Dolittle:A Heuristic Approach to Improving Error Messaging Module Based on Error Feedback Strategy for K12

YongChul Yeum[†], *HyeSun Jang*[†], *DaeYong Kwon*[†], *SeungWook Yoo*[†], *Susumu Kanemune*[††], *and WonGyu Lee*[†††]

[†]Department of Computer Science Education, Graduate School, Korea University
[††]Computer Center, Hitotsubashi University
[†††]Department of Computer Science Education, College of Education, Korea University

Summary
Learning computer programming is a decisive issue in computer science education and error feedback is important to novice in programming learning because it might affect their learning effect affirmatively. But programming is a notorious difficult activity and some of the difficulty can be attributed the programming error. In case of Dolittle[5] which has a lot of advantages as an educational programming language, it has poor module for issuing error message in itself, i.e. redundant and tedious error messages generated by parser which is made by SableCC[16]. Current error message of the Dolittle, therefore, is vague, unfriendly, or misleading to some degree for a novice in K12. In order to issue more effective error message, the Dolittle needs to reform the current error module towards unambiguous, informative, and successful in having a programming experience for the novice. In this paper, we examine the error case carefully and improve the error messaging module generated by LALR parser heuristically.

Key words:
Computer Science Education, Programming Education. Syntax error messages, Error FeedBack.

1. Introduction

Programming is a vital area in computer science education and a fundamental part of the computer science curriculum [8]. Many researches show that computer programming languages help students develop problem solving ability and analytical skills [2][3][4]. In addition, Programming experience as a part of IT education allows students to get a better understanding of software, which is an essential part of computers [7].

However, there has no been EPL (Educational Programming Language) suitable to elementary and secondary computer education such as Dolittle [5]. EPL can help novice to learn programming in a shorter amount of time and lessens the cognitive burdens[18]. Moreover, EPL can be used in classes and getting students to learn various aspects of computers through their programming experience. For instance, Dolittle has lots of advantages as an EPL. It can be expressed in multiple localized languages, e.g. Japanese, Korean, English, and so on. The syntax of Dolittle is so simple and easy to learn that it can be used to learn the fundamental contents of computer science instead of spending a long time to learn the language itself. Consequently, many researches have been conducted and the results proved to be effective and adequate for learning programming experiences for K12 [6][7][12].

Even though it has a lot of strong point as a programming language for EPL, however, there is a weak point. That is the error messages generated by Dolittle interpreters. The problem with Dolittle's error messages is that it just tells the user what error occurred, but do not tell the user what to do to fix the error condition. In general, a proper error diagnosis is a crucial aspect of learning to program, and compiler/interpreter error messages are the main form of interaction between student and machine. Also, they can be an opportunity to learn more about a task by providing error feedback appropriately. In this respect, eliminating all errors is not the best way to solve the problem. The significance of programming derives not only from the works of professional programmer, but also from the work of ordinary people [19]. Especially, the goal of programming education for K12 is not training program developers who can make a code efficiently and rapidly. It focuses on developing problem solving skills by programming. Thus, programming error feedback strategy for novice differs from for developer in that it should affect their learning effect. Therefore, current error massage generated by Dolittle needs to be improved in a more user-friendly manner from the educational viewpoint.

2. Generic Overview of Error Messages

2.1 Taxonomy of Errors

Typically, there are tree kinds of programming errors [2]. The first class is "Lexical errors" which occurs a token in unrecognized. Few errors are discernible at this level alone[10]. Most of programmers make hardly any this type of errors. The second class is "syntax errors" which

Manuscript received July 5, 2006.
Manuscript revised July 25, 2006.

concerns the grammar, or spelling, punctuation and order of words in the program. Often much of the error detection and recovery in a compiler is centered around the syntax analysis phase[10]. The Third class of errors is "semantic errors" which are generated when we have a mistaken idea of how the language interprets certain code. For example, mismatch in variable's type, using variables not declare and method call with wrong argument.

2.2 Error Messages in the case of Novice

Error messages play a fundamental role when novice learns to program [2]. By examining how novice really behaves, we could find that messages in fact determine what novice think and do when confronted with problems. The most basic aspect of learning to program is learning the syntax of a programming language [12]. So, we need to understand novices' behavior when learning to programming.

Studies have shown that excessive time spent on correcting syntax problems can be detrimental to long-term success as students become disheartened with programming [12]. Syntax errors are more important because most of novice programmers can't make a difficult and complex program in logical view. Furthermore, novices struggle with syntactic knowledge because they have trouble recognizing incorrect grammar and a novice may concentrate on small details of syntax as part of the problem-solving strategy[13].

2.3 Factors of Good Error Messages

Message quality is a critical factor in influencing user acceptance of a program product. Good error messages can reduce the time and cost, as well as help users learn about product[14]. Many compilers make error messages that contain unnecessary jargon, are cryptic, unfriendly, or misleading. Many guide line have been written in an effort to improve error messages[14][15].

Good error Messages have to satisfy these factors.

- The message should be appropriate to the user's knowledge and employ user-centered phrasing, that is, from the user's viewpoint.
- The message should provide meaningful suggestions to the user about what to do next
- The message should be brief but informative. It should avoid vague terminology to be sure additional confusion is not created as a result of the error message, that is, easy and understandable.
- The message should guide problem-solving behavior and foster learning, that is, do not simply alert users to problems.

3. Error Feedback Strategy

Error feedback consists of three elements - form, timing, quality. There are many forms of error feedback. If learner makes a syntax error, these forms can be provided. The timing of feedback is also crucial. We can give an immediate or delayed feedback to learner. The quality of feedback is connected with which is good message. According to the form of feedback, it can give different effect to learner.

Following issue should be considered to propose a strategy. Which programming errors can play constructive role in learning? To find out answer of this question, the relation with problem solving skill can be considered. We suggest a strategy that errors which are not concerned with problem solving adopt error preventing strategy.

Syntax errors occur during routine action. Errors of this type should be prevented as much as possible because they are not concerned with problem solving activity. And programming system has to provide an immediate feedback to syntax errors. However, at the beginning stage, syntax errors can occur due to limitation of knowledge. Thus, programming system can provide feedback with more friendly information about the error in a stage of practicing grammar.

Semantic error is related to incomplete or incorrect knowledge on the rule (grammar) of language. In this case, if feedback is given too early before students have a chance to work on a problem, then they will learn less. Thus, it is desirable to provide feedback after learner try to correct several times. For example, programming system prevents going to the next line when the previous line has semantic errors. If the learners cannot correct a code appropriately after they have tried several times, then the system should give a feedback including error information. In this case, we consider that they have a little knowledge about it. The learners had an incentive to think carefully, and this greater 'mindfulness' led to more learning[8]. However, we should adopt different form of feedback about most of the dynamic semantic error because it cannot be detect before program running. Also, delayed feedback should be given to learner in different form.

Logical error is not actual error but unintentional result. It is difficult to provide specific feedback about logical error because it is not always simple to draw a clear distinction between logical error and intentional result. Thus, we can provide feedback of this error in two ways. First, the learners can be provided error feedback indirectly like as simple debugger for novice. Second, we can design a model of programming teaching, and then provide feedback according to stage of the teaching model. If we know what error is often occurred in a certain stage, we can give specific feedback to logical error in each stage.

3. Analysis and Evaluation of Dolittle Error Message

The object-oriented and interpreter based programming language Dolittle was built by the SableCC[16][17] which is an object-oriented framework that generates complier (and interpreters) in the Java programming language. SableCC generates an LALR parser which is one of LR based parsers, and a detailed error message is given to the programmer. But, though Dolittle is based on LR based parser which generators are powerful and well-understood, but the parsers they generate are not suited to provide good error messages as of error feedback strategy.

Dolittle has a printErr module for issuing error messages. But this module can not help having such an inherent weakness of announcing confusing error messages. There are three kinds of error in Dolittle shown in Fig 1.

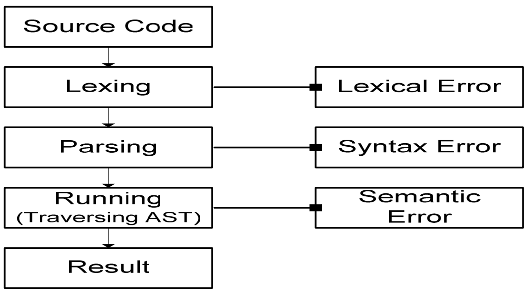
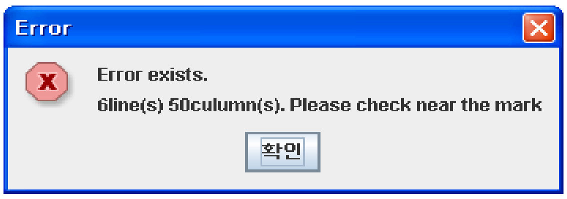


Fig. 1 Three kinds of error in Dolittle.

3.1 Syntactic Error

Dolittle syntactic error happens when missing some symbol of such as period ("."), square brackets ("[...]") and an exclamation mark ("!") etc. But it is so duplicate and redundant that it is hard to grasp what exact error is because error message could be equal in spite of different error situation. For example, in case of the 6th statements of Dolittle source shown in Fig 3 respectively omit the last period (".") , equal sign("="), the 1th left square brackets ("["), errorMessage case obtained by parser module is same. This is the reason why the printErr method issues a similar and vague error message to the user in spite of different error case, especially last line. This problem needs to be solved by analyzing index numbers on errors and classifying the results. Fig 2 is an error message pop up on same syntactic error message of a different case


Error

Error exists.

6line(s) 50culumn(s). Please check near the mark

Fig. 2 Same syntactic error message of a different case

```
(1) Friend = turtle ! create.
(2) [Friend! 100 forward 120 rightturn ] !  3 repeat.
(3) tri = Friend! makefigure (red) paint.
(4) clock = timer ! create 1 period 10 duration.
(5) rBtn = button ! "Run" create.
(6) rBtn:click = [ clock ! [ tri ! 36 rightturn ] execute ].
```

Fig. 3 A sample Dolittle program.

3.2 Semantic Error

Dolittle's parser whose sole purpose is to build a typed abstract syntax tree (AST) while parsing the input. Hence, some node is required to work on an AST to get some action code for being executed. But if a certain AST class returns null because there is not method or object shown in Fig 4 while traversing at some node, i.e., parent is null, it issues an error message generated by OxObject module. That is, Semantic error occurs when trying to use an undefined method or object.

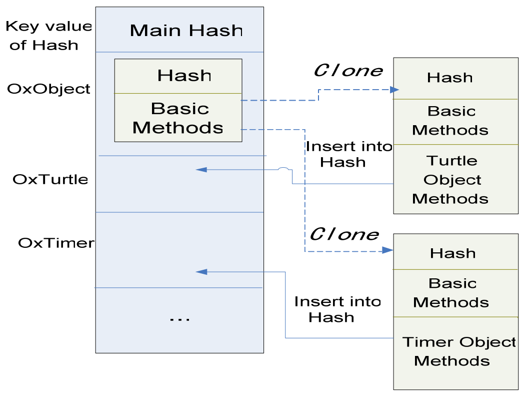
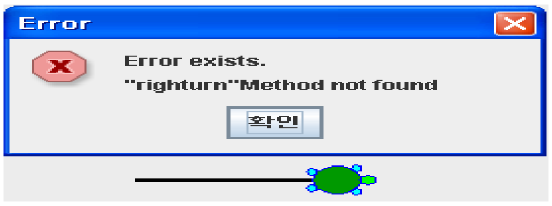


Fig. 4 Object Hash Table in Dolittle.

For instance, Fig 3 illustrates one of examples such case. That is, if the 2th statement shown in Fig 3 is wrong typed "rightturn" as "rightturn", then error message window shown in Fig 5 comes out.


Error

Error exists.
"rightturn"Method not found

확인

Fig.5 Semantic Error Sample caused by trying to use an undefined method.

3.3 Lexical Error

Lexical error occurs when a token is unrecognized such as single quotation mark (‘) which is not undefined token in Dolittle grammer. The lexer package which is generated by SableCC contains the Lexer and LexerException classes that throw an error.

For example, if the 5th statements of Dolittle source shown in Fig 3 is typed wrongly double quotes(“) into single quotes(‘) or omitted double quotes, then lexical error happens. But, this kind of error hardly happens in Dolittle

3.4 Analysis of Error Messages from Experimental Lesson

We designed the experimental lesson to collect real error messages after implementing the server-client Dolittle. This system can make all messages obtained from clients to server Dolittle when students click the run button automatically. These messages can be either error case or not.

The lesson was conducted with 5th grade elementary student. The result of error message’s ratio is shown in Table 1.

Table 1: Ratio of Error Messages Case

Syntactic error	Semantic error	Lexical error
71(%)	27(%)	2(%)

3.5 Evaluation of Dolittle Error Messages

Although there is error message routine in Dolittle, such as printErr module, it needs to be improved for K12 based on error feedback strategy. As we described earlier in this paper, one of the good error messages condition is to provide meaningful suggestion to the user about what to do next. For example, error message shown in Fig 2 could be meaningless all but perceiving something wrong. This kind of error message can make novice computer users feel inadequate and intimidated.

4. Improving Error Messaging Module

As the result of the experimental lesson, we resolve to concentrate on syntactic and semantic error case. The algorithm to find syntax error states is shown in Fig 6 and it is to conform to a typical mechanism of LALR parser. The parser reads one token from an input buffer at a time. The parser uses a stack to store a string of the form s0X1s1X2s2 … Xmsm, that the sequence of tokens returned by the lexer conforms to a grammar. The parsing table determines sm , currently on the top of the stack, the state symbol that summarizes the information that might be considered in selecting an error message, and ai, the current input symbol. It then consults action[sm, ai], the parsing action table entry for state sm and input ai, which can have one of four values, such as shift, reduce, accept and error. If the state action[sm, ai] is an error state, an error message consists of all the terminal and nonterminal symbol that are on the parse stack, plus the current token.

However, this sort of mechanism for consisting of error message is not successful in finding an exact error situation and issuing an informative error message because distinct error states are not listed in the errorMessage table. So Dolittle errorMessages obtained by parser module is the same number shown in Table 2 and it is duplicate and redundant

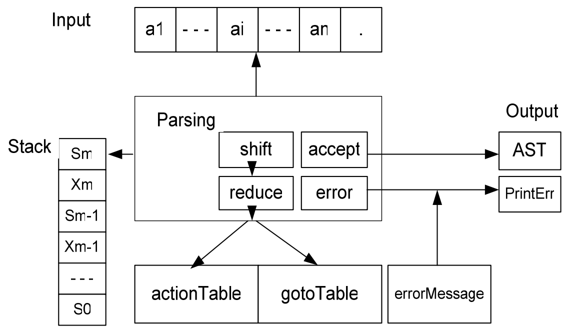
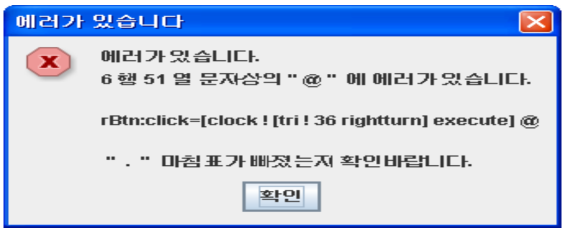


Fig. 6 Schematic form of an LR parser.

Table 2: Index of errorMessage, Error number from a Sample Dolittle Source [7]

token / row	dot .	1th/2th !	=	1th/2th [	1th/2th]
1	(4,59)	(7,10)	(7,10)		
2	(4,59)	(8,29) (7,10)	(7,10)	(7,10)	(4,59)
3	(4,59)	(7,10)	(7,10)		
4	(4,59)	(7,10)	(7,10)		
5	(4,59)	(7,10)	(7,10)		
6	**(7,10)**	(8,29)	**(7,10)**	**(7,10)** (4,59)	(4, 33) (21,56)

To solve this problem, we used the pair of integers, i.e. state and token, and last token. This improved PrintError module can get these states and rebuild error message. Using the input token in producing error messages can help to report syntax errors with a better message, or a suggestion of how to fix the error. To do this, we examine the each error case carefully and heuristically and make a more effective error message table that considers the current token. Fig. 6 shows more correct error messaging.



에러가 있습니다.
6 행 51 열 문자상의 " @ " 에 에러가 있습니다.

```
rBtn:click=[clock ! [tri ! 36 rightturn] execute] @
```

" . " 마침표가 빠졌는지 확인바랍니다.

Fig. 6 Syntactic improved error message on missing out a period.

In case of semantic error, we add new module which search related object or method. As explained previously, the current module just issues an error message generated by OxObject module unfriendly when a certain AST class returns null because there is not method or object while traversing at some node. But we add a module to find correct error case, while searching the hash table.

5. Conclusion

Education programming language such as Dolittle can be useful to teaching computer programming for K12. Although EPL can be an important tool, however, its usefulness will be decreased if its error messaging mechanism is poor. In case of Dolittle as an EPL, it has some weakness that is the error messaging methodology, while many researches have shown the merits or possibility of Dolittle for K12 education[7]. The defect of Dolittle is inherent in itself to some degree, because of being made by SableCC, i.e. LALR based compiler.

We proposed error feedback strategy from a conceptual point of view. Especially, when novice programming system is designed, error feedback is critical point that a designer should consider. The next step is designing concrete error feedback by taking account of its elements - form, timing, quality.

Moreover we designed a better error handling module for Dolittle, which distinctly identifies the proviso issued by parser. And this module was designed and implemented heuristically by analyzing index numbers on each error case and classifying the result.

Finally, a better error message describing the parse state can be further improved for a token with a more specific error messages towards user-centered, informative, and successful for the novice.

References

[1]Marzieh Ahmadzade, Dave Elliman, Colin Higgins. An Analysis of Pattern of Debugging Among Novice Computer Science Students. ACM. ITiCSE. 84-88. 2005.

[2]J.D. Bransford, A. L. Brown, and R.R. Cocking, editors. How People Learn: Brain, Mind, Experience, and School. National Academy Press, Washington, D.D., 2000.

[3]M. Resnick,. . New paradigms for computing, new paradigms for thinking. In A. diSessa, Hoyles, C., & Noss, R. (Eds.), Computers and Exploratory Learning (pp. 31-43). New York: Springer-Verlag, 1995

[4]Seymour Papert, Mindstorms: children, computers, and powerful ideas, Basic Books, 1980

[5]Dolittle Programming Language, **http://kanemune.cc.hit-u.ac.jp/dolittle/**

[6]Susumu Kanemune, Takako Nakatani, Rie Mitarai, Shingo Fukui, and Yasushi Kuno. Dolittle - Experiences in Teaching Programming at K12 Schools. The Second International Conference on Creating, Connecting and Collaborating through Computing, IEEE, 177-184, 2004

[7]Susumu Kanemune, Yasushi Kuno. Dolittle : an object-oriented language for K12 education, Eurologo 2005, 2005

[8]Allen Tucker. A Model Curriculum for K-12 Computer Science. Final Report of the ACM K-12 Education Task Force Curriculum Committee. ACM. 2003.

[9]Edward J. Shaw Jr. Making APL Error Messages Kinder and Gentler, ACM. pp 321, 1989

[10]Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers Principles, Techniques, and Tools, 1986

[11]H.S.kim, H.S.Jang, H.C.Lee, D.Y.Kwon, Y.C. Yeum. S.W.Yoo, H.C.Kim, W.G.Lee, Teach Programming to Non-CS Major Students : Experiments with Storymaking Approach, SSS2004, 2004

[12]Sarah K Kummerfeld and Judy Kay. The neglected battle fields of Syntax Errors. Australian Computer Society. In Proceedings of the fifth Australasian conference on Computing education. 105-111. 2002.

[13]Linda Mclver. Syntactic and Semantic Issues in Introductory Programming Education. Monash University. Doctor Thesis. 2001.

[14]Barbara s. Isa, James M. Boyle, Alan S. Neal, Roger M. Simons. A Methodology for Objective Evaluating Error Message. ACM. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems 68-71. 1983.

[15]Rolf Molich and Jakob Nielsen. Improving a Human Computer Dialogue. ACM. Communications of the ACM. 33(3). 338-348. 1990.

[16]SableCC, **http://sablecc.org/**

[17]Etienne Gagnon, SableCC : An Object Oriented Compiler Framework, McGill University. Master Thesis. 1998.

[18]SeungWook Yoo, Empirical Study of Educational Programming Languages for K12: Between Dolittle and Visual Basic, IJCSNS, 2006

[19]John F. Pane. A Programming System for Children that is Designed for Usability. Carnegie Mellon University. 2002.

YongChul Yeum received the B.S. degree in Mathematics Education from Seoul National University of Education in 1991 and the M.S. degree in Computer Science Education from Seoul National University of Education in 2002. He has been studying in the Department of Computer Science Education of Korea University. Main research field is the educational programming language in K12 especially.

HyeSun Jang received the B.S. degree in Computer Science Education from Korea University in 2005. She has been studying in the Department of Computer Science Education of Korea University. Main research field is the error feedback of educational programming language.

DaiYong Kwon received the B.S. degree in Computer Science Education from Korea University in 2004 and the M.S degree in Computer Science Education from Korea University in 2006. He has been studying in the Department of Computer Science Education of Korea University. Main research field is the education programming environment and educational robot.

SeungWook Yoo received the B.S. degree in Mechanical Engineering Education from Chungnam National University in 1983 and the M.S. degree in Computer Science Education from Korea University in 2002. He has been studying in the Department of Computer Science Education of Korea University. Main research field is the educational programming language in K12 especially.

Susumu Kanemune received the PhD in Systems Management from University of Tsukuba in 2004. His research interests are in the programming language and information education. He is an Associate Professor of Hitotsubashi University.

WonGyu Lee received the B.A. degree in English Language and Literature from Korea University in 1985, M.S. and Ph. D. degree in System and Information Engineering from University of Tsukuba in 1993 respectively. During 1993-1995, he stayed in the Korean Culture & Art Foundation. Since 1996, he is a professor of Computer Science Education at Korea University.