行政院國家科學委員會專題研究計畫 期中進度報告

程式執行時間分析的新理論(2/3)
期中進度報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 95-2221-E-002-072-
執 行 期 間 ：95 年 08 月 01 日至 96 年 07 月 31 日
執 行 單 位 ：國立臺灣大學電機工程學系暨研究所

計 畫 主 持 人 ：王凡

處 理 方 式 ：本計畫可公開查詢

中 華 民 國 96 年 07 月 31 日

行政院國家科學委員會補助專題研究計畫 □ 成 果 報 告 □ 期中進度報告

程式執行時間分析的新理論(2/3)

計畫類別： 個別型計畫
計畫編號：NSC 95－2221－E －002－072－
執行期間：2006 年 08 月 01 日至 2007 年 07 月 31 日

計畫主持人：王凡
共同主持人：
計畫參與人員：

成果報告類型(依經費核定清單規定繳交)：精簡報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：

Minimum Model Repair with an Automata-Theoretical Approach

I. Introduction

The construction of large complex software with quality assurance is becoming more important than ever. In general, quality assurance is achieved with verification techniques, i.e., checking if the behavior of the design meets the specification. Up to now, for program verification, various techniques have been developed, including testing [Myers04] and model checking [CE81]. Once a bug is reported in the verification process, locating and repairing the bug still rely heavily on human intervention which is costly, time-consuming, and error-prone. In most projects, the cost for debugging and repairing programs still cannot be managed well. Thus, without taking repair cost into consideration, research work in program repair is not likely to be useful in practice. It is the goal of this work to develop a formal theory for the automatic construction of minimum-cost repairs of computer systems.

In our framework, we have a *model automaton* A_M[1] [Land69, PR89, VW86] and a *specification* ψ[2]. A *repair* is defined as a sequence of *edit operations* to A_M to satisfy ψ according to a repair *criterion*. We propose several repair criteria, including *language containment*, *language equivalence*, and *bisimulation*. We consider four types of *edit operations*: addition of a state, deletion of a state, addition of a transition, and deletion of a transition. Then we formally define a *repair* as an edit sequence that transforms A_M to an automaton that satisfies ψ with respect to a criterion. Then the length of an edit sequence naturally defines the *cost* of the corresponding repair. Thus our *minimum repair problem* is to construct the minimum-length repair of A_M for ψ and a criterion.

As for the computation issues of the minimum repair problem, we have adapted Bunke's theorem of minimum edit-sequence [Bun97] for an upper-bound on the minimum repair cost for a given problem instance. The upper-bound is irrelevant to the repair criteria. We then present an algorithm based on *maximal common subgraph (MCS)* construction to calculate this upper-bound. Then we discuss the complexities of the minimum repair problem for the criteria that we have proposed. Then we also present a procedure that uses a breadth-first style to explore the space of repairs to search for the minimum repair. Then we present an algorithm for repair for bisimulation based on heuristics of MCS construction and bisimulation construction. We have implemented the algorithms and report their performance against several benchmarks.

[1] The model automaton can be constructed as an abstraction of, for example, a C program.
[2] Ψ can be another automaton or a temporal logic formula [Pnu71].

II. Main Theoretical Results Related to Model Repair

In the following are some results we had proven.

Lemma 1. *The minimum repair problem for language equivalence, and LTL equivalence are all PSPACE-hard and solvable in EXPTIME.*

Lemma 2. *The minimum repair problem for criteria LTL containment is PSPACE-complete.*

Lemma 3. *The minimum repair problem for bisimulation $\equiv B$ is solvable in NP.*

Lemma 4. *Suppose we are given a model automaton A_M and a specification automaton A_ψ. If G_{MCS} is the MCS of A_M and A_ψ, then the minimum repair costs of A_M for A_ψ and all criteria are no greater than $|A_M|+|A_\psi|-2|G_{MCS}|$.*

III. A BFS-based Enumeration Algorithm to Find Minimum Repair

Based on the results in the last three sections, we have designed an exploration algorithm that searches through the space of edit sequences for a minimum-cost repair for a given criterion. The algorithm is developed with the availability assumption of a checker for relation Ξ between the model automata and the specifications. The algorithm is as follows.

Repair_by_Repair_Space_Exploration(A_M, ψ, Ξ)
/* A_M is an automata. Ξ is a repair criterion. */ {
    Let $\Phi := \{A_M\}$. $i := 0$.
    Repeat forever {
        If there is an $A \in \Phi$ such that $A \; \Xi \; \psi$,
            return "Minimum repair cost is i with result A."
        $\Phi' := \varnothing$. $i := i + 1$.
        For each $A \in \Phi$ {
            Create a state q not in A.
            For each $L \subseteq \{ini, acc\}$, $\Phi' := \Phi' \cup \{state\ add(A, q, L)\}$
            For each state q in A without incoming and outgoing transitions,
                $\Phi' := \Phi' \cup \{state\ del(A, q)\}$
            For each q, q' in A and $a \in \Sigma$, $\Phi' := \Phi' \cup \{xtion\ add(A, q, a, q')\}$
            For each $(q, a, q') \in E$, $\Phi' := \Phi' \cup \{xtion\ del(A, q, a, q')\}$
        }
    $\Phi := \Phi \cup \Phi'$.
    }
}

IV. An MCS-based algorithm for repair for bisimulation

The algorithm in section III may suffer from low performance since the space of repairs could be huge even within the upper-bound. In this section, we present a repair algorithm for bisimulation based on MCS construction and bisimulation construction. Specifically, we construct a *bisimulation* between a graph and itself and partition the states into equivalence classes. The algorithm may not yield the minimum-cost repair. But it may efficiently find a repair that is less expensive than the upper-bound in lemma 4.

The idea of our new algorithm is as follows. Let G_{MCS} be the MCS between A_M and A_ψ. Let $A_M - G_{\text{MCS}}$ denote the difference between A_M and G_{MCS}. Similarly $A_\psi - G_{\text{MCS}}$ is the difference between A_ψ and G_{MCS}. The repair cost upper-bound in lemma 4 is derived from the edit sequence from A_M to A_ψ. Conceptually, this upper-bound suggests a repair that discards $A_M - G_{\text{MCS}}$ and adds $A_\psi - G_{\text{MCS}}$. It could be the case that neither the whole $A_M - G_{\text{MCS}}$ need be discarded nor the whole $A_\psi - G_{\text{MCS}}$ need be added. This intuition is realized with the following two steps.

- **Step 1: minimizing $A_\psi - G_{\text{MCS}}$**
- **Step 2: maximizing the reusability in the difference of A_M**

V. Implementation and Experiments

Our experimental tool ModelRepair ver.0.1 realizes part of our idea in finding a minimum repair. The tool is available at http://val.ee.ntu.edu.tw/ModelRepair. It finds minimum repairs for many repair criteria. The language containment checking is achieved by a specialized version of GOAL [TCWC07]. To visualize the model, we offer interfaces to convert our graph representations into the GOAL format. The users can thus conveniently see the difference between the repaired model and the original model. A snapshot of the tool is shown in Fig. 1.

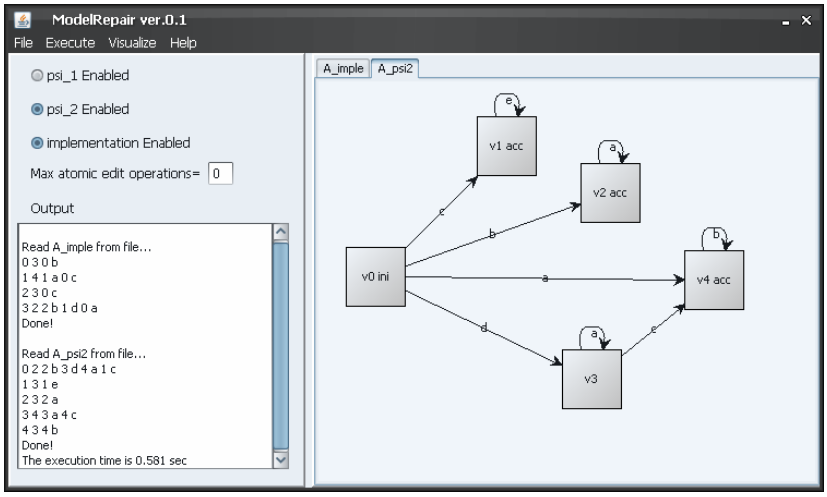

Fig.1. Snapshot of Model Repair ver.0.1

We have applied our tool to a few examples. Table 1 summarizes the result of the experiment. As can be seen, for benchmarks 1 to 5, the MCS-based heuristic algorithm runs much faster than the repair-space exploration algorithm.

B	A_M			A_ψ			UB	exploration for $\equiv_\mathcal{L}$					MCS-based heuristics				
	$\|A_M\|$	$\|Q\|$	$\|E\|$	$\|A_\psi\|$	$\|Q\|$	$\|E\|$		$\|A_M\sigma\|$	$\|Q\|$	$\|E\|$	$\|\sigma\|$	time	$\|A_M\sigma\|$	$\|Q\|$	$\|E\|$	$\|\sigma\|$	time
1	11	4	7	14	5	9	5	10	4	6	1	1.19s	14	5	9	5	0.37s
2	9	4	5	15	6	9	10	9	4	5	2	8.16s	9	4	5	4	0.36s
3	4	2	2	7	3	4	3	7	3	4	3	168s	7	3	4	3	0.15s
4	23	9	14	29	11	18	22	$N/A, > 30min$					23	9	14	16	521s
5	39	16	23	15	6	9	38	$N/A, > 30min$					15	6	9	38	72.6s
6	12	4	8	15	5	10	5	No repair needed, 1.07s									

B: benchmarks; UB: minimum cost upper-bound predicted with lemma 4;
σ: the corresponding repair; $|Q|$: # nodes; $|E|$: # arcs; s: seconds;

Table 1. Performance of ModelRepair ver.0.1

VI. Conclusion

Our work focuses on the automatic generation of minimum-cost repair that could be helpful in controlling the budget for program debugging and preserving the original design intention. We feel that our work could be used as a general foundation for the future research in this direction. For example, we may want to develop new heuristics to efficiently find a good enough repair for real cases in practice. We may also want to investigate new repair criteria and the corresponding problem complexity. We may also consider extending the problem to models like timed automata, hierarchical automata, etc.

References

[Bun97]. H. Bunke. On a Relation between Graph Edit Distance and Maximum Common Subgraph. Pattern Recognition Letters. 19(1997) pp. 255-259.

[CE81]. E. Clarke, E.A. Emerson. Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic, Proceedings of Workshop on Logic of Programs, Lecture Notes in Computer Science 131, Springer-Verlag, 1981.

[Land69]. L.H. Landweber. Decision problems for ω-automata. Mathematical Systems Theory, 3:376-384, 1969.

[Myers04]. G. J. Myers, C. Sandler, T. Badgett, and T. M. Thomas. The Art of Software Testing. Wiley, 2004.

[Pnu71]. A. Pnueli. A Temporal Logic of Concurrent Programs. Theoretical Computer Science 13: 45-60.

[PR89]. A. Pnueli, R. Rosner. On the Synthesis of a Reactive Module. POPL'89, ACM Press, pp. 179-190.