Toward Language-Independent Sugar Libraries

ELTON M. CARDOSO
Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brazil

RODRIGO G. RIBEIRO
Universidade Federal de Ouro Preto
Ouro Preto, Minas Gerais, Brazil

LEONARDO V. S. REIS
Universidade Federal de Juiz de Fora
Juiz de Fora, Minas Gerais, Brazil
lvsreis@ice.ufjf.br

ABSTRACT

KEYWORDS

PEG, parsing, semmantics

ACM Reference Format:
ELTON M. CARDOSO, RODRIGO G. RIBEIRO, and LEONARDO V. S. REIS. 2021. Toward Language-Independent Sugar Libraries. In *Proceedings of XXV BRAZILIAN SYMPOSIUM ON PROGRAMMING LANGUAGES (SBLP2021)*. ACM, New York, NY, USA, 2 pages.

1 INTRODUCTION

2 DEFINITION FO THE PARSING GRAMMAR EXPRESSIONS AND THE ABSTRACT MACHINE

The parsing expressions syntax is given by the grammar of the Figure 1,as defined by [1].

Figure 1 Grammar for Parsing Expression

$$
\begin{array}{lcl}
\langle e \rangle & ::= & a \\
& | & \epsilon \\
& | & e\ e \\
& | & e/e \\
& | & e* \\
& | & !e \\
& | & v
\end{array}
$$

The parsing expression grammar G is a set of pairs (V, e) whose V is a varibale. The evaluation context for a PEG is defined by the grammar of the Figure 2.

Figure 2 Grammar for evaluation context

$$
\begin{array}{lcl}
\langle m \rangle & ::= & a \\
& | & \epsilon \\
& | & \odot e \\
& | & e \odot \\
& | & \oslash e \\
& | & e \oslash \\
& | & \star \\
& | & \neg
\end{array}
$$

SBLP2021, September 23–27, 2021, Brazil

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

The machine state is described by 5-upla $(G, e, \Gamma, \langle z \bullet w \rangle)$. where G is a Peg Grammar, e is a peg expression, Γ is an evaluation context, the $\langle z \bullet w \rangle$ is a zipper describing on the input string, where z is the consumed portion of the input and w is the reminder of the input. The empty input, represented by λ. Failed computations fail, the zipper will be subscripted, becoming a $\langle z \bullet w \rangle_{\perp}$.

The parsing expression e will be preceded by a $\downarrow$, to indicate that the processing of that expression is to be started, or by an $\uparrow$ to indicate that the processing of that expression is to be finished.

The Γ context is managed as a stack, the empty contexto is written $[]$. A non-empty context is written $m : \Gamma$, where m is a context expression.

A PEG grammar G is a set of pairs $(\mathcal{V}, e)$ and denotes a rule $\mathcal{V} \leftarrow e$ onde $\mathcal{V}$ is a variable. For simplicity reasons it is assumed that there is only one varibale $\mathbb{V}$ in G.

3 SMALL STEP SEMMANTICS

The semmantics relation has the form $(G, e, \Gamma, z \bullet w) \triangleright (G, e, \Gamma, \langle z' \bullet w' \rangle)$ where G is a Parsing Expression Grammars, e is an expression, Γ is a stack of m expr , z is the consumed input and w is the input.

The rule 1 displayed in Figure 3 shows that when beginning processing the empty PEG, $\downarrow \epsilon$, the result is a state where the empty peg finished without consuming any input string. This rule always succeeds.

Figure 3 Rules for a simple terminal

$$(1) \quad (G, \downarrow \epsilon, \Gamma, \langle z \bullet w \rangle) \triangleright (G, \uparrow \epsilon, \Gamma, \langle z \bullet w \rangle)$$

The rule 1 displayed in Figure 3 shows that when beginning processing the empty PEG, $\downarrow \epsilon$, the result is a state where the empty peg finished without consuming any input string. This rule always succeeds.

Figure 4 Rules for a simple terminal

$$
\begin{array}{ll}
(2) & (G, \downarrow a, \Gamma, \langle z \bullet aw \rangle) \triangleright (G, \uparrow a, \Gamma, \langle za \bullet w \rangle) \\
(3) & (G, \downarrow a, \Gamma, \langle z \bullet bw \rangle) \triangleright (G, \uparrow a, \Gamma, \langle z \bullet bw \rangle_{\perp}) \\
(4) & (G, \downarrow a, \Gamma, \langle z \bullet \lambda \rangle) \triangleright (G, \uparrow a, \Gamma, \langle z \bullet \lambda \rangle_{\perp})
\end{array}
$$

Rules 5 to 9 determine the behavior on a sequence construction. Rule 5 states that the result of processing sequence $e_1 e_2$ is to begin the processing of the subexpression e_1 inserting the expression $\odot e_2$ in the evaluation context. Rule 6 shows that in a state where the processing of subexpression e_1 has ended and the expression $\odot e_2$ is on top of the evaluation stack the result is

Figure 5 Rules for seqeunce

$$
\begin{array}{rl}
(5) & (G, \downarrow e_1e_2, \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle) \\
(6) & (G, \uparrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle) \\
(7) & (G, \uparrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle) \\
(8) & (G, \uparrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle_\perp) \\
(9) & (G, \uparrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle_\perp)
\end{array}
$$

Figure 6 Rules for alternative

$$
\begin{array}{rl}
(10) & (G, \downarrow e_1/e_2, \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e_1, \oslash e_2 : \Gamma, \langle z \bullet w \rangle) \\
(11) & (G, \uparrow e_1, \oslash e_2 : \Gamma, \langle z \bullet w \rangle) \rhd (G, \uparrow e_1/e_2, \Gamma, \langle z \bullet w \rangle) \\
(12) & \dfrac{(G, \uparrow e_1, e_2 \oslash : \Gamma, \langle z' \bullet w' \rangle_\perp)}{(G, \uparrow e_1, e_2 \oslash : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \downarrow e_2, e1 \oslash : \Gamma, \langle z \bullet w \rangle)} \\
(13) & (G, \uparrow e_2, e_1 \oslash : \Gamma, \langle z \bullet w \rangle) \rhd (G, \uparrow e_1/e_2, \Gamma, \langle z \bullet w \rangle) \\
(14) & (G, \uparrow e_2, e_1 \oslash : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow e_1/e_2, \Gamma, \langle z \bullet w \rangle_\perp)
\end{array}
$$

Figure 7 Rules for not

$$
\begin{array}{rl}
(10) & (G, \downarrow !e, \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e, \neg : \Gamma, \langle z \bullet w \rangle_\perp) \\
(11) & (G, \uparrow e, \neg : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow !e, \Gamma, \langle z \bullet w \rangle)
\end{array}
$$

Figure 8 Rules for klenee

$$
\begin{array}{rl}
(10) & (G, \downarrow e_1e_2, \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle) \\
(11) & (G, \uparrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle) \rhd (G, \downarrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle) \\
(12) & (G, \uparrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle) \\
(13) & (G, \uparrow e_1, \odot e_2 : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle_\perp) \\
(14) & (G, \uparrow e_2, e_1 \odot : \Gamma, \langle z \bullet w \rangle_\perp) \rhd (G, \uparrow e_1e_2, \Gamma, \langle z \bullet w \rangle_\perp)
\end{array}
$$

$$
\begin{array}{lcl}
(G, \downarrow \epsilon, \Gamma, z \bullet w) & \rhd & (G, \uparrow \epsilon, \Gamma, z \bullet w) \\
(G, \downarrow e1/e2, \Gamma, z \bullet w) & \rhd & (G, \uparrow e1, \otimes e2 : \Gamma, z \bullet w)
\end{array}
$$

4 HASKELL IMPLEMENTATION

5 RELATED WORK

6 CONCLUSIONS

ACKNOWLEDGMENTS

This work is supported by the CNPq – Brazil under grant No.: 426232/2016.

REFERENCES

[1] Bryan Ford. 2004. Parsing Expression Grammars: A Recognition-Based Syntactic Foundation. *SIGPLAN Not.* 39, 1 (Jan. 2004), 111â122. https://doi.org/10.1145/982962.964011