PEGMATCH: Parsing Expression Grammars for TeX

Jianrui Lyu (tolvjr@163.com)
https://github.com/lvjr/pegmatch

Version 2025B (2025-02-16)

Contents

Chapter 1

Package Interfaces

1.1 Introduction

The `pegmatch` package ports PEG (Parsing Expression Grammars)[1] to TeX. Following the design in LPEG (Parsing Expression Grammars for Lua),[2] it defines patterns as LaTeX3 variables, and offers several operators to compose patterns.

In general, PEG matching is much more powerful than RE (Regular Expressions) matching. At this time, `pegmatch` package only supports TeX strings.[3] Also it is still in experimental status, hence some interfaces may change in future releases.

1.2 The first example

The following is the first example:

```
\NewSpeg\lMyTestSpeg
\SetSpeg\lMyTestSpeg{\SpegP{abc}}
\IfSpegMatchTF\lMyTestSpeg{a}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{ab}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{abc}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{abcd}{T}{F}
```

F F T T

In this example, we use `\NewSpeg` to create a new `speg` variable `\lMyTestSpeg`, and use `\SetSpeg` to set the variable with a pattern expression, then use `\IfSpegMatchTF` to match it against different subject strings. The pattern `\SpegP{abc}` matches the string `abc` literally.

1.3 Basic commands

This package provides the following commands for creating and matching `speg` patterns:

Table 1.1: Basic commands

Command	Description
`\NewSpeg#1`	create `speg` variable `#1`
`\SetSpeg#1{#2}`	set `speg` variable `#1` with `speg` expresssion `#2`
`\IfSpegMatchT#1{#2}{#3}`	match `#1` against string `#2`, then run code `#3` if the match succeeds
`\IfSpegMatchF#1{#2}{#3}`	match `#1` against string `#2`, then run code `#3` if the match fails

Continued on next page

---

[1] See Parsing Expression Grammars page: https://bford.info/packrat/.

[2] See Parsing Expression Grammars for Lua page: https://www.inf.puc-rio.br/~roberto/lpeg/.

[3] I started to write it for my `codehigh` package to get rid of `l3regex` dependency.

Table 1.1: Basic commands (Continued)

Command	Description
`\IfSpegMatchTF#1{#2}{#3}{#4}`	match `#1` against string `#2`, then run code `#3` if the match succeeds, othewise run code `#4`
`\IfSpegExtractT#1{#2}#3{#4}`	match `#1` against string `#2`, then store captures in `#3` and run code `#4` if the match succeeds
`\IfSpegExtractF#1{#2}#3{#4}`	match `#1` against string `#2`, then clear `#3` and run code `#4` if the match fails
`\IfSpegExtractTF#1{#2}#3{#4}{#5}`	match `#1` against string `#2`, then store captures in `#3` and run code `#4` if the match succeeds, othewise clear `#3` and run code `#5`

1.4 Scratch variables

There are two predefined scratch `speg` variables for setting `speg` patterns: `\lTmpaSpeg` and `\lTmpbSpeg`. Also there are two predefined scratch `seq` variables for storing captures (see Section 1.8): `\lSpegTmpaSeq` and `\lSpegTmpbSeq`.

1.5 Primitive patterns

This package provides the following commands for making primitive patterns:

Table 1.2: Primitive patterns

Pattern	Description
`\SpegP{<string>}`	match `<string>` literally
`\SpegQ{<n>}`	match exactly `<n>` characters
`\SpegR{<X><Y><x><y>}`	match any character between `<X>` and `<Y>` or between `<x>` and `<y>`
`\SpegS{<string>}`	match any character in `<string>`

The following examples demonstrate pattern matching with other primitive patterns:

```
\SetSpeg\lMyTestSpeg{\SpegQ{2}}
\IfSpegMatchTF\lMyTestSpeg{u}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{vw}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{xyz}{T}{F}
```

F T T

```
\SetSpeg\lMyTestSpeg{\SpegR{AZ}}
\IfSpegMatchTF\lMyTestSpeg{Qq}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{q1}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{1Q}{T}{F}
\SetSpeg\lMyTestSpeg{\SpegR{AZaz}}
\IfSpegMatchTF\lMyTestSpeg{Qq}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{q1}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{1Q}{T}{F}
```

T F F T T F

```
\SetSpeg\lMyTestSpeg{\SpegS{world}}
\IfSpegMatchTF\lMyTestSpeg{one}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{two}{T}{F}
```

T F

By default, PEG always starts at the first character. Since both `\SpegR` and `\SpegS` match only one letter, both last commands in previous two examples give `F`.

1.6 Pattern operators

This package provides the following pattern operators for composing patterns:

Table 1.3: Pattern operators

Operator	Precedence	Description
`patt1/patt2`	1 (choice)	match `patt1` or `patt2` (ordered choice)
`patt1*patt2`	2 (concat)	match `patt1` followed by `patt2`
`!patt`	3 (not predicate)	match only if `patt` does not match, and consume no input
`&patt`	3 (and predicate)	match `patt` but consume no input
`patt^{<n>}`	4 (repeat)	match at least `<n>` ($n \geq 0$) repetitions of `patt`
`patt^{-<n>}`	4 (repeat)	match at most `<n>` ($n > 0$) repetitions of `patt`
`{patt expr}`	5 (group)	match `patt expr` (pattern expression)

With `!` and `*` operators, we can create negative character sets:

```
\SetSpeg\lMyTestSpeg{!\SpegR{09} * \SpegQ{1}}
\IfSpegMatchTF\lMyTestSpeg{A}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{5}{T}{F}
\SetSpeg\lMyTestSpeg{!\SpegS{abc} * \SpegQ{1}}
\IfSpegMatchTF\lMyTestSpeg{B}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{b}{T}{F}
```

```
T F  T F
```

With `^` operator, we can match words:

```
\SetSpeg\lMyTestSpeg{\SpegR{AZaz} ^ {1}}
\IfSpegMatchTF\lMyTestSpeg{HELLO}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{world}{T}{F}
\IfSpegMatchTF\lMyTestSpeg{ text }{T}{F}
\IfSpegMatchTF\lMyTestSpeg{(text)}{T}{F}
```

```
T T F F
```

In fact, `patt^{-1}` is similar to `expr?`, `patt^0` is similar to `expr*`, and `patt^1` is similar to `expr+` in regular expression matching.

1.7 Pattern variables

In using `\SetSpeg` command to set a `speg` variable with a pattern expression, you can use other `speg` variables. For example:

```
\SetSpeg\lTmpaSpeg{\SpegR{AZ} / \SpegR{az}}
\SetSpeg\lTmpbSpeg{\SpegS{135} * \lTmpaSpeg}
\IfSpegMatchTF\lTmpbSpeg{2ab}{[T]}{[F]}
\IfSpegMatchTF\lTmpbSpeg{3ab}{[T]}{[F]}
\SetSpeg\lTmpbSpeg{\SpegS{135} * \lTmpaSpeg^{3}}
\IfSpegMatchTF\lTmpbSpeg{3ab}{[T]}{[F]}
\IfSpegMatchTF\lTmpbSpeg{3abcd}{[T]}{[F]}
```

```
[F] [T]  [F] [T]
```

By using another recursive pattern, we can make `speg` find a pattern anywhere in a string. The following example demonstrates how to match a word with at least three letters inside a string:

```
\NewSpeg\lMyWordSpeg
\NewSpeg\lMyAnywhereSpeg
\SetSpeg\lMyWordSpeg{\SpegR{AZaz}^{3}}
\SetSpeg\lMyAnywhereSpeg{\lMyWordSpeg / \SpegQ{1} * \lMyAnywhereSpeg}
\IfSpegMatchTF\lMyAnywhereSpeg{foo bar}{[T]}{[F]}
\IfSpegMatchTF\lMyAnywhereSpeg{fo bar}{[T]}{[F]}
\IfSpegMatchTF\lMyAnywhereSpeg{123 ba}{[T]}{[F]}
\IfSpegMatchTF\lMyAnywhereSpeg{123 bar}{[T]}{[F]}
```

```
[T] [T] [F] [T]
```

In this example, `\lMyAnywhereSpeg` tries to match `\lMyWordSpeg`, skipping one letter and tries again if it fails.

1.8 Capture patterns

This package provides the following commands for making capture patterns:

Table 1.4: Primitive patterns

Pattern	Name	Description
`\SpegC{<patt>}`	simple capture	capture the match for `<patt>`
`\SpegCp`	position capture	capture current position

Position capture `\SpegCp` must be concatenated with other patterns (by using `*` operator):

```
\SetSpeg\lTmpaSpeg{\SpegCp * \SpegR{az}^{1} * \SpegCp * \SpegR{09}^{1} * \SpegCp}
\IfSpegExtractTF\lTmpaSpeg{12ab}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
\IfSpegExtractTF\lTmpaSpeg{ab12}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
\IfSpegExtractTF\lTmpaSpeg{abcd12345}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
```

```
Failed [1][3][5] [1][5][10]
```

In this example, we use `\IfSpegExtractTF` command to extract all captures, which are stored in the `seq` variable (`\lSpegTmpaSeq`) specified by the third argument. Then we use `\MapSpegSeqInline` command to print each capture.

If you want to capture the substrings, you can modified the above example as follows:

```
\SetSpeg\lTmpaSpeg{\SpegC{\SpegR{az}^{1}} * \SpegC{\SpegR{09}^{1}}}
\IfSpegExtractTF\lTmpaSpeg{12ab}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
\IfSpegExtractTF\lTmpaSpeg{ab12}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
\IfSpegExtractTF\lTmpaSpeg{abcd12345}\lSpegTmpaSeq{%
  \MapSpegSeqInline\lSpegTmpaSeq{[#1]}%
}{Failed}
```

```
Failed [ab][12] [abcd][12345]
```

Chapter 2

The Source Code

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesExplPackage{pegmatch}{2025-02-16}{v2025B}
  {Parsing Expression Grammars for TeX}

\cs_generate_variant:Nn \iow_log:n {V}
\cs_generate_variant:Nn \str_range:nnn {nne}
\cs_generate_variant:Nn \tl_analysis_map_inline:nn {e}

\prg_generate_conditional_variant:Nnn \int_compare:nNn {eN} {p,TF}
\prg_generate_conditional_variant:Nnn \str_if_eq:nn {en} {TF}
\prg_generate_conditional_variant:Nnn \str_if_in:nn {nV} {TF}
\prg_generate_conditional_variant:Nnn \tl_if_head_is_group:n {V} {TF}
\prg_generate_conditional_variant:Nnn \tl_if_head_is_space:n {v} {TF}

%% Every speg variable starts with scan mark \s__speg.
\scan_new:N \s__speg

\cs_new_protected:Npn \speg_new:N #1
  {
    \tl_new:N #1
    \tl_set:Nn #1 { \s__speg }
  }
\cs_set_eq:NN \NewSpeg \speg_new:N

\speg_new:N \lTmpaSpeg
\speg_new:N \lTmpbSpeg
\speg_new:N \gTmpaSpeg
\speg_new:N \gTmpbSpeg

\seq_new:N \lSpegTmpaSeq
\seq_new:N \lSpegTmpbSeq
\cs_set_eq:NN \MapSpegSeqInline \seq_map_inline:Nn
```

2.1 Set speg variables

```
\int_new:N \g__speg_prg_map_int

%% Split tl #1 into items separated by tl #2, and pass each item to code #3.
%% Braces around each item are kept but spaces around each item are removed.
\cs_new_protected:Npn \__speg_tl_split_map_inline:nnn #1 #2 #3
  {
    \int_gincr:N \g__speg_prg_map_int
    \cs_gset_protected:cpn
      {__speg_map_ \int_use:N \g__speg_prg_map_int :w} ##1 {#3}
    \__speg_tl_split_map_function:nnc
      {#1} {#2} {__speg_map_ \int_use:N \g__speg_prg_map_int :w}
    \int_gdecr:N \g__speg_prg_map_int
  }
\cs_generate_variant:Nn \__speg_tl_split_map_inline:nnn {V}

%% Split tl #1 into items separated by tl #2, and pass each item to function #3.
%% Braces around each item are kept but spaces around each item are removed.
%% We insert \prg_do_nothing: before each item to avoid losing outermost braces.
\cs_new_protected:Npn \__speg_tl_split_map_function:nnN #1 #2 #3
  {
    \cs_set_protected:cpn
      { __speg_tl_split_map_ \int_use:N \g__speg_prg_map_int _aux:Nw } ##1 ##2 #2
      {
        \tl_if_eq:nnF {\prg_do_nothing: \c_novalue_tl} {##2}
          {
            \exp_args:Ne ##1 {\tl_trim_spaces:o {##2}}
            \use:c {__speg_tl_split_map_\int_use:N \g__speg_prg_map_int _aux:Nw}
              ##1 \prg_do_nothing:
          }
      }
    \use:c {__speg_tl_split_map_ \int_use:N \g__speg_prg_map_int _aux:Nw}
      #3 \prg_do_nothing: #1 #2 \c_novalue_tl #2
  }
\cs_generate_variant:Nn \__speg_tl_split_map_function:nnN {V, nnc}

\tl_new:N \l__speg_result_tl

\cs_new_protected:Npn \speg_set:Nn #1 #2
  {
    \__speg_tracing:nn {set} { \tl_log:n {Input=#2} }
    \tl_clear:N \l__speg_result_tl
    \__speg_parse_expr:n {#2}
    \tl_set:Ne #1 { \s__speg \exp_not:V \l__speg_result_tl }
    \__speg_tracing:nn {set} { \speg_log:N #1 }
  }
\cs_set_eq:NN \SetSpeg \speg_set:Nn

\cs_new_protected:Npn \__speg_parse_expr:n #1
  {
    \__speg_parse_choice:n {#1}
    %\tl_log:N \l__speg_result_tl
    \tl_set:Ne \l__speg_result_tl { \l__speg_result_tl }
  }

\tl_const:Nn \c__speg_left_brace_tl { \exp_after:wN {\if_false:} \fi: }
\tl_const:Nn \c__speg_right_brace_tl { \if_false: {\fi:} }

%% parse choice operator
\cs_new_protected:Npn \__speg_parse_choice:n #1
  {
    \tl_if_in:nnTF {#1} {/}
      {
        \tl_put_right:Nn \l__speg_result_tl {\SpegChoice \c__speg_left_brace_tl}
        \__speg_tl_split_map_inline:nnn {#1} {/}
          {
            \tl_put_right:Nn \l__speg_result_tl {\c__speg_left_brace_tl}
            \__speg_parse_concat:n {##1}
            \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
          }
        \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
      }
      { \__speg_parse_concat:n {#1} }
  }
\cs_generate_variant:Nn \__speg_parse_choice:n {e}

%% parse concat operator
\cs_new_protected:Npn \__speg_parse_concat:n #1
  {
    \tl_if_in:nnTF {#1} {*}
      {
        \tl_put_right:Nn \l__speg_result_tl {\SpegConcat \c__speg_left_brace_tl}
        \__speg_tl_split_map_inline:nnn {#1} {*}
          {
            \tl_put_right:Nn \l__speg_result_tl {\c__speg_left_brace_tl}
            \__speg_parse_predicate:n {##1}
            \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
          }
        \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
      }
      { \__speg_parse_predicate:n {#1} }
  }

\tl_new:N \l__speg_predicate_head_tl
\tl_new:N \l__speg_predicate_tail_tl
\tl_const:Nn \c__speg_not_tl {!}
\tl_const:Nn \c__speg_and_tl {&}

%% parse "not predicate" and "and predicate" operators
\cs_new_protected:Npn \__speg_parse_predicate:n #1
  {
    \tl_set:Ne \l__speg_predicate_head_tl {\tl_head:n {#1}}
    \tl_set:Ne \l__speg_predicate_tail_tl {\tl_trim_spaces:e {\tl_tail:n {#1}}}
    \tl_case:NnTF \l__speg_predicate_head_tl
      {
        \c__speg_not_tl
          {
            \tl_put_right:Nn \l__speg_result_tl {\SpegNot}
          }
        \c__speg_and_tl
          {
            \tl_put_right:Nn \l__speg_result_tl {\SpegAnd}
          }
      }
      {
        \tl_put_right:Nn \l__speg_result_tl {\c__speg_left_brace_tl}
        \__speg_parse_repeat:V \l__speg_predicate_tail_tl
        \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
      }
      { \__speg_parse_repeat:n {#1} }
  }

\quark_new:N \q__speg_stop

%% parse repeat operator
\cs_new_protected:Npn \__speg_parse_repeat:n #1
```

```
  {
    \tl_if_in:nnTF {#1} {^}
      {
        %% we prepend \prg_do_nothing: to avoid losing outermost braces
        \__speg_parse_repeat_aux:wnw \prg_do_nothing: #1 \q__speg_stop
      }
      { \__speg_parse_or_store_atom:e { \tl_trim_spaces:n {#1} } }
  }
\cs_generate_variant:Nn \__speg_parse_repeat:n {V}

\msg_new:nnn {speg} {invalid-repeat-arguments}
  {Invalid ~ arguments ~ '#1' ~ for ~ \token_to_str:N \SpegRepeat!}

\cs_new_protected:Npn \__speg_parse_repeat_aux:wnw #1 ^ #2 #3 \q__speg_stop
  {
    %% remove \prg_do_nothing: at the beginning of #1
    \__speg_parse_repeat_real:eee { \tl_trim_spaces:e { \tl_tail:n {#1} } }
      { \tl_trim_spaces:n {#2} } { \tl_trim_spaces:n {#3} }
  }

\cs_new_protected:Npn \__speg_parse_repeat_real:nnn #1 #2 #3
  {
    \tl_if_empty:eF {#3} {\msg_error:nne {speg} {invalid-repeat-arguments} {#3}}
    \tl_put_right:Nn \l__speg_result_tl {\SpegRepeat \c__speg_left_brace_tl}
    \__speg_parse_or_store_atom:n {#1}
    \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl {#2}}
  }
\cs_generate_variant:Nn \__speg_parse_repeat_real:nnn {eee}

%% We need to put speg atom (especially speg variable) inside \exp_not:n,
%% to protect it from e-expansion in \__speg_parse_expr:n function.
\cs_new_protected:Npn \__speg_parse_or_store_atom:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \__speg_parse_choice:n #1 }
      {
        \tl_if_head_eq_meaning:nNTF {#1} \SpegC
          {
            \tl_put_right:Nn \l__speg_result_tl {\SpegC \c__speg_left_brace_tl}
            \__speg_parse_choice:e { \tl_tail:n {#1} }
            \tl_put_right:Nn \l__speg_result_tl {\c__speg_right_brace_tl}
          }
          { \tl_put_right:Nn \l__speg_result_tl { \exp_not:n {#1} } }
      }
  }
\cs_generate_variant:Nn \__speg_parse_or_store_atom:n {e}
```

2.2 Log speg variables

```
\int_new:N \l__speg_log_indent_int
\tl_new:N \l__speg_log_line_tl
\seq_new:N \l__speg_log_seq

\cs_new_protected:Npn \speg_log:N #1
  {
    \int_zero:N \l__speg_log_indent_int
    \seq_clear:N \l__speg_log_seq
    \tl_clear:N \l__speg_log_line_tl
    %% Remove leading \s__speg in speg variable #1
    \tl_analysis_map_inline:en { \tl_tail:N #1 }
      {
        \int_case:nnF {"##3} % convert hexadecimal digit to integer
          {
            {1} % begin-group
            {
              \tl_if_empty:NF \l__speg_log_line_tl {\__speg_log_line:}
              \tl_set_eq:NN \l__speg_log_line_tl \c_left_brace_str
              \__speg_log_line:
              \int_incr:N \l__speg_log_indent_int
            }
            {2} % end-group
            {
              \tl_if_empty:NF \l__speg_log_line_tl {\__speg_log_line:}
              \int_decr:N \l__speg_log_indent_int
              \tl_set_eq:NN \l__speg_log_line_tl \c_right_brace_str
              \__speg_log_line:
            }
          }
          {\tl_put_right:Ne \l__speg_log_line_tl {##1}}
      }
    \iow_log:e
      {
        >~Compiled~speg~variable~\token_to_str:N #1:^^J
        \seq_use:Nn \l__speg_log_seq {^^J}
      }
  }
\cs_set_eq:NN \LogSpeg \speg_log:N

\cs_new_protected:Npn \__speg_log_line:
  {
    \seq_put_right:Ne \l__speg_log_seq
      {
        \prg_replicate:nn {\l__speg_log_indent_int * 2} {~}
        \exp_not:V \l__speg_log_line_tl
      }
    \tl_clear:N \l__speg_log_line_tl
  }
```

2.3 Match speg variables

```
\int_new:N \g__speg_match_level_int
\int_new:N \g__speg_match_level_max_int
\prop_new_linked:N \g__speg_match_index_prop

\cs_new_protected:Npn \__speg_match_gzero_level:
  {
    \int_gzero:N \g__speg_match_level_int
  }

\cs_new_protected:Npn \__speg_match_gincr_level:
  {
    \int_gincr:N \g__speg_match_level_int
    \int_compare:nNnT
      { \g__speg_match_level_int } > { \g__speg_match_level_max_int }
      {
        \int_gset_eq:NN \g__speg_match_level_max_int \g__speg_match_level_int
        \bool_new:c
          { g__speg_match_ \int_use:N \g__speg_match_level_int _bool }
        \str_new:c { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
      }
  }

\cs_new_protected:Npn \__speg_match_gdecr_level:
  {
    \int_gdecr:N \g__speg_match_level_int
  }

\cs_new_protected:Npn \__speg_match_gset_true:
  {
    \bool_gset_true:c
      { g__speg_match_ \int_use:N \g__speg_match_level_int _bool }
  }

\cs_new_protected:Npn \__speg_match_gset_false:
  {
    \bool_gset_false:c
      { g__speg_match_ \int_use:N \g__speg_match_level_int _bool }
  }

\cs_new:Npn \__speg_match_use_status:
  {
    \bool_if:cTF
      { g__speg_match_ \int_use:N \g__speg_match_level_int _bool }
      {true} {false}
  }

\prg_new_conditional:Npnn \__speg_match_if_success: { p, T, F, TF }
  {
    \bool_if:cTF
      { g__speg_match_ \int_use:N \g__speg_match_level_int _bool }
      { \prg_return_true: } { \prg_return_false: }
  }

\prg_new_conditional:Npnn \__speg_match_if_sub_success: { p, T, F, TF }
  {
    \bool_if:cTF
      { g__speg_match_ \int_eval:n { \g__speg_match_level_int + 1 } _bool }
      { \prg_return_true: } { \prg_return_false: }
  }

\cs_new_protected:Npn \__speg_match_gincr_index:
  {
    \prop_gput:Nee \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int }
      {
        \int_eval:n
          {
            \prop_item:Ne \g__speg_match_index_prop
              { \int_use:N \g__speg_match_level_int }
            + 1
          }
      }
  }
```

```
\cs_new_protected:Npn \__speg_match_gdecr_index:
  {
    \prop_gput:Nee \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int }
      {
        \int_eval:n
          {
            \prop_item:Ne \g__speg_match_index_prop
              { \int_use:N \g__speg_match_level_int }
            - 1
          }
      }
  }

\cs_new_protected:Npn \__speg_match_gset_index:n #1
  {
    \prop_gput:Nen \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int } {#1}
  }
\cs_generate_variant:Nn \__speg_match_gset_index:n {e}

\cs_new_protected:Npn \__speg_match_gset_eq_index:
  {
    \prop_gput:Nee \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int }
      {
        \prop_item:Ne \g__speg_match_index_prop
          { \int_eval:n { \g__speg_match_level_int + 1 } }
      }
  }

\cs_new_protected:Npn \__speg_match_update_index:
  {
    \prop_gput:Nee \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int }
      {
        \int_eval:n
          {
            \prop_item:Ne \g__speg_match_index_prop
              { \int_use:N \g__speg_match_level_int }
            +
            \prop_item:Ne \g__speg_match_index_prop
              { \int_eval:n { \g__speg_match_level_int + 1 } }
            - 1
          }
      }
  }

\cs_new:Npn \__speg_match_use_index:
  {
    \prop_item:Ne \g__speg_match_index_prop
      { \int_use:N \g__speg_match_level_int }
  }

\cs_new:Npn \__speg_match_use_sub_index:
  {
    \prop_item:Ne \g__speg_match_index_prop
      { \int_eval:n { \g__speg_match_level_int + 1 } }
  }
```

```
\cs_new:Npn \__speg_match_use_index:n #1
  {
    \prop_item:Nn \g__speg_match_index_prop {#1}
  }

\cs_new_protected:Npn \__speg_match_get_position:N #1
  {
    \int_set:Nn #1 {1}
    \int_step_inline:nnn {2} { \g__speg_match_level_int }
      {
        \int_add:Nn #1
          { \prop_item:Nn \g__speg_match_index_prop {##1} - 1 }
      }
    %\int_log:N #1
  }

\cs_new_protected:Npn \__speg_match_gclear_subject:
  {
    \str_gclear:c
      { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
  }

\cs_new_protected:Npn \__speg_match_gset_subject:n #1
  {
    \str_gset:cn
      { g__speg_match_ \int_use:N \g__speg_match_level_int _str } {#1}
  }
\cs_generate_variant:Nn \__speg_match_gset_subject:n {e}

\cs_new_protected:Npn \__speg_match_gset_eq_subject:
  {
    \str_gset_eq:cc
      { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
      { g__speg_match_ \int_eval:n { \g__speg_match_level_int + 1 } _str }
  }

\cs_new_protected:Npn \__speg_match_get_capture:nN #1 #2
  {
    \str_set:Ne #2
      {
        \str_range:nne {#1} {1}
          { \int_eval:n { \__speg_match_use_sub_index: - 1 } }
      }
  }

\cs_new:Npn \__speg_match_use_subject:
  {
    \str_use:c { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
  }

\cs_new_protected:Npn \__speg_match_apply_subject:n #1
  {
    \__speg_match_apply_subject_aux:vn
      { g__speg_match_ \int_use:N \g__speg_match_level_int _str } {#1}
  }

\cs_new_protected:Npn \__speg_match_apply_subject_aux:nn #1 #2
  {
    #2 {#1}
  }
\cs_generate_variant:Nn \__speg_match_apply_subject_aux:nn {vn}

\prg_new_conditional:Npnn \__speg_match_if_head_is_space: {TF}
  {
    \tl_if_head_is_space:vTF
      { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
      { \prg_return_true: } { \prg_return_false: }
  }

\cs_new:Npn \__speg_match_head_subject:
  {
    \tl_head:v { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
  }

\cs_new:Npn \__speg_match_tail_subject:
  {
    \tl_tail:v { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
  }

\exp_last_unbraced:NNo \cs_new:Npn \__speg_gobble_space:w \c_space_tl { }

\cs_new:Npn \__speg_match_head_subject_keep_space:
  {
    \__speg_match_if_head_is_space:TF { ~ } { \__speg_match_head_subject: }
  }

\cs_new:Npn \__speg_match_tail_subject_keep_space:
  {
    \__speg_match_if_head_is_space:TF
      {
        \exp_last_unbraced:Nv \__speg_gobble_space:w
          { g__speg_match_ \int_use:N \g__speg_match_level_int _str }
      }
      { \__speg_match_tail_subject: }
  }

\cs_new_protected:Npn \__speg_match_gpop_subject:N #1
  {
    \tl_set:Ne #1 { \__speg_match_head_subject_keep_space: }
    \__speg_match_gset_subject:e
      { \__speg_match_tail_subject_keep_space: }
  }

%% #1: pattern variable; #2: subject string.
\cs_new_protected:Npn \__speg_match:Nn #1 #2
  {
    %\iow_log:e { Matching ~ \exp_not:V #1 : }
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Match {#2} }
    \__speg_match_gset_index:n {1}
    \__speg_match_gset_false:
    #1 {#2}
    \__speg_match_if_sub_success:T
      {
        \__speg_match_gset_true:
        \__speg_match_update_index:
      }
    \__speg_tracing_match_stop:n { Match }
    \__speg_match_gdecr_level:
  }
\cs_set_eq:NN \MatchSpeg \__speg_match:Nn

\prg_new_protected_conditional:Npnn \speg_match:Nn #1 #2 { T, F, TF }
  {
    \__speg_match:Nn #1 {#2}
    \bool_if:cTF { g__speg_match_1_bool }
      { \prg_return_true: } { \prg_return_false: }
  }
\cs_set_eq:NN \IfSpegMatchT  \speg_match:NnT
\cs_set_eq:NN \IfSpegMatchF  \speg_match:NnF
\cs_set_eq:NN \IfSpegMatchTF \speg_match:NnTF

\seq_new:N \g__speg_capture_seq

%% #1: pattern variable; #2: subject string; #3: seq variable for captures.
\cs_new_protected:Npn \__speg_extract:NnN #1 #2 #3
  {
    \seq_gclear:N \g__speg_capture_seq
    \__speg_match:Nn #1 {#2}
    \seq_set_eq:NN #3 \g__speg_capture_seq
  }
\cs_set_eq:NN \ExtractSpeg \__speg_extract:NnN

\prg_new_protected_conditional:Npnn \speg_extract:NnN #1 #2 #3 { T, F, TF }
  {
    \__speg_extract:NnN #1 {#2} #3
    \bool_if:cTF { g__speg_match_1_bool }
      { \prg_return_true: }
      {
        \seq_clear:N #3
        \prg_return_false:
      }
  }
\cs_set_eq:NN \IfSpegExtractT  \speg_extract:NnNT
\cs_set_eq:NN \IfSpegExtractF  \speg_extract:NnNF
\cs_set_eq:NN \IfSpegExtractTF \speg_extract:NnNTF

%% #1: pattern from "patt1 / patt2 / patt3"; #2: subject string.
\cs_new_protected:Npn \speg_choice:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Choice {#2} }
    \__speg_match_gset_index:n {1}
    \__speg_match_gset_false:
    \tl_map_inline:nn {#1}
      {
        ##1 {#2}
        \__speg_match_if_sub_success:T
          {
            \__speg_match_gset_true:
            \tl_map_break:
          }
      }
    \__speg_match_if_success:T
      {
        \__speg_match_gset_eq_subject:
        \__speg_match_update_index:
      }
    \__speg_tracing_match_stop:n { Choice }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_choice:nn {nV}
\cs_set_eq:NN \SpegChoice \speg_choice:nn

\tl_new:N \l__speg_pattern_head_tl

%% #1: pattern from "patt1 * patt2 * patt3"; #2: subject string.
\cs_new_protected:Npn \speg_concat:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Concat {#2} }
    \__speg_match_gset_subject:n {#2}
    \__speg_match_gset_index:n {1}
    \__speg_match_gset_true:
    \tl_map_inline:nn {#1}
      {
        \tl_set:Ne \l__speg_pattern_head_tl { \tl_head:n {##1} }
        \exp_after:wN \tl_if_eq:NNTF \l__speg_pattern_head_tl \SpegCp
          { ##1 }
          {
            \__speg_match_apply_subject:n {##1}
            \__speg_match_if_sub_success:TF
              {
                \__speg_match_gset_eq_subject:
                \__speg_match_update_index:
              }
              {
                \__speg_match_gset_false:
                \tl_map_break:
              }
          }
      }
    \__speg_tracing_match_stop:n { Concat }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_concat:nn {nV}
\cs_set_eq:NN \SpegConcat \speg_concat:nn

%% #1: pattern from "! patt"; #2: subject string.
\cs_new_protected:Npn \speg_not:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Not {#2} }
    \__speg_match_gset_subject:n {#2}
    \__speg_match_gset_index:n {1}
    #1 {#2}
    \__speg_match_if_sub_success:TF
      { \__speg_match_gset_false: } { \__speg_match_gset_true: }
    \__speg_tracing_match_stop:n { Not }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_not:nn {nV}
\cs_set_eq:NN \SpegNot \speg_not:nn

%% #1: pattern from "& patt"; #2: subject string.
```

```
\cs_new_protected:Npn \speg_and:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { And {#2} }
    \__speg_match_gset_subject:n {#2}
    \__speg_match_gset_index:n {1}
    #1 {#2}
    \__speg_match_if_sub_success:TF
      { \__speg_match_gset_true: } { \__speg_match_gset_false: }
    \__speg_tracing_match_stop:n { And }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_and:nn {nV}
\cs_set_eq:NN \SpegAnd \speg_and:nn

%% #1 and #2: pattern and repeat number from "patt ^ {n}"; #3: subject string.
\cs_new_protected:Npn \speg_repeat:nnn #1 #2 #3
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Repeat {#2} {#3} }
    \__speg_match_gset_index:n {1}
    \__speg_match_gset_subject:n {#3}
    \int_compare:nNnTF {#2} < {0}
      %% at most -#2 occurrences
      {
        \__speg_match_gset_true:
        \int_step_inline:nn {-#2}
          {
            \__speg_match_apply_subject:n {#1}
            \__speg_match_if_sub_success:TF
              {
                \__speg_match_gset_eq_subject:
                \__speg_match_update_index:
              }
              { \prg_break: }
          }
      }
      %% at least #2 occurrences
      {
        \__speg_match_gset_true:
        \int_step_inline:nn {#2}
          {
            \__speg_match_apply_subject:n {#1}
            \__speg_match_if_sub_success:TF
              {
                \__speg_match_gset_eq_subject:
                \__speg_match_update_index:
              }
              {
                \__speg_match_gset_false:
                \prg_break:
              }
          }
        \__speg_match_if_success:T
          {
            \bool_do_while:nn { \__speg_match_if_sub_success_p: }
              {
                \__speg_match_apply_subject:n {#1}
                \__speg_match_if_sub_success:T
                  {
                    \__speg_match_gset_eq_subject:
                    \__speg_match_update_index:
                  }
              }
          }
      }
    \__speg_tracing_match_stop:n { Repeat }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_repeat:nnn {nnV}
\cs_set_eq:NN \SpegRepeat \speg_repeat:nnn

%% #1: string to match; #2: subject string.
\cs_new_protected:Npn \speg_p:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { P {#1} {#2} }
    \__speg_match_gset_true:
    \__speg_match_gset_index:n {1}
    \__speg_match_gset_subject:n {#2}
    \str_map_inline:nn {#1}
      {
        \__speg_match_gpop_subject:N \l_tmpa_str
        \str_if_eq:VnTF \l_tmpa_str {##1}
          {
            \__speg_match_gincr_index:
          }
          {
            \__speg_match_gset_false:
            \str_map_break:
          }
      }
    \__speg_tracing_match_stop:n { P }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_p:nn {nV}
\cs_set_eq:NN \SpegP \speg_p:nn

%% #1: number of characters; #2: subject string.
\cs_new_protected:Npn \speg_q:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { Q {#1} {#2} }
    \int_compare:nNnTF { \str_count:n {#2} } < {#1}
      {
        \__speg_match_gset_false:
        \__speg_match_gset_index:n {1}
      }
      {
        \__speg_match_gset_true:
        \__speg_match_gset_subject:e { \str_range:nnn {#2} {1 + #1} {-1} }
        \__speg_match_gset_index:e { \int_eval:n {1 + #1} }
      }
    \__speg_tracing_match_stop:n { Q }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_q:nn {nV}
\cs_set_eq:NN \SpegQ \speg_q:nn

%% #1: character ranges; #2: subject string.
```

```
\cs_new_protected:Npn \speg_r:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { R {#1} {#2} }
    \__speg_match_gset_false:
    \__speg_match_gset_index:n {1}
    \str_if_empty:nF {#2}
      {
        %% \l_tmpa_str is empty when #2 is empty
        \__speg_match_gset_subject:n {#2}
        \__speg_match_gpop_subject:N \l_tmpa_str
        \__speg_r_aux:nn #1 {} {} \prg_break_point:
      }
    \__speg_tracing_match_stop:n { R }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_r:nn {nV}
\cs_set_eq:NN \SpegR \speg_r:nn

\cs_new_protected:Npn \__speg_r_aux:nn #1 #2
  {
    \tl_if_empty:nTF {#2}
      { \prg_break: }
      {
        \bool_lazy_or:nnTF
          { \int_compare_p:eNn { `\l_tmpa_str } < { `#1 } }
          { \int_compare_p:eNn { `\l_tmpa_str } > { `#2 } }
          { \__speg_r_aux:nn }
          {
            \__speg_match_gset_true:
            \__speg_match_gset_index:n {2}
            \prg_break:
          }
      }
  }

%% #1: character set; #2: subject string.
\cs_new_protected:Npn \speg_s:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_tracing_match_start:n { S {#1} {#2} }
    \tl_if_empty:nTF {#2}
      {
        \__speg_match_gset_false:
        \__speg_match_gset_index:n {1}
      }
      {
        \__speg_match_gset_subject:n {#2}
        \__speg_match_gpop_subject:N \l_tmpa_str
        \str_if_in:nVTF {#1} \l_tmpa_str
          {
            \__speg_match_gset_true:
            \__speg_match_gset_index:n {2}
          }
          {
            \__speg_match_gset_false:
            \__speg_match_gset_index:n {1}
          }
      }
    \__speg_tracing_match_stop:n { S }
    \__speg_match_gdecr_level:
  }
\cs_generate_variant:Nn \speg_s:nn {nV}
\cs_set_eq:NN \SpegS \speg_s:nn

\int_new:N \l__speg_pos_int

%% \SpegC: simple capture. #1: pattern; #2: subject string.
\cs_new_protected:Npn \speg_c:nn #1 #2
  {
    \__speg_match_gincr_level:
    \__speg_match_gset_subject:n {#2}
    \__speg_tracing_match_start:n { C {#2} }
    #1 {#2}
    \__speg_match_if_sub_success:TF
      {
        \__speg_match_gset_true:
        \__speg_match_gset_eq_index:
        \__speg_match_gset_eq_subject:
        \__speg_match_get_capture:nN {#2} \l_tmpa_str
        \seq_gput_right:NV \g__speg_capture_seq \l_tmpa_str
        %\seq_log:N \g__speg_capture_seq
      }
      {
        \__speg_match_gset_false:
        \__speg_match_gset_index:n {1}
      }
    \__speg_tracing_match_stop:n { C }
    \__speg_match_gdecr_level:
  }
\cs_set_eq:NN \SpegC \speg_c:nn

%% \SpegCp: position capture.
\cs_new_protected:Npn \speg_cp:
  {
    \__speg_tracing_match_start:n { Cp }
    \__speg_match_get_position:N \l__speg_pos_int
    \seq_gput_right:Ne \g__speg_capture_seq { \int_use:N \l__speg_pos_int }
    \__speg_tracing_match_stop:n { Cp }
  }
\cs_set_eq:NN \SpegCp \speg_cp:
```

2.4 Trace speg functions

```
\NewDocumentCommand \SetSpegTracing { m }
  {
    \keys_set:nn { speg/tracing/main } {#1}
  }

\bool_new:N \g__speg_tracing_set_bool
\bool_new:N \g__speg_tracing_match_bool

\keys_define:nn { speg/tracing/main }
  {
    +set .code:n = \bool_gset_true:N  \g__speg_tracing_set_bool,
    -set .code:n = \bool_gset_false:N \g__speg_tracing_set_bool,
    +match .code:n = \bool_gset_true:N  \g__speg_tracing_match_bool,
    -match .code:n = \bool_gset_false:N \g__speg_tracing_match_bool,
    all  .code:n = \__speg_enable_all_tracings:,
    none .code:n = \__speg_disable_all_tracings:,
  }

\cs_new_protected_nopar:Npn \__speg_enable_all_tracings:
  {
    \bool_gset_true:N \g__speg_tracing_set_bool
    \bool_gset_true:N \g__speg_tracing_match_bool
  }

\cs_new_protected_nopar:Npn \__speg_disable_all_tracings:
  {
    \bool_gset_false:N \g__speg_tracing_set_bool
    \bool_gset_false:N \g__speg_tracing_match_bool
  }

\cs_new_protected:Npn \__speg_tracing:nn #1 #2
  {
    \bool_if:cT { g__speg_tracing_ #1 _bool } {#2}
  }

\cs_new_protected:Npn \__speg_tracing_match_start:n #1
  {
    \__speg_tracing:nn {match}
      {
        \iow_log:e
          {
            \prg_replicate:nn { (\g__speg_match_level_int - 1) * 2 } {~}
            Start#1
          }
      }
  }

\cs_new_protected:Npn \__speg_tracing_match_stop:n #1
  {
    \__speg_tracing:nn {match}
      {
        \iow_log:e
          {
            \prg_replicate:nn { (\g__speg_match_level_int - 1) * 2 } {~}
            Stop#1: \__speg_match_use_status:, \__speg_match_use_index:
                    %, {\__speg_match_use_subject:}
          }
      }
  }
```