The clistmap package[*]

Erwann Rogard[†]

Released 2022-01-29

Abstract

Let $\langle \mathit{clist}\rangle \doteq \langle e_1\rangle$`,...,`$\langle e_n\rangle$[1, l3clist]. This package provides a key-based interface for defining templates whose job is to partition $\langle \mathit{clist}\rangle$, and map differentiatedly across its components. `\clistmap:nnn{`$\langle \mathit{clist}\rangle$`}{...,`$\langle \mathit{instance}_i\rangle$`,...}`$\langle \mathit{args}\rangle$ iterates over the i's. Implicit in $\langle \mathit{instance}_i\rangle$ is $\langle \mathit{rule\ sequence}_i\rangle$ (the template), $\langle \mathit{cs\ name}_i\rangle$, and $\langle \mathit{signature}_i\rangle = \langle \mathit{args}\rangle$' signature. A sequence of instances can be made into a new instance: `serial_math_and:N`$\doteq$`{first_math:N,serial_rest_math_and:N}`, and likewise for the second component. `$\clistmap_inline:nnn{Z,C,Q,R}{serial_math_and:N} {\mathbb{#1}}$` expands to $\mathbb{Z}$, $\mathbb{C}$, $\mathbb{Q}$, and $\mathbb{R}$. `\clistmap:nnnn` takes an additional argument, $\langle \mathit{chain}\rangle \sim$`end`|`append`|`nest`|`join`, narrowing the set of instances needed to obtain a particular behaviour.

Contents

I Usage 3

1 Overview 3

2 Programming 3
- 2.1 key . . . 3
- 2.2 cs . . . 4

II Listing 6

1 Using keys 6
- `rule` . . . 6
- `rule_sequence` . . . 6
- `instance` . . . 6
- `instance_sequence` . . . 6

2 Preset keys 7
- `rule` . . . 7
- `rule_sequence` . . . 7
- `instance` . . . 7
- `instance_sequence` . . . 8

---

[*] This file describes version v1.2, last revised 2022-01-29.

[†] first.lastname at gmail.com

3 cs 9
  3.1 plain 9
  math 9
  3.2 chain 9
  append 9
  nest 9
  join 10

III Other 10

1 Bibliograhy 10

2 To do 10

3 Support 11

IV Implementation 11

1 boilerplate 11

2 name 12

3 c 12

4 rule_link 13

5 inline 14

6 eval 15

7 chain 17

8 use_w 20

9 rule 21

10 rule template 22

11 instantiate 24

12 property 25

13 instance 30

14 preset 33
  14.1 rule 33
  14.2 rule_sequence 34
  14.3 cs 35
  14.4 instance 36
  14.5 instance_sequence 37

Part I

Usage

1 Overview

Let ⟨*clist*⟩≡⟨*head*⟩`,`⟨*rest*⟩. The lifecycle has four stages. First, one provides templates called *rule*s, parameterized by ⟨*rule sequence*⟩, ⟨*cs name*⟩, and ⟨*signature*⟩. Typically, a rule checks for the recursion tail[1, l3quark] in some combination of ⟨*head*⟩ and ⟨*rest*⟩, based on which it does either of: stop, recurse, forward to ⟨*rule sequence*⟩, and in each case optionally expands `\`⟨*`cs name`*⟩`:`⟨*`signature`*⟩`n{`⟨*args*⟩`}{`⟨*head*⟩`}`. Second, one associates keys to sequences of rules, *rule sequence*. Those preset are `first`, `middle`, `last`, `serial_second`, and `serial_last`, for which the stated expression is evaluated for each $\langle e_i \rangle$ in their respective subsets. Brace groups are preserved. Third, one declares *instance*s of combinations of ⟨*rule sequence*⟩, ⟨*cs name*⟩, and ⟨*signature*⟩. For example, `middle_comma:N` and `serial_middle:` bind together `middle` and `,#1{#2}`, and `,~#1`, respectively. Fourth, define sequences of instances under the constraint that ⟨*signature*⟩ is identical across them, *instance sequence*s. Among presets, `comma:N` and `serial:` comprise in their natural order the matches for `(?:first_apply|comma_middle|comma_last):N`, and `(?:first_apply|serial_middle|serial_second|serial_last):`, respectively. They expand to `#1{`$\langle e_1 \rangle$`},...,#1{`$\langle e_n \rangle$`}`, and $\langle e_1 \rangle$`,~...,~and~`$\langle e_n \rangle$, respectively. `\clistmap:nnn` works the same with an instance sequence or the list of its constituents.

2 Programming

2.1 key

`rule`

```
\clistmap_keys_set:n{ rule = {⟨key⟩}{⟨code⟩} }
```

Parameter semantics

- `#1` ⟨*rule sequence*⟩
- `#2` ⟨*cs name*⟩
- `#3` ⟨*signature*⟩
- `#4` ⟨*head is group*⟩
- `#5` ⟨*arguments*⟩
- `#6` ⟨*clist head*⟩
- `#7` ⟨*clist rest*⟩

Requirement ⟨*code*⟩ is in terms of `#1-#7`

`rule_if_rest_is_tail_eval_else`
`rule_if_empty_stop_else`

```
\clistmap_keys_-
set:n{ rule_if_rest_is_tail_eval_else = {⟨name⟩}{⟨code⟩} }
```

Semantics Specialization of `rule`

`rule_sequence`

```
\clistmap_keys_set:n{ rule_sequence = {...,⟨key_j⟩ = { ...{⟨rule_i⟩}...},...} }
```

instance

\clistmap_keys_set:n{ instance = { ⟨*key prefix*⟩ = {⟨*rule sequence*⟩}{⟨*cs name*⟩}{⟨*signature*⟩} } }

Semantics Associates `\clistmap_instance_key:nn`{⟨*key prefix*⟩}{⟨*signature*⟩} with the RHS of ⟨*key prefix*⟩ =

instance_sequence

\clistmap_keys_set:n{ instance_sequence = { ⟨*key*⟩ = { ...,⟨*instance*$_i$⟩,...},... } }

2.2 cs

clistmap_keys_set:n

\clistmap_keys_set:n{⟨*keyval list*⟩}

\clistmap_info_clist:nn ⋆
\clistmap_info_prop:nn ⋆

\clistmap_info_clist:nn{⟨*key*⟩}{⟨*code*⟩}

Note Used for generating this doc

\clistmap_signature:n ⋆
\clistmap_instance_key:nn ⋆

\clistmap_instance_key:n{⟨*key prefix*⟩}{⟨*signature*⟩}

Expands to ⟨*key prefix*⟩:⟨*signature*⟩

\clistmap_instance_sequence_p:n ⋆
\clistmap_instance_p:n ⋆

\clistmap_instance_p:n{⟨*key*⟩}

Semantics Whether the instance has been registered

\clistmap_use_w:nnnn ⋆
\clistmap_use_w:nnnnn ⋆
\clistmap_use_w_group:nnnnnn ⋆

\clistmap_use_w:nnnnn
{⟨*rule*⟩}
{⟨*rule sequence (internal)* ⟩}
{⟨*cs name*⟩}
{⟨*signature*⟩}
{⟨*head is group*⟩}⟨*more*⟩\q_recursion_stop

Semantics Evaluates ⟨*code*⟩ associated with ⟨*rule*⟩
Note For use inside ⟨*code*⟩ on the RHS of `rule =` ⟨*rule bis*⟩⟨*code*⟩

\clistmap_bound_cs_group:nnnnn ⋆

\clistmap_bound_cs_group:nnnnn
{⟨*cs name*⟩}
{⟨*signature*⟩}
{⟨*group*⟩}
{⟨*args*⟩}
{⟨*elem*⟩}

Definition ⟨*new elem*⟩=`\bool_if:nTF`{⟨*group*⟩}{{⟨*elem*⟩}}{⟨*elem*⟩}
Semantics \⟨*cs name*⟩:⟨*signature*⟩⟨*args*⟩{⟨*new elem*⟩}
Note For use in conjunction with `\clistmap_use_w:nnnnn` and variants

\clistmap:nnn ⋆

```
\clistmap:nnn{⟨clist⟩}{ ...,⟨instance_i⟩,... }{⟨args⟩}
\clistmap:nnn{⟨clist⟩}{ ...,⟨instance sequence_i⟩,... }{⟨args⟩}
```

Requirement

⟨*clist*⟩ has no trailing `,`

⟨*args*⟩ has signature `\clistmap_signature:n{`⟨*instance*$_i$⟩`}`

Expands to

First version For each i, the ⟨*code*⟩ associated with ⟨*rule*$_i$⟩.

Second version Iterates over the constituents of ⟨*rule sequence*$_i$⟩

\clistmap_inline:nnn

```
\clistmap_inline:nnn{ ...,⟨instance_i⟩,... }{⟨code⟩}
```

Requirement `\clistmap_signature:n{`⟨*instance*$_i$⟩`}`=`N`

\clistmap:nnnn ⋆

```
\clistmap:nnnn{⟨clist⟩}{⟨instances⟩}{⟨args⟩}{⟨end⟩}
\clistmap:nnnn{⟨clist⟩}{⟨instances⟩}{⟨args⟩}{⟨append⟩}
\clistmap:nnnn{⟨clist⟩}{⟨instances⟩}{⟨args⟩}{⟨nest⟩}
\clistmap:nnnn{⟨clist_1⟩}{⟨instances⟩}{⟨args⟩}{⟨join⟩}{⟨clist_2⟩}
```

Semantics

end `\clistmap:nnn{`⟨*clist*⟩`}{`⟨*instances*⟩`}{`⟨*args*⟩`}`

append ⟨*end*⟩`\clistmap:nnnn{`⟨*clist*⟩`}`

nest `\clistmap:nnnn{`⟨*end*⟩`}`

join `\clistmap:nnnn{`⟨*end*⟩,⟨*clist*$_2$⟩`}`

\clistmap_inline:nnnn ⋆

```
\clistmap_inline:nnnn{⟨clist⟩}{⟨instances⟩}{⟨code⟩}{⟨chain⟩}
```

Requirement `\clistmap_signature:n{`⟨*instance*$_i$⟩`}`=`empty` or `N`

Part II

Listing

1 Using keys

Listing 1. `rule`

```
\clistmap_keys_set:n
{%
  rule = {if_rest_is_tail_stop_else_forward_rest}
  {%
    \quark_if_recursion_tail_stop:n{#7}
    \clistmap_use_w:nnne
    {#1}{#2}{#3}
    {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop
  }
}
```

Listing 2. `rule_sequence`

```
\clistmap_keys_set:n
{
  rule_sequence =
  {
    first =
    {
      {if_empty_stop_else_forward_head}
      {if_rest_is_tail_eval_else_error}
    }
  }
}
```

Listing 3. `instance`

```
\clistmap_keys_set:n
{
  instance =
  {
    {N}{first_apply}{first}{@@_apply},
    {}{first_apply}{first}{@@_apply}
  }
}
```

Listing 4. `instance_sequence`

```
\clistmap_keys_set:n
{%
  instance_sequence =
  {
      {N}{comma:}{first_apply:, rest_comma:},
      {}{serial_and:}{first_apply:, serial_rest_and:},
    }
  }
```

2 Preset keys

Listing 5. `rule`

```
if_rest_is_tail_stop_else_eval_recurse
if_rest_is_tail_stop_else_forward_rest
if_empty_stop_else_error
if_empty_stop_else_forward_head
if_empty_stop_else_forward_rest
if_empty_stop_else_forward_all
if_rest_is_tail_eval_else_error
if_rest_is_tail_eval_else_stop
if_rest_is_tail_eval_else_recurse
```

Listing 6. `rule_sequence`

```
first
middle
last
serial_second
serial_last
```

Listing 7. `instance`

```
first_apply:N
first_map:N
first_math:N
first_noindent:N
last_apply:N
last_comma_map:N
last_comma_math:N
last_comma:N
serial_last:N
serial_second:N
middle_apply:N
middle_comma_map:N
middle_comma_math:N
middle_comma:N
serial_last_math_and:N
serial_middle_math:N
serial_second_math_and:N
first_apply:
```

first_math:

first_noindent:

first_unbrace:

last_apply:

last_comma_math:

last_comma_unbrace:

last_comma:

last_newline:

last_unbrace:

middle_apply:

middle_comma_math:

middle_comma_unbrace:

middle_comma:

middle_newline:

middle_unbrace:

serial_last_and:

serial_last_math_and:

serial_middle_math:

serial_middle:

serial_second_and:

serial_second_math_and:

Listing 8. `instance_sequence`

apply:N

comma_map:N

comma_math:N

comma:N

rest_apply:N

rest_comma_map:N

rest_comma_math:N

rest_comma:N

serial_and:N

serial_math_and:N

serial_rest_and:N

serial_rest_math_and:N

apply:

comma_math:

newline:

comma_unbrace:

comma:

rest_apply:

rest_comma_math:

rest_newline:

rest_comma_unbrace:

rest_comma:

rest_unbrace:

serial_and:

serial_math_and:

unbrace:

serial_rest_and:

serial_rest_math_and:

3 cs

3.1 plain

Listing 9. `math`

```
\ExplSyntaxOn
\clistmap:nnn{Z, C, Q, R}
{ first_math:N, serial_rest_math_and:N }
{\mathbb}
\ExplSyntaxOff
```

$\mathbb{Z}$, $\mathbb{C}$, $\mathbb{Q}$, and $\mathbb{R}$

3.2 chain

Listing 10. `append`

```
\ExplSyntaxOn
\clistmap_inline:nnnn
{{J,u,l,e,s},Jim,Catherine}
{first_map:N}
{#1}
{append}
{middle_comma:N}
{~#1}
{append}
{%^^A
  serial_second:N,%^^A ignored in this case
  serial_last:N
}
{~et~#1}
{end}
\ExplSyntaxOff
```

Jules, Jim, et Catherine

Listing 11. `nest`

```
\ExplSyntaxOn
\noindent
\clistmap_inline:nnnn
{{foo},{bar,baz},{qux}}
```

```
{comma_unbrace:}
{}
{nest}
{newline:}
{}
{end}
\ExplSyntaxOff
```

foo
bar
baz
qux

Listing 12. `join`

```
\ExplSyntaxOn
\clistmap_inline:nnnn
{foo,bar}
{comma:}
{}
{join}
{baz}
{comma:}
{}
{end}
\ExplSyntaxOff
```

foo,bar,baz

Part III

Other

1 Bibliograhy

[1] The LaTeX3 Project Team. *The LaTeX3 interfaces*. `https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf`. 2019.

2 To do

1. “Warning: A control sequence of the form ...___clistmap” That’s because of the way `\__clist_instance_name:nn` is set up, and passing to it an internal control sequence. So? Modify `\__clist_instance_name:nn`.

3 Support

This package is available from https://github.com/rogard/clistmap.

Part IV
Implementation

```
⟨*package⟩
⟨@@=clistmap⟩
%      \ExplSyntaxOn
```

1 boilerplate

`\clistmap_keys_set:n`
`\clistmap_info_clist:nn`

```
\cs_generate_variant:Nn\str_if_eq:nnTF{e}
\cs_generate_variant:Nn\tl_to_str:n{e}
\cs_generate_variant:Nn\prop_gput:Nnn{Nee}
\cs_generate_variant:Nn\erw_parameter:n{e}
\cs_generate_variant:Nn\erw_argument:nn{ne}
\cs_generate_variant:Nn\erw_parameter:nn{ne}
\cs_generate_variant:Nn\erw_clist_tl:nn{ne}
\cs_new:Npn\__clistmap_empty:w#1\q_recursion_stop{}
\clist_new:N\__clistmap_helper_clist
\cs_new_protected:Nn
\clistmap_keys_set:n{ \keys_set:nn{ __clistmap }{ #1 } }
\prop_new:N\__clistmap_info_clist_prop
\cs_new_protected:Npn
\__clistmap_info_clist_put:nn
#1 % <key>
#2 % <name:signature>
{\prop_gput:Nnn\__clistmap_info_clist_prop{#1}{#2}}
\cs_new_protected:Npn
\clistmap_info_clist:nn
#1 % <key>
#2 % <code>
{\clist_map_inline:cn{\prop_item:Nn\__clistmap_info_clist_prop{#1}}{#2}}
\prop_new:N\__clistmap_info_prop_prop
\cs_new_protected:Npn
\__clistmap_info_prop_put:nn
#1 % <key>
#2 % <name:signature>
{\prop_gput:Nnn\__clistmap_info_prop_prop{#1}{#2}}
\cs_new:Nn
\__clistmap_brace:nn{{{#1}{#2}}}
\cs_new:Npn
\clistmap_info_prop:n
#1 % <key>
{ \prop_map_function:cN
  {\prop_item:Nn\__clistmap_info_prop_prop{#1}}\__clistmap_brace:nn }
\cs_new:Npn
\clistmap_info_prop:nn
```

```
#1 % <key>
#2 % <code>
{ \prop_map_inline:cn
  {\prop_item:Nn\__clistmap_info_prop_prop{#1}}{#2} }
\cs_new:Nn
\__clistmap_group_if:nn
{\bool_if:nTF{#2}{{#1}}{#1}}
\cs_generate_variant:Nn\__clistmap_group_if:nn{e}
\cs_new:Nn
\__clistmap_head_clist:n
{%
  \exp_args:Ne
  \tl_head:n
  { \clist_map_function:nN{#1}\__clistmap_head_clist_aux:n }
}
\cs_new:Nn
\__clistmap_head_clist_aux:n{#1}
```

(*End definition for* `\clistmap_keys_set:n` *and* `\clistmap_info_clist:nn`*. These functions are documented on page 4.*)

2 name

`\__clistmap_rule_name:n`
`\__clistmap_instance_name:nnn`
`\__clistmap_instance_signature:n`
`\__clistmap_rule_sequence_name:n`

```
\cs_new:Npn
\__clistmap_rule_name:n
#1 % <rules>
{rule_#1}
\cs_new:Npn
\__clistmap_instance_name:nn
#1 % <rules>
#2 % <cs name>
{instance_#1_#2}
\cs_new:Npn
\__clistmap_instance_name:nnn
#1 % <rule>
#2 % <next rules>
#3 % <cs name>
{\__clistmap_instance_name:nn{#1_#2}{#3}}
\cs_new:Npn
\__clistmap_instance_signature:n
#1 % <signature>
{n#1w}
```

(*End definition for* `\__clistmap_rule_name:n` *and others.*)

3 c

```
\cs_new:Npn
\__clistmap_c:n
#1 % <name>
{__clistmap_#1}
\cs_generate_variant:Nn\__clistmap_c:n{e}
```

```
\cs_new:Npn
\__clistmap_c:nn
#1 % <name>
#2 % <signature>
{\__clistmap_c:n{#1:#2}}
\cs_generate_variant:Nn\__clistmap_c:nn{e, ee}
\cs_new:Npn
\__clistmap_bound_cs_c:nn
#1 % <name>
#2 % <signature>
{#1:#2n}
\cs_new:Npn
\__clistmap_rule_c:n
#1 % <rule>
{%
  \__clistmap_c:en
  {\__clistmap_rule_name:n{#1}}
  {nnnnnnnn}
}
\cs_new:Npn
\__clistmap_instance_c:nn
#1 % <rules>
#2 % <cs name>
{ \__clistmap_c:e
  { \__clistmap_instance_name:nn{#1}{#2} } }
\cs_generate_variant:Nn\__clistmap_instance_c:nn{e}
\cs_new:Npn
\__clistmap_instance_c:nnn
#1 % <rules>
#2 % <cs name>
#3 % <signature>
{%
  \__clistmap_c:ee
  { \__clistmap_instance_name:nn{#1}{#2} }
  { \__clistmap_instance_signature:n{#3} }
}
\cs_generate_variant:Nn\__clistmap_instance_c:nnn{e, nne}
\cs_new:Npn
\__clistmap_instance_c_this:nnnn
#1 % <rule>
#2 % <next rules>
#3 % <cs name>
#4 % <signature>
{ \__clistmap_instance_c:enn
  {\__clistmap_rule_link:nn{#1}{#2}}{#3}{#4} }
```

4 rule_link

```
\cs_new:Npn
\__clistmap_rule_link:nn
#1 % <rule 1>
#2 % <rule 2>
{#1_#2}
\cs_new:Npn
```

```
\__clistmap_rule_link:n
#1 % <{rule{1}}...>
{%
  \__clistmap_rule_link:w#1\q_recursion_tail\q_recursion_stop
}
\cs_generate_variant:Nn\__clistmap_rule_link:n{e}
\cs_new:Npn
\__clistmap_rule_link:w
#1
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#1}
  \__clistmap_rule_link:nw #1 \q_recursion_stop}
\cs_new:Npn
\__clistmap_rule_link:nw
#1 % <rules>
#2 % <{rule{1}}...>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop_do:nn{#2}{#1}
  \__clistmap_rule_link:nnw{#1}#2\q_recursion_stop}
\cs_generate_variant:Nn\__clistmap_rule_link:nw{e}
\cs_new:Npn
\__clistmap_rule_link:nnw
#1 % <rules>
#2 % <rule{1}>
#3 % <{rule{2}}...>
\q_recursion_stop
{%
  \__clistmap_rule_link:ew
  {%
    \__clistmap_rule_link:nn
    {#1} % <rule 1>
    {#2} % <rule 2>
  } % <rules>
  #3 % <{rule{1}}...>
  \q_recursion_stop
}
```

5 inline

```
\cs_new_protected:Nn
\__clistmap_inline_set_exp_nnnot:Nn
{\cs_set:Nn#1
  {\exp_not:n
    {\exp_not:n
      {\exp_not:n{#2}}}}}
\cs_generate_variant:Nn\__clistmap_inline_set_exp_nnnot:Nn{c}
\cs_new:Nn\__clistmap_inline_c:n{__clistmap_#1:n}
\cs_new:Nn\__clistmap_inline_use:n
{%^^A BUG
  \use:c{\__clistmap_inline_c:n{#1}}}
\cs_new_protected:Nn
\__clistmap_inline_set_exp_nnnot:nn
```

```
{\__clistmap_inline_set_exp_nnnot:cn
  {\__clistmap_inline_c:n{#1}}{#2}}
\msg_new:nnn{__clistmap}
{inline-empty-N}
{instance~signature~must~be~empty~or~N;~got~’#1’}
\msg_new:nnn{__clistmap}
{inline-empty-args}
{instance~signature=empty;~so~should~args=#1}
```

6 eval

\clistmap:nnn
\clistmap_inline:nnn

```
\msg_new:nnn{__clistmap}{key}
{no~match~for~#1~in~instance~or~instance~sequence}
\msg_new:nnn{__clistmap}{signature-mismatch}
{instance~signature~must~be~#1;~instances:~#2}
\cs_new_protected:Npn
\clistmap_inline:nnn
#1 % <clist>
#2 % <instances>
#3 % <empty|code using #1>
{%^^A
  \bool_if:nTF
  { \__clistmap_instance_signature_p:nn{#2}{N} }
  {%^^A
    \__clistmap_inline_set_exp_nnnot:nn{a}{#3}
    \clistmap:nnn
    {#1} % <clist>
    {#2} % <key 1>
    {\__clistmap_a:n}
  }
  {%^^A
    \bool_if:nTF
    { \__clistmap_instance_signature_p:nn{#2}{} }
    {%^^A
      \tl_if_empty:nTF
      {#3}
      {%^^A
        \clistmap:nnn
        {#1} % <clist>
        {#2} % <key 1>
        {}
      }
      {%^^A
        \msg_error:nnnn{__clistmap}
        {inline-empty-args}
        {#3}
      }
    }
    {%^^A
      \msg_error:nnnn{__clistmap}
      {inline-empty-N}
      {#2}
    }
  }
}
\cs_new:Npn
\clistmap:nnn
% ^^A Warning: trailing ',' inside #2 => Error
#1 % <clist>
#2 % <key,...>
#3 % <arguments>
{%
  \__clistmap_eval:nenn
  {#2} % <instance key>,...
  {\tl_if_head_is_group_p:n{#1}} % <head is group>
  {#3} % <arguments>
  {#1} % <clist>
}
\cs_generate_variant:Nn\clistmap:nnn{e,f,x}
\cs_new:Npn
\__clistmap_eval:nnnn
#1 % <instance key>,...
#2 % <head is group>
#3 % <arguments>
#4 % <clist>
{%
  \exp_args:Ne
  \__clistmap_eval_aux:nnnn
  {\__clistmap_instance_expand:n{#1}}
  {#2} % <head is group>
  {#3} % <arguments>
  {#4} % <clist>
}
\cs_new:Npn
\__clistmap_eval_aux:nnnn
#1 % <instance key>,...
#2 % <head is group>
#3 % <arguments>
#4 % <clist>
{%
  \__clistmap_eval:nnnw
  {#2} % <head is group>
  {#3} % <arguments>
  {#4} % <clist>
  #1 % <instance key>,...
  , \q_recursion_tail
  \q_recursion_stop
}
\cs_generate_variant:Nn\__clistmap_eval:nnnn{ ne }
\cs_new:Npn
\__clistmap_eval:nnnw
#1 % <head is group>
#2 % <arguments>
#3 % <clist>
#4 % <instance key>
\q_recursion_stop
{%
```

```
  \quark_if_recursion_tail_stop:n{#4}
  \__clistmap_eval:nnnnw
  {#1} % <head is group>
  {#2} % <arguments>
  {#3} % <clist>
  #4 % <instance key>
  \q_recursion_stop
}
\cs_new:Npn
\__clistmap_eval:nnnnw
#1 % <head is group>
#2 % <arguments>
#3 % <clist>
#4 % <instance key>
, #5 % <instance key,...>
\q_recursion_stop
{%
  \exp_last_unbraced:Ne
  \__clistmap_eval:nnnnnn
  { \__clistmap_instance_get:n{#4} }
  {#1}{#2}{#3}
  \__clistmap_eval:nnnw
  {#1} % <head is group>
  {#2} % <arguments>
  {#3} % <clist>
  #5 % <instance key>
  \q_recursion_stop
}
\cs_new:Npn
\__clistmap_eval:nnnnnn
#1 % <rule sequence>
#2 % <cs name>
#3 % <signature>
#4 % <head is group>
#5 % <arguments>
#6 % <clist>
{%
  \exp_args:Ne
  \clistmap_use_w:nnnn
  { \__clistmap_rule_sequence_name:n{#1} } % <rule sequence>
  {#2}  % <cs name>
  {#3}  % <signature>
  {#4} % <head is group>
  #5
  #6, \q_recursion_tail\q_recursion_stop
}
```

(*End definition for* `\clistmap:nnn` *and* `\clistmap_inline:nnn`*. These functions are documented on page 5.*)

7 chain

```
\msg_new:nnn{__clistmap}
{chain}{unknown~chain~tag~#1}
```

```
\cs_new_protected:Npn
\__clistmap_append:NNN
#1 % <new>
#2 % <\__clistmap_append(?:_inline):nnn>
#3 % <\clistmap(?_inline):nnnn>
{%^^A
  #1
  #2
  {%^^A
    \clistmap:nnn{##1}{##2}{##3}
    #3{##1}
  }
}
\__clistmap_append:NNN
\cs_new:Nn
\__clistmap_append:nnn
\clistmap:nnnn
\__clistmap_append:NNN
\cs_new_protected:Nn
\__clistmap_append_inline:nnn
\clistmap_inline:nnnn
\cs_new_protected:Npn
\__clistmap_nest:NNN
#1 % <new>
#2 % <\__clistmap_nest(?:_inline):nnn>
#3 % <\clistmap(?_inline):nnnn>
{%^^A
  #1
  #2
  {%^^A
    \exp_args:Ne
    #3{ \clistmap:nnn{##1}{##2}{##3} }
  }
}
\__clistmap_nest:NNN
\cs_new:Nn
\__clistmap_nest:nnn
\clistmap:nnnn
\__clistmap_nest:NNN
\cs_new_protected:Nn
\__clistmap_nest_inline:nnn
\clistmap_inline:nnnn
\cs_new_protected:Npn
\__clistmap_join:NNNN
#1 % <new>
#2 % <\__clistmap_join(?:_inline):nnnn>
#3 % <\__clistmap_join(?:_inline):nnn>
#4 % <\clistmap(?_inline):nnnn>
{%^^A
  #1
  #2
  { #4{##1,##2}{##3}{##4} }
  #1
  #3
  { #2{\clistmap:nnn{##1}{##2}{##3}} }
}
\__clistmap_join:NNNN
\cs_new:Nn
\__clistmap_join:nnnn
\__clistmap_join:nnn
\clistmap:nnnn
\__clistmap_join:NNNN
\cs_new_protected:Nn
\__clistmap_join_inline:nnnn
\__clistmap_join_inline:nnn
\clistmap_inline:nnnn
\cs_new_protected:Npn
\__clistmap_chain:NNNNN
#1 % <new>
#2 % <__clistmap_chain(?:_inline):nnnn>
#3 % <__clistmap_append(?:_inline):nnn>
#4 % <__clistmap_nest(?:_inline):nnn>
#5 % <__clistmap_join(?:_inline):nnn>
{%^^A
  #1
  #2
  {%^^A
    \str_case:nnTF
    {##4}
    {%^^A
      {end}
      { \clistmap:nnn{##1}{##2}{##3} }
      {append}
      { #3{##1}{##2}{##3} }
      {nest}
      { #4{##1}{##2}{##3} }
      {join}
      { #5{##1}{##2}{##3} }
    }
    {}
    { \msg_error:nnn{__clistmap}{chain}{##4} }
  }
}
\__clistmap_chain:NNNNN
\cs_new:Nn
\clistmap:nnnn
\__clistmap_append:nnn
\__clistmap_nest:nnn
\__clistmap_join:nnn
\__clistmap_chain:NNNNN
\cs_new_protected:Nn
\__clistmap_inline_aux:nnnn
\__clistmap_append_inline:nnn
\__clistmap_nest_inline:nnn
\__clistmap_join_inline:nnn
\cs_new_protected:Npn
\clistmap_inline:nnnn
#1 % <clist>
```

```
#2 % <inst>
#3 % <args>
#4 % <chain>
{%^^A
  \bool_if:nTF
  { \__clistmap_instance_signature_p:nn{#2}{N} }
  {%^^A
    \__clistmap_inline_set_exp_nnnot:nn{a}{#3}
    \__clistmap_inline_aux:nnnn{#1}{#2}{\__clistmap_a:n}{#4}
  }
  { \__clistmap_inline_aux:nnnn{#1}{#2}{}{#4} }
}
```

8 use_w

\clistmap_use_w_group:nnnnnn
\clistmap_use_w:nnnn
\clistmap_use_w:nnnnn

For use inside ⟨*code*⟩ inside rule

```
\cs_new:Npn
\clistmap_use_w_group:nnnnnn
#1  % <rule sequence>
#2  % <cs name>
#3  % <signature>
#4  % <head is group>
#5  % <arguments>
#6  % <clist head>
{%
  \clistmap_use_w:nnnn
  {#1}{#2}{#3}
  {#4}#5{#6}
}
\cs_new:Npn
\clistmap_use_w:nnnn
#1 % <rule sequence>
#2 % <cs name>
#3 % <signature>
#4 % <head is group>
{%
  \use:c{ \__clistmap_instance_c:nnn{#1}{#2}{#3} }{#4}
}
\cs_generate_variant:Nn\clistmap_use_w:nnnn{nnne}
\cs_new:Npn
\clistmap_use_w:nnnnn
#1 % <rule>
#2 % <next rule sequence>
#3 % <cs name>
#4 % <signature>
#5 % <head is group>
{%
  \use:c{%
    \__clistmap_instance_c_this:nnnn
    {#1} % <rule>
    {#2} % <next rules>
    {#3} % <cs name>
    {#4} % <signature>
  }{#5}
}
\cs_generate_variant:Nn\clistmap_use_w:nnnnn{nnnne}
```

(*End definition for* `\clistmap_use_w_group:nnnnnn`*,* `\clistmap_use_w:nnnn`*, and* `\clistmap_use_w:nnnnn`*. These functions are documented on page 4.*)

`\clistmap_bound_cs_group:nnnnn`

```
\cs_new:Npn
\clistmap_bound_cs_group:nnnnn
#1 % <cs name>
#2 % <signature>
#3 % <group (bool)>
#4 % <arguments>
#5 % <clist>
{\__clistmap_bound_cs:nnne{#1}{#2}{#4}{\bool_if:nTF{#3}{{#5}}{#5}}}
\cs_generate_variant:Nn\clistmap_bound_cs_use_group:nnnnn{nnenn}
\cs_new:Npn
\__clistmap_bound_cs:nnnn
#1 % <cs name>
#2 % <signature>
#3 % <arguments>
#4 % <clist>
{ \use:c{\__clistmap_bound_cs_c:nn{#1}{#2}}#3{#4} }
\cs_generate_variant:Nn\__clistmap_bound_cs:nnnn{nnne}
```

(*End definition for* `\clistmap_bound_cs_group:nnnnn`*. This function is documented on page 4.*)

9 rule

`rule`

```
\keys_define:nn{ __clistmap }
{ rule.code:n = \__clistmap_rule:nn#1 }
```

(*End definition for* `rule`*. This function is documented on page 3.*)

`\__clistmap_rule:nn`

```
\prop_new:N\__clistmap_rule_clist
\__clistmap_info_clist_put:nn{rule}{__clistmap_rule_clist}
\cs_new_protected:Npn
\__clistmap_rule:nn
#1 % <rule>
#2 % <code>
{%
  \clist_gput_right:Nn\__clistmap_rule_clist{#1}
  \exp_args:Nno
  \cs_new_protected:cn
  { \__clistmap_rule_c:n{#1} }
  {%
    \__clistmap_rule_apply:nnnnnnnn
    {#1} % {<rule>}
    {#2} % {<code>}
    {##1} % <next rule>
    {##2} % <cs name>
    {##3} % <signature>
    {{##4}{##5}{##6}} % <head is group>
    % ^^A <arguments>
    % ^^A <clist head>
    {##7} % <clist rest>
    {##8} % <parameters}
  }
}
% ^^A  ##1 % <next rules>
% ^^A  ##2 % <cs name>
% ^^A  ##3 % <signature>
% ^^A  ##4 % <head is group>
% ^^A  ##5 % <arguments>
% ^^A  ##6 % <clist head>
% ^^A  ##7 % <clist rest>
% ^^A  ##8 % <parameters>
\cs_new_protected:Npn
\__clistmap_rule_apply:nnnnnnnn
#1 % <rule>
#2 % <code>
#3 % <next rules>
#4 % <cs name>
#5 % <signature>
#6 % {<head is group>}{<arguments>}{<clist head>}
#7 % <clist rest>
#8 % <parameters>
{%
  \__clistmap_rule_apply:ennnnnn
  {\__clistmap_instance_c_this:nnnn{#1}{#3}{#4}{#5}}
  {#2}#6{#7}{#8}
}
\cs_new_protected:Npn
\__clistmap_rule_apply:nnnnnnn
#1 % <instance>
#2 % <code>
#3 % <head is group>
#4 % <arguments>
#5 % <clist head>
#6 % <clist rest>
#7 % <parameters>
{%
  \cs_if_exist:cF{#1}
  {%^^A
    \cs_new:cpn{#1}
    #3#7#5, #6\q_recursion_stop % <parameters>
    {#2}
  }
}
\cs_generate_variant:Nn\__clistmap_rule_apply:nnnnnnn{e}
```

(*End definition for* `\__clistmap_rule:nn`.)

10 rule template

```
\cs_new:Nn
```

```
\__clistmap_quark_if_recursion_tail_stop:nn
{\quark_if_recursion_tail_stop:n{#1#2}}
\cs_generate_variant:Nn\__clistmap_quark_if_recursion_tail_stop:nn{e}
```

rule_if_rest_is_tail_eval_else

```
\keys_define:nn{ __clistmap }
{%
  rule_if_rest_is_tail_eval_else.code:n
  = {\__clistmap_rule_if_rest_is_tail_eval_else:nn#1}
}
\cs_new_protected:Npn
\__clistmap_rule_if_rest_is_tail_eval_else:nn
#1 % <name>
#2 % <else code>
{%
  % ^^A  ##1 % <next rules>
  % ^^A  ##2 % <cs name>
  % ^^A  ##3 % <signature>
  % ^^A  ##4 % <head is group>
  % ^^A  ##5 % <arguments>
  % ^^A  ##6 % <clist head>
  % ^^A  ##7 % <clist rest>
  % ^^A  ##8 % <parameters>
  \clistmap_keys_set:n
  {%
    rule = {if_rest_is_tail_eval_else_#1}
    {%
      \quark_if_recursion_tail_stop_do:nn{##7}
      {%
        \clistmap_bound_cs_group:nnnnn
        {##2} % <cs name>
        {##3} % <signature>
        {##4} % <head is group>
        {##5} % <arguments>
        {##6} % <clist>
      }
      #2
    }
  }
}
```

(*End definition for* `rule_if_rest_is_tail_eval_else`. *This function is documented on page 3.*)

rule_if_empty_stop_else

```
\keys_define:nn
{ __clistmap }
{
  rule_if_empty_stop_else.code:n
  = {\__clistmap_rule_if_empty_stop_else:nn#1}
}
\cs_new_protected:Npn
\__clistmap_rule_if_empty_stop_else:nn
#1 % <name>
#2 % <else code>
{%
```

```
  % ^^A  ##1 % <next rules>
  % ^^A  ##2 % <cs name>
  % ^^A  ##3 % <signature>
  % ^^A  ##4 % <head is group>
  % ^^A  ##5 % <arguments>
  % ^^A  ##6 % <clist head>
  % ^^A  ##7 % <clist rest>
  % ^^A  ##8 % <parameters>
  \clistmap_keys_set:n
  {%
    rule = {if_empty_stop_else_#1}
    {%
      \__clistmap_quark_if_recursion_tail_stop:en
      {\bool_if:nTF{##4}{{##6}}{##6}}{##7}
      #2
    }
  }
}
```

(*End definition for* `rule_if_empty_stop_else`*. This function is documented on page 3.*)

11 instantiate

`\__clistmap_instantiate:nnnn`

```
\cs_new_protected:Npn
\__clistmap_instantiate:nnnn
#1 % <rule>
#2 % <next rules>
#3 % <cs name>
#4 % <signature>
{%
  \exp_args:Ne
  \__clistmap_instantiate:nnnnn
  {\tl_count:n{#4}} % <signature arity>
  {#1} % <rule>
  {#2} % <next rules>
  {#3} % <cs name>
  {#4} % <signature>
}
\cs_new_protected:Npn
\__clistmap_instantiate:nnnnn
#1 % <signature arity>
#2 % <rule>
#3 % <next rules>
#4 % <cs name>
#5 % <signature>
{%^^A
  \__clistmap_instantiate:eeeeennn
  { \erw_parameter:n{ 1 } } % <head is group>
  { \erw_parameter:ne{2}{ #1 } } % <parameters>
  { \erw_parameter:e{ \int_eval:n{#1+2} } } % <clist head>
  { \erw_parameter:e{ \int_eval:n{#1+3} } } % <clist rest>
  { \erw_argument:ne{2}{ #5 } } % <arguments>
  { #2 } % <rule>
  { #3 } % <next rules>
  { #4 } % <cs name>
  { #5 } % <signature>
}
\cs_new:Npn
\__clistmap_instantiate:nnnnnnnn
#1 % <head is group>
#2 % <parameters>
#3 % <clist head>
#4 % <clist rest>
#5 % <arguments>
#6 % <rule>
#7 % <next rules>
#8 % <cs name>
#9 % <signature>
{%
  \use:c{ \__clistmap_rule_c:n{#6} }
  {#7} % <next rules>
  {#8} % <cs name>
  {#9} % <signature>
  {#1} % <head is group>
  {#2} % <arguments>
  {#3} % <clist head>
  {#4} % <clist rest>
  {#2} % <parameters>
}
\cs_generate_variant:Nn\__clistmap_instantiate:nnnnnnnn{eeeee}
```

(*End definition for* `\__clistmap_instantiate:nnnn`.)

12 property

`rule_sequence`

```
\cs_new:Npn
\__clistmap_rule_sequence_name:n
#1 % <rule sequence>
{%
  \__clistmap_rule_link:e
  {\__clistmap_rule_sequence_get:n{#1}{null}}
}
\keys_define:nn{__clistmap}
{ rule_sequence.code:n = \__clistmap_rule_sequence_from_keyval:n{#1} }
\prop_new:N\__clistmap_rule_sequence_prop
\__clistmap_info_prop_put:nn{rule_sequence}{__clistmap_rule_sequence_prop}
\cs_new_protected:Npn
\__clistmap_rule_sequence_from_keyval:n
#1 % <key = {{rule{1}}...>
{%
  \prop_set_from_keyval:Nn
  \__clistmap_rule_sequence_prop{#1}
}
\cs_new:Npn
```

```
\__clistmap_rule_sequence_get:n
#1 % <key>
{%
  \exp_args:Ne
  \__clistmap_rule_sequence_aux:n
  {%
    \prop_item:Nn
    \__clistmap_rule_sequence_prop{#1}
  }
}
\cs_new:Npn
\__clistmap_rule_sequence_aux:n
#1 % <value>
{%
  \prop_if_in:NnTF
  \__clistmap_rule_sequence_prop
  {#1}
  {\__clistmap_rule_sequence_get:n{#1}}
  {#1}
}
```

(*End definition for* `rule_sequence`. *This function is documented on page 3.*)

`\clistmap_signature:n`
`\clistmap_instance_p:n`

```
\prg_new_conditional:Npnn
\clistmap_instance:n
#1
{p}
{\prop_if_in:NnTF
  \__clistmap_instance_prop{#1}
  {\prg_return_true:}
  {\prg_return_false:}
}
\msg_new:nnn{__clistmap}{instance-not}{#1~is~not~an~instance}
\msg_new:nnn{__clistmap}{key-conflict}{key~#1~already~exists~in~prop~#2}
\prop_new:N\__clistmap_instance_prop
\__clistmap_info_prop_put:nn{instance}{__clistmap_instance_prop}
\cs_new_protected:Npn
\__clistmap_instance_put:nnnn
#1 % <key>
#2 % <rule sequence>
#3 % <name>
#4 % <signature>
{%
  \prop_gput:Nnn
  \__clistmap_instance_prop{#1}
  { {#2}{#3}{#4} }
}
\cs_new:Npn
\__clistmap_instance_get:n
#1 % <key>
{ \prop_item:Nn\__clistmap_instance_prop{#1} }
\cs_new:Nn
\clistmap_signature:n
```

```
{%^^A
  \bool_if:nTF
  { \clistmap_instance_p:n{#1} }
  { \__clistmap_instance_signature_get:n{#1} }
  { \msg_error:nnn{__clistmap}{instance-not}{#1} }
}
\cs_new:Npn
\__clistmap_instance_signature_get:n
#1 % <instance>
{\exp_last_unbraced:Ne\use_iii:nnn
  {\__clistmap_instance_get:n{#1}}}
\cs_new:Npn
\__clistmap_instance_expand:n
#1 %^^A <instance(?:_sequence)_1,...>
{%^^A
  \__clistmap_instance_expand:w
  #1, \q_recursion_tail
  \q_recursion_stop
}
\cs_new:Npn
\__clistmap_instance_expand:w
#1 %^^A <instance(?:_sequence)_1,...>
,#2
\q_recursion_stop
{
  \quark_if_recursion_tail_stop:n{#1#2}
  \__clistmap_instance_expand:nw#1, #2\q_recursion_stop
}
\cs_new:Npn
\__clistmap_instance_expand:nw
#1 % <head>
, #2 % <rest>
\q_recursion_stop
{
  \bool_if:nTF
  {\clistmap_instance_sequence_p:n{#1}}
  {%^^A
    \exp_args:Ne
    \__clistmap_instance_expand:n
    { \__clistmap_instance_sequence_get:n{#1} }
  }
  {%
    \bool_if:nTF
    {\clistmap_instance_p:n{#1}}
    {#1}
    {\msg_error:nnn{__clistmap}{neither-inst-seq}{#1}}
  }
  \quark_if_recursion_tail_stop:n{#2},%^^A comma
  \__clistmap_instance_expand:nw#2\q_recursion_stop
}
\msg_new:nnn{__clistmap}{neither-inst-seq}
{#1~is~neither~an~instance~nor~a~sequence}
\prg_new_conditional:Npnn
\__clistmap_instance_signature:nn
```

```
#1 % <instance_1,...>
#2 % <signature>
{p}
{%^^A
  \bool_if:nTF
  {
    \exp_args:Ne
    \__clistmap_instance_signature_aux_p:nn
    {%^^A
      \exp_args:Ne
      \clist_map_function:nN
      { \__clistmap_instance_expand:n{#1} }
      \clistmap_signature:n
    }
    {#2}
  }
  {\prg_return_true:}
  {\prg_return_false:}
}
\prg_new_conditional:Npnn
\__clistmap_instance_signature_aux:nn
#1 % <signature_1,...>
#2 % <signature>
{p}
{%
  \tl_if_empty:nTF
  {#1}
  {%^^A
    \tl_if_empty:nTF{#2}
    {\prg_return_true:}
    {\prg_return_false:}
  }
  {%^^A
    \bool_if:nTF
    {%^^A
      \erw_and_tl_p:nn
      { \str_if_eq_p:nn{#2} }
      { #1 }
    }
    {\prg_return_true:}
    {\prg_return_false:}
  }
}
```

(*End definition for* `\clistmap_signature:n` *and* `\clistmap_instance_p:n`*. These functions are documented on page 4.*)

instance_sequence
`\clistmap_instance_sequence_p:n`

```
\keys_define:nn{ __clistmap }
{%^^A
  instance_sequence.code:n
  = {%^^A
    \clist_map_function:nN{#1}
    \__clistmap_instance_sequence_put:n
  }
}
\prg_new_conditional:Npnn
\clistmap_instance_sequence:n
#1
{p}
{%
  \prop_if_in:NnTF
  \__clistmap_instance_sequence_prop{#1}
  {\prg_return_true:}
  {\prg_return_false:}
}
\prop_new:N
\__clistmap_instance_sequence_prop
\__clistmap_info_prop_put:nn{instance_sequence}{__clistmap_instance_sequence_prop}
\cs_new:Nn\__clistmap_first_braced:nn{{#1}}
\cs_new:Nn\__clistmap_instance_sequence_keys:
{%
  \prop_map_function:NN
  \__clistmap_instance_sequence_prop
  \__clistmap_first_braced:nn
}
% ^^A\cs_new_protected:Npn
% ^^A\__clistmap_instance_sequence_put:n
% ^^A#1 % <{key}{key{1},...}>
% ^^A{ \__clistmap_instance_sequence_put:nn#1 }
\cs_new_protected:Npn
\__clistmap_instance_sequence_put:n
#1 % <{signature}{prefix key}{prefix key{1},...}>
{ \__clistmap_instance_sequence_put:nnn#1 }
\cs_new:Npn
\__clistmap_instance_sequence_value:nn
#1 % <signature>
#2 % <key prefix 1,...>
{%
  \exp_args:Nne
  \erw_clist_tl:nn{\c_false_bool}
  {%^^A
    \clist_map_tokens:nn
    {#2}
    { \__clistmap_instance_sequence_value_aux:nn{#1} }
  }
}
\cs_new:Nn
\__clistmap_instance_sequence_value_aux:nn
{{\clistmap_instance_key:nn{#2}{#1}}}
\cs_new_protected:Npn
\__clistmap_instance_sequence_put:nnn
#1 % <signature>
#2 % <prefix key>
#3 % <prefix key{1}>,...
{%^^A
  \exp_args:Nee
  \__clistmap_instance_sequence_put:nn
  { \clistmap_instance_key:nn{#2}{#1} }
  { \__clistmap_instance_sequence_value:nn{#1}{#3} }
}
\cs_new_protected:Npn
\__clistmap_instance_sequence_put:nn
#1 % <key>
#2 % <instance key{1}>,...
{%
  \prop_if_in:NnTF
  \__clistmap_instance_prop{#1}
  {\msg_error:nnnn{__clistmap}{key-conflict}{#1}{instance}}
  {%
    \prop_gput:Nnn
    \__clistmap_instance_sequence_prop{#1}
    { #2 }
  }
}
\cs_new:Nn
\clistmap_instance_sequence:n
{\__clistmap_instance_sequence_get:n{#1}}
\cs_new:Npn
\__clistmap_instance_sequence_get:n
#1 % <key>
{\prop_item:Nn\__clistmap_instance_sequence_prop{#1}}
```

(*End definition for* `instance_sequence` *and* `\clistmap_instance_sequence_p:n`*. These functions are documented on page 4.*)

13 instance

`instance`
`\clistmap_instance_key:nn`

```
\keys_define:nn{__clistmap}
{ instance.code:n = \clist_map_function:nN{#1} \__clistmap_instance:n }
\cs_new_protected:Npn
\__clistmap_instance:n
% ^^A#1 % {key prefix}{<rule sequence>}{<cs name>}{<signature>}
#1 % {<signature>}{key prefix}{<rule sequence>}{<cs name>}
{ \__clistmap_instance:nnnn#1 }
\cs_new_protected:Npn
\__clistmap_instance:nnnn
% ^^A#1 % <key prefix>
% ^^A#2 % <rule sequence>
% ^^A#3 % <cs name>
% ^^A#4 % <signature>
#1 % <signature>
#2 % <key prefix>
#3 % <rule sequence>
#4 % <cs name>
{%
  \exp_args:Ne
  \__clistmap_instance_aux:nnnn
  { \clistmap_instance_key:nn{#2}{#1} }
  {#3}{#4}{#1}
```

```
}
\cs_new:Npn
\clistmap_instance_key:nn
#1 % <key prefix>
#2 % <signature>
{#1:#2}
\cs_new_protected:Npn
\__clistmap_instance_aux:nnnn
#1 % <key>
#2 % <rule sequence>
#3 % <signature>
#4 % <cs name>
{%
  \__clistmap_instance_put:nnnn{#1}{#2}{#3}{#4}
  \__clistmap_instance_using_key:nnn{#2}{#3}{#4}
}
\cs_new_protected:Npn
\__clistmap_instance_using_key:nnn
#1 % <rule sequence>
#2 % <cs name>
#3 % <signature>
{%
  \__clistmap_instance_using_list:enn
  { \__clistmap_rule_sequence_get:n{#1}{null} } % <{rule{1}}...>
  {#2} % <cs name>
  {#3}% <signature>
}
\cs_new_protected:Npn
\__clistmap_instance_using_list:nnn
#1 % <{rule{1}}{rule{2}}...>
#2 % <cs name>
#3 % <signature>
{%
  \exp_last_unbraced:Ne
  \__clistmap_instance_backward:nnnnn
  {%
    { \tl_count:n{#3} } % <signature arity>
    \erw_last:n{#1}  % <rule{n}>
    { \erw_remove_first:e{\tl_reverse:n{#1}} } % <{rule{n-1}}{rule{n-2}}...>
  }
  { #2 } % <cs name>
  { #3 } % <signature>
}
\cs_generate_variant:Nn\__clistmap_instance_using_list:nnn{enn}
\msg_new:nnn{__clistmap}{null}
{clistmap~expects~’null’~as~the~last~rule;~got~’#1’}
\cs_new_protected:Npn
\__clistmap_instance_backward:nnnnn
#1 % <signature arity>
#2 % <rule{n}>
#3 % <{rule{n-1}}{rule{n-2}}...>
#4 % <cs name>
#5 % <signature>
{%
```

```
  \str_case:nnTF{#2}
  { {null}{} }
  {%
    \__clistmap_instance_backward:nnnw
    {#2} % <next rules>
    {#4} % <cs name>
    {#5} % <signature>
    #3\q_recursion_tail % <{rule{n}}{rule{n-1}}...>
    \q_recursion_stop
  }
  {%
    \msg_error:nnn{__clistmap}
    {null}
    {#2}
  }
}
\cs_generate_variant:Nn\__clistmap_instance_backward:nnnnn{eee}
\cs_new_protected:Npn
\__clistmap_instance_backward:nnnw
#1 % <next rules>
#2 % <cs name>
#3 % <signature>
#4 % <{rule{n}}{rule{n-1}}...>
\q_recursion_stop
{%
  \quark_if_recursion_tail_stop:n{#4}
  \__clistmap_instance_backward:nnnnw
  {#1} % <next rules>
  {#2} % <cs name>
  {#3} % <signature>
  #4 % <rule{n}>
  % <{rule{n-1}}...>
  \q_recursion_stop
}
\cs_generate_variant:Nn\__clistmap_instance_backward:nnnw{e}
\cs_new_protected:Npn
\__clistmap_instance_backward:nnnnw
#1 % <next rules>
#2 % <cs name>
#3 % <signature>
#4 % <rule{n}>
#5 % <{rule{n-1}}...>
\q_recursion_stop
{%
  \__clistmap_instantiate:nnnn
  {#4} % <rule>
  {#1} % <next rules>
  {#2} % <cs name>
  {#3} % <signature>
  \__clistmap_instance_backward:ennw
  {\__clistmap_rule_link:nn{#4}{#1}} % <next rules>
  {#2} % <cs name>
  {#3} % <signature>
  #5 % <{rule{n}}...>
  \q_recursion_stop
}
```

(*End definition for* `instance` *and* `\clistmap_instance_key:nn`*. These functions are documented on page 4.*)

14 preset

14.1 rule

```
\msg_new:nnn{__clistmap}{tail}{expects~tail;~got~’#1’}
% ^^A ##1 % <next rules>
% ^^A ##2 % <cs name>
% ^^A ##3 % <signature>
% ^^A ##4 % <head is group>
% ^^A ##5 % <arguments>
% ^^A ##6 % <clist head>
% ^^A ##7 % <clist rest>
% ^^A ##8 % <args>
\clistmap_keys_set:n
{%
  rule = {if_rest_is_tail_stop_else_eval_recurse}
  {%
    \quark_if_recursion_tail_stop:n{#7}
    \clistmap_bound_cs_group:nnnnn
    {#2} % <cs name>
    {#3} % <signature>
    {#4} % <head is group>
    {#5} % <arguments>
    {#6} % <clist>
    \clistmap_use_w:nnnne
    {if_rest_is_tail_stop_else_eval_recurse} % <rule>
    {#1} % <next rule rule sequence>
    {#2} % <cs name>
    {#3} % <signature>
    {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop % <head is group>
  },
  rule = {if_rest_is_tail_stop_else_forward_rest}
  {%
    \quark_if_recursion_tail_stop:n{#7}
    \clistmap_use_w:nnne
    {#1}{#2}{#3}
    {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop
  },
  rule_if_empty_stop_else = {error}
  {%
    \msg_error:nnn{__clistmap}{tail}{#6#7}
    \__clistmap_empty:w{}\q_recursion_stop
  },
  rule_if_empty_stop_else = {forward_head}
  {%
    \bool_if:nTF{#4}
    {%
      \clistmap_use_w_group:nnnnnn{#1}{#2}{#3}{#4}{#5}{#6}
      ,\q_recursion_tail\q_recursion_stop
    }
    {%
      \clistmap_use_w:nnnn{#1}{#2}{#3}
      {#4}#5#6,\q_recursion_tail\q_recursion_stop
    }
  },
  rule_if_empty_stop_else = {forward_rest}
  {%
    \clistmap_use_w:nnne
    {#1}{#2}{#3}
    {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop
  },
  rule_if_empty_stop_else = {forward_all}
  {%
    \bool_if:nTF{#4}
    {%
      \clistmap_use_w_group:nnnnnn{#1}{#2}{#3}{#4}{#5}{#6},
      #7\q_recursion_stop
    }
    {%
      \clistmap_use_w:nnnn
      {#1}{#2}{#3}{#4}#5#6, #7\q_recursion_stop
    }
  },
  rule_if_rest_is_tail_eval_else = {error}
  {%
    \msg_error:nnn{__clistmap}{tail}{#6}
    \__clistmap_empty:w\q_recursion_stop
  },
  rule_if_rest_is_tail_eval_else = {stop}
  {%
    \__clistmap_empty:w{}\q_recursion_stop
  },
  rule_if_rest_is_tail_eval_else = {recurse}
  {%
    \clistmap_use_w:nnnne
    {if_rest_is_tail_eval_else_recurse} % <rule>
    {#1} % <next rule rule sequence>
    {#2} % <cs name>
    {#3} % <signature>
    {\tl_if_head_is_group_p:n{#7}} % <head is group>
    #5 % <argument>
    #7 % <clist>
    \q_recursion_stop
  }
}
```

14.2 rule_sequence

```
\clistmap_keys_set:n
{%
  rule_sequence =
  {%
    first =
    {
      {if_empty_stop_else_forward_head}
      {if_rest_is_tail_eval_else_error}
    },
    middle =
    {
      {if_empty_stop_else_forward_all}
      {if_rest_is_tail_stop_else_forward_rest}
      {if_rest_is_tail_stop_else_eval_recurse}
    },
    last =
    {
      {if_empty_stop_else_forward_all}
      {if_rest_is_tail_stop_else_forward_rest}
      {if_rest_is_tail_eval_else_recurse}
    },
    serial_second =
    {
      {if_empty_stop_else_forward_all}
      {if_rest_is_tail_stop_else_forward_rest}
      {if_rest_is_tail_eval_else_stop}
    },
    serial_last =
    {
      {if_empty_stop_else_forward_all}
      {if_rest_is_tail_stop_else_forward_rest}
      {if_rest_is_tail_stop_else_forward_rest}
      {if_rest_is_tail_eval_else_recurse}
    }
  }
}
```

14.3 cs

```
\msg_new:nnnn{__clistmap}{text}{text~is~not~loaded}{amsmath}
\cs_new:Nn\__clistmap_unbrace_aux:n{#1}
\erw_keys_set:n
{
  clist_map_inline =
  {%
    {Nn}{apply}{#1{#2}},
    {Nn}{math}{\ensuremath{#1{#2}}},
    {Nn}{comma_map}{,\clist_map_function:nN#2#1},
    {Nn}{comma}{,#1{#2}},
    {Nn}{serial_math}{\text{,~}\ensuremath{#1{#2}}},
    {Nn}{serial_math_and}{\text{,~and~}\ensuremath{#1{#2}}},
    {Nn}{map}{\clist_map_function:nN#2#1},
    {Nn}{noindent}{\noindent},
    {n}{apply}{#1},
    {n}{math}{\ensuremath{#1}},
    {n}{comma_math}{,\ensuremath{#1}},
    {n}{newline}{\\#1},
    {n}{comma_unbrace}{,\__clistmap_unbrace_aux:n#1},
    {n}{comma}{,#1},
    {n}{noindent}{\noindent},
    {n}{serial_and}{,~and~#1},
    {n}{serial_math_and}{\text{,~and~}\ensuremath{#1}},
    {n}{serial_math}{\text{,~}\ensuremath{#1}},
    {n}{serial}{,~#1},
    {n}{unbrace}{\__clistmap_unbrace_aux:n#1}
  }
  {nnn}
  {
    \clist_gput_right:Nn\__clistmap_helper_clist{#2:#1}
    \cs_new:cn{__clistmap_#2:#1}{#3}
  }
}
```

14.4 instance

```
\clistmap_keys_set:n
{
  instance =
  {
    {N}{first_apply}{first}{__clistmap_apply},
    {N}{first_map}{first}{__clistmap_map},
    {N}{first_math}{first}{__clistmap_math},
    {N}{first_noindent}{first}{__clistmap_noindent},
    {N}{last_apply}{last}{__clistmap_apply},
    {N}{last_comma_map}{last}{__clistmap_comma_map},
    {N}{last_comma_math}{last}{__clistmap_comma_math},
    {N}{last_comma}{last}{__clistmap_comma},
    {N}{serial_last}{serial_last}{__clistmap_comma},
    {N}{serial_second}{serial_second}{__clistmap_comma},
    {N}{middle_apply}{middle}{__clistmap_apply},
    {N}{middle_comma_map}{middle}{__clistmap_comma_map},
    {N}{middle_comma_math}{middle}{__clistmap_comma_math},
    {N}{middle_comma}{middle}{__clistmap_comma},
    {N}{serial_last_math_and}{serial_last}{__clistmap_serial_math_and},
    {N}{serial_middle_math}{middle}{__clistmap_serial_math},
    {N}{serial_second_math_and}{serial_second}{__clistmap_serial_math_and},
    {}{first_apply}{first}{__clistmap_apply},
    {}{first_math}{first}{__clistmap_math},
    {}{first_noindent}{first}{__clistmap_noindent},
    {}{first_unbrace}{first}{__clistmap_unbrace},
    {}{last_apply}{last}{__clistmap_apply},
    {}{last_comma_math}{last}{__clistmap_comma_math},
    {}{last_comma_unbrace}{last}{__clistmap_comma_unbrace},
    {}{last_comma}{last}{__clistmap_comma},
    {}{last_newline}{last}{__clistmap_newline},
    {}{last_unbrace}{last}{__clistmap_unbrace},
    {}{middle_apply}{middle}{__clistmap_apply},
    {}{middle_comma_math}{middle}{__clistmap_comma_math},
    {}{middle_comma_unbrace}{middle}{__clistmap_comma_unbrace},
    {}{middle_comma}{middle}{__clistmap_comma},
    {}{middle_newline}{middle}{__clistmap_newline},
    {}{middle_unbrace}{middle}{__clistmap_unbrace},
    {}{serial_last_and}{serial_last}{__clistmap_serial_and},
    {}{serial_last_math_and}{serial_last}{__clistmap_serial_math_and},
    {}{serial_middle_math}{middle}{__clistmap_serial_math},
    {}{serial_middle}{middle}{__clistmap_serial},
    {}{serial_second_and}{serial_second}{__clistmap_serial_and},
    {}{serial_second_math_and}{serial_second}{__clistmap_serial_math_and},
  }
}
```

14.5 instance_sequence

```
\clistmap_keys_set:n
{%
  instance_sequence =
  {
    {N}{apply}{first_apply, rest_apply},
    {N}{comma_map}{first_map, rest_comma_map},
    {N}{comma_math}{first_math, rest_comma_math},
    {N}{comma}{first_apply, rest_comma},
    {N}{rest_apply}{middle_apply, last_apply},
    {N}{rest_comma_map}{middle_comma_map, last_comma_map},
    {N}{rest_comma_math}{middle_comma_math, last_comma_math},
    {N}{rest_comma}{middle_comma, last_comma},
    {N}{serial_and}{first_apply, serial_rest_and},
    {N}{serial_math_and}{first_math, serial_rest_math_and},
    {N}{serial_rest_and}{serial_middle, serial_second_and, serial_last_and},
    %^^A <one long entry>
    {N}
    {serial_rest_math_and}
    {serial_middle_math, serial_second_math_and, serial_last_math_and}
    %^^A </one long entry>
    ,
    {}{apply}{first_apply, rest_apply},
    {}{comma_math}{first_math, rest_comma_math},
    {}{newline}{first_apply, rest_newline},
    {}{comma_unbrace}{first_unbrace, rest_comma_unbrace},
    {}{comma}{first_apply, rest_comma},
    {}{rest_apply}{middle_apply, last_apply},
    {}{rest_comma_math}{middle_comma_math, last_comma_math},
    {}{rest_newline}{middle_newline, last_newline},
    {}{rest_comma_unbrace}{middle_comma_unbrace, last_comma_unbrace},
    {}{rest_comma}{middle_comma, last_comma},
    {}{rest_unbrace}{middle_unbrace, last_unbrace},
    {}{serial_and}{first_apply, serial_rest_and},
    {}{serial_math_and}{first_apply, serial_rest_math_and},
    {}{unbrace}{first_unbrace, rest_unbrace},
    % ^^A <one long entry>
    {}{serial_rest_and}
    {serial_middle, serial_second_and, serial_last_and}
    % ^^A </one long entry>
    ,
    % ^^A <one long entry>
    {}{serial_rest_math_and}
    {serial_middle_math, serial_second_math_and, serial_last_math_and}
    % ^^A </one long entry>
  }
}
```

15 other

```
\ProcessKeysOptions{__clistmap}
\ExplSyntaxOff
⟨/package⟩
```