

The CLiS_TEX package*

Erwann Rogard[†]

Released 2022-01-27

Abstract

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

Contents

I	Usage	3
1	Overview	3
2	Document	3
3	Programming	4
3.1	key	4
3.2	cs	4
II	Listing	6
1	Using keys	6
	rule	6
	rule_sequence	6
	instance	6
	instance_sequence	6

*This file describes version v1.0, last revised 2022-01-27.

[†]first.lastname at gmail.com

2	Preset keys	7
	rule	7
	rule_sequence	7
	instance	7
	instance_sequence	8
3	cs	9
	3.1 plain	9
	math	9
	3.2 chain	9
	append	9
	nest	9
	join	10
III	Other	10
1	Bibliograh	10
2	Support	10
IV	Implementation	10
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
15	other	37

Part I

Usage

1 Overview

Let $\langle clist \rangle \equiv \langle head \rangle, \langle rest \rangle$. The lifecycle has four stages. First, one provides templates called *rules*, parameterized by $\langle rule\ sequence \rangle$, $\langle cs\ name \rangle$, and $\langle signature \rangle$. Typically, a rule checks for the recursion tail [1, l3quark] in some combination of $\langle head \rangle$ and $\langle rest \rangle$, based on which it does either of: stop, recurse, forward to $\langle rule\ sequence \rangle$, and in each case optionally expands $\backslash \langle cs\ name \rangle : \langle signature \rangle n \{ \langle args \rangle \} \{ \langle head \rangle \}$. 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 *instances* of combinations of $\langle rule\ sequence \rangle$, $\langle cs\ name \rangle$, and $\langle signature \rangle$. 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 $\langle signature \rangle$ is identical across them, *instance sequences*. 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 $\#1\{\langle e_1 \rangle\}, \dots, \#1\{\langle e_n \rangle\}$, and $\langle e_1 \rangle, \sim \dots, \sim \text{and} \sim \langle e_n \rangle$, respectively. `\clistex:nnn` works the same with an instance sequence or the list of its constituents.

2 Document

3 Programming

3.1 key

<code>rule</code>	<code>\clistex_keys_set:n{ rule = {\langle key \rangle}{\langle code \rangle} }</code>
-------------------	--

Parameter semantics

- #1 $\langle rule\ sequence \rangle$
- #2 $\langle cs\ name \rangle$
- #3 $\langle signature \rangle$
- #4 $\langle head\ is\ group \rangle$
- #5 $\langle arguments \rangle$
- #6 $\langle clist\ head \rangle$
- #7 $\langle clist\ rest \rangle$

Requirement $\langle code \rangle$ is in terms of #1-#7

<code>rule_if_rest_is_tail_eval_else</code>	<code>\clistex_keys_-</code>
<code>rule_if_empty_stop_else</code>	<code>set:n{ rule_if_rest_is_tail_eval_else = {\langle name \rangle}{\langle code \rangle} }</code>

Semantics Specialization of rule

<code>rule_sequence</code>	<code>\clistex_keys_set:n{ rule_sequence = { ..., \langle key_j \rangle = { ...{\langle rule_i \rangle} ... }, ... } }</code>
----------------------------	---

<code>instance</code>	<code>\clistex_keys_set:n{ instance = { \langle key prefix \rangle = {\langle rule sequence \rangle}{\langle cs name \rangle}{\langle signature \rangle} } }</code>
-----------------------	---

Semantics Associates `\clistex_instance_key:nn{\langle key prefix \rangle}{\langle signature \rangle}` with the RHS of $\langle key\ prefix \rangle =$

<code>instance_sequence</code>	<code>\clistex_keys_set:n{ instance_sequence = { \langle key \rangle = { ..., \langle instance_i \rangle, ..., ... } }</code>
--------------------------------	---

3.2 cs

<code>clistex_keys_set:n</code>	<code>\clistex_keys_set:n{\langle keyval list \rangle}</code>
---------------------------------	---

<code>\clistex_info_clist:nn *</code>	<code>\clistex_info_clist:nn{\langle key \rangle}{\langle code \rangle}</code>
<code>\clistex_info_prop:nn *</code>	

Note Used for generating this doc

<code>\clistex_signature:n *</code>	<code>\clistex_instance_key:n{\langle key prefix \rangle}{\langle signature \rangle}</code>
<code>\clistex_instance_key:nn *</code>	

Expands to $\langle key\ prefix \rangle : \langle signature \rangle$

<code>\clistex_instance_sequence_p:n *</code>	<code>\clistex_instance_p:n{\langle key \rangle}</code>
<code>\clistex_instance_p:n *</code>	

Semantics Whether the instance has been registered

<hr/>	<hr/>	<hr/>
<code>\clistex_use_w:nnnn</code>	<code>*</code>	<code>\clistex_use_w:nnnnn</code>
<code>\clistex_use_w:nnnnn</code>	<code>*</code>	<code>{\rule}</code>
<code>\clistex_use_w_group:nnnnnn</code>	<code>*</code>	<code>{\rule sequence (internal) }</code>
<hr/>		<code>{\cs name}</code>
		<code>{\signature}</code>
		<code>{\head is group}\more\q_recursion_stop</code>
Semantics Evaluates $\langle code \rangle$ associated with $\langle rule \rangle$		
Note For use inside $\langle code \rangle$ on the RHS of <code>rule = \langle rule bis \rangle \langle code \rangle</code>		

<hr/>	<hr/>	<hr/>
<code>\clistex_bound_cs_group:nnnnn</code>	<code>*</code>	<code>\clistex_bound_cs_group:nnnnn</code>
<hr/>		<code>{\cs name}</code>
		<code>{\signature}</code>
		<code>{\group}</code>
		<code>{\args}</code>
		<code>{\elem}</code>
Definition $\langle new\ elem \rangle = \backslash bool_if:nTF\{\langle group \rangle\}\{\{\langle elem \rangle\}\}\{\langle elem \rangle\}$		
Semantics $\backslash \langle cs\ name \rangle : \langle signature \rangle \langle args \rangle \{\langle new\ elem \rangle\}$		
Note For use in conjunction with <code>\clistex_use_w:nnnnn</code> and variants		

<hr/>	<hr/>	<hr/>
<code>\clistex:nnn</code>	<code>*</code>	<code>\clistex:nnn{\clist}{\dots,\langle instance_i \rangle,\dots}\{\langle args \rangle\}</code>
<hr/>		<code>\clistex:nnn{\clist}{\dots,\langle instance\ sequence_i \rangle,\dots}\{\langle args \rangle\}</code>
Requirement		
$\langle clist \rangle$ has no trailing ,		
$\langle args \rangle$ has signature <code>\clistex_signature:n{\langle instance_i \rangle}</code>		
Expands to		
First version For each i , the $\langle code \rangle$ associated with $\langle rule_i \rangle$.		
Second version Iterates over the constituents of $\langle rule\ sequence_i \rangle$		

<hr/>	<hr/>	<hr/>
<code>\clistex_inline:nnn</code>		<code>\clistex_inline:nnn{\dots,\langle instance_i \rangle,\dots}\{\langle code \rangle\}</code>
<hr/>	Requirement	<code>\clistex_signature:n{\langle instance_i \rangle}=N</code>

<hr/>	<hr/>	<hr/>
<code>\clistex:nnnn</code>	<code>*</code>	<code>\clistex:nnnn{\clist}{\langle instances \rangle}\{\langle args \rangle\}\{\langle end \rangle\}</code>
<hr/>		<code>\clistex:nnnn{\clist}{\langle instances \rangle}\{\langle args \rangle\}\{\langle append \rangle\}</code>
		<code>\clistex:nnnn{\clist}{\langle instances \rangle}\{\langle args \rangle\}\{\langle nest \rangle\}</code>
		<code>\clistex:nnnn{\clist_1}{\langle instances \rangle}\{\langle args \rangle\}\{\langle join \rangle\}\{\langle clist_2 \rangle\}</code>
Semantics		
end <code>\clistex:nnn{\langle clist \rangle}\{\langle instances \rangle\}\{\langle args \rangle\}</code>		
append <code>\langle end \rangle \backslash \clistex:nnnn{\langle clist \rangle}</code>		
nest <code>\clistex:nnnn{\langle end \rangle}</code>		
join <code>\clistex:nnnn{\langle end \rangle,\langle clist_2 \rangle}</code>		

<hr/>	<hr/>	<hr/>
<code>\clistex_inline:nnnn</code>	<code>*</code>	<code>\clistex_inline:nnnn{\clist}{\langle instances \rangle}\{\langle code \rangle\}\{\langle chain \rangle\}</code>
<hr/>	Requirement	<code>\clistex_signature:n{\langle instance_i \rangle}=empty or N</code>

Part II

Listing

1 Using keys

Listing 1. rule

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

Listing 2. rule_sequence

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

Listing 3. instance

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

Listing 4. instance_sequence

```
\clistex_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
\clistex:nnn{Z, C, Q, R}
{ first_math:N, serial_rest_math_and:N }
{\mathbb{b}}
\ExplSyntaxOff

```

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

3.2 chain

Listing 10. append

```

\ExplSyntaxOn
\clistex_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
\clistex_inline:nnnn
{{foo},{bar,baz},{qux}}

```

```

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

```

```

foo
bar
baz
qux

```

Listing 12. join

```

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

```

```

foo,bar,baz

```

Part III Other

1 Bibliography

- [1] The L^AT_EX3 Project Team. *The L^AT_EX3 interfaces*. <https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf>. 2019.

2 Support

This package is available from <https://github.com/rogard/clistex>.

Part IV

Implementation

```

1 <*package>
2 <@@=clistex>
3 %      \ExplSyntaxOn

```

1 boilerplate

```

\clistex_keys_set:n
\clistex_info_clist:nn
4 \cs_generate_variant:Nn\str_if_eq:nnTF{e}
5 \cs_generate_variant:Nn\tl_to_str:n{e}
6 \cs_generate_variant:Nn\prop_gput:Nnn{Nee}
7 \cs_generate_variant:Nn\erw_parameter:n{e}
8 \cs_generate_variant:Nn\erw_argument:nn{ne}
9 \cs_generate_variant:Nn\erw_parameter:nn{ne}
10 \cs_generate_variant:Nn\erw_clist_tl:nn{ne}
11 \cs_new:Npn\__clistex_empty:w#1\q_recursion_stop{}
12 \clist_new:N\__clistex_helper_clist
13 \cs_new_protected:Nn
14 \clistex_keys_set:n{ \keys_set:nn{ __clistex }{ #1 } }
15 \prop_new:N\__clistex_info_clist_prop
16 \cs_new_protected:Npn
17 \__clistex_info_clist_put:nn
18 #1 % <key>
19 #2 % <name:signature>
20 {\prop_gput:Nnn\__clistex_info_clist_prop{#1}{#2}}
21 \cs_new_protected:Npn
22 \clistex_info_clist:nn
23 #1 % <key>
24 #2 % <code>
25 {\clist_map_inline:cn{\prop_item:Nn\__clistex_info_clist_prop{#1}{#2}}
26 \prop_new:N\__clistex_info_prop_prop
27 \cs_new_protected:Npn
28 \__clistex_info_prop_put:nn
29 #1 % <key>
30 #2 % <name:signature>
31 {\prop_gput:Nnn\__clistex_info_prop_prop{#1}{#2}}
32 \cs_new:Nn
33 \__clistex_brace:nn{{{#1}{#2}}}}
34 \cs_new:Npn
35 \clistex_info_prop:n
36 #1 % <key>
37 { \prop_map_function:cn
38   {\prop_item:Nn\__clistex_info_prop_prop{#1}}\__clistex_brace:nn }
39 \cs_new:Npn
40 \clistex_info_prop:nn
41 #1 % <key>
42 #2 % <code>
43 { \prop_map_inline:cn
44   {\prop_item:Nn\__clistex_info_prop_prop{#1}{#2}} }
45 \cs_new:Nn

```

```

46 \__clistex_group_if:nn
47 {\bool_if:nTF{#2}{#{#1}}{#1}}
48 \cs_generate_variant:Nn\__clistex_group_if:nn{e}
49 \cs_new:Nn
50 \__clistex_head_clist:n
51 {%
52   \exp_args:Ne
53   \tl_head:n
54   { \clist_map_function:nN{#1}\__clistex_head_clist_aux:n }
55 }
56 \cs_new:Nn
57 \__clistex_head_clist_aux:n{#1}

```

(End definition for \clistex_keys_set:n and \clistex_info_clist:nn. These functions are documented on page 4.)

2 name

```

\__clistex_rule_name:n
\__clistex_instance_name:nnn
\__clistex_instance_signature:n
\__clistex_rule_sequence_name:n
58 \cs_new:Npn
59 \__clistex_rule_name:n
60 #1 % <rules>
61 {rule_#1}
62 \cs_new:Npn
63 \__clistex_instance_name:nn
64 #1 % <rules>
65 #2 % <cs name>
66 {instance_#1_#2}
67 \cs_new:Npn
68 \__clistex_instance_name:nnn
69 #1 % <rule>
70 #2 % <next rules>
71 #3 % <cs name>
72 {\__clistex_instance_name:nn{#1_#2}{#3}}
73 \cs_new:Npn
74 \__clistex_instance_signature:n
75 #1 % <signature>
76 {n#1w}

```

(End definition for __clistex_rule_name:n and others.)

3 c

```

77 \cs_new:Npn
78 \__clistex_c:n
79 #1 % <name>
80 {\__clistex_#1}
81 \cs_generate_variant:Nn\__clistex_c:n{e}
82 \cs_new:Npn
83 \__clistex_c:nn
84 #1 % <name>
85 #2 % <signature>
86 {\__clistex_c:n{#1:#2}}

```

```

87 \cs_generate_variant:Nn\__clistex_c:nn{e, ee}
88 \cs_new:Npn
89 \__clistex_bound_cs_c:nn
90 #1 % <name>
91 #2 % <signature>
92 {#1:#2n}
93 \cs_new:Npn
94 \__clistex_rule_c:n
95 #1 % <rule>
96 {%
97   \__clistex_c:en
98   { \__clistex_rule_name:n{#1} }
99   {nnnnnnnn}
100 }
101 \cs_new:Npn
102 \__clistex_instance_c:nn
103 #1 % <rules>
104 #2 % <cs name>
105 { \__clistex_c:e
106   { \__clistex_instance_name:nn{#1}{#2} } }
107 \cs_generate_variant:Nn\__clistex_instance_c:nn{e}
108 \cs_new:Npn
109 \__clistex_instance_c:nnn
110 #1 % <rules>
111 #2 % <cs name>
112 #3 % <signature>
113 {%
114   \__clistex_c:ee
115   { \__clistex_instance_name:nn{#1}{#2} }
116   { \__clistex_instance_signature:n{#3} }
117 }
118 \cs_generate_variant:Nn\__clistex_instance_c:nnn{e, nne}
119 \cs_new:Npn
120 \__clistex_instance_c_this:nnnn
121 #1 % <rule>
122 #2 % <next rules>
123 #3 % <cs name>
124 #4 % <signature>
125 { \__clistex_instance_c:enn
126   { \__clistex_rule_link:nn{#1}{#2}}{#3}{#4} }

```

4 rule_link

```

127 \cs_new:Npn
128 \__clistex_rule_link:nn
129 #1 % <rule 1>
130 #2 % <rule 2>
131 {#1_#2}
132 \cs_new:Npn
133 \__clistex_rule_link:n
134 #1 % <{rule{1}}...>
135 {%
136   \__clistex_rule_link:w#1\q_recursion_tail\q_recursion_stop
137 }

```

```

138 \cs_generate_variant:Nn\__clistex_rule_link:n{e}
139 \cs_new:Npn
140 \__clistex_rule_link:w
141 #1
142 \q_recursion_stop
143 {%
144   \quark_if_recursion_tail_stop:n{#1}
145   \__clistex_rule_link:nw #1 \q_recursion_stop}
146 \cs_new:Npn
147 \__clistex_rule_link:nw
148 #1 % <rules>
149 #2 % <{rule{1}}...>
150 \q_recursion_stop
151 {%
152   \quark_if_recursion_tail_stop_do:nn{#2}{#1}
153   \__clistex_rule_link:nnw{#1}#2\q_recursion_stop}
154 \cs_generate_variant:Nn\__clistex_rule_link:nw{e}
155 \cs_new:Npn
156 \__clistex_rule_link:nnw
157 #1 % <rules>
158 #2 % <rule{1}>
159 #3 % <{rule{2}}...>
160 \q_recursion_stop
161 {%
162   \__clistex_rule_link:ew
163   {%
164     \__clistex_rule_link:nn
165     {#1} % <rule 1>
166     {#2} % <rule 2>
167   } % <rules>
168   #3 % <{rule{1}}...>
169   \q_recursion_stop
170 }

```

5 inline

```

171 \cs_new_protected:Nn
172 \__clistex_inline_set_exp_nnnnot:Nn
173 {\cs_set:Nn#1
174   {\exp_not:n
175     {\exp_not:n
176       {\exp_not:n{#2}}}}}
177 \cs_generate_variant:Nn\__clistex_inline_set_exp_nnnnot:Nn{c}
178 \cs_new:Nn\__clistex_inline_c:n{\__clistex_#1:n}
179 \cs_new:Nn\__clistex_inline_use:n
180 {%~^A BUG
181   \use:c{\__clistex_inline_c:n{#1}}}
182 \cs_new_protected:Nn
183 \__clistex_inline_set_exp_nnnnot:nn
184 {\__clistex_inline_set_exp_nnnnot:cn
185   {\__clistex_inline_c:n{#1}}{#2}}
186 \msg_new:nnn{\__clistex}
187 {inline-empty-N}
188 {instance-signature-must-be-empty-or-N;~got~'~#1'~}

```

```

189 \msg_new:nnn{__clistex}
190 {inline-empty-args}
191 {instance-signature=empty;~so~should~args=#1}

```

6 eval

```

\clistex:nnn
\clistex_inline:nnn
192 \msg_new:nnn{__clistex}{key}
193 {no-match-for~#1~in~instance-or~instance~sequence}
194 \msg_new:nnn{__clistex}{signature-mismatch}
195 {instance-signature-must-be~#1;~instances:~#2}
196 \cs_new_protected:Npn
197 \clistex_inline:nnn
198 #1 % <clist>
199 #2 % <instances>
200 #3 % <empty|code using #1>
201 {%^^A
202   \bool_if:nTF
203   { \__clistex_instance_signature_p:nn{#2}{N} }
204   {%^^A
205     \__clistex_inline_set_exp_nnnot:nn{a}{#3}
206     \clistex:nnn
207     {#1} % <clist>
208     {#2} % <key 1>
209     {\__clistex_a:n}
210   }
211   {%^^A
212     \bool_if:nTF
213     { \__clistex_instance_signature_p:nn{#2}{ } }
214     {%^^A
215       \tl_if_empty:nTF
216       {#3}
217       {%^^A
218         \clistex:nnn
219         {#1} % <clist>
220         {#2} % <key 1>
221         {}
222       }
223       {%^^A
224         \msg_error:nnnn{__clistex}
225         {inline-empty-args}
226         {#3}
227       }
228     }
229     {%^^A
230       \msg_error:nnnn{__clistex}
231       {inline-empty-N}
232       {#2}
233     }
234   }
235 }
236 \cs_new:Npn
237 \clistex:nnn
238 % ^^A Warning: trailing ', ' inside #2 => Error

```

```

239 #1 % <clist>
240 #2 % <key,...>
241 #3 % <arguments>
242 {%
243   \__clistex_eval:nenn
244   {#2} % <instance key>,...
245   {\tl_if_head_is_group_p:n{#1}} % <head is group>
246   {#3} % <arguments>
247   {#1} % <clist>
248 }
249 \cs_generate_variant:Nn\clistex:nnn{e,f,x}
250 \cs_new:Npn
251   \__clistex_eval:nnnn
252   #1 % <instance key>,...
253   #2 % <head is group>
254   #3 % <arguments>
255   #4 % <clist>
256   {%
257     \exp_args:Ne
258     \__clistex_eval_aux:nnnn
259     {\__clistex_instance_expand:n{#1}}
260     {#2} % <head is group>
261     {#3} % <arguments>
262     {#4} % <clist>
263   }
264   \cs_new:Npn
265     \__clistex_eval_aux:nnnn
266     #1 % <instance key>,...
267     #2 % <head is group>
268     #3 % <arguments>
269     #4 % <clist>
270     {%
271       \__clistex_eval:nnnw
272       {#2} % <head is group>
273       {#3} % <arguments>
274       {#4} % <clist>
275       #1 % <instance key>,...
276       , \q_recursion_tail
277       \q_recursion_stop
278     }
279   \cs_generate_variant:Nn\__clistex_eval:nnnn{ ne }
280   \cs_new:Npn
281     \__clistex_eval:nnnw
282     #1 % <head is group>
283     #2 % <arguments>
284     #3 % <clist>
285     #4 % <instance key>
286     \q_recursion_stop
287     {%
288       \quark_if_recursion_tail_stop:n{#4}
289       \__clistex_eval:nnnw
290       {#1} % <head is group>
291       {#2} % <arguments>
292       {#3} % <clist>

```



```

293   #4 % <instance key>
294   \q_recursion_stop
295 }
296 \cs_new:Npn
297   \__clistex_eval:nnnnw
298   #1 % <head is group>
299   #2 % <arguments>
300   #3 % <clist>
301   #4 % <instance key>
302   , #5 % <instance key,...>
303   \q_recursion_stop
304   {%
305     \exp_last_unbraced:Ne
306     \__clistex_eval:nnnnnn
307     { \__clistex_instance_get:n{#4} }
308     {#1}{#2}{#3}
309     \__clistex_eval:nnnw
310     {#1} % <head is group>
311     {#2} % <arguments>
312     {#3} % <clist>
313     #5 % <instance key>
314     \q_recursion_stop
315   }
316 \cs_new:Npn
317   \__clistex_eval:nnnnnn
318   #1 % <rule sequence>
319   #2 % <cs name>
320   #3 % <signature>
321   #4 % <head is group>
322   #5 % <arguments>
323   #6 % <clist>
324   {%
325     \exp_args:Ne
326     \clistex_use_w:nnnn
327     { \__clistex_rule_sequence_name:n{#1} } % <rule sequence>
328     {#2} % <cs name>
329     {#3} % <signature>
330     {#4} % <head is group>
331     #5
332     #6, \q_recursion_tail\q_recursion_stop
333   }

```

(End definition for `\clistex:nnn` and `\clistex:inline:nnn`. These functions are documented on page 5.)

7 chain

```

334 \msg_new:nnn{\__clistex}
335 {chain}{unknown~chain~tag~#1}
336 \cs_new_protected:Npn
337   \__clistex_append:NNN
338   #1 % <new>
339   #2 % <\__clistex_append(?:_inline):nnn>
340   #3 % <\clistex(?_inline):nnnn>

```

```

341 {%^^A
342   #1
343   #2
344   {%^^A
345     \clistex:nnn{##1}{##2}{##3}
346     #3{##1}
347   }
348 }
349 \__clistex_append:NNN
350 \cs_new:Nn
351 \__clistex_append:nnn
352 \clistex:nnnn
353 \__clistex_append:NNN
354 \cs_new_protected:Nn
355 \__clistex_append_inline:nnn
356 \clistex_inline:nnnn
357 \cs_new_protected:Npn
358 \__clistex_nest:NNN
359 #1 % <new>
360 #2 % <\__clistex_nest(?:_inline):nnn>
361 #3 % <\clistex(?:_inline):nnnn>
362 {%^^A
363   #1
364   #2
365   {%^^A
366     \exp_args:Ne
367     #3{ \clistex:nnn{##1}{##2}{##3} }
368   }
369 }
370 \__clistex_nest:NNN
371 \cs_new:Nn
372 \__clistex_nest:nnn
373 \clistex:nnnn
374 \__clistex_nest:NNN
375 \cs_new_protected:Nn
376 \__clistex_nest_inline:nnn
377 \clistex_inline:nnnn
378 \cs_new_protected:Npn
379 \__clistex_join:NNNN
380 #1 % <new>
381 #2 % <\__clistex_join(?:_inline):nnnn>
382 #3 % <\__clistex_join(?:_inline):nnn>
383 #4 % <\clistex(?:_inline):nnnn>
384 {%^^A
385   #1
386   #2
387   { #4{##1,##2}{##3}{##4} }
388   #1
389   #3
390   { #2{\clistex:nnn{##1}{##2}{##3}} }
391 }
392 \__clistex_join:NNNN
393 \cs_new:Nn
394 \__clistex_join:nnnn

```

```

395 \__clistex_join:nnn
396 \clistex:nnnn
397 \__clistex_join:NNNN
398 \cs_new_protected:Nn
399 \__clistex_join_inline:nnnn
400 \__clistex_join_inline:nnn
401 \clistex_inline:nnnn
402 \cs_new_protected:Npn
403 \__clistex_chain:NNNNN
404 #1 % <new>
405 #2 % <__clistex_chain(?:_inline):nnnn>
406 #3 % <__clistex_append(?:_inline):nnn>
407 #4 % <__clistex_nest(?:_inline):nnn>
408 #5 % <__clistex_join(?:_inline):nnn>
409 {%^^A
410   #1
411   #2
412   {%^^A
413     \str_case:nnTF
414     {##4}
415     {%^^A
416       {end}
417       { \clistex:nnn{##1}{##2}{##3} }
418       {append}
419       { #3{##1}{##2}{##3} }
420       {nest}
421       { #4{##1}{##2}{##3} }
422       {join}
423       { #5{##1}{##2}{##3} }
424     }
425     {}
426     { \msg_error:nnn{__clistex}{chain}{##4} }
427   }
428 }
429 \__clistex_chain:NNNNN
430 \cs_new:Nn
431 \clistex:nnnn
432 \__clistex_append:nnn
433 \__clistex_nest:nnn
434 \__clistex_join:nnn
435 \__clistex_chain:NNNNN
436 \cs_new_protected:Nn
437 \__clistex_inline_aux:nnnn
438 \__clistex_append_inline:nnn
439 \__clistex_nest_inline:nnn
440 \__clistex_join_inline:nnn
441 \cs_new_protected:Npn
442 \clistex_inline:nnnn
443 #1 % <clist>
444 #2 % <inst>
445 #3 % <args>
446 #4 % <chain>
447 {%^^A
448   \bool_if:nTF

```

```

449 { \_clistex_instance_signature_p:nn{#2}{N} }
450 {%^^A
451   \_clistex_inline_set_exp_nnnot:nn{a}{#3}
452   \_clistex_inline_aux:nnnn{#1}{#2}{\_clistex_a:n}{#4}
453 }
454 { \_clistex_inline_aux:nnnn{#1}{#2}{-}{#4} }
455 }

```

8 use_w

`\clistex_use_w_group:nnnnnn` For use inside `<code>` inside **rule**

```

\clistex_use_w:nnnn
\clistex_use_w:nnnnn
456 \cs_new:Npn
457 \clistex_use_w_group:nnnnnn
458 #1 % <rule sequence>
459 #2 % <cs name>
460 #3 % <signature>
461 #4 % <head is group>
462 #5 % <arguments>
463 #6 % <clist head>
464 {%
465   \clistex_use_w:nnnn
466   {#1}{#2}{#3}
467   {#4}#5{#6}
468 }
469 \cs_new:Npn
470 \clistex_use_w:nnnn
471 #1 % <rule sequence>
472 #2 % <cs name>
473 #3 % <signature>
474 #4 % <head is group>
475 {%
476   \use:c{ \_clistex_instance_c:nnn{#1}{#2}{#3} }{#4}
477 }
478 \cs_generate_variant:Nn\clistex_use_w:nnnn{nnne}
479 \cs_new:Npn
480 \clistex_use_w:nnnnn
481 #1 % <rule>
482 #2 % <next rule sequence>
483 #3 % <cs name>
484 #4 % <signature>
485 #5 % <head is group>
486 {%
487   \use:c{%
488     \_clistex_instance_c_this:nnnn
489     {#1} % <rule>
490     {#2} % <next rules>
491     {#3} % <cs name>
492     {#4} % <signature>
493   }{#5}
494 }
495 \cs_generate_variant:Nn\clistex_use_w:nnnnn{nnnnne}

```

(End definition for `\clistex_use_w_group:nnnnnn`, `\clistex_use_w:nnnn`, and `\clistex_use_w:nnnnn`.
These functions are documented on page 5.)

`\clistex_bound_cs_group:nnnnn`

```

496 \cs_new:Npn
497 \clistex_bound_cs_group:nnnnn
498 #1 % <cs name>
499 #2 % <signature>
500 #3 % <group (bool)>
501 #4 % <arguments>
502 #5 % <clist>
503 {\__clistex_bound_cs:nnne{#1}{#2}{#4}{\bool_if:nTF{#3}{#{#5}}{#5}}}
504 \cs_generate_variant:Nn\clistex_bound_cs_use_group:nnnnn{nnenn}
505 \cs_new:Npn
506 \__clistex_bound_cs:nnnn
507 #1 % <cs name>
508 #2 % <signature>
509 #3 % <arguments>
510 #4 % <clist>
511 { \use:c{\__clistex_bound_cs_c:nn{#1}{#2}}#3{#4} }
512 \cs_generate_variant:Nn\__clistex_bound_cs:nnnn{nnne}

```

(End definition for `\clistex_bound_cs_group:nnnnn`. This function is documented on page 5.)

9 rule

`rule`

```

513 \keys_define:nn{ __clistex }
514 { rule.code:n = \__clistex_rule:nn#1 }

```

(End definition for `rule`. This function is documented on page 4.)

`__clistex_rule:nn`

```

515 \prop_new:N\__clistex_rule_clist
516 \__clistex_info_clist_put:nn{rule}{__clistex_rule_clist}
517 \cs_new_protected:Npn
518 \__clistex_rule:nn
519 #1 % <rule>
520 #2 % <code>
521 {%
522 \clist_gput_right:Nn\__clistex_rule_clist{#1}
523 \exp_args:Nno
524 \cs_new_protected:cn
525 { \__clistex_rule_c:n{#1} }
526 {%
527 \__clistex_rule_apply:nnnnnnnn
528 {#1} % {<rule>}
529 {#2} % {<code>}
530 {##1} % <next rule>
531 {##2} % <cs name>
532 {##3} % <signature>
533 {##4}{##5}{##6} % <head is group>
534 % ^^A <arguments>
535 % ^^A <clist head>
536 {##7} % <clist rest>
537 {##8} % <parameters>

```

```

538 }
539 }
540 % ^^A ##1 % <next rules>
541 % ^^A ##2 % <cs name>
542 % ^^A ##3 % <signature>
543 % ^^A ##4 % <head is group>
544 % ^^A ##5 % <arguments>
545 % ^^A ##6 % <clist head>
546 % ^^A ##7 % <clist rest>
547 % ^^A ##8 % <parameters>
548 \cs_new_protected:Npn
549 \__clistex_rule_apply:nnnnnnnn
550 #1 % <rule>
551 #2 % <code>
552 #3 % <next rules>
553 #4 % <cs name>
554 #5 % <signature>
555 #6 % {<head is group>}{<arguments>}{<clist head>}
556 #7 % <clist rest>
557 #8 % <parameters>
558 {%
559 \__clistex_rule_apply:ennnnnnn
560 { \__clistex_instance_c_this:nnnn{#1}{#3}{#4}{#5}}
561 {#2}#6{#7}{#8}
562 }
563 \cs_new_protected:Npn
564 \__clistex_rule_apply:nnnnnnnn
565 #1 % <instance>
566 #2 % <code>
567 #3 % <head is group>
568 #4 % <arguments>
569 #5 % <clist head>
570 #6 % <clist rest>
571 #7 % <parameters>
572 {%
573 \cs_if_exist:cF{#1}
574 {% ^^A
575 \cs_new:cpn{#1}
576 #3#7#5, #6\q_recursion_stop % <parameters>
577 {#2}
578 }
579 }
580 \cs_generate_variant:Nn\__clistex_rule_apply:nnnnnnnn{e}

```

(End definition for __clistex_rule:nn.)

10 rule template

```

581 \cs_new:Nn
582 \__clistex_quark_if_recursion_tail_stop:nn
583 {\quark_if_recursion_tail_stop:n{#1#2}}
584 \cs_generate_variant:Nn\__clistex_quark_if_recursion_tail_stop:nn{e}

```

rule_if_rest_is_tail_eval_else

```

585 \keys_define:nn{ __clistex }
586 {%
587   rule_if_rest_is_tail_eval_else.code:n
588   = {\__clistex_rule_if_rest_is_tail_eval_else:nn#1}
589 }
590 \cs_new_protected:Npn
591 \__clistex_rule_if_rest_is_tail_eval_else:nn
592 #1 % <name>
593 #2 % <else code>
594 {%
595   % ^^A ##1 % <next rules>
596   % ^^A ##2 % <cs name>
597   % ^^A ##3 % <signature>
598   % ^^A ##4 % <head is group>
599   % ^^A ##5 % <arguments>
600   % ^^A ##6 % <clist head>
601   % ^^A ##7 % <clist rest>
602   % ^^A ##8 % <parameters>
603   \clistex_keys_set:n
604   {%
605     rule = {if_rest_is_tail_eval_else_#1}
606     {%
607       \quark_if_recursion_tail_stop_do:nn{##7}
608       {%
609         \clistex_bound_cs_group:nnnnn
610         {##2} % <cs name>
611         {##3} % <signature>
612         {##4} % <head is group>
613         {##5} % <arguments>
614         {##6} % <clist>
615       }
616       #2
617     }
618   }
619 }

```

(End definition for rule_if_rest_is_tail_eval_else. This function is documented on page 4.)

rule_if_empty_stop_else

```

620 \keys_define:nn
621 { __clistex }
622 {
623   rule_if_empty_stop_else.code:n
624   = {\__clistex_rule_if_empty_stop_else:nn#1}
625 }
626 \cs_new_protected:Npn
627 \__clistex_rule_if_empty_stop_else:nn
628 #1 % <name>
629 #2 % <else code>
630 {%
631   % ^^A ##1 % <next rules>
632   % ^^A ##2 % <cs name>
633   % ^^A ##3 % <signature>
634   % ^^A ##4 % <head is group>

```

```

635 % ^^A ##5 % <arguments>
636 % ^^A ##6 % <clist head>
637 % ^^A ##7 % <clist rest>
638 % ^^A ##8 % <parameters>
639 \clistex_keys_set:n
640 {%
641   rule = {if_empty_stop_else_#1}
642   {%
643     \__clistex_quark_if_recursion_tail_stop:en
644     {\bool_if:nTF{##4}{##6}{##6}}{##7}
645     #2
646   }
647 }
648 }

```

(End definition for rule_if_empty_stop_else. This function is documented on page 4.)

11 instantiate

__clistex_instantiate:nnnn

```

649 \cs_new_protected:Npn
650 \__clistex_instantiate:nnnn
651 #1 % <rule>
652 #2 % <next rules>
653 #3 % <cs name>
654 #4 % <signature>
655 {%
656   \exp_args:Ne
657   \__clistex_instantiate:nnnnn
658   {\tl_count:n{#4}} % <signature arity>
659   {#1} % <rule>
660   {#2} % <next rules>
661   {#3} % <cs name>
662   {#4} % <signature>
663 }
664 \cs_new_protected:Npn
665 \__clistex_instantiate:nnnnn
666 #1 % <signature arity>
667 #2 % <rule>
668 #3 % <next rules>
669 #4 % <cs name>
670 #5 % <signature>
671 {%^^A
672   \__clistex_instantiate:eeeeennn
673   { \erw_parameter:n{ 1 } } % <head is group>
674   { \erw_parameter:ne{2}{ #1 } } % <parameters>
675   { \erw_parameter:e{ \int_eval:n{#1+2} } } % <clist head>
676   { \erw_parameter:e{ \int_eval:n{#1+3} } } % <clist rest>
677   { \erw_argument:ne{2}{ #5 } } % <arguments>
678   { #2 } % <rule>
679   { #3 } % <next rules>
680   { #4 } % <cs name>
681   { #5 } % <signature>

```



```

682 }
683 \cs_new:Npn
684   \__clistex_instantiate:nnnnnnnn
685   #1 % <head is group>
686   #2 % <parameters>
687   #3 % <clist head>
688   #4 % <clist rest>
689   #5 % <arguments>
690   #6 % <rule>
691   #7 % <next rules>
692   #8 % <cs name>
693   #9 % <signature>
694   {%
695     \use:c{ \__clistex_rule_c:n{#6} }
696     {#7} % <next rules>
697     {#8} % <cs name>
698     {#9} % <signature>
699     {#1} % <head is group>
700     {#2} % <arguments>
701     {#3} % <clist head>
702     {#4} % <clist rest>
703     {#2} % <parameters>
704   }
705   \cs_generate_variant:Nn\__clistex_instantiate:nnnnnnnn{eeeeee}

```

(End definition for __clistex_instantiate:nnnn.)

12 property

rule_sequence

```

706 \cs_new:Npn
707   \__clistex_rule_sequence_name:n
708   #1 % <rule sequence>
709   {%
710     \__clistex_rule_link:e
711     {\__clistex_rule_sequence_get:n{#1}{null}}
712   }
713   \keys_define:nn{__clistex}
714   { rule_sequence.code:n = \__clistex_rule_sequence_from_keyval:n{#1} }
715   \prop_new:N\__clistex_rule_sequence_prop
716   \__clistex_info_prop_put:nn{rule_sequence}{__clistex_rule_sequence_prop}
717   \cs_new_protected:Npn
718     \__clistex_rule_sequence_from_keyval:n
719     #1 % <key = {{rule{1}}...>
720     {%
721       \prop_set_from_keyval:Nn
722       \__clistex_rule_sequence_prop{#1}
723     }
724   \cs_new:Npn
725     \__clistex_rule_sequence_get:n
726     #1 % <key>
727     {%
728       \exp_args:Ne

```

```

729 \__clistex_rule_sequence_aux:n
730 {%
731   \prop_item:Nn
732   \__clistex_rule_sequence_prop{#1}
733 }
734 }
735 \cs_new:Npn
736 \__clistex_rule_sequence_aux:n
737 #1 % <value>
738 {%
739   \prop_if_in:NnTF
740   \__clistex_rule_sequence_prop
741   {#1}
742   {\__clistex_rule_sequence_get:n{#1}}
743   {#1}
744 }

```

(End definition for rule_sequence. This function is documented on page 4.)

\clistex_signature:n
\clistex_instance_p:n

```

745 \prg_new_conditional:Npnn
746 \clistex_instance:n
747 #1
748 {p}
749 {\prop_if_in:NnTF
750   \__clistex_instance_prop{#1}
751   {\prg_return_true:}
752   {\prg_return_false:}
753 }
754 \msg_new:nnn{__clistex}{instance-not}{#1~is~not~an~instance}
755 \msg_new:nnn{__clistex}{key-conflict}{key~#1~already~exists~in~prop~#2}
756 \prop_new:N\__clistex_instance_prop
757 \__clistex_info_prop_put:nn{instance}{__clistex_instance_prop}
758 \cs_new_protected:Npn
759 \__clistex_instance_put:nnnn
760 #1 % <key>
761 #2 % <rule sequence>
762 #3 % <name>
763 #4 % <signature>
764 {%
765   \prop_gput:Nnn
766   \__clistex_instance_prop{#1}
767   { {#2}{#3}{#4} }
768 }
769 \cs_new:Npn
770 \__clistex_instance_get:n
771 #1 % <key>
772 { \prop_item:Nn\__clistex_instance_prop{#1} }
773 \cs_new:Nn
774 \clistex_signature:n
775 {%^^A
776   \bool_if:nTF
777   { \clistex_instance_p:n{#1} }
778   { \__clistex_instance_signature_get:n{#1} }

```

```

779 { \msg_error:nnn{__clistex}{instance-not}{#1} }
780 }
781 \cs_new:Npn
782 \__clistex_instance_signature_get:n
783 #1 % <instance>
784 {\exp_last_unbraced:Ne\use_iii:nnn
785  {\__clistex_instance_get:n{#1}}}
786 \cs_new:Npn
787 \__clistex_instance_expand:n
788 #1 %^A <instance(?:_sequence)_1,...>
789 {%^A
790  \__clistex_instance_expand:w
791  #1, \q_recursion_tail
792  \q_recursion_stop
793 }
794 \cs_new:Npn
795 \__clistex_instance_expand:w
796 #1 %^A <instance(?:_sequence)_1,...>
797 ,#2
798 \q_recursion_stop
799 {
800  \quark_if_recursion_tail_stop:n{#1#2}
801  \__clistex_instance_expand:nw#1, #2\q_recursion_stop
802 }
803 \cs_new:Npn
804 \__clistex_instance_expand:nw
805 #1 % <head>
806 , #2 % <rest>
807 \q_recursion_stop
808 {
809  \bool_if:nTF
810  {\clistex_instance_sequence_p:n{#1}}
811  {%^A
812   \exp_args:Ne
813   \__clistex_instance_expand:n
814   { \__clistex_instance_sequence_get:n{#1} }
815  }
816  {%
817   \bool_if:nTF
818   {\clistex_instance_p:n{#1}}
819   {#1}
820   {\msg_error:nnn{__clistex}{neither-inst-seq}{#1}}
821  }
822  \quark_if_recursion_tail_stop:n{#2},%^A comma
823  \__clistex_instance_expand:nw#2\q_recursion_stop
824 }
825 \msg_new:nnn{__clistex}{neither-inst-seq}
826 {#1~is~neither~an~instance~nor~a~sequence}
827 \prg_new_conditional:Npnn
828 \__clistex_instance_signature:nn
829 #1 % <instance_1,...>
830 #2 % <signature>
831 {p}
832 {%^A

```

```

833 \bool_if:nTF
834 {
835   \exp_args:Ne
836   \__clistex_instance_signature_aux_p:nn
837   {%^^A
838     \exp_args:Ne
839     \clist_map_function:nN
840     { \__clistex_instance_expand:n{#1} }
841     \clistex_signature:n
842   }
843   {#2}
844 }
845 {\prg_return_true:}
846 {\prg_return_false:}
847 }
848 \prg_new_conditional:Npnn
849 \__clistex_instance_signature_aux:nn
850 #1 % <signature_1,...>
851 #2 % <signature>
852 {p}
853 {%
854   \tl_if_empty:nTF
855   {#1}
856   {%^^A
857     \tl_if_empty:nTF{#2}
858     {\prg_return_true:}
859     {\prg_return_false:}
860   }
861   {%^^A
862     \bool_if:nTF
863     {%^^A
864       \erw_and_tl_p:nn
865       { \str_if_eq_p:nn{#2} }
866       { #1 }
867     }
868     {\prg_return_true:}
869     {\prg_return_false:}
870   }
871 }

```

(End definition for `\clistex_signature:n` and `\clistex_instance_p:n`. These functions are documented on page 4.)

```

instance_sequence
\clistex_instance_sequence_p:n
872 \keys_define:nn{ __clistex }
873 {%^^A
874   instance_sequence.code:n
875   = {%^^A
876     \clist_map_function:nN{#1}
877     \__clistex_instance_sequence_put:n
878   }
879 }
880 \prg_new_conditional:Npnn
881 \clistex_instance_sequence:n

```

```

882 #1
883 {p}
884 {%
885   \prop_if_in:NnTF
886   \__clistex_instance_sequence_prop{#1}
887   {\prg_return_true:}
888   {\prg_return_false:}
889 }
890 \prop_new:N
891 \__clistex_instance_sequence_prop
892 \__clistex_info_prop_put:nn{instance_sequence}{\__clistex_instance_sequence_prop}
893 \cs_new:Nn\__clistex_first_braced:nn{#{1}}
894 \cs_new:Nn\__clistex_instance_sequence_keys:
895 {%
896   \prop_map_function:NN
897   \__clistex_instance_sequence_prop
898   \__clistex_first_braced:nn
899 }
900 % ^^A\cs_new_protected:Npn
901 % ^^A\__clistex_instance_sequence_put:n
902 % ^^A#1 % <{key}{key{1},...}>
903 % ^^A{ \__clistex_instance_sequence_put:nn#1 }
904 \cs_new_protected:Npn
905 \__clistex_instance_sequence_put:n
906 #1 % <{signature}{prefix key}{prefix key{1},...}>
907 { \__clistex_instance_sequence_put:nnn#1 }
908 \cs_new:Npn
909 \__clistex_instance_sequence_value:nn
910 #1 % <signature>
911 #2 % <key prefix 1,...>
912 {%
913   \exp_args:Nne
914   \erw_clist_tl:nn{\c_false_bool}
915   {%^^A
916     \clist_map_tokens:nn
917     {#2}
918     { \__clistex_instance_sequence_value_aux:nn{#1} }
919   }
920 }
921 \cs_new:Nn
922 \__clistex_instance_sequence_value_aux:nn
923 { {\__clistex_instance_key:nn{#2}{#1}} }
924 \cs_new_protected:Npn
925 \__clistex_instance_sequence_put:nnn
926 #1 % <signature>
927 #2 % <prefix key>
928 #3 % <prefix key{1}>,...
929 {%^^A
930   \exp_args:Nee
931   \__clistex_instance_sequence_put:nn
932   { \__clistex_instance_key:nn{#2}{#1} }
933   { \__clistex_instance_sequence_value:nn{#1}{#3} }
934 }
935 \cs_new_protected:Npn

```

```

936 \__clistex_instance_sequence_put:nn
937 #1 % <key>
938 #2 % <instance key{1}>,...
939 {%
940   \prop_if_in:NnTF
941   \__clistex_instance_prop{#1}
942   {\msg_error:nnnn{\__clistex}{key-conflict}{#1}{instance}}
943   {%
944     \prop_gput:Nnn
945     \__clistex_instance_sequence_prop{#1}
946     { #2 }
947   }
948 }
949 \cs_new:Nn
950 \clistex_instance_sequence:n
951 {\__clistex_instance_sequence_get:n{#1}}
952 \cs_new:Npn
953 \__clistex_instance_sequence_get:n
954 #1 % <key>
955 {\prop_item:Nn\__clistex_instance_sequence_prop{#1}}

```

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

13 instance

```

instance
\clistex_instance_key:nn
956 \keys_define:nn{\__clistex}
957 { instance.code:n = \clist_map_function:nn{#1} \__clistex_instance:n }
958 \cs_new_protected:Npn
959 \__clistex_instance:n
960 % ^^A#1 % {key prefix}{<rule sequence>}{<cs name>}{<signature>}
961 #1 % {<signature>}{key prefix}{<rule sequence>}{<cs name>}
962 { \__clistex_instance:nnnn#1 }
963 \cs_new_protected:Npn
964 \__clistex_instance:nnnn
965 % ^^A#1 % <key prefix>
966 % ^^A#2 % <rule sequence>
967 % ^^A#3 % <cs name>
968 % ^^A#4 % <signature>
969 #1 % <signature>
970 #2 % <key prefix>
971 #3 % <rule sequence>
972 #4 % <cs name>
973 {%
974   \exp_args:Ne
975   \__clistex_instance_aux:nnnn
976   { \clistex_instance_key:nn{#2}{#1} }
977   {#3}{#4}{#1}
978 }
979 \cs_new:Npn
980 \clistex_instance_key:nn
981 #1 % <key prefix>

```

```

982 #2 % <signature>
983 {#1:#2}
984 \cs_new_protected:Npn
985 \__clistex_instance_aux:nnnn
986 #1 % <key>
987 #2 % <rule sequence>
988 #3 % <signature>
989 #4 % <cs name>
990 {%
991 \__clistex_instance_put:nnnn{#1}{#2}{#3}{#4}
992 \__clistex_instance_using_key:nnn{#2}{#3}{#4}
993 }
994 \cs_new_protected:Npn
995 \__clistex_instance_using_key:nnn
996 #1 % <rule sequence>
997 #2 % <cs name>
998 #3 % <signature>
999 {%
1000 \__clistex_instance_using_list:enn
1001 { \__clistex_rule_sequence_get:n{#1}{null} } % <{rule{1}}...>
1002 {#2} % <cs name>
1003 {#3}% <signature>
1004 }
1005 \cs_new_protected:Npn
1006 \__clistex_instance_using_list:nnn
1007 #1 % <{rule{1}}{rule{2}}...>
1008 #2 % <cs name>
1009 #3 % <signature>
1010 {%
1011 \exp_last_unbraced:Ne
1012 \__clistex_instance_backward:nnnnn
1013 {%
1014 { \tl_count:n{#3} } % <signature arity>
1015 \erw_last:n{#1} % <rule{n}>
1016 { \erw_remove_first:e{\tl_reverse:n{#1}} } % <{rule{n-1}}{rule{n-2}}...>
1017 }
1018 { #2 } % <cs name>
1019 { #3 } % <signature>
1020 }
1021 \cs_generate_variant:Nn\__clistex_instance_using_list:nnn{enn}
1022 \msg_new:nnn{\__clistex}{null}
1023 {clistex~expects~'null'~as~the~last~rule;~got~'#1'}
1024 \cs_new_protected:Npn
1025 \__clistex_instance_backward:nnnnn
1026 #1 % <signature arity>
1027 #2 % <rule{n}>
1028 #3 % <{rule{n-1}}{rule{n-2}}...>
1029 #4 % <cs name>
1030 #5 % <signature>
1031 {%
1032 \str_case:nnTF{#2}
1033 { {null}{ } }
1034 {%
1035 \__clistex_instance_backward:nnnw

```

```

1036     {#2} % <next rules>
1037     {#4} % <cs name>
1038     {#5} % <signature>
1039     #3\q_recursion_tail % <{rule{n}}{rule{n-1}}...>
1040     \q_recursion_stop
1041   }
1042   {%
1043     \msg_error:nnn{__clistex}
1044     {null}
1045     {#2}
1046   }
1047 }
1048 \cs_generate_variant:Nn\__clistex_instance_backward:nnnnn{eee}
1049 \cs_new_protected:Npn
1050   \__clistex_instance_backward:nnnw
1051   #1 % <next rules>
1052   #2 % <cs name>
1053   #3 % <signature>
1054   #4 % <{rule{n}}{rule{n-1}}...>
1055   \q_recursion_stop
1056   {%
1057     \quark_if_recursion_tail_stop:n{#4}
1058     \__clistex_instance_backward:nnnnw
1059     {#1} % <next rules>
1060     {#2} % <cs name>
1061     {#3} % <signature>
1062     #4 % <rule{n}>
1063     % <{rule{n-1}}...>
1064     \q_recursion_stop
1065   }
1066 \cs_generate_variant:Nn\__clistex_instance_backward:nnnw{e}
1067 \cs_new_protected:Npn
1068   \__clistex_instance_backward:nnnnw
1069   #1 % <next rules>
1070   #2 % <cs name>
1071   #3 % <signature>
1072   #4 % <rule{n}>
1073   #5 % <{rule{n-1}}...>
1074   \q_recursion_stop
1075   {%
1076     \__clistex_instantiate:nnnn
1077     {#4} % <rule>
1078     {#1} % <next rules>
1079     {#2} % <cs name>
1080     {#3} % <signature>
1081     \__clistex_instance_backward:ennw
1082     {\__clistex_rule_link:nn{#4}{#1}} % <next rules>
1083     {#2} % <cs name>
1084     {#3} % <signature>
1085     #5 % <{rule{n}}...>
1086     \q_recursion_stop
1087   }

```

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

14 preset

14.1 rule

```
1088 \msg_new:nnn{__clistex}{tail}{expects~tail;~got~'#1'}
1089 % ^^A ##1 % <next rules>
1090 % ^^A ##2 % <cs name>
1091 % ^^A ##3 % <signature>
1092 % ^^A ##4 % <head is group>
1093 % ^^A ##5 % <arguments>
1094 % ^^A ##6 % <clist head>
1095 % ^^A ##7 % <clist rest>
1096 % ^^A ##8 % <args>
1097 \clistex_keys_set:n
1098 {%
1099   rule = {if_rest_is_tail_stop_else_eval_recurse}
1100   {%
1101     \quark_if_recursion_tail_stop:n{#7}
1102     \clistex_bound_cs_group:nnnnn
1103     {#2} % <cs name>
1104     {#3} % <signature>
1105     {#4} % <head is group>
1106     {#5} % <arguments>
1107     {#6} % <clist>
1108     \clistex_use_w:nnnne
1109     {if_rest_is_tail_stop_else_eval_recurse} % <rule>
1110     {#1} % <next rule rule sequence>
1111     {#2} % <cs name>
1112     {#3} % <signature>
1113     {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop % <head is group>
1114   },
1115   rule = {if_rest_is_tail_stop_else_forward_rest}
1116   {%
1117     \quark_if_recursion_tail_stop:n{#7}
1118     \clistex_use_w:nnne
1119     {#1}{#2}{#3}
1120     {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop
1121   },
1122   rule_if_empty_stop_else = {error}
1123   {%
1124     \msg_error:nnn{__clistex}{tail}{#6#7}
1125     \__clistex_empty:w{}\q_recursion_stop
1126   },
1127   rule_if_empty_stop_else = {forward_head}
1128   {%
1129     \bool_if:nTF{#4}
1130     {%
1131       \clistex_use_w_group:nnnnnn{#1}{#2}{#3}{#4}{#5}{#6}
1132       ,\q_recursion_tail\q_recursion_stop
1133     }
1134     {%
1135       \clistex_use_w:nnnn{#1}{#2}{#3}
1136       {#4}#5#6,\q_recursion_tail\q_recursion_stop
1137     }
1138   }
```

```

1138 },
1139 rule_if_empty_stop_else = {forward_rest}
1140 {%
1141   \clistex_use_w:nnne
1142   {#1}{#2}{#3}
1143   {\tl_if_head_is_group_p:n{#7}}#5#7\q_recursion_stop
1144 },
1145 rule_if_empty_stop_else = {forward_all}
1146 {%
1147   \bool_if:nTF{#4}
1148   {%
1149     \clistex_use_w_group:nnnnn{#1}{#2}{#3}{#4}{#5}{#6},
1150     #7\q_recursion_stop
1151   }
1152   {%
1153     \clistex_use_w:nnnn
1154     {#1}{#2}{#3}{#4}#5#6, #7\q_recursion_stop
1155   }
1156 },
1157 rule_if_rest_is_tail_eval_else = {error}
1158 {%
1159   \msg_error:nnn{__clistex}{tail}{#6}
1160   \__clistex_empty:w\q_recursion_stop
1161 },
1162 rule_if_rest_is_tail_eval_else = {stop}
1163 {%
1164   \__clistex_empty:w{}\q_recursion_stop
1165 },
1166 rule_if_rest_is_tail_eval_else = {recurse}
1167 {%
1168   \clistex_use_w:nnnne
1169   {if_rest_is_tail_eval_else_recurse} % <rule>
1170   {#1} % <next rule rule sequence>
1171   {#2} % <cs name>
1172   {#3} % <signature>
1173   {\tl_if_head_is_group_p:n{#7}} % <head is group>
1174   #5 % <argument>
1175   #7 % <clist>
1176   \q_recursion_stop
1177 }
1178 }

```

14.2 rule_sequence

```

1179 \clistex_keys_set:n
1180 {%
1181   rule_sequence =
1182   {%
1183     first =
1184     {
1185       {if_empty_stop_else_forward_head}
1186       {if_rest_is_tail_eval_else_error}
1187     },
1188     middle =
1189     {

```

```

1190     {if_empty_stop_else_forward_all}
1191     {if_rest_is_tail_stop_else_forward_rest}
1192     {if_rest_is_tail_stop_else_eval_recurse}
1193 },
1194 last =
1195 {
1196     {if_empty_stop_else_forward_all}
1197     {if_rest_is_tail_stop_else_forward_rest}
1198     {if_rest_is_tail_eval_else_recurse}
1199 },
1200 serial_second =
1201 {
1202     {if_empty_stop_else_forward_all}
1203     {if_rest_is_tail_stop_else_forward_rest}
1204     {if_rest_is_tail_eval_else_stop}
1205 },
1206 serial_last =
1207 {
1208     {if_empty_stop_else_forward_all}
1209     {if_rest_is_tail_stop_else_forward_rest}
1210     {if_rest_is_tail_stop_else_forward_rest}
1211     {if_rest_is_tail_eval_else_recurse}
1212 }
1213 }
1214 }

```

14.3 cs

```

1215 \msg_new:nnnn{__clistex}{text}{text~is~not~loaded}{amsmath}
1216 \cs_new:Nn\__clistex_unbrace_aux:n{#1}
1217 \erw_keys_set:n
1218 {
1219     clist_map_inline =
1220     {%
1221         {Nn}{apply}{#1{#2}},
1222         {Nn}{math}{\ensuremath{#1{#2}}},
1223         {Nn}{comma_map}{,\clist_map_function:nN#2#1},
1224         {Nn}{comma}{,{#1{#2}}},
1225         {Nn}{serial_math}{\text{,~}\ensuremath{#1{#2}}},
1226         {Nn}{serial_math_and}{\text{,~and~}\ensuremath{#1{#2}}},
1227         {Nn}{map}{\clist_map_function:nN#2#1},
1228         {Nn}{noindent}{\noindent},
1229         {n}{apply}{#1},
1230         {n}{math}{\ensuremath{#1}},
1231         {n}{comma_math}{,\ensuremath{#1}},
1232         {n}{newline}{\\#1},
1233         {n}{comma_unbrace}{,\__clistex_unbrace_aux:n#1},
1234         {n}{comma}{,{#1}},
1235         {n}{noindent}{\noindent},
1236         {n}{serial_and}{,~and~#1},
1237         {n}{serial_math_and}{\text{,~and~}\ensuremath{#1}},
1238         {n}{serial_math}{\text{,~}\ensuremath{#1}},
1239         {n}{serial}{,~#1},
1240         {n}{unbrace}{\__clistex_unbrace_aux:n#1}
1241     }

```

```

1242 {nnn}
1243 {
1244   \clist_gput_right:Nn\__clistex_helper_clist{#2:#1}
1245   \cs_new:cn{__clistex_#2:#1}{#3}
1246 }
1247 }

```

14.4 instance

```

1248 \clistex_keys_set:n
1249 {
1250   instance =
1251   {
1252     {N}{first_apply}{first}{__clistex_apply},
1253     {N}{first_map}{first}{__clistex_map},
1254     {N}{first_math}{first}{__clistex_math},
1255     {N}{first_noindent}{first}{__clistex_noindent},
1256     {N}{last_apply}{last}{__clistex_apply},
1257     {N}{last_comma_map}{last}{__clistex_comma_map},
1258     {N}{last_comma_math}{last}{__clistex_comma_math},
1259     {N}{last_comma}{last}{__clistex_comma},
1260     {N}{serial_last}{serial_last}{__clistex_comma},
1261     {N}{serial_second}{serial_second}{__clistex_comma},
1262     {N}{middle_apply}{middle}{__clistex_apply},
1263     {N}{middle_comma_map}{middle}{__clistex_comma_map},
1264     {N}{middle_comma_math}{middle}{__clistex_comma_math},
1265     {N}{middle_comma}{middle}{__clistex_comma},
1266     {N}{serial_last_math_and}{serial_last}{__clistex_serial_math_and},
1267     {N}{serial_middle_math}{middle}{__clistex_serial_math},
1268     {N}{serial_second_math_and}{serial_second}{__clistex_serial_math_and},
1269     {}{first_apply}{first}{__clistex_apply},
1270     {}{first_math}{first}{__clistex_math},
1271     {}{first_noindent}{first}{__clistex_noindent},
1272     {}{first_unbrace}{first}{__clistex_unbrace},
1273     {}{last_apply}{last}{__clistex_apply},
1274     {}{last_comma_math}{last}{__clistex_comma_math},
1275     {}{last_comma_unbrace}{last}{__clistex_comma_unbrace},
1276     {}{last_comma}{last}{__clistex_comma},
1277     {}{last_newline}{last}{__clistex_newline},
1278     {}{last_unbrace}{last}{__clistex_unbrace},
1279     {}{middle_apply}{middle}{__clistex_apply},
1280     {}{middle_comma_math}{middle}{__clistex_comma_math},
1281     {}{middle_comma_unbrace}{middle}{__clistex_comma_unbrace},
1282     {}{middle_comma}{middle}{__clistex_comma},
1283     {}{middle_newline}{middle}{__clistex_newline},
1284     {}{middle_unbrace}{middle}{__clistex_unbrace},
1285     {}{serial_last_and}{serial_last}{__clistex_serial_and},
1286     {}{serial_last_math_and}{serial_last}{__clistex_serial_math_and},
1287     {}{serial_middle_math}{middle}{__clistex_serial_math},
1288     {}{serial_middle}{middle}{__clistex_serial},
1289     {}{serial_second_and}{serial_second}{__clistex_serial_and},
1290     {}{serial_second_math_and}{serial_second}{__clistex_serial_math_and},
1291   }
1292 }

```

14.5 instance_sequence

```

1293 \clistex_keys_set:n
1294 {%
1295   instance_sequence =
1296   {
1297     {N}{apply}{first_apply, rest_apply},
1298     {N}{comma_map}{first_map, rest_comma_map},
1299     {N}{comma_math}{first_math, rest_comma_math},
1300     {N}{comma}{first_apply, rest_comma},
1301     {N}{rest_apply}{middle_apply, last_apply},
1302     {N}{rest_comma_map}{middle_comma_map, last_comma_map},
1303     {N}{rest_comma_math}{middle_comma_math, last_comma_math},
1304     {N}{rest_comma}{middle_comma, last_comma},
1305     {N}{serial_and}{first_apply, serial_rest_and},
1306     {N}{serial_math_and}{first_math, serial_rest_math_and},
1307     {N}{serial_rest_and}{serial_middle, serial_second_and, serial_last_and},
1308     % ^^A <one long entry>
1309     {N}
1310     {serial_rest_math_and}
1311     {serial_middle_math, serial_second_math_and, serial_last_math_and}
1312     % ^^A </one long entry>
1313     ,
1314     {}{apply}{first_apply, rest_apply},
1315     {}{comma_math}{first_math, rest_comma_math},
1316     {}{newline}{first_apply, rest_newline},
1317     {}{comma_unbrace}{first_unbrace, rest_comma_unbrace},
1318     {}{comma}{first_apply, rest_comma},
1319     {}{rest_apply}{middle_apply, last_apply},
1320     {}{rest_comma_math}{middle_comma_math, last_comma_math},
1321     {}{rest_newline}{middle_newline, last_newline},
1322     {}{rest_comma_unbrace}{middle_comma_unbrace, last_comma_unbrace},
1323     {}{rest_comma}{middle_comma, last_comma},
1324     {}{rest_unbrace}{middle_unbrace, last_unbrace},
1325     {}{serial_and}{first_apply, serial_rest_and},
1326     {}{serial_math_and}{first_apply, serial_rest_math_and},
1327     {}{unbrace}{first_unbrace, rest_unbrace},
1328     % ^^A <one long entry>
1329     {}{serial_rest_and}
1330     {serial_middle, serial_second_and, serial_last_and}
1331     % ^^A </one long entry>
1332     ,
1333     % ^^A <one long entry>
1334     {}{serial_rest_math_and}
1335     {serial_middle_math, serial_second_math_and, serial_last_math_and}
1336     % ^^A </one long entry>
1337   }
1338 }

```

15 other

`\ClisTeXLogo`

```

1339 \NewDocumentCommand
1340 \ClisTeXLogo{}

```

```

1341 { \textsf{ Cl\raisebox{.5ex}{i}sT\raisebox{-.5ex}{E}X} }
(End definition for \ClisTeXLogo. This function is documented on page 3.)
1342 \ProcessKeysOptions{__clistex}
1343 \ExplSyntaxOff
1344 \</package>

```