The \texttt{ottalt} package

Jesse A. Tov
tov@eecs.harvard.edu

This document corresponds to \texttt{ottalt} v0.11, dated 2013/03/14.

Contents

1 Introduction ........................................ 1
  1.1 The Line Break Hack .......................... 4
  1.2 Requirements ................................. 5

2 Command Reference .................................. 5
  2.1 Loading & Options ............................ 5
  2.2 Rule References ............................... 6
  2.3 Including Inference Rules .................... 7
    2.3.1 Configuration Macros .................. 8
  2.4 Nonterminal Styling ......................... 9
    2.4.1 Nonterminal Classes .................... 9
    2.4.2 Subscript-and-Prime Helpers .......... 11

3 Implementation ....................................... 12
  3.1 Loading & Options ............................ 12
    3.1.1 Renewing Ott Commands ................. 13
  3.2 Rule References ................................ 15
  3.3 Including Inference Rules and Grammars .... 16
  3.4 Nonterminal Styling .......................... 20

A Sample Ott Makefile .................................. 22

B Additional Ott Code .................................. 22

1 Introduction

The purpose of this package is to provide an alternate style for laying out grammars and inference rules generated by Ott (\url{http://www.cl.cam.ac.uk/~pes20/ott/}). The goal is to produce high-quality output suitable for inclusion in articles or technical reports, while making it easier to access various commands that Ott
defines in its output. (This package makes no attempt to compress Ott-generated
grammars to fit in space-limited situations, however.)

To use ottalt, one must generate standalone \LaTeX{} definitions from an .ott
source file. To make these suitable for inclusion from another \LaTeX{} docu-
ment, pass the -tex_wrap false option to Ott. For details, I've included a
sample Makefile in §A. Then, supposing you've generated Ott definitions in
\langle stem\rangle.ott.tex, load them in your \LaTeX{} document with

\inputott{\langle stem\rangle.ott}

Optionally, if you supply the -tex_name_prefix \langle prefix \rangle option to Ott, you
should also tell \inputott about the prefix, since that will tell it the names of
some commands to redefine: \inputott{\langle prefix\rangle\langle stem\rangle.ott}.

Then, the easiest way to print Ott-defined grammars and rules is with the
\drules and \nonterms commands, each of which takes a comma separated list
of Ott rule or nonterminal names. For example, suppose your Ott file contains
nonterminals t and e, like this:

1 (*ott)
2 grammar
3 t {{ tex \nonterm t }} :: tp_ ::= {{ com types }}
4 | a :: :: var {{ com type variable }}
5 | t1 -> t2 :: :: arr {{ com function }}
6 | all a . t :: :: all {{ com universal }}
7 | \{ t' / a \} t :: M :: subst
8 | ( t ) :: S :: parens
9
10 e {{ tex \nonterm e }} :: tm_ ::= {{ com terms }}
11 | x :: :: var {{ com variable }}
12 | \ x : t . e :: :: abs {{ com abstraction }}
13 | e1 e2 :: :: app {{ com application }}
14 | \ / a . e :: :: tabs {{ com type abstraction }}
15 | e [ t ] :: :: tapp {{ com type application }}
16 | \{ t / a \} e :: M :: tsubst
17 | ( e ) :: S :: parens
18
19 G {{ tex \nonterm G }} :: env_ ::= {{ com typing contexts }}
20 | empty :: :: nil {{ com empty }}
21 | G , x : t :: :: var {{ com variable assumption }}
22 | G , a :: :: tvar {{ com type variable assumption }}
23 (/ott)

Then you can print grammars for t and e like this:

\nonterms{t,e}

We could easily add G as well, or we could print it elsewhere. The result appears
in figure 1.

Similarly, we can define inference rules in Ott and then easily include them in
a document. Suppose we define these type rules using Ott:

\nonterms{t,e}
\[ \begin{align*} 
\tau & ::= \text{types} \\
& \quad | \alpha \quad \text{type variable} \\
& \quad | \tau_1 \to \tau_2 \quad \text{function} \\
& \quad | \forall \alpha.\tau \quad \text{universal} \\
\end{align*} \]

\[ \begin{align*} 
e & ::= \text{terms} \\
& \quad | x \quad \text{variable} \\
& \quad | \lambda x:\tau.e \quad \text{abstraction} \\
& \quad | e_1 e_2 \quad \text{application} \\
& \quad | \Lambda \alpha.e \quad \text{type abstraction} \\
& \quad | e[\tau] \quad \text{type application} \\
\end{align*} \]

Figure 1: An example of \texttt{\nonterms}

24 (*ott*)
25 defns Jtype ::= ’’ ::= 
26 defn G |- e : t ::= \text{typing} : T_ \{ \text{com term typing} \} \text{ by} 
27 
28 x : t in G 
29 ---------- ::= \text{Var} 
30 G |- x : t 
31 
32 G, x:t1 |- e : t2 
33 --------------------- ::= \text{Abs} 
34 G |- \lambda x:t1.e : t1 \to t2 
35 
36 G |- e1 : t' \to t 
37 G |- e2 : t' 
38 ------------- ::= \text{App} 
39 G |- e1 e2 : t 
40 
41 G, a |- e : t 
42 --------------------- ::= \text{TAbs} 
43 G |- \forall a.e : \text{all a.t} 
44 
45 G |- e : \text{all a.t'} 
46 G |- t ok 
47 --------------------- ::= \text{TApp} 
48 G |- e[t] : \{t/a\}t' 
49 */ott*

This defines macros such as \texttt{\ottt} for typesetting rule \texttt{T-Var}. We can display some or all of them using \texttt{\drules}. For example, we may want to show only three of them:

\texttt{\drules[T]}\{$[G |- e : t]$}\$
{\text{term typing, selected rules}}
{\text{Var, Abs, App}}
\[ \Gamma \vdash e : \tau \]

(term typing, selected rules)

\[ \begin{align*}
\text{T-VAR} & \quad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \\
\text{T-Abs} & \quad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1.e : \tau_1 \to \tau_2} \\
\text{T-App} & \quad \frac{\Gamma \vdash e_1 : \tau' \to \tau, \Gamma \vdash e_2 : \tau'}{\Gamma \vdash e_1 \ e_2 : \tau}
\end{align*} \]

Figure 2: An example of \( \texttt{rules} \)

The result appears in figure 2.\footnote{In the sample code, we use the double bracket notation that requires passing the \( \texttt{IsT\TeX} \) document through \texttt{Ott} as a preprocessor. This documentation is not actually processed in that way, but the \texttt{Makefile} in §A gives a rule for doing that preprocessing.}

The examples above use this package’s facility for managing nonterminal symbols. For example, the Ott code gives a tex morphism of \( \texttt{nonterm t} \) for nonterminal \( t \). This is set up in two stages. First, we declare a new group of nonterminals using \( \texttt{newNTclass} \), naming the new class \texttt{nonterm}:

\begin{verbatim}
\newNTclass{nonterm}
\end{verbatim}

This results in the definition of three new macros: \( \texttt{newnonterm} \) and \( \texttt{newnonterms} \) for declaring new nonterminal symbols belonging to the \texttt{nonterm} class, and \( \texttt{nonterm} \) for referring to them. We then declare five nonterminal symbols in the new class:

\begin{verbatim}
\newnonterm t \tau
\newnonterm e e
\newnonterm a \alpha
\newnonterm x x
\newnonterm G \Gamma
\end{verbatim}

This associates a short name, such as \( t \), with the actual code for printing that nonterminal, in this case \( \tau \). Then, we can refer to it as \( \texttt{nonterm t} \). This makes it convenient to configure how nonterminal symbols appear in Ott output. It includes support for switching between two versions of the nonterminals, which is useful, for example, to have color and black-and-white modes.

1.1 The Line Break Hack

The “line break hack” is a method for specifying line breaks between premises in Ott inference rules so that the line breaks appear in the final \( \texttt{IsT\TeX} \) document. The idea is to emit a \( \texttt{\LaTeX} \) control sequence, \( \texttt{ottlinebreakhack} \), after any premises where we want to force a break. Then, the definition of \( \texttt{ottpremise} \) from this package looks for that and uses it to force a break.

One way to do this is to add a production the Ott’s \texttt{formula} nonterminal, which allows specifying additional formulas that may be used as premises in judgments. I add a production that looks like this:
The effect of this is to allow writing \textbackslash\linebreakhack after any premise in an Ott inference rule (on the same line). This yields a formula that translates to \LaTeX just like the original formula, but with \textbackslash\linebreakhack after it.

Interpretation of the line break hack is not enabled by default. To turn it on, you need to pass the \texttt{lineBreakHack} option when loading the package. See §2.1 for information on the other two options related to line breaking.

1.2 Requirements

The \texttt{ottalt} package depends on five other packages. Three are part of the standard \LaTeX distribution, but two others must be downloaded.

\begin{itemize}
\item \texttt{iftthen} Standard \LaTeX package
\item \texttt{keyval} Standard \LaTeX package
\item \texttt{listproc} Non-standard, available at \url{http://www.eecs.harvard.edu/~tov/code/latex/}
\item \texttt{mathpartir} Non-standard, available at \url{http://cristal.inria.fr/~remy/latex/}
\end{itemize}

Additionally, \texttt{ottalt} will cooperate with the \texttt{hyperref} package, if it is loaded, to provide hyperlinks from inference rule references to their definitions. There is also an option to load the \texttt{supertabular} package and use it for presenting grammars.

2 Command Reference

2.1 Loading & Options

\texttt{\usepackage[\langleottalt-options\rangle]{ottalt}}

The package supports these options:

\begin{itemize}
\item \texttt{alternateNonterms}
  
  This tells the package to use the alternate styling for nonterminal symbols defined using the system described in §2.4.1. When a new nonterminal is declared with the \texttt{\new\{\langleclass\rangle\}\{\langlename\rangle\}\{\langletext\rangle\}\{\langlealt-text\rangle\}}, then it normally prints as \langle\text\rangle, but this option causes it to print as \langle\text\text\rangle instead.

\item \texttt{supertabular}
  
  Load the \texttt{supertabular} package and use the \texttt{supertabular} environment for grammars.
\end{itemize}
implicitPremiseBreaks, lineBreakHack, implicitLineBreakHack

These three options specify different linebreaking behavior in inference rules, all of which involve the “line break hack” (§1.1). If option implicitPremiseBreaks is enabled, then any line breaks specified by the line break hack are ignored, and all \inferrule* premises are separated by \\
, which allows \inferrule* to decide where to break lines between premises. If option lineBreakHack is enabled, then there are no implicit breaks: requested line breaks are passed to \inferrule as \\
, which forces a break, and other premises are separated by \and, which prohibits a break. Finally, option implicitLineBreakHack respected requested line breaks by passing them as \\
, but allows automatic line breaks elsewhere by using \\
to separate remaining premises.

\inputott{(prefix)}{(ott-tex-src)}

Load the Ott definitions from \{ott-tex-src\} and then redefine several Ott macros. The default names for Ott macros all begin with ott, but if the prefix has been changed by providing the \-tex_name_prefix \langle prefix \rangle option to Ott, then passing the same prefix to \inputott will cause it to redefine the correct macros.

\renewottcommands{(prefix)}

Redefine the Ott commands (using \langle prefix \rangle if supplied or ott by default). This is ordinarily called by \inputott, but can be used directly if the Ott definitions are loaded some other way first.

2.2 Rule References

This package provides a system for hyperlinking rule references to rule definitions. The links will be generated only if the hyperref package is loaded.

\ranchor{\langle rule-name \rangle}

Print \langle rule-name \rangle as a rule name and create a hypertarget with that name.

\rref{\langle rule-name \rangle_1, \ldots, \langle rule-name \rangle_k}
\Rref{\langle rule-name \rangle_1, \ldots, \langle rule-name \rangle_k}

Print \langle rule-name \rangle_1 through \langle rule-name \rangle_k as a nice list, with hyperlinks to the corresponding \ranchors if hyperref is loaded. For example,

\rref{T-Var,T-Abs,T-App}

prints

rules T-Var, T-Abs, and T-App

The uppercase variant \Rref is for the beginning as a sentence, as it capitalizes the introduction word (“Rule[s]”) by default. The starred variants \rref* and \Rref*
do not print the introduction word, but merely a bare list of rule references. The style of the references and text used to introduce them is configurable using by renewing several commands:

<table>
<thead>
<tr>
<th>command</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\rrefruletext</td>
<td>rule</td>
</tr>
<tr>
<td>\Rrefruletext</td>
<td>\expandafter\MakeUppercase\rrefruletext</td>
</tr>
<tr>
<td>\rrefruletext</td>
<td>\rrefruletext s</td>
</tr>
<tr>
<td>\Rrefruletext</td>
<td>\Rrefruletext s</td>
</tr>
<tr>
<td>\ranchorstyle</td>
<td>\rrefstyle</td>
</tr>
<tr>
<td>\rrefstyle</td>
<td>\normalfont\scshape</td>
</tr>
<tr>
<td>\wraparoundrref</td>
<td>\relax</td>
</tr>
</tbody>
</table>

Commands \rrefruletext and \Rrefruletext are used to introduce single rules made by \rref and \Rref, respectively. Commands \rrefruletext and \Rrefruletext are the plural versions, used to introduce lists rules made by \rref and \Rref. To change the style of rule anchors and rule references, redefine \ranchorstyle and \rrefstyle. Finally, \wraparoundrref is wrapped around each rule reference, which can be used to prevent line and page breaks within the rule name, among other things. (Older versions of hyperref crash when a link breaks between pages, so to prevent this, use \renewcommand\wraparoundrref{\mbox}.)

To redefine how lists are separated (as in “,” and “and”), see the documentation for the listproc package. The relevant commands to redefine are \FormatListSepTwo, \FormatListSepMore, and \FormatListSepLast.

2.3 Including Inference Rules

\drule{⟨opt⟩}{⟨rule-name⟩}

Include the Ott rule with name ⟨rule-name⟩. This translates a name like T-Var to the name that Ott uses for the rule, ⟨prefix⟩druleTXXVar. If the optional argument is given, then that is passed to the Ott rule command; otherwise we pass the empty argument.

This macro is primarily intended to be used in the drulepar environment. Multiple uses of \drule will try to space themselves using \and, which will be issued by all but the first rule in a scope. Thus, a group of rules should appear in a \TeX\ group. To suppress automatic spacing altogether, wrap each rule in its own group.
Create a block for setting rules. Argument \langle judgment\rangle will be set as a heading in a framed box on the left; or for the starred variant, \langle judgments\rangle should be a comma-separated list, each of which will be set in its own framed box. The description, \langle descr\rangle, is right aligned in text mode. Then the contents of the environment are set as a math paragraph using \texttt{mathpartir}'s \texttt{mathparpagebreakable} environment.

If the optional \langle rule-prefix\rangle is specified, then any uses of \texttt{drule} within the group will have \langle rule-prefix\rangle prepended to their rule name.

\begin{drulepar} \langle rule-prefix\rangle \{\langle judgment\rangle\} \{\langle descr\rangle\} \end{drulepar}
\begin{drulepar*} \langle rule-prefix\rangle \{\langle judgments\rangle\} \{\langle descr\rangle\} \end{drulepar*}

Create a block with the specified rules, which are the comma-separated argument. This is essentially a shortcut for using the \texttt{drulepar} environment and \texttt{drule} command. The above sample syntax is equivalent to

\begin{verbatim}
\begin{drulepar} \langle rule-prefix\rangle \{\langle judgment\rangle\} \{\langle descr\rangle\}
  \drule\{\langle rule-name\rangle\} [...]
\end{drulepar}
\end{verbatim}

\begin{rulesection} \langle rule-prefix\rangle \{\langle judgment\rangle\} \{\langle descr\rangle\} \end{rulesection}
\begin{rulesection*} \langle rule-prefix\rangle \{\langle judgments\rangle\} \{\langle descr\rangle\} \end{rulesection*}

Create a block with \langle judgment\rangle (or \langle judgments\rangle) and \langle descr\rangle heading, like \texttt{drulepar}, but the contents of the environment are (initial) in text mode. Set the optional \langle rule-prefix\rangle for its duration if specified.

\textbf{2.3.1 Configuration Macros}

These commands are used by the commands above, and may be redefined to change their behavior:

\begin{verbatim}
\begin{ottaltgrammar} \langle dimen\rangle \end{ottaltgrammar}
  \nt \{\langle nonterm-name\rangle\}
  ...
\end{verbatim}

Build an Ott grammar. The optional argument \langle dimen\rangle specifies the default vertical spacing between grammar items. Within the grammar, the command \texttt{nt}
is defined, which takes an Ott nonterminal name and adds its productions to the
grammar.

\texttt{nonterms \((\text{dimen}\)) \{(\text{nonterm-name})\,...\}\}}

This is a convenience macro for typesetting a grammar with the specified nontermi-
nals. It’s easily defined in terms of the \texttt{ottaltgrammar} environment.

\texttt{ottaltinferrule \{(\text{rule-name})\} \{(\text{infer-opts})\} \{(\text{premises})\} \{(\text{conclusion})\}}

For setting inference rules, by default this delegates to mathpartir’s \texttt{inferrule*}. The
\langle rule-name \rangle is passed to \texttt{inferrule*} using key \texttt{lab}, and the remaining
options \langle infer-opts \rangle are passed as is. This command may be redefined in order
to set inference rules with different options or to use a different mechanism than
\texttt{inferrule*}.

\texttt{drulesectionhead \{(\text{judgment})\} \{(\text{descr})\}}
\texttt{drulesectionhead* \{(\text{judgments})\} \{(\text{descr})\}}

These commands are used to set the heading for the \texttt{drulepar} and \texttt{rulesection}
environments. Argument \langle judgment \rangle is the judgment to set in a box on the left,
and \langle descr \rangle is the judgment description to set on the right. Or, \langle judgments \rangle is a
comma-separated list of judgments, each of which should be set in its own box.

<table>
<thead>
<tr>
<th>command</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>\FormatDruleSectionHead[1]</td>
<td>\fbox{#1}</td>
</tr>
<tr>
<td>\FormatDruleSectionHeads[1]</td>
<td>\fbox{\strut#1}</td>
</tr>
<tr>
<td>\FormatDruleSectionHeadRight[1]</td>
<td>\texttt{\textbf{#1}}</td>
</tr>
<tr>
<td>\FormatDruleSepTwo</td>
<td>,,</td>
</tr>
<tr>
<td>\FormatDruleSepMore</td>
<td>\FormatDruleSepTwo</td>
</tr>
<tr>
<td>\FormatDruleSepLast</td>
<td>\FormatDruleSepTwo</td>
</tr>
</tbody>
</table>

These are used to customize the output of \texttt{drulesectionhead} and \texttt{drulesectionhead*},
and by extension, the \texttt{drulepar} and \texttt{rulesection} environments. \texttt{FormatDruleSectionHead}
and \texttt{FormatDruleSectionHeads} specify how to format the left-side head (usual-
ly the form of a rule) for \texttt{drulesectionhead} and \texttt{drulesectionhead*}, re-
spectively; by default, they use a framed box, and the latter adds a \texttt{\strut} so
that several boxes have the same height. \texttt{FormatDruleSectionHeadRight} is
used to format the right-side head (usually a description). \texttt{FormatDruleSepTwo},
\texttt{FormatDruleSepMore}, and \texttt{FormatListSepLast} are used by \texttt{drulesectionhead*}
to separate several left-side heads. \texttt{FormatDruleSepTwo} is used if there are only
two heads to separate. If there are more, then \texttt{FormatDruleSepMore} is used for
all but the last separator, and \texttt{FormatDruleSepLast} is used between the last two.
2.4 Nonterminal Styling

2.4.1 Nonterminal Classes

Ott allows specifying the \LaTeX code for printing nonterminal symbols. For example, we may use the concrete syntax G in Ott for typing contexts, but want it to appear as Γ in typeset output. We can do this in Ott by specifying a “morphism” after declaring the nonterminal, like this:

\[
G \{\ tex \ \text{\textbackslash Gamma} \} :: \ env_ ::= \{\ com \ typing \ contexts \}
\]

I’ve found this method of specifying how nonterminals should be printed to be difficult to manage in a large Ott development, because it’s scattered throughout the Ott source, and because I often want to parameterize how nonterminals are printed, for example, to produce both color and black-and-white versions of the same document. This package provides a facility for managing nonterminal symbols so that instead, one writes something like

\[
G \{\ tex \ \text{\textbackslash nonterm G} \} :: \ env_ ::= \{\ com \ typing \ contexts \}
\]

in the Ott source and then can configure what \textit{\textbackslash nonterm G} means from the \LaTeX side.

To do this, we declare an “NT class” and then define the meaning of nonterminal symbols within it:

\[
\texttt{\newNTclass} \langle \text{mode-if} \rangle \langle \text{class-name} \rangle
\]

Declare a new nonterminal class named \langle \text{class-name} \rangle. If \langle \text{mode-if} \rangle is supplied, then it is used as a conditional to choose between regular and alternate styles for the nonterminals, as explained below. The conditional should take two arguments, and return the first for \texttt{true} and the second for \texttt{false}. (The default for \langle \text{mode-if} \rangle is \texttt{ifnotalternateNonterms}, which is set by the \texttt{alternateNonterms} package option.)

This command results in the definition of three new commands:

\[
\texttt{\new} \langle \text{class-name} \rangle \langle \text{style} \rangle \langle \text{nt-name} \rangle \langle \text{defn} \rangle
\]

Declare nonterminal name \langle \text{nt-name} \rangle to be printed as \langle \text{style} \rangle\langle \text{defn} \rangle in the regular case (when \langle \text{mode-if} \rangle is true) and \langle \text{defn} \rangle in the alternate case (when \langle \text{mode-if} \rangle is false).

\[
\texttt{\new} \langle \text{class-name} \rangle \langle \text{style} \rangle \langle \text{nt-name} \rangle \langle \text{defn} \rangle \langle \text{alt-defn} \rangle
\]

Declare nonterminal name \langle \text{nt-name} \rangle to be printed as \langle \text{style} \rangle\langle \text{defn} \rangle in the regular case (when \langle \text{mode-if} \rangle is true) and \langle \text{alt-defn} \rangle in the alternate case (when \langle \text{mode-if} \rangle is false).

\[
\texttt{\new} \langle \text{class-name} \rangle \langle \text{nt-name} \rangle
\]

Print nonterminal \langle \text{nt-name} \rangle.
Note that the mode selector (mode-if) takes effect when nonterminals are declared, not when they are used.

\texttt{\textbackslash ifnotalternateNonterms \{\textit{regular}\} \{\textit{alternate}\}}

Select between regular and alternate presentations. By default, this is defined to return (regular), but if the alternateNonterms package option is given when loading the package, then it returns (alternate) instead. This is used by \texttt{\textbackslash newNTclass} as the mode selector if none is explicitly supplied.

\texttt{\textbackslash @ifToif \{\textit{LaTeX-style-if}\}}
\texttt{\textbackslash ifTo@if \{\textit{TeX-style-if}\}}

Convert between \TeX-style conditional, which expects to see \texttt{\textbackslash else} (optionally) and \texttt{\textbackslash fi}, and \LaTeX-style conditional, which takes two arguments. For example, we can convert the \TeX-style \texttt{\textbackslash ifnum\textbackslash dim<5} to \LaTeX-style and give it two arguments, like so:

\texttt{\textbackslash ifTo@if \{\textbackslash ifnum\textbackslash dim<5\} \{(\textit{then-text})\} \{(\textit{else-text})\}}

Or, we can convert \LaTeX-style \texttt{\@ifundefined{somecs}} to \TeX-style:

\texttt{\textbackslash @ifToif \{\@undefined{somecs}\} \{\texttt{then-text}\}\textbackslash else \{\texttt{else-text}\}\textbackslash fi}

The former is useful to convert a \TeX-style conditional declared with \texttt{\newif} for passing to \texttt{\newNTclass}. Here’s an example. We begin by declaring a new \TeX conditional, which we’ll use to switch color on and off:

\texttt{\newif\ifcolor \ifcolortrue}

Later, we declare an nonterminal class for “calculus A”, whose presentation mode is controlled by \texttt{\ifcolor}:

\texttt{\newNTclass[\ifTo@if\ifcolor]{calcA}}

Then we can add nonterminal symbols to calculus A. For example, we declare a nonterminal named \texttt{t}, which prints as \texttt{τ} when \texttt{\ifcolor} was true when it was declared and as \texttt{τ} when \texttt{\ifcolor} was false:

\texttt{\textcolor{teal}\newcalcA[\textcolor{teal}] t \tau}

### 2.4.2 Subscript-and-Prime Helpers

Ott allows writing nonterminals and metavariables followed by digits and apostrophes to produce numeric subscripts and primes. For example, if \texttt{t} is set to print \texttt{τ}, then \texttt{t12\textasciitilde} prints \texttt{τ\textasciitilde12}. However, if we style \texttt{τ}, it will not automatically style the subscripts and primes as well. For example, if we give a declare \texttt{t} as above to be \texttt{calcA t}, which expands to \texttt{\textcolor{teal}\tau}, then \texttt{t12\textasciitilde} prints \texttt{τ\textasciitilde12}, which is unsatisfactory.

In this section, we describe commands that capture subscripts and primes that follow a nonterminal symbol so that we can style them along with the nonterminal. For example, you could define \texttt{calcA t} as
\textcolor{teal}{\texttt{t}}\texttt{\tau}, and then \texttt{t12}' appears as $\tau'_{12}$.

\[ \texttt{\textcolor{teal}{\texttt{\langle}} style \texttt{\textcolor{teal}{\texttt{\rangle}} \{\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} \{\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} \} \} \{\texttt{\textcolor{teal}{\langle}}} empty \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} subscript \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} prime \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} \} \} \{\texttt{\textcolor{teal}{\langle}}} empty \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} subscript \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} prime \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} \} \texttt{\textcolor{teal}{\texttt{\langle}} styler \texttt{\textcolor{teal}{\texttt{\rangle}} \{\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} subscript \texttt{\textcolor{teal}{\texttt{\rangle}} \} \ldots \} \{ prime \} \ldots \} \texttt{\textcolor{teal}{\texttt{\langle}} styler \texttt{\textcolor{teal}{\texttt{\rangle}} \{\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} subscript \texttt{\textcolor{teal}{\texttt{\rangle}} \} \ldots \} \{ prime \} \ldots \} \texttt{\textcolor{teal}{\texttt{\langle}} styler \texttt{\textcolor{teal}{\texttt{\rangle}} \{\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} \} \{\texttt{\textcolor{teal}{\langle}}} subscript \texttt{\textcolor{teal}{\texttt{\rangle}} \} \ldots \} \{ prime \} \ldots \}

This lets \texttt{\textcolor{teal}{\langle}} styler \texttt{\textcolor{teal}{\texttt{\rangle}}} have individual control over how each part is styled.

<table>
<thead>
<tr>
<th>Usage</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} \texttt{\textcolor{teal}{\texttt{\langle}}} overline \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} } {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
</tr>
<tr>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} \texttt{\textcolor{teal}{\texttt{\langle}}} underline \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} } {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
</tr>
<tr>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
<td>\texttt{\textcolor{teal}{\texttt{\langle}}} \texttt{\textcolor{teal}{\texttt{\langle}}} \texttt{\textcolor{teal}{\texttt{\langle}}} textcolor \texttt{\textcolor{teal}{\texttt{\rangle}}} {\texttt{\textcolor{teal}{\langle}}} color \texttt{\textcolor{teal}{\texttt{\rangle}} } {\texttt{\textcolor{teal}{\langle}}} text \texttt{\textcolor{teal}{\texttt{\rangle}} } {\texttt{\textcolor{teal}{\langle}}} capture-seq \texttt{\textcolor{teal}{\texttt{\rangle}} }</td>
</tr>
</tbody>
</table>

Convenience macros that serve as example uses of \texttt{\textcolor{teal}{\texttt{\langle}}}\texttt{\textcolor{teal}{\texttt{\rangle}}} for overlined, underlined, and colored nonterminals.

## 3 Implementation

We begin by loading several packages:

\begin{verbatim}
(*package)
\RequirePackage{mathpartir}
\RequirePackage{ifthen}
\RequirePackage{keyval}
\RequirePackage{listproc}
\end{verbatim}

### 3.1 Loading & Options

There are five package options. The first three determine the line break behavior between inference rule premises. The fourth turns on alternate presentation of nonterminal symbols using the NT class system, and the fifth uses the \texttt{supertabular} package for grammars.

\begin{verbatim}
\DeclareOption{implicitPremiseBreaks}{
\end{verbatim}
We set the default values for the options (automatic line breaks, ignoring the line break hack, and not in alternate NT mode), and then process the actual options.

\inputott inputs the file named by its argument and then calls \renewottcommands, which redefines several commands that are originally defined in the code generated by Ott. \ottaltprecommand is the current Ott prefix set by the most recent call to \renewottcommands.

3.1.1 Renewing Ott Commands

\renewcommand\ottaltprecommand{\}
\renewcommand\ottaltpremisebreak{\}
\DeclareOption{lineBreakHack}{\renewcommand\ottaltprecommand{\}}
\DeclareOption{implicitLineBreakHack}{\renewcommand\ottaltprecommand{\}}
\DeclareOption{alternateNonterms}{\let\ifnotalternateNonterms\@secondoftwo}
\DeclareOption{supertabular}{\ottalt@supertabulartrue}
\ProcessOptions
Load the supertabular package if requested.
\ifottalt@supertabular
\RequirePackage{supertabular}
\fi

\inputott[2][ott]{inputott[2][ott]}
\input[#2]{input[#2]}
\renewottcommands[#1]{\renewottcommands[#1]}
\newcommand\ottaltprecommand{ott}
\newcommand\ottaltprecommand{ott}
\let\ifnotalternateNonterms@secondoftwo
\newif\ifottalt@supertabular
\ProcessOptions

Helper macro for redefining Ott commands using the supplied prefix #1:
\def\renewcomm2nd#1{\def\renewcomm2nd#1{\expandafter\renewcommand\csname #1\endcsname
}
\ottrule This command cooperates with the next one, \ottpremise, to print with the proper line breaking premises. It does two things to set up evaluating the premises: it clears a token register in which premises will accumulator, and defines \ottalt@nextpremise to do nothing. The latter will be added to the token register before the next premise, so it's used by each premise to communicate to the next one what to place between them to produce spacing or a line break.

Each call to \ottpremise will add itself to the token register. Then \ottdrule passes the token register to \inferrule* to use as the premise(s) of the rule.

91 \renewottcommand{drule}[4][{}{
92 \def\ottalt@nextpremise{}
93 \ottalt@premisetoks={}
94 \ottalt@premisetoks=#2
95 \expandafter\ottalt@inferrule\expandafter
96 \{\the\ottalt@premisetoks}{##3}{##4}{##1}\}
97 }

\ottpremise This is the command that Ott wraps each inference rule premise with, so we redefine it to add each premise to the token register. It checks whether its contents end with \, in which case it sets \ottalt@nextpremise to produce a line break before the next premise (if line-break-hack-mode is on).

98 \renewottcommand{premise}[1]{%
99 \ottalt@premisetoks=
100 \expandafter\expandafter\expandafter
101 \{\expandafter\the\expandafter\ottalt@premisetoks
102 \ottalt@nextpremise##1\}
103 \ottalt@iflinebreakhack##1\ottlinebreakhack\ottalt@iflinebreakhack{
104 \let\ottalt@nextpremise\ottaltpremisebreak
105 \}
106 \let\ottalt@nextpremise\ottaltpre misssep
107 }
108 }

\ottusedrule We redefine \ottusedrule to automatically insert \textbackslash and between rules. We assume that for the first rule in the group, \ifottalt@firstrule will be true, and we change it to false for subsequent rules. This works provided each sequence of rules is in the same \TeX group, because we set locally.

109 \renewottcommand{usedrule}[1]{%
110 \ifottalt@firstrule
111 \ottalt@firstrulefalse
112 \else
113 \and
114 \fi
115 \ensuremath{##1}
116 }

\ottdefnblock Ott uses this command for enclosing all the rules in a judgment. We delegate to \drulepar, which is defined in §3.3.

117 \renewenvironment{#1defnblock}[3][]
\ottdrulename

To print rule names, we replace underscore with hyphen and turn the rule name into a rule anchor:
\renewottcomm@nd{drulename}[1]{%
  \ottalt@replace@cs\ranchor\_-{}##1\%
}\}

\ottprodline \ottprodnewline

For each production line in a grammar, we test for M and S rules and don’t print them. Because we add newlines in the \ottprodline, we make \ottprodnewline do nothing.
\renewottcomm@nd{prodline}[6]{%
  \ifthenelse{\equal{##3}{}}{\\ & & $##1$ & $##2$ & & $##5$ & $##6$
  }{}}%
\renewottcomm@nd{prodnewline}{\relax}

\ottgrammartabular

This is the command that Ott uses for grammars.
\renewottcomm@nd{grammartabular}[1]{%
\begin{ottaltgrammar}##1\end{ottaltgrammar}%
}%

\drule@h@lper \nonterm@h@lper

These two macros are not Ott commands that we are redefining, but helper macros that depend on the most recent prefix passed to \renewottcommands as #1.
\newcommand*\drule@h@lper[3]{%
  \expandafter\ifx\csname\ottaltcurrentprefix drule#3\endcsname\relax
  \PackageWarning{ottalt}{Unknown ott rule: #3}%
  \mbox{\textbf{(##2?)}}%
  \else
  \csname\ottaltcurrentprefix usedrule\endcsname
  \{(\csname\ottaltcurrentprefix drule#3\endcsname\#1)}%
  \fi
%
}\newcommand*\nonterm@h@lper[1]{\csname\ottaltcurrentprefix#1\endcsname}

The name of an Ott-defined nonterminal is just the prefix name concatenated with the nonterminal name:
\newcommand*\nonterm@h@lper[1]{\csname\ottaltcurrentprefix#1\endcsname}
3.2 Rule References

These commands are for formatting rule references. Redefine these to change how rule references appear.

\rrefruletext
\Rrefruletext
\rrefrulestext
\Rrefrulestext
\rrefstyle
\ranchorstyle
\wraparoundrref

These commands are for formatting rule references. Redefine these to change how rule references appear.

\newcommand{rrefruletext}{rule}
\newcommand{Rrefruletext}{\expandafter{MakeUppercase}{rrefruletext}}
\newcommand{rrefrulestext}{s}
\newcommand{Rrefrulestext}{s}
\newcommand{rrefstyle}{\normalfont{scshape}}
\newcommand{ranchorstyle}{rrefstyle}
\providecommand{wraparoundrref}{relax}

These commands are for formatting rule references. Redefine these to change how rule references appear.

\rref
\rref*
\Rref
\Rref*

These commands are for formatting rule references. Redefine these to change how rule references appear.

\newcommand{\rref}[2]{\FormatList{#1}{#2}{\one@rref}}
\newcommand{\rref@star}[1]{\FormatList{}{}{\one@rref}}
\newcommand{\@one@rref@nohyper}[1]{\wraparoundrref{\rrefstyle{#1}}}
\newcommand{\@ranchor@nohyper}[1]{\ranchorstyle{#1}}
\newcommand*{\rref@with}[2]{\FormatList{#1}{#2}{\one@rref}}
\newcommand*{\rref@star}[1]{\FormatList{}{}{\one@rref}}
\newcommand*{\@one@rref@nohyper}[1]{\wraparoundrref{\rrefstyle{#1}}}
\newcommand*{\@ranchor@nohyper}[1]{\ranchorstyle{#1}}

These commands are for formatting rule references. Redefine these to change how rule references appear.

\newcommand{\rref}[2]{\FormatList{#1}{#2}{\one@rref}}
\newcommand{\rref@star}[1]{\FormatList{}{}{\one@rref}}
\newcommand{\@one@rref@nohyper}[1]{\wraparoundrref{\rrefstyle{#1}}}
\newcommand{\@ranchor@nohyper}[1]{\ranchorstyle{#1}}

We try to detect the hyperref package, and define the rule reference macros to create hyperlinks if it's loaded. Rather than do the detection now, we wait until the end of the preamble, so that it doesn't matter in what order the packages are loaded.

\AtBeginDocument{
  \ifcsname hypertarget\endcsname
    \newcommand*{\rref@with}[2]{\FormatList{#1}{#2}{\one@rref}}
    \newcommand*{\rref@star}[1]{\FormatList{}{}{\one@rref}}
    \newcommand*{\@one@rref@nohyper}[1]{\wraparoundrref{\rrefstyle{#1}}}
    \newcommand*{\@ranchor@nohyper}[1]{\ranchorstyle{#1}}
  \endcsname
}
3.3 Including Inference Rules and Grammars

\drules Create a \drulepar and iterate through the requested rules, adding each one.

\begin{drulepar}[#1]{#2}{#3}
\@for\@ottalt@each:=#4\do{\expandafter\drule\expandafter{\@ottalt@each}}
\end{drulepar}

\drulepar A rule paragraph is merely a math paragraph wrapped in a rule section:

\begin{rulesection}[#1]{#2}{#3}
\begin{mathparpagebreakable}
\end{mathparpagebreakable}
\end{rulesection}

\rulesection A rule section is set as a one-item \texttt{trivlist}, with the heading set by \drulesectionhead. If the optional argument is given, then we save it in \ottalt@rulesection@prefix, so that \drule can add the prefix to each rule name. Initially, the rule section prefix is empty.

\begin{trivlist}
\item \ifx#1\relax\else\def\ottalt@rulesection@prefix{#1-}\fi
\drulesectionhead{#2}{#3}
\noindent
\end{trivlist}
This macros allow customizing the formatting of \drulesectionhead and \drulesectionhead*.

\FormatDruleSepTwo
\FormatDruleSepMore
\FormatDruleSepLast
\FormatDruleSectionHead
\FormatDruleSectionHeads
\FormatDruleSectionHeadRight

\drule
\drule@helper

To include a rule by name. Turns \- into XX, since that’s how the rule names become defined as macros. We call \drule@helper, which is defined by \renewottcommands to refer to the correct Ott prefix.

\ottaltinferrule
\ottalt@inferrule

Here we define several commands that help with typesetting rules. We begin with the final macro for typesetting rules. Redefine this to set inference rules differently. It is called by \ottalt@inferrule, which reorders its arguments from a useful order from the perspective of \otttdrule to the natural order for \ottaltinferrule.
These are used when dealing with rule premises to maintain rule spacing and eal with the line break hack. Several of these are defined here but used by \ottdrule, which is redefined by \renewottcommands.

\newif\ifottalt@firstrule \ottalt@firstruletrue
\newcommand{\ottalt@nextpremise}{\relax}
\newtoks{\ottalt@premisetoks}
\newcommand{\ottlinebreakhack}{\relax}
\def{\ottalt@iflinebreakhack#1\ottlinebreakhack #2\ottalt@iflinebreakhack}{\ifthenelse{\equal{#2}{}}\@secondoftwo\@firstoftwo}

This helper macro is used to search for a control sequence in a list of tokens and replace it with a given list of tokens. The usage is:

\ottalt@replace@cs{⟨next⟩}{⟨needle⟩}{⟨replacement⟩}{⟨haystack⟩}

This replaces ⟨needle⟩ with ⟨replacement⟩ in ⟨haystack⟩ and passes the result to ⟨next⟩. It is important that ⟨needle⟩ be a single token, and that ⟨haystack⟩ not be wrapped in curly braces. The extra curly braces before ⟨haystack⟩ and the double backslash after are necessary as well.

\newcommand{\ottalt@replace@cs}[5]{\ifx\null#5\relax\def{\ottalt@replace@cs@kont}{#1{#4}}\else\ifx#2#5\relax\def{\ottalt@replace@cs@kont}{\ottalt@replace@cs{#1}{#2}{#3}{#4{#5}}}\else\def{\ottalt@replace@cs@kont}{\ottalt@replace@cs{#1}{#2}{#3}{#4}}\fi\fi\ottalt@replace@cs@kont}

To print a grammar, we use the \ottaltgrammar environment defined right after this, and iterate through the requested nonterminal symbols, calling \nt with each.

\newenvironment{ottaltgrammar}[1][8pt]{\begingroup\trivlist\item}{\end{ottaltgrammar}}

The \ottaltgrammar environment sets up the table for printing grammars. It defines the \nt macro, which adds the grammar for a named nonterminal, for the extent of the environment.
We use a trick here to make `\nt` automatically insert a newline before all but the first nonterminal in a grammar. We define `\OTTALTNEWLINE` to insert a newline, but then right before the body of the environment starts, inside the `supertabular` environment, we redefine `\OTTALTNEWLINE` to do nothing. Because `supertabular` places each table cell in a group, the no-op redefinition prevents the first but not subsequent nonterminals from inserting a line break.

\[
\def\OTTALTNEWLINE{\[\#1\]}\%
\def\nt##1{\OTTALTNEWLINE\relax\nonterm@h@lper{##1}\ignorespaces}\
\newcommand\ottaltintertext[2]{\multicolumn{8}{l}{\begin{minipage}{##1}##2\end{minipage}}}\%
\ifottalt@supertabular\begin{supertabular}{llcllllll}\else\begin{tabular}{llcllllll}\fi\let\OTTALTNEWLINE\relax\ignorespaces}}\%
\ifundefined{ottafterlastrule}{\}{\ottafterlastrule}%
\ifottalt@supertabular\end{supertabular}\else\end{tabular}\fi\endtrivlist\endgroup\ignorespaces\%
\ifottalt@supertabular\else\fi\}

\newNTclass
This command defines three new commands with names based on its second argument. The `\new#2s` command uses the first argument to decide how to print nonterminals that are defined with it.

\[
\newcommand{\newNTclass}[2][\ifnotalternateNonterms]{\expandafter{\newcommand{\csname new#2s\endcsname}[4][{}]{#1{\expandafter{\newcommand{\csname ottalt@NT@#2@##2\endcsname}{##1{##3}}}#4}}}}\]

3.4 Nonterminal Styling

\newNTclass
\providecommand\@ifToif[1]{% #1\iftrue\iffalse}
\providecommand\ifTo@if[1]{% #1\expandafter\@firstoftwo\else\expandafter\@secondoftwo\fi}

Commands for converting between two styles of conditionals.

\@ifToif \ifTo@if

\newcommand\NTOVERLINE{
NTCAPTURE\overline}
\newcommand\NTUNDERLINE{
NTCAPTURE\underline}
\newcommand\NTTEXTCOLOR[1]{
NTCAPTURE{\textcolor{#1}}}

These are the commands for capturing subscripts and primes following a non-
terminal. We define \NTCAPTURE by passing the helper \NTCAPTURE@FINISH to
\NTCAPTURELOW, and it composes the nonterminal, subscript, and primes in the
usual way. Then \NTCAPTURELOW uses \NT@CAPTURE@LOOP to do the work of gath-
ering subscripts and primes.

\newcommand\NTCAPTURE[1]{
NTCAPTURELOW{\NTCAPTURE@FINISH{#1}}}
\newcommand\NTCAPTURE@FINISH[4]{#1{#2_{#3}#4}}
\newcommand\NTCAPTURELOW[2]{
\NT@CAPTURE@LOOP{#1}{#2}\relax\relax}

This is the main loop for the nonterminal capture macros. The arguments are:

#1 The operator to apply to the #2–#4 when done
#2 Hold the main nonterminal throughout the loop
#3 Accumulates subscripts
#4 Accumulates primes

The main loop checks whether the next character is an underscore or apostrophe,
and if so, dispatches to either \NTCAPTURE@SUB or \NTCAPTURE@PRIME to grab
it and loop again. If the next character is neither, then it finally calls #1 with the
nonterminal and the accumulated arguments.

\newcommand\NTCAPTURE@LOOP[4]{%
A Sample Ott Makefile

This Makefile has rules suitable for a workflow integrating Ott and \LaTeX{}. It should be available in the file \texttt{Makefile.sample}.

From an Ott file \texttt{⟨stem⟩.ott}, \texttt{make ⟨stem⟩.ott.tex} will generate an includable \LaTeX{} file contain definitions of macros for typesetting the grammars and rules in \texttt{⟨stem⟩.ott}. Given some other \LaTeX{} file \texttt{⟨stem⟩.tex} that wants to use Ott for preprocessing, \texttt{make ⟨stem⟩.mng.tex} will filter the file through Ott.

By default, Ott does not stop on all parse errors in filter mode, but emits \LaTeX{} code that may result in errors later when compiling the \LaTeX{} file. The rule for running Ott as a filter attempts to detect such errors, issue an error message, and halt (unless the environment variable \texttt{$\text{DONTSTOP}$} is set).

```
\%._ifnextchar _{%
  \%NT@CAPTURESUB{% #1% #2% #3% #4%}
)%_ifnextchar _{%
  \%NT@CAPTUREPRIME{% #1% #2% #3% #4%}
)%
{#1% #2% #3% #4%}
}}%
}
\def\NT@CAPTURESUB#1#2#3#4#5{\NT@CAPTURELOOP{#1}{#2}{#3#5}{#4}}
\def\NT@CAPTUREPRIME#1#2#3#4'#{\NT@CAPTURELOOP{#1}{#2}{#3}{#4'}}

⟨/package⟩
```

\begin{verbatim}
\%._ifnextchar _{%
  \%NT@CAPTURESUB{% #1% #2% #3% #4%}
)%_ifnextchar _{%
  \%NT@CAPTUREPRIME{% #1% #2% #3% #4%}
)%
{#1% #2% #3% #4%}
}}%
}
\def\NT@CAPTURESUB#1#2#3#4#5{\NT@CAPTURELOOP{#1}{#2}{#3#5}{#4}}
\def\NT@CAPTUREPRIME#1#2#3#4'#{\NT@CAPTURELOOP{#1}{#2}{#3}{#4'}}
\end{verbatim}

\texttt{//package}

\section{Sample Ott Makefile}

This Makefile has rules suitable for a workflow integrating Ott and \LaTeX{}. It should be available in the file \texttt{Makefile.sample}.

From an Ott file \texttt{⟨stem⟩.ott}, \texttt{make ⟨stem⟩.ott.tex} will generate an includable \LaTeX{} file contain definitions of macros for typesetting the grammars and rules in \texttt{⟨stem⟩.ott}. Given some other \LaTeX{} file \texttt{⟨stem⟩.tex} that wants to use Ott for preprocessing, \texttt{make ⟨stem⟩.mng.tex} will filter the file through Ott.

By default, Ott does not stop on all parse errors in filter mode, but emits \LaTeX{} code that may result in errors later when compiling the \LaTeX{} file. The rule for running Ott as a filter attempts to detect such errors, issue an error message, and halt (unless the environment variable \texttt{$\text{DONTSTOP}$} is set).

\begin{verbatim}
/*makefile*/
\OTT = ott $(OTTFLAGS)
\OTTFLAGS = -signal_parse_errors true \-
tex_wrap false -tex_show_meta false

%.ottdump: %.ott
$(OTT) -picky_multiple_parses true -i $< -writesys $@

%.ott.tex: %.ottdump
$(OTT) -readsys $< -o $@

%.mng.tex: %.tex %.ottdump
$(OTT) -readsys $*.ottdump -tex_filter $< $@

  @if grep '<< no parses (' $@ >/dev/null 2>&1 && 
    [ -z "$(DONTSTOP)" ]; then \
      echo; \ 
      echo "***** OTT PARSE ERROR(S) *****"; \ 
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "***** OTT PARSE ERROR(S) *****"; \
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "$RM" $@; \ 
      exit 1; \ 
    fi >&2

/*makefile*/
\end{verbatim}

\section{Sample Ott Makefile}

This Makefile has rules suitable for a workflow integrating Ott and \LaTeX{}. It should be available in the file \texttt{Makefile.sample}.

From an Ott file \texttt{⟨stem⟩.ott}, \texttt{make ⟨stem⟩.ott.tex} will generate an includable \LaTeX{} file contain definitions of macros for typesetting the grammars and rules in \texttt{⟨stem⟩.ott}. Given some other \LaTeX{} file \texttt{⟨stem⟩.tex} that wants to use Ott for preprocessing, \texttt{make ⟨stem⟩.mng.tex} will filter the file through Ott.

By default, Ott does not stop on all parse errors in filter mode, but emits \LaTeX{} code that may result in errors later when compiling the \LaTeX{} file. The rule for running Ott as a filter attempts to detect such errors, issue an error message, and halt (unless the environment variable \texttt{$\text{DONTSTOP}$} is set).

\begin{verbatim}
/*makefile*/
\OTT = ott $(OTTFLAGS)
\OTTFLAGS = -signal_parse_errors true \-
tex_wrap false -tex_show_meta false

%.ottdump: %.ott
$(OTT) -picky_multiple_parses true -i $< -writesys $@

%.ott.tex: %.ottdump
$(OTT) -readsys $< -o $@

%.mng.tex: %.tex %.ottdump
$(OTT) -readsys $*.ottdump -tex_filter $< $@

  @if grep '<< no parses (' $@ >/dev/null 2>&1 && 
    [ -z "$(DONTSTOP)" ]; then \
      echo; \ 
      echo "***** OTT PARSE ERROR(S) *****"; \ 
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "***** OTT PARSE ERROR(S) *****"; \
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "$RM" $@; \ 
      exit 1; \ 
    fi >&2

/*makefile*/
\end{verbatim}

\section{Sample Ott Makefile}

This Makefile has rules suitable for a workflow integrating Ott and \LaTeX{}. It should be available in the file \texttt{Makefile.sample}.

From an Ott file \texttt{⟨stem⟩.ott}, \texttt{make ⟨stem⟩.ott.tex} will generate an includable \LaTeX{} file contain definitions of macros for typesetting the grammars and rules in \texttt{⟨stem⟩.ott}. Given some other \LaTeX{} file \texttt{⟨stem⟩.tex} that wants to use Ott for preprocessing, \texttt{make ⟨stem⟩.mng.tex} will filter the file through Ott.

By default, Ott does not stop on all parse errors in filter mode, but emits \LaTeX{} code that may result in errors later when compiling the \LaTeX{} file. The rule for running Ott as a filter attempts to detect such errors, issue an error message, and halt (unless the environment variable \texttt{$\text{DONTSTOP}$} is set).

\begin{verbatim}
/*makefile*/
\OTT = ott $(OTTFLAGS)
\OTTFLAGS = -signal_parse_errors true \-
tex_wrap false -tex_show_meta false

%.ottdump: %.ott
$(OTT) -picky_multiple_parses true -i $< -writesys $@

%.ott.tex: %.ottdump
$(OTT) -readsys $< -o $@

%.mng.tex: %.tex %.ottdump
$(OTT) -readsys $*.ottdump -tex_filter $< $@

  @if grep '<< no parses (' $@ >/dev/null 2>&1 && 
    [ -z "$(DONTSTOP)" ]; then \
      echo; \ 
      echo "***** OTT PARSE ERROR(S) *****"; \ 
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "***** OTT PARSE ERROR(S) *****"; \
      grep -n '<< no parses (' $@>/dev/null 2>&1 && 
      echo "$RM" $@; \ 
      exit 1; \ 
    fi >&2

/*makefile*/
\end{verbatim}
B  Additional Ott Code

This is additional Ott code that is used to produce the Ott examples in §1.

```ott
(*ott)
grammar
metavar x {{ tex \nonterm x }} ::= {{ com variables }}
metavar a {{ tex \nonterm a }} ::= {{ com type variables }}
grammar
formula :: formula_ ::= {{ com formulas from meta-language }}
| judgement ::= judgement
| formula \\ :: :: lbh {{ tex [[formula]] \ottlinebreakhack }}
| x : t in G ::= xinenv
| a in G ::= ainenv
| G |- t ok ::= tokay
grammar
terminals :: terminals_ ::= |
  | -> ::= arr {{ tex \to }}
  | all ::= all {{ tex \forall }}
  | \ ::= lam {{ tex \lambda }}
  | \\ ::= Lam {{ tex \Lambda }}
  | in ::= in {{ tex \in }}
  | |- ::= proves {{ tex \vdash }}
  | empty ::= empty {{ tex \bullet }}
(/ott)
```

Change History

v0.1
General: Initial documented release

v0.11
General: Fixed typo in formula production that was preventing ott example from parsing.

`\drulesectionhead*`: Changed rulesection, rulesection* and derived environments to suppress page breaks after the rule section head.

v0.2
General: Included listproc.sty

v0.3
`\Rrefrulestext`: Bugfix: was RULES; now Rules.

v0.4
General: Uses \textbf{tabular} instead of \textbf{supertabular} environment for grammars by default now; added the \textbf{supertabular} package option to get the old behavior.

v0.5
`\ottaltintertext`: Added command for inserting unaligned text within a grammar.

v0.6
`\ottaltgrammar`: Bugfix: no longer assumes that `\ottafterlastrule` is already defined.

v0.7
`\nonterm@h@lper`: Using \textbf{instead of \bf} to select bold for the unknown rule error message.
\texttt{drulepar*}: Added \texttt{drulepar*} and \texttt{rulesection*} environments and \texttt{\rulesectionhead*} macro. Each of these is like its unstarred variant, except that its first non-optional argument is a comma-separated list of judgment names rather than a single one, and these are by default set in separate boxes.

\texttt{\ranchor}: Rule reference names now incorporate \texttt{\ottaltcurrentprefix}, so that the same document can use multiple rules with the same name in different prefix namespaces.
Symbols
\, ........................................ 229
\@ifToif ................................... 10, 322, 322
\@ifundefined ................................ 297
\@one@rref@nohyper .......................... 158, 160, 167, 175
\@ottalt@each ............................. 181, 182, 271, 272
\@ranchor@nohyper ......................... 158, 161, 172, 176
\relax ..................................... (13)
_ ..................................................... 121
_ ..................................................... 121
\alternateNonterms (package option) .... 5, (9, 10)
\and ........................................... (5), (13), 60, 113
\AtBeginDocument ........................... 162
\bullet ........................................... 392
\csname ....................................... 89, 134, 138, 139, 142, 308, 310, 312, 315, 316, 318, 319
\drulepar (environment) ..................... (7), 7, (8), (13), (16), 186
\drulepar* (environment) ................... (8), 7, 186
\drules ....................................... (1), (3), 7, 179, 179
\drulesectionhead ......................... (8), 8, (16, 17), 196, 199, 206, 211
\drulesectionhead* ........................ (8), (17), 196
\drulesectionheadMany ..................... 212, 219
\drulesectionheadOne ....................... 212, 214
\equal .......................................... 124, 255
\emph .......................................... 234
\endcsname 89, 134, 138, 139, 142, 163, 308, 310, 312, 315, 316, 318, 319
\endtrivlist ................................ 202, 209, 303
\ensuremath ................................... 115
\equal .......................................... 124, 255
\fbox ......................................... 232, 233
\forall ......................................... 387
\FormatDruleSectionHead ................. (8), 215, 220, 232
\FormatDruleSectionHead[1] ............ 8
\FormatDruleSectionHeadRight .......... (8), 216, 226, 229, 234
\FormatDruleSectionHeadRight[1] ..... 8
\FormatDruleSectionHeads ................. (8), 224, 229, 233
\FormatDruleSectionHeads[1] .......... 8
\FormatDruleSepLast ....................... (8), 9, 223, 239, 231
\FormatDruleSepMore ....................... (8), 9, 222, 229, 230
\FormatDruleSepTwo ........................ (8), 8, 221, 229, 239, 230, 231
\FormatList ................................ 158, 159, 224
\FormatListSepLast ......................... (15), 158, 159, 224
\FormatListSepMore ........................ (6), 8, 223
\FormatListSepTwo ........................ (6), 222
\FormatListSepTwo ........................ (6), 221
\fill ........................................... 216, 226
\hfill .......................................... 165
\hyperlink ..................................... 170

Index
Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols
\, ........................................... 229
\@ifToif ................................... 10, 322, 322
\@ifundefined ................................ 297
\@one@rref@nohyper .......................... 158, 160, 167, 175
\@ottalt@each ............................. 181, 182, 271, 272
\@ranchor@nohyper ......................... 158, 161, 172, 176
\relax ..................................... (13)
_ ..................................................... 121
\alternateNonterms (package option) .... 5, (9, 10)
\and ........................................... (5), (13), 60, 113
\AtBeginDocument ........................... 162
\bullet ........................................... 392
\csname ....................................... 89, 134, 138, 139, 142, 308, 310, 312, 315, 316, 318, 319
\drulepar (environment) ..................... (7), 7, (8), (13), (16), 186
\drulepar* (environment) ................... (8), 7, 186
\drules ....................................... (1), (3), 7, 179, 179
\drulesectionhead ......................... (8), 8, (16, 17), 196, 199, 206, 211
\drulesectionhead* ........................ (8), (17), 196
\drulesectionheadMany ..................... 212, 219
\drulesectionheadOne ....................... 212, 214
\equal .......................................... 124, 255
<table>
<thead>
<tr>
<th>Command</th>
<th>Page Numbers</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>\PackageWarning</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>\par</td>
<td>217, 227</td>
<td></td>
</tr>
<tr>
<td>\ProcessOptions</td>
<td>77</td>
<td></td>
</tr>
<tr>
<td>\anchor</td>
<td>(6), 6, 121,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>162, 169, 176</td>
<td></td>
</tr>
<tr>
<td>\anchors</td>
<td>6, 143, 148,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>161</td>
<td></td>
</tr>
<tr>
<td>\renewenvironment</td>
<td>117</td>
<td></td>
</tr>
<tr>
<td>\renewcommand</td>
<td>88, 91, 98,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>109, 120, 123</td>
<td></td>
</tr>
<tr>
<td></td>
<td>128, 129</td>
<td></td>
</tr>
<tr>
<td>\renewcommand*</td>
<td>5, 17, 18, 81</td>
<td></td>
</tr>
<tr>
<td></td>
<td>83, 86</td>
<td></td>
</tr>
<tr>
<td>\Rref</td>
<td>51, 52, 53, 54</td>
<td>79</td>
</tr>
<tr>
<td>\left</td>
<td>6, 150, 154</td>
<td></td>
</tr>
<tr>
<td>\right</td>
<td>6, 150, 150</td>
<td></td>
</tr>
<tr>
<td>\Ref*</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>\rref*</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>\rref@star</td>
<td>152, 156, 159</td>
<td></td>
</tr>
<tr>
<td>\rref@start</td>
<td>153, 157, 158</td>
<td></td>
</tr>
<tr>
<td>\rref@with</td>
<td>143, 146, 157</td>
<td></td>
</tr>
<tr>
<td>\Rreffrulestext</td>
<td>6, 143, 145,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>153</td>
<td></td>
</tr>
<tr>
<td>\Rreffrulestext*</td>
<td>6, 143, 144,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>146, 157</td>
<td></td>
</tr>
<tr>
<td>\rreffrulestext*</td>
<td>6, 143, 144,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>144, 145, 153</td>
<td></td>
</tr>
<tr>
<td>\rreffstyle</td>
<td>6, 143, 147,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>148, 160</td>
<td></td>
</tr>
<tr>
<td>\rulesection</td>
<td>7, (8), 196</td>
<td></td>
</tr>
<tr>
<td>\rulesection*</td>
<td>7, 196</td>
<td></td>
</tr>
<tr>
<td>\scshape</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>\strut</td>
<td>(8), 233</td>
<td></td>
</tr>
<tr>
<td>\textbf</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>\textcolor</td>
<td>334</td>
<td></td>
</tr>
<tr>
<td>\trivlist</td>
<td>(16), 197, 204, 278</td>
<td>386</td>
</tr>
<tr>
<td>\textbf</td>
<td></td>
<td></td>
</tr>
<tr>
<td>\underline</td>
<td>333</td>
<td></td>
</tr>
<tr>
<td>\usepackage</td>
<td>(5)</td>
<td></td>
</tr>
<tr>
<td>\vdash</td>
<td>391</td>
<td></td>
</tr>
<tr>
<td>\wraparoundrref</td>
<td>6, 143, 149, 160</td>
<td></td>
</tr>
</tbody>
</table>

27