to retain the memo table — or as much of it as possible — and enable the intermediate parse results to be used across multiple invocations of the parser.

A standard packrat parser can be modeled by the function:

$$\textsc{Parse} : (G, s) \to T$$

where G is a grammar, s is an input string, and T is a parse tree (or the special value Fail). Similarly, an *incremental* packrat parser can be modeled by:

$$\textsc{Parse} : (G, s, M) \to (M', T)$$

where M and M' are memo tables that (ideally) share some structure. Together, G, s, and M comprise the state or *environment* of the parser, which is carried forward in an explicit environment-passing scheme.

When the input is modified, it will also be necessary to modify the memo table to account for the changes. For this, we introduce a new function:

$$\textsc{ApplyEdit} : (s, M, e) \to (s', M')$$

where e is an *edit operation* consisting of:

- a start position p_s,
- an end position p_e, and
- a replacement string r.

In the general case, applying e means replacing the characters in the interval $[p_s, p_e)$ with the characters in r, which may be of any length. Under this definition, insertion (when $p_s = p_e$) and deletion (when r is empty) are special cases of replacement.

After modifying s to produce s', ApplyEdit produces a new memo table M' which shares as many entries as possible with M. The details of this procedure form the core of our algorithm, which will be described below.

The remainder of this section describes the primary contribution of this paper, namely:

(a) a strategy for detecting which memo table entries are affected by an edit (Section 3.1), allowing the other entries to be reused on the next parse

(b) a modification to the representation of the memo table (Section 3.2) which allows ApplyEdit to efficiently compute M', and

(c) an optimization for our algorithm that allows it to operate more efficiently on large inputs with real-world grammars (Section 3.4).

3.1 Detecting Affected Memo Table Entries

The basic correctness criterion of an incremental parser is that it must produce the same result as parsing from scratch. We say that a memo table entry is *affected* by an edit if, after editing, the entry must be modified or deleted to preserve correctness.

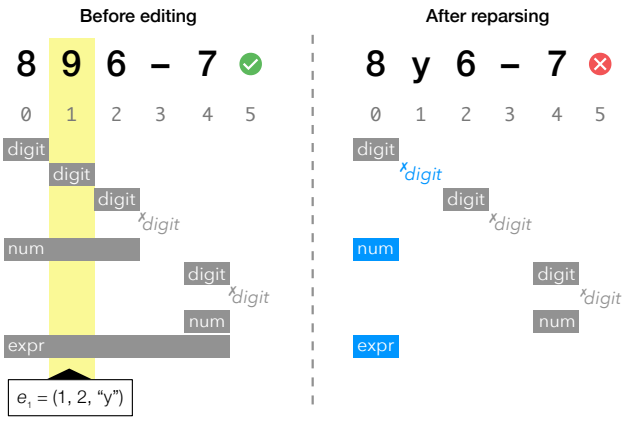

Figure 2. Memo table contents before and after applying edit operation e_1. *Left*, Using the basic overlap rule, ApplyEdit will invalidate the three memo table entries that are affected by the edit. *Right*, the entries shown in blue are new, representing additional work that was done in reparsing the modified input.

> To better explain the technique presented in this paper, we have implemented an interactive memo table visualization, available at https://ohmlang.github.io/sle17/. Readers are encouraged to use it to follow along with the examples in this section.

In explaining our strategy for detecting affected memo table entries, we will first discuss a subset of possible edit operations: replacement operations that do not change the length of the input. After applying the edit to the input string, the next step is to determine which (if any) of the entries are affected by the edit.

For example, consider the input "896–7" which was used in Section 1, and an edit operation $e_1 = (1, 2, \text{"}y\text{"})$ that replaces the "9" with a "y". After applying the operation, the new input string is "8y6–7", as shown in Figure 2.

Previously, the `digit` rule succeeded at position 1; but now, in a from-scratch parse of the modified input, it will fail (since "y" is not a digit). To preserve correctness, we can remove the entry for `digit` at position 1 from the memo table, ensuring that the parser will re-attempt `digit` at that position.

3.1.1 Basic "Overlap Rule"

In general, we can observe that if the extent of a memo table entry overlaps with an edit operation e, then the entry is possibly affected by e. Depending on the edit, it may *not* be affected (e.g., if the replacement text is the same as the original text), but we adopt a conservative approach: any memo table entry overlapping the edit is considered *invalid* and will be removed from the memo table.