HYBRID ZIA AND ITS APPROXIMATED REFINEMENT RELATION*

Zining Cao[1,2,3] and Hui Wang[1]
[1]*National Key Laboratory of Science and Technology on Avionics System Integration, Shanghai 200233, China*
[2]*Department of Computer Science and Technology, Nanjing University of Aero. & Astro., Nanjing 210016, China*
[3]*Provincial Key Laboratory for Computer Information Processing Technology, Soochow University, Suzhou 215006, China*

Keywords: Interface automata, Z notation, Hybrid automata, Approximated refinement relation.

Abstract: In this paper, we propose a specification model combining interface automata, hybrid automata and Z language, named HZIA. This model can be used to describe temporal properties, hybrid properties, and data properties of hybrid software/hardware components. We also study the approximated refinement relation on HZIAs.

1 INTRODUCTION

Modern software systems are comprised of numerous components, and are made larger through the use of software frameworks. Hybrid software/hardware systems exhibit various behavioural aspects such as discrete and continuous transition, communication between components, and state transformation inside components. Formal specification techniques for such systems have to be able to describe all these aspects. Unfortunately, a single specification technique that is well suited for all these aspects is yet not available. Instead one needs various specialised techniques that are very good at describing individual aspects of system behaviour. This observation has led to research into the combination and semantic integration of specification techniques. In this paper we combine three well researched specification techniques: Interface automata, hybrid automata and Z.

Interface automaton is a light-weight automata-based languages for component specification, which was proposed in (Luca de Alfaro, 2001). An interface automaton (IA), introduced by de Alfaro and Henzinger, is an automata-based model suitable for specifying component-based systems. IA is part of a class of models called interface models, which are intended to specify concisely how systems can be used and to adhere to certain well-formedness criteria that make them appropriate for modelling component-based systems.

Hybrid automaton (Henzinger, 1996) is a formal model for a mixed discrete-continuous system. A paradigmatic example of a mixed discrete-continuous system is a digital controller of an analog plant. The discrete state of the controller is modelled by the vertices of a graph (control modes), and the discrete dynamics of the controller is modelled by the edges of the graph (control switches). The continuous state of the plant is modelled by points in $\mathbf{R}^n$, and the continuous dynamics of the plant is modelled by flow conditions such as differential equations. The behavior of the plant depends on the state of the controller: each control mode determines a flow condition, and each control switch may cause a discrete change in the state of the plant, as determined by a jump condition. Dually, the behavior of the controller depends on the state of the plant: each control mode continuously observes an invariant condition of the plant state, and by violating the invariant condition, a continuous change in the plant state will cause a control switch.

Z (Bowen, 2003; Spivey, 1998; Woodcock and Davies, 1996) is a typed formal specification notation based on first order predicate logic and set theory. The formal basis for Z is first order predicate logic extended with type set theory. Using mathemat-

*This work was supported by the Aviation Science Fund of China under Grant No. 20085552023, the National Natural Science Foundation of China under Grants No. 60873025, the Natural Science Foundation of Jiangsu Province of China under Grant No. BK2008389, and the Foundation of Provincial Key Laboratory for Computer Information Processing Technology of Soochow University under Grant No. KJS0920.

Cao Z. and Wang H..
HYBRID ZIA AND ITS APPROXIMATED REFINEMENT RELATION.
DOI: 10.5220/0003504002600265
In *Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2011), pages 260-265
ISBN: 978-989-8425-57-7

ics for specification is all very well for small examples, but for more realistically sized problems, things start to get out of hand. To deal with this, Z includes the schema notation to aid the structuring and modularization of specifications. A boxed notation called schemas is used for structuring Z specifications. This has been found to be necessary to handle the information in a specification of any size. In particular, Z schemas and the schema calculus enable a structured way of presenting large state spaces and their transformation. But Z itself is cumbersome for specifying parallel systems. Its use will produce a much longer specification than if some other specification languages are used. Hence it is more convenient to use a language like CSP (Hoare, 1985) in such cases. Work has been undertaken to attempt to combine some of the features of CSP with Z (Fischer, 1996; Fischer, 1997; Fischer, 1998).

In this paper, we present a new specification language which combines interface automata, hybrid automata and Z language. Interface automata are a kind of intuitive models for interface property of software components. Hybrid automata are a model of mixed discrete-continuous systems. Z can describe the data property of states and transitions of a system. To specify mixed discrete-continuous software/hardware components, we give the definition of HZIA. Roughly speaking, a HZIA is in a style of hybrid interface automata but its states and operations are described by Z language. Furthermore, we define the approximated refinement relation between HZIAs and prove some propositions of such refinement relation. This paper is organized as follows: Section 2 gives a brief review of interface automata, hybrid automata and Z language. In Section 3, we propose a specification language-HZIA. Furthermore, the approximated refinement relation for HZIA are presented and studied. The paper is concluded in Section 4.

2 OVERVIEW OF INTERFACE AUTOMATA, HYBRID AUTOMATA AND Z LANGUAGE

In this section, we give a brief overview of interface automata, hybrid automata and Z language.

2.1 Interface Automata

An interface automaton (IA) (Luca de Alfaro, 2001), introduced by de Alfaro and Henzinger, is an automata-based model suitable for specifying component-based systems. IA is part of a class of models called interface models, which are intended to specify concisely how systems can be used and to adhere to certain well-formedness criteria that make them appropriate for modelling component-based systems. The two main characteristics of interface models are that they assume a helpful environment and support top-down design.

Definition 1. An interface automaton (IA) $P = \langle V_P, V_P^i, A_P^I, A_P^O, A_P^H, T_P \rangle$ consists of the following elements:

(1) V_P is a set of states,

(2) $V_P^i \subseteq V_P$ is a set of initial states. If $V_P^i = \emptyset$ then P is called empty.

(3) A_P^I, A_P^O and A_P^H are disjoint sets of input, output, and internal actions, respectively. We denote by $A_P = A_P^I \cup A_P^O \cup A_P^H$ the set of all actions.

(4) T_P is the set of transitions between states such that $T_P \subseteq V_P \times A_P \times V_P$.

The interface automaton P is closed if it has only internal actions, that is, $A_P^I = A_P^O = \emptyset$; otherwise we say that P is open.

The composition of two IAs consists of all possible interleaved transitions of the two IAs, except for those actions that are shared. Two IAs are composable if they do not take any of the same inputs, do not produce any of the same outputs and the internal actions of the two components do not overlap. An internal action is created through the composition of IA when an output action of one component is internally consumed by an input action of another component. This synchronization reduces the two actions to an internal action on a single transition.

IA Q refines IA P if Q provides the services of P; it can have more inputs but no more output actions. As such, a refinement of an IA does not constrain the environment more than the original IA does.

2.2 Hybrid Automata

A hybrid system is a dynamical system with both discrete and continuous components. Hybrid automata (Henzinger, 1996) are a model of hybrid systems.

Definition 2. A hybrid automaton H consists of the following components.

(1) Variables. A finite set $X = \{x_1, ..., x_n\}$ of real-numbered variables. The number n is called the dimension of H. We write X' for the set $\{x_1', ..., x_n'\}$ of primed variables (which represent values at the conclusion of change).

(2) Q is a finite set of states.

(3) $q_0 \in Q$ is the initial state.

(4) $\phi_{init} \in \Phi(X)$ is the initial condition.

(5) $T \subseteq Q \times \Phi(x_1,...,x_{|X|},x'_1,...,x'_{|X|}) \times Q$ is a finite set of transitions. Variables $x'_1,...,x'_{|X|}$ represent the new values taken by the variables $x_1,...,x_{|X|}$ after the firing of the transition.

(6) $Act : Q \rightarrow \Phi(x_1,...,x_{|X|},t,x'_1,...,x'_{|X|})$ is the activity function assigning to each state q a formula $Act(q)$. The variable t represents time elapsing.

For the sake of space, more details of Hybrid automata can be refereed to the article (Henzinger, 1996).

2.3 Z Language

Z was introduced in the early 80's in Oxford by Abrial as a set-theoretic and predicate language for the specification of data structure, state spaces and state transformations. The first systematic description of Z is (Spivey, 1998). Since then the language has been used in many case studies and industrial projects (e.g. (Bowen, 2003; Woodcock and Davies, 1996)).

Z includes the schema notation to aid the structuring and modularization of specifications. A boxed notation called schemas is used for structuring Z specifications.

Schemas are primarily used to specify state spaces and operations for the mathematical modelling of systems. For example, here is a schema called StateSpace:

StateSpace

$x_1 : S_1;\ ...;\ x_n : S_n$

$Inv(x_1,...,x_n)$

This schema specifies a state space in which x_1, . . . , x_n are the state variables and S_1, . . . , S_n are expressions from which their types may be systematically derived. Z types are sets - x_1, . . . , x_n should not occur free in S_1, . . . , S_n, or if they do, they refer instead to other occurrences of these variables already in scope (e.g., globally defined variables). $Inv(x_1,...,x_n)$ is the state invariant, relating the variables in some way for all possible allowed states of the system during its lifetime.

Z makes use of identifier decorations to encode intended interpretations. A state variable with no decoration represents the current (before) state and a state variable ending with a prime ($'$) represents the next (after) state. A variable ending with a question mark (?) represents an input and a variable ending with an exclamation mark (!) represents an output.

The vertical form of schema

S

$D_1;\ ...;\ D_m$

$P_1;\ ...;\ P_n$

is equivalent to the horizontal form of schema $S \widehat{=} [D_1;\ ...;\ D_m \mid P_1;\ ...;\ P_n]$

In the following, we sometimes use the horizontal form.

In Z (Bowen, 2003; Spivey, 1998; Woodcock and Davies, 1996), there are many schema operators. For example, we write $S \wedge T$ to denote the conjunction of these two schemas: a new schema formed by merging the declaration parts of S and T and conjoining their predicate parts. $S \Rightarrow T$ ($S \Leftrightarrow T$) is similar to $S \wedge T$ except connecting their predicate parts by $\Rightarrow$ ($\Leftrightarrow$). The hiding operation $S \backslash (x_1,...,x_n)$ removes from the schema S the components $x_1,...,x_n$ explicitly listed, which must exist. Formally, $S \backslash (x_1,...,x_n)$ is equivalent to $(\exists x_1 : t_1;\ ...;\ x_n : t_n \bullet S)$, where $x_1,...,x_n$ have types $t_1,...,t_n$ in S. The notation $\exists x : a \bullet S$ states that there is some object x in a for which S is true. The notation $\forall x : a \bullet S$ states that for each object x in a, S is true.

For the sake of space, more details of Z can be refereed to some books on Z (Bowen, 2003; Spivey, 1998; Woodcock and Davies, 1996).

3 HYBRID INTERFACE AUTOMATA WITH Z NOTATION

In many cases, systems have both discrete and continuous property. To specify hybrid systems, we now propose hybrid interface automata with Z notation, named HZIA.

3.1 Model

In (Cao, 2010), we propose a model which combines interface automata and Z notation, named ZIA. In this paper, the ZIA (Cao, 2010) is extended to hybrid version of ZIA, which can be used to specify hybrid behavioural and the data structure aspects of a system.

Definition 3. A hybrid interface automaton with Z notation (HZIA) $P = \langle S_P,\ S^i_P,\ A^I_P,\ A^O_P,\ A^H_P,\ X_P, V^I_P,\ V^O_P,\ V^H_P,\ C_P,\ F^S_P,\ F^A_P,\ T_P \rangle$ consists of the following elements:

(1) S_P is a set of states;

(2) $S^i_P \subseteq S_P$ is a set of initial states. If $S^i_P = \emptyset$ then P is called empty;

(3) A_P^I, A_P^O and A_P^H are disjoint sets of input, output, and internal actions, respectively. We denote by $A_P = A_P^I \cup A_P^O \cup A_P^H$ the set of all actions;

(4) $X_P = \{x_1,...,x_n\}$ is a finite set of real-numbered variables; The number n is called the dimension of P. We write X' for the set $\{x_1',...,x_n'\}$ of primed variables (which represent values at the conclusion of change).

(5) V_P^I, V_P^O and V_P^H are disjoint sets of input, output, and internal variables, respectively. We denote by $V_P = V_P^I \cup V_P^O \cup V_P^H$ the set of all variables. We have that $X_P \subseteq V_P$ which are all continuous valued variables and $V_P - X_P$ are all discrete valued variables;

(6) C_P is a variable representing time, whose value is a real number, $C_P \notin V_P$;

(7) F_P^S is a map, which maps any state in S_P to a state schema $\Phi(V_P \cup \{C_P\})$ in Z language;

(8) F_P^A is a map, which maps any input action in A_P^I to an input operation schema $\Phi(V_P)$ in Z language, and maps any output action in A_P^O to an output operation schema $\Phi(V_P)$ in Z language, and maps any internal action in A_P^H to an internal operation schema $\Phi(V_P)$ in Z language;

(9) T_P is the set of transitions between states such that $T_P \subseteq S_P \times A_P \times S_P$. If $(s,a,t) \in T_P$ then $\models ((F_P^S(s) \wedge F_P^A(a)) \backslash (x_1,...,x_m) \leftrightarrow F_P^S(t)[y_1'/y_1,...,y_n'/y_n])$, where $\{x_1,...,x_m\}$ is the set of the variables in $F_P^S(s)$, $\{y_1,...,y_n\}$ is the set of the variables in $F_P^S(t)$, the set of variables in $F_P^A(a)$ is the subset of $\{x_1,...,x_m\} \cup \{y_1',...,y_n'\}$.

4 APPROXIMATED REFINEMENT RELATION ON HZIAS

The refinement relation aims at formalizing the relation between abstract and concrete versions of the same component, for example, between an interface specification and its implementation.

Since HZIA have some real-numbered variables which may be measured with small errors. We should consider the measuring errors of real-numbered variables in the definition of approximated refinement relation on HZIAs. Roughly, a HZIA P approximately refines a HZIA Q if all the input or output actions of P can be simulated by Q except that some small measuring errors of real numbered variables. The precise definition must take into account the fact that the internal actions of P and Q are independent. For this, we need some preliminary notions. The closure of a state s consists of the set of states that can be reached from s by taking only internal actions.

We now give the following definition which describes the set of states after performing a sequence of internal actions from a given state.

Definition 4. Given a HZIA P and a state $s \in S_P$, the set $\varepsilon - closure_P(s)$ is the smallest set $U \subseteq S_P$ such that (1) $s \in U$ and (2) if $t \in U$, $a \in A_P^H$, and $(t,a,t^*) \in T_P$ then $t^* \in U$.

The environment of a HZIA P cannot see the internal actions of P. Consequently if P is at a state s then the environment cannot distinguish between s and any state in $\varepsilon - closure_P(s)$.

The following definition describes the set of states after performing several internal actions and an external action from a given state.

Definition 5. Consider a HZIA P and a state $s \in S_P$. For an action a, we let

$ExtDest_P(s,a) = \{s^* \mid \exists(t,a,t^*) \in T_P.t \in \varepsilon - closure_P(s)$ and $s^* \in \varepsilon - closure_P(t^*)\}$.

In the following, we use $V^I(A)$ to denote the set of input variables in Z schema A, $V^O(A)$ to denote the set of output variables in Z schema A, $V^H(A)$ to denote the set of internal variables in Z schema A, and $CV(A)$ to denote the set of real-numbered variables in Z schema A.

In order to define the approximated refinement relation between Z schemas, we need the following notation.

Definition 6. Given a positive real-numbered assignment δ on $\{x_1,...,x_m\}$ which represents the measuring errors of real-numbered variables $\{x_1,...,x_m\}$, an assignment ρ on $\{v_1,...,v_n\}$, where $\{x_1,...,x_m\}$ are set of all real-numbered variables in $\{v_1,...,v_n\}$. We use the notation $\rho \oplus \delta$ to denote the set of assignment $\{\sigma \mid \sigma(v) = \rho(v)$ if $v \notin \{x_1,...,x_m\}$, and $\sigma(x) = \rho(x) + a$, if $x \in \{x_1,...,x_m\}$, where $-\delta(x) \leq a \leq \delta(x)\}$.

Definition 7. Consider two Z schemas A and B with $V^I(A) = V^I(B)$, $V^O(A) = V^O(B)$, $V^H(A) = V^H(B) = \emptyset$ and $CV(A) = CV(B)$. δ is a positive real-numbered assignment on $CV(A)$. We use the notation $A \geq^\delta B$ if one of the following cases holds:

(1) if $V^I(A) \neq \emptyset$ and $V^O(A) \neq \emptyset$ then given an assignment ρ on $V^I(A)$, for any assignment σ on $V^O(A)$, $(\rho \cup \sigma) \models B$ implies $(\rho \cup \sigma) \oplus \delta \models A$, and given an assignment σ on $V^O(A)$, for any assignment ρ on $V^I(A)$, $(\rho \cup \sigma) \models A$ implies $(\rho \cup \sigma) \oplus \delta \models B$, where $(\rho \cup \sigma) \oplus \delta \models A$ means that there is an assignment $\lambda \in (\rho \cup \sigma) \oplus \delta$, such that $\lambda \models A$, and $\rho \cup \sigma$ is an assignment which is the union of ρ and σ;

(2) if $V^I(A) \neq \emptyset$ and $V^O(A) = \emptyset$ then for any assignment ρ on $V^I(A)$, $\rho \models A$ implies $\rho \oplus \delta \models B$;

(3) if $V^I(A) = \emptyset$ and $V^O(A) \neq \emptyset$ then for any assignment ρ on $V^O(A)$, $\rho \models B$ implies $\rho \oplus \delta \models A$;

(4) $V^I(A) = \emptyset$ and $V^O(A) = \emptyset$.

Intuitively, $A \geq^{\delta} B$ means that schemas A and B have the same input variables and the same output variables, and except of small measuring errors δ of real numbered variables, schema B has bigger domains of input variables but smaller ranges of output variables than schema A. For example, $A \widehat{=} [x? : \mathbf{N};\ y! : \mathbf{R} \mid x \text{ is an even number},\ y! = \pi \times x?] \geq^{\delta=[y!:=0.1]} B \widehat{=} [x? : \mathbf{N};\ y! : \mathbf{R} \mid y! = 2\pi \times \lfloor x?/2 \rfloor \pm 0.02]$, where $\mathbf{N}$ is the set of natural numbers, $\mathbf{R}$ is the set of reals, and $\lfloor x?/2 \rfloor$ is the largest natural number that is not larger than $x?/2$.

Now we give the approximated refinement relation between Z schemas, which describe the approximated refinement relation between data structures properties of states.

Definition 8. Consider two Z schemas A and B, we use the notation $A \trianglerighteq^{\delta} B$ if

(1) $V^I(A) \subseteq V^I(B)$, $V^O(A) \subseteq V^O(B)$, and $CV(A) \cap (V^I(A) \cup V^O(A)) = CV(B) \cap (V^I(A) \cup V^O(A))$;

(2) $A \backslash (x_1, ..., x_m) \geq^{\delta} B \backslash (y_1, ..., y_n)$, where $\{x_1, ..., x_m\} = V(A) - V^I(A) - V^O(A)$, $\{y_1, ..., y_n\} = V(B) - V^I(A) - V^O(A)$, and δ is a positive real-numbered assignment on $CV(A) \cap (V^I(A) \cup V^O(A))$.

For example, $A \widehat{=} [x? : \mathbf{N};\ y! : \mathbf{R} \mid x \text{ is an even number}, y! = \pi \times x?] \trianglerighteq^{\delta=[y!:=0.1]} B \widehat{=} [x? : \mathbf{N};\ u? : \mathbf{R};\ y! : \mathbf{R};\ v! : \mathbf{R};\ z : \mathbf{N} \mid y! = 2\pi \times \lfloor x?/2 \rfloor \pm 0.02;\ v! = z * u?]$.

In the following, we give a approximated refinement relation between HZIAs. For HZIAs, a state has not only behaviour properties but also data properties. Therefore this approximated refinement relation involves both the refinement relation between behaviour properties and the approximated refinement relation between data properties.

Definition 9. Given a HZIA P and a configuration $(s, D_P) \in S_P \times \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers, the set $\varepsilon - transition_P((s, D_P), d) = \{(s^*, D_P + d) \mid t \in \varepsilon - closure_P(s)$, for every $0 \leq e \leq d$, the invariant $F_P^S(t)$ holds for $C_P = D_P + e$, and $s^* \in \varepsilon - closure_P(t)\}$.

Intuitively, $\varepsilon - transition_P((s, T_P), d)$ is the set of configurations after performing some internal actions with d time delay.

Definition 10. Given a HZIA P and a configuration $(s, D_P) \in S_P \times \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers, the set $\varepsilon - delay_P((s, D_P), d) = \{(s^i, D_P^i) \mid (s^1, D_P^1) \in \varepsilon - transition_P((s, D_P), d_1), ..., (s^i, D_P^i) \in \varepsilon - transition_P((s^{i-1}, D_P^{i-1}), d_i)$, where $d = d_1 + ... + d_i\}$.

The set $\varepsilon - delay_P((s, D_P), d)$ is the set of configurations after performing several internal actions with time delay d.

Definition 11. Consider a HZIA P and a configuration $(s, D_P) \in S_P \times \mathbf{R}$, where $\mathbf{R}$ is the set of real numbers. For an action a, we let

$ExtDest_P((s, D_P), a, d) = \{(s^*, D_P^*) \mid (t, D_P^1) \in \varepsilon - delay_P((s, D_P), d_1),\ D_P^1 = D_P + d_1.\ \exists (t, a, t^*) \in T_P, (s^*, D_P^*) \in \varepsilon - delay_P((t^*, D_P^1), d_2)$, and $d = d_1 + d_2$, $D_P^* = D_P^1 + d_2\}$.

The set $ExtDest_P((s, D_P), a, d)$ is the set of configurations after performing several internal actions and an external action with time delay d.

The approximated refinement relation between HZIAs is similar to that of original ZIAs (Cao, 2010) except that a transition with time delay d should be matched by another transition with time delay d and small measuring errors δ of real numbered variables should be considered.

Definition 12. Consider two HZIAs P and Q. Suppose $CV_P \cap (V_P^I \cup V_P^O) = CV_Q \cap (V_Q^I \cup V_Q^O)$, where CV_P (CV_Q) denotes the set of real-numbered variables in HZIA P (Q). δ is a positive real-numbered assignment on $CV_P \cap (V_P^I \cup V_P^O)$. A binary relation $R_A^{\delta} \subseteq (S_P \times \mathbf{R}) \times (S_Q \times \mathbf{R})$ is an approximated simulation from Q to P, if for all configurations $(s, D_P) \in S_P \times \mathbf{R}$, there exists $(t, D_Q) \in S_Q \times \mathbf{R}$ such that $(s, D_P)\ R_A^{\delta}\ (t, D_Q)$ the following conditions hold:

(1) $F_P^S(s) \trianglerighteq^{\delta} F_Q^S(t)$;

(2) For any $(s^*, D_P^*) \in \varepsilon - delay_P((s, D_P), d)$, there is a configuration $(t^*, D_Q^*) \in \varepsilon - delay_Q((t, D_Q), d)$, such that $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P^*)\ R_A^{\delta}\ (t^*, D_Q^*)$;

(3) For any input action a, if $(s^*, D_P^*) \in ExtDest_P((s, D_P), a, d)$, there is a configuration $(t^*, D_Q^*) \in ExtDest_Q((t, D_Q), a, d)$, such that $F_P^A(a) \trianglerighteq^{\delta} F_Q^A(a)$, $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P^*)\ R_A^{\delta}\ (t^*, D_Q^*)$;

(4) For any output action a, if $(s^*, D_P^*) \in ExtDest_P((s, D_P), a, d)$, there is a configuration $(t^*, D_Q^*) \in ExtDest_Q((t, D_Q), a, d)$, such that $F_P^A(a) \trianglerighteq^{\delta} F_Q^A(a)$, $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P^*)\ R_A^{\delta}\ (t^*, D_Q^*)$.

We write $(s, D_P) \succeq_A^{\delta} (t, D_Q)$ if there is an approximated simulation R_A^{δ} such that $(s, D_P)\ R_A^{\delta}\ (t, D_Q)$.

Definition 13. The HZIA Q approximately refines the HZIA P written $P \succeq_A^{\delta} Q$ if

There is an approximated simulation $\succeq_A^{\delta}$ from Q to P, a configuration $(s, 0) \in S_P^i$ and a configuration $(t, 0) \in S_Q^i$ such that $(s, 0) \succeq_A^{\delta} (t, 0)$.

We have the following proposition about approximated refinement relation $\succeq_A^{\delta}$.

Proposition 1. (1) $P \succeq_A^{\delta} P$, where $\delta(x) = 0$ for any x;

(2) If $P \succeq_A^{\delta} Q$ and $\delta(x) \leq \eta(x)$ for any x, then $P \succeq_A^{\eta} Q$;

(3) If $P \succeq_A^{\delta} Q$ and $Q \succeq_A^{\eta} R$, then $P \succeq_A^{(\delta+\eta)\downarrow_{dom(\delta)}} R$, where $(\delta+\eta)\downarrow_{dom(\delta)}$ is a positive real-numbered assignment such that the domain of $(\delta+\eta)\downarrow_{dom(\delta)}$ is the same as that of δ and $((\delta+\eta)\downarrow_{dom(\delta)})(x) = \delta(x)+\eta(x)$ for any x in the domain of δ.

In the following, we present a simplified approximated refinement relation $\succeq_S^{\delta}$ where the appearance of time delay d in Clauses (3) and (4) in the definition of $\succeq_A^{\delta}$ is replaced by 0. Then we show the equivalence between $\succeq_A^{\delta}$ and $\succeq_S^{\delta}$.

Definition 14. Consider two HZIAs P and Q. Suppose $CV_P \cap (V_P^I \cup V_P^O) = CV_Q \cap (V_P^I \cup V_P^O)$, where CV_P (CV_Q) denotes the set of real-numbered variables in HZIA P (Q). δ is a positive real-numbered assignment on $CV_P \cap (V_P^I \cup V_P^O)$. A binary relation $R_S^{\delta} \subseteq (S_P \times \mathbf{R}) \times (S_Q \times \mathbf{R})$ is a simplified approximated simulation from Q to P, if for all configurations $(s, D_P) \in S_P \times \mathbf{R}$, there exists $(t, D_Q) \in S_Q \times \mathbf{R}$ such that $(s, D_P)\ R_S^{\delta}\ (t, D_Q)$ the following conditions hold:

(1) $F_P^S(s) \trianglerighteq^{\delta} F_Q^S(t)$;

(2) For any $(s^*, D_P^*) \in \varepsilon - delay_P((s, D_P), d)$, there is a configuration $(t^*, D_Q^*) \in \varepsilon - delay_Q((t, D_Q), d)$, such that $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P^*)\ R_S^{\delta}\ (t^*, D_Q^*)$;

(3) For any input action a, if $(s^*, D_P) \in ExtDest_P((s, D_P), a, 0)$, there is a configuration $(t^*, D_Q) \in ExtDest_Q((t, D_Q), a, 0)$, such that $F_P^A(a) \trianglerighteq^{\delta} F_Q^A(a)$, $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P)\ R_S^{\delta}\ (t^*, D_Q)$;

(4) For any output action a, if $(s^*, D_P) \in ExtDest_P((s, D_P), a, 0)$, there is a configuration $(t^*, D_Q) \in ExtDest_Q((t, D_Q), a, 0)$, such that $F_P^A(a) \trianglerighteq^{\delta} F_Q^A(a)$, $F_P^S(s^*) \trianglerighteq^{\delta} F_Q^S(t^*)$, and $(s^*, D_P)\ R_S^{\delta}\ (t^*, D_Q)$.

We write $(s, D_P) \succeq_S^{\delta} (t, D_Q)$ if there is a simplified approximated simulation R_S^{δ} such that $(s, D_P)\ R_S^{\delta}\ (t, D_Q)$.

Definition 15. The HZIA Q simply approximately refines the HZIA P written $P \succeq_S^{\delta} Q$ if

There is a simplified approximated simulation $\succeq_S^{\delta}$ from Q to P, a configuration $(s, 0) \in S_P^i$ and a configuration $(t, 0) \in S_Q^i$ such that $(s, 0) \succeq_S^{\delta} (t, 0)$.

We have the following proposition about the relation between approximated refinement relations $\succeq_A^{\delta}$ and $\succeq_S^{\delta}$.

Proposition 2. (1) $(s, D_P) \succeq_A^{\delta} (t, D_Q)$ iff $(s, D_P) \succeq_S^{\delta} (t, D_Q)$ for any HZIA P and Q, any state s in P and t in Q, and any real number D_P and D_Q;

(2) $P \succeq_A^{\delta} Q$ iff $P \succeq_S^{\delta} Q$ for any HZIA P and Q.

5 CONCLUSIONS

This paper proposed a specification approach of hybrid software/hardware components which is suitable to specify the hybrid behaviour and data structures properties of a system. There are several other works for such topic (Fischer, 1996; Fischer, 1997; Fischer, 1998). But the specification languages in these works are based on process calculi which are abstract and difficult to be applied by programmers. Interface automata are a kind of models for programmers to specify the behaviour properties of a system. However it can not specify the hybrid property and data structures property of a system. In this paper, we define a combination of interface automata, hybrid automata and Z called HZIA, which can be applied to specify the behaviour and data structures properties of a hybrid system. Moreover, it is intuitive, and it is easy to be understood and to be applied by programmers. We also define the approximated refinement relation for HZIA. The properties of the approximated refinement relation for HZIA are also studied. HZIA is well suited for specification and development of hybrid software/hardware components. It provides powerful techniques to model control aspects, hybrid property and data structures in a common framework.

REFERENCES

Bowen, J. (2003). *Formal Specification and Documentation using Z: A Case Study Approach*. Thomson Publishing.

Cao, Z. (2010). Refinement checking for interface automata with z notation. *Proceeding of Software Engineering and Knowledge Engineering*, pages 399–404.

Fischer, C. (1996). *Combining CSP and Z*. Technical report, TRCF-97-1, University of Oldenburg.

Fischer, C. (1997). Csp-oz: A combination of object-z and csp. *Proceedings of FMODDS'97*, pages 423–438.

Fischer, C. (1998). How to combine z with a process algebra. *Proceedings of ZUM'98*, pages 5–23.

Henzinger, T. A. (1996). The theory of hybrid automata. *Proceedings of LICS 96*, pages 278–292.

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. Prentice-Hall.

Luca de Alfaro, T. A. H. (2001). Interface automata. *Proceedings of FSE2001*, pages 109–120.

Spivey, J. M. (1998). *The Z Notation: A Reference Manual. Sencond Edition*. Prentice Hall International.

Woodcock, J. and Davies, J. (1996). *Using Z - Specification, Refinement, and Proof*. Prentice-Hall.