Francisco J. Almeida-Martínez
Jaime Urquiza-Fuentes
Antonio Pérez-Carrasco

Building LR (1) grammars and parsers using VAST and CUP.

Number 2011-11

Serie de Informes Técnicos DLSI1-URJC
ISSN 1988-8074
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos

Index

1. Introduction 3
2. Description of the evaluation 3
2.1. Subjects..3
2.2. Experimental design………… 4
2.3. Tasks and protocol...…………………………………………………4
3. Results…..5
3.1. Instructors observations..5
3.2. Answers to questionnaires..5
3.3. Results of educational effectiveness ……………..…………………………7
3.4. Interviews……………………………………………………………8
4. Conclusions..9
Appendix..10
References……………………………………………………………..10

Building LR (1) grammars and parsers using VAST and CUP

Francisco J. Almeida-Martínez,Jaime Urquiza-Fuentes and Antonio Pérez-Carrasco

Universidad Rey Juan Carlos
Departamento de Lenguajes y Sistemas Informáticos I
c/ Tulipn s/n, 28933 Madrid, Spain
`francisco.almeida@urjc.es, jaime.urquiza@urjc.es, antonio.perez.carrasco@urjc.es`

Abstract. In this document we describe an evaluation performed with VAST to build LR (1) parsers and grammars. The results of the evaluation are satisfactory, so although we did not find any statistics significant differences, we noticed that students could finish the exercises faster. Besides the number of deliveries was higher in the treatment group than in the control one.

1 Introduction

In the previous evaluation we used VAST to design LL (1) grammars and parser in just one session [1], [2]. In this case we did not find any statistical significant difference between the control and treatment group. However, we observed that using VAST the students were more motivated in the subject, so the number of deliveries was higher in the treatment group. The objective of this evaluation was to analyze if using VAST, with just one session, it would help students to design better grammars and LR (1) parsers. The version of the tool used in this case was the same that the one used in previous evaluations.

2 Description of the evaluation

In this section we describe the evaluation. We refer to the participants, the experiment's design, the tasks performed during the session and the protocol.

2.1 Subjects

In this evaluation participated **19 students** of the *Language Processing* subject of the Rey Juan Carlos University during the 2010-2011 course. The participation was voluntary and based-incentive in a 1.25% over the final mark only if they passed the exam.

2.2 Experimental design

This evaluation was designd as an educational effectiveness study plus an usability-quality and observational evaluation. Students were divided in two groups: control and treatment. The control group used the tools JFlex-Cup while the treatment one VAST. In order to create the groups we used the marks of a pretest of knowledge. We used these marks to separate the students in control and treatment group, although the assigment to each group was random. The pretest of knowledge included questions according to the Bloom's taxonomy [3]. In table 1 we show the results of the pretest. As it can be observed there does not exist any difference between the groups. In this evaluation we did not consider the level of *Understanding*. The independent variable was the tool used by each group; VAST in treatment group and JFlex-Cup in control one. The dependent variables were the educational effectiveness according to the differences between postest-prestest, the students' opinion about four fundamental aspects related to the usability-quality: ease of use, learning support and quality of the tool. This evaluation lasted 2 hours (one session).

Level	**Control**	**Treatment**	**Stats.**
Knowledge (K)	0.01	0.01	U=308.50, p=0.57
Application (Ap)	0.07	0.00	U=98.00, p=0.33
Synthesis (S)	0.00	0.00	U=105.00, p=1.00
Total	0.01	0.001	U=348.50, p=0.95

Table 1. Results of the pretest of knowledge according to the Bloom's taxonomy

2.3 Tasks and protocol

The tasks performed by students had to be documented al the end of the evaluation (see appendix 5) with text explanations and visualizations using VAST in treatment group and any other software in case of control group. The tasks consisted in 3 exercises about design of grammar for LR (1) parsers. One week before the experiment we explained how to used JFlex-Cup in class. Besides, at the beginning of the evaluation we explained to each group how to work with the corresponding tools.

Treatment group. This group had to solve the exercises using VAST's grammar editor and get a correct grammar in order to recognize all the input streams given. The solutions to each exercise consisted in the correct grammar specification with the input streams tested (correct and incorrect ones) together with the visualizations and the text explanations.

Control group. This group had to work only with the tools JFlex-Cup. As in the treatment group, they were given the features of the grammar and they had to build the corresponding parser. In this case, the solutions to the exercises were the grammar specification, the input streams used (correct and incorrect) together with visualizations and text explanations.

In table 2 we show the protocol used in this evaluation. Three weeks before the experiment we did the pretest of knowledge in theory class. Two weeks later (one week before the experiment) we explained how to work with the parser generators JFlex-Cup. In the evaluation we explained how to work with VAST (only in treatment group) for 15 minutes. At the end of the session, we asked the students for completing an usability-quality questionary about the tools used. The exercises of the evaluation had to be sent by email two days after. The same day we did the postest of knowledge.

Control	**Treatment**
Pretest of knowledge	
JFlex-Cup session	
JFlex-Cup experiment	VAST experiment
JFlex-Cup questionary	VAST questionary
Postest of knowledge	

Table 2. Protocol followed during the evaluation

3 Results

In this section we describe the results of the evaluation. During the evaluation the instructors observed how the students worked with the tools and the problems they found using both tools. The results are divided in three categories: instructor's observations, answers to usability-quality questionaries, results of educational effectiveness and interviews to students.

3.1 Instructor's observations

During the evaluation the instructors annotated how the students used the tools. In the control group we observed that students used general porpuse editors to implement the parsers (grammar design) using JFlex-Cup. At the end of the evaluation none of the students finished the exercises.

According to the treatment group we observed that students remmembered how to work with VAST. Many of them found useful the functionality of resuming subtrees due to the size they can reach. After one hour we observed that many students had finished the first exercise.

3.2 Answers to questionnaires

The opinion questionaries of VAST and JFlex-Cup (see appendix 5) were designed to make possible students give their opinion about the tools. The questionaries used a Likert scale with 5 values where the 1 is the lowest mark and

5 the highest. Besides we included open questions. We performed an analysis of the average marks obtained in the questionaries, which allows to get information about the ease of use, quality (general and/or specific) and the learning support. In table 3 we show the average marks obtained for the general and specific parts in each group.

Aspect	**Control**	**Treatment**	**Stats.**
Ease of use			
General ease of use	3.25	3.92	t(14)=-1.51, p=0.15
Average of parts	3.45	3.48	t(14)=-0.05, p=0.97
Learning support			
Syntax tree building	1.33	3.67	t(13)=-7.12, **p<0.01**
Stack	2.00	3.50	t(13)=-2.52, **p=0.03**
Input stream	2.33	3.41	t(13)=-1.96, p=0.34
Quality			
General quality	2.33	3.92	t(13)=-3.21, **p<0.01**
Average of parts	3.40	3.52	t(14)=-0.34, p=0.74

Table 3. Usability-quality marks

Apart from rating each different aspect of the tools, the questionaries asked for the students' opinions about the difficult aspects, the best and worst parts, the lack of features and the positive and negative aspects. As difficult aspects to use the control group mentioned the way of communicating JFlex-Cup and the error messages used by the tools. In the treatment group students pointed out that the resume of subtrees was extremely useful because it helps to work with large trees. According to the best quality aspects in the treatment group students mentioned the input stream processing, the compatibility with LL and LR parser, the generation of the syntax tree and the grammar editor. As aspects to add to the tools, in the treatment group students thought that it would be fine to edit the input stream directly. In the control group they asked for better information to debug the grammar, a grammar editor and a graphical interface. As positive aspects, in the treatment group they pointed out the compatibility with both LL and LR parsers.

3.3 Results of educational effectiveness

The results of educational effectivenes are divided in two parts. On the one hand, the differences between protest-pretest. On the other hand, the marks obtained in the exercises of the evaluation.

In table 4 we show the differences of the marks according to the Bloom's taxonomy [3]. As we can observe there does not exist significant statistic differences in any of the taxonomy's levels.

Level	**Control**	**Treatment**	**Stats.**
Knowledge (K)	0.44	0.25	t(17)=1.22, p=0.24
Application (Ap)	0.81	0.36	U=21.00, p=0.08
Synthesis (S)	0.13	0.09	U=35.00, p=0.69
Total	0.41	0.21	t(17)=1.64, p=0.12

Table 4. Results of educational effectiveness according to the Bloom's taxonomy

According to the solutions to the practices, we have observed that in the treatment group, a larger number of students delivered the exercises. The result of the analysis is t(9)=-3.77, p<0.01.

3.4 Interviews

Once we have analyzed the results of the evaluation, we performed an interview to those students who left the evaluation

In the control group, 3 students had problems to perform the configuration, generation and compilation with Cup. One student commented that the time was not enough. Another student pointed out that he/she did not know how to solve the practice.

In the treatment group, some students said that there was not enough time although they know how to solve the exercises. Two students pointed out that the tool was easy to use.

4 Conclusions

The results of this evaluation are divided in two parts: results of educational effectiveness and students' opinion about the usability-quality of the tool.

According to the educational effectiveness, results are divided in two parts: solutions to the practices and differences between postest-pretest. The results obtained show that students in the treatment group have performed better the exercises than the control one t(9)=-3.77, p<0.01. We have also observed that the delivery of the exercises has been larger in the treatment group (72.73% 8/11) than in the control one (42.86% 3/7). The results obtained in the two last evaluations [2] show that the number of desertions is larger in the control group than in treatment one. This could be due to the problems found by students when they work directly with the generation tools without any visual support, which explains the behaviour of the parser while the input stream is being processed. As the treatment group used VAST, which makes easier the grammar design, the generation of parsers and allows to display their behaviour during the analysis, students can deliver the exercises easilly. The differences between postest-prestest show that there does not exist any significant difference.

From these results we have observed that the design of the evaluation has failed due to the date chosen to perform it. However, as in previous evaluations, the use of VAST has motivated studetns not to leave the evaluation. The fact of getting more deliveries in the treatment group, means that students in this group have worked more in the subject.

The results of usability-quality show that there are statistical significant differences in favor of the treatment group (VAST) according to the opinion about the *building of the syntax tree*, the*input stream* and the *general qualitiy of the tool*.

From this study, we think that results are satisfactory. This evaluation was designed as an usability-quality study plus an educatinal effectiveness evaluation (short-term). However, previous to this evaluation we performed another one [2] with the tool ANTLR for LL (1) parsers. In these evaluations the students of both groups (treatment and control) were the same. Due to this in this evaluation students were used to work with VAST. This fact could affect the students' opinion about the tool.

Once we have analyzed the results we plan a complete evaluation of the syntax error recovery. Taking into account the reduce number of students in this evalaution we decide to increment the incentive.

5 Acknowledgment

This project is supported by project TIN2008- 04103/TSI of the Spanish Ministry of Science and Innovation.

A Exercise used for the pre-postest

1. What are the main features of the bottom-up parser?
2. Which technics of bottom-up parsers do you know? Which is it the more powerfull?
3. Which are the limitations of the LALR parsers?
4. Given the following grammar and the LR (1) parsing table (see table 5):

```
(1) S::=L=R
(2) S::=R
(3) L::= *R
(4) L::= id
(5) R::= L
```

   Given the following input stream *id=id*id$*, simulate its parsing indicating the state of the stack pointing out the part processed.
5. Design a grammar to parser input stream to declare **for-endfor** blocks. Each block has a counter, a condition in parenthesis and a counter's step. This one can be specified using **increment/decrement or any maths expression**. Each part of the sentence has to be separate with **;**. Each **for** sentence can be nested. Following we show a valid input stream for this grammar:

	Action-Table				Go-to Table		
State	=	*	**id**	**$**	**S**	**L**	**R**
0		s4	s5		1	2	3
1				acc			
2	s6			r5			
3				r2			
4		s4	s5			8	7
5	r4			r4			
6		s11	s13			10	9
7	r3			r3			
8	r5			r5			
9				r1			
10				r5			
11		s11	s13			10	12
12				r3			
13				r4			

Table 5. LR (1) parsing table

```
for (int i=0; i<=(i+3); i++)
   var1=var2+var3;
   for (var1; var1<=30; var*2)
      var4=var1-2
   endfor
endfor
```

B Exercises used during the evaluation

1. **Problem 1.** We want to know if an input stream has a certain structure to parser postal address. In particular, the input stream has to follow this scheme:

   `NAME: ID_N SURNAME1: ID_SURNAME1 SURNAME2: ID_SURNAME2 MARKS: NUMBER, NUMBER, NUMBER.`

   Where:
   – NAME, SURNAME1, SURNAME2 and MARKS are text which allow to distinguish the element to detect.
   – NUMBER is an integer or float number (using `.` as separator), the sign is optional. Examples: -2, +1, 3, 4.5, +5.0.
   – ID: text that can contain any alphanumeric character.

   As example of input stream we can consider:
   Nombre: Fernando Apellido1: Arroyo Apellido2: Montoso Notas: 0.9, 0.5, 6.3

2. **Problem 2.** We want to build a parser to detect file's path. A path can refer to a local o remote file. For local files we use the format : UNIT: FILE. In case of remote files we use the following format:

   ```
   (PROTOCOL:://)? SIMBOLIC_ADDRESS (/FILE)?
   ```

   Where:
   - UNIT: it can be a text with the following values: C, E or F.
   - DOMAIN: a sequence of alphanumberic characters. In our case, its value will be restricted to characters a-z and a number (except at the beginning).
   - PROTOCOL: it will be a text to indicate the type of the protocol. We allow the following protocols: "http", "ftp", "gopher", "ssh" y "plPro". If the protocol is not indicated we assume http.
   - SYMBOLIC_ADDRESS: it is a sequence of labels separated by **:** which represent the address of the remote machine. The format is DOMAIN. DOMAIN (. DOMAIN)*
   - FILE: it is the path inside the remote machine. The format is: (DOMAIN/)* DOMAIN

   As example of input streams we have:
   C:test/test2
   http:://remote1.remote2/test1/test2
3. **Problem 3.** We wan to implement a parser for a specific programming language which source code is embbed into HTML language. The format used is: HTML LANGUAGE HTML2
   Where:
   - HTML1: it refers to the init HTML code: HTML_ini, BODY_ini
   - HTML2: it refers to the finish HTML code: BODY_end, HTML_end
   - LANGUAGE: it refers to the language which want to be detected. This new language has two different zones: declaration and use. We use the following format:
     - DECLARATION (DECLARATION)*
     - (USE)*
   - DECLARATION: it cannot be empty. We allow to declare integer, boolenas and float variables. Ex: var int x.
   - USE: it can be empty. It allows the use of the following senteces: IFELSE, WHILE, REPEATUNTIL and SENTENCES.
     - SENTECES: it refers to assigment sentences (::=) and arithmetic expressions (adds, subs, divs, mults).
     - IFELSE: it allows to declare conditional sentences with *else* obligatory and some conditions using the logic operator (*and, or*) and the relational operators (<, >). Inside of the if we can use any other sentences.
     - REPEATUNTIL: it allows the declaration of postevaluated structures.

   An example of a valid input stream could be:

```
html_ini
body_ini
var int i
var bool i2
if(i2 or i2)
{
   i2=i2+2
}
else
{
   i2=i2-2
}
body_end
html_end
```

References

1. Francisco J. Almeida-Martinez and Jaime Urquiza-Fuente. Teaching LL(1) parsers with VAST- An Usability Evaluation-. Technical Report 2009-01, Universidad Rey Juan Carlos, 2009.
2. Francisco J. Almeida-Martnez, Jaime Urquiza-Fuente, and Antonio Prez-Carrasco. Building LL(1) grammars and parsers using VAST and ANTLR. Technical Report 2011-06, Universidad Rey Juan Carlos, 2011.
3. B. Bloom, E. Furst, W. Hill, and D.R. Krathwohl. *Taxonomy of Educational Objetives: Handbook I, The Cognitive Domain.* Addison-Wesley, 1959.