavior annex is expressed as

$trans\_sequence =$
$Source[<pre\_act><guard>] \rightarrow \{DesS1[\{<act1>\}],$
$DesS2[\{<act2>\} \ldots, error]\}$

guard has two forms: periodic execution on dispatch and conditional execution. Next, we map *guard*, *pre_act*, act_i, *error* etc. to HS-TCSP;

(5.1) *guard* is a condition for the transition, and it includes a time trigger, a condition trigger and an input / output event trigger occurring on a port. The timing trigger transforms into $P \overset{d}{\triangleright} Q$, P is a process that will not communicate with any process, then process Q is executed after time d; The condition trigger corresponds to the condition in the *Pcon* for $Con \gg P$ in HS-TCSP; The input and output that occur at the port events are transformed into input and output events in HS-TCSP.

(5.2) *Pre_act* and act_i can be transformed into the changes of the variable values in the differential equations in the space-time conditional execution operator of HS-TCSP. The operators *vara? varb* and *vara! varb* are transformed into the change of the value of variables. | is transformed into the process synchronization which only contains one event.

(5.3) *error* is transformed into the *ERROR* process in HS-TCSP

(5.4) The *trans_sequence* is a transition sequence that if it is not executed at the delay time, and the next state is executed. Transform it into:

$P \blacktriangleright (t > delay) \vdash Q$

P is the process of the previous state,and Q is the process of the next state of the transition sequence.

B. TRANSFORMATION FROM HAADL TO HS-TCSP BASED ON ATL

Atlas transformation language(ATL) is a language which is based on meta-model transformation. It is developed by OBEO and AtlanMod. The model transformation rules are a hybrid transformation rules language and are designed in both declarative and imperative ways. In model-driven engineering, there are several ways to transform a source model into a target model. ATL is based on the Eclipse Model Frame (EMF), its meta-model and the model are defined based on EMF. Essentially, it is a rule-based model transformation language. As a hybrid language, it has both descriptive language features and imperative language content, and can design rich transformation rules. As a rule-based language, description is its primary feature, but imperative content has been added to complete some complex transformations.

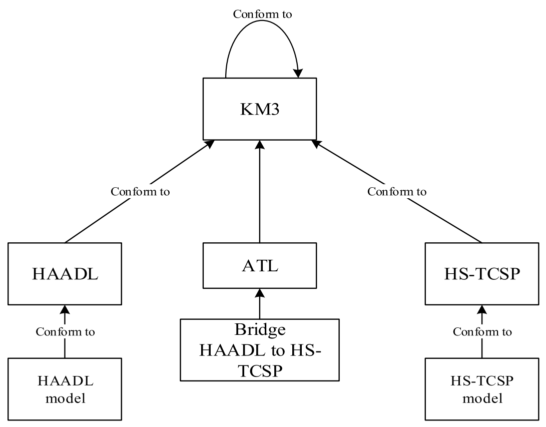


FIGURE 5. Hierarchical structure of transformation from HAADL to HS-TCSP.

The model-to-model transformation is based on an EMF architecture, and the hierarchy to model transformation is as follows (as shown in Figure 5):

HAADL model is the source model and HS-TCSP model is the target model. HAADL model transforms to its meta-model HAADL, while HS-TCSP model transforms to its meta-model HS-TCSP. They all conforms to the unique meta-meta-model KM3. Bridge HAADL to HS-TCSP is an instance of model transformation, which is also a model that transforms to the meta-model ATL of model transformation. It also transforms to the unique meta-meta-model KM3.

The only meta-meta-model in ATL is Ecore, which is equivalent to MOF. HAADL and HS-TCSP are the meta-model created by Ecore. HAADL and HS-TCSP are model instances transforming to these two meta-model. When ATL has been defined, Bridge HAADL to HS-TCSP is the model transformation model to be defined by the user. Thus, the model transformation program is described.

Therefore, a complete model transformation program needs four contents: HAADL, HS-TCSP, HAADL Model, and Bridge HAADL to HS-TCSP. The target model generated is HS-TCSP Model.

VI. CASE ANALYSIS

A. CASE OF AIRCRAFT COLLISION AVOIDANCE

In this section, an Aircraft Collision Avoidance System (ACAS) will be modeled using HAADL, and then the model is transformed from HAADL to HS-TCSP to analyze its space-time compositions satisfiability.

ACAS is a typical CPS. The research on the safety of ACAS has also been a research hotspot. Between 2000 and 2014, the AIR Force's F-16s lost 75 percent of their pilots to controllable crashes. The U.S. Air Force Research Laboratory (AFRL) discovered that the causes of crashes with ground or other aircraft include acceleration-induced loss of consciousness (G-LOC), space lost, over-targeting, distraction and task saturation. So it is necessary to keep the system safe. The description of AADL behavior annex is presented through an example of a simple ACAS. The transformation rules are used to transform the HAADL into a formal HS-TCSP to verify its space-time compositions, so as to verify the safety of ACAS.

An aircraft collision avoidance system in a vertical plane is considered. It assumes that the planes are in the same vertical plane position, as shown in the Figure 6. The coordinates of the two planes *p1* and *p2* are (*x1*, *z1*) and (*x2*, *z2*). The flight initial speeds are *v1* and *v2*. Assume that two planes are moving in a straight line at a uniform acceleration. The accelerations are *a1* and *a2*. Assuming that the aircraft collides at a certain point k after t time period, then $x1 + (v1 *t + 1/2*a1*t^2)= x2 + (v2 *t + 1/2*a2*t^2)$. When the aircraft is in collision avoidance, $a1 = 0$ and $a2 = 0$. If a danger of collision exists, *p1* aircraft and *p2* aircraft take certain collision avoidance measures.

The whole process is represented by UML sequence diagram as follows (as shown in Figure 7):

The whole process is divided into three modules:*Monitor*, *Management* and *Control*. *Monitor* module detects aircraft flight status and environment. *Management* module is responsible for collision avoidance management. *Control* module gives the corresponding collision avoidance instruction by calculating and estimating.

The specific scenarios of the two aircraft collision avoidance process are as follows:

(1) Two planes sail normally

(2) When there is a danger of collision, a response must be made within the latest response time *delay*. It is that if $distance = x2\text{-}x1 = (v1*delay + 1/2*a1*delay^2)+(v2*delay + 1/2*a2*delay^2)$, *Monitor* module detects danger distance and sends Danger of Collision(dc). Then *Management* module sends Collision Alarm(ca) information to *Control* module. And the *Control* module give the Rotate Instruction(ri). The planes center on the collision center, on the concentric circles with the radius of *r1* and *r2*, adjust the direction at an angle of ω to smoothly enter their collision avoidance orbits.

(3) The two planes rotate and fly on concentric circles at the same angular velocity ω, and the linear velocity (*linear velocity = angular velocity * radius*) in the process is the same as the speed of the original aircraft.

(4) After the rotation flight, *Monitor* module sends Collision Avoidance Completed(cac) to *Management* module and the Complete Information(ci) is sent to *Control* module. And then the *Control* module gives leave Instruction(li). Two aircrafts left the collision avoidance orbit at an angular velocity of -ω. When returning to the original course, the angular velocity ω is 0.

(5)After leaving the collision avoidance orbit, the *Monitor* module sends Safety Distance(sd) to *Management* module. Safety Information(si) and Normal Instruction(ni) are sent sequentially between *Management* module and *Control* module as shown in the sequence diagram.

Next, we describe the collision avoidance process with HAADL behavior annex:

thread implementation example.impl
annex behavior_specification{**
variables
v1: Base_Types:: float; *v2*: Base_Types:: float;
a1: Base_Types:: float; *a2*: Base_Types:: float;
r1: Base_Types:: float; *r2*: Base_Types:: float;
ω: Base_Types:: float; *distance*: Base_Types:: float;
dc:Base_Types:: String; ca:Base_Types:: String;
ri:Base_Types:: String; cac:Base_Types:: String;
ci:Base_Types:: String; li:Base_Types:: String;
sd:Base_Types:: String; si:Base_Types:: String;
ni:Base_Types:: String;
times
delay: Base_Types:: float;
position
x1: Base_Types:: float; *x2*: Base_Types:: float;
z1: Base_Types:: float; *z2*: Base_Types:: float;
states
s1: initial complete state;
s2: state;
s3: state;
s4: complete state;
error: state;
transitions
T_0:
s1[<*p1v*?*v1*|*p2v*?*v2*|*position*?|*delay*?|*worst_time*?|
currdelay_time?|DC!|CA!|RI?><($distance = x2\text{-}x1 = (v1*delay + 1/2*a1*delay^2)+(v2*delay + 1/2*a2*delay^2)) \wedge (<currdelay\_time<delay)>] \rightarrow$
{s2[{<*position*?|*currexe_time*?>}], *error*]}
T_1:
s2[< *delay*?|*currdelay_time*? >< $(r_1\omega)^2 = v_1^2 \wedge (r_2\omega)^2 = v_2^2$ >] $\rightarrow$
{*s3*[{< $p1w = \omega | p2w = \omega | pa1 = a1 | pa2 = a2$|*position*?|
currexe_time?|CAC!CI!>}], *error*]}
T_2:
s3 [<*delay*?|*currdelay_time*?|*p1w*?|*p2w*?|LI?>
< $(r_1^*(-\omega))^2 = v_1^2 \wedge (r_2^*(-\omega))^2 = v_2^2) \wedge$ curr*delay* < *delay* $\wedge\, p1w \neq 0 \wedge p2w \neq 0$ >]$\rightarrow$
{*s4*[{<*p1w*=0|*p2w*=0|*pa1*=*a1*,
pa2=*a2*|*position*?|*currexe_time*?|SD!|SI!)>}], *error*]}
T_3:
s4 [<*delay*?|*currdelay_time*?|NI?><
p1w=0|*p2w*=0|*p1v*=*v1*|*p2v*=*v2*>]$\rightarrow$
{*s1*[{<*p1v*?*v1*|*p2v*?*v2*)>}], *error*]}
**};
end example.impl;

Next, we transform the HAADL behavior annex to HS-TCSP through a defined transformation rule:

The states *s1*, *s2*, *s3*, and *s4* correspond to the processes *NORMAL*, *ENTRY*, *ROTATION* and *EXIT*, and the variables in the HAADL behavior annex corresponds to the set of

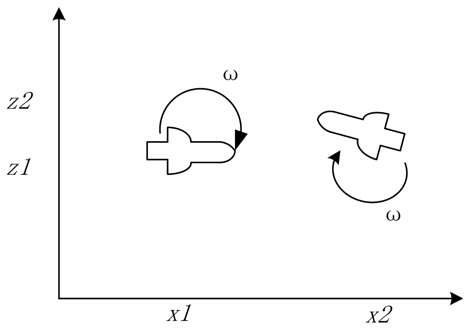


FIGURE 6. Collision avoidance.

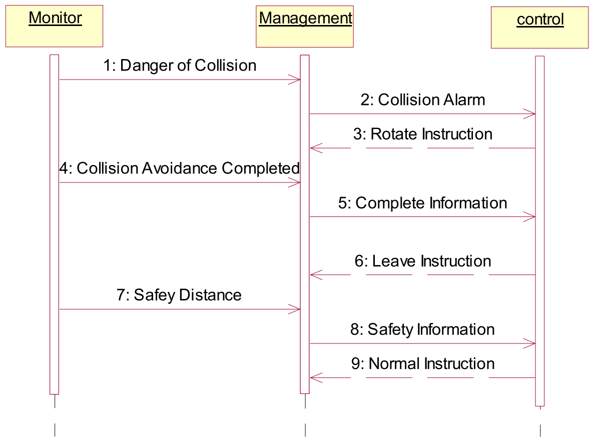


FIGURE 7. UML sequence diagram for ACAS.

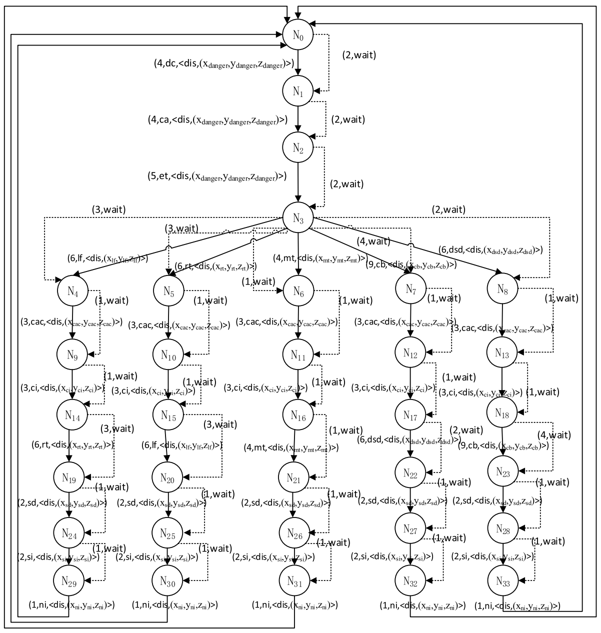


FIGURE 8. The space-time transition system *G* of aircraft collision avoidance.

discrete variables *DV*, *TV* and *CV* in HS-TCSP. The *guard* in execution and *position* involved correspond to *conditions*. The *conditions* correspond to *Con* and *Fin_Con* in the operator. The resulting HS-TCSP description is as follows:

NORMAL $=$
p1v?v1→STOP||p2v?v2→STOP||position?v $\rightarrow$
STOP||delay?→STOP||worst_time?→STOP||
currdelay_time?→STOP||
DC!→STOP||CA!→STOP||RI?→STOP||(<Place
*[x2-x1= (v1*delay +1/2*a1*delay*^{2}*)+(v2*delay*
*+1/2*a2*delay*^{2}*)],*
(*currdelay_time* $<$ *delay*) $>$) $\gg$ ***ENTRY*** $\blacktriangleright$
(*currdelay_time* $>$ *delay*)$\vdash$ ***ERROR***
ENTRY $=$
delay?→STOP|| currdelay_time?→STOP
$||$*currexe_time?* $\rightarrow$ *STOP||position?* $\rightarrow$ *STOP||*$(<$
$(r_1\omega)^2 = v_1^2 \wedge (r_2\omega)^2 = \mathrm{v}_2^2$, *currdelay_time* $<$
delay $>)\gg$ ***ROTATION*** $\blacktriangleright$
$((r_1\omega)^2 \neq v_1^2 \vee (r_2\omega)^2 \neq v_2^2 \vee$ *currdelay_time* $>$
delay)$\vdash$
ERROR
ROTATION $=$
delay? $\rightarrow$ *STOP||currdelay_time?* $\rightarrow$ *STOP||LI?* $\rightarrow$
STOP $||p1w?\omega \rightarrow$ *STOP*$||p2w?\omega \rightarrow$ *STOP*$||(<$
$(r_1 * (-\omega))^2 = v_1^2 \wedge (r_2 * (-\omega))^2 \neq v_2^2 \wedge \omega \neq$
0, *currdelay_time* $<$ *delay* $>)\gg$ ***EXIT*** $\blacktriangleright$
$(r_1 * (-\omega))^2 \neq v_1^2 \vee (r_2 * (-\omega))^2 \neq v_2^2 \vee$
currdelay_time $>$ *delay* $\vee\ \omega = 0)\vdash$ ***ERROR***
EXIT=*delay?→STOP||currdelay_time?→STOP*
$||NI? \rightarrow$ *STOP*$||(p1\omega = 0 \wedge p2\omega = 0 \wedge p1v = v1 \wedge$
$p2v = v2)\gg$ ***NORMAL*** $\blacktriangleright$
$(p1\omega \neq 0 \vee p2\omega \neq 0 \vee p1v \neq v1 \vee p2v \neq v2)\vdash$
ERROR
ERROR=*call→STOP*

B. ANALYSIS OF THE CASE OF AIRCRAFT COLLISION AVOIDANCE

After the transformation from HAADL to HS-TCSP, all model-related function tests on HS-TCSP can be completed by the model test tool of TCSP. The space-time properties of the system are analyzed below.

The state space is a hybrid space-time transition system. For different instructions, time for execution is different. When the risk of collision exists, the aircraft takes several collision avoidance actions: turn left, turn right, climb, descend and maintain. The locations and time points designed from the above aircraft collision avoidance scene are shown in the following table.

The state space diagram G shows as Figure 8. Five types of collision avoidance actions may be taken to avoid collisions.

Assuming that the current *worst_execute* is 52. And the CPS environment requires the security of the input array is {0, 6, 12, 19, 28, 28, 24, 32, 27, 32, 32, 28, 36, 31, 36, **32**, 32, 40, 35, 45, 45, 37, 48, 48, 48, 48, 40, 51, 51, 51, 51, 43, **52**, **52**}. And the array gets data from the past experience database. According to the algorithm for space-time compositions satisfiability checking in CPS environment, the return value is *false* and *abnormal* $= \{N_{15}, N_{32}, N_{33}\}$. *totalt* is 54 at the node

TABLE 2. The time for aircraft collision avoidance.

event	delay_ time	execute _time	position
dc	2	4	(x_{dc},y_{dc},z_{dc})
ca	2	4	(x_{ca},y_{ca},z_{ca})
enter(et)	2	5	(x_{et},y_{et},z_{et})
left(lf)	3	6	(x_{lf},y_{lf},z_{lf})
right(rt)	3	6	(x_{rt},y_{rt},z_{rt})
climb(cb)	4	9	(x_{cb},y_{cb},z_{cb})
descend(dsd)	2	6	$(x_{dsd},y_{dsd},z_{dsd})$
maintain(mt)	1	4	(x_{mt},y_{mt},z_{mt})
cac	1	3	$(x_{cac},y_{cac},z_{cac})$
ci	1	3	(x_{ci},y_{ci},z_{ci})
leave(the direction is opposite to the above)	3	8	$(x_{leave},y_{leave},z_{leave})$
sd	1	2	(x_{sd},y_{sd},z_{sd})
si	1	2	(x_{si},y_{si},z_{si})
ni	0	1	(x_{ni},y_{ni},z_{ni})

N_{32} and N_{33}. "*totalt* >52" means the worst execute time is not satisfied. Climbing and descending collision avoidance measures cannot safely avoid collisions in this scenario. Besides, at position (x_{ci}, y_{ci}, z_{ci}), the system requires that the time at N_{15} is 32. In the state space diagram $G.\ totalt$ of N_{15} is 36. It should arrive at (x_{ci}, y_{ci}, z_{ci}) at time 32. The time for the system to reach this point will be 36, so there is a space-time combination safety problem. The current CPS space-time compositions does not meet the security requirements of the system. The fault-tolerant measures can be taken at this time. For example, the next step is to speed up to meet the security requirements of the system.

VII. CONCLUSION

The application of CPS is extremely broad, and its modeling must take the environment into account. Besides, heterogeneous modeling of computing models and physical models also brings many difficulties to the modeling of CPS. We have solved the problem of space-time consistency before. Based on the assumption of space-time consistency, this paper verifies whether space-time compositions meet the security requirements of the current CPS environment so as to ensure the security of CPS. In this paper, the space and time factors are extended in the AADL behavior annex. In order to verify its correctness, the differential equations and space-time compositions vector are extended on the basis of TCSP, and the transformation rules from HAADL to HS-TCSP are defined. Finally, the corresponding transformation rules are defined to convert HAADL into HS-TCSP.

This article mainly expands the behavior annex of AADL. There is not much introduction to the expansion of physical annex, but physical annex is also involved in the CPS. Therefore, related research on physical annex will be a focus of the next step. In addition, the CPS environment is complex. CPS has a complex operating environment and a wide range of application scenarios in the current development environment. Changes in space and time may cause information leakage problems. Therefore, how to ensure the information security in the process of space and time changes is also the focus of the next research.

REFERENCES

[1] X. Zheng, C. Julien, M. Kim, and S. Khurshid, "Perceptions on the state of the art in verification and validation in cyber-physical systems," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2614–2627, Dec. 2017, doi: 10.1109/JSYST.2015.2496293.

[2] *A Particularly Serious Railway Traffic Accident on the Ningbo-Wenzhou Line*. Accessed: Dec. 7, 2015. [Online]. Available: https://baike.so.com/doc/5381626-5617962.html

[3] P. H. Feiler, B. Lewis, S. Vestal, and E. Colbert, "An overview of the SAE Architecture Analysis & Design Language (AADL) standard: A basis for model-based architecture-driven embedded systems engineering," in *Architecture Description Languages*. Toulouse, France: Springer, 2005, pp. 3–15.

[4] L. Zhang, *Specifying and Modeling Cloud Cyber Physical Systems Based on AADL*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, 2018, pp. 26–29.

[5] F. S. Goncalves, D. Pereira, E. Tovar, and L. B. Becker, "Formal verification of AADL models using UPPAAL," in *Proc. IEEE Comput. Soc.*, Nov. 2017, pp. 117–124.

[6] J. Liu, T. Li, Z. Ding, Y. Qian, H. Sun, and J. He, "AADL+: A simulation-based methodology for cyber-physical systems," *Frontiers Comput. Sci.*, vol. 13, no. 3, pp. 516–538, Jun. 2019, doi: 10.1007/s11704-018-7039-7.

[7] R. He, W. Zhang, L. Wu, and Y. Jian, "AADL-based reliability modeling method of cyber-physical systems," in *Proc. World Symp. Softw. Eng.*, 2019, pp. 47–58.

[8] Y. Li, J. Li, and Z. Cao, "Research on a formal combined modeling method," *Comput. Technol. Develop.*, vol. 27, no. 11, pp. 106–109, 2017.

[9] M. Li, Y. Zhuang, Y. Hu, "A method for modeling and evaluating reliability of embedded software combining AADL and Z," *Comput. Sci.*, vol. 46, no. 08, pp. 217–223, 2019.

[10] Z. Yang, K. Hu, Y. Zhao, and D. Ma, "AADL model verification based on time abstract state machine," *J. Softw.*, vol. 26, no. 2, pp. 202–222, 2015.

[11] F. Zhang and X. Cao, "Research on MA hybrid probabilistic hybrid automaton transformation method," *Comput. Technol. Develop.*, vol. 29, no. 2, pp. 19–22, 2019.

[12] C. A. R. Hoare, "Communicating sequential processes," *Commun. ACM*, vol. 21, no. 8, pp. 666–677, 1978.

[13] G. M. Reed and A. W. Roscoe, "A timed model for communicating sequential processes," *Theor. Comput. Sci.*, vol. 58, pp. 249–261, 1988.

[14] C. Luo, R. Wang, Y. Guan, X. Li, Z. Shi, and X. Song, "CPS integrated modeling method for real-time data," *J. Softw.*, vol. 30, no. 7, pp. 1966–1979, 2019.

[15] J. Zhang, Y. Zhu, and F. Xiao, "Modelling and analysis of real-time and reliability for WSN-based CPS," *Int. J. Internet Protocol Technol.*, vol. 12, no. 2, p. 76, 2019.

[16] Q. Su, T. Wang, T. Chen, and R. Chen, "CPS security modeling and validation based on time automaton," *Inf. Secur. Res.*, vol. 3, no. 7, pp. 601–609, 2017.

[17] X. Cao, Z. Cao, and X. Bu, "CPS-oriented hybrid AADL modeling and model transformation," *Comput. Technol. Develop.*, vol. 29, no. 10, pp. 35–40, 2019.

[18] I. Graja, S. Kallel, N. Guermouche, and A. H. Kacem, *Modeling and Verification of Temporal Properties in Cyber-Physical Systems*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, 2017, pp. 325–330.

[19] C. Li-Na, H. Hong-bin, and D. Su, "Research on CPS spatio-temporal event model based on the state," in *Proc. 6th Int. Conf. Comput. Sci. Educ. (ICCSE)*, Aug. 2011, pp. 195–198.

[20] J. Zhang, L. Wang, and H. Fan, "Modeling and analysis of temporal and spatial consistency of physical entities in CPS," *Comput. Eng. Appl.*, vol. 54, no. 14, pp. 41–44, 2018.

[21] H. Shen, S. Huo, J. Cao, and T. Huang, "Generalized state estimation for Markovian coupled networks under round-robin protocol and redundant channels," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1292–1301, Apr. 2019, doi: 10.1109/TCYB.2018.2799929.

[22] Y. Zhang, Y. Chen, and H. Wu, "A process algebra with qualitative calculus for modeling spatio-temporal behaviour," *IEEE Access*, vol. 7, pp. 57172–57187, 2019, doi: 10.1109/ACCESS.2019.2914228.

[23] E. Ahmad, B. R. Larson, S. C. Barrett, N. Zhan, and Y. Dong, "Hybrid annex: An AADL extension for continuous behavior and cyber-physical interaction modeling," *ACM SIGAda Ada Lett.*, vol. 34, no. 3, pp. 29–38, Nov. 2014.

[24] C. Yang, Y. Dong, F. Zhang, E. Ahmad, and B. Gu, "Formal semantics of AADL models with machine-readable CSP," in *Proc. Int. Conf. Comput. Soc.*, May 2012, pp. 565–571.

[25] Z. Yu, D. Yunwei, Z. Fan, and Z. Yunfeng, *Research on Modeling and Analysis of CPS*. Banff, AB, Canada: Springer-Verlag, 2011, pp. 92–105.

[26] E. Ahmad, Y. Dong, S. Wang, N. Zhan, and L. Zou, *Adding Formal Meanings to AADL With Hybrid Annex*. Niteról, Brazil: Springer-Verlag, 2015, pp. 228–247.

[27] E. Ahmad, Y. Dong, B. Larson, J. D. Lü, T. Tang, and N. Zhan, "Behavior modeling and verification of movement authority scenario of Chinese train control system using AADL," *Sci. China Inf. Sci.*, vol. 58, no. 11, pp. 1–20, Nov. 2015, doi: 10.1007/s11432-015-5346-2.

[28] J. Tang, M. A. Piera, and T. Guasch, "Coloured Petri net-based traffic collision avoidance system encounter model for the analysis of potential induced collisions," *Transp. Res. C, Emerg. Technol.*, vol. 67, pp. 357–377, Jun. 2016, doi: 10.1016/j.trc.2016.03.001.

[29] J. Tang, F. Zhu, and M. A. Piera, "A causal encounter model of traffic collision avoidance system operations for safety assessment and advisory optimization in high-density airspace," *Transp. Res. C, Emerg. Technol.*, vol. 96, pp. 347–365, Nov. 2018, doi: 10.1016/j.trc.2018.10.006.

[30] L. Zhang, "An integration approach to specify and model automotive cyber physical systems," in *Proc. Int. Conf. Connected Vehicles Expo*, Dec. 2013, pp. 568–573.

[31] P. Dissaux, J. P. Bodeveix, and M. Filali. (2006). *AADL Behavioral Annex*. [Online]. Available: http://ftp.estec.esa.nl/pub/wm/anonymous/wme/Web/AADLBehaviour2006.pdf

[32] S. Schneider, "An operational semantics for timed CPS," *Inf. Comput.*, vol. 116, no. 2, pp. 193–213, 1995.

XIAOYING CHEN was born in 1997. She is currently pursuing the master's degree with Jiangsu Normal University. Her research interests include software engineering, formal methods, cyber physical systems, and spatio-clock constraint specification. She is a Student Member of China Computer Federation.

YI ZHU (Member, IEEE) was born in 1976. He received the Ph.D. degree. He holds a postdoctoral position and is also a Professor. His current research interests include software engineering, formal methods, cyber physical systems, and intelligent software development. He is a member of China Computer Federation.

YU ZHAO was born in 1997. He is currently pursuing the master's degree with Jiangsu Normal University. His research interests include software engineering, formal methods, software defect prediction, and machine learning. He is a Student Member of China Computer Federation.

JINYONG WANG was born in 1983. He is currently pursuing the Ph.D. degree with the Nanjing University of Aeronautics and Astronautics. His research interests include spatio-clock constraint specification, collaboratively autonomous driving safety analysis, and formal modeling checking.

ANARBEKOV ALTYNBEK was born in 1997. He is currently pursuing the master's degree with Jiangsu Normal University. His research interests include computer vision, artificial neural networks, software engineering, and cyber physical space.