Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080454

Approximate Bisimulation Equivalence and Variable Refinement

Bai Liu[1,∗], *Jinzhao Wu*[2] *and Zhucheng Xie*[3]

[1] Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu, Sichuan 610041, China
[2] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning, Guangxi 530006, China
[3] YouJiang Medical University for Nationalities, Baise, Guangxi 533000, China

Received: 23 Aug. 2013, Revised: 25 Nov. 2013, Accepted: 26 Nov. 2013
Published online: 1 Jul. 2014

Abstract: In this paper, we consider an operator for refinement of variables to be used in the design of hybrid system. Variables on a given level of abstraction are replaced by more complicated processes on a lower level just like the function are called in the program. Then we established the equivalence, bisimulation equivalence and approximate bisimulation equivalence which are by polynomial flow event structures. These equivalence, bisimulation equivalence and approximate bisimulation equivalence are based on the common forms of their zeros. The example show that the equivalence, bisimulation equivalence and approximate bisimulation equivalence are preserved or not under the variables refinement, if the equivalence is not preserved precisely, then we can use the approximate methods to make them approximate equivalence. Lastly we show that our refinement has some nice properties.

Keywords: Equivalence, Approximate, Bisimulation, Variable Refinement, polynomial flow event structures

1 Introduction

For traditional methods a complex system is always to be described as a simple abstract specification and then to be refined it into a more concrete implementation. This is called the hierarchical specification methodology. It has been successfully developed for systems where abstract-level instructions are expanded until a concrete implementation is reached [1]. In the method the core operation is refinement, which leads to a successful technique known as top-down system design. It allows the representation of systems in a hierarchical way.

The author discussed the action refinement detailed in concurrent systems in literature [2,3]. The action refinement operator is generally described on three aspects in these literatures, on language level of various process algebra [4,5,6], on Logic levels, and on structure levels (transition systems [7,9] and event structure [8]). However, the tradition systems are mostly discrete, and the discrete methods biggest drawback is unable to describe and process the data flow. Therefore, recently years, more and more people began to pay attention to hybrid system [10,11,12], now it is one of the most importance field of computer science. The paper [13] firstly used polynomial equation to define the state transition hybrid system, and through polynomials Groebner base generate the invariant. Based on these, some equivalence and approximate equivalence were proposed [14,15,16]. But the refinement of hybrid system is not mentioned. So in this paper, we investigate variable refinement in hybrid systems that encompass the notation of polynomial equations with variables.

In this paper we choice the model based on the flow event structures [17]. Because the model of flow event structure has some advantages: firstly flow event structures closely resemble prime event structures [18], but flow event structure overcome the unique enabling problem of prime event structures, and flow event structures are suitable for defining parallel composition; secondly, flow event structure overcomes the stability constraint [18] problem of stable event structures; thirdly, flow event structure overcome the all events in a bundle set should in conflict [18] of bundle event structures. In[18], Glabbeek and Goltz defined an operator of action refinement on flow event structures and other causality based event-oriented models of concurrency.

∗ Corresponding author e-mail: liubai999@hotmail.com

In this paper, we use polynomial flow event structures which is based on the flow event structure as the model, and refine the variables of the polynomials. Why use polynomials? Firstly, polynomials enhance the ability of description. Secondly, polynomials are among the simplest functions. Thirdly, computer can directly evaluate polynomials, so polynomials have both theoretical and practical relevance. Fourthly, refinement variables are very easy to do. Fifthly the events from the abstract to the concrete, we can more clear understanding of the event itself.

The variables refinement operation should be safe. If the behaviors of two systems before refinement are same, then after the variables refinement they are should be the same too. So an important question being considered is the following: which equivalence notions for concurrent systems are preserved under refinement? Taking two system representations, P,Q, which are equivalent for some equivalence notion $\approx$, note $P \approx Q$. f is a refined function, we would expect that$f(P) \approx f(Q)$, for any refinement and many other approaches for moving to a lower of abstraction, our expectation is based on the idea that in the original system representations and it is not know how the events will be implemented. However, it is decided that different occurrences of the events will be implemented in the same way if they have the same polynomial zero, so that any different between and that could arise after refinement is already visible in the abstract representations, namely through the use of different events names for corresponding events but have the same polynomial zero.

After refinement then we should research preservation question under the refinement. In the paper we take some equivalence notions, just like equivalence, bisimulation equivalence, and approximate bisimulation equivalence. Try to find which equivalence are preserved under the refinement. Generally speaking, we use the variables zero of polynomial as standards of equivalence. But when we study the practical cases; an exact equivalence is too restrictive and also not robust sometimes. Some equivalence are kept after refinement, but some are not. For instance, in the process of using two microwaves to heat the same bread, there are two main parameters for the microwaves, functional parameter and time parameter. If the functional parameters of two microwaves are the same, it means bread can be cooked by the two microwaves, then we can set the time for cooking bread, maybe ones time parameter is 1.0s, anothers is 0.99s, it does not matter, we can say they are abstractly the same. Such example show that some unimportant and less frequently used states are able to be ignored, and the result will not be changed much significantly. In this paper, this process can be called approximation. So if the equivalence is kept after the refinement, we can say it preserved. If not matins, we can use the approximation to keep it matins. For approximation, in recent years, many researchers have been studied in this field [19,20]. Most of the methods were applied to deal with problems for dynamical systems and process them directly, which cost a large amount of computations and had high complexities.

Symbolic computation is a useful formal method [25], which is chosen to standardize the system before processing it. After standardizing, the forms of polynomial equations are normalized, and several properties of the system will be easier to be verified.

In this paper, in order to implement the above goals, the definition of polynomial flow event structures based on flow event structures is achieved for hybrid system. Firstly refine the variables for the events. Then observe the equivalent relationship of two systems is determined by using their common zeros. With bisimulation semantic, if they can't preserve the equivalence, after ignoring unimportant conditions, we propose a notion of approximate bisimulation. The error caused by approximation is calculated and controlled by symbolic and numerical calculation.

2 polynomial flow event structures

In this section, we introduce polynomial flow event structure and operators for variable refinement and approximate equivalence.

Polynomial flow event structure describes a hybrid system as a set of events modeling actions occurrences, together with three relations: the flow relation represents possible immediate causes of events; the conflict relation expresses which events mutually exclude each other; the independent relation means events occur in any order or simultaneously. In this model the abstract events are replaced by polynomial equations, for example, the abstract event e is replaced by $y = x+1$.

Definition1. A polynomial flow event structure $(PFES)$ is a tuple $\xi < V, E, \leq, \sharp, ||| >$, where

V is a finite set of variables,

E is a denumerable set of events, every event is represented by polynomial over V,

$\leq \subseteq E \times E$ is the flow relation, it is an irreflexive relation,

$\sharp \subseteq E \times E$ is the conflict relation, it is a symmetric relation,

$||| = \subseteq E \times E - \sharp - \leq$ is the independent relation, it is a independent relation.

Note that independent means events occur in any order or simultaneously, so they neither need to happen in parallel nor to occur one before the other. Polynomial flow event structures are depicted as follows, events are polynomial with variable denoted by closed dots, conflict relation $e\sharp e'$ is indicated by a dotted line between e and e', flow relation is indicated by an arrow line, independent relation omit by inheritance.

For example, in one program, each assignment $y = l(v_1, ..., v_n)$ is an event, l is a function about variables V. $V(l(v_1, ..., v_n))$ is the variables in V that needed to evaluate for y, then we use polynomial equation to

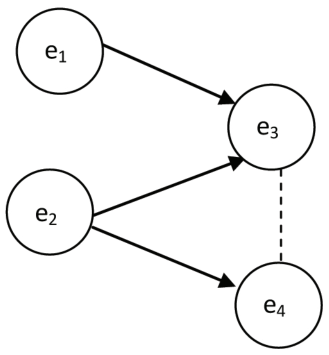

Fig. 1: The corresponding polynomial flow event structure of the program.

express the event e_y, for $V_e \in V(l(v_1,...,v_n))$, $e \le e_y$ this is the causality relation between the assignment left-side and right-side. Each group of conditional choice is a cartesian product conflict relation between the events in different choice.

Such as the following program (a part of the long program):

```
$x = 0$;
{
if $m = a$;
  $y = x + m$;
else if
  $z = x$;
}
```

The corresponding polynomial flow event structure of the program is figure 1.

In the figure,

event e_1 expressed the polynomial $m = a$;
event e_2 expressed the polynomial $x = 0$;
event e_3 expressed the polynomial $y = x + m$;
event e_4 expressed the polynomial $z = x$.

The conflict and flow relation's interpretation are formalised by defining which subsets of events constitute possible runs of the represented system, and which of these runs terminate successfully. These subsets are called configurations.

Definition2. A evaluation sequence is a sequence of distinct events $C = \{e_1,e_2,...,e_n\} \in E$, satisfying:

(1)$\{e_1,e_2,...,e_n\}$ is conflict-free, i.e. . $\forall e_i,e_j : \neg(e_i \sharp e_j)$
(2)$\forall i : e \le e_i \Rightarrow \exists j < i : e_j \le e_i \wedge (e = e_j \vee e \sharp e_j)$
C is called a configuration.

Notice that any two events will be in conflict or cause each other cannot occur in the same configuration. Hereafter, we will use $C(\xi)$ to denote the set of all the configurations of E .

A configuration C of ξ is called maximal iff $C \subseteq C' \in C(\xi)$ implies $C = C'$.

A configuration C of ξ is called successfully terminated iff $\forall d \in E : d \notin C \Rightarrow \exists e \in C$ with $d \sharp e$. We will use $PFES$ to denoted the set of all $pfes$, $\xi, \xi_i \in PFES$ and $\xi =< V,E,\sharp,||| >$, $\xi_i =< V_i,E_i,\sharp_i,|||_i >$. For two different $pfes$ ξ_f andξ_g, we always assume that$E_f \cap E_g = \emptyset$.

3 Bisimulation Equivalence

In this section, we will consider some equivalence notions for investigating them are preserved or not under the variables refinement.

3.1 Equivalence

By the definition of polynomial flow event structures, we find it is easily to find that there exists equivalence between two different polynomial flow event structures based on the common zeros of their polynomials. In this subsection, we consider that such equivalence as behavior equivalence.

Theorem 1. Events have equal zeros and equal zeros of enabling event are equivalence.

Definition3. There is an equivalence between two polynomial flow event structures ξ_1 and ξ_2, if exists $Zero(P_1) = Zero(P_2)$, notation $\xi_1 \approx_e \xi_2$, where $Zero(P_i)$ is the algebraic variety for polynomial system of ξ_i , P_i is the configurations of ξ_i which is belong to $C(\xi_i)$, and the $Zero$ are defined in an arbitrary field of characteristic 0 in the whole paper.

By the definition3 and the theorem1, the system's equivalence is very simple from the *zero* of the polynomial. But it demand us to analysis the system rigorously, and propose the most precise equivalent relations to the structure of the systems.

3.2 Bisimulation Equivalence

Based on the equivalent relationship for polynomial flow event structures, we expect to look for an abstraction which is equivalent with the whole system, we can call it a structure equivalence. Bisimulation is the finest semantic and widely used in non-deterministic equivalence verification for different concurrent systems.

Definition4. There are two polynomial systems ξ_1 and ξ_2, a relation $R \subseteq C(\xi_1) \times C(\xi_2)$ is a bisimulation equivalent relation between the two polynomial systems ξ_1 and ξ_2, where, $R_1 \in P_1, R_2 \in P_2$, $x,y \in R$, notated $\xi_1 \approx_b \xi_2$. If satisfy:

$\forall x \xrightarrow{R_1} x'$, $\exists y \xrightarrow{R_2} y'$ and $Zero(P_1) = Zero(P_2)$, such that$(x',y') \in R$.

$\forall y \xrightarrow{R_2} y'$, $\exists x \xrightarrow{R_1} x'$ and $Zero(P_2) = Zero(P_1)$, such that$(x',y') \in R$.

By the definition, we are able to find whether exist a bisimulation or not between two polynomial systems. Two polynomial systems are bisimular if there exists a causality relation for one, there exists the same causality relation for the other, and conversely the same. So for processing the same causality relation based on bisimulation, large scale system can reduced or simplified.

3.3 Approximate Bisimulation Equivalence

For two given systems, the exact equivalence requires that the common zeros must be identical or totally the same. But sometimes it is too restrictive for some real problems and also not robust in the technology field while several special systems only have approximate models. So defining the approximation sometimes for the different systems is necessary. It may have some advantages, by approximation, making the two different systems are equivalence, thus, we can select a simpler one to reduce the complex one. secondly. After behavior and structure approximate, we can get a system which is an approximate system for given system and also much simpler than it. Thus, approximate bisimulation is an appropriate relation for system. And for refinement system, if two systems behavior is the same, after refinement it is not exact equivalence, then we can use approximate to make them equivalence.

Definition5. There are two polynomial systems ξ_1 and ξ_2, a relation $R \subseteq C(\xi_1) \times C(\xi_2)$ is an approximate bisimulation equivalent relation between two polynomial flow event structure ξ_1 and ξ_2, where $R_1 \in P_1, R_2 \in P_2$, $(x,y) \in R$, $D < \varepsilon$. Notated $\xi_1 \approx_a \xi_2$. D is an error.

$\forall x \xrightarrow{R_1} x'$, $\exists y \xrightarrow{R_2} y'$ and $Zero(P_1) \approx_D Zero(P_2)$, Such that$(x',y') \in R$.

$\forall y \xrightarrow{R_2} y'$, $\exists x \xrightarrow{R_1} x'$ and $Zero(P_2) \approx_D Zero(P_1)$, Such that$(x',y') \in R$.

By the definition, we can get the approximate bisimulation relation of two system. If there exists an approximate bisimulation relation, how to calculate the zero of polyomials and the error are the key problems. We can find some methods in references [15,16,20], we also give some methods in the following section.

3.4 Approximate methods

For the given polynomial flow event structure, after establishing the approximate bisimulation equivalence relations, we should determine whether the error is in our acceptance or not.

3.4.1 Zero Computation

We can use symbolic computation to calculate zeros, just like Groebner basis and characteristic sets methods [23]. Before getting their zeros these methods need to represent the polynomial equations standardize and unique firstly. Then, based on the unique form, we can know that whether there exists equivalence or not.

Let $T(P)$ be Irreducible Characteristic Sets for P, G be Groebner basis for P, $Ideal(P)$ be Ideal for P, $GB(P)$ be Reduced Groebner Basis for P. These theories of characteristic set , Ideal and Groebner Bases can be found in referance[23].

Proposition1.If $GB(P_1) = GB(P_2)$, then $Zero(P_1) = Zero(P_2)$.

Proof: With the polynomial sets $P \subset k[x]$, an ideal I which is called ideal generated by P must exist, then

$Zero(Ideal(P)) = Zero(P)$ exist also.

With the given polynomial set and its ideal, it will be existed a Groebner basis G, and $Ideal(P) = Ideal(G) = I$. If G is irreducible, then, $Zero(P) = Zero(G)$. Therefore, $Zero(P_1) = Zero(P_2)$, if $G_1 = G_2$. However if G is reducible and has a factorization $G = G_1G_2$, then, to consider each G_i as G, and continue process as the same as the above way, there is a decomposition of the following form

$Zero(P) = \bigcup_{i=1}^{e} Zero(G_i)$,

We get its unique Reduced Groebner Basis with the Groebner basis of G_i. If $GB(P_1) = GB(P_2)$, Then

$Zero(P_1) = Zero(P_2)$,

Proposition2. If $T(P_1) = T(P_2)$, then $Zero(P_1) = Zero(P_2)$.

Proof We all known that because the multivariate polynomial rings are Noetherian over feild, thus, we can be decomposed every ideal I into some primary componets $Q : I = Q_1 \cap ... \cap Q_m$. Each Q_i is a primary. Then, by calculating $T(Q_i)$ for Q_i, It is unique, so we get a algebraic variety for the ideal, $V(I) = V(Q_1) \cup ... \cup V(Q_m) = V(T_1 \setminus I_1) \cup ... \cup V(T_m \setminus I_m)$ Thus, $Zero(P_1) = Zero(P_2)$, if $T(P_1) = T(P_2)$, where I_i is initial for T_i.

3.4.2 Approximate methods

There are some approximate methods for different polynomial equations we can use, according to the unary or multivariate variables, we can choice the different methods.

For unary, we can use the following methods:

Theorem2. Weierstrass approximation theorem: if $f(x)$ is a continuous function defined on real interval$[a,b]$, and if $D > 0$ is given, then $\exists p(x)$ on$[a,b]$ such that $\forall x \in [a,b]$, $|f(x) - p(x)| < D$.

this theorem makes sure that given a error limit there exist a approximate polynomial about $f(x)$ only needs that it is a continuous function defined on real interval$[a,b]$.

theorem3.Taylors theorem:

Let $k \leq 1$ be an integer and function $f : R \to R$ be k times differentiable at the point $a \in R$. then $\exists h_k$, such that

$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-2)^2 + ... + \frac{f^k(a)}{k!}(x-a)^k + h_k(x)(x-a)^k$, and $\lim h_k(x) = 0$. If $|f^{k+1}(x)| \leq N$ on $(a-r, a+r)$, $r > 0$, then $|D| \leq N\frac{r^{k+1}}{(k+1)!}$.

The last equation gives the error limit.

Theorem4. Piecewise linear interpolation theorem:

$I(x)$ is the piecewise linear interpolation function of $f(x) \in C^2[a,b]$, $h = max_{0\leq k\leq (n-1)}\{x_{k+1} - x_k | a = x_0 < x_1 < \ldots < x_n = b\}$, then $max_{0\leq k\leq n-1}|f(x) - I(x)| \leq \frac{1}{8} max_{a\leq x\leq b}|f''(x)|h^2$.

Theorem5. The best first order approximation theorem:

Suppose $f(x) \in C^2[a,b]$, $f''(x) \in (a,b)$ does not change sign, then the best first order approximation p(x) satisfies, $p(x) = \frac{1}{2}[f(a) + f(x_2)] + \frac{f(b)-f(a)}{(b-a)}(x - \frac{(a+x_2)}{2})$

$f(x^2) = \frac{f(b)-f(a)}{(b-a)}$

$max_{a\leq x\leq b}|f(x) - p(x)| = f(x_2) - p(x_2)$

The last equation gives the error limit.

For multivariate, we can use the following methods:

Given two polynomial flow event structures, we are going to determine if there is an approximate bisimulation equivalence or not for the two systems with an error D when $D < \varepsilon$, andε is given. Let M and N be two polynomial flow event structures. In fact if they are equivalent the number and value of variables must be the same. The number of variables also must be the same even if they are approximate equivalent. In this section, for the approximate bisimulation equivalence, we mainly use numerical calculation to compute the error D between the two systems. The detailed algorithm are following.

Step 1. Firstly we should order the relation $<$ for the variables of two systems are: $x < y < x_1 < \ldots < x_n; x < y < y_1 < \ldots < y_n$;

Step 2. Then choose one method about symbolic computation to computer the equivalent relationship between M and N.

If $GB(M) = GB(N)$, then, M and N are equivalent, end. Else, go to Step 3.

Step 3. According to the prem program [24]to get $prem(x_1,$

$y_1, y), \ldots, prem(x_n, y_n, y)$; Then, $I_1^{q_1} x_1 = Q_1 y_1 + R_1, \ldots, I_n^{q_n} x_n = Q_n y_n + R_n$;

where, $I_i = lc(y_i, y)$; $l_i = deg(x_i, y)$; $m_i = deg(y_i, y)$; $q_i = max(l_i - m_i + 1, 0)$;

Thus,$|x_i - \frac{Q_i}{I_i^{q_i}} y_i| = |\frac{R_i}{I_i^{q_i}}|$;

Then, to compute

$min(\sum_{i=1}^{n}(\|x_i - \frac{Q_i}{I_i^{q_i} y_i}\| + \|y_i - \frac{Q_i}{I_i^{q_i}\|})) = min(\sum_{i=1}^{n}($

$\|\frac{R_i}{I_i^{q_i}}\| + \|\frac{Q_i}{I_i^{q_i}}\|)) < \varepsilon$.

The arrange of some variable can be achieved, Then, M and N, N and N will be approximate equivalent.

Thus, y_i turns into $\frac{Q_i}{I_i^{q_i} y_i}$, N turns into N. and $N = \{\frac{Q_1}{I_1^{q_1} y}, \ldots, \frac{Q_n}{I_n^{q_n} y}\}$, end.

Else, M and N will be not approximate equivalent, end.

In the three steps, we will find that $\frac{Q_i}{I_i^{q_i}}$ are an approximate greatest common factor for two polynomial system M and N. By analysis, equivalence and approximate equivalence methods are useful methods, they can reduce the structure and behavior for the large scale original systems.

4 Variables refinement

4.1 Refinement

The refinement of variables, as formalized in refinement action[21], is based on the following definition6. A variable is refined by an event structure, denoted by $v = f(v')$. It is just close to the event which is used in Murphy and Pitt [22].

Definition6. We called $f : v \rightarrow pfes$ a refinement function. $f(v')$ is a refinement of v . And $f(v')$ require have non-empty configurations, so $f(v')$ couldnt be forgetful [17].

Let f is the refinement function. Then we define the refined $pfes$ of ξ to be

$f(\xi) = (V_f, E_f, <_f, \sharp_f, |||_f)$, where

$V_f = \{V \cup V' | V \in E \cup V' \in E_f\}$,

$E_f = \{(e,e')|(e \in E) \wedge (V = f(V')) \wedge l(V \cup f(V')) = e'\} \cup \{(e,e)|(e \in E) \wedge l(V) = e\}$

For $(e_1,e_2),(e_1',e_2') \in E_f$, $(e_1,e_2)\sharp(e_1',e_2')$ iff

If$e_1 = e_1'$, then $e_2\sharp_f(\xi)e_2'$, or $e_2, e_2' \in exit(f(V'))$ and e$e_2 \neq e_2'(e_2 \neq e_1$ and $e_2' \neq e_1'$ in the case),

If$e_1 \neq e_1'$ then

$e_2 = e_1$ and $e_2' = e_1'$ then $e_1\sharp e_1'$,

If $e_2 \neq e_1$ and $e_2' = e_1'$ then $e_1\sharp e_1'$ and $e_2 \in \{l(f(V') \cup V) \cup exit(l(v \cup f(V')))\}$,

If$e_2 = e_1$ and $e_2' \neq e_1'$ then $e_1\sharp e_1'$ and $e_2' \in \{l(f(V') \cup V) \cup exit(l(v \cup f(V')))\}$,

If$e_2 \neq e_1$, $e_2' \neq e_1'$ then $e_1\sharp e_1'$ and $e_2 \in \{l(f(V') \cup V) \cup exit(l(v \cup f(V')))\}$, and $e_2' \in \{l(f(V') \cup V) \cup exit(l(v \cup f(V')))\}$,

For (e_1,e_2), $(e_1',e_2') \in E_f$, $(e_1,e_2) < (e_1',e_2')$ iff $e_1 < e_1'$ or $(e_1 = e_1' \vee e_2 < e_2')$

4.2 Example

There is a program and the program have the calling function, the bellowing is the program and the corresponding polynomial flow event structures in figure2.

The program code(partially):

```
x,y;
{
if $y > 0$
    $r = e^x + y$;
else
    $p = e^x$;
    $q = x$;
}
```

In the figure2, the events are corresponding to the polynomial of the program.

e_1 expressed the variables x, y;

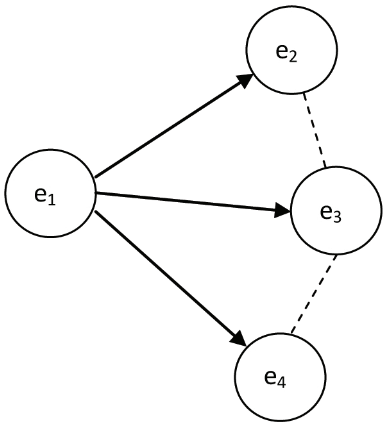

Fig. 2: The corresponding polynomial flow event structures of the program.

e_2 expressed the polynomial $p = e^x$;
e_3 expressed the polynomial $r = e^x + y$;
e_4 expressed the polynomial $q = x$,
we called the polynomial flow event structure is ξ .
Then if the program called the subfunction y,
The subfunction of the program code:

```
f()
{
if $x \geq 0$
    $y = e^x$;
else
    $y = e^{(-x)}$;
}
```

Then we can get the variable refinement's polynomial flow event structure $f(y)$ in figure3, e_0 expressed the empty event.

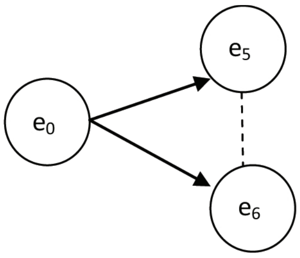

Fig. 3: The polynomial flow event structures of $f(y)$.

let ξ be the $pfes$ in figure2. And the variable refinement's polynomial flow event structure $f(y)$ in figure3. Then the $pfes$ of $f(\xi)$ is figure4.

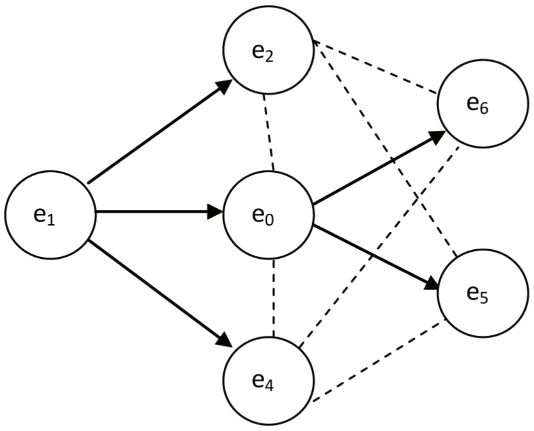

Fig. 4: The polynomial flow event structures $f(\xi)$.

5 Properties

Proposition3.

(1)If $\xi \in E_{flow}$, and f is a refinement function, then $f(\xi)$ is a polynomial flow event structure.

(2)If $\xi \in E_{flow}$, and f , f' are refinement functions with $f(v) \approx_x f(v')$,$x \in \{e,b,a\}$, $v \in V$, then $f(\xi) \approx_x f(\xi)'$.

(3)If $\xi, F \in E_{flow}$, $\xi \approx_x F$,$x \in \{e,b,a\}$, and f is a refinement function, then $f(\xi) \approx_x f(F)$ is a polynomial flow event structure.

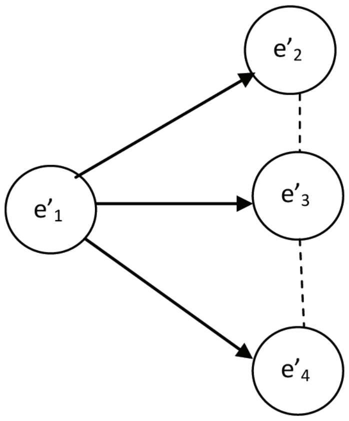

Fig. 5: The polynomial flow event structure F .

Example: The figure5 is the polynomial flow event structure of$F$, e'_1 expressed the variablesx, y; e'_2 expressed the polynomial $p = 1 + x + x^2/(2!) + \quad + x^9/(9!)$; e'_3 expressed the polynomial $r = 1 + x + x^2/(2!) + \quad + x^9/(9!) + y$; e'_4 expressed the polynomial $q = x$. We assume the error D is 10^{-5}, then

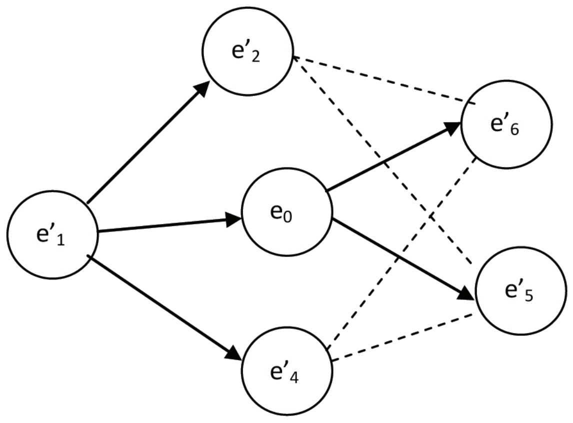

Fig. 6: The polynomial flow event structure $f(F)$.

we can see that F is approximate bisimulation equivalent with ξ. Notate that $\xi \approx_a F$, after refinement y we can get the $f(F)$ in figure6 . Now we can see that $f(\xi) \approx_a f(F)$.

C_f is a configuration of $f(\xi)$, let $\pi_1(C_f) = \{e|(e,e') \in E(C_f)\}$, so $e \in \pi_1(E(C_f))$, let $\pi_2(C_f,e) = \{e'|(e,e') \in E(C_f)\}$, in fact $\pi_1(C_f)$ is the projection of C_f on ξ, and $\pi_2(C_f,e)$ is the projection of C_f on $f(\xi)$. From the following lemma we can see that the projection of C_f on ξ is a configuration of ξ, and the projection of C_f on $f(\xi)$ is a configuration of $f(\xi)$.

Lemma1.

(1) $\pi_1(C_f) \in C(\xi)$;

(2) $\pi_2(C_f,e) \in C(f(\xi))$;

(3)If e isn't maximal in $E(\pi(C_f))$, (C_f,e) can successfully terminates.

C is a configuration of ξ, C_e is a configuration of $f(v)$, $f(v)$ satisfying the following condition: if event e is not maximal in C where $e \in E(C)$ and $v \in V(e)$, $V(e)$ is the variables for valuation event e , C_e can successfully terminates. Let $f(C,\cup_e C'_e) = \{(e,e')|e \in E(C)$ and if $\{v \in V(e)|e' = e\}$. otherwise $v \in (V(e) \cup f(v)), e' = l(V(e) \cup E(C_e))\}$. In fact $f(C,\cup C'_e)$ is obtained by a configuration C_E of $f(v)$ with replacing each event e in configuration with $l(V(e) \cup f(v))$.

Lemma2. $f(C,\cup_e C'_e) \in C(f(\xi))$.

From the lemma, we can see that the configurations of the refined $pfes$ $f(\xi)$ can get from the configuration refinements of ξ.

Theorem6. $C(f(\xi)) = \{C_f|C_f$ is the refinement of $C \in C(\xi)\}$.

We use $f(v)$ to substitute the variables v of event in ξ.the start event is the start point of the system. Theorem 5.3 denotes that the refinement of a configuration of ξ can be obtained by calling the function of variable of events. The causality relation in each C is respected to the refinement of C . The causality relation is that if e causes e' in C, then some successful termination event of $l(V(e) \cup f(v))$ causes the start-event of $l(V(e') \cup f(v'))$.

So the behavior of $f(\xi)$ can be get from the behavior of ξ calling for the refinement variable v of the behavior of $f(v)$.

Example:we consider again ξ in figure6 of example. $C_f = e_1, e_0, e_5$ is a configuration of $f(\xi)$.

$\pi_1(C_f) = \{e_1, e_3\}$

$\pi_2(C_f, e_3) = \{e_0, e_5\}$

π_1 is a configuration of ξ and π_2 is a configuration $f(v)$.

6 Conclusion

In this paper, we proposed the new concept of polynomial flow event structure which is suitable for parallel composition, refinement and many other operators of CCS-like languages. And then we translated the program code to *PFES*, Considered an operator for refinement of variables on polynomial flow event structure. For the variables refinement we demonstrated that they are safe and equivalence (equivalence, bisimulation equivalence or approximate bisimulation equivalence) under refinement. The equivalence (equivalence, bisimulation equivalence or approximate bisimulation equivalence) is based on the common forms of their zeros. With using symbolic method and numerical calculation, the value of zero or error is can calculate.

But how to improve the efficiency of our approach are still a very big challenge as well as our main future work. In addition, the system is limited which can be expressed by polynomial flow event structure, we also demand to establish more appropriate systems to describe different systems in the future just like different system in the future.

Acknowledgement

The work is partially Supported by the National Natural Science Foundation of China under Grant No.11371003, the Natural Science Foundation of Guangxi under Grant No.2011GXNSFA018154 and No.2012GXNSFGA060003, the Science and Technology Foundation of Guangxi under Grant No.10169-1, and the Scientific Research Project No.201012MS274 from Guangxi Education Department.
The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] N. Wirth. Program Development by Stepwise Refinement, Communications of the ACM, **14**, 221 (1971).

[2] Gorrieri, R., Rensink, A. Action Refinement, Technical Report UBLCS-99-09, University of Bologna, (To appear as Chapter XVI of the Handbook of Process Algebra, Elsevier Science), (1999).
[3] Aceto, L, Action refinement in Process Algebra, Cambridge Univ. Press, (1992).
[4] C. A. R. Hoare, Communicating sequential process, Prentice-Hall, (1985).
[5] J. C. M. Baeten, W. P. Weijland, Process Algebra, Cambridge Uni., Press, (1990).
[6] R. Milner, A calculus of communicating systems, LNCS92, Springer, (1980).
[7] J. P. Queille, J. Sifakis, Specification and Verification of Concurrent Systems in CESAR, Lecture Notes in Computer Science, **137**, 337-351 (1982).
[8] G. Winskel, An introduction to event structure, in de Bakker et al., 364-397.
[9] E. M. Clarke, E. A. Emerson, and P. A. Sistla, Automated Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications, ACM Trans. Program. Lang. Syst., **8**, 244-263 (1986).
[10] B. Becker, A. Podelski, W. Damm, M. Franzle, E.-R. Olderog, and R. Wilhelm, SFB/TR 14 AVACS CAutomatic Verification and Analysis of Complex Systems, IT Info. Technology, **49**, 118-125 (2007).
[11] J. A. Bergstra, C. A. Middleburg. Process algebra for hybrid systems. Theoretical Computer Science, **335**, 215-280 (2005).
[12] A. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, The Algorithmic Analysis of Hybrid Systems, Theo. Computer Sci, **138**, 3-34 (1995).
[13] S. Sankaranarayanan, H. B. Sipma, Z. Manna. NonLinear Loop Invariant Generation Using Groebner Bases. POPL04. ACM Press, (2004).
[14] Hao Yang, Anping He, Zhiwei Zhang, Approximate Completed Trace Equivalence of Linear Algebra Transition Systems. Proceedings of The Eighth International Conference on Bio-Inspired Computing, Advances in Intelligent Systems and Computing, **212**, 233-237 (2013).
[15] Hui DENG, Jinzhao WU. Approximate Equivalence and Optimization for High-Level Datapath. Journal of Information and Computational Science, **8**, 4131-4142 (2011).
[16] Girard, A.; Pappas, G.J. Approximate Bisimulations for Nonlinear Dynamical Systems, Decision and Control, and 2005 European Control Conference. CDC-ECC ’05. 44th IEEE Conference on, 684-689 (2005).
[17] G. Winskel, Event structures, in: W. Brauer, W. Reisig, G. Rozenberg (Eds.), Petri Nets: Applications and Relationships to other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proc. Advanced Course, Bad Honnef, September 1986, Lecture Notes in Computer Science, Springer, Berlin, **255**, 325-392 (1987).
[18] R. J. van Glabbeek, U. Goltz. Refinement of actions and equivalence notions for concurrent systems, Acta Inform, **37**, 229-327 (2001).
[19] A. Girard, G. J. Pappas, Approximate metrics for discrete and continuous systems, IEEE Trans. Autom. Control, 782-798 (2007).
[20] A. Girard, G. J. Pappas, Approximate bisimulations for constrained linear systems, Automatica, 1307-1317 (2007).
[21] R. van Glabbeek and U. Goltz, Well behaved flow event structures for parallel composition and action refinement, Theoretical Computer Science, **311**, 463-478 (2004).
[22] Murphy, D., Pitt, D., Real-Timed Concurrent Refineable Behaviours, Lecture Notes in Computer Science, **571**, 529-545 (1992).
[23] Wang Liping. Groebner bases and characteristic sets for ideal. Ji Lin university, (2008).
[24] E. Asarin, T. Dang, A. Girard, Reachability of non-linear systems using conservative approxima- tions, Hybrid Systems: Computation and Control, 22-35 (2003).
[25] D. Wang, Selected Lectures in Symbolic Computation, Tsinghua University Press, Beijing, (2003).

Bai Liu is a doctorial student in Chengdu Institute of Computer Application, Chinese Academy of Sciences, China. She has published more than ten articles in international journal about computer sciences. Her research interests are in the areas of formal verification and intelligent algorithm

Jinzhao Wu is currently a professor in Guangxi University for Nationalities and Beijing Jiaotong University. His research interests lie in formal verification and software engineering.

Zhucheng Xie is a teacher in YouJiang Medical University for Nationalities. his research interests are in the areas of formal verification, neural network and network security.