Evaluation and Implementation of the Dynamic Slicer Expressing the Execution History by XML

Yosuke Takeda (110073)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 2003

Keywords: Dynamic Analysis, Data Schema, XML, Dynamic Slicing.

1 Background

Dynamic slicing is a useful technology in engineering and much research have been done. Dynamic slices are useful in debugging, testing, maintenance, and program understanding. Korel and Laski proposed the first dynamic slicing. After that, Agrawal proposed some approaches of dynamic slicing. Besides such research of a technique, there is much research for mitigation of the size of an execution history. In addition, there is the technique of marking to a program tree for improvement of accuracy. In order to use dynamic slicing as a practical use target more, there is also research which supports a function call and pointer analysis. However, these all are not research works on development efficiency.

Research of dynamic slicing has problems, such as the size of an execution history becoming large and being hard to treat, execution environment is required, and the difficulty of data expression. Therefore, although research is prosperous, as far as we know, there is only a dynamic slicer for a small language. Moreover, if implemented for every technique, before evaluating, which results in high development cost. Furthermore, accuracy also becomes worse. Then, the data schema is introduced into the dynamic slicer. Extraction and analysis of an execution history are divided.

Thereby, we can concentrate on development of the essential of the technique. Therefore, the development cost reduction of each dynamic slicer is important.

Generally, a dynamic slicer needs a data schema with the fine-grained information. There are ACML, I-model of Sapid, and JavaML as an example of modeling of program information. JavaML is coarse-grained. Therefore, information required for slicing is insufficient. ACML and I-model of Sapid are fine-grained. But, it does not have dynamic information. Therefore, a data schema with dynamic information is required.

2 Purpose

This research defines the data schema which uses XML for the execution history. And development efficiency is improved. Separation of extraction and analysis can be performed by defining a data schema using XML. Thereby, development of an extraction part and an analysis part can be performed simultaneously. By test case creation of handmade business, even if there is no execution environment, evaluation of a dynamic slicer can be performed. An improvement of the development efficiency of a dynamic slicer can greatly contribute to promotion of research of dynamic slicing.

Our goal is to realize a dynamic slicer corresponding to a full set of ANSI C. As the first phase, the dynamic slicer using the subset of ANSI C is realized.
The following three steps are required for realization.

- The design of the suitable data schema for execution histories.
- Experimentally implementing a dynamic slicer based on the data schema
- The check of the validity of a data schema, the convenience of XML related technology, and the usefulness of the common format by XML.

3 Proposal and Implementation of Dynamic ANSI C Markup Language(Dynamic ACML)

The data schema for dynamic slicers is proposed in this research. Below, the conditions of a data schema are shown.

- Compact design
- Structure which is easy to treat to extraction side and analysis side
- Simple structure which is easy to understand

The design factors for fulfilling these conditions are choice alternative of required information(Syntax information, variable name, variable value, address value, etc), the setup of the degree of abstraction, and the setup of grained.

First, Dynamic ACML was defined based on the technique of Korel and Laski. This has dynamic information, such as a variable value and a boolean of a control instruction, at the execution time. Furthermore, it has the static information such as the control range of a line number, a variable name, typedef, definition or use of variable, and a control command. Next, Dynamic ACML was extended based on the technique of Agrawal. This has the address and size of a variable. Therefore, a dependency is extracted not in a variable name but in the address value and size of a variable. Furthermore, a structure object and array are added to a variable and it enabled to distinguish them clearly. This is for holding the information on offset of a structure object and array. The accuracy of a slice goes up using this information. For example, it turns out that it is not the whole `a` but dependence to `a[1]` if `i` of `a[i]` is 1 when array A exists.

4 Implementation of Dependence Analysis Tool

In this research, we implemented two kinds of the technique of Korel and Laski, and the technique of Agrawal, based on two kinds of Dynamic ACML stated for the preceding clause. It is from the following reasons to have used these techniques.

- The former is the most fundamental technique. Therefore, the usefulness using a dynamic slicer and XML is checked.
- The latter is a more practical technique. Dynamic slicer including pointer analysis is implemented.

Our implementation was straight forwardly done. By one developer, each took about two weeks. Although restricted to our small-scale experiment, the development cost reduction of XML was checked. In our opinion, this is because the conditions of a definition of the data schema of the foregoing paragraph were realized well.

For example, there is a data dependency with the important element of dynamic slicing The definition of a variable and the information on use are indispensable to this analysis. When the definition of a variable and the information on use are described to a data schema, an extract side want to extract simply in order of evaluation on the abstract syntax tree. On the other hand, if an analysis side collects a definition and after use, it is easy to search. If it is going to collect a definition and use in consideration of the convenience by the analysis side, we will think that the increase in the development cost which an extraction side wears becomes large. Therefore, we expressed them in the form which gave the information on a definition and use to the attribute of a variable, and accompanied it in order of evaluation. For this reason, an analysis side must take into consideration the information on expression which is different for the same information. However, there is a method "getElementByTagName" which returns a set of child elements with the specified tag name in DOM. For this reason, the flexibility of expression of a data schema was absorbable. Therefore, it was able to implement without caring about the evaluation order of variables.

5 Discussions

The factors of development cost reduction are considered to be the following items.

- The design of a suitable data schema
  - The compact execution history was realizable with deletion of syntax information and the necessary minimum data check. Actual

size was settled about 8 times to the source code, about 20 times to the Comma Separated Value.

  - A trade-off is between an extraction side and an analysis side about the ease of treating of data. This difference was solved by giving flexibility to a data schema.
  - If structure of a data schema is complicated, it will be difficult narrow-minded both extraction side and analysis side. Time to understand structure is added to development cost. For example, the information of a certain execution time can be acquired as one tree structure. This element is a child element of the root. However, when there was a control dependence, it is considered as the child element of a control instruction which it depends on.

- Text file format as intermediate data.
  - Since it extracts as a file, the analysis and separation of an execution history can be performed and data extraction and dependence analysis can be implemented in parallel.
  - Since it is a text file, a test case can be created by the text editor etc.
  - Since it is text form, an execution history can be checked without using a special viewer.

- Execution history is expressed by XML.
  - DTD can define a data schema easily by using EBNF form. Moreover, validity verification can be easily performed using an XML parser.
  - Lexical analysis and syntactic analysis are also available.
  - DOM has much simple operation. Therefore, acquisition does not take much time. Moreover, if the above-mentioned method is used, the child element of a tree structure can be treated as a set. For this reason, it can be used also for set operation.

6 Conclusion and Future Works

We expressed the execution history using XML. DACML which is the data schema used for a dynamic slicer was proposed. And the technique of two kinds of dynamic slicers was implemented as implementation experiments. One developer realized each of them in about two weeks, respectively. Although restricted to our small experimental implementation, the usefulness of development cost reduction and XML was checked by dynamic ACML. Especially, we have a plan to study the following as future works:

- Introducing more pointer analysis
  In the present Dynamic ACML, neither the pointer to pointer nor the pointer returned by malloc is taken into consideration.

- Coping with a function call
  The function call has many problems, such as reference delivery and value delivery of an argument and access to a function pointer. These cannot be coped with by the present dynamic ACML. However, this is important for dynamic slicing.