Generator for Z Conceptual Model Using JFlex and BYACC/J Specification

Maria Ulfah S* & Zarina Shukur
Department of Information Technology, Faculty of Science and Technology,
State University of Islamic Sunan Kalijaga
Jl. Marsda Adisucipto, Yogyakarta 55281
ulfahtc96@yahoo.com
Department of Computer Science, Faculty of Information Science and Technology, National University of Malaysia
UKM Bangi 43000
zs@ftsm.ukm.my

Abstract

The Z notation is one of leading formal specification language based on standard mathematical notation, used for describing and modeling computer systems. The system, which is called ZC06, is composed by some phases, namely lexical analysis, syntax analysis, semantic analysis, and model conceptual-generation. ZC06 accepts Z specification and produces conceptual model whose format is based on format that had been proposed by researchers around 1990's. The lexical analyzer is developed by using JFlex, whilst the three other phases are developed by using BYACC/J. ZC06 is implemented by interfacing JFlex with BYACC/J. ZC06 has been tested by using two specifications. The first specification is a specification that specifies security room (SBK), which has been used by previous researcher. The output is evaluated by comparing ZC06's output with the one, which had been developed by that researcher. The second one is a specification that specifies a system, which records people's birthday (SHJ). This specification has not been used as an input in previous researches. Therefore, the output is evaluated by redeveloping manually its specification based on its conceptual model. The result shows that ZC06 produces conceptual model, which is equally the same as the previous one. Moreover, SHJ has been redeveloped successfully.

1. Introduction

Available tools, which use formal method, are less in number. Although there is a number of formal software available that can be used by users, sometimes their uses are not suitable with user requirement. Formal software, which generates and output conceptual model of a system, is also less in number. The conceptual model is a description of data concerns with the system. This model is mostly needed for understanding the system behavior so that it will be easier for the change and development of the system.

Therefore, to involve many people in using formal specification, it is necessary to supply varied specification tools, which meet user requirements. Developing a formal specification tool such as type checker, verifier, easier as conceptual model of specification has produced before.

A conceptual model describes application design from owner/ user point of view. This model does not depend on any implementation and thus, it is stable. The conceptual model supports present requirements and supports future changes without having significance structure changes [4].

Objectives of this research are:

- To change Z conceptual model that suggested by Abdullah [1] and Mohd. Zaki [8]. These changes exist in tables of conceptual model.
- To implement a generator for Z conceptual model using JFlex [5] and BYACC/J [3] specification

Researches concern for generating conceptual model has begun by previous researchers. One of them is Abdullah Mohd. Zin. Although his research has wider scope, ZFDSS still covers conceptual model generator. ZFDSS is a system used to develop formal software by using liberal approach. Liberal approach means that mathematics is used as needed. The system is developed in Unix.

Mohd. Zaki bin Mohd. Ghazali has developed a system for verifying Z specification, which also includes conceptual model generator. The system can identify user requirement and can check for error. This project changes ZFDSS that running in Unix to PC.

With conceptual model, it is hoped that the developing formal method software will be easier in the future. This is because with the availability of the conceptual model, the changes will not so difficult. In addition, if this conceptual model does not picture out system requirement precisely, specification can be repaired in early stage.

2. Z Conceptual Model

ZC06 consists of processes that involved in a compiler, which are lexical analysis, syntax analysis, semantic analysis and intermediate code generator [2]. This system is composed by three modules, which are lexical analyzer module, parser module and main module. Parser module has function as syntax analyzer, semantic analyzer, and conceptual model generator. The conceptual model acts as an intermediate code. The system is implemented by using lexical analyzer software (JFlex) and grammar parser (BYACC/J) which has code in Java program and run in Windows XP Pro environment. Overall system is depicted in Figure 1.

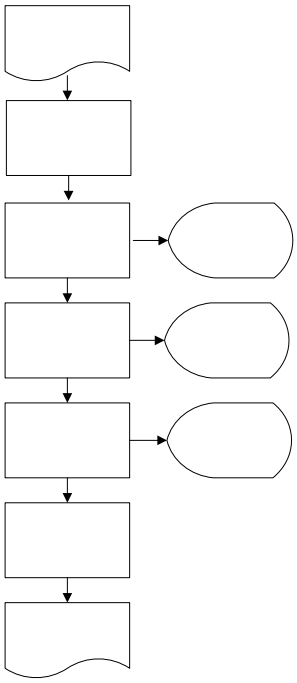

Figure 1. Overall system

Input to the system is acquired from Z specification document in LaTeX format which uses fuzz style [7], then some of its regular expressions is changed to Zaki's one. Z specification document consists of these commands:

```
\documentclass[12pt, zed]{article}
\usepackage{z-eves}
\begin{document}
  ...
 Contents of Z specification
  ...
\end{document
```

Specification document will be scanned by lexical analyzer. If any of lexical errors are found, process will be stopped. Otherwise, expressions produced will be sent to syntax analyzer.

Syntax of specification will be checked and syntax error messages will be presented if any are found. The process is then continued by semantic analyzer. This analyzer will check specification type. Semantic analyzer is taken to verify that a program is not true in syntax but also sentences in it are meaningful [9]. If any of type errors found, error messages will be presented. Last, if specification does not contain any error, conceptual model will be generated.

2.1. System Evaluation

System evaluation is carried out by using two files that contain Z specification namely Security System file (sbk.tex) and Birthday Book System file (shj.tex).

The evaluation is undertaken to test correctness of conceptual model. For SBK specification file, the correctness of conceptual model is accomplished by comparing conceptual model produced by system with the one that Zaki produced. Meanwhile, for SHJ specification file, the correctness of conceptual model is accomplished by reproducing the SHJ specification, based on conceptual model produced.

2.2. Evaluation for SBK Specification

SBK specification is a system, which is developed by Zaki to manage the use of security room. This room can only be used for those who have special pass cards. Functions that added to system are user addition, user deletion, incoming user registration, outcoming user registration, and user list displayer.

2.3 Evaluaion for SHJ Specification

SHJ specification fetched from ZRM second edition which is developed by Spivey [6]. This specification has been changed from its original form, which is in definition of schema calculus RAddBirthday. In original form, this schema's order is above schema NotKnown, while in SHJ specification, it is put above schema RFindBirthday. These changes are undertaken to ease the conceptual model generation. Moreover, ZC06 does not include the second schema RAddBirthday definition.

3. Result

ZC06 produces conceptual model, which is for the first specification (SBK), it is equally the same as the previous research's conceptual model. Moreover, for the second specification (SHJ), from its conceptual model, the specification can be redeveloped successfully.

Figure 2 below shows words table and variables table from conceptual model that Zaki suggested. While the tables below figure 2, are words table and variables table that ZC06 produces. If both of conceptual model are scanned, there is no significant difference is found. In words table, ZC06 generates more lines than Zaki's system. This is because ZC06 also includes function and relation operator as a type of words. The differences in type column found in variables table exists because of types table differences for both systems.

Z Specification

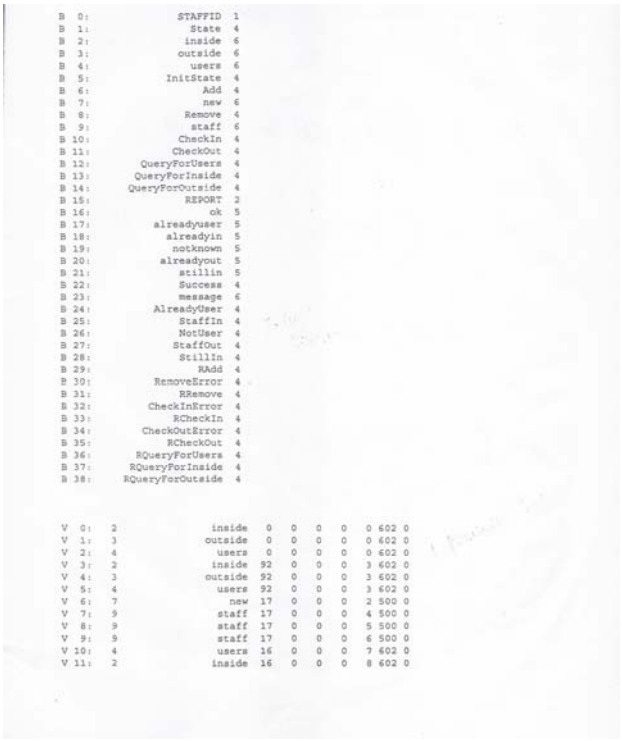

Figure 2 Words table and some of variables table from Zaki's conceptual model

Words Table

W	0:	STAFFID	1
W	1:	State	4
W	2:	inside	7
W	3:	outside	6
W	4:	users	6
W	5:	\cap	10
W	6:	\emptyset	7
W	7:	\cup	10
W	8:	InitState	4
W	9:	Add	4
W	10:	new	6
W	11:	\notin	10
W	12:	Remove	4
W	13:	staff	6
W	14:	\setminus	10
W	15:	CheckIn	4
W	16:	CheckOut	4
W	17:	QueryForUsers	4
W	18:	QueryForInside	4
W	19:	QueryForOutside	4
W	20:	REPORT	2
W	21:	ok	5
W	22:	alreadyuser	5
W	23:	alreadyin	5
W	24:	notknown	5
W	25:	alreadyout	5
W	26:	stillin	5
W	27:	Success	4
W	28:	message	6
W	29:	AlreadyUser	4
W	30:	StaffIn	4
W	31:	NotUser	4
W	32:	StaffOut	4
W	33:	StillIn	4
W	34:	Radd	4
W	35:	RemoveError	4
W	36:	Rremove	4
W	37:	CheckInError	4
W	38:	RcheckIn	4
W	39:	CheckOutError	4
W	40:	RcheckOut	4
W	41:	RqueryForUsers	4
W	42:	RqueryForInside	4
W	43:	RqueryForOutside	4

Variables Table

v	:	0	:	2	inside	0	0	0	0	0	000	0
v	:	1	:	3	outside	0	0	0	0	0	000	0
v	:	2	:	4	users	0	0	0	0	0	000	0
v	:	3	:	2	inside	92	0	0	0	3	000	0
v	:	4	:	3	outside	92	0	0	0	3	000	0
v	:	5	:	4	users	92	0	0	0	3	000	0
v	:	6	:	10	new	17	0	0	0	2	500	0
v	:	7	:	13	staff	17	0	0	0	4	500	0
v	:	8	:	13	staff	17	0	0	0	5	500	0
v	:	9	:	13	staff	17	0	0	0	6	500	0
v	:	10	:	4	users	16	0	0	0	7	000	0
v	:	11	:	2	inside	16	0	0	0	8	000	0

Now, we will evaluate the conceptual model from SHJ specification. Refers to outputTable.txt file, from basic table and words table, basic type description is resulted. See illustration, which is described in Figure 3:

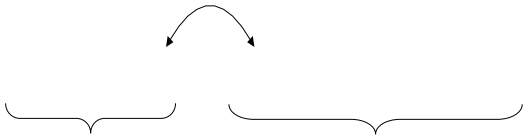

Figure 3 Combining tables in order to redeveloped specification input

Location of basic table is gained from words table. Therefore, NAME and DATE are type of basic (showed by integer number that is preceded by 500) and basic type description (showed by both type of words are 1). Finally, it results [NAME, DATE].

The first schema, the BirthdayBook, is produced by using both of outputTable.txt and outputSkema.txt. Refers to schemas table, it is mentioned that the first schema is BirthdayBook (its location number is 0). Then, refers to schema parser tree, especially the first schema (S 0), we found that this schema declares two variables which are known and birthday. Both of these variables do not decorated. Variable known has 000 as type, which indicate that

its type is not in basic type. So, we refer to types table and found that 000 represent only one operator (indicated by number 1 in the end of line), which is "\power" (integer 10). This operator's first argument is NAME (600 + 0 = 600, word at 0 location is NAME), and it has no second argument (showed by -1).

001 type represents one operator which is "\pfun" (integer 30). This operator shows composit type (1000) and has two argument. The first argument is NAME, produced by 1000 + 0 = 1000. While the second one is DATE, produced by 1000 + 1 = 1001.

If we refer back to schema parser tree, there are two predicate lines ended by 0 at first line and 1 at second line. So, BirthdayBook schema has only one predicate. First predicate line represents "\dom" operator (from integer 51). This operator is unary. Its first argument is 3040, with 300 represents the type of the word is variable, 4 represents birhtday (as 4 is its location number in words table) and 0 represents that it is undecorated.

Second predicate line in parser tree represents "=" operator (integer 147) and has two arguments. The first argument is 3030 (third location in words table is known). The second one is number of first predicate line. So, the full description of BirthdayBook schema is:

```
\begin{schema}{BirthdayBook}
 known: \power NAME \\
 birthday: NAME \pfun DATE
\where
 known = \dom birthday
end{schema}
```

4. Conclusion

The same researches that aim to generate ideal Z conceptual model needs to be encouraged. This is due to the benefit of conceptual model. The conceptual model flares up the use of formal specification. The developing formal software tools is easier if the conceptual model available. The system nowadays and in the future is supported by conceptual model. Moreover, if changes happen in the future, changing works would not be so enormous.

ZC06 has generated Z conceptual model. Conceptual model has been checked by using two methods, namely comparation to conceptual model of previous system and regenerate input specification. Indirectly, ZC06 plays a role to flare up the use of formal specification. Although not so much, ZC06 changes the format of conceptual model developed by Zaki and Abdullah.

We suggest that the future researches integrate specification document preparation with conceptual model generator. Therefore, user will not waste time just to move from one system to another. Actually, Z tools that integrate specification editor, checker and verifier, are large in number, like Z/EVES [10], ZedEdit [11], and others. However, they do not generate conceptual model as output.

Other work aims to advance the ability of this generator, as to check and display lexical, syntax and type error. Moreover, to decrease generator's constraints, which aim to enrich the specification input, operator, and mathematic library that is recognized by the system.

In addition, researches in the future should develop a system that not only generates conceptual model but also integrates an evaluator for the correctness of the conceptual model that has been produced.

Reference

[1] Abdullah Mohd.Zin, ZFDSS: A Formal Development Support System Based on The Liberal Appoarch, Tesis Doktor Falsafah, Universiti Nottingham, 1993.
[2] A.V. Aho, Ravi Sethi, & J.D. Ullman, Compilers: Principles, Techniques, and Tools, Addison Wesley Longman, 1988.
[3] B. Jamison, BYACC/J, 2006.
[4] E. Oxborrow, Database and Database Systems Concepts and Issues, 1990.
[5] G. Klein, The Fast Lexical Analyser Generator: JFlex User's Manual, 2005.
[6] J.M. Spivey, The Z Notation: A Reference Manual., 1998.
[7] J.M. Spivey, The Fuzz Manual, 2000.
[8] Mohd. Zaki bin Haji Ghazali, Sistem untuk Mentahkik Spesifikasi Formal Z, Tesis Sarjana, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 1996.
[9] M. Naeem & C. Harrison, "A Formal Description of A Type Checking Algorithm", Journal of Object Technology 4(9), 93-100, 2005.
[10] M. Saaltink, The Z/EVES 2.0 User's Guide, 1999.
[11] Nantha Kumar Subramaniam & Zarina Shukur, "ZedEdit: Suatu Penyunting Layar untuk Bahasa Spesifikasi Z". Prosiding Bengkel Malaysiana: Spesifikasi Z dan Kaedah Formal dalam Pembangunan Perisian, 82-91, 2003.