Intra-Transition Data Dependence

Shenghui Shi
College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China
Email: shish@mail.buct.edu.cn

Qunxiong Zhu* Zhiqiang Geng Wenxing Xu
College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, China
Email: zhuqx@mail.buct.edu.cn gengzhiqiang@mail.buct.edu.cn estellaxu@163.com

***Abstract*—Currently, two main approaches to data dependence of EFSM(Extended Finite State Machine) haven't refined intra-transition data dependence, instead they consider that every definition variable in a transition depends on all the use variables (including condition variables). For data dependence of a specific definition variable, not only the relevant use variables but also the irrelevant use variables (including condition variables) are considered, which obviously causes redundancy. Without a doubt, further analysis based on this brings hidden danger to the dependent analysis of the entire system and practical application. With the idea of introducing program dependence graph into to EFSM, this paper studies intra-transition data dependence, and describes the data dependence between every intra-transition definition variable and the use and condition variables which influence or are influenced by it. Thus irrelevant dependence variables are removed to reduce redundancies and errors. Also, theoretical and experimental analyses are conducted.**

***Index Terms*—Extended Finite State Machine(EFSM); Dependence Analysis; Intra-Transition Data Dependence(IaTDD)**

I. INTRODUCTION

With the gradual expansion of computer software applications, the size and complexity of computer software are growing rapidly, which leads to an increasing growth of the cost and difficulty of software analysis, understanding, test, maintenance, evolution, and other aspects in software engineering. Software slicing, as an "energy saving" tool for software system, therefore, plays an important role. In 1979, M. Weiser first proposed the basic idea of program slicing [1] to achieve the program's reduction. After thirty years of development, program slicing has been widely recognized and applied. From the point of view of software engineering development cycle, program slicing has penetrated into the application of requirement and design layer from coding and testing layer. In 1990s, Heimdahl et al [2,3] proposed model-based slicing, which started model slicing research of FSM (Finite State Machine). In the same period, Savage, P. and Dssouli R. proposed model slicing based on EFSM [4, 5]. In 2003, Korel [6] normalized EFSM model structure by specifying the composition of EFSM and transitions, developed EFSM slicing tool, and proposed irrelevant control dependence. But EFSM model must be built on the premise that there is a termination node. Korel applied the method of program slicing, but failed to conduct in-depth study of the differences between program slicing and EFSM structure. However, this method made EFSM model more clear and specific, which lays a solid theoretical basis for the present extensive study of the EFSM slicing technology. With further research and development of programming languages, the limitations for program slicing method to be used in the EFSM were gradually exposed. Scholars have devoted more attention to control dependence, and several solutions have been put forward. But there are also many limitations for the corresponding data dependence. Due to the limitations of requirements of study objects and their own structure, the data dependence of EFSM has not yet been well solved.

Currently, there are two main methods for the implementation of EFSM data dependence. The first one is traditional EFSM data dependence proposed by Korel[6] (hereinafter referred to as K method). This commonly used method uses the data dependent methods of program slicing, which realizes EFSM data dependence based on traversing algorithm of marking visited nodes. The other is the transitive dependence function method proposed by Chinese scholars of Miao Li and so on[7] (hereinafter referred to as M method), which analyzes the problem that data dependence of EFSM may be intransitive.

The two main methods ignore the specific data dependence of intra-transition variables, and consider that any definition variable depends on all the use variables (including condition variables) in that transition. Actually, a certain definition variable is associated only with the relevant use variables. But if we randomly identify that all use variables are related to a certain definition variable, those irrelevant use variables would be certainly included in the data dependence, which would result in redundancy. This paper presents intra-transition data dependent method, and is verified by experiments to analyze the degree of redundancy reduction.

The paper is organized as follows: Section Ⅱ provides an overview of intra-transition data dependence. Section Ⅲ analyzes the differences between traditional intra-transition data dependence method and the method put forward in this paper. Section Ⅳ compares the method in this paper and the traditional method by means of

© 2012 ACADEMY PUBLISHER
doi:10.4304/jsw.7.12.2663-2670

experiment, and analyses the result. Finally, future research is discussed.

II. INTRA-TRANSITION DATA DEPENDENCE

Intra-transition data dependence gets definition variable set and the relation set between use variable set and condition variable set.

Definition 1: Data Dependence between the Variables(DDV)[8]

DDV is an inner transition set composed of definition variable set and relationship between use variable set and condition variable set, which can be expressed as follows:

DDV: (v_{di}, {V_{ui}, V_{ci}})

Where all the variables and variable sets are in a transition, and V_d is definition variable set in action or event, v_{di} is a definition variable in action or event, $v_{di} \in V_d$. V_{ui} is use variable set influencing the value of v_{di} in action, which can be Null. V_{ci} is condition variable set influencing the value of v_{di} in condition, which can be Null. In $d \in I$, $u \in I$,$c \in I$, I represents integer. (v_{di}, {V_{ui}, V_{ci}}) indicates that the value of v_{di} is data dependent on V_{ui} and V_{ci}, or rather that V_{ui} and V_{ci} have influences on the value of v_{di}. {V_u-V_{ui}} is called v_{di}'s independent use variable data dependence set, {V_c-V_{ci}} is called v_{di}'s independent condition variable data dependence set, { V_u-V_{ui} } ∪ { V_c-V_{ci} } as v_{di}'s independent data dependence set. This article will be deleted with the set of variables unrelated to v_{di}, in order to reduce redundancy.

Definition 2: Intra-Transition Data Dependence (IaTDD)[8]

Data dependence of intra-transition is the data dependence set composed of definition variable and the set of use variable set and condition variable set:

IaTDD T: {(v_{d1}, {V_{u1}, V_{c1}}), (v_{d2}, {V_{u2}, V_{c2}}),…, (v_{di}, {V_{ui}, V_{ci}}), …}

Where v_{di} is a definition variable in action or an input variable in event, $v_{di} \in V_d$. v_{d1}, v_{d2}, …, v_{di}, … constitute universal set of definition variables in action or event. v_{d1}, v_{d2}, …, v_{di}, …are not equal to each other. $v_{di} \neq v_{dj}$, $i \neq j$. V_{ui} is set of use variables influencing the value of v_{di} in action, which can be Null. V_{ci} is set of condition variables influencing the value of v_{di} in condition, which can be Null. $V_d \subset V_T$, $V_u \subset V_T$, $V_c \subset V_T$, V_T is the variable set in transition. $d \in I$, $u \in I$,$c \in I$, $i \in I$. IaTDD indicate all data dependence between variables in a transition.

Definition variables include the input variables in the event and the input variables, definition variables and output variables in action. The specific dependence is as follows:

1. If a variable is the input variable in the event of transition T, its data depends on the empty set. The complete set of condition variables in the condition and use variables in the action are the irrelevant set of the data dependent variable. For example, *EventName*(v_{in1},v_{in2},...) , then IaTDD(T, v_{ini}): (v_{ini}, { }). $i \in I$, I for integers. At this point, $V_u \cup V_c$ has nothing to do with the dependent variable set of the variable v_{ini}, so $V_u \cup V_c$ is removed from the dependent variable set v_{ini} to reduce redundancy. Event is like the definition of a function in computer program language; input variables the formal parameters of the function. Condition and action are like the body of the function. But it's likely that if the condition is true then the event and the action will be executed. In this case, the input variable data depends on the condition variable. This article focuses on the former situation. For example, the variable pin in the event *card(pin)* data dependence is described as IaTDD(T, *pin*): (*pin*, { }). $V_u \cup V_c$ in the condition and action sequences has nothing to do with the variable pin for the dependent variable set.

2. If a variable is the input variable in the action, its data depends on the empty set or a set of condition variables. The complete set of use variables is irrelevant dependent variable set. If the condition variable does not exist in T, its data depends on the empty set. Otherwise, if the condition variable exists, it depends on the condition variable. In both cases the complete sets of use variables are irrelevant dependent variable set. For example, *Input*(v_{in1},v_{in2},...) , then IaTDD(T, v_{ini}): (v_{ini},{ V_c }), $i \in I$ I for integers. At this point, V_u has nothing to do with the variable v_{ini} for the dependent variable set, so V_u is removed from the dependent variable set v_{ini} to reduce redundancy. For example, if the condition is empty, the variable *p* in the input statement *Input(p)* data dependence is described as IaTDD(T, *p*): (*p*, { }). If the condition is not empty and *attempts* <= *3*, the variable *p* in the input statement *Input(p)* data dependence is described as IaTDD(T, *p*): (*p*, {*attempts*}). In both cases V_u has nothing to do with *p*.

3. If a variable is the definition variable in the action, its data depends on the use variable set or set of use variables and condition variables. The set, that is complete set of use variables minus dependent use variables, is irrelevant dependent variable set. If the condition variable does not exist in T, its data depends on the use variable set. Otherwise, if the condition variable exists, it depends on the relevant use variable and condition variable. For example, $v_{d=}v_{u1}+v_{u2}+...$, v_{u1}, $v_{u2} \in V_{ui}$, then IaTDD(T, v_d): (v_d, {V_{ui}, V_c}). At this point, V_u-V_{ui} has nothing to do with the variable v_d for the dependent variable set, so V_u-V_{ui} is removed from the dependent variable set v_d to reduce redundancy. For example, if the condition is empty, the variable *attempts* in the assignment statement *attempts = attempts+1* data dependence is described as IaTDD(T, *attempts*): (*attempts*, { *attempts* }). If the condition is not empty and *(p != pin) and (attempts < 3)*, the variable attempts in the assignment statement *attempts = attempts+1* data dependence is described as IaTDD(T, *attempts*):(*attempts*, { *attempts*, *p*, *pin*}). In both cases V_u-{*attempts*} has nothing to do with attempts.

4. If a variable is the output variable in the action, its data depends on the output variable or condition variables. The set, that is complete set of use variables minus dependent use variables, is irrelevant dependent variable set. If the condition variable does not exist in T, its data depends on the output variable. Otherwise, if the condition variable exists, it depends on the output variable and condition variable. For example, *Output*(v_{out}), then IaTDD(T, v_{out}): (v_{out}, { v_{out} ,V_c }). At this point, V_u-

© 2012 ACADEMY PUBLISHER

V_{out} has nothing to do with the variable v_{out} for the dependent variable set, so V_u-V_{out} is removed from the dependent variable set v_{out} to reduce redundancy. For example, if the condition is empty, the variable p in the output statement *Output(p)* data dependence is described as IaTDD(T, p): (p, {p}). If the condition is not empty and *attempts*==*3*, the variable p in the output statement *Output(p)* data dependence is described as IaTDD(T, p): (p, {p, *attempts*}). In both cases V_u-{p} has nothing to do with p.

In dealing with condition variables, common practice is to consider condition variables as use variables. In order to facilitate follow-up studies and lay a good foundation for dynamic and conditional slicing, our research separate the condition variable from the set of use variables.

III. Different Intra-Transition Data Dependent Methods

For IaTDD and traditional methods (this refers to the K and M method, hereinafter referred to as K&M method) data dependence is as follows:

K&M T: {(v_{d1}, {V_u, V_c}), (v_{d2}, {V_u, V_c}),…, (v_{di}, {V_u, V_c}), …}

IaTDD T: {(v_{d1}, {V_{u1}, V_{c1}}), (v_{d2}, {V_{u2}, V_{c2}}),…, (v_{di}, {V_{ui}, V_{ci}}), …}

Which $\sum_{i=1,...,n} \cup v_{di} = V_d$, $v_{di} \in V_d$, V_d is the definition variable set of the transition T. v_{di} is a definition variable. V_u is the use variable set. V_c is the condition variable set. In K&M method, definition variable v_{di} is dependent on the complete set of use variables V_u and the complete set of condition variables, described as (v_{di}, {V_u, V_c}). In the IaTDD method, definition variable v_{di} is dependent on the use variable set V_{ui} and condition variable set V_{ci} that influence the changes of $v_{di,}$ in which v_{di} can be empty, described as (v_{di}, {V_{ui}, V_{ci}}) $V_{ui} \subset V_u$, $V_{ci} \subset V_c$. Thus, for both methods, the use variable set and condition variable set that are dependent on the same definition variable in IaTDD is the a subset of K&M method. That is, both methods have the same number of intra-transition definition variable, but K&M method can get more dependence than IaTDD, and in fact these variables did not affect definition variables, which resulted in redundancy.

$\forall v_{di} \in V_d, \exists v_{di\,(Method="K\&M")} = v_{di\,(Method="IaTDD")}$, $\therefore V_{d\,(Method="K\&M")} = V_{d\,(Method="IaTDD")}$.

$\forall$ (v_{di}, {V_u, V_c})$\in$K&M T, (v_{di}, {V_{ui}, V_{ci}})$\in$IaTDD T, V_{ui} is v_{di} related to the use variable set, which affect v_{di} or be affected by v_{di}, not including the use variables and condition variables irrelevant with the $v_{di,}$ $\therefore V_{ui} \subset V_u$, $V_u = V_{ui} + (V_u - V_{ui})$, $V_{ui} \cap (V_u - V_{ui}) = \Phi$. Similarly, $V_{ci} \subset V_c$, $V_c = V_{ci} + (V_c - V_{ci})$, $V_{ci} \cap (V_c - V_{ci}) = \Phi$.

Based on the above two conditions can be drawn: IaTDD T $\subset$ K&M T.

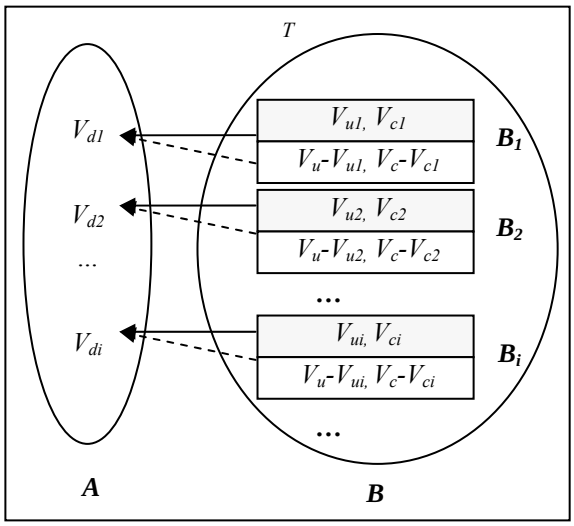


Figure 1 Data Dependence of Variables in Transition T Derived by K&M Method

The relationship between definition variables and use, condition variables is shown in Figure 1. Set A= V_d={ v_{d1}, v_{d2}, ..., v_{di},...}, that is the complete set of definition variables, A can be empty. Set B_1= B_2=...B_i=...B= $V_u \cup V_c$, B_i indicates the set of use variables and condition variables. A is dependent on B. $\forall$ (v_{di}, {V_u, V_c})$\in$K&M T, each definition variable is dependent on the complete set of use variables and condition variable, but it is not the case. Actually, v_{di} *is dependent on* V_u and V_c, omitting use variable and condition variable irrelevant to v_{di}. Thus, a new dependence is made, described as (v_{di}, {V_{ui}, V_{ci}}), $V_{ui} \subset V_u$, $V_{ci} \subset V_c$. In Figure 1 set B_1 deletes dependent variable set irrelevant to v_{d1}, that is, B_1 deletes { V_u -V_{u1} , V_c -V_{c1} }, B_2 deletes { V_u -V_{u2} , V_c –V_{c2} }, …, B_i deletes { V_u -V_{ui} , V_c –V_{ci} }, etc. In other words, B_1 equals {V_{u1}, V_{c1}}, B_2 equals {V_{u2}, V_{c2}}, …, B_i equals {V_{ui}, V_{ci}}, etc. Re-construct the data dependence and get Figure 2 IaTDD T.

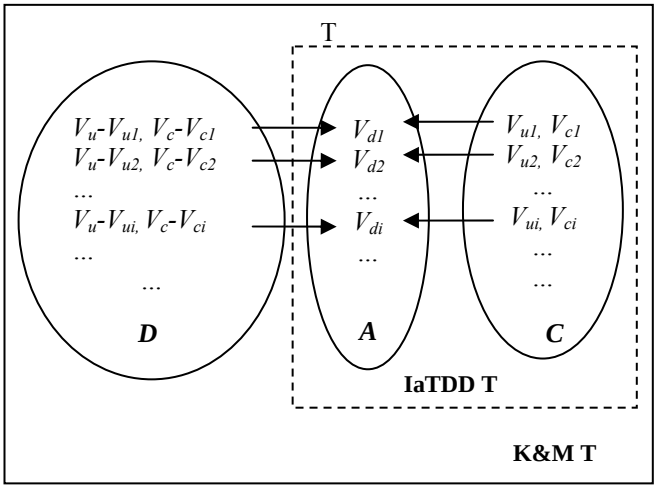


Figure. 2 Comparison of Data Dependence between IaTDD and K&M Method

© 2012 ACADEMY PUBLISHER

$\forall$ (v_{di}, {V_u, V_c})$\in$K&M T, delete (v_{di}, {V_u -V_{ui}, V_c–V_{ci} }), then get (v_{di}, {V_{ui}, V_{ci}})$\in$IaTDD T, so IaTDD T$\subset$K&M T. Figure 2 shows that set *A* is dependent on set *C*, that is, each definition variable of set *A* depends on the definition variables associated with the use variables and condition variables of set *C*. Each definition variable has nothing to do with the use variables and condition variables of set *D*. If there's any relations, redundancy will result.

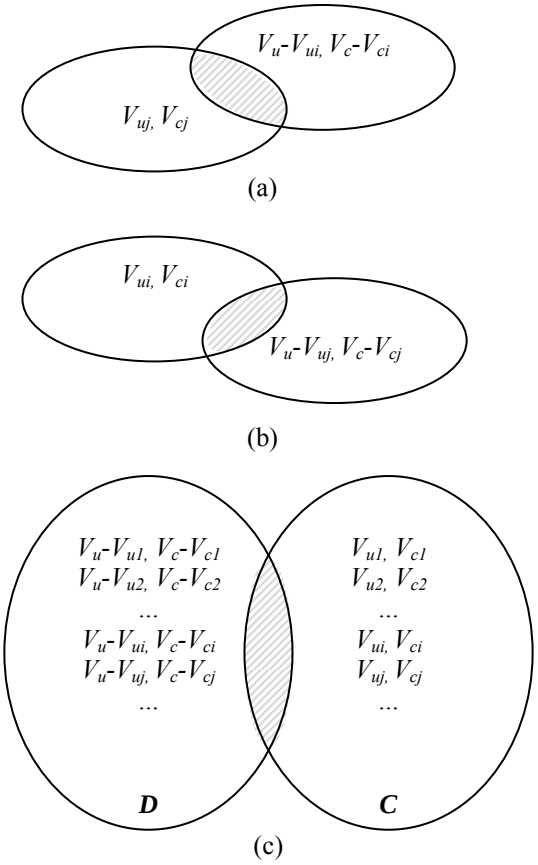


Figure 3 Variable Set Dependent by Definition Variable of IaTDD and K&M Method

$\forall$ (v_{di}, {V_u, V_c}), (v_{dj}, {V_u, V_c})$\in$K&M T, (v_{di}, {V_{ui}, V_{ci}}),(v_{dj}, {V_{uj}, V_{cj}})$\in$IaTDD T, we get V_u =V_{ui} + (V_u -V_{ui}), V_{ui} $\cap$(V_u -V_{ui})=Φ, V_u =V_{uj} +(V_u -V_{uj}), V_{uj} $\cap$(V_u -V_{uj})=Φ. But V_{ui} and (V_u -V_{uj}), V_{uj} and (V_u -V_{ui}) may intersect, that is, repeated elements may exist. Let's mainly see the use variables and condition variables that are dependent. $\because \exists \forall$ $V_{ui}\cap(V_u - V_{uj}) \neq \Phi$, $V_{uj}\cap(V_u - V_{ui}) \neq \Phi$, are shown in (a) and (b) of Figure 3, the illustrated variables dependent by definition variable include use variables and condition variables. $\therefore \exists$ ($V_{ui} \cup V_{uj}$)$\cap$((V_u -V_{ui}) $\cup$(V_u -V_{uj})) $\neq\Phi$. Therefore $\exists$ ($V_{ci} \cup V_{cj}$)$\cap$((V_c –V_{ci}) $\cup$(V_c –V_{cj})) $\neq\Phi$, as (c) of Figure 3 shows, $\exists$ $C\cap D\neq\Phi$, C={ V_{u1}, V_{c1} , V_{u2} , V_{c2} ,..., V_{ui} , V_{ci} , V_{uj} , V_{cj} ,... }, D={ V_u-V_{u1}, V_c-V_{c1} , V_u-V_{u2} , V_c-V_{c2} ,..., V_u-V_{ui} , V_c-V_{ci} , V_u-V_{uj} , V_c-V_{cj} ,... }.

IV EXPERIMENT AND ANALYSIS

This paper compares the EFSM models commonly used in various documents to analyze the impact of intra-transition data dependence.

A. Experimental Model

Specific experimental model data is shown in Table 1, in which #S is the number of states, #T is the number of transition.

TABLE 1
EXPERIMENTAL MODELS

EFSM Model	#S	#T
ATM[6]	9	23
Cashier[9]	12	21
Cruise Control[10]	5	17
Fuel Pump[10]	13	25
PrintToken[9]	11	89
Door Control[11]	6	12
Vending Machine[9]	7	28
INRES protocol[12]	8	18
TCP[13]	12	57

B. Experimental Data

This section is about experiments on the 9 EFSM models in Table 1. Through the IaTDD method and K&M method, experiments will be done to get the number of data dependence between variables, the number of redundant variables, and comparison of the number of definition variables, data dependence relations between numbers, and number of redundancies.

Comparing the results of the experiments that apply the two methods, as is shown in Table 2, "#K&M method" indicates the number of data dependence that is acquired without using IaTDD method. "#IaTDD method" indicates the number of data dependence that is acquired with IaTDD method.

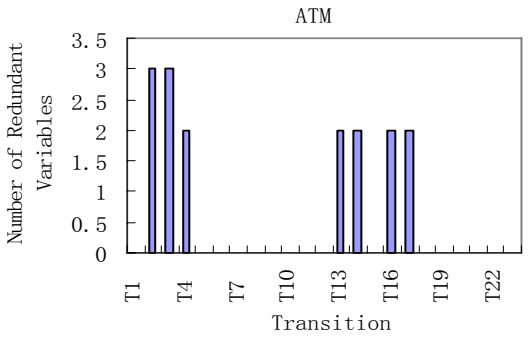
TABLE 2
NUMBER OF DATA DEPENDENCE BETWEEN INTRA-TRANSITION VARIABLES DERIVED BY TWO METHODS

EFSM Model	#K&M method	#IaTDD
ATM	28	28
Cashier	30	30
Cruise Control	50	50
Fuel Pump	44	44
PrintToken	49	49
Door Control	6	6
Vending Machine	30	30
INRES Protocol	14	14
TCP	135	135

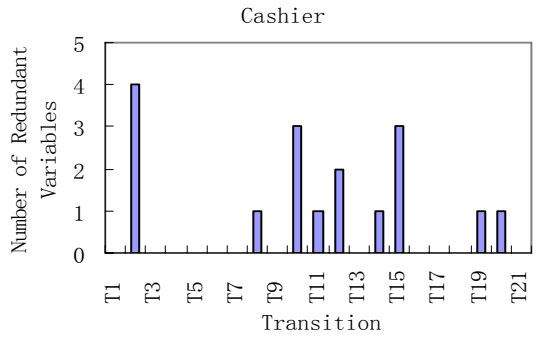
The results of Table 2 show that the two methods get the same intra-transition data dependence between variables, but IaTDD method does not produce redundant variables, while K&M method producing a lot of

© 2012 ACADEMY PUBLISHER

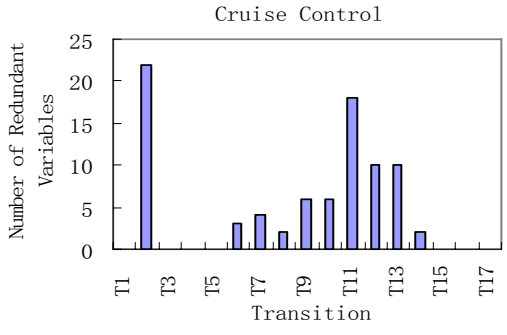
redundant variables. The numbers of redundant variables of the specific nine models are shown in Figure 4. It describes the number of redundant variables contained in each transition of the models. The horizontal axis indicates the specific transition, and the vertical axis indicates the number of redundant variables contained in transition. With K&M method, Door Control model does not produce redundant variables, because the Door Control model includes at most one definition variable, therefore the data dependence is simple. But in reality, the situation is not always so rational. The other eight models, as Figure 4 shows, produce redundant variables to a different extent. Redundancy caused by K&M method is shown in Table 3.



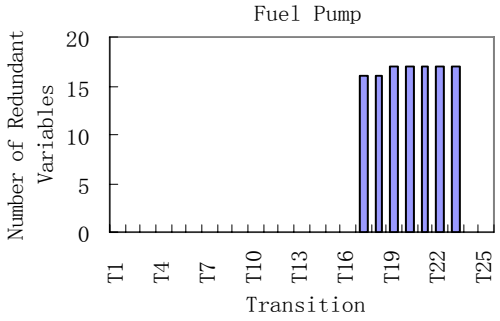
(a)



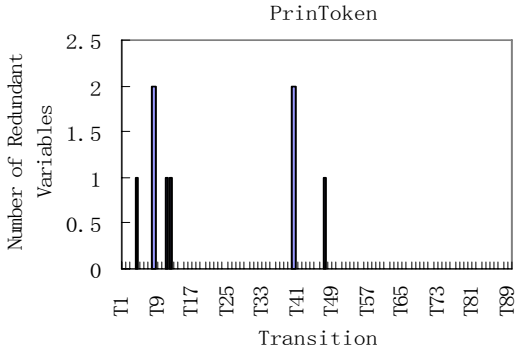
(b)



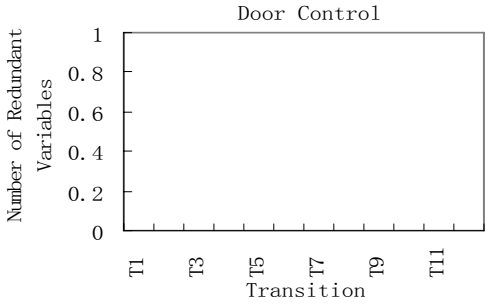
(c)



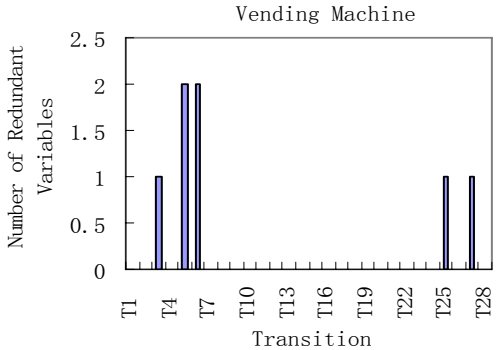
(d)



(e)

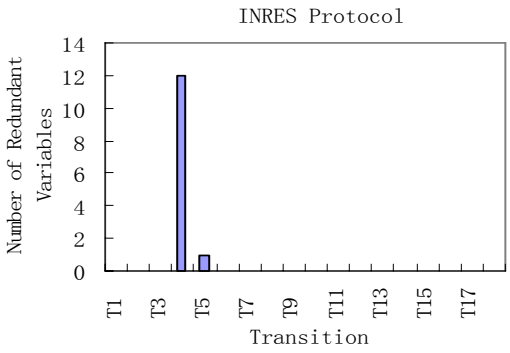


(f)

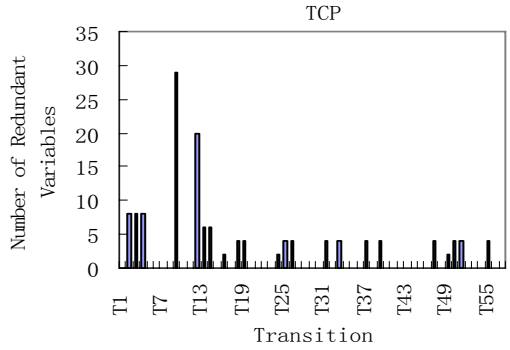


(g)

© 2012 ACADEMY PUBLISHER



(h)



(i)

Figure 4 Number of Redundant Variables Derived by K&M Method

TABLE 3
REDUNDANT VARIABLES DERIVED BY K&M METHOD

EFSM Model	Number of Redundant Variables	Number of data dependence among redundant variables
ATM	16	7
Cashier	17	10
Cruise Control	83	31
Fuel Pump	117	28
PrintToken	8	8
Door Control	0	0
Vending Machine	7	4
INRES protocol	13	7
TCP	139	78

290 samples of transition in the nine models are collected to undergo comparative experiments on the number of definition variables contained in each transition and the number of data dependence among redundant variables, as is shown in Figure 5.

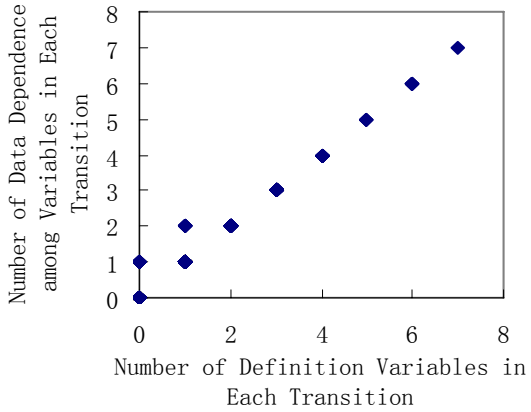


Figure 5 Comparison of Number of Definition Variables and Number of Data Dependence among Variables in Each Transition

It can be seen from Figure 5 that the more the number of variables contained in each transition, the more the data dependence among variables; the more complex the transition structure. Figure 6 and Figure 7 show the comparative experiments on the relationship between number of variables contained in each transition and the number of redundant variables, by using the method of K&M.

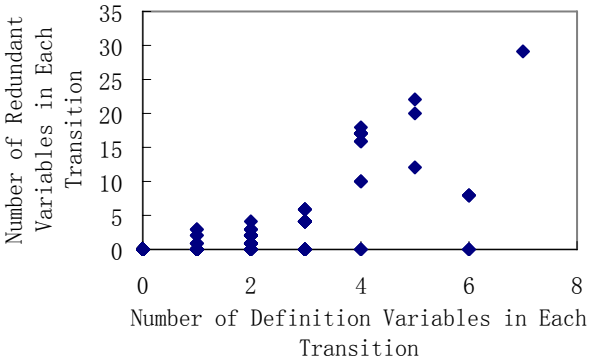


Figure 6 Scatter Diagram of Number of Definition Variables and Redundant Variables in Each Transition of Traditional Method

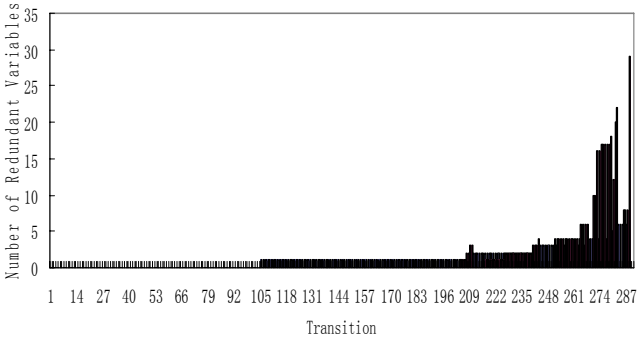


Figure 7 Histogram of Number of Definition Variables and Redundant Variables in Each Transition of K&M Method

Figure 6 and 7 show that the more the number of definition variables contained in each transition, the more

© 2012 ACADEMY PUBLISHER

redundant variables. In Figure 7, 105 transitions have 0 definition variable, 105 have 1 definition variable, and 80 have two or more definition variables. Figure 7 shows that after sorting out the experiment data, since the 210th transition, the redundant variables increase significantly.

C. Results

The two methods produce the same number of intra-transition variable data dependence, but K&M method produces more redundant variables. These redundant variables lead to errors in the next phase and new redundancies.

However, if the intra-transition definition variable is 0 or 1 or many, or the average number of data dependence is very low in each transition, for instance, within each intra-transition of the Door Control model, the average number of data dependence is 0.416667, then the use of K&M method can help to get all the data dependence, in other words, there is no need to use IaTDD method. If there're relatively few definition variables, use variables and condition variables in EFSM model, each action sequence of transition is relatively simple. Therefore, the corresponding relationship between the variables is relatively simple. We can consider not use IaTDD method on condition that the focus of a research is not data, and redundancy as well as a small amount of errors can be tolerated. But when the intra-transition definition variables reaches 2 or more, K&M method produces more and more redundant variables, then the use of the proposed method in this paper is more appropriate. It is more simplifying and is a necessary method. Also, IaTDD method can be applied during pre-EFSM stage. When an EFSM input file finishes scanning, the intra-transition data dependence is created. Even if one-pass scanning is performed, the time complexity is only decided by the number of statements. It is not time-consuming, and can be completed by positive traverse of all the statements in intra-transition. Therefore, it's a feasible method.

V. SUMMARIES

Due to the problem that direct application of existing data dependence methods of EFSM model can cause redundant variables, this paper compares the intra-transition data dependence method and the K&M methods, proves that the new method can reduce further redundancy, and is a necessary and feasible method, providing theoretical basis for follow-up study.

With the application of model slicing in different sections, the qualitative description of intra-transition data dependence is our next subject of research.

ACKNOWLEDGMENT

We would like to acknowledge the generous financial support of Fundamental Research Funds for the Central Universities (ZZ1136).

REFERENCES

[1] Weiser M. Program slices: formal, psychological, and practical investigations of an automatic program abstraction method [D]. Ann Arbor: University of Michigan, 1979

[2] Mats P E, Heimdahl, Michael W, Whalen. Reduction and slicing of hierarchical state machines [A]. *In Proc. Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering* [C]. Springer Verlag, 1997.

[3] Mats P E, Heimdahl, Je_rey M, Thompson, Michael W, Whalen. On the e_ectiveness of slicing hierarchical state machines: A case study [A/J]. *In EUROMICRO '98: Proceedings of the 24th Conference on EUROMICRO*[C]. l*EEE Computer Society*. USA, 1998, 10435

[4] Savage P, Walters S, Stephenson M. Automated Test Methodology for Operational Flight Programs [A]. *Proceedings of IEEE Aerospace Conference* [C]. 1997, 4: 293-305

[5] Dssouli R, Saleh K, Aboulhamid E, En-Nouaary A, Bourhfir C. Test Development For Communication Protocols: Towards Automation [J]. *Computer Networks*, 1999, 31: 1835–1872

[6] Korel B, Singh I, Tahat L, Vaysburg B. Slicing of state based models[A]. *In IEEE International Conference on Software Maintenance (ICSM'03)*[C]. USA: IEEE Computer Society Press Sept. 2003, 34–43

[7] Miao Li, Zhang Dafang, Computing Backward Slice of EFSMs[J]. *Journal of Software*, China, 2004,15:169-178

[8] Shenghui Shi, Qunxiong Zhu, Wenxing Xu. Intra-Transition and Inter-Transition Data Dependence for EFSM[C]. 2011 *International Conference on Computer Application and System Modeling (ICCASM 2011)*

[9] Korell B. *Private communication*, 2009

[10] Korel B, Koutsogiannakis G, Tahat L H. Model-based test prioritization heuristic methods and their evaluation [A]. *In A-MOST '07: Proceedings of the 3rd international workshop on Advances in model-based testing*[C]. USA: ACM, 2007, 34–43

[11] Strobl F, Wisspeintner A. Specification of an elevator control system – an autofocus case study[R]. *Technical Report TUM-I9906*, Technische Universität München, 1999.

[12] Bourhfir C, Dssouli R, Aboulhamid E, Rico N. Automatic executable test case generation for extended finite state machine protocols [A]. *In IWTCS'97*[C]. 1997, 75–90

[13] Zaghal R Y, Khan J I. EFSM/SDL modeling of the original TCP standard (RFC793) and the congestion control mechanism of TCP Reno[R]. *Technical Report TR2005-07-22*, Internetworking and Media Communications Research Laboratories, Department of Computer Science, Kent State University, 2005.

Shenghui Shi 1974-, China, Ph.D, Lecturer in Beijing University of Chemical Technology. Area of Research: Slicing Technology, Compiler Technology, Fault Detection, Safety Analysis.
Email: shish@mail.buct.edu.cn

© 2012 ACADEMY PUBLISHER

Qunxiong Zhu 1960-, China, Ph.D, Professor, Dean of College of Information Science and Technology in Beijing University of Chemical Technology. Area of Research: Fault Detection, Artificial Intelligence, Data Mining, Decision-Making and Control Research Area.
Email: zhuqx@mail.buct.edu.cn

Zhiqiang Geng 1973-, China, Ph.D, Associate Professor in Beijing University of Chemical Technology. Area of Research: Artificial Intelligence, Control Research Area.
Email: gengzhiqiang@mail.buct.edu.cn

Wenxing Xu 1982-, China, Ph.D candidate in Beijing University of Chemical Technology. Area of Research: Intelligent Computing.
Email: estellaxu@163.com

© 2012 ACADEMY PUBLISHER