The **ccool** package[*]

Erwann Rogard[†]

Released 2020/10/10

Abstract

The package ccool for LaTeX is a *key-value* interface, `\Ccool`, on top of xparse's document command parser. Global options control input processing and its expansion. By default, they are set to meet likely requirements, depending on context: the selected language, and which of text and math mode is active. These options can be overriden inline. This versality could find its use, for example, to encode notational conventions (such as `\Real → \mathbb{R}`) at the point where they are introduced in the `document` ("Let ℝ denote real numbers"). Polymorphic commands can be generated by parameterizing the keys (for instance, one parameter value for style, another for a property). User input to `\Ccool` can optionally be serialized. This can useful for typesetting documents sharing the same notation.

Résumé

L'extension ccool pour LaTeX met à disposition une interface de type *clé-valeur*, `\Ccool`, destinée à faciliter la géneration de commandes. Les paramètres globaux contrôlant le traitement de ces *clé-valeur* sont fixés par défaut pour répondre aux besoins courants, suivant le contexte (langage, mode textuel ou mathématique). Un exemple d'application, est la command-isation des conventions de notation (`\Reel → \mathbb{R}`), au point dans le `document` où elles sont introduites ("Soit ℝ les nombres réels."). Des commandes polymorphes peuvent être générées, en associant aux clés un paramètre (par exemple, une valeur pour le style typographique, une autre pour la description du concept associé). En option, les instructions passées à cette interface peuvent être sauvegardées, ce qui peut être utile pour la rédaction de documents faisant appel à des conventions typographiques communes.

Contents

---

[*]This file describes version v3.1, last revised 2020/10/10.

[†]firstname dot lastname AusTria gmail dot com

Part I
Usage

Convention

a) Loosely, those of [2], for example as to the meaning of *token list*, which may be abbreviated *tl*.

b) Those of [5], for example [arg] is an *'o'-type argument*.

c) $\langle X \rangle \leftarrow$ Y: set $\langle X \rangle$ to Y

d) \X $\rightarrow$ Y: \X expands to Y

e) If unspecified, the environment in which a macro is to be used is `document`.

`\usepackage`

\usepackage{ccool}

Requirement

1. `ccool.sty` is in the path of the LaTeX engine. See Part III, section 5.
2. Put in the *preamble*

`\Ccool`

\Ccool[$\langle tl_1 \rangle$]<$\langle tl_2 \rangle$>c{$\langle code_1 \rangle$}{$\langle keyval\ list_1 \rangle$}+*s{$\langle separators \rangle$}c{$\langle code_2 \rangle$}[$\langle tl_6 \rangle$]

where $\langle separators \rangle$ is either of: {$\langle tl_3 \rangle$}, {$\langle tl_3 \rangle$}{$\langle tl_4 \rangle$}, and {$\langle tl_3 \rangle$}{$\langle tl_4 \rangle$}{$\langle tl_5 \rangle$}.

Semantics See subsection 2.1-2.8.

2.1 Core feature

\Ccool{$\langle keyval\ list_1 \rangle$} executes for each $\langle key_i \rangle$=$\langle val_i \rangle$,

1) $\langle val_i \rangle$ $\leftarrow$ \function{$\langle val_i \rangle$}

2) define \$\langle key_i \rangle$ such that \$\langle key_i \rangle$ $\rightarrow$ $\langle val_i \rangle$,

where \function is encoded in *global option* Inner. For instance, the side effect of \Ccool{ Real = \mathbb{R} } is \Real $\rightarrow$ \mathbb{R}. To be sparingly used, *global option* Expans controls the type of expansion of $\langle key_i \rangle$ and $\langle val_i \rangle$.

See \CcoolLambda to allow command \$\langle key_i \rangle$ to take arguments.

2.2 Process the *val*$_i$s

`\Ccool c{`⟨*code*$_1$⟩`}{`⟨*keyval list*$_1$⟩`}` is identical to the Core feature, except it overrides Inner.

In our example, if multiple number systems are defined with `\Ccool` (natural, reals, . . .), it is more efficient to omit `\mathbb{.}` inside ⟨*val*$_i$⟩, and instead use `c{\mathbb{#1}}`, where `#1` means "parameter to be replaced".

2.3 Append to a hook

`\Ccool{`⟨*keyval list*$_1$⟩`}+` is identical to the Core feature, except it repeats after `\CcoolHook`. This is useful to make the side effect persist after a *local group* (such as `theorem`).

2.4 Expand the *val*$_i$s

`\Ccool{`⟨*keyval list*$_1$⟩`}*` supplements the Core feature with the expansion of the ⟨*val*$_i$⟩'s using typesetting rules encoded in *global option* `Separ` and `Outer`. The first are *separators* applied to the ⟨*val*$_i$⟩'s to form a *token list*, and the second a function applied to the latter.

They can be overriden inline by appending further `s{`⟨*separators*⟩`}` and `c{`⟨*code*$_2$⟩`}`, respectively, to the list of arguments.

2.5 Head

`\Ccool[`⟨*tl*$_1$⟩`]{`⟨*keyval list*$_1$⟩`}` expands ⟨*tl*$_1$⟩ and executes the Core feature.

There may be situations where it is convenient to pass ⟨*tl*$_1$⟩ as empty.

2.6 Tail

`\Ccool{`⟨*keyval list*$_1$⟩`}[`⟨*tl*$_6$⟩`]{`⟨*keyval list*$_2$⟩`}` is identical to `\Ccool{`⟨*keyval list*$_1$⟩`}` followed by `\Ccool[`⟨*tl*$_6$⟩`]{`⟨*keyval list*$_2$⟩`}`.

The combination of Core feature, Head, and Tail allows to integrate typesetting and the creation of commands.

2.7 Parameterize the *key*$_i$s

`\Ccool<`⟨*tl*$_2$⟩`>{`⟨*keyval list*$_1$⟩`}` is identical to the Core feature, except ⟨*key*$_i$⟩ is replaced by ⟨*key*$_i$`<`*tl*$_2$`>`⟩. The default value of ⟨*tl*$_2$⟩ is encoded in `Param`. In our example, ⟨*tl*$_2$⟩ could be `Style`.

2.8 Write

global option `Write` is identical to the Core feature, except that if `Write` is set to `\BooleanTrue`, the code is written to a file whose path is encoded in *global option* `File`.

\CcoolClear   `\CcoolClear<`⟨*tl*$_2$⟩`>{...`⟨*key*$_i$⟩`,...}`

Semantics Clears all `\`⟨*key*$_i$`<tl`$_2$`>`⟩'s

\CcoolHook | \CcoolHook

Semantics No side effect or expansion

\CcoolLambda | \CcoolLambda[⟨*arg spec*⟩]{⟨*code*⟩},

where *arg spec* is by default an *'o'-type argument.*

Example \Ccool{ EvalAt = \CcoolLambda{(#1)} }

Semantics Returns a command of type \DeclareDocumentCommand[5],

---

\CcoolOption | \CcoolOption[...⟨*key$_i$*⟩|⟨*key$_i$*⟩=⟨*val$_i$*⟩,...]

where ⟨*key$_i$*⟩ is either of And, Expans, File, Inner, Param, Outer, Separ, and Write.

Semantics Modify the default behavior of \Ccool

And

Also see Part IV And

Semantics Sets the translation of *and* in language ⟨*key*⟩ to ⟨*val*⟩

Syntax ⟨*keyval list*⟩

Expans

Also see Core feature and Part IV Expans

Syntax eo|ee|ex|xo|xe|xx

File

Also see Part I Write and Part IV File

Syntax ⟨*path*⟩

Inner

Also see Process the *val$_i$*s and Part IV Inner

Syntax ⟨*code*⟩, with ####1 as the *placeholder*

Param

Also see Parameterize the *key$_i$*s, and Part IV Param

Syntax ⟨*token list*⟩

Outer

Also see Expand the *val$_i$*s, and Part IV Outer

Default \ensuremath{####1}

7

Syntax $\langle code \rangle$, with `####1` as the *placeholder*

Separ

Also see Expand the *val$_i$*s; Listing 8; and Part IV Separ

Other Default behavior depends on whether `babel` and `amsmath` are loaded

Syntax That of *separators* in [2, Section 8 of l3seq]

Write

Also see Part I Write and Part IV Write

Syntax `\BooleanFalse|\BooleanTrue`

`\CcoolRead` `\CcoolRead[`$\langle path \rangle$`]`

Also see Part IV `\CcoolRead`

Semantics

1. Reads the definitions in $\langle path \rangle$.
2. Writes to `ccool.log`: 'read from $\langle path \rangle$'

`\CcoolVers` `\CcoolVers`

Semantics $\rightarrow$ the package's version

9   Do's and dont's

1)

   Don't: `Inner=\{####1\}`

Symptom: `\CcoolRead` fails

      Do: ``Inner={\char`{####1\char`}}``

2)

   Don't: `$`$\langle key_i \rangle$`<x$.`

      Do: `$\`$\langle key_i \rangle$`{<}x$`

3)

   Don't: `[a, b)`

      Do: `{[}a, b{)}`

4)

Don't: \cal F.

    Do: \cal{F} or \mathcal{F}

5)

Don't: \[x_0,x\]

    Do: \left[x_0,x\right]

6)

Don't: Use *'d'-type* or *'e'-type* arguments[5] for \CcoolLambda

    Do: Use only *'m'-type* and *'o'-type* arguments

7)

Don't: \usepackage[spanish]{babel}

    Do: \usepackage[spanish.noquoting]{babel}[11]

8) Also see

Part II
Listing

NB:

1. Some statements affect only the output of listings that come after that in which they appear. The demarcation is indicated by `%^^A--->` and `%^^A<---`, where applicable

---

Listing 1. `\CcoolVers`

```
\CcoolVers
```
- -
2020/10/10 v3.1 cool — A key-value document command parser

Listing 2. This doc's dependencies

```
%^^A  \printfilelist
  \ExplSyntaxOn
  \__ccooldoc_usepackage:
  \ExplSyntaxOff
```
- -
```
\usepackage{amsmath,amssymb}
\usepackage[french,german,spanish.noquoting,english]{babel}
\usepackage{bookmark,ccool,microtype,tabto,tcolorbox}
\usepackage[T1]{fontenc}
```

Listing 3. "Let $\mathbb{N}$ and $\mathbb{R}$ denote..." (start of the tutorial)

```
  Let~$\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers.
```
- -
Let $\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers.

Listing 4.   Equivalent to 3, with `\NewDocumentCommand`

```
  \DeclareDocumentCommand\Nat{}{\mathbb{N}}
  \DeclareDocumentCommand\Real{}{\mathbb{R}}
  Let~$\Nat$ and $\Real$ denote the natural and real numbers.
```
- -
Let $\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers.

Listing 5.   Equivalent to 4, with `\Ccool`

```
  % ^^A--->
  \Ccool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
  Let~$\Nat$ and $\Real$~denote the natural and real numbers.
  % ^^A<---
  \CcoolClear
```

Let ℕ and ℝ denote the natural and real numbers.

Listing 6.   Equivalent to 5, with expansion

```
% ^^A--->
\Ccool[Let~]
c{\mathbb{#1}}{ Nat = {N}, Real = {R} }*
[~denote the natural and real numbers.]{}
% ^^A<---
\CcoolClear
```

Let ℕ and ℝ denote the natural and real numbers.

Listing 7.   Equivalent to 5, parameterized (end of the tutorial)

```
% ^^A--->
\Ccool<Style>c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
[Let $\Nat<Style>$ and $\Real<Style>$ denote the natural and real
  numbers.]{}
% ^^A<---
\CcoolClear<Style>
```

Let ℕ and ℝ denote the natural and real numbers.

Listing 8. Language and mode

```
% ^^A--->
\textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\begin{otherlanguage}{spanish}
  \CcoolOption[ Separ ]\\
  \textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\end{otherlanguage}\\
\textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\\[1em]
\CcoolOption[ Outer = ####1 ]
\textbf{\languagename}{:}~\Ccool{ X = this, Y = that }*
\begin{otherlanguage}{spanish}
  \CcoolOption[ Separ ]\\
  \textbf{\languagename}{:}~\Ccool{ X = esto, Y = aquello }*
\end{otherlanguage}\\
\textbf{\languagename}{:}~\Ccool{ X = this, Y = that }*
\CcoolOption[ Separ ]\\
% ^^A<---
\CcoolOption
```
----

english: x and y
spanish: x y y
english: x and y

> **english**: this and that
> **spanish**: esto y aquello
> **english**: this and that

Listing 9. Separators (*Note*[a])

---

[a][bug]: Removing the closing \CcoolOption subsequently causes inconsistent separators between text and math mode (case replicated in uncommented form in dtx)

```
% ^^A--->
\CcoolOption[ Separ={{\ \char`@\ }{\ \%\ }{\ \char`@\ }} ]
\Ccool{ X = x, Y = y }*[\\]
{ X = x, Y = y }*s{{~\&~}}[\\]
{ X = x, Y = y }*s{{,~}{~\&~}}[\\[1em]]
{ X = x, Y = y, Z = z }*[\\]
{ X = x, Y = y, Z = z }*s{{~\&~}}[\\]
{ X = x, Y = y, Z = z }*s{{,~}{\&~}}[\\]
{ X = x, Y = y, Z = z }*s{{~\&~}{,~}{,~\&~}}\\
% ^^A<---
\CcoolOption
\CcoolClear
```

- -

$x @ y$
$x \& y$
$x \& y$

$x \% y @ z$
$x \& y \& z$
$x, y, \& z$
$x, y, \& z$

Listing 10. Hello, world! (testing)

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\CcoolOption[Separ = {{}{.}{.}}, Outer = {####1}]
\Ccool
<Test>{ KeyA = {.}, KeyB = {!}, KeyC = {\%} }[]
<Test>{ KeyD = {d}, KeyE = {\%} }[]
<Test>c{\{#1\}}{ KeyF = {H}, KeyG = {e}, KeyH = {l} }*[]
<Test>{ KeyI = {\%}, KeyJ = {\%}, KeyK = {\%} }[.\{l\}.\{o\}]
<Test>{ KeyL = {l}, KeyM = {\char`[}, KeyN = {\char`]} }[]
<Test>{ KeyO = {o}, KeyP = {\%}, KeyQ = {\%} }[{,\ }]
<Test>{ KeyR = {w}, KeyS = {o}, KeyT = {r} }*
s{{}{}{}}c{{\char`[}#1}[]
<Test>{ KeyU = {\%}, KeyV = {\%}, KeyW = {\%} }[]
<Test>{ KeyX = {\%}, KeyY = {\%}, KeyZ = {\KeyB<Test>} }\nobreak
\KeyL<Test>\KeyD<Test>\KeyZ<Test>\KeyN<Test>\\
% ^^A<---
```

```
    \CcoolOption
    \CcoolClear
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 11. Listing 10 read from file

```
% ^^A--->
\CcoolRead
\KeyF<Test>\KeyA<Test>\nobreak
\KeyG<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
{\{}\nobreak\KeyO<Test>{\}},{\ }\nobreak
\KeyM<Test>\KeyR<Test>\nobreak
\KeyO<Test>\nobreak
\KeyT<Test>\nobreak
\KeyL<Test>\nobreak
\KeyD<Test>\nobreak
\KeyZ<Test>\nobreak
\KeyN<Test>\nobreak
% ^^A<---
\CcoolClear
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 12. Probability space

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\Ccool[Let~]
{ Space = \Omega, Field = \mathcal{F}, Meas = \mathcal{P} }
*s{{,}}c{$\{#1\}$}
[~denote the probability space, where~]{ PowerSet = { 2^{\Space} } }
[$\Field\subset \PowerSet$.]
{}
% ^^A<---
\CcoolClear
\CcoolOption
```

Let $\{\Omega, \mathcal{F}, \mathcal{P}\}$ denote the probability space, where $\mathcal{F} \subset 2^{\Omega}$.

Listing 13. Listing 12 read from file

```
% ^^A--->
\CcoolRead \tab $\Omega$ $\Field$ $\Meas$
% ^^A<---
\CcoolClear
```

Listing 14. Mittelwertsatz für n Variable[4, 17.3]

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\selectlanguage{german}
\newtheorem{theorem}{Theorem}
\AfterEndEnvironment{theorem}{\CcoolHook}
\Ccool c{\mathbb{#1}}
{ N = { N } , R = { R } }+[]
{ Grad = { \operatorname{grad} } }+
[\begin{theorem}
  [Mittelwertsatz f\"ur $n$ Variable]Es~sei~]
  { OffMenge = {D}, Ci = {C^{1}}, Strecke = { \left[x_0,x\right] } }+
  [$n\in\N$,~$\OffMenge\subseteq\N^n$ eine offene Menge und
  $f\in\Ci(\OffMenge,\R)$.
  Dann gibt es auf jeder Strecke $\Strecke\subset\OffMenge$ einen Punkt
  $\xi\in\Strecke$,~]
  { Steig = { \frac{ f(x)-f(x_0) }{ x-x_0 } }, Punkt = { \xi } }+
  [so dass gilt
  \begin{equation*}
    \Steig = \Grad f(\Punkt)^{\top}
  \end{equation*}
\end{theorem}]
{}
(Check: $\N$, $\Punkt$)
% ^^A<---
\CcoolClear
\CcoolOption
```

Theorem 1 (Mittelwertsatz für n Variable) *Es sei $n \in \mathbb{N}$, $D \subseteq \mathbb{N}^n$ eine offene Menge und $f \in C^1(D, \mathbb{R})$. Dann gibt es auf jeder Strecke $[x_0, x] \subset D$ einen Punkt $\xi \in [x_0, x]$, so dass gilt*

$$\frac{f(x) - f(x_0)}{x - x_0} = \operatorname{grad} f(\xi)^{\top}$$

(Check: $\mathbb{N}$, ξ)

Listing 15. Listing 14 read from file

```
% ^^A--->
\CcoolRead \tab $\N$ $\R$ $\OffMenge$ $\Ci$ $\Strecke$
% ^^A<---
\CcoolClear
```

$\mathbb{N}$ $\mathbb{R}$ D C^1 $[x_0, x]$

Listing 16. Families of polynomial functions

```
    \CcoolOption[ Write = \BooleanTrue ]
    % ^^A--->
    \Ccool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
    [Let~]
    { PolyR = \CcoolLambda[o]{\Real\IfValueT{#1}{_#1}[X] } }
    [$\PolyR[n]$ and $\PolyR$, denote the families of polynomial functions
      on $\Real$, of order $n$ et and their union over $n \in \Nat$,
      respectively. ]
    {}
    % ^^A<---
    \CcoolClear
    \CcoolOption
```

Let $\mathbb{R}_n[X]$ and $\mathbb{R}[X]$, denote the families of polynomial functions on $\mathbb{R}$, of order n et and their union over $n \in \mathbb{N}$, respectively.

Listing 17. Listing 16 read from file

```
    % ^^A--->
    \CcoolRead \tab $\PolyR[n]$ et $\PolyR$
    % ^^A<---
    \CcoolClear
```

$$\mathbb{R}_n[X] \text{ et } \mathbb{R}[X]$$

Listing 18. Same as Listing 16, but arbitrary number system

```
    \CcoolOption[ Write = \BooleanTrue ]
    % ^^A--->
    \selectlanguage{french}
    \Ccool c{\mathbb{#1}}{ Corps = {K}, Nat = {N}, Reel = {R} }
    [Soient~]
    {
      Poly = \CcoolLambda[om]{#2\IfValueT{#1}{_#1}[X] },
      PolyR = \CcoolLambda[o]{\Poly[#1]{\Reel}}
    }
    [$\Poly[n]{\Corps}$ et $\Poly{\Corps}$, les familles de polyn\^omes sur
      $\Corps$, de degr\'e $n$ et leur union pour $n \in \Nat$,
      respectivement. En particulier,
    ils sont d\'enot\'es $\PolyR[n]$ et $\PolyR$, pour $\Corps=\Reel$.]
    {}
    % ^^A<---
    \CcoolClear
    \CcoolOption
```

Soient $\mathbb{K}_n[X]$ et $\mathbb{K}[X]$, les familles de polynômes sur $\mathbb{K}$, de degré n et leur union pour $n \in \mathbb{N}$, respectivement. En particulier, ils sont dénotés $\mathbb{R}_n[X]$ et $\mathbb{R}[X]$, pour $\mathbb{K} = \mathbb{R}$.

Listing 19. Listing 18 read from file

```
% ^^A--->
\CcoolRead \tab $\PolyR[n]$ et $\PolyR$
% ^^A<---
\CcoolClear
```

$$\mathbb{R}_n[X] \text{ et } \mathbb{R}[X]$$

Listing 20. Fonction et fonctionelle

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\selectlanguage{french}
\Ccool{ EvalAt = \CcoolLambda{(#1)}, ApplyOp = \CcoolLambda[mm]{#1[#2]} }
[Supposons une fonction $f\EvalAt{t}$, et \'etudions le probl\`eme o\`u
  la fonctionnelle $\ApplyOp{S}{f}$ est donn\'ee par\dots]{}
% ^^A<---
\CcoolClear
\CcoolOption
```

Supposons une fonction $f(t)$, et étudions le problème où la fonctionnelle $S[f]$ est donnée par. . .

Listing 21. Listing 20 read from file

```
% ^^A--->
\CcoolRead \tab $f\EvalAt{t}$, $\ApplyOp{S}{f}$
% ^^A<---
\CcoolClear
```

$$f(t), S[f]$$

Listing 22. CUSUM statistic[6]

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\newtheorem{definition}{Definition}
\AfterEndEnvironment{definition}{\CcoolHook}
\Ccool{
  SuchThat = { ;~ },
  Time = { t },
  Process = { \xi },
  StopT = { T },
  EvalAt = \CcoolLambda{(#1)}
}
[The CUSUM statistic process and the corresponding one-sided CUSUM
  stopping time are defined as follows:
\begin{definition}\label{the CUSUM statistic}. Let~]
  {
    Scale = { \lambda },
```

```
      Real = {\mathcal{R}}
    }+*s{{~\in~}}
    [~and~]
    { CUSUMthresh = { \nu } }+*c{$#1\in\Real^{+}$.}
    [~Define the following processes:]
    {
      LogWald = { u },
      CUSUMst = { \StopT_{c} },
      CUSUM = { y },
      LogWaldInf = { m }
    }+
    [\begin{enumerate}
    \item{
        $\LogWald_{\Time}\EvalAt{ \Scale } = \Scale\Process_{\Time} -
    \frac{1}{2}\Scale^2\Time$;
        $\LogWaldInf_{\Time}\EvalAt{ \Scale } = \inf_{ 0\le s \le \Time
    }\CUSUM_{s} \EvalAt{ \Scale }$.
      }
    \item{
        $\CUSUM_{\Time}\EvalAt{ \Scale } = \LogWaldInf_{\Time}\EvalAt{
    \Scale } - \LogWald_{\Time}\EvalAt{ \Scale }\ge0$,
        which is the CUSUM statistic process.
      }
    \item{
        $\CUSUMst \EvalAt{ \Scale, \LogWaldInf } = \inf\left[ \Time \ge 0
    \SuchThat \CUSUM_{\Time}\EvalAt{\Scale} \ge \LogWaldInf \right]$,
        which is the CUSUM stopping time.
      }
    \end{enumerate}
  \end{definition}\par]{}

  (Check: $\Scale$, $\CUSUM$)
  % ^^A<---
  \CcoolClear
  \CcoolOption
%
```

- -

The CUSUM statistic process and the corresponding one-sided CUSUM stopping time are defined as follows:

Definition 1 . Let $\lambda \in \mathcal{R}$ and $\nu \in \mathcal{R}^+$. *Define the following processes:*

1. $u_t(\lambda) = \lambda\xi_t - \frac{1}{2}\lambda^2 t$; $m_t(\lambda) = \inf_{0 \le s \le t} y_s(\lambda)$.

2. $y_t(\lambda) = m_t(\lambda) - u_t(\lambda) \ge 0$, *which is the CUSUM statistic process.*

3. $T_c(\lambda, m) = \inf [t \ge 0; y_t(\lambda) \ge m]$, *which is the CUSUM stopping time.*

(Check: λ, y)

Listing 23. Listing 22 read from file

```
% ^^A--->
\CcoolRead \tab $\Time $ $\Process$ $\Scale$ $\Real$ $\CUSUMthresh$
   $\LogWald$  $\CUSUMst$ $\CUSUM$ $\LogWaldInf$
% ^^A<---
\CcoolClear
```

$t \; \xi \; \lambda \; \mathcal{R} \; \nu \; u \; T_c \; y \; m$

Part III
Other

1   Acknowledgment

This work has benefited from Q&A's from the LaTeXcommunity[7][10]. Specific attributions are made throughout this document.

2   Genealogy

"Give commands the ability to contain the mathematical meaning while retaining the typesetting versatility" (cool[1]). The addition of 'c', in ccool, is for *custom*. With hinsdight it is restrictive to describe ccool as a tool for encoding mathematical convention.

3   Install

   1) Compile `ccool.dtx` (under Unix, `$pdflatex ccool.dtx`)

   2) Put the generated `ccool.sty` in the search path of the LaTeXengine

4   Issue

Listed under:

   a) `NOTE` or `\NB`, tagged either `bug` or `fixed`, inside ccool.`dtx`

5   Support

This package is available from `https://www.ctan.org/pkg/ccool` and `https://github.com/rogard/ccool`.

6   Testing

6.1   Technicality

Not possible to compile-check the expansion of a certain class of macros against predefined values[8]. Instead, one can

   a) Follow the steps in section 3 on one's own machine to generate ccool.`pdf`

   b) Visually check Part II, against that of the repository, for the same version.

Also see:

   c) Also see: section 7 *a)*

6.2 Platform

i)   `Linux laptop 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24`
↪   `06:16:15 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux`

6.3 Engine

a)   `pdfTeX 3.14159265-2.6-1.40.20 (TeX Live 2019)`

b)   `pdfTeX 3.14159265-2.6-1.40.21 (TeX Live 2020)`

c)   `LuaHBTeX, Version 1.12.0 (TeX Live 2020)`

d)   `XeTeX 3.14159265-2.6-0.999992 (TeX Live 2020)`

6.4 Results

1) ccool v1.8 compiles satisfactorily on platform *i)* and engine *a)*

2) ccool v1.8 compiles satisfactorily on platform *i)* and engine *b)*

3) ccool v1.9 compiles satisfactorily on platform *i)* and engines *b)* and *c)*

4) ccool v2.0 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

5) ccool v2.1 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

6) ccool v2.3 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

7) ccool v2.7 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

8) ccool v2.8 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

6.5 Other

Check [6] for testing ccool with llncs

7 To do

a) Regression testing using [3, Section 3.2—Specifying expectations].

Also see:

b) `NOTE` or `\NB` tagged `abandon|done|todo` inside ccool.`dtx`

References

[1] Nick Setzer *The cool package*, 2005, https://www.ctan.org/pkg/cool

[2] The LaTeX3 Project Team *The LaTeX3 interfaces*, 2019, http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf

[3] The LaTeX3 Project Team *The l3build package*, 2020, http://mirror.utexas.edu/ctan/macros/latex/contrib/l3build/l3build.pdf

[4] Thomas F. Sturm *The tcolorbox package*, 2019, http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf

[5] The LaTeX3 Project Team *The xparse package*, 2020, http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf

[6] Erwann Rogard and Olympia Hadjiliadis *Typesetting a math thesis with ccool*, 2020, https://github.com/rogard/ccool/blob/master/thesis.pdf

[7] https://tex.stackexchange.com/users/112708/erwann?tab=questions

[8] @joseph-wright's answer to "Checking a function's expansion against a string", https://tex.stackexchange.com/a/534100

[9] @frougon's answer to "Journaling calls to a function []", https://tex.stackexchange.com/a/536620

[10] \Ccool, extension à LaTeX à vocation mathématique, http://forum.mathematex.net/latex-f6/ccool-extension-latex-a-vocation-mathematique-t17314.html

[11] @Javier Bezos's answer to https://tex.stackexchange.com/a/547018/112708

Change History

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

24

Part IV
Implementation

1 Opening

```
1 ⟨*package⟩
2 ⟨@@=ccool⟩
3 \ExplSyntaxOn
```

2 aux

`\__ccool_aux_inner_set:n` #1 : ⟨code⟩

```
4 \cs_new_protected:Nn \__ccool_aux_inner_set:n
5 {
6   \cs_gset:Npn \__ccool_aux_inner:n ##1 {#1}
7   \cs_generate_variant:Nn \__ccool_aux_inner:n { e }
8 }
```

(*End definition for* `\__ccool_aux_inner_set:n`.)

`\__ccool_aux_key:w` #1 : ⟨ key ⟩
#2 : ⟨ value ⟩

```
9 \cs_new_protected:Npn \__ccool_aux_key:w #1 = #2 \q_stop
10 {
11   \seq_gput_right:Nx \g__ccool_aux_key_seq { \tl_trim_spaces:n{#1} }
12 }
```

(*End definition for* `\__ccool_aux_key:w`.)

`\__ccool_aux_key:n` #1 : ⟨ key = value ⟩

```
13 \cs_new_protected:Nn \__ccool_aux_key:n
14 {
15   \__ccool_aux_key:w #1 \q_stop
16 }
```

(*End definition for* `\__ccool_aux_key:n`.)

`\__ccool_aux_key:N` #1 : ⟨ seq ⟩

```
17 \cs_new_protected:Nn \__ccool_aux_key:N
18 {
19   \seq_gclear_new:N \g__ccool_aux_key_seq
20   \seq_map_function:NN #1 \__ccool_aux_key:n
21 }
```

(*End definition for* `\__ccool_aux_key:N`.)

`\__ccool_aux_outer_set:n` #1 : ⟨ inline code ⟩

```
22 \cs_new_protected:Nn \__ccool_aux_outer_set:n
23 {
24   \cs_gset:Npn \__ccool_aux_outer:n ##1 {#1}
25 }
```

(*End definition for* __ccool_aux_outer_set:n.)

__ccool_aux_prop:nn

```
26 \prop_new:N \g__ccool_aux_prop
27 \cs_new_protected:Nn \__ccool_aux_prop:nn
28 {
29   \prop_gput:Nnn \g__ccool_aux_prop{#1}{#2}
30 }
31 \cs_generate_variant:Nn \__ccool_aux_prop:nn { eo, ee, ex, xo, xe, xx }
```

(*End definition for* __ccool_aux_prop:nn.)

__ccool_aux_prop:w   #1 : ⟨ key ⟩
                      #2 : ⟨ value ⟩

```
32 \tl_new:N \g__ccool_option_expans_tl
33 \cs_new_protected:Npn \__ccool_aux_prop:w #1 = #2 \q_stop
34 {
35   \exp_args:Nx
36   \use:c{__ccool_aux_prop:\g__ccool_option_expans_tl}
37   { \tl_trim_spaces:n{#1} }
38   { \__ccool_aux_inner:n{ \tl_trim_spaces:n{#2} } }
39 }
```

(*End definition for* __ccool_aux_prop:w.)

__ccool_aux_prop:n   #1 : ⟨ key = value ⟩

```
40 \cs_new_protected:Nn \__ccool_aux_prop:n
41 {
42   \__ccool_aux_prop:w #1 \q_stop
43 }
```

(*End definition for* __ccool_aux_prop:n.)

__ccool_aux_prop:N   #1 : ⟨keyval list⟩

```
44 \cs_new_protected:Nn \__ccool_aux_prop:N
45 {
46   \prop_gclear_new:N \g__ccool_aux_prop
47   \seq_if_empty:NTF #1
48   { \c_empty_tl }
49   {
50     \seq_map_function:NN #1 \__ccool_aux_prop:n
51   }
52 }
```

(*End definition for* __ccool_aux_prop:N.)

__ccool_aux_val:Nn   #1 : ⟨ seq ⟩
                      #2 : ⟨ tl var name ⟩

```
53 \cs_new_protected:Nn \__ccool_aux_val:Nn
54 {
55   \seq_gclear_new:N \g__ccool_aux_val_seq
56   \__ccool_seq_from_prop:NNn \g__ccool_aux_val_seq #1 { \__ccool_prop_name:n{#2} }
57 }
```

(*End definition for* __ccool_aux_val:Nn.)

3 lang

58 `\prop_new:N \g__ccool_lang_and_prop`

`\__ccool_lang_and_update:n`

59 `\cs_new_protected:Nn \__ccool_lang_and_update:n`
60 `{`
61 `  \erw_prop_keyval_parse:NNNn`
62 `  \g__ccool_lang_and_prop`
63 `  \erw_keyval_error:Nn`
64 `  \prop_gput:Nnn`
65 `  { #1 }`
66 `}`
67 `\cs_generate_variant:Nn \__ccool_lang_and_update:n { e }`

(*End definition for* `\__ccool_lang_and_update:n`.)

`\__ccool_lang_and:n`
`\__ccool_lang_and:`

68 `\cs_new:Nn \__ccool_lang_and:n`
69 `{`
70 `  \prop_if_in:NnTF`
71 `  \g__ccool_lang_and_prop`
72 `  {#1}`
73 `  {\prop_item:Nn\g__ccool_lang_and_prop{#1}}`
74 `  {`
75 `    \msg_warning:nnn{__ccool}{lang_and}{#1}`
76 `    \__ccool_lang_and:n{english}`
77 `  }`
78 `}`
79 `\ifcsdef{languagename}`
80 `{`
81 `  \cs_new:Nn \__ccool_lang_and:{\exp_args:No\__ccool_lang_and:n{\languagename}}`
82 `}`
83 `{`
84 `  \cs_new:Nn \__ccool_lang_and:{english}`
85 `}`

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

`\c__ccool_lang_and_tl` (*Note*[1])

86 `\tl_const:Nn \c__ccool_lang_and_tl`
87 `{`
88 `%^^A https://www.overleaf.com/learn/latex/International_language_support`
89 `  afrikaans=en,`
90 `  basque=eta,`
91 `  catalan=i,`
92 `  croatian=i,`
93 `  czech=a,`
94 `  danish=og,`
95 `  dutch=en,`
96 `  english=and,`
97 `  esperanto=kaj,`
98 `  estonian=ja,`

---

[1] [todo]: Non latin-alphabet languages

```
99    finnish=ja,
100   french=et,
101   galician=e,
102   german=und,
103   hungarian=\'es,
104   icelandic=og,
105   indonesian=dan,
106   irish=agus,
107   italian=e,
108   kurmanji=\^u,
109   latin=et,
110   latvian=un,
111   lithuanian=ir,
112   ngerman=und,
113   polish=i,
114   portuguese=e,
115   romanian=\c{s}i,
116   slovak=a,
117   spanish=y,
118   swedish=och,
119   swissgerman=und,
120   turkish=ve,
121   turkmen=we,
122   welsh=a
123 }
```

(End definition for `\c__ccool_lang_and_tl`*.)*

4 log

`\__ccool_log_close:`

```
124 \iow_new:N \g__ccool_log_iow
125 \AtEndDocument{\iow_close:N \g__ccool_log_iow}
126 \bool_set_false:N \g__ccool_log_open_bool
127 \cs_new_protected:Nn \__ccool_log_close:
128 {
129   \iow_close:N \g__ccool_log_iow
130   \bool_gset_false:N \g__ccool_log_open_bool
131 }
```

(End definition for `\__ccool_log_close:`*.)*

`\__ccool_log_open:`

```
132 \tl_new:N \g__ccool_log_file_tl
133 \cs_new_protected:Nn \__ccool_log_open:
134 {
135   \tl_gset:Nx \g__ccool_log_to_tl{\g__ccool_log_file_tl}
136   \iow_open:Nn \g__ccool_log_iow {\g__ccool_log_to_tl}
137   \bool_gset_true:N \g__ccool_log_open_bool
138 }
```

(End definition for `\__ccool_log_open:`*.)*

`\__ccool_log_read:n` #1 : ⟨*path*⟩

```
139 \cs_new_protected:Nn \__ccool_log_read:n
140 {
141   \file_input:n{#1}
142   \tl_log:n{read~from~#1}
143 }
144 \cs_generate_variant:Nn \__ccool_log_read:n { e }
```

(*End definition for* `\__ccool_log_read:n.`)

`\__ccool_log_read:`

```
145 \cs_new_protected:Nn \__ccool_log_read:
146 {
147   \__ccool_log_read:e{\g__ccool_log_to_tl}
148 }
```

(*End definition for* `\__ccool_log_read:.`)

`\__ccool_log_write:n`

```
149 \tl_new:N \g__ccool_log_to_tl
150 \cs_new_protected:Nn \__ccool_log_write:n
151 {
152   \bool_if:nTF{ \g__ccool_log_open_bool }
153   {
154     \iow_now:Nn \g__ccool_log_iow {#1}
155     \tl_log:n{ write~to~#1 }
156   }
157   { \msg_error:nnn{ __ccool }{ iow }{ \g__ccool_log_iow }  }
158 }
159 \cs_generate_variant:Nn \__ccool_log_write:n { e }
```

(*End definition for* `\__ccool_log_write:n.`)

5   make_key

`\__ccool_make_key:Nn` #1 : ⟨ *token* ⟩
#2 : ⟨ *key* ⟩

```
160 \cs_new_protected:Nn \__ccool_make_key:Nn
161 {
162   \exp_args:NNx
163   \DeclareDocumentCommand{#1}
164   { D<>{\g__ccool_option_param_tl} }
165   {
166     \__ccool_prop_item:nn{##1}{#2}
167   }
168 }
169 \cs_generate_variant:Nn \__ccool_make_key:Nn {c}
```

(*End definition for* `\__ccool_make_key:Nn.`)

`\__ccool_make_key:n` **#1 :** ⟨ *key* ⟩

```
170 \cs_new_protected:Nn \__ccool_make_key:n
171 {
172   \__ccool_make_key:cn{#1}{#1}
173 }
174 \cs_generate_variant:Nn \__ccool_make_key:n { e }
```

(*End definition for* `\__ccool_make_key:n`.)

`\__ccool_make_key:N` **#1 :** ⟨ *seq* ⟩

```
175 \cs_new_protected:Nn \__ccool_make_key:N
176 {
177   \seq_map_function:NN #1 \__ccool_make_key:e
178 }
```

(*End definition for* `\__ccool_make_key:N`.)

6 make_ccool

`\__ccool_make_ccool_exp:nnn`

```
179 \cs_new_protected:Nn \__ccool_make_ccool_exp:nnn
180 {
181   \__ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
182   \__ccool_aux_outer_set:n{#3}
183   \__ccool_aux_outer:n
184   {
185     \exp_args:NNf
186     \erw_seq_use:Nn
187     \g__ccool_aux_val_seq
188     {#2}
189   }
190 }
```

(*End definition for* `\__ccool_make_ccool_exp:nnn`.)

`\__ccool_make_ccool_key:nnn`

```
191 \cs_new_protected:Nn \__ccool_make_ccool_key:nnn
192 {
193   \__ccool_prop_if_exist:nTF{#1}
194   { \c_empty_tl }
195   { \__ccool_prop_new:n{#1} }
196   \exp_args:No \__ccool_aux_inner_set:n{#2}
197   \seq_set_from_clist:Nn \g__ccool_aux_keyval_seq {#3}
198   \__ccool_aux_prop:N \g__ccool_aux_keyval_seq
199   \__ccool_prop_append:Nn \g__ccool_aux_prop {#1}
200   \__ccool_aux_key:N \g__ccool_aux_keyval_seq
201   \__ccool_make_key:N \g__ccool_aux_key_seq
202 }
```

(*End definition for* `\__ccool_make_ccool_key:nnn`.)

```
203 \cs_new_protected:Nn \__ccool_make_ccool_sideeffect:nnn
204 {
205   \__ccool_make_ccool_key:nnn{#1}{#2}{#3}
206   \bool_if:nTF{ \g__ccool_log_open_bool }
207   {
208     \__ccool_log_write:n
209     {
210       \begingroup
211       \def \__ccool_log_entry { \Ccool<#1>c{#2}{#3} } \expandafter
212       \endgroup \__ccool_log_entry
213     }
214   }{\c_empty_tl}
215 }
```

(*End definition for* `\__ccool_make_ccool_sideeffect:nnn`.)

#1 : ⟨ *token list* ⟩
#2 : ⟨ seq_1 ⟩
#3 : ⟨ seq_2 ⟩
#4 : ⟨ *prop* ⟩

```
216 \cs_new_protected:Npn \__ccool_make_ccool:nnnn #1 #2 #3 #4
217 {
218   \exp_args:NNx \DeclareDocumentCommand \Ccool
219   {%^^A     2        3    4 5 6            7   8    9
220     +o D<>{#1} E{ c }{{#2}} m t+ s E{ s c }{{#3}{#4}} +o
221   }
222   {
223     \IfValueT{##1}{##1}
224     \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
225     \IfBooleanT{##6}
226     {
227       \__ccool_make_ccool_exp:nnn{##2}{##7}{##8}
228     }
229     \bool_if:nTF{##5}
230     {
231       \gappto{\CcoolHook}
232       {
233         \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
234       }
235     }
236     {\c_empty_tl}
237     \IfValueT{##9}
238     {
239       \exp_not:n{ \Ccool[##9] }
240     }
241   }
242 }
```

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

7 msg

```
243 \msg_new:nnn {__ccool}
244 { iow }
245 {#1~is~closed~can't~write}
246 \msg_new:nnn {__ccool}
247 {lang_and}
248 {~key~#1~missing~for~global~option~'And';~falling~back~on~'english'}
```

8 option

__ccool_option_inner:n **#1** : ⟨*code*⟩

```
249 \tl_new:N \g__ccool_option_inner_tl
250 \cs_new_protected:Nn \__ccool_option_inner:n
251 {
252   \tl_gset:Nn \g__ccool_option_inner_tl {#1}
253 }
```

(*End definition for* __ccool_option_inner:n.)

__ccool_option_param:n **#1** : ⟨*token list*⟩

```
254 \tl_new:N \g__ccool_option_param_tl
255 \cs_new_protected:Nn \__ccool_option_param:n
256 {
257   \tl_gset:Nn \g__ccool_option_param_tl{#1}
258 }
```

(*End definition for* __ccool_option_param:n.)

__ccool_option_outer:n **#1** : ⟨ *inline code* ⟩

```
259 \tl_new:N \g__ccool_option_outer_tl
260 \cs_new_protected:Nn \__ccool_option_outer:n
261 {
262   \tl_gset:Nn \g__ccool_option_outer_tl {#1}
263 }
```

(*End definition for* __ccool_option_outer:n.)

__ccool_option_separ:n **#1** : {⟨ tl_1 ⟩}{⟨ tl_2 ⟩}{⟨ tl_3 ⟩}

```
264 \tl_new:N \g__ccool_option_separ_tl
265 \cs_new_protected:Nn \__ccool_option_separ:n
266 {
267   \cs_gset:Npn \g__ccool_option_separ_tl {#1}
268 }
```

(*End definition for* __ccool_option_separ:n.)

\g__ccool_option_separ_tl

```
269 \ifcsdef{text}
270 {
271   \tl_const:Nn \c__ccool_option_separ_default_tl
272   {
273     { \text{{\ }\__ccool_lang_and:{\ }} }
274     { \text{,{\ }} }
275     { \text{,{\ }\__ccool_lang_and:{\ }} }
276   }
```

```
277 }
278 {
279   \tl_const:Nn \c__ccool_option_separ_default_tl
280   {
281     { {\ }\__ccool_lang_and:{\ } }
282     { ,{\ } }
283     { ,{\ }\__ccool_lang_and:{\ } }
284   }
285 }
```

(End definition for `\g__ccool_option_separ_tl`.*)*

9   prop

`\__ccool_prop_append:NN`   #1 : ⟨ *prop₁* ⟩
#2 : ⟨ *prop₂* ⟩

```
286 \cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
287 {
288   \cs_set:Nn \__ccool_prop_append:nn
289   {
290     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
291   }
292   \prop_map_function:NN #2 \__ccool_prop_append:nn
293 }
294 \cs_generate_variant:Nn \__ccool_prop_append:NN { cN }
```

(End definition for `\__ccool_prop_append:NN`.*)*

`\__ccool_prop_append:Nn`   #1 : ⟨ *prop* ⟩
#2 : ⟨ *tl var name* ⟩

```
295 \cs_new_protected:Nn \__ccool_prop_append:Nn
296 {
297   \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
298 }
```

(End definition for `\__ccool_prop_append:Nn`.*)*

`\__ccool_prop_clear_new:n`   #1 : ⟨ *tl var name* ⟩

```
299 \cs_new_protected:Nn \__ccool_prop_clear_new:n
300 {
301   \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
302 }
```

(End definition for `\__ccool_prop_clear_new:n`.*)*

`\__ccool_prop_clear_new_map:n`   #1 : ⟨ *keyval list* ⟩

```
303 \cs_new_protected:Nn \__ccool_prop_clear_new_map:n
304 {
305   \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
306   \seq_map_function:NN \g__ccool_aux_key_seq \__ccool_prop_clear_new:n
307 }
```

(End definition for `\__ccool_prop_clear_new_map:n`.*)*

`\__ccool_prop_if_exist:nTF` #1 : $\langle tl_1 \rangle$
#2 : $\langle tl_2 \rangle$
#3 : $\langle tl_3 \rangle$

```
308 \cs_new:Nn \__ccool_prop_if_exist:nTF
309 {
310   \prop_if_exist:cTF{ \__ccool_prop_name:n {#1} }{#2}{#3}
311 }
```

(*End definition for* `\__ccool_prop_if_exist:nTF`.)

`\__ccool_prop_item:nn` #1 : $\langle$ *tl var name* $\rangle$
#2 : $\langle$ *key* $\rangle$

```
312 \cs_new:Nn \__ccool_prop_item:nn
313 {
314   \prop_item:cn { \__ccool_prop_name:n {#1} } {#2}
315 }
```

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

`\__ccool_prop_name:n` #1 : $\langle$ *tl var name* $\rangle$

```
316 \cs_new:Npn \__ccool_prop_name:n #1{ __ccool_#1 }
```

(*End definition for* `\__ccool_prop_name:n`.)

`\__ccool_prop_new:n` #1 : $\langle$ *tl var name* $\rangle$

```
317 \cs_new_protected:Nn \__ccool_prop_new:n
318 {
319   \prop_new:c{ \__ccool_prop_name:n {#1} }
320 }
```

(*End definition for* `\__ccool_prop_new:n`.)

10   seq

`\__ccool_seq_from_prop:NNn` #1 : $\langle seq_1 \rangle$
#2 : $\langle seq_2 \rangle$ (keys)
#3 : $\langle prop \rangle$

```
321 \cs_new_protected:Nn \__ccool_seq_from_prop:NNn
322 {
323   \cs_set_protected:Nn \__ccool_seq_from_prop:n
324   {
325     \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
326   }
327   \seq_map_function:NN #2 \__ccool_seq_from_prop:n
328 }
```

(*End definition for* `\__ccool_seq_from_prop:NNn`.)

11   Front-end

```
329 \NewDocumentCommand{ \CcoolClear }
330 { D<>{\g__ccool_option_param_tl} }
331 {
332   \__ccool_prop_clear_new_map:n{#1}
333 }
```

(*End definition for* \CcoolClear*. This function is documented on page* 6*.*)

\CcoolHook

```
334 \NewDocumentCommand{\CcoolHook}{}{\c_empty_tl}
```

(*End definition for* \CcoolHook*. This function is documented on page* 7*.*)

\CcoolLambda   (*Note*[2])

```
335 \ProvideDocumentCommand \CcoolLambda { O{m} m }
336 {
337   \erw_lambda:nnn \DeclareDocumentCommand { #1 } { #2 }
338 }
```

(*End definition for* \CcoolLambda*. This function is documented on page* 7*.*)

\CcoolOption   (*Note*[3]) (*Note*[4])

```
339 \NewDocumentCommand{ \CcoolOption }
340 { O{ And, Expans, File, Inner, Param, Outer, Separ, Write } }
341 {
342   \keys_set:nn{ __ccool }{#1}
343 }
```

(*End definition for* \CcoolOption*. This function is documented on page* 7*.*)

```
344 \keys_define:nn { __ccool }
345 {
```

And

```
346 And .code:n = { \__ccool_lang_and_update:e{ #1 } },
347 And .default:n = { \c__ccool_lang_and_tl },
348 And .initial:n = { \c__ccool_lang_and_tl },
```

Expans

```
349 Expans .multichoices:nn = { eo, ee, ex, xo, xe, xx }
350 { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
351 Expans .default:n = { xo },
352 Expans .initial:n = { xo },
```

```
      File
   353  File .code:n = {
   354    \tl_gset:Nx \g__ccool_log_file_tl{#1}
   355  },
   356  File .default:n = { \erw_sys_jobnametimestamp: },
   357  File .initial:n = { \erw_sys_jobnametimestamp: },

Inner

   358  Inner .code:n={
   359    \__ccool_option_inner:n{#1}
   360    \exp_last_unbraced:Nf
   361    \__ccool_make_ccool:nnnn
   362    {
   363      { \g__ccool_option_param_tl }
   364      { \g__ccool_option_inner_tl }
   365      { \g__ccool_option_separ_tl }
   366      { \g__ccool_option_outer_tl }
   367    }
   368  },
   369  Inner .value_required:n = false,
   370  Inner .default:n = {####1},
   371  Inner .initial:n = {####1},

Param

   372  Param .code:n={
   373    \__ccool_option_param:n{#1}
   374    \exp_last_unbraced:Nf
   375    \__ccool_make_ccool:nnnn
   376    {
   377      { \g__ccool_option_param_tl }
   378      { \g__ccool_option_inner_tl }
   379      { \g__ccool_option_separ_tl }
   380      { \g__ccool_option_outer_tl }
   381    }
   382  },
   383  Param .value_required:n = false,
   384  Param .default:n = { Default },
   385  Param .initial:n = { Default },

Outer

   386  Outer .code:n={
   387    \__ccool_option_outer:n{#1}
   388    \exp_last_unbraced:Nf
   389    \__ccool_make_ccool:nnnn
   390    {
   391      { \g__ccool_option_param_tl }
   392      { \g__ccool_option_inner_tl }
   393      { \g__ccool_option_separ_tl }
   394      { \g__ccool_option_outer_tl }
   395    }
   396  },
```

---

[2][todo]: allow only m- or o-type arguments

[3][todo]: Fix placeholders passed to options requiring code (only one pound sign)

[4][abandon]: Requirement: write to file if Write; Update: redundant with \cs{Ccool}+Write

```
397 Outer .value_required:n = false,
398 Outer .default:n = { \ensuremath{####1} },
399 Outer .initial:n = { \ensuremath{####1} },
```

Separ

```
400 Separ .code:n={
401   \__ccool_option_separ:n{#1}
402   \exp_last_unbraced:Nf
403   \__ccool_make_ccool:nnnn
404   {
405     { \g__ccool_option_param_tl }
406     { \g__ccool_option_inner_tl }
407     { \g__ccool_option_separ_tl }
408     { \g__ccool_option_outer_tl }
409   }
410 },
411 Separ .value_required:n = false,
412 Separ .default:n = { \c__ccool_option_separ_default_tl },
413 Separ .initial:n = { \c__ccool_option_separ_default_tl },
```

Write

```
414 Write .code:n = {
415   \bool_if:nTF{#1}
416   {\__ccool_log_open:}
417   {\__ccool_log_close:}
418 },
419 Write .value_required:n = false,
420 Write .default:n = \BooleanFalse,
421 Write .initial:n = \BooleanFalse
422 }
```

\CcoolRead

```
423 \NewDocumentCommand{\CcoolRead}
424 {o}
425 {
426   \IfValueTF{#1}
427   {\__ccool_log_read:e{#1}}
428   {\__ccool_log_read:}
429 }
```

(*End definition for* \CcoolRead. *This function is documented on page* 8.)

\CcoolVers

```
430 \NewDocumentCommand{\CcoolVers}
431 {}
432 {\use:c{ver@ccool.sty}}
```

(*End definition for* \CcoolVers. *This function is documented on page* 8.)

12   Closing

```
433 \ExplSyntaxOff
434 ⟨/package⟩
```