Searching for asymptotic error repair[⋆]

M. Vilares, V.M. Darriba, and M.A. Alonso

Department of Computer Science, University of A Coruña
Campus de Elviña s/n, 15071 A Coruña, Spain
{vilares,alonso}@udc.es, darriba@dc.fi.udc.es

© Springer-Verlag

Abstract. We work in the domain of a regional least-cost strategy with dynamic validation in order to avoid cascaded errors [3], extending the theoretical model to illustrate its asymptotic equivalence with global repair algorithms. This is an objective criterion to measure the quality of an error repair algorithm, since the point of reference is a technique that guarantees the best quality for a given error metric when all contextual information is available. To the best of our knowledge, it is the first time that such a discussion takes place. We also reformulate the parsing framework using parsing schemata [1], simplifying the description.

1 The parsing model

Our aim is to parse a sentence $w_{1\ldots n} = w_1 \ldots w_n$ according to an unrestricted context-free grammar $\mathcal{G} = (N, \Sigma, P, S)$, where the empty string is represented by ε. We generate from $\mathcal{G}$ a *push-down automaton* (PDA) for the language $\mathcal{L}(\mathcal{G})$. In practice, we chose an LALR(1) device generated by ICE [2], although any shift-reduce strategy is adequate. A PDA is a 7-tuple $\mathcal{A} = (\mathcal{Q}, \Sigma, \Delta, \delta, q_0, Z_0, \mathcal{Q}_f)$ where: $\mathcal{Q}$ is the set of states, Σ the set of input symbols, Δ the set of stack symbols, q_0 the initial state, Z_0 the initial stack symbol, $\mathcal{Q}_f$ the set of final states, and δ a finite set of transitions of the form $\delta(p, X, a) \ni (q, Y)$ with $p, q \in \mathcal{Q}$, $a \in \Sigma \cup \{\varepsilon\}$ and $X, Y \in \Delta \cup \{\varepsilon\}$.

To get polynomial complexity, we avoid duplicating stack contents when ambiguity arises, storing them in a table $\mathcal{I}$ of *items*, $\mathcal{I} = \{[q, X, i, j],\ q \in \mathcal{Q},\ X \in \{\varepsilon\} \cup \{\nabla_{r,s}\},\ 0 \leq i \leq j\}$; where q is the current state, X is the top of the stack, and the positions i and j indicate the substring $w_{i+1} \ldots w_j$ spanned by the last category pushed onto the stack. The symbol $\nabla_{r,s}$ indicates that the part $A_{r,s+1} \ldots A_{r,n_r}$ of a rule $A_{r,0} \rightarrow A_{r,1} \ldots A_{r,n_r}$ has been recognized.

We describe the parser using *parsing schemata* [1]. A *parsing schema* is a triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, with $\mathcal{H} = \{[a, i, i+1],\ a = w_i\}$ an initial set of items called *hypothesis* that encodes the sentence to be parsed[1], and $\mathcal{D}$ a set of *deduction*

⋆ Research partially supported by the Spanish Government under projects TIC2000-0370-C02-01 and HP2001-0044, and the Autonomous Government of Galicia under project PGIDT01PXI10506PN.

[1] The empty string, ε, is represented by the empty set of hypothesis, $\emptyset$. An input string $w_{1\ldots n}$, $n \geq 1$ is represented by $\{[w_1, 0, 1],\ [w_2, 1, 2],\ \ldots\ ,\ [w_n, n-1, n]\}$.

steps that allow new items to be derived from already known items. Deduction steps are of the form $\{\eta_1, \ldots, \eta_k \vdash \xi \,/ \mathit{conds}\}$, meaning that if all antecedents η_i are present and the conditions *conds* are satisfied, then the consequent ξ should be generated. In our case, $\mathcal{D} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}^{\text{Shift}} \cup \mathcal{D}^{\text{Sel}} \cup \mathcal{D}^{\text{Red}} \cup \mathcal{D}^{\text{Head}}$, where:

$$\mathcal{D}^{\text{Shift}} = \{[q, X, i, j] \vdash [q', \varepsilon, j, j+1] \Big/ \begin{array}{l} \exists\, [a, j, j+1] \in \mathcal{H} \\ \mathit{shift}_{q'} \in \mathit{action}(q, a) \end{array} \}$$
$$\mathcal{D}^{\text{Sel}} = \{[q, \varepsilon, i, j] \vdash [q, \nabla_{r,n_r}, j, j] \Big/ \begin{array}{l} \exists\, [a, j, j+1] \in \mathcal{H} \\ \mathit{reduce}_r \in \mathit{action}(q, a) \end{array} \}$$
$$\mathcal{D}^{\text{Red}} = \{[q, \nabla_{r,s}, k, j][q', \varepsilon, i, k] \vdash [q', \nabla_{r,s-1}, i, j] \,/ q' \in \mathit{reveal}(q)\}$$
$$\mathcal{D}^{\text{Init}} = \{\vdash [q_0, \varepsilon, 0, 0]\} \qquad \mathcal{D}^{\text{Head}} = \{\, [q, \nabla_{r,0}, i, j] \vdash [q', \varepsilon, i, j] \,/ q' \in \mathit{goto}(q, A_{r,0}) \,\}$$

with $q_0 \in \mathcal{Q}$ the initial state, and *action* and *goto* entries in the PDA tables. We say that $q' \in \mathit{reveal}(q)$ iff $\exists Y \in N \cup \Sigma$ such that $\mathit{shift}_q \in \mathit{action}(q', Y)$ or $q \in \mathit{goto}(q', Y)$, that is, when there exists a transition from q' to q in $\mathcal{A}$. A deduction step *Init* is in charge of starting the parsing process. The step *Shift* corresponds to pushing a terminal a onto the top of the stack when the action to be performed is a shift to state st'. A step *Sel* corresponds to pushing the ∇_{r,n_r} symbol onto the top of the stack in order to start the reduction of a rule r. The reduction of a rule of length $n_r > 0$ is performed by a set of n_r steps *Red*, each of them corresponding to a pop transition replacing the two elements $\nabla_{r,s}\, X_{r,s}$ placed on the top of the stack by the element $\nabla_{r,s-1}$. The reduction of a rule r is finished by a step *Head* corresponding to a swap transition that recognizes the top element $\nabla_{r,0}$ as equivalent to the left-hand side $A_{r,0}$ of that rule, and performs the corresponding change of state. The parse attains a worst case time (resp. space) complexity $\mathcal{O}(n^3)$ (resp. $\mathcal{O}(n^2)$). The input string has been recognized iff the final item $[q_f, \nabla_{0,0}, 0, n]$, $q_f \in \mathcal{Q}_f$ has been generated.

2 The error repair algorithm

We first assume that we are dealing with the first error detected, using the terminology introduced in [3]. We extend the item structure with the accumulated error counter e, resulting in items $[p, X, i, j, \mathrm{e}]$. Once the detection items have been fixed, we apply the set of deduction steps in error mode, $\mathcal{D}_{\text{error}}$, that follows:

$$\mathcal{D}^{\text{Shift}}_{\text{error}} = \{[q, X, i, j, 0] \vdash [q', a, j, j+1, 0] \Big/ \begin{array}{l} \exists [a, j, j+1] \in \mathcal{H} \\ \mathit{shift}_{q'} \in \mathit{action}(q, a) \end{array} \}$$
$$\mathcal{D}^{\text{Insert}}_{\text{error}} = \{[q, \varepsilon, i, j, 0] \vdash [q, \varepsilon, j, j, I(a)] \,\big/ \nexists\ \mathit{shift}_{q'} \in \mathit{action}(q, a)\}$$
$$\mathcal{D}^{\text{Delete}}_{\text{error}} = \{[q, \varepsilon, i, j, 0] \vdash [q, \varepsilon, j, j+1, D(w_i)]\}$$
$$\mathcal{D}^{\text{Replace}}_{\text{error}} = \{[q, \varepsilon, i, j, 0] \vdash [q, \epsilon, j, j+1, R(a)] \,\big/ \nexists\ \mathit{shift}_{q'} \in \mathit{action}(q, a)\}$$

This process continues until a repair covers both error and detection items. Once this has been performed on each detection item, we select the corresponding regional repairs and the parse goes back to standard mode. Error counters are summarized at the time of reductions by adding counters on popped items:

$$\mathcal{D}^{\text{Sel}}_{\text{error}} = \{[q, \varepsilon, i, j, e] \vdash [q, \nabla_{r,n_r}, j, j, e],\ \mathit{reduce}_r \in \mathit{action}(q, a)\}$$
$$\mathcal{D}^{\text{Red}}_{\text{error}} = \{[q, \nabla_{r,s}, k, j, e][q', \varepsilon, i, k, e'] \vdash [q', \nabla_{r,s-1}, i, j, e+e'] \,/ q' \in \mathit{reveal}(q)\}$$
$$\mathcal{D}^{\text{Head}}_{\text{error}} = \{\, [q, \nabla_{r,0}, i, j, e] \vdash [q', \varepsilon, i, j, e] \,/ q' \in \mathit{goto}(q, A_{r,0}) \,\}$$

with $\mathcal{D}^{\text{Init}}_{\text{error}} = \mathcal{D}^{\text{Init}}$. When the current repair is not the first one, it can modify a previous repair in order to avoid cascaded repairs by adding the cost of the new error hypotheses to profit from the experience gained from previous ones.

3 Asymptotic behavior

We consider the arithmetical expressions to illustrate this point. In the worst case, when the error repair zone becomes the entire input string, performance and cost are the same as for global error repair. We introduce two deterministic grammars, $\mathcal{G}_L$ and $\mathcal{G}_R$, and a non-deterministic one $\mathcal{G}_N$:

$\mathcal{G}_L$: E → E + T | T
T → (E) | number

$\mathcal{G}_R$: E → T + E | T
T → (E) | number

$\mathcal{G}_N$: S → S + S | (S) | number

As $\mathcal{G}_N$ contains a rule "$S \rightarrow S+S$", sentences of the form $b_1+b_2+\ldots+b_{i+1}$ have a number of exponential parses, which allows us to evaluate strongly ambiguous contexts. In the deterministic case, parses are built from the left-associative (resp. right-associative) interpretation for $\mathcal{G}_L$ (resp. $\mathcal{G}_R$), in order to estimate the impact of traversal orientation. Erroneous input strings are of the form: "$b_1 + \ldots + b_{i-1} + (b_i + \ldots + (b_{[n/3]} + b_{[n/3]+1} b_{[n/3]+2} + \ldots + b_\ell b_{\ell+1} + b_{\ell+2} + \ldots + b_n$", where $i \in \{[n/3], \ldots, 1\}$ and $\ell = 3[n/3]-2i+1$, with $[n/3]$ being the integer part of $n/3$. Given i, regional repairs are obtained by replacing tokens b_{3i} by closed brackets to obtain "$b_1+\ldots+b_{i-1}+(b_i+\ldots+(b_{[n/3]}+b_{[n/3]+1})+\ldots+b_\ell)+b_{\ell+2}+\ldots+b_n$".

3.1 The error repair region

We focus on the evolution of this region in relation to the location of the point of error, in opposition to static strategies associated to global repair approaches.

Location of points of detection. As is shown in the left-hand-side of Fig. 1, when we deal with global approach all input positions are points of detection. In the regional case, results depend on the grammar. So, although the number of points of detection grows with i because of the increase in the number of points of error, this number is higher for $\mathcal{G}_N$ (resp. $\mathcal{G}_R$). This is due to the right-associativity introduced by the rule "$S \rightarrow S + S$" (resp. "$E \rightarrow T + E$"), which generates a reduction for each "+" operator in the parsed prefix, illustrating the convergence of regional repairs with global ones. The reason for which results for $\mathcal{G}_N$ and $\mathcal{G}_R$ do not agree with results for the global case is because in regional repairs, operators "+" are not points of detection, while this is possible in a global one. The maximal number of points for $\mathcal{G}_N$ and $\mathcal{G}_R$, corresponding to the maximum size of the repair region as is shown in the right-hand-side of Fig. 1, is approximately half of those related to the global case.

Factors determining the size. We first focus on the case of $\mathcal{G}_R$ (resp. $\mathcal{G}_N$), profiting from the sequence of cascaded errors raised by the repair process exemplified. When the algorithm detects the first point of error at $b_{[n/3]+1}$, it takes

$b_{[n/3]}$ as point of detection and proposes as regional repair the replacement of $b_{[n/3]+2}$ by a closed bracket. Once this has been done, the algorithm returns the control to the parse until a new point of error is detected at $b_{[n/3]+3}$. In this case, "$(b_{[n/3]}$" is taken as the point of detection, which implies that we have moved back to a point previous to that proposed for the first error detected at $b_{[n/3]+1}$.

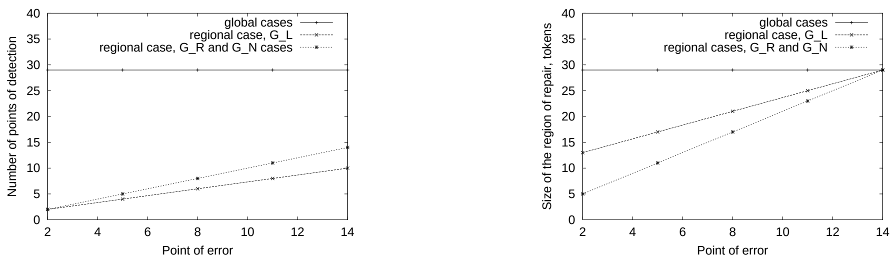


Fig. 1. Points of detection (resp. repair scope) *vs.* position of the point of error

More exactly, the algorithm asks whether the first regional repair applied was not optimal, taking into account the information about the parsing process now available. Perhaps the best solution for this first error would have been either to delete $b_{[n/3]+2}$ or to insert "+" between $b_{[n/3]+1}$ and $b_{[n/3]+2}$, which at that moment were not considered because the reductions defining the scopes of these repairs were not minimal in relation to that of the regional repair finally applied.

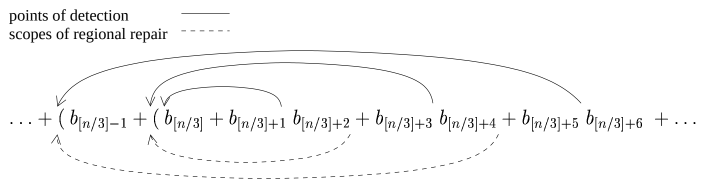

points of detection
scopes of regional repair
$\ldots + (b_{[n/3]-1} + (b_{[n/3]} + b_{[n/3]+1}\ b_{[n/3]+2} + b_{[n/3]+3}\ b_{[n/3]+4} + b_{[n/3]+5}\ b_{[n/3]+6}\ + \ldots$

Fig. 2. An example on cascaded errors for $\mathcal{G}_R$ and $\mathcal{G}_N$

We then repeat the same steps as in the first case, proposing the regional repair that replaces $b_{[n/3]+4}$ by a closed bracket, followed by a shift over "+". So, the frontier of the new error repair region is "$(b_{[n/3]-1} + (b_{[n/3]} + b_{[n/3]+1}) + b_{[n/3]+3})$", which includes the scope of the previous regional repair, whose frontier was "$(b_{[n/3]} + b_{[n/3]+1})$"; as is shown in Fig. 2. The algorithm continues to apply the previous process for all $i \in \{[n/3], \ldots, 1\}$, until the size of the repair region extends to the whole original input string, as is shown in Fig. 1.

In the case of $\mathcal{G}_L$, the size of the repair region grows with the position of the point of error, b_l, $l \in \{[n/3]+1, [n/3]+3, \ldots, 3[n/3]-2i+1\}$. This behavior is also a consequence of the presence of cascaded errors, as is shown in Fig. 3. In

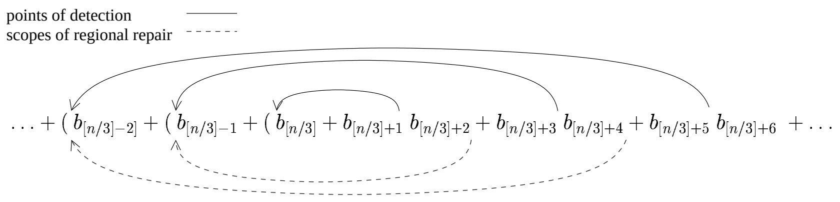

points of detection
scopes of regional repair
$\ldots + (b_{[n/3]-2} + (b_{[n/3]-1} + (b_{[n/3]} + b_{[n/3]+1}\ b_{[n/3]+2} + b_{[n/3]+3}\ b_{[n/3]+4} + b_{[n/3]+5}\ b_{[n/3]+6} + \ldots$

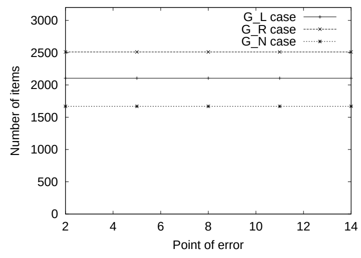
Fig. 3. An example on cascaded errors for $\mathcal{G}_L$

comparison with previous results, when the algorithm detects the first point of error at $b_{[n/3]+1}$, it takes $b_{[n/3]}$ as the point of detection and proposes as regional repair the replacement of $b_{[n/3]+2}$ by a closed bracket, as was the case for $\mathcal{G}_R$ and $\mathcal{G}_N$. As for $\mathcal{G}_R$, the rule providing the reduction is "$F \rightarrow (E)$". However, in this case, this reduction does not characterize a regional repair because it is followed by a chain of reductions in $\mathcal{G}_L$ previous to the next shift action, and not by an immediate shift action. These reductions are given by the rules "$T \rightarrow F$" and "$E \rightarrow E + T$", and the frontier of the repair region associated to this first error is "$b_{[n/3]-1} + (b_{[n/3]} + b_{[n/3]+1})$". Applying a similar reasoning to the next errors in the input string, we conclude that the sizes of the error repair regions are now larger, as is shown in the right-hand-side of Fig. 1, which also illustrates the asymptotic convergence with global repairs. So, the repair region when the last point of error, b_ℓ, is to the right of the input, includes the total input string.

3.2 The computational cost

Items are the basis for showing the computational behavior of our proposal. The cost of the algorithm is, in the worst case, given by the cost of global error repair approaches, due to asymptotic equivalence between regional and global repairs. Our aim is to focus on the dependency of grammar design.

The case of global repairs. The generation is illustrated in the left-hand-side of Fig. 4. In all cases the number of items generated remains constant because it is only dependent on the length of the input string. These strategies expend equal effort on all parts of the program, including areas without errors. The situation of the curve for $\mathcal{G}_N$ is justified by subsumption phenomena between items generated by the parse process. In effect, the compact representation of $\mathcal{G}_N$ in relation to $\mathcal{G}_R$ and $\mathcal{G}_L$ in terms of the number of rules facilitates the application of such mechanisms. The greater cost of $\mathcal{G}_R$ in relation to $\mathcal{G}_L$ is due to the introduction of a non-determinism by the error hypotheses. When a token "b_l" is shifted in $\mathcal{G}_L$, the only PDA action available is the reduction of all or part of the analyzed prefix, since we can assume that the lookahead is "+", ")" or $\dashv$. For $\mathcal{G}_R$, two possibilities exist. When the lookahead is a "+", a shift takes place; but when it is a ")" or $\dashv$, a reduction is made. Thus, in the case or $\mathcal{G}_R$, the error repair algorithm introduces a larger number of parse conflicts, and hence items.



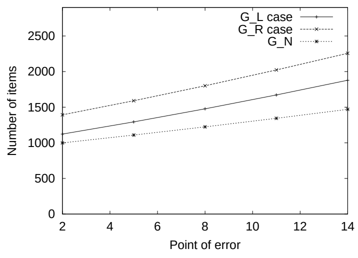


Fig. 4. Items for global (resp. regional) repairs *vs.* position of the point of error

The case of regional repairs. The generation is discussed with reference to the right-hand-side of Fig. 4. The general distribution of curves for $\mathcal{G}_R$, $\mathcal{G}_L$ and $\mathcal{G}_N$ is the same as mentioned for global repairs and it can be justified in the same manner. It is of interest to compare the results for global and regional repairs. So, Fig. 5 shows the number of items whose generation has been saved going from global to regional repair, illustrating the asymptotic convergence. The difference in terms of items generated is minor when the point of error is situated to the right of the input string, enlarging the repair region. This difference does not reach zero, which is in apparent contradiction with the above-mentioned convergence. We should take here into account that even though the size of the repair region can be the same for both global and regional repairs, the latter are not forced to apply the error hypotheses on all the error parse branches.

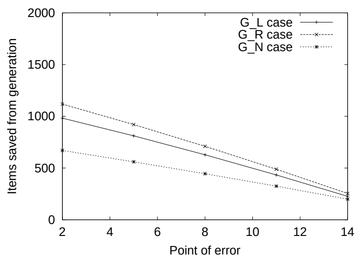


Fig. 5. Saved items from global to regional repair *vs.* position of the point of error

References

1. K. Sikkel. *Parsing Schemata.* PhD thesis, Univ. of Twente, The Netherlands, 1993.
2. M. Vilares. *Efficient Incremental Parsing for Context-Free Languages.* PhD thesis, University of Nice. ISBN 2-7261-0768-0, France, 1992.
3. M. Vilares, V.M. Darriba, and F.J. Ribadas. Regional least-cost error repair. In S. Yu and A. Păun, editors, *Implementation and Application of Automata*, volume 2088 of *LNCS*, pages 293–301. Springer-Verlag, Berlin-Heidelberg-New York, 2001.