TPS3 Facilities Guide for Users

1998 August 28

Peter Andrews
Matthew Bishop
Sunil Issar
Dan Nesmith
Frank Pfenning
Hongwei Xi

This manual is based upon work supported by NSF grants MCS81-02870, DCR-8402532, CCR-8702699, CCR-9002546, CCR-9201893, and a grant from the Center for Design of Educational Computing, Carnegie Mellon University. Any opinions, findings, and conclusions or recommendations are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1. Top-Level Commands

The internal name of this category is MEXPR. A top-level command can be defined using DEFMEXPR. Allowable properties are: ARGTYPES, WFFARGTYPES, WFFOP-TYPELIST, ARGNAMES, ARGHELP, DEFAULTFNS, MAINFNS, ENTERFNS, CLOSEFNS, PRINT-COMMAND, DONT-RESTORE, MHELP.

1.1. Top Levels

<n>BEGIN-PRFW Begin proofwindow top level. Open Current Subproof, Current Subproof & Line Numbers, and Complete Proof windows with text size determined by the value of the flag CHARSIZE. Printing in various windows can be modified by changing the flags PROOFW-ACTIVE, PROOFW-ALL, PROOFW-ACTIVE+NOS, BLANK-LINES-INSERTED and PRINTLINEFLAG. The initial size of the windows can be modified with the flags PROOFW-ALL-HEIGHT, PROOFW-ALL-WIDTH, PROOFW-ACTIVE-HEIGHT, PROOFW-ACTIVE-WIDTH, PROOFW-ACTIVE+NOS-HEIGHT, and PROOFW-ACTIVE+NOS-WIDTH; after the windows are open, they can simply be resized as normal. PSTATUS will update the proofwindows manually if necessary. Close the proofwindows with END-PRFW.

<n>DO-GRADES Invoke the grading package.

<n>ED *edwff* Enter the editor on a given wff. Editor windows may be initialised, depending the values of the flags EDWIN-TOP, EDWIN-CURRENT, EDWIN-VPFORM. The flags BLANK-LINES-INSERTED and CHARSIZE determine the layout of these windows. The flags EDWIN-{CURRENT,TOP,VPFORM}-WIDTH and EDWIN-{CURRENT,TOP,VPFORM}-HEIGHT determine the intial size of these windows; they may be resized after they are opened in the usual way. WARNING: Since editing is non-destructive, nothing is done with the result of the editing process!

<n>END-PRFW End proofwindow top level; close all open proofwindows.

<n>HISTORY *n reverse*
Show history list. Shows the N most recent events; N defaults to the value of HISTORY-SIZE, showing entire history list. Values of N that are greater than HISTORY-SIZE have the same effect as the default value. REVERSE defaults to NO; if YES, most recent commands will be shown first.

<n>LIB Enter the library top-level.

<n>MATE *gwff deepen reinit window*
Begin an expansion proof for a gwff.

<n>MTREE *gwff deepen reset window*
Begin to enter the mating tree top level.

<n>POP Return from a top level started with PUSH.

<n>PUSH Start a new top level. This command is almost useless, except from within a prompt (e.g. one can type PUSH in the middle of converting an etree to a ND proof interactively, call SCRIBEPROOF, and then type POP to return to the conversion).

<n>REVIEW Enter REVIEW to examine and change flags or parameters.

<n>TEST *gwff deepen reinit window*
Enter the test top level. In this top level, the user can search for an optimal mode in which to prove a particular theorem, by defining a list of flags to be varied and then running matingsearch repeatedly with different flag settings.

<n>UNIFORM-SEARCH *gwff window mode slist modify*
Enter the test top level to search for any mode that will prove a given theorem. The mode provided by the user should list flag settings that are not to be varied, and the searchlist provided by the user should list all of the flags to be varied. The default settings for the mode and searchlist are UNIFORM-SEARCH-MODE and UNIFORM-SEARCH-2. If you opt for the searchlist to be automatically modified, TPS will inspect the given wff to check whether it is first order, whether it contains any definitions, whether it contains any equalities (and if so whether the LEIBNIZ and ALL instantiations are different), and whether it has any possible primitive substitutions, and will then remove or modify any unnecessary flags from the searchlist (respectively, unification bounds will be deleted, REWRITE-DEFNS will be deleted, REWRITE-EQUALITIES will be deleted or modified, and DEFAULT-MS will be changed to a search without option sets). Also, if you opt for the searchlist to be modified and there is a proof of this theorem in memory, AUTO-SUGGEST will be run and you will be asked whether to modify the searchlist using the results it provides.

After entering the test top level with this command, type GO ! to start searching for a successful mode.

<n>UNIFORM-SEARCH-L *goal support line-range window mode slist modify*
Enter the test top level to search for any mode that will prove a given lemma. (Compare DIY-L) The mode provided by the user should list flag settings that are not to be varied, and the searchlist provided by the user

should list all of the flags to be varied. The default settings for the mode and searchlist are UNIFORM-SEARCH-MODE and UNIFORM-SEARCH-2. If you opt for the searchlist to be automatically modified, TPS will inspect the given wff to check whether it is first order, whether it contains any definitions, whether it contains any equalities (and if so whether the LEIBNIZ and ALL instantiations are different), and whether it has any possible primitive substitutions, and will then remove or modify any unnecessary flags from the searchlist (respectively, unification bounds will be deleted, REWRITE-DEFNS will be deleted, REWRITE-EQUALITIES will be deleted or modified, and DEFAULT-MS will be changed to a search without option sets). After entering the test top level with this command, type GO ! to start searching for a successful mode.

<n>UNIFY Enter the unification top-level. The user can define disagreement sets using the command ADD-DPAIR available in the unification top-level. If you are entering from the MATE top level, the unification tree associated with the active-mating is passed on to the unification top-level. Any changes made to this tree are destructive. Applicable only for a higher-order unification problem. Uses MS88-style unification.

1.2. Help

<n>? Type ? to obtain a list of possible options.

<n>?? Type ?? to get general help on TPS, command completion and history substitution.

<n>ABBREVIATIONS *show-defns*
This command will list the names of all abbreviations available in TPS.

<n>ENVIRONMENT
Helps to find out about TPS' current environment, i.e. categories of TPS objects, commands, argument types, logical constants, etc.

<n>HELP *keyword* Give information about a TPS object like a command or argument type. The amount of help given for inference rules may be changed by setting the flag SHORT-HELP.

<n>HELP* *keywords*
Give information about each of a list of TPS objects. This is equivalent to doing HELP on each of them. The amount of help given for inference rules may be changed by setting the flag SHORT-HELP.

<n>HELP-GROUP *keywords*
Give information about a group of TPS objects; specifically, given the name of a category, a context, or a top level, list the help messages for every object in that class. If given a list of names, it will list the help messages for all the objects that fall into the intersection of these classes (e.g. HELP-GROUP (MEXPR REWRITING) will show all the top-level commands in the context REWRITING). NOTE: Remember that the name of a context is not necessarily the name that prints on the screen; do HELP CONTEXT to show their real names.

<n>LIST-RULES List all rules with their suggestion priority.

<n>OOPS *position replacement*
Replace the word at a given position in the previous line with another word. Positions start from 0, and the substituted-for command will be entered into the command history list, so for example: <9>HELP GR-FILENAMES <10>OOPS 0 LIST (calls LIST GR-FILENAMES instead) <11>OOPS 1 GR-MISC (calls LIST GR-MISC)

<n>PROBLEMS *show-defns*
This command will list the names of all exercises available in ETPS.

<n>SEARCH *phrase search-names*
Look for a key phrase in all help strings (or just all names) of TPS objects. See also KEY, in the review top level (where it searches through the flags) and the library top level (where it searches through the library objects).

1.3. Collecting Help

<n>CHARDOC *output-style styles filename*
List the special characters of certain output styles in a TeX or Scirbe file. The output file can be processed by TeX or Scribe and will have multicolumn format.

<n>COLLECT-HELP *modules categories filename*
Collect help for the specified modules into a file. Prints out a # every time it finds a help message, and a * every time it finds a TPS object with no help message.

<n>HELP-LIST *category filename*
List all help available for objects of the given category into a file.

<n>HTML-DOC *directory*
Produce HTML documentation in the specified directory. This requires an empty directory and a lot of disk space,

and will take quite some time to produce.

<n>QUICK-REF *filename*
Produce a quick reference to the rules available in TPS.

<n>SCRIBE-DOC *category-list context-list filename*
Produce Scribe documentation about the specified categories.

1.4. Concept

<n>LOADKEY *key mssg*
Load one of the function keys f1-f10 on a concept terminal with a string.

<n>RESET Put a Concept terminal into correct mode and load the function keys.

1.5. Starting and Finishing

<n>ALIAS *name def*
Define an alias DEF for the symbol NAME. Works just like the alias command in the Unix csh. If the value of NAME is *ALL*, all aliases will be printed; if the value of DEF is the empty string, then the current alias definition of NAME will be printed. See UNALIAS.

<n>CLEANUP If the proof is complete, will delete unnecessary lines from a proof. It may also eliminate or suggest eliminating unnecessary hypotheses. If the proof is incomplete, will do a partial cleanup in which only unnecessary lines justified by SAME will be removed.

<n>DONE Signal that the current proof is complete.

<n>EXERCISE *excno*
Start the proof of a new exercise.

<n>EXIT Exit from TPS.

<n>NEWS Type TPS news on the terminal.

<n>PROVE *wff prefix num*
Start a new proof of a given wff.

<n>RECONSIDER *prefix*
Reconsider a proof. The following proofs are in memory:

For more details, use the PROOFLIST command.

<n>REMARK *remark*
Send a message to the teacher or maintainer.

<n>SUMMARY Tells the user what exercises have been completed.

<n>UNALIAS *name*
Remove an alias for the symbol NAME. Like the Unix csh unalias, except that NAME must exactly match the existing alias; no filename completion is done.

1.6. Printing

<n>BUILD-PROOF-HIERARCHY
This command builds hierarchical information into the proof outline. The information includes associations between lines and linear chains of inferences which trace the consequences of the most recent hypothesis of a line. That is, a line

ln) Hn,m |- an

would be associated with a linear chain of lines l1,...,ln where m is the line corresponding to the most recent hypothesis and the proof would justify the modified lines

l1) H1,m |- l1 l2) H2,l1 |- l2 l3) H3,l2 |- l3 . . . ln) Hn,ln-1 |- ln

where H1 < H2 < . . . < Hn (subset relation).

That is, we trace the consequences of the hypothesis m to the consequence ln. Such a linear chain is on one level of the hierarchy. One level down on the hierarchy would be the linear chains associated with each of the lines used to justify l1,...,ln (except those which appear in the chain l1,...,ln). If the proof is complete, then lines l1 and m will be the same.

Lines without hypotheses are also associated with such

<n>DEPTH *num* Causes all subformulas at depth greater than n to be printed as &.

<n>EXPLAIN *line depth*
This command explains a line of a proof outline. In particular, the command BUILD-PROOF-HIERARCHY builds dependency information into a proof outline which allows the proof outline to be viewed as a hierarchy of subproofs (see help for BUILD-PROOF-HIERARCHY). The command EXPLAIN shows the lines included in the levels of this hierarchy (to the specified depth) starting at the level associated with the specified line. Some flags which affect the printing include: PRINT-COMBINED-UIS, PRINT-COMBINED-UGENS, PRINT-COMBINED-EGENS, and PRINT-UNTIL-UI-OR-EGEN.

<n>FIND-LINE *wff vars meta*
Find all lines matching a certain wff, up to alphabetic change of bound variables and (possibly) alphabetic change of a given list of free variables. Optionally, you can treat the remaining free variables as matching any given term (as you might do if you were asserting an axiom). e.g. (suppose P is an abbreviation or constant): FIND-LINE "P a" () NO finds all lines that say "P a" FIND-LINE "P a" ("a") NO also finds "P x" and "P y" FIND-LINE "P a" () YES finds all the above, plus "P [COMPOSE f g]" FIND-LINE "a x" ("x") YES finds all lines of the form "SOME-TERM some-var"

<n>PALL Print all the lines in the current proof outline.

<n>PBRIEF *depth* This command prints a proof outline, hiding some lines. In particular, the command BUILD-PROOF-HIERARCHY builds dependency information into a proof outline which allows the proof outline to be viewed as a hierarchy of subproofs (see help for BUILD-PROOF-HIERARCHY). The command PBRIEF shows the lines included in the top levels of this hierarchy (to the specified depth). PBRIEF is essentially a call to the command EXPLAIN with the last line of the proof outline as the LINE argument (see help for EXPLAIN). Some flags which affect the printing include: PRINT-COMBINED-UIS, PRINT-COMBINED-UGENS, PRINT-COMBINED-EGENS, and PRINT-UNTIL-UI-OR-EGEN.

<n>PL *num1 num2* Print all proof lines in a given range.

<n>PL* *print-ranges*
Print all proof lines in given ranges.

<n>PLINE *line* Print a specified line.

<n>PPLAN *pline* Print a planned line and all its supports.

<n>PRINT-PROOF-STRUCTURE
This prints the structure of the proof outline. The structure is generated by BUILD-PROOF-HIERARCHY. Linear chains of line numbers are printed which indicate the logical chains of inferences. Each link in a linear chain is indicated by an arrow (l1)->(l2) where l1 and l2 are line numbers. If line l2 does not follow in a single step from l1 (i.e., by a single application of an inference rules), then PRINT-PROOF-STRUCTURE will also show the linear chains of inference used to justify (l1)->(l2). Some lines (such as those without hypotheses and planned lines) are ceptions. These top level lines are sometimes printed alone (instead of in arrow notation). This could be read TRUE->(l) to maintain consistent notation, but the notation (l) appears more readable in practice.

<n>PRW *gwff* Print real wff. Turns off special characters (including FACE definitions), infix notation, and dot notation, and then prints the wff.

<n>PW *gwff* Print gwff.

<n>PWSCOPE *gwff* print gwff with all brackets restored.

<n>PWTYPES *gwff* Prints a wff showing types.

<n>SHOWNOTYPES
Suppress the printing of types on all wffs.

<n>SHOWTYPES From now on show the types on all wffs.

<n>TABLEAU *line* Print the part of the proof which justifies the given line, in a natural deduction tableau format.

<n>^P Print current plan-support pair in the proof.

<n>^PN Print current plan-support pair in the proof, as in ^P, but also print just the line numbers of the other lines in the proof.

1.7. Saving Work

<n>EXECUTE-FILE *comfil execprint outfil stepping*
Execute commands from a SAVE-WORK file. Call this from the main top level or the proofwindows top level of TPS. Note that this will not save subsequent commands in the same file, which distinguishes it from RESTORE-WORK. Single-stepping only works between commands on the main top level; it will not stop at prompts which are internal to a command, nor between commands on a different top level. To force a work-file to stop in such a place, use the PAUSE command when creating the work file. If you are single-stepping through a file, you can abort at any time by typing ^G<RETURN>.

<n>FINDPROOF *name*
Searches your home directory and the directories listed in SOURCE-PATH, looking for a proof whose name contains the given string.

<n>FINISH-SAVE
Finishing saving work in a file. The difference between STOP-SAVE and FINISH-SAVE is: the former is temporary because you can use RESUME-SAVE to resume saving work into the same file; the latter closes the output stream, so you can not save work into the same file after executing it.

<n>PAUSE Force a work file to stop and query the user. PAUSE, like ABORT, is valid both as a top-level command and as a response to a prompt; it prints the message "Press RETURN, or ^G RETURN to abort.", waits for such a response from the user, and then repeats the original prompt. This command is of no use unless a work file is being created; see EXECUTE-FILE for more details.

<n>RESTORE-WORK *comfil execprint outfil*
Execute commands from a SAVE-WORK file and continue to save in that file. See EXECUTE-FILE for more information.

<n>RESTOREPROOF *savefile*
Reads a natural deduction proof from a file created by SAVEPROOF and makes it the current proof. A security feature prevents the restoration of saved proofs which have been altered in any way. Retrieve any definitions which are used in the proof and stored in the library before restoring the proof. If you don't specify a directory, it will first try your home directory and then all the directories listed in SOURCE-PATH.

<n>RESUME-SAVE
Use this command to resume saving commands into the most recent save-work file. Unlike RESTORE-WORK, this command doesn't execute commands from the file, but simply appends subsequent commands to the file. You can not use this command if you are already saving work. Also, you may run into trouble if you forgot to save some commands.

<n>SAVE-FLAGS-AND-WORK *savefile*
Start saving commands in the specified file, first storing all flag settings.

<n>SAVE-SUBPROOF *savefile lines subname*
Saves part of the current natural deduction proof to the specified file in a form in which it can be restored. The line ranges specified will be increased to include all the other lines on which the given lines depend. See the help message for LINE-RANGE to find out what a line-range should look like. An example list is: 1--10 15--23 28 34--35 Also creates a new proof in memory with the given name, and makes that the current proof. Use RESTOREPROOF to restore the proof. Overwrites the file if it already exists.

<n>SAVE-WORK *savefile*
Start saving commands in the specified file. These commands can be executed subsequently by using EXECUTE-FILE or RESTORE-WORK. If you are creating a work file for a demonstration, and need it to pause at certain points as it is reloaded by TPS, then see the help message for EXECUTE-FILE for more information on how to do this.

<n>SAVEPROOF *savefile*
Saves the current natural deduction proof to the specified file in a form in which it can be restored. Use RESTOREPROOF to restore the proof. Overwrites the file if it already exists.

<n>SCRIPT *scriptfile if-exists-append*
Saves a transcript of session to a file. If the current setting of STYLE is SCRIBE or TEX, an appropriate header will be output to the script file (unless the file already exists). **NOTE** If you start SCRIPT from a PUSHed top level, be sure to do UNSCRIPT before you POP that top level, or your transcript may be lost. The same also applies to starting SCRIPT from subtoplevels such as MATE; you can enter further subtoplevels like LIB and ED from the MATE top level, and SCRIPT will carry on recording, but before leaving the MATE top level you should type UNSCRIPT or your work will be lost.

<n>STOP-SAVE Stop saving commands in a SAVE-WORK file.

<n>UNSCRIPT Closes the most recent file opened with the SCRIPT command.

1.8. Saving Wffs

<n>APPEND-WFF *weak-label help-string filename*
Append a definition of a weak label to a file. If the file does not yet exist, it will be created. You may wish to use LIB instead.

<n>APPEND-WFFS *weak-labels filename*
Append the definitions of a list of weak labels to a file. If the file does not yet exist, it will be created. You may wish to use LIB instead.

1.9. Printing Proofs into Files

<n>PRINTPROOF *filename*
Print the current proof into a file.

<n>SCRIBEPROOF *filename timing*
Print the current proof into a MSS file. After leaving TPS, run this .MSS file through Scribe and print the resulting file.

<n>SETUP-SLIDE-STYLE
Sets flags to produce slides in scribe style.

<n>SLIDEPROOF *filename*
Print the current proof into a MSS file. Use this command to make slides. After leaving TPS, run this .MSS file through Scribe and print the resulting file.

<n>TEXPROOF *filename timing*
Print the current proof into a tex file. After leaving tps, run this .tex file through tex and print the resulting file.

1.10. Proof Outline

<n>CREATE-SUBPROOF *lines subname*
Creates a new proof in memory from the given lines, plus all the lines on which they depend, and makes that the current proof.

<n>LINE-COMMENT *line comment*
Attach a comment to a given existing line. The comment will be parsed for gwffs and line numbers as follows: anything enclosed in # symbols is assumed to be a gwff, and anything enclosed in $ symbols is assumed to be the number of an existing line. Line numbers in comments will be updated as lines are moved around; gwffs will be printed in the current STYLE. Examples: "1st copy of line 5, instantiated with #COMPOSE#" "2nd copy of line 5, instantiated with ITERATE" "3rd copy of line 5, instantiated with #a OR b#" (The first prints the definition of COMPOSE; the second prints the word "ITERATE", and the third prints the given gwff. If line 5 is subsequently renumbered, the line number will change in all these comments.)

<n>MERGE-PROOFS *proof subproof*
Merges all of the lines of a subproof into the current proof. If EXPERTFLAG is NIL, no line number may occur in both proofs. If EXPERTFLAG is T, then if a line number occurs in both proofs, the lines to which they refer must be the same (with one exception: if one is a planned line and the other is the same line with a justification, then the justified line will overwrite the planned one). Compare TRANSFER-LINES.

The following proofs are in memory:

For more details, use the PROOFLIST command.

<n>PROOF-COMMENT *comment*
Attaches a comment to the current proof. The default value is the current comment. Uses the same comment syntax as LINE-COMMENT; see the help message of that command for more information. You can see the comments on all the current proofs by using PROOFLIST.

<n>PROOFLIST Print a list of all proofs or partial proofs currently in memory. Also prints the final line of each proof and the comment, if any, attached to it.

<n>TRANSFER-LINES *proof subproof lines*
Copies all of the given lines of a subproof, and all lines on which they depend, into the current proof. If EXPERTFLAG is NIL, no line number may occur in both proofs. If EXPERTFLAG is T, then if a line number occurs in both proofs, the lines to which they refer must be the same (with one exception: if one is a planned line and the other is the same line with a justification, then the justified line will overwrite the planned one). Different comments from two otherwise identical lines will be concatenated to form the comment in the resulting proof.

This is equivalent to CREATE-SUBPROOF followed by MERGE-PROOFS.

The following proofs are in memory:

For more details, use the PROOFLIST command.

1.11. Mating search

<n>CLOSE-TESTWIN
Closes the window that displays the test-top and TPS-TEST summary. Use ..../tps/utilities/vpshow (from a shell, not from TPS) to view the output file again.

<n>DIY *goal support window*

DO IT YOURSELF. Calls matingsearch procedure specified by the flag DEFAULT-MS with specified planned line and supports, then translates the resulting proof to natural deduction. Allows some of the output to be sent to a separate vpform window (equivalent to issuing the OPEN-MATEVPW command before typing DIY).

<n>DIY-L *goal support window range*
DIY for lemmas. Behaves as for DIY, but puts all new lines into a specified range rather than scattering them throughtout the proof.

<n>MONITOR Turns the monitor on, and prints out the current monitor function and parameters. See NOMONITOR. See also QUERY-USER for an alternative way to monitor the progress of the matingsearch. For a list of monitor functions, type MONITORLIST. To change the current monitor function, enter the name of the desired new monitor function from the main top level or the mate top level.

<n>MONITORLIST
List all monitor functions.

<n>NOMONITOR Turns the monitor off, and prints out the current monitor function and parameters. See MONITOR. For a list of monitor functions, type MONITORLIST. To change the current monitor function, enter the name of the desired new monitor function from the main top level or the mate top level.

1.12. MS91-6 and MS91-7 search procedures

<n>SEARCH-ORDER *num vpf verb*
Generates the first n option sets that will be searched under the current flag settings (assuming that the first (n-1) searches fail because they run out of time rather than for any other reason). This will show the names and weights of the option sets, the primitive substitutions and duplications. Note : "Ordinary" duplications are duplications that have not had a primsub applied to them. So, for example, "X has 2 primsubs plus 3 ordinary duplications" means that the vpform now contains five copies of the relevant quantifier, two of which have had primsubs applied to them.

1.13. Proof Translation

<n>AUTO-SUGGEST
Given a completed natural deduction proof (which must be the current dproof; use RECONSIDER to return to an old proof), suggest flag settings for an automatic proof of the same theorem.

This will also automatically remove (or attempt to remove) all uses of SUBST= and SYM= from the proof (you will be prompted before this happens, as it permanently modifies the proof).

This will show all of the instantiations (and primitive substitutions) that are necessary for the proof, and suggest settings for NUM-OF-DUPS, MAX-MATES, DEFAULT-MS, MAX-PRIM-DEPTH, MAX-PRIM-LITS and REWRITE-DEFNS

<n>ETREE-NAT *prefix num tac mode*
Translates the current expansion proof, which is value of internal variable current-eproof, into a natural deduction style proof. The default value of the tactic is given by the flag DEFAULT-TACTIC.

<n>NAT-ETREE *prefix*
Translates a natural deduction proof, (which must be the current dproof -- use RECONSIDER to return to an old proof in memory), into an expansion proof. This will not work on all proofs: in particular, proofs containing ASSERT of anything but REFL= and SYM=, proofs using rewrite rules and proofs containing SUBST= or SYM= cannot be translated at present.

Note: The command AUTO-SUGGEST will (optionally) attempt to remove instances of SUBST= and SYM= from a proof; running AUTO-SUGGEST before using NAT-ETREE may improve your chances of a successful translation.

<n>NAT-ETREE-OLD *prefix*
Translates a natural deduction proof into an expansion proof using the old method. The default value is the name of the current proof, namely the value of DPROOF.

<n>PFNAT *proof* To generate a NATREE from given proof and store it in CURRENT-NATREE. This may evolve into a command for rearranging natural deduction style proofs.

<n>PNTR Print out the current natree stored in CURRENT-NATREE. Mainly for the purpose of debugging.

<n>TIDY-PROOF *old-prfname new-prfname*
Translate a ND proof to an eproof and back again (into a proof with a new name) in the hope of tidying it up a bit. Equivalent to NAT-ETREE; MATE ! ; PROP-MSEARCH ; MERGE-TREE ; LEAVE ; ETREE-NAT ; CLEANUP ; SQUEEZE

1.14. Unification

<n>LEAST-SEARCH-DEPTH
Print the least needed unification tree depth for the last proven higher-order theorem. Also suggest to lower flags MAX-SEARCH-DEPTH to the least needed value if they are greater than it.

1.15. Tactics

<n>ECHO *echothing*
Echo a string.

<n>USE-TACTIC *tac tac-use tac-mode*
Use a tactic on the current goal. The default tactic is given by the flag DEFAULT-TACTIC.

1.16. suggestions

<n>ADVICE Give some advice on how to proceed with the current proof.

<n>CHECK-STRUCTURE
Check various structural properties of the current proof. You will be informed about suspect constellations in the incomplete proof which may make it difficult for ETPS to provide advice or for you to finish the proof.

<n>GO Start producing and applying suggestions until no more are found. Suggestions are treated according to their priority and the state of the global parameter GO-INSTRUCTIONS.

<n>GO2 *tacmode* Apply all possible invertible tactics, until no more are possible. This is equivalent to typing USE-TACTIC GO2-TAC NAT-DED. The amount of output to the main window and the proofwindows is determined by the flag ETREE-NAT-VERBOSE.

<n>MONSTRO *tacmode*
This is equivalent to typing USE-TACTIC MONSTRO-TAC NAT-DED. It applies all the same tactics as GO2, and also ui-herbrand-tac. The amount of output to the main window and the proofwindows is determined by the flag ETREE-NAT-VERBOSE.

<n>SUGGEST *pline*
Suggest some applicable inference rule for proving a planned line.

1.17. Vpforms

<n>CLOSE-MATEVPW
Closes the window that displays the current vpform and substitution stack. Use ..../tps/utilities/vpshow (from a shell, not from TPS) to view the output file again.

<n>OPEN-MATEVPW *filename*
Open a window which will display the current vpform and substitution stack, if any. The window can be closed with the command CLOSE-MATEVPW. The size of the text is determined by the flag CHARSIZE, and the current width of the window by the flag VPW-WIDTH. The initial height of the window is determined by VPW-HEIGHT Use ..../tps/utilities/vpshow to view the file from the monitor level.

1.18. Rearranging the Proof

<n>DELETE *del-lines*
Delete lines from the proof outline.

<n>DELETE* *ranges*
Delete ranges of lines from the proof outline.

<n>INTRODUCE-GAP *line num*
Introduce a gap in an existing proof.

<n>LOCK-LINE *line*
Prevent a line from being deleted.

<n>MODIFY-GAPS *num1 num2*
Remove unnecessary gaps from the proof structure, and modify linenumbers so that the length of each gap is neither less than the first argument, nor greater than the second.

<n>MOVE *old-line new-line*

Renumber one particular line.

<n>MOVE* *range-to-move new-start*
Move all proof lines in given range to begin at new start number, but preserving the relative distances between the lines.

<n>PLAN *line* Change a justified line to a planned line.

<n>RENUMBERALL *num*
Renumber all the lines in the current proof.

<n>SQUEEZE Removes unnecessary gaps from the proof structure.

<n>UNLOCK-LINE *line*
The opposite of LOCK-LINE.

1.19. Status

<n>ARE-WE-USING *linelist*
Determines if given lines are being used to justify any other lines. Notice that the argument is a list of lines, not a range (i.e. 1 2 3 4 rather than 1--4).

<n>COUNT-LINES
Show the number of lines in the current proof.

<n>PSTATUS Give the current status information, i.e. planned lines and their supports. If work is being saved, issues an appropriate message.

<n>SPONSOR *pline linelist*
Add new sponsoring lines to the sponsors of a planned line.

<n>SUBPROOF *pline*
Concentrate on proving a particular planned line.

<n>UNSPONSOR *pline linelist*
Remove a list of unwanted sponsoring lines from among the sponsors of a planned line.

1.20. Miscellaneous Rules

<n>ASSERT *theorem line*
Use a theorem as a lemma in the current proof. If the line already exists, ETPS will check whether it is a legal instance of the theorem schema, otherwise it will prompt for the metavariables in the theorem schema (usually x or P, Q, ...).

<n>HYP *p2 h1 a b p2-hyps h1-hyps*
Introduce a new hypothesis.

<n>LEMMA *p2 p1 a b p2-hyps p1-hyps*
Introduce a Lemma.

<n>SAME *p2 d1 a p2-hyps d1-hyps*
Use the fact that two lines are identical to justify a planned line.

1.21. Propositional Rules

<n>ASSOC-LEFT *d1 d2 p assoc-l d1-hyps d2-hyps*
Rule to associate a support line leftwards. Use before calling CASES3 or CASES4.

<n>CASES *p6 d1 p5 h4 p3 h2 b a c p6-hyps d1-hyps p5-hyps h4-hyps p3-hyps h2-hyps*
Rule of Cases.

<n>CASES3 *p8 d1 p7 h6 p5 h4 p3 h2 c b a d p8-hyps d1-hyps p7-hyps h6-hyps p5-hyps h4-hyps p3-hyps h2-hyps*
Rule of Cases.

<n>CASES4 *p10 d1 p9 h8 p7 h6 p5 h4 p3 h2 d c b a e p10-hyps d1-hyps p9-hyps h8-hyps p7-hyps h6-hyps p5-hyps h4-hyps p3-hyps h2-hyps*
Rule of Cases.

<n>DEDUCT *p3 d2 h1 b a p3-hyps d2-hyps h1-hyps*
The deduction rule.

<n>DISJ-IMP *d1 d2 b a d1-hyps d2-hyps*
Rule to replace a disjunction by an implication.

<n>DISJ-IMP-L *d1 d2 b a d1-hyps d2-hyps*
Rule to replace a disjunction by an implication.

<n>DISJ-IMP-R *d1 d2 b a d1-hyps d2-hyps*
Rule to replace a disjunction by an implication.

<n>ECONJ *d1 d3 d2 b a d1-hyps d3-hyps d2-hyps*
Rule to infer two conjuncts from a conjunction.

<n>EQUIV-IMPLICS *d1 d2 r p d1-hyps d2-hyps*
Rule to convert an equivalence into twin implications.

<n>ICONJ *p3 p2 p1 b a p3-hyps p2-hyps p1-hyps*
Rule to infer a conjunction from two conjuncts.

<n>IDISJ-LEFT *p2 p1 b a p2-hyps p1-hyps*
Introduce a disjunction (left version).

<n>IDISJ-RIGHT *p2 p1 a b p2-hyps p1-hyps*
Introduce a disjunction (right version).

<n>IMP-DISJ *d1 d2 b a d1-hyps d2-hyps*
Rule to replace an implication by a disjunction.

<n>IMP-DISJ-L *d1 d2 b a d1-hyps d2-hyps*
Rule to replace an implication by a disjunction.

<n>IMP-DISJ-R *d1 d2 a b d1-hyps d2-hyps*
Rule to replace an implication by a disjunction.

<n>IMPLICS-EQUIV *p2 p1 r p p2-hyps p1-hyps*
Rule to convert twin implications into an equivalence.

<n>INDIRECT *p3 p2 h1 a p3-hyps p2-hyps h1-hyps*
Rule of Indirect Proof.

<n>INDIRECT1 *p3 p2 h1 b a p3-hyps p2-hyps h1-hyps*
Rule of Indirect Proof Using One Contradictory Line.

<n>INDIRECT2 *p4 p3 p2 h1 b a p4-hyps p3-hyps p2-hyps h1-hyps*
Rule of Indirect Proof Using Two Contradictory Lines.

<n>MP *d2 d3 p1 b a d2-hyps d3-hyps p1-hyps*
Modus Ponens.

<n>RULEC1 *p4 d1 d3 h2 b x a p4-hyps d1-hyps d3-hyps h2-hyps*
RuleC1 -- the special case of RULEC where the chosen variable has the same name as the bound variable.

<n>SUBST-EQUIV *d2 d3 p1 p r t s d2-hyps d3-hyps p1-hyps*
Substitution of Equivalence. Useable when R and P are the same modulo the equivalence s EQUIV t.

1.22. Negation Rules

<n>ABSURD *p2 p1 a p2-hyps p1-hyps*
Rule of Intuitionistic Absurdity.

<n>ENEG *p3 d1 p2 a p3-hyps d1-hyps p2-hyps*
Rule of Negation Elimination.

<n>INEG *p3 p2 h1 a p3-hyps p2-hyps h1-hyps*
Rule of Negation Introduction

<n>PULLNEG *p2 p1 a push-negation p2-hyps p1-hyps*
Pull out negation.

<n>PUSHNEG *d1 d2 a push-negation d1-hyps d2-hyps*
Push in negation.

1.23. Quantifier Rules

<n>AB* *d1 d2 b a d1-hyps d2-hyps*
Rule to alphabetically change embedded quantified variables.

<n>ABE *d1 d2 y a x s d1-hyps d2-hyps*
Rule to change a top level occurrence of an existentially quantified variable.

<n>ABU *p2 p1 y a x s p2-hyps p1-hyps*

Rule to change a top level occurrence of a universally quantified variable.

<n>EGEN *p2 p1 t a x lcontr p2-hyps p1-hyps*
Rule of Existential Generalization.

<n>RULEC *p4 d1 d3 h2 y b x a lcontr p4-hyps d1-hyps d3-hyps h2-hyps*
RuleC

<n>UGEN *p2 p1 a x p2-hyps p1-hyps*
Rule of Universal Generalization.

<n>UI *d1 d2 t a x lcontr d1-hyps d2-hyps*
Rule of Universal Instantiation.

1.24. Substitution Rules

<n>SUBSTITUTE *d1 d2 x t a s d1-hyps d2-hyps*
Rule to substitute a term for a variable.

1.25. Equality Rules

<n>EQUIV-EQ *d1 d2 b a d1-hyps d2-hyps*
Rule to infer a line from one which is equal up to definitions, lambda conversion, alphabetic change of bound variables and the Leibniz definition of the symbol = . You may use the editor command EXPAND= to create the desired line from the existing one.

<n>EQUIV-EQ-CONTR *p2 p1 a instantiate-top-equality p2-hyps p1-hyps*
Rule to contract the outermost instance of the Leibniz definition of equality into instances of the symbol = .

<n>EQUIV-EQ-CONTR* *p2 p1 a instantiate-equalities p2-hyps p1-hyps*
Rule to contract all instances of the Leibniz definition of equality into instances of the symbol = .

<n>EQUIV-EQ-EXPD *d1 d2 a instantiate-top-equality d1-hyps d2-hyps*
Rule to expand the outermost equality using the Leibniz definition.

<n>EQUIV-EQ-EXPD* *d1 d2 a instantiate-equalities d1-hyps d2-hyps*
Rule to expand all equalities using the Leibniz definition.

<n>EXT= *p2 p1 x g f p2-hyps p1-hyps*
Rule of Extensionality.

<n>EXT=0 *p2 p1 r p p2-hyps p1-hyps*
Rule to convert equality at type o into an equivalence.

<n>LET *p5 p4 h3 d2 d1 a x c p5-hyps p4-hyps h3-hyps d2-hyps d1-hyps*
Bind a variable to a term.

<n>SUBST= *d2 d3 p1 p r t s d2-hyps d3-hyps p1-hyps*
Substitution of Equality. Useable when R and P are the same modulo the equality s=t.

<n>SUBST=L *d2 d3 p1 p r t s d2-hyps d3-hyps p1-hyps*
Substitution of Equality. Replaces some occurrences of the left hand side by the right hand side.

<n>SUBST=R *d2 d3 p1 p r s t d2-hyps d3-hyps p1-hyps*
Substitution of Equality. Replaces some occurrences of the right hand side by the left hand side.

<n>SYM= *p2 p1 a b p2-hyps p1-hyps*
Rule of Symmetry of Equality.

1.26. Definition Rules

<n>EDEF *d1 d2 a inst-def d1-hyps d2-hyps*
Rule to eliminate first definition, left to right.

<n>EQUIV-WFFS *d1 d2 r p d1-hyps d2-hyps*
Rule to assert equivalence of lines up to definition.

<n>IDEF *p2 p1 a inst-def p2-hyps p1-hyps*
Rule to introduce a definition.

1.27. Lambda Conversion Rules

<n>BETA* *d1 d2 b a d1-hyps d2-hyps*
Rule to infer a line from one which is equal up to lambda conversion using beta rule (but NOT eta rule) and alphabetic change of bound variables.

<n>ETA* *d1 d2 b a d1-hyps d2-hyps*
Rule to infer a line from one which is equal up to lambda conversion using eta rule (but NOT beta rule) and alphabetic change of bound variables.

<n>LAMBDA* *d1 d2 b a d1-hyps d2-hyps*
Rule to infer a line from one which is equal up to lambda conversion using both beta and eta rules and alphabetic change of bound variables.

<n>LCONTR* *d1 d2 a lnorm d1-hyps d2-hyps*
Rule to put an inferred line into Lambda-normal form using both beta and eta conversion.

<n>LCONTR*-BETA *d1 d2 a lnorm-beta d1-hyps d2-hyps*
Rule to put an inferred line into beta-normal form.

<n>LCONTR*-ETA *d1 d2 a lnorm-eta d1-hyps d2-hyps*
Rule to put an inferred line into eta-normal form.

<n>LEXPD* *p2 p1 a lnorm p2-hyps p1-hyps*
Rule to put a planned line into Lambda-normal form using both beta and eta conversion.

<n>LEXPD*-BETA *p2 p1 a lnorm-beta p2-hyps p1-hyps*
Rule to put a planned line into beta-normal form.

<n>LEXPD*-ETA *p2 p1 a lnorm-eta p2-hyps p1-hyps*
Rule to put a planned line into eta-normal form.

1.28. Rewriting commands

<n>ACTIVATE-RULES *rlist*
Activate a list of rewrite rules. Activating a rule which is already active has no effect.

<n>DEACTIVATE-RULES *rlist*
Deactivate a list of rewrite rules. Deactivating a rule which is already inactive has no effect.

<n>DELETE-RRULE *rule*
Delete a rewrite rule from TPS.

<n>LIST-RRULES
Show all the current rewrite rules.

<n>MAKE-ABBREV-RRULE *name bidir*
Make a rewrite rule corresponding to a known abbreviation.

<n>MAKE-INVERSE-RRULE *rule newname*
Make the inverse rewrite rule of an existing rule.

<n>MAKE-THEORY *name extends axioms rrules other mhelp*
Create a new theory. A theory is defined by (optionally) starting from an old theory, and adding rewrite rules and axioms. You can also attach other library objects to the theory, which will then be loaded with it. This will also make an abbreviation of the same name. All of the objects in the theory should be defined in the library.

<n>PERMUTE-RRULES
Permute the list of rewrite rules.

<n>REWRITE-SUPP* *d1 d2 a apply-rrule-any* d1-hyps d2-hyps*
Rewrite a supporting line using all rewrite rules possible.

<n>REWRITE-SUPP1 *d1 d2 a apply-rrule-any d1-hyps d2-hyps*
Rewrite a supporting line using the first rewrite rule that applies.

<n>SIMPLIFY-PLAN *p2 p1 a simplify-up p2-hyps p1-hyps*
Justify a planned line using the first rewrite rule that applies.

<n>SIMPLIFY-PLAN* *p2 p1 a simplify-up* p2-hyps p1-hyps*
Justify a planned line using the first rewrite rule that applies.

<n>SIMPLIFY-SUPP *d1 d2 a simplify-down d1-hyps d2-hyps*
Rewrite a supporting line using the first rewrite rule that applies.

<n>SIMPLIFY-SUPP* *d1 d2 a simplify-down* d1-hyps d2-hyps*
Rewrite a supporting line using the first rewrite rule that applies.

<n>UNREWRITE-PLAN* *p2 p1 a unapply-rrule-any* p2-hyps p1-hyps*
Justify a planned line using all rewrite rules possible.

<n>UNREWRITE-PLAN1 *p2 p1 a unapply-rrule-any p2-hyps p1-hyps*
Justify a planned line using the first rewrite rule that applies.

<n>USE-RRULES *p2 p1 a b p2-hyps p1-hyps*
Rewrite a line. The line may be rewritten several steps, but rewrites may not be nested.

<n>USE-THEORY *theory*
Activate all the rewrite rules in a theory, and deactivate all other rewrite rules.

1.29. RuleP

<n>RULEP *conclusion antecedents*
Justify the CONSEQUENT line by RULEP using the lines in the list ANTECEDENTS.

1.30. Events

<n>DISABLE-EVENTS
Disable recording of TPS events. You will need to start a new session of TPS to enable recording of events after they have been disabled.

1.31. Statistics

<n>DATEREC *name type comment*
Records times used in the following processes: DIY, Mating Search, Merging Expansion Tree, Proof Transformation. All times recorded are in seconds. Internal-runtime includes GC-time. GC-time is garbage-collecting-time. I-GC-time is Internal-runtime minus GC-time. DATEREC also records the values of the flags listed in RECORDFLAGS, and will offer the user the chance to reset the provability status of a gwff in the library.

<n>DISPLAY-TIME *name*
Show time used in several processes: display-time diy: show the time used in DIY process display-time mating: show the time used in mating-search process display-time merge: show the time used in merging-expansion-tree process display-time eproof: show the time used in proof-transformation process display-time all: show all the times above All times are in seconds. Internal-runtime includes GC-time. GC-time is garbage-collecting-time. I-GC-time is Internal-runtime minus GC-time.

1.32. Maintenance

<n>CLOAD *file* Compile and load a file.

<n>CLOAD-MODULES *modules*
Compile and Load a list of modules.

<n>COMPILE-LIST *directory-list source-only*
Returns a list of files that need to be compiled.

<n>COMPL *filespeclist*
Compile 1 or more files.

<n>FILETYPE *filename*
Type a file on the screen. TPS will look for the file in a list of directories.

<n>LEDIT Call the resident Lisp editor (if there is one) inside TPS. It takes a filename as an optional argument. In most lisps, this will probably start up Emacs. In CMU lisp, this will start up Hemlock; use ^X^Z to leave Hemlock again. In some lisps, this command may not work at all.

<n>LOAD-SLOW *filename*
Step through loading a file.

<n>ORGANIZE Organizes the ENVIRONMENT help tree (e.g. after loading modules).

<n>QLOAD *filespec*
Load the most recent compiled or uncompiled file from your default directory, home directory, or source path. In general, the following rules are used to determine whether compiled or uncompiled file should be load in: (1) If the file name with extension '.lisp', always load the uncompiled source code. (2) If the file name without extension, then (2.1) if both compiled and uncompiled file exist, and (2.1.1) the compiled one is newer, it is loaded

in. (2.1.2) the uncompiled one is newer, (2.1.2.1) if the flag 'expertflag' is NIL, always load the uncompiled source code. (2.1.2.2) if the flag 'expertflag' is T, ask user whether load the uncompiled one, or compile it and load the compiled one then. (2.2) if only the compiled one exists, load it in. (2.3) if only the uncompiled one exists, do the same as case (2.1.2)

<n>SYS-LOAD *modulelist*
Load all the modules in the given list, whether they are loaded already or not.

<n>TLIST *symbol* Use a help function to display all of the property list of a symbol.

<n>TLOAD *filespec*
Load the most recent compiled or uncompiled file from your default directory, home directory, or source-path. In general, the following rules are used to determine whether compiled or uncompiled file should be load in: (1) If both compiled and uncompiled file exist, and (1.1) the compiled one is newer, it is loaded in. (1.2) the uncompiled one is newer, then (1.2.1) if the global varibale core::*allow-compile-source* is T, the name of the file contains extension

<n>TPS-TEST *mate-only record quiet-run expu modify output timing testwin*
Attempt to prove a list of theorems; used for testing new versions of TPS.

The list of theorems, with the modes to be used, is stored as (theorem . mode) pairs in the flag TEST-THEOREMS. These theorems and modes will be fetched from the library, if they cannot be found in TPS and if you have a library. You should set DEFAULT-LIB-DIR and BACKUP-LIB-DIR appropriately. You can only do DATEREC after each theorem if you have the library facilities loaded. Quiet running uses the mode QUIET to switch off as much screen output as possible. You can EXPUNGE between proofs (this will reduce the amount of memory required, but will mean that other expansion proofs in the memory may be lost; it will also re-assert your default flag values between each proof).

The output file is kept independently of DATEREC records, and consists of single lines, one per theorem, either stating that the theorem was proved at a certain time using a certain mode, or that the proof terminated with proof lines still remaining (i.e. you've got a bug somewhere) or that tps-test ended abnormally (i.e. either a serious bug or you pressed Ctrl-C). Timing information can also be sent to the short file if necessary.

If the short file already exists, the old copy will be renamed by adding .bak to its name.

<n>TPS-TEST2 *searchlist quiet-run expu output testwin*
Like TPS-TEST (see the help message for that command), but calls the TEST top level and attempts to prove one theorem repeatedly with several different values of some crucial flags, to see how the time taken will vary.

TEST-THEOREMS should contain a list of dotted pairs of theorems and modes in which they can be proven; the searchlist which is used should have at least one setting in which the theorem can be proven (otherwise tps-test2 will never finish that theorem).

The output file (by default, tps-test2-output.doc) will contain a summary of the results. If this file already exists, it will be renamed by adding .bak to its name.

<n>TPS3-SAVE Save the current TPS3 as the new TPS3 core image.

1.33. Modules

<n>LOADED-MODS
Returns list of loaded modules.

<n>MODULES *modulelist*
Load the specified modules.

<n>UNLOADED-MODS
Returns list of unloaded modules.

1.34. Rules Module

<n>ASSEMBLE-FILE *rule-file part-of*
Parse, build and write every rule in a given rule file. Be sure to set the correct mode (MODE RULES) before using this command.

<n>ASSEMBLE-MOD *module*
Produce a file with rule commands for every rule file in a module.

<n>BUILD *rule* Process a rule without writing the resulting code to a file.

<n>WRITE-RULE *rule filename*
Write the various functions and definitions for a rule into a file.

1.35. Lisp packages

<n>PACK-STAT Give information about the current status of the Lisp package structure.

<n>UNUSE *lisp-package*
Make a Lisp package inaccessible.

<n>USE *lisp-package*
Make a Lisp package accessible in the current Lisp package. An error will be issued by Lisp if this leads to name conflicts.

1.36. Display

<n>DISPLAYFILE *filename bigwin*
Open a (big) window in which the contents of the given file will be displayed. Once the end of the file is reached, a message will be printed and some additional blank lines will be added. Once the end of the blank lines is reached, the window will vanish.

1.37. Bugs

<n>BUG-DELETE *name*
Delete a bug record. Exactly the same as the library DELETE command, but will use the DEFAULT-BUG-DIR if USE-DEFAULT-BUG-DIR is T.

<n>BUG-HELP *name*
Show the help message of a bug record.

<n>BUG-LIST Show all the saved bugs in the appropriate directory. See USE-DEFAULT-BUG-DIR.

<n>BUG-RESTORE *name*
Restore a bug from the library (see USE-DEFAULT-BUG-DIR). This must have been a bug which was saved with BUG-SAVE; this command will reload all the necessary library objects, reset all the flags and reload the proof. This does NOT create a new mode; it just resets the flags.

<n>BUG-SAVE *name comment*
Records details of a bug. Saves the current flag settings, the output of the HISTORY command, all currently loaded library objects, the current proof, the date and time and any comments (the best idea is to copy any error messages in to the "comments" prompt). This setup can then be retrieved with BUG-RESTORE. The details are saved as a MODE1, under the name that the user provides (in a file of the same name) with the assertion and library objects in other-attributes and other-remarks respectively, and the context set to BUG. The file will be saved in an appropriate directory (see USE-DEFAULT-BUG-DIR).

2. Inference Rules

The internal name of this category is SRULE. An inference rule can be defined using DEFSRULE. Allowable properties are: MATCHFN, MATCH1FN, SHORTFN, PRIORITY.

2.1. Miscellaneous Rules

HYP Introduce a new hypothesis.

```
 (H1)  H1     ├ A_o                                                    Hyp
*(P2)  H      ├ B_o
Transformation: (P2 ss) ==> (P2 H1 ss)
```

LEMMA Introduce a Lemma.

```
 (P1)  H1     ├ A_o
*(P2)  H2     ├ B_o
Transformation: (P2 ss) ==> (P2 P1 ss) (P1 ss)
```

SAME Use the fact that two lines are identical to justify a planned line.

```
*(D1)  H      ├ A_o
*(P2)  H      ├ A_o                                          Same as: D1
Transformation: (P2 D1 ss) ==>
```

2.2. Propositional Rules

ASSOC-LEFT Rule to associate a support line leftwards. Use before calling CASES3 or CASES4.

```
*(D1)  H      ├ P_o
 (D2)  H      ├ '(ASSOC-L  P_o)                                 Assoc: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

CASES Rule of Cases.

```
*(D1)  H      ├ A_o ∨ B_o
 (H2)  H,H2   ├ A_o                                          Case 1: D1
 (P3)  H,H2   ├ C_o
 (H4)  H,H4   ├ B_o                                          Case 2: D1
 (P5)  H,H4   ├ C_o
*(P6)  H      ├ C_o                                    Cases: D1 P3 P5
Transformation: (P6 D1 ss) ==> (P3 H2 ss) (P5 H4 ss)
```

CASES3 Rule of Cases.

```
*(D1)  H      ├ A_o ∨ B_o ∨ C_o
 (H2)  H,H2   ├ A_o                                          Case 1: D1
 (P3)  H,H2   ├ D_o
 (H4)  H,H4   ├ B_o                                          Case 2: D1
 (P5)  H,H4   ├ D_o
 (H6)  H,H6   ├ C_o                                          Case 3: D1
 (P7)  H,H6   ├ D_o
*(P8)  H      ├ D_o                                Cases: D1 P3 P5 P7
Transformation: (P8 D1 ss) ==> (P3 H2 ss) (P5 H4 ss) (P7 H6 ss)
```

CASES4 Rule of Cases.

```
*(D1)  H      ├ A_o ∨ B_o ∨ C_o ∨ D_o
 (H2)  H,H2   ├ A_o                                          Case 1: D1
 (P3)  H,H2   ├ E_o
 (H4)  H,H4   ├ B_o                                          Case 2: D1
 (P5)  H,H4   ├ E_o
 (H6)  H,H6   ├ C_o                                          Case 3: D1
 (P7)  H,H6   ├ E_o
 (H8)  H,H8   ├ D_o                                          Case 4: D1
 (P9)  H,H8   ├ E_o
*(P10) H      ├ E_o                                Cases: D1 P3 P5 P7 P9
Transformation: (P10 D1 ss) ==> (P3 H2 ss) (P5 H4 ss) (P7 H6 ss) (P9 H8 ss)
```

DEDUCT The deduction rule.

```
 (H1)  H,H1   ├ A_o                                                Hyp
 (D2)  H,H1   ├ B_o
*(P3)  H      ├ A_o ⊃ B_o                                  Deduct: D2
Transformation: (P3 ss) ==> (D2 H1 ss)
```

DISJ-IMP Rule to replace a disjunction by an implication.

```
*(D1)  H      ├ ~A_o ∨ B_o
 (D2)  H      ├ A_o ⊃ B_o                              Disj-Imp: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

DISJ-IMP-L Rule to replace a disjunction by an implication.

```
*(D1)  H      ├ A_o ∨ B_o
 (D2)  H      ├ ~A_o ⊃ B_o                           Disj-Imp-L: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

DISJ-IMP-R Rule to replace a disjunction by an implication.

```
*(D1)  H      ├ A_o ∨ B_o
 (D2)  H      ├ ~B_o ⊃ A_o                           Disj-Imp-R: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

ECONJ Rule to infer two conjuncts from a conjunction.

```
*(D1)  H      ├ A_o ∧ B_o
 (D2)  H      ├ A_o                                        Conj: D1
 (D3)  H      ├ B_o                                        Conj: D1
Transformation: (pp D1 ss) ==> (pp D2 D3 ss)
```

EQUIV-IMPLICS Rule to convert an equivalence into twin implications.

```
*(D1)  H      ├ P_o ≡ R_o
 (D2)  H      ├ [P_o ⊃ R_o] ∧.R ⊃ P                    EquivImp: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

ICONJ Rule to infer a conjunction from two conjuncts.

```
 (P1)  H      ├ A_o
 (P2)  H      ├ B_o
*(P3)  H      ├ A_o ∧ B_o                               Conj: P1 P2
Transformation: (P3 ss) ==> (P1 ss) (P2 ss)
```

IDISJ-LEFT Introduce a disjunction (left version).

```
 (P1)  H      ⊢ A_o
*(P2)  H      ⊢ A_o ∨ B_o                                   Idisj-L: P1
Transformation: (P2 ss) ==> (P1 ss)
```

IDISJ-RIGHT Introduce a disjunction (right version).

```
 (P1)  H      ⊢ A_o
*(P2)  H      ⊢ B_o ∨ A_o                                   Idisj-R: P1
Transformation: (P2 ss) ==> (P1 ss)
```

IMP-DISJ Rule to replace an implication by a disjunction.

```
*(D1)  H      ⊢ A_o ⊃ B_o
 (D2)  H      ⊢ ~A_o ∨ B_o                                  Imp-Disj: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

IMP-DISJ-L Rule to replace an implication by a disjunction.

```
*(D1)  H      ⊢ ~A_o ⊃ B_o
 (D2)  H      ⊢ A_o ∨ B_o                                 Imp-Disj-L: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

IMP-DISJ-R Rule to replace an implication by a disjunction.

```
*(D1)  H      ⊢ ~B_o ⊃ A_o
 (D2)  H      ⊢ A_o ∨ B_o                                 Imp-Disj-R: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

IMPLICS-EQUIV Rule to convert twin implications into an equivalence.

```
 (P1)  H      ⊢ [P_o ⊃ R_o] ∧.R ⊃ P
*(P2)  H      ⊢ P_o ≡ R_o                                   ImpEquiv: P1
Transformation: (P2 ss) ==> (P1 ss)
```

INDIRECT Rule of Indirect Proof.

```
 (H1)  H,H1   ⊢ ~A_o                                 Assume negation
 (P2)  H,H1   ⊢ ⊥
*(P3)  H      ⊢ A_o                                      Indirect: P2
Transformation: (P3 ss) ==> (P2 H1 ss)
```

INDIRECT1 Rule of Indirect Proof Using One Contradictory Line.

```
 (H1)  H,H1   ⊢ ~A_o                                 Assume negation
 (P2)  H,H1   ⊢ B_o ∧ ~B
*(P3)  H      ⊢ A_o                                      Indirect: P2
Transformation: (P3 ss) ==> (P2 H1 ss)
```

INDIRECT2 Rule of Indirect Proof Using Two Contradictory Lines.

```
 (H1)  H,H1   ⊢ ~A_o                                 Assume negation
 (P2)  H,H1   ⊢ B_o
 (P3)  H,H1   ⊢ ~B_o
*(P4)  H      ⊢ A_o                                 Indirect: P2 P3
Transformation: (P4 ss) ==> (P2 H1 ss) (P3 H1 ss)
```

MP Modus Ponens.

```
 (P1)  H      ⊢ A_o
*(D2)  H      ⊢ A_o ⊃ B_o
 (D3)  H      ⊢ B_o                                        MP: P1 D2
Transformation: (pp D2 ss) ==> (P1 ss) (pp D3 ss P1)
```

RULEC1 RuleC1 -- the special case of RULEC where the chosen variable has the same name as the bound variable.

```
*(D1)  H      ├ ∃x_α B_o
 (H2)  H,H2   ├ B_o                                          Choose: x_α D1
 (D3)  H,H2   ├ A_o
*(P4)  H      ├ A_o                                          RuleC: D1 D3
Restrictions:  (NOT-FREE-IN-HYPS x_α) (NOT-FREE-IN x_α A_o)
Transformation: (P4 D1 ss) ==> (D3 H2 ss)
```

SUBST-EQUIV Substitution of Equivalence. Useable when R and P are the same modulo the equivalence s EQUIV t.

```
 (P1)  H      ├ P_o
*(D2)  H      ├ s_o ≡ t_o
 (D3)  H      ├ R_o                                  Sub-equiv: P1 D2
Restrictions:  (SAME-MODULO-EQUALITY P_o R_o s_o t_o)
Transformation: (pp D2 ss) ==> (P1 ss) (pp D3 ss P1 D2)
```

2.3. Negation Rules

ABSURD Rule of Intuitionistic Absurdity.

```
 (P1)  H      ├ ⊥
*(P2)  H      ├ A_o                                          Absurd: P1
Transformation: (P2 ss) ==> (P1 ss)
```

ENEG Rule of Negation Elimination.

```
*(D1)  H      ├ ~A_o
 (P2)  H      ├ A_o
*(P3)  H      ├ ⊥                                    NegElim: D1 P2
Transformation: (P3 D1 ss) ==> (P2 ss)
```

INEG Rule of Negation Introduction

```
 (H1)  H,H1   ├ A_o                                             Hyp
 (P2)  H,H1   ├ ⊥
*(P3)  H      ├ ~A_o                                 NegIntro: P2
Transformation: (P3 ss) ==> (P2 H1 ss)
```

PULLNEG Pull out negation.

```
 (P1)  H      ├ '(PUSH-NEGATION  [~A_o])
*(P2)  H      ├ ~A_o                                         Neg: P1
Restrictions:  (NON-ATOMIC A_o)
Transformation: (P2 ss) ==> (P1 ss)
```

PUSHNEG Push in negation.

```
*(D1)  H      ├ ~A_o
 (D2)  H      ├ '(PUSH-NEGATION  [~A_o])                     Neg: D1
Restrictions:  (NON-ATOMIC-OR-TRUTHVALUE A_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

2.4. Quantifier Rules

AB* Rule to alphabetically change embedded quantified variables.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ B_o                                                    AB: D1
Restrictions:  (WFFEQ-AB A_o B_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

ABE Rule to change a top level occurrence of an existentially quantified variable.

```
*(D1)  H      ⊢ ∃x_α A_o
 (D2)  H      ⊢ ∃y_α '(S  y  x_α  A_o)                                 AB: y_α D1
Restrictions:  (FREE-FOR y_α x_α A_o) (NOT-FREE-IN y_α A_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

ABU Rule to change a top level occurrence of a universally quantified variable.

```
 (P1)  H      ⊢ ∀y_α '(S  y  x_α  A_o)
*(P2)  H      ⊢ ∀x_α A_o                                               AB: x_α P1
Restrictions:  (FREE-FOR y_α x_α A_o) (NOT-FREE-IN y_α A_o)
Transformation: (P2 ss) ==> (P1 ss)
```

EGEN Rule of Existential Generalization.

```
 (P1)  H      ⊢ '(LCONTR  [[λx_α A_o] t_α])
*(P2)  H      ⊢ ∃x_α A_o                                             EGen: t_α P1
Transformation: (P2 ss) ==> (P1 ss)
```

RULEC RuleC

```
*(D1)  H      ⊢ ∃x_α B_o
 (H2)  H,H2   ⊢ '(LCONTR  [[λx_α B_o] y_α])                  Choose: y_α D1
 (D3)  H,H2   ⊢ A_o
*(P4)  H      ⊢ A_o                                            RuleC: D1 D3
Restrictions:  (IS-VARIABLE y_α) (NOT-FREE-IN-HYPS y_α) (NOT-FREE-IN y_α [∃x_α
B_o]) (NOT-FREE-IN y_α A_o)
Transformation: (P4 D1 ss) ==> (D3 H2 ss)
```

UGEN Rule of Universal Generalization.

```
 (P1)  H      ⊢ A_o
*(P2)  H      ⊢ ∀x_α A_o                                             UGen: x_α P1
Restrictions:  (NOT-FREE-IN-HYPS x_α)
Transformation: (P2 ss) ==> (P1 ss)
```

UI Rule of Universal Instantiation.

```
*(D1)  H      ⊢ ∀x_α A_o
 (D2)  H      ⊢ '(LCONTR  [[λx_α A_o] t_α])                            UI: t_α D1
Transformation: (pp D1 ss) ==> (pp D2 D1 ss)
```

2.5. Substitution Rules

SUBSTITUTE Rule to substitute a term for a variable.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(S  t_α  x_α  A_o)                        Subst: t_α  x_α D1
Restrictions:  (NOT-FREE-IN-HYPS x_α) (FREE-FOR t_α x_α A_o)
Transformation: (pp D1 ss) ==> (pp D2 ss D1)
```

2.6. Equality Rules

EQUIV-EQ Rule to infer a line from one which is equal up to definitions, lambda conversion, alphabetic change of bound variables and the Leibniz definition of the symbol = . You may use the editor command EXPAND= to create the desired line from the existing one.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ B_o                                          Equiv-eq: D1
Restrictions:  (WFFEQ-DEFEQ A_o B_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

EQUIV-EQ-CONTR

Rule to contract the outermost instance of the Leibniz definition of equality into instances of the symbol = .

```
 (P1)  H      ⊢ '(INSTANTIATE-TOP-EQUALITY  A_o)
*(P2)  H      ⊢ A_o                                          Equiv-eq: P1
Transformation: (P2 ss) ==> (P1 ss)
```

EQUIV-EQ-CONTR*

Rule to contract all instances of the Leibniz definition of equality into instances of the symbol = .

```
 (P1)  H      ⊢ '(INSTANTIATE-EQUALITIES  A_o)
*(P2)  H      ⊢ A_o                                          Equiv-eq: P1
Transformation: (P2 ss) ==> (P1 ss)
```

EQUIV-EQ-EXPD Rule to expand the outermost equality using the Leibniz definition.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(INSTANTIATE-TOP-EQUALITY  A_o)             Equiv-eq: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

EQUIV-EQ-EXPD*

Rule to expand all equalities using the Leibniz definition.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(INSTANTIATE-EQUALITIES  A_o)               Equiv-eq: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

EXT= Rule of Extensionality.

```
 (P1)  H      ⊢ ∀x_β.f_αβ x = g_αβ x
*(P2)  H      ⊢ f_αβ = g_αβ                                      Ext=: P1
Transformation: (P2 ss) ==> (P1 ss)
```

EXT=0 Rule to convert equality at type o into an equivalence.

```
 (P1)  H      ⊢ P_o ≡ R_o
*(P2)  H      ⊢ P_o = R_o                                        Ext=: P1
Transformation: (P2 ss) ==> (P1 ss)
```

LET Bind a variable to a term.

```
 (D1)  H      ⊢ A_α = A                                          Refl=
 (D2)  H      ⊢ ∃x_α.x = A_α                                EGen: x_α D1
 (H3)  H,H3   ⊢ x_α = A_α                                    Choose: x_α
 (P4)  H,H3   ⊢ C_o
*(P5)  H      ⊢ C_o                                          RuleC: D2 P4
Restrictions:  (NOT-FREE-IN-HYPS x_α) (NOT-FREE-IN x_α C_o)
Transformation: (P5 ss) ==> (P4 ss D1 D2 H3)
```

SUBST= Substitution of Equality. Useable when R and P are the same modulo the equality s=t.

```
 (P1)  H      ⊢ P_o
*(D2)  H      ⊢ s_α = t_α
 (D3)  H      ⊢ R_o                                              Sub=: P1 D2
Restrictions:  (SAME-MODULO-EQUALITY P_o R_o s_α t_α)
Transformation: (pp D2 ss) ==> (P1 ss) (pp D3 ss P1 D2)
```

SUBST=L Substitution of Equality. Replaces some occurrences of the left hand side by the right hand side.

```
 (P1)  H      ⊢ P_o
*(D2)  H      ⊢ s_α = t_α
 (D3)  H      ⊢ R_o                                            Subst=: P1 D2
Restrictions:  (R-PRIME-RESTR s_α P_o t_α R_o)
Transformation: (pp D2 ss) ==> (P1 ss) (pp D3 ss P1 D2)
```

SUBST=R Substitution of Equality. Replaces some occurrences of the right hand side by the left hand side.

```
 (P1)  H      ⊢ P_o
*(D2)  H      ⊢ t_α = s_α
 (D3)  H      ⊢ R_o                                            Subst=: P1 D2
Restrictions:  (R-PRIME-RESTR s_α P_o t_α R_o)
Transformation: (pp D2 ss) ==> (P1 ss) (pp D3 ss P1 D2)
```

SYM= Rule of Symmetry of Equality.

```
 (P1)  H      ⊢ A_α = B_α
*(P2)  H      ⊢ B_α = A_α                                        Sym=: P1
Transformation: (P2 ss) ==> (P1 ss)
```

2.7. Definition Rules

EDEF Rule to eliminate first definition, left to right.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(INST-DEF  A_o)                                 Defn: D1
Restrictions:  (CONTAINS-DEFN A_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

EQUIV-WFFS Rule to assert equivalence of lines up to definition.

```
*(D1)  H      ⊢ P_o
 (D2)  H      ⊢ R_o                                          EquivWffs: D1
Restrictions:  (WFFEQ-DEF P_o R_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

IDEF Rule to introduce a definition.

```
 (P1)  H      ⊢ '(INST-DEF  A_o)
*(P2)  H      ⊢ A_o                                              Defn: P1
Restrictions:  (CONTAINS-DEFN A_o)
Transformation: (P2 ss) ==> (P1 ss)
```

2.8. Lambda Conversion Rules

BETA* Rule to infer a line from one which is equal up to lambda conversion using beta rule (but NOT eta rule) and alphabetic change of bound variables.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ B_o                                          Beta Rule: D1
Restrictions:  (WFFEQ-AB-BETA A_o B_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

ETA* Rule to infer a line from one which is equal up to lambda conversion using eta rule (but NOT beta rule) and alphabetic change of bound variables.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ B_o                                           Eta Rule: D1
Restrictions:  (WFFEQ-AB-ETA A_o B_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

LAMBDA* Rule to infer a line from one which is equal up to lambda conversion using both beta and eta rules and alphabetic change of bound variables.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ B_o                                          Lambda=: D1
Restrictions:  (WFFEQ-AB-LAMBDA A_o B_o)
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

LCONTR* Rule to put an inferred line into Lambda-normal form using both beta and eta conversion.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(LNORM  A_o)                                 Lambda: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

LCONTR*-BETA Rule to put an inferred line into beta-normal form.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(LNORM-BETA  A_o)                           Beta rule: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

LCONTR*-ETA Rule to put an inferred line into eta-normal form.

```
*(D1)  H      ⊢ A_o
 (D2)  H      ⊢ '(LNORM-ETA  A_o)                             Eta rule: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

LEXPD* Rule to put a planned line into Lambda-normal form using both beta and eta conversion.

```
 (P1)  H      ⊢ '(LNORM  A_o)
*(P2)  H      ⊢ A_o                                           Lambda: P1
Transformation: (P2 ss) ==> (P1 ss)
```

LEXPD*-BETA Rule to put a planned line into beta-normal form.

```
 (P1)  H      ⊢ '(LNORM-BETA  A_o)
*(P2)  H      ⊢ A_o                                          Beta rule: P1
Transformation: (P2 ss) ==> (P1 ss)
```

LEXPD*-ETA Rule to put a planned line into eta-normal form.

```
 (P1)  H      ⊢ '(LNORM-ETA  A_o)
*(P2)  H      ⊢ A_o                                           Eta rule: P1
Transformation: (P2 ss) ==> (P1 ss)
```

2.9. Rewriting commands

REWRITE-SUPP* Rewrite a supporting line using all rewrite rules possible.

```
*(D1)  H      ├ A_o
 (D2)  H      ├ '(APPLY-RRULE-ANY*  A_o)                    Rewrites: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

REWRITE-SUPP1 Rewrite a supporting line using the first rewrite rule that applies.

```
*(D1)  H      ├ A_o
 (D2)  H      ├ '(APPLY-RRULE-ANY  A_o)                      Rewrite: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

SIMPLIFY-PLAN Justify a planned line using the first rewrite rule that applies.

```
 (P1)  H      ├ '(SIMPLIFY-UP  A_o)
*(P2)  H      ├ A_o                                          Rewrite: P1
Transformation: (P2 ss) ==> (P1 ss)
```

SIMPLIFY-PLAN*

Justify a planned line using the first rewrite rule that applies.

```
 (P1)  H      ├ '(SIMPLIFY-UP*  A_o)
*(P2)  H      ├ A_o                                          Rewrite: P1
Transformation: (P2 ss) ==> (P1 ss)
```

SIMPLIFY-SUPP Rewrite a supporting line using the first rewrite rule that applies.

```
*(D1)  H      ├ A_o
 (D2)  H      ├ '(SIMPLIFY-DOWN  A_o)                        Rewrite: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

SIMPLIFY-SUPP*

Rewrite a supporting line using the first rewrite rule that applies.

```
*(D1)  H      ├ A_o
 (D2)  H      ├ '(SIMPLIFY-DOWN*  A_o)                       Rewrite: D1
Transformation: (pp D1 ss) ==> (pp D2 ss)
```

UNREWRITE-PLAN*

Justify a planned line using all rewrite rules possible.

```
 (P1)  H      ├ '(UNAPPLY-RRULE-ANY*  A_o)
*(P2)  H      ├ A_o                                        Rewrites: P1
Transformation: (P2 ss) ==> (P1 ss)
```

UNREWRITE-PLAN1

Justify a planned line using the first rewrite rule that applies.

```
 (P1)  H      ├ '(UNAPPLY-RRULE-ANY  A_o)
*(P2)  H      ├ A_o                                          Rewrite: P1
Transformation: (P2 ss) ==> (P1 ss)
```

USE-RRULES Rewrite a line. The line may be rewritten several steps, but rewrites may not be nested.

```
 (P1)  H      ├ A_o
*(P2)  H      ├ B_o                                          Rewrite: P1
Restrictions:  (INSTANCE-OF-REWRITING A_o B_o)
Transformation: (P2 ss) ==> (P1 ss)
```

3. Mating-Search Commands

The internal name of this category is MATEOP. A mating-search command can be defined using DEFMATEOP. Allowable properties are: `MATE-ALIAS`, `MATE-RESULT->`, `MATEWFF-ARGNAME`, `MATE-DEFAULTFNS`, `MATE-APPLICABLE-P`, `MATE-MOVE-FN`, `MHELP`.

3.1. Top Levels

LEAVE — Exit mating-search. If the current expansion proof is complete, the user will be prompted as to whether to apply MERGE-TREE before exiting.

3.2. Printing

ETD — Etree Display: print an expansion tree into list form, printing shallow formulas for leaf nodes only. The format used is NODE [selection and expansion terms] ; CHILDREN or SHALLOW FORMULA

ETP — Etree Print: print an expansion tree into list form, printing shallow formulas for all nodes. The format used is NODE [selection and expansion terms] ; CHILDREN ; SHALLOW FORMULA

P — Print the current node

PDEEP — Print the deep formula of an expansion tree.

PP — Print an expansion tree with node-names.

PPDEEP — Pretty-print the deep formula of an expansion tree.

PPF — Print the current proof.

PPNODE — Print an expansion tree with node-names.

PSH — Print the shallow formula of an expansion tree.

PTREE *gwff* — Print out the etree below the current topnode, showing expansion variables, skolem terms, selection terms, and rewrite justifications. Branches with *'s denote nodes that are being omitted for lack of space. The cure for this is to either start printing from a node lower in the tree, or make the screen wider, or use SHOWNOTYPES. See also PTREE*

PTREE* *gwff* — Print out the etree below the current topnode, showing expansion variables, skolem terms, selection terms, and rewrite justifications. For all other nodes, show the shallow formula at that node. Branches with *'s denote nodes that are being omitted for lack of space. The cure for this is to either start printing from a node lower in the tree, or make the screen wider, or use SHOWNOTYPES. See also PTREE

PTREE-FILE *file width fmlas*
As for PTREE or PTREE*, but send the output to a file. For a width of 200 characters, you can print the results using some variant of the following: "enscript -r -fCourier-Bold6 -dberyl <filename> "

SHOW-OPTION-TREE
Show the current option-tree.

3.3. Recording

O — Invert PRINTMATEFLAG, that is switch automatic recording of mating-search into a file either on or off. This has not actually been implemented!

REM *rm* — Write a remark into the PRINTMATEFILE.

3.4. Expansion Trees

DP — Deepen a single leaf of an expansion tree.

DP* — Iteratively deepen an expansion tree until all leaves are literals.

DP= — Deepen top level equality in the etree.

DPTREE — Deepen every leaf node of an expansion tree.

DUP-ALL — Duplicate all variables in an expansion tree.

DUP-OUTER — Duplicate all outermost variables in an expansion tree.

DUP-VAR — Duplicate a variable at an expansion node.

EXP *term* EXPAND a given universal or existential quantifier.

MOD-STATUS *status*
Set the status of the current-topnode to the specified value. If the status of a node is not positive, it is ignored during mating search.

NAME-PRIM Lists all possible primitive substitutions for the current shallow formula. See the flags PRIM-BDTYPES, MIN-PRIM-DEPTH, MAX-PRIM-DEPTH and PRIM-QUANTIFIER for information on how to change which substitutions are generated.

PRIM-ALL Apply primitive substitutions at all outermost expansion nodes.

PRIM-OUTER Apply primitive substitutions at all outer expansion nodes.

PRIM-SINGLE *subst var*
Applies a single primsub. These can be generated by using the NAME-PRIM command. The command PRIM-SINGLE destructively alters the etree and creates a new jform, and is basically equivalent to SUB-ETREE followed by DP* and CJFORM. The variable must be specified in full detail, with both superscript and type, as in the vpform (e.g. "r^1(ob(ob))").

PRIM-SUB Apply primitive substitutions at an expansion node.

SEL SELECT for a given universal or existential quantifier.

SUB *gwff skolemize deepen*
Create an expansion tree from a gwff0.

SUB-ETREE *term var*
Substitute a term for a variable throughout an expansion tree. Destructively alters the expansion tree.

TERMS Get the expansion terms of an expansion node or the selected variable of a selection node.

3.5. Mating search

GO Start mating search using default mating search (controlled by flag DEFAULT-MS).

NOOP Do nothing. (TPS uses this internally.)

UNIFY Call unification in interactive mode for active mating. The unification tree associated with the active-mating is passed on to the unification top-level. Any changes made to this tree are destructive. Applicable only for a higher-order unification problem. Uses MS88-style unification.

3.6. MS88 search procedure

ADD-CONN *first second*
Add a connection to the current mating. TPS will not allow you to add a connection to a mating if adding it causes the resulting mating to be non unifiable. No check is made to determine if the connection spans an open path.

ADD-CONN* Repeatedly call ADD-CONN.

APPLY-SUBSTS Apply substitutions found during mating search to JFORM. Applicable only if mating is complete.

COMPLETE-P Test whether current mating is complete. Will return a path that is not spanned by the mating otherwise.

INIT-MATING No further help available. Sorry.

MS88 Call mating search procedure on the current eproof. This procedure uses a naive level-saturation method, exhaustively searching a single jform before applying any duplications. Quantifier duplications are applied uniformly to outermost quantifiers. Will try primitive substitution for outermost variable only. Works on only a single jform at a time.

MS88-SUB *etree* Call MS88 on a partial expansion tree (subtree).

REM-CONN *first second*
Remove a connection from the current mating.

REM-CONN* Repeatedly call REM-CONN.

REM-LAST-CONN
Remove the last connection to the current mating.

SHOW-MATING Show the connections in the current mating.

SHOW-SUBSTS Show the substitutions suggested by mating search for the complete active mating.

3.7. MS89 search procedure

MS89 Begin mating search MS89 on the current expansion proof. Primitive substitutions and duplications are performed systematically, with multiple jforms being worked on simultaneously. On each particular jform, the search procedure MS88 is used. The flags MAX-SEARCH-LIMIT, SEARCH-TIME-LIMIT, and RANK-EPROOF-FN are used to control the search. See also the command SHOW-OPTION-TREE.

3.8. MS90-3 search procedure

EXPAND-ETREE Convert the jform proof found by path-focused duplication procedures MS90-3 and MS90-9 into an expansion proof.

MS90-3 Start mating search procedure MS90-3 on current eproof. This search procedure incorporates Issar's path-focused duplication, but works on just one jform at a time. Only duplications are done, not primitive substitutions. This is not an interactive procedure.

PROP-MSEARCH Start Sunil's propositional mating search procedure. This search procedure only works on propositional jforms.

3.9. MS90-9 search procedure

MS90-9 Begin mating search MS90-9 on the current expansion proof. Primitive substitutions and duplications are performed systematically, with multiple jforms being worked on simultaneously. On each particular jform, the search procedure MS90-3 is used. The flags MAX-SEARCH-LIMIT, SEARCH-TIME-LIMIT, and RANK-EPROOF-FN are used to control the search. See also the command SHOW-OPTION-TREE.

3.10. MS91-6 and MS91-7 search procedures

MS91-6 Begin mating search MS91-6 on the current expansion proof. Primitive substitutions and duplications are performed systematically, with multiple jforms being worked on simultaneously. On each particular jform, the search procedure MS88 is used. The flags MAX-SEARCH-LIMIT and SEARCH-TIME-LIMIT are used to control the amount of time spent on each jform.

The order in which the possible jforms are considered depends on a number of flags. Firstly, the primitive substitutions which are generated are determined by the values of MAX-PRIM-DEPTH, MIN-PRIM-DEPTH, PRIM-QUANTIFIER and NEG-PRIM-SUB. If DUP-ALLOWED is T, then additional options are generated corresponding to duplicated quantifiers. These options are then combined into sets; because there can be many such sets, the flag NEW-OPTION-SET-LIMIT controls how many are generated at once. Each set is given a weighting (see flags WEIGHT-x-COEFFICIENT and WEIGHT-x-FN, for x = A,B,C), and the lowest-weighted set is chosen for searching. If the weight of the lowest-weighted set is too large, TPS may generate more sets; the interpretation of "too large" is given by MS91-WEIGHT-LIMIT-RANGE. If the search fails, it will be discarded; if it runs out of time then it will be re-weighted to be continued later (see RECONSIDER-FN).

MS91-7 Begin mating search MS91-7 on the current expansion proof. Primitive substitutions and duplications are performed systematically, with multiple jforms being worked on simultaneously. On each particular jform, the search procedure MS90-3 is used. The flags MAX-SEARCH-LIMIT and SEARCH-TIME-LIMIT are used to control the amount of time spent on each jform.

The order in which the possible jforms are considered depends on a number of flags. Firstly, the primitive substitutions which are generated are determined by the values of MAX-PRIM-DEPTH, MIN-PRIM-DEPTH, PRIM-QUANTIFIER and NEG-PRIM-SUB. If DUP-ALLOWED is T, then additional options are generated corresponding to duplicated quantifiers. These options are then combined into sets; because there can be many such sets, the flag NEW-OPTION-SET-LIMIT controls how many are generated at once. Each set is given a weighting (see flags WEIGHT-x-COEFFICIENT and WEIGHT-x-FN, for x = A,B,C), and the lowest-weighted set is chosen for searching. If the weight of the lowest-weighted set is too large, TPS may generate more sets; the interpretation of "too large" is given by MS91-WEIGHT-LIMIT-RANGE. If the search fails, it will be discarded; if it runs out of time then it will be re-weighted to be continued later (see RECONSIDER-FN).

3.11. MS92-9 search procedure

MS92-9 Call mating search procedure MS92-9 on the current eproof. This procedure uses a naive level-saturation method, exhaustively searching a single jform before applying any duplications. Quantifier duplications are applied uniformly to outermost quantifiers. Will try primitive substitution for outermost variable only. Works on only a single jform at a time. The procedure is almost identical to MS88, except that the flag NUM-OF-DUPS is

used to govern how many times the outermost quantifier may be duplicated. The internal representation of variables is as in MS90-3.

3.12. MS93-1 search procedure

MS93-1 Begin mating search MS93-1 on the current expansion proof. The search is basically identical to MS89, but is performed using the internal variable representations of MS90-9. Primitive substitutions and duplications are performed systematically, with multiple jforms being worked on simultaneously. On each particular jform, the search procedure MS92-9 is used. The flags MAX-SEARCH-LIMIT, SEARCH-TIME-LIMIT, and RANK-EPROOF-FN are used to control the search. See also the command SHOW-OPTION-TREE.

3.13. MS98-1 search procedure

MS98-1 Begin the MS98-1 mating search.

MS98-DUP Make NUM-OF-DUPS duplications in the current etree.

MS98-PRIM Make all possible primitive substitutions and then NUM-OF-DUPS duplications in the current etree.

3.14. Proof Translation

MERGE-TREE If the mating is complete, applies substitutions to the expansion tree, then applies Pfenning's MERGE algorithm, eliminating redundant expansion terms.

3.15. Vpforms

CJFORM Create a new jform for the expansion tree associated with the current mating-search top-level. You need to use this command only if you modify the expansion tree interactively and you are constructing a mating interactively.

CW *label*

CWD *label*

CWS *label*

NUM-HPATHS Counts the number of horizontal paths through the given jform.

NUM-VPATHS Counts the number of vertical paths through the given jform.

VP Use this operation for displaying vertical path diagram on the terminal with default settings. For complete control over the defaults use edop VPF.

VPD Use this operation for saving VP diagrams in a file. You may want to change the values of the variables VPD-FILENAME, VPD-STYLE, VPD-PTYPES, VPD-BRIEF, VPD-VPFPAGE.

VPETREE Display the VP diagram of the ETREE as used in mating-search.

VPT *file* Prints the path diagram, in a format understood by TeX, for a JForm or a GWFF. At present, it chops off whatever will not fit on one page. The following flags affect the output: 1. VPD-BRIEF controls whether labels or wffs are printed. 2. VPD-PTYPES controls whether types are printed. 3. TEXFORMAT controls whether the vertical or horizontal path diagram is printed. 4. ALLSCOPEFLAG controls where square brackets are printed.

3.16. Moving Commands

0 Move back to previous node, e.g., undo the last L or R command. Note that 0 stands for the numeral zero.

D Move down one node in etree (to leftmost node if more than one successor).

FB Find the topmost binder.

FI Find an infix node.

GOTO *node* Move to a specified node.

L For an infix etree node, move to the left argument.

R For an infix etree node, move to the right argument.

UP Move up one node in etree.

^ Move upwards to root of expansion tree.

3.17. Statistics

DEL-DUP-CONNS Deletes duplicate connections from a mating. This should be necessary only for propositional formulas.

STATS Display statistics for the active mating and totals for all matings in this expansion proof.

3.18. Miscellaneous

EXPUNGE Frees up space by getting rid of all expansion proofs and option trees. If you only want to get rid of old expansion proofs and option trees, you can use EXPUNGE-OLD to do you job. Warning : After using EXPUNGE, many commands such as ETD, VP, ..., don't work until you re-initialise the current expansion proof by using commands such as SUB, MATE, ...

EXPUNGE-OLD Frees up space by getting rid of all old expansion proofs and option trees. If you'd like to get rid of all(not only old) expansion proofs and option trees, you must use EXPUNGE to do your job. Warning : Never use EXPUNGE-OLD if you are going to use EXPUNGE, or you cannot get the expected result!

4. Matingstree Commands

The internal name of this category is MTREEOP. A matingstree command can be defined using DEFMTREEOP. Allowable properties are: `MTREE-ALIAS`, `MTREE-MOVE`, `MTREE-PRINT`, `MTREE-DEFAULT`, `MTREE-ARGS`, `MHELP`.

4.1. Top Levels

LEAVE leaving the mtree top level.

4.2. Mtree Operations

ADD-CONN *literal1 oblig1 literal2 oblig2*
: Add a connection. The subsumption is considered. The usage of the command is exactly as the usage of ADD-CONN in MATE.

CHOOSE-BRANCH
: Remove all matingstree branches except the one leading to the current matingstree node (which must be a leaf of the matingstree, and must be complete). This will also do some preliminary merging, by deleting all of the jforms which are associated with the deleted nodes.

COMPLETE-P Check the completeness of the current mating. The usage of the command is exactly the same as the usage of the mate command COMPLETE-P.

D *node matingstree* Go down one level. D <nth> means go down along the nth subnode. Counting begins from 0. Without argument, D means go down along the leftmost subnode.

GOTO *node matingstree*
: GOTO <node> means to go to the given node. If <node> is not given, it means to go to the root of the matingstree

INIT Initialise the matingstree. This is done automatically when you enter the matingstree top level, but can be used subsequently to return everything to the state it was in when you first entered the mtree top level.

KILL *node* KILL <node> means to mark the given node and all nodes below it as dead.

PICK *literal obligation*
: Pick a leaf which you may try to mate with another later. (MB: I think that PICK N behaves as though you had just added a connection to N, and generates the appropriate obligations, without actually demanding another leaf to connect with. I think.)

PRUNE Remove all dead leaves below (but not including) the current matingstree.

REM-NODE Remove the last connection. The subsumption is considered. If the node is the root, the whole matingstree is removed. The usage of the command is exactly the as the usage of REM-LAST-CONN. Please check the help message for REM-LAST-CONN if necessary.

RESURRECT *node* RESURRECT <node> means to mark the given node and all nodes below it as alive.

SHOW-MATING Show the connections in the mating associated with the current node.

SHOW-SUBSTS Show the substitution stack associated with a matingstree node.

SIB *matingstree* Go to the next sibling of this node.

UNIFY Go into UNIFY toplevel and check the UTREE structure associated with the current node in the matingstree. The unification tree associated with the mating is passed on to the unification top-level. Any changes made to this tree are destructive. Applicable only for a higher-order unification problem. Mainly use to check the UTREE structure.

UP *matingstree* Go up one level.

4.3. Mtree Printing

CONNS-ADDED *name*
: Print out all of the connections which have already been added to the given matingstree node. If no node is given, the current node is used.

LIVE-LEAVES *name*
: Print out all of the live leaves in the tree below the given matingstree node. If no node is given, the root node is used.

PM-NODE *name* Print out the given matingstree node in detail. If no node is given, the current matingstree is used.

PMTR *name* Print out the given matingstree as a tree, showing the obligations at each node. If no matingstree is given, the current-matingstree is printed out.

Matingstrees enclosed in curly brackets are marked as dead. Branches with *'s denote nodes that are being omitted for lack of space. The cure for this is to either start printing from a node lower in the tree, or make the screen wider. See also PMTR*.

PMTR* *name* Print out the given matingstree as a tree, showing the obligations at each node. If no matingstree is given, the current-matingstree is printed out.

Numbers in round brackets are open obligations. If the brackets end in "..", there are too many open obligations to fit under the mstree label.

Leaves underlined with ^'s are closed matingstrees. Matingstrees enclosed in curly brackets are marked as dead. Branches with *'s denote nodes that are being omitted for lack of space. The cure for this is to either start printing from a node lower in the tree, or make the screen wider. See also PMTR.

PMTR-FLAT *name* Print out the given matingstree in a flat format. If no matingstree is given, the current matingstree is printed out.

POB *name* Print out the vpform associated with the given obligation node. If no obligation is specified, then the first open obligation in the current obligation tree is used. See the flag DEFAULT-OB-DEEP.

POB-LITS *name* Print out the unblocked literals in a given obligation tree. If no argument is given, the current-obligation tree is the default.

POB-NODE *name* Print out the given obligation in detail. If no obligation is given, then the first open obligation in the current obligation tree is used. See the flag DEFAULT-OB-DEEP.

POTR *name* Print out the given obligation tree as a tree. If no obligation is given, the tree below the current obligation is printed out.

Numbers in round brackets are open obligations; those in square brackets are closed. Branches with *'s denote nodes that are being omitted for lack of space. The cure for this is to either start printing from a node lower in the tree, or make the screen wider.

POTR*-FLAT *name*
Print out the given obligation tree in flat form, with the jforms attached to all nodes. If no argument is given, the whole obligation tree is printed out.

POTR-FLAT *name* Print out the given obligation tree in flat form, with the jforms attached to the leaves. If no argument is given, the current-obligation tree is printed out.

PPATH *name* Print out the path containing the given obligation. If no obligation is specified, then the first open obligation in the current obligation tree is used. See the flag DEFAULT-OB-DEEP.

PPATH* *name* Print out the path containing the given obligation, and show all of the obligations on this path. If no obligation is specified, then the first open obligation in the current obligation tree is used. See the flag DEFAULT-OB-DEEP.

4.4. Mtree Auto

ADD-ALL-LIT *literal obligation*
Attempt to mate a literal with all potential mates on the current path.

ADD-ALL-OB *obligation*
Attempt to mate all literals in an obligation with all potential mates on the current path.

EXPAND-LEAVES *mtree*
Apply ADD-ALL-OB to all live leaves of the current matingstree that lie below the given node (or the current node, if no node is given). WARNING: Potential combinatorial explosion!

GO Call the matingstree procedure given in DEFAULT-MS.

MT94-11 *mtree* Apply EXPAND-LEAVES repeatedly to all live leaves of the current matingstree that lie below the given node (or the current node, if no node is given), until a closed leaf is generated. WARNING: Potential combinatorial explosion!

MT94-12 *mtree* Least Branching Search: In each leaf node, take the current obligation and find a literal that can be mated, but with as few mates as possible. Add all of these mates as sons to this node. Repeat until a closed leaf is generated. This search is probably not complete.

MT95-1 *mtree* Fewest Obligations Search: Choose the matingstree node (from the entire tree, not just the tree below the current node) with the fewest open obligations. Go to that node and do one step of MT94-12 (i.e. choose the literal with the fewest number of mates, and generate all of the associated branches of the mtree). Repeat until a closed leaf is generated. This search is probably not complete.

QRY *literal obligation*
Output a list of literals which can be mated with a given literal.

5. Unification Commands

The internal name of this category is UNIFOP. An unification command can be defined using DEFUNIFOP. Allowable properties are: `UNIF-ARGTYPES`, `UNIF-ARGNAMES`, `UNIF-ARGHELP`, `UNIF-DEFAULTFNS`, `UNIF-APPLICABLEP`, `UNIF-MAINFNS`, `PRINT-COMMAND`, `MOVE-COMMAND`, `MHELP`.

5.1. Top Levels

LEAVE Exit unification.

5.2. Unification

0 Replace the current topnode with the node on top of the nodestack. Generally, typing an integer n will go to the nth son of the current node. Compare the command NTH-SON.

APPLY-SUBST *var term*
Apply a substitution, suggested by the user, to the current topnode. Modifies the unification tree.

EPROOF-UTREE Create a new utree whose root has all the dpairs associated with the current mating. (The existing utree may have some of the dpairs added lower down the tree; this will bring them all to the top). See also NAME-DPAIR.

GO Call unification in automatic mode. Will search for unifiers only below the current-topnode.

GOTO *name* Go to the specified node in the unification tree.

MATCH This command is applicable only if current-topnode is a non-terminal leaf node. Calls TPS's version of Huet's MATCH algorithm to find substitutions at the current topnode. The pair selected by MATCH is determined by the value of the flag APPLY-MATCH.

MATCH-PAIR *n* This command is applicable only if current-topnode is a non-terminal leaf node. Calls TPS's version of Huet's MATCH algorithm to find substitutions at the current topnode. n refers to the nth dpair, and this must be a flexible-rigid dpair.

NAME-DPAIR *name*
Give a name to the dpairset associated with the current topnode. This is most useful when UNIFY has been issued from the MATE top level, and you want to name the current dpair so that you can save it in the library. See also EPROOF-UTREE.

NTH-SON *n* Go to the nth descendant of the current-topnode. Instead of using this command, you can simply type n on the unification top level to go to the nth descendant. It has no effect if the current-topnode has no descendents.

P *name* Displays the current unification node; show its name, measure, number of descendants, substitutions added and free variables. Does not display the disagreement pairs (use PP or PP* for that), or the cumulative substitutions from this node to the root (use SUBST-STACK for that).

PALL *name filename verbose*
Displays all the disagreement pairs at every node below the given node. (Similar to PP, but for the entire tree below the current node.)

PP Displays the disagreement pairs at the current node. See also PP*. More information about the current node is given by the command P.

PP* Displays the disagreement pairs at the current-topnode, including the order of each pair and other information. See also PP. The other information displayed includes (for each wff, each disagreement pair and the whole set of disagreement pairs): 1) the order (e.g. "x(i)" is first order, and so on). 2) whether it is monadic (all function constants are unary). 3) whether it is linear (all free vars occur once only). 4) whether it is a matching problem (one side of a dpair has no free vars). 5) whether it is a relaxed pattern (all free vars have only bound vars as arguments). 6) whether it is a pattern (all free vars have distinct bound vars as arguments). 7) whether a disagreement pair is variable-disjoint (the free vars on the left are disjoint from those on the right). 8) whether the set of disagreement pairs can be partitioned into sets in which each free var in the whole problem occurs in at most one set. 9) whether there are any free vars that occur only once, or not at all, in the whole problem. These conditions all appear in the literature on higher-order unification; see, for example, Prehofer's paper in CADE '94.

More information about the current node is given by the command P.

SIMPLIFY A call to TPS's version of Huet's SIMPL algortihm. Dpairs in the current topnode are replaced by the dpairs returned by the call. It will also find substitutions of the form (var . term) provided 'var' does not occur in 'term'. This command will alter the unification tree.

STATS Statistics about the current unification tree.

SUBST-STACK *filename*

Displays the substitution stack for the current topnode. See also P, PP, PP* for other information about the current node.

UTREE *name filename verbose*
Displays the unification tree and the associated substitutions at each node which is below the specified node. Display is in a flat format; UTREE* prints the same information in a tree format.

UTREE* *name print-subs*
Displays the unification tree and the associated substitutions at each node which is below the specified node. Display is in a tree format; UTREE prints the same information in a flat format. Display shows nodes as numbers, followed by I for imitation, P for projection, ~ for negation, A for administrative (e.g. anything generated by SIMPL). Optionally shows the most recent substitution on the subst-stack at each node.

^ Go to the parent node of the current-topnode. (i.e. move up one level in the tree).

^^ Go to the root node in the unification tree (the node with name "0").

5.3. Dpairs

ADD-DPAIR *name elt1 elt2*
If the disagreement set already exists, insert a disagreement pair at the front. Else create a new disagreement set consisting of this dpair only.

ADD-DPAIRS-TO-NODE *name free-vars*
Add new dpairs to the disagreement set at the CURRENT-TOPNODE. Applicable only if CURRENT-TOPNODE is a non failure leaf node. ‘Name’, the first argument to this command must already represent a disagreement set. Use the command ADD-DPAIR,etc., to create this set.

ADD-DPAIRS-TO-UTREE *name free-vars*
Add new dpairs at all non failure leaf nodes.

FIND-NESTING Find the values for MAX-SUBSTS-* implied by the current node.

PRUNE Prune all the branches which have either reached the maximum allowed depth, or which end only in failure nodes.

RM-DPAIR *name elt1 elt2*
Remove a disagreement pair from a disagreement set.

SHOW-DPAIRSET *name*
Show a disagreement set.

UNIF-PROBLEM *name free-vars*
Set up a new unification problem. ‘Name’, the first argument to this command must already represent a disagreement set. Use the command ADD-DPAIR to create this set. This is in some ways the inverse of the NAME-DPAIR command.

6. Test-Top Commands

The internal name of this category is TESTCMD. A test-top command can be defined using DEFTEST. Allowable properties are: TEST-ARGTYPES, TEST-ARGNAMES, TEST-ARGHELP, TEST-DEFAULTFNS, TEST-MAINFNS, MHELP.

6.1. Top Levels

LEAVE Leave TEST-TOP to the next enclosing top level.

6.2. Mating search

BREADTH-FIRST-SEARCH *modename testwin*
: Equivalent to setting TEST-NEXT-SEARCH-FN to BREADTH-FIRST-SEARCH and then typing GO. Permanently changes TEST-NEXT-SEARCH-FN.

CLOSE-TESTWIN Closes the window that displays the test-top summary. Use ..../tps/utilities/vpshow (from a shell, not from TPS) to view the output file again.

CONTINUE *modename testwin*
: Continue searching with current searchlist & current problem (similar to GO, but will continue from the last point reached rather than restarting at the beginning again).

EXHAUSTIVE-SEARCH *modename testwin*
: Equivalent to setting TEST-NEXT-SEARCH-FN to EXHAUSTIVE-SEARCH and then typing GO. Permanently changes TEST-NEXT-SEARCH-FN.

FIND-BEST-MODE *modename testwin*
: This command effectively runs PUSH-UP until it finds a mode that works, and then runs PRESS-DOWN until it finds the best mode it can. Before using this command, use the MODE command to set up a mode in which the current theorem can not be proven. Also check the value of the TEST-INCREASE-TIME flag (it should probably not be zero). Then PUSH-UP will systematically vary the values of the flags listed in the TEST-EASIER-IF-* flags, using the PUSH-UP search function (see the help message for TEST-NEXT-SEARCH-FN). Once a correct mode is discovered, it will systematically vary the values of the flags listed in the TEST-FASTER-IF-* flags, using the PRESS-DOWN search function, until it finds as good a mode as it can. The values of TEST-REDUCE-TIME, TEST-NEXT-SEARCH-FN, TEST-INCREASE-TIME and TEST-FIX-UNIF-DEPTHS will be permanently changed.

GO *modename testwin*
: Start searching with current searchlist & current problem.

OPEN-TESTWIN *filename*
: Open a window which will display a summary of the test-top output. The window can be closed with the command CLOSE-TESTWIN. The size of the text is determined by the flag CHARSIZE, and the current width of the window by the flag TESTWIN-WIDTH. The initial height of the window is determined by TESTWIN-HEIGHT.

PRESS-DOWN *modename testwin*
: Before using this command, use the MODE command to set up a mode in which the current theorem can be proven. Also check the value of the TEST-INITIAL-TIME-LIMIT flag (it should be high enough that the first attempt at proof will succeed). Then PRESS-DOWN will systematically vary the values of the flags listed in the TEST-FASTER-IF-* flags, using the PRESS-DOWN search function (see the help message for TEST-NEXT-SEARCH-FN). The values of TEST-REDUCE-TIME, TEST-NEXT-SEARCH-FN and TEST-FIX-UNIF-DEPTHS will be permanently changed (to T, PRESS-DOWN and T respectively).

Note that this is NOT the same as PRESS-DOWN-2, since it automatically generates a searchlist rather than relying on the user to provide one.

PRESS-DOWN-2 *modename testwin*
: Equivalent to setting TEST-NEXT-SEARCH-FN to PRESS-DOWN-2 and then typing GO. Permanently changes TEST-NEXT-SEARCH-FN. Note that this is NOT the same as typing PRESS-DOWN; this will use the user-defined searchlist rather than an automatically generated one.

PUSH-UP *modename testwin*
: This command effectively runs PUSH-UP until it finds a mode that works, and then stops. Before using this command, use the MODE command to set up a mode in which the current theorem can not be proven. Also check the value of the TEST-INCREASE-TIME flag (it should probably not be zero). Then PUSH-UP will systematically vary the values of the flags listed in the TEST-EASIER-IF-* flags, using the PUSH-UP search function (see the help message for TEST-NEXT-SEARCH-FN). The value of TEST-NEXT-SEARCH-FN will be

changed to PUSH-UP.

Note that this is NOT the same as PUSH-UP-2, since it automatically generates a searchlist rather than relying on the user to provide one

PUSH-UP-2 *modename testwin*
: Equivalent to setting TEST-NEXT-SEARCH-FN to PUSH-UP-2 and then typing GO. Permanently changes TEST-NEXT-SEARCH-FN. Note that this is NOT the same as typing PUSH-UP; this will use the user-defined searchlist rather than an automatically generated one.

SEARCH-ORDER *name*
: Show the order in which things will be changed if the search is started now using the given searchlist.

6.3. Searchlists

ADD-FLAG *flag init range*
: Add a single flag to the current searchlist. To change the current searchlist, use NEW-SEARCHLIST.

ADD-FLAG* Repeatedly add new flags to the current searchlist.

ADD-FUNCTION *name*
: Add a function to a searchlist. This function will be evaluated on every iteration of the search, and will generally reset certain flags. The special functions defined so far are: UNIFORM-SEARCH-FUNCTION sets max-utree-depth, max-search-limit and max-substs-quick using the values of max-search-depth, search-time-limit and max-substs-var respectively, and then sets TEST-INITIAL-TIME-LIMTI to allow 5 option sets on the first try, then 10, then 15, and so on. BASIC-SEARCH-THEN-UNIFORM-SEARCH runs the current searchlist once over, allowing 1 hour for each setting of the flags. Then it switches the searchlist to UNIFORM-SEARCH-2 and contines with that.

ADD-SUBJECTS *subjects*
: Add all the flags concerning the given subjects to the current searchlist.

NEW-SEARCHLIST *name*
: Make a new searchlist; i.e. begin a new list of flags to be varied. This command also changes the current searchlist.

QUICK-DEFINE *name succ*
: Define a searchlist the quick and dirty way! If the current flag settings are OK (i.e. are a successful mode), will create a searchlist in which the flags given in the values of the TEST-FASTER-* flags (do LIST TEST-TOP for a listing) vary over values which ought to give a faster search than the current values. If the current flag settings are not OK, will create a searchlist in which the flags given in the values of the TEST-EASIER-* flags vary over values which ought to make the search easier than the current values. The maximum number of values for any flag to take is governed by TEST-MAX-SEARCH-VALUES.

REM-FLAG *flag* Remove a single flag from the current searchlist. To change the current searchlist, use NEW-SEARCHLIST.

REM-FLAG* Repeatedly remove flags from the current searchlist.

REVISE-DEFAULTS *old-slist new-slist*
: For each flag in the given searchlist, change the default setting to the current value of the flag, and put the default setting into the range (unless it's already there). This is useful in conjunction with SCALE-UP and SCALE-DOWN; you can keep one searchlist (let's call it MASTER-SLIST) containing all of the flags you're likely to want to vary. Then if the current flag settings are a good mode and you want to try and find a better one, do REVISE-DEFAULTS followed by SCALE-DOWN MASTER-SLIST; if the current settings are a bad mode and you want to try to find one that works, do REVISE-DEFAULTS followed by SCALE-UP MASTER-SLIST.

SCALE-DOWN *old-slist new-slist*
: Rewrites a searchlist under the assumption that the initial values in the searchlist (together with appropriate settings of the other flags) constitute a successful mode, and that TEST is being run in order to find a faster mode. This will discard all settings that would make the search slower, and will arrange the range of values in such a way that the bounds of the search will gradually decrease until the proof cannot be completed. If this makes the range empty or a singleton, the flag is removed from the searchlist. See the TEST-FASTER-* flags

SCALE-UP *old-slist new-slist*
: Rewrites a searchlist under the assumption that the initial values in the searchlist (together with appropriate settings of the other flags) do not constitute a successful mode, and that TEST is being run in order to find a mode that works. This will discard all settings that would make the search harder, and will arrange the range of values in such a way that the bounds of the search will gradually increase until the proof (with a bit of luck) can be completed. If this makes the range empty or a singleton, the flag is removed from the searchlist. See the TEST-EASIER-* flags.

SEARCHLISTS Print a list of all searchlists currently in memory.

SHOW-SEARCHLIST *name*
: Show contents of a searchlist.

VARY-MODE *modename slistname use-mode*
: Go through an existing mode, flag by flag, creating a searchlist by picking out relevant flags from it. All useless flags (i.e. ones that cannot affect the search time) will be automatically stripped out. The default flag value in the searchlist will be its value in the mode. You can also optionally set the current flag values to the values in the mode (equivalent to the MODE command).

6.4. Library

DELETE *name type*
: Delete a saved searchlist or mode (equivalent to the library command DELETE.

FETCH *name type*
: Retrieve a searchlist or mode from the library. Exactly like the library function FETCH, except that when a searchlist is retrieved, it will become the current searchlist.

INSERT *name type comment*
: Like the library command INSERT; will save a searchlist in the library. Will also save a mode that has been found by using GO.

7. Editor Commands

The internal name of this category is EDOP. An editor command can be defined using DEFEDOP. Allowable properties are: ALIAS, RESULT->, EDWFF-ARGNAME, DEFAULTFNS, MOVE-FN, MHELP.

7.1. Top Levels

LEAVE	Exit the editor with all the changes in place.
NOOP	Do nothing.
OK	Exit the editor with all the changes in place.

7.2. Printing

P	Print a wff using the global settings of all flags.
PP	Pretty-print a wff.
PS	Print a wff showing all brackets and dots.
PT	Print a wff showing types.

7.3. Weak Labels

CW *label*	Assigns a label to the edwff, but does not change the edwff. You can use the label to refer to this wff later.
DELWEAK *label*	Replaces all occurrences of the label with the wff it represents in the current wff.
DW	Replace a top level occurence of the label by the wff it represents.
DW*	Replace all labels in a wff by the wffs represented by them.
NAME *label*	Assign a label to the edwff, and replace the edwff with this label.
RW *label*	Makes current edwff the new value of label (which must already exist).

7.4. Saving Wffs

SAVE *label*	Save a wff by appending it to the file SAVEDWFFS. The weak label name should not already exist (if it does, remove it using RW). The wffs that are saved to this file can be reloaded using the command TLOAD "savedwffs.lisp". This command dates from before the LIBRARY top level was introduced; you should probably avoid it. If you want to save a gwff, use CW to create a weak label, then go into the library with LIB and use INSERT to save the wff.

7.5. Recording

O	Invert PRINTEDTFLAG, that is switch automatic recording of wffs in a file either on or off. When switching on, the current wff will be written to the PRINTEDTFILE. Notice that the resulting file will be in Scribe format; if you want something you can reload into TPS, then use the SAVE command.
REM *rm*	Write a remark into the PRINTEDTFILE.

7.6. Vpforms

CJFORM	Converts the given GWFF to JFORM.
DJFORM	Converts the given JFORM to GWFF. May not work with skolemised jforms.
NUM-HPATHS	Counts the number of horizontal paths through the given jform.
NUM-VPATHS	Counts the number of vertical paths through the given jform.
PJ	Prints the given gwff, using lists for jforms.
PROP-CJFORM	Converts the given GWFF to JFORM.
VP	Prints a vertical path diagram. This is like VP in the MATE top level, but will use the current edwff to create a jform if none is currently available.

VPD Use this operation for saving VP diagrams in a file. You may want to change the values of the variables VPD-FILENAME, VPD-STYLE, VPD-PTYPES, VPD-BRIEF, VPD-VPFPAGE.

VPF *file style ptypes brief vpfpage comment*
Prints the vertical path diagram for a JForm or a GWFF.

VPT *file* Prints the path diagram, in a format understood by TeX, for a JForm or a GWFF. At present, it chops off whatever will not fit on one page. The following flags affect the output: 1. VPD-BRIEF controls whether labels or wffs are printed. 2. VPD-PTYPES controls whether types are printed. 3. TEXFORMAT controls whether the vertical or horizontal path diagram is printed. 4. ALLSCOPEFLAG controls where square brackets are printed.

7.7. Moving Commands

0 Move up one-level, i.e., undo the last L, R, D, or A command. Note that 0 stands for the numeral zero.

A for an expression like P x y, delete the rightmost element; in this example the result will be to make Px the current expression. For a quantified expression, it will move to the quantified variable.

D for an expression like P x y, move to the rightmost element; in this example y. For a quantified expression it will move to the scope of the quantifier.

FB Find the first binder (left to right)

FI Find an infix operator.

L for an infix-operator, move to the left argument.

R for an infix-operator, move to the right argument.

UNDO Moves up (like 0), but throws away any editing since your last downward moving command (typically A,D,L,or R.

XTR Makes the current edwff the top wff.

^ Move upwards through enclosing wffs all the way to the top.

7.8. Changing Commands

ASRB Apply the following laws to a wff: A and (A or B), (A or B) and A --> A or B A and (B or A), (B or A) and A --> B or A A or (A and B), (A and B) or A --> A (B and A) or A, (B and A) or A --> A.

ASSL Apply the left associative law: A op (B op C) --> (A op B) op C.

ASSR Apply the right associative law: (A op B) op C --> A op (B op C).

CMRG Delete the truth constants from a wff: A and TRUTH, TRUTH and A --> A A and FALSEHOOD, FALSEHOOD and A --> FALSEHOOD A or TRUTH, TRUTH or A --> TRUTH A or FALSEHOOD, FALSEHOOD or A --> A A implies TRUTH --> TRUTH TRUTH implies A --> A A implies FALSEHOOD --> not A FALSEHOOD implies A --> TRUTH A equiv TRUTH, TRUTH equiv A --> A A equiv FALSEHOOD, FALSEHOOD equiv A --> not A not TRUTH --> FALSEHOOD not FALSEHOOD --> TRUTH.

CMUT Apply the commutative laws to a formula: A and B --> B and A A or B --> B or A A implies B --> not B implies not A A equiv B --> B equiv A.

CNTOP *conn* Change the top connective of a formula. For example, "cntop or" will change "A and B" into "A or B"; "cntop exists" will change "forall x P x" into "exists x P x".

DIST-CTR Apply the distributivity laws to a wff in the contracting direction: (A and B) or (A and C) --> A and (B or C) (A or B) and (A or C) --> A or (B and C) (B and A) or (C and A) --> (B or C) and A (B or A) and (C or A) --> (B and C) or A.

DIST-EXP Apply the distributivity laws to a wff in the expanding direction: A and (B or C) --> (A and B) or (A and C) A or (B and C) --> (A or B) and (A or C) (B or C) and A --> (B and A) or (C and A) (B and C) or A --> (B or A) and (C or A).

DL Delete the topmost binary connective and its left scope

DNEG Remove a double negation: not not A --> A.

DR Delete the topmost binary connective and its right scope

MRG Apply the following laws to a wff: A and A --> A A or A --> A A implies A --> TRUTH A and not A, not A and A --> FALSEHOOD A or not A, not A or A --> TRUTH A implies not A --> not A not A implies A --> A A equiv not A, not A equiv A --> FALSEHOOD.

PMUT Permute the two components of an infix operator: A op B --> B op A

SUBEQ Apply the following law to a formula: A equiv B --> (A implies B) and (B implies A).

SUBIM Apply the following law to a formula: A implies B --> not A or B.

7.9. Recursively Changing Commands

ASRB* Recursively apply the following laws to a wff: A and (A or B), (A or B) and A --> A or B A and (B or A), (B or A) and A --> B or A A or (A and B), (A and B) or A --> A (B and A) or A, (B and A) or A --> A.

ASSL* Recursively apply the left associative law: A op (B op C) --> (A op B) op C.

ASSR* Recursively apply the right associative law: (A op B) op C --> A op (B op C).

CMRG* Recursively delete the truth constants in a wff: A and TRUTH, TRUTH and A --> A A and FALSEHOOD, FALSEHOOD and A --> FALSEHOOD A or TRUTH, TRUTH or A --> TRUTH A or FALSEHOOD, FALSEHOOD or A --> A A implies TRUTH --> TRUTH TRUTH implies A --> A A implies FALSEHOOD --> not A FALSEHOOD implies A --> TRUTH A equiv TRUTH, TRUTH equiv A --> A A equiv FALSEHOOD, FALSEHOOD equiv A --> not A not TRUTH --> FALSEHOOD not FALSEHOOD --> TRUTH.

CMUT* Recursively apply the commutative laws to a formula: A and B --> B and A A or B --> B or A A implies B --> not B implies not A A equiv B --> B equiv A.

DIST-CTR* Recursively apply the distributive laws to a wff in the contracting direction: (A and B) or (A and C) --> A and (B or C) (A or B) and (A or C) --> A or (B and C) (B and A) or (C and A) --> (B or C) and A (B or A) and (C or A) --> (B and C) or A.

DIST-EXP* Recursively apply the distributive laws to a wff in the expanding direction: A and (B or C) --> (A and B) or (A and C) A or (B and C) --> (A or B) and (A or C) (B or C) and A --> (B and A) or (C and A) (B and C) or A --> (B or A) and (C or A).

DNEG* Recursively remove double negations: not not A --> A.

MRG* Recursively apply the following laws to a wff: A and A --> A A or A --> A A implies A --> TRUTH A equiv A --> TRUTH A and not A, not A and A --> FALSEHOOD A or not A, not A or A --> TRUTH A implies not A --> not A not A implies A --> A A equiv not A, not A equiv A --> FALSEHOOD.

PMUT* Recursively permute the two components of an infix operator: A op B --> B op A

SUBEQ* Recursively apply the following law to a formula: A equiv B --> (A implies B) and (B implies A).

SUBIM* Recursively apply the following law to a formula: A implies B --> not A or B.

7.10. Embedding Commands

MBED-AL *rgwff* Embed the current edwff in the left scope of AND. The right scope is provided by the user.

MBED-AR *lgwff* Embed the current edwff in the right scope of AND. The left scope is provided by the user.

MBED-E *vquant* Embed the current edwff in the scope of an existential quantifier. The variable of quantification is provided by the user.

MBED-E1 *vquant* Embed the current edwff in the scope of an exists1 quantifier. The variable of quantification is provided by the user.

MBED-F *vquant* Embed the current edwff in the scope of a universal quantifier. The variable of quantification is provided by the user.

MBED-IL *rgwff* Embed the current edwff as the antecedent of a conditional. The consequent is provided by the user.

MBED-IR *lgwff* Embed the current edwff as the consequent of a conditional. The antecedent is provided by the user.

MBED-L *vquant* Embed the current edwff in the scope of lambda. The variable of quantification is provided by the user.

MBED-OL *rgwff* Embed the current edwff in the left scope of OR. The right scope is provided by the user.

MBED-OR *lgwff* Embed the current edwff in the right scope of OR. The left scope is provided by the user.

MBED-QL *rgwff* Embed the current edwff on the left side of equivalence. The right side is provided by the user.

MBED-QR *lgwff* Embed the current edwff on the right side of equivalence. The left side is provided by the user.

MBED=L *rgwff* Embed the current edwff on the left side of equality. The right side is provided by the user.

MBED=R *lgwff* Embed the current edwff on the right side of equality. The left side is provided by the user.

7.11. Rewriting commands

ARR — Apply one active rewrite rule to the current edwff; attempt different active rules in the order in which they are listed by LIST-RRULES until one works. If any current rules are bidirectional, you will be prompted about which direction to apply them in.

ARR* — Apply one active rewrite rule to the current edwff; attempt different active rules in the order in which they are listed by LIST-RRULES until one works. If any current rules are bidirectional, you will be prompted about which direction to apply them in. Repeat this until no more rules are applicable. CAUTION: may not terminate.

ARR1 *rule* — Apply a rewrite rule (active or inactive) to the current edwff. If the rule is bidirectional, you will be prompted about which direction to apply it in.

ARR1* *rule* — Apply a rewrite rule (active or inactive) repeatedly to the current edwff. If the rule is bidirectional, you will be prompted about which direction to apply it in. CAUTION: may not terminate.

MAKE-RRULE *name gwff2 func types bidir appfn mhelp*
Create a rewrite rule whose left-hand side is the current edwff.

UNARR — Unapply one active rewrite rule to the current edwff (i.e. apply it in the reverse direction); attempt different active rules in the order in which they are listed by LIST-RRULES until one works. If any current rules are bidirectional, you will be prompted about which direction to apply them in.

UNARR* — Unapply one active rewrite rule to the current edwff (i.e. apply it in the reverse direction); attempt different active rules in the order in which they are listed by LIST-RRULES until one works. Repeat this until no more rules are applicable. If any current rules are bidirectional, you will be prompted about which direction to apply them in. CAUTION: may not terminate.

UNARR1 *rule* — Unapply a rewrite rule (active or inactive) to the current edwff. (i.e. apply it in the reverse direction). If the rule is bidirectional, you will be prompted about which direction to apply it in.

UNARR1* *rule* — Unapply a rewrite rule (active or inactive) repeatedly to the current edwff. (i.e. apply it in the reverse direction). If the rule is bidirectional, you will be prompted about which direction to apply it in. CAUTION: may not terminate.

7.12. Substitution

AB *newvar* — Alphabetic change of variable at top-level.

IB *term* — Instantiate a top-level universal or existential binder with a term.

PRIM-SUBST *var sub*
Replaces a variable with a primitive substitution. Differs from SUBST in that it will also replace quantified variables, and their quantifiers, as necessary.

REW-EQUIV — Replaces all occurences of the form ‘A EQUIV B’ according to the setting of the flag REWRITE-EQUIVS.

RP *rep-sym rep-by* — Replace one occurrence of a symbol (such as AND) by a predefined equivalent wff (such as [lambda p lambda q.~.p IMPLIES ~q]). In this example repsym is AND and rep-by is IMPLIES. To see if a symbol can be replaced by this command, enter HELP symbol; any such replacements will be listed under the heading ’Replaceable Symbols’.

RPALL *rep-sym rep-by*
Replace a all occurrences of a symbol by a predefined equivalent wff.

SUB *gwff* — Replaces the current wff by the wff supplied.

SUBST *term var* — Substitute a term for the free occurrences of variable in a gwff. Bound variables may be renamed, using the function in the global variable REN-VAR-FN.

SUBSTYP *typevar typesym*
Substitutes a type for a type variable in edwff.

7.13. Basic Abbreviations

ABBR — Lists all the abbreviations used in a gwff.

CONSTANTS — Lists all the logical constants used in a gwff, apart from the primitive constants AND FALSEHOOD IMPLIES NOT OR TRUTH.

EXPAND= — Instantiate outermost equality in gwff. Consults the flag REWRITE-EQUALITIES (but ignores it if it’s set to NONE).

EXPAND=* — Instantiate all equalities in gwff. Consults the flag REWRITE-EQUALITIES (but ignores it if it’s set to NONE).

INST *gabbr* Instantiate all occurrences of an abbreviation. The occurrences will be lambda-contracted, but not lambda-normalized.

INST1 Instantiate the first abbreviation, left-to-right.

INSTALL *exceptions*
Instantiate all definitions, except the ones specified in the second argument.

LIB-ABBR Lists all the library abbreviations used in a gwff.

NEW-DEFS Lists all the definitions used in a gwff that are either library abbreviations or weak labels.

7.14. Lambda-Calculus

ABNORM Convert the gwff to alphabetic normal form.

ETAB Eta-expands until original wff is part of a wff of base type.

ETAC Reduces [lambda x.fx] to f at top.

ETAN Reduces [lambda x.fx] to f from inside out.

ETAX Performs a one-step eta expansion.

LETA Returns the long-eta normal form of wff.

LEXP *var term occurs*
Converts the wff into the application of a function to the term. The function is formed by replacing given valid occurences of a term with the variable and binding the result.

LNORM Put a wff into lambda-normal form, using beta or beta-eta conversion according to the value of flag LAMBDA-CONV. Compare LNORM-BETA and LNORM-ETA.

LNORM-BETA Put a wff into beta-normal form, not using eta conversion. Compare LNORM and LNORM-ETA.

LNORM-ETA Put a wff into eta-normal form, not using beta conversion. Compare LNORM-BETA and LNORM.

RED Lambda-contract a top-level reduct. Bound variables may be renamed using REN-VAR-FN

ULNORM Convert a untyped wff into lambda-normal form. Be aware of unterminated reduction in untyped lambda calculus.

7.15. Negation movers

NEG Negates current wff, erasing double negations.

NNF Return the negation normal form of the given wff.

PULL-NEG Pulls negations out one level.

PUSH-NEG Pushes negation through the outermost operator or quantifier.

7.16. Miscellaneous

CLAUSE-FORM Converts the given wff to clause form, as if the resulting wff is to be given to a resolution theorem prover. The gwff is skolemized, rectified, etc.

7.17. Primitive Substitutions

NAME-PRIM Creates weak labels for primitive substitutions for the head variables of a wff.

PRT-PRIM Prints primitive substitutions for the head variables of a wff.

7.18. Miscellaneous

CNF Find the conjunctive normal form of a wff.

HEAD Find the head of a gwff.

HVARS Find all head variables of a wff.

MIN-SCOPE Minimize the scope of quantifiers in a gwff. Deletes vacuous quantifiers. During proof transformation, the gap between a formula and its min-quant-scope version is filled by RULEQ.

SUBFORMULAS *type*

Find all subformulas of a given type in a wff.

7.19. RuleP

SAT	Check whether a propositional wff is satisfiable.
VALID	Check whether a propositional wff is valid.

7.20. Skolemizing

SK1 *univflag*	Skolemize a wff using method S1. See page 127 of Andrews' book. If equivalences are present, you must eliminate them first by REW-EQUIV.
SK3 *univflag*	Skolemize a wff using method S3. At the moment it takes only those free variables which are universally quantified somewhere before, all other variables are considered to be constants. See page 127 of Andrews' book. If equivalences are present, you must eliminate them first by REW-EQUIV.

7.21. Quantifier Commands

DB	Delete the leftmost binder in a wff.
EP	Delete all accessible essentially existential quantifiers.
OP	Delete all accessible essentially universal quantifiers.

7.22. Wellformedness

DUPW *connective*	duplicates wff across connective.
EDILL	Find a minimal ill-formed subformula.
ILL	Return a list of messages, each the describing the error in a minimal ill-formed subparts of the argument.
TP	Return the type of a gwff.
WFFP	Test for a gwff (general well-formed formula).

8. Library Commands

The internal name of this category is LIBRARYCMD. A library command can be defined using DEFLIBRARY. Allowable properties are: `LIB-ARGTYPES`, `LIB-ARGNAMES`, `LIB-ARGHELP`, `LIB-DEFAULTFNS`, `LIB-MAINFNS`, `MHELP`.

8.1. Top Levels

LEAVE Leave LIBRARY to the next enclosing top level.

8.2. Display

KEY *string backup* Search for a string in the names of all library objects. If the given string is also a keyword (see SHOW-KEYWORDS), then the keywords for each library object will also be searched. This command does not search the help messages of library objects.

LIBFILES *backup* Lists all library files in the current default directory.

LIBOBJECTS-IN-FILE *file*
Lists the contents of a file.

LIST-OF-LIBOBJECTS *type backup*
List all objects or all objects of specified TYPE.

SCRIBE-ALL-WFFS *backup filter fname verbosity*
Write all wffs in all files in DEFAULT-LIB-DIR (and optionally BACKUP-LIB-DIR) to an mss file. The three verbosity settings are: MIN, which just shows the names of the objects, MED, which shows the help messages, keywords, provability and wffs as well, and MAX, which shows everything. As a filter, you can select any known keywords; only the wffs which satisfy all of the given keywords will be shown. See SHOW-KEYWORDS for a list of keywords.

SCRIBELIBDIR *directory types filename verbosity eject*
Print all the library files in a given directory into MSS files. See SCRIBELIBFILE for details.

SCRIBELIBFILE *filenamesin filenamesout verbosity*
Print the specified library files into MSS files. The three verbosity settings are: MIN, which just shows the names of the objects, MED, which shows the help messages, keywords, provability and wffs as well, and MAX, which shows everything. It can take a list of filenames and a corresponding list of output files; if the latter list is too long it will be truncated, and if it is too short then the last filename given will be used for all the remaining output (so you can write a group of library files to a single output file by only supplying one output filename). After leaving TPS, run the .mss files through Scribe and print the resulting files.

SEARCH *type stringlist backup*
Search the entire library, including all comments, for any one of a given list of strings, and return the names of all objects which contain such a string. This is useful for finding out, for example, which gwffs can be proven using either MS88 or MS89. WARNING: THIS COMMAND IS SLOW, AND CAN USE A LOT OF MEMORY. You might want to think about using the Unix "grep" command instead.

SEARCH2 *type stringlist backup*
Search the entire library, including all comments, for a given combination of strings. See also SEARCH. The syntax for the given list is essentially conjunctive normal form -- it should be a list of conjuncts, each of which is a list of disjuncts. For example: ((MS88) (THM)) finds everything containing THM and MS88 ((MS88 THM)) finds everything containing THM or MS88 ((MS88 MS89) (THM EXERCISE)) finds everything containing (MS88 or MS89) and (THM or EXERCISE). WARNING: THIS COMMAND IS SLOW, AND CAN USE A LOT OF MEMORY. You might want to think about using the Unix "grep" command instead.

SHOW *name type* Display a library object.

SHOW-ALL-WFFS *backup filter*
Show all wffs in all files in DEFAULT-LIB-DIR (and optionally BACKUP-LIB-DIR). As a filter, you can select any known keywords; only the wffs which satisfy all of the given keywords will be shown. See SHOW-KEYWORDS for a list of keywords.

SHOW-HELP *name type*
Display the help message associated with a library object.

SHOW-OBJECTS-IN-FILE *file types*
Lists all the objects of the given type (or types) in a file.

SHOW-TIMING *name screen*
Display the timing information of a gwff in the library. NOTE: Will only display timing information that has been recorded in standard DATEREC format. If you opt for output to go to a file as well as to the screen, the format of

the file will be SCRIBE or TEX if this is the current value of the STYLE flag, and GENERIC otherwise.

SHOW-WFF *name* Display the wff of a gwff in the library.

SHOW-WFF&HELP *name*
Display the wff of a gwff in the library, with the associated help message, keywords and provability status.

SHOW-WFFS-IN-FILE *file*
Lists the wffs in a file.

TEX-ALL-WFFS *backup filter fname verbosity*
Write all wffs in all files in DEFAULT-LIB-DIR (and optionally BACKUP-LIB-DIR) to a TeX file. The three verbosity settings are: MIN, which just shows the names of the objects, MED, which shows the help messages, provability and wffs as well, and MAX, which shows everything. As a filter, you can select any known keywords; only the wffs which satisfy all of the given keywords will be shown. See SHOW-KEYWORDS for a list of keywords.

TEXLIBDIR *directory types filename verbosity eject*
Print all the library files in a given directory into TEX files. See TEXLIBFILE for details.

TEXLIBFILE *filenamesin filenamesout verbosity*
Print the specified library files into TeX files. The three verbosity settings are: MIN, which just shows the names of the objects, MED, which shows the help messages, keywords, provability and wffs as well, and MAX, which shows everything. It can take a list of filenames and a corresponding list of output files; if the latter list is too long it will be truncated, and if it is too short then the last filename given will be used for all the remaining output (so you can write a group of library files to a single output file by only supplying one output filename). After leaving TPS, run the .tex files through TeX and print the resulting files.

8.3. Reading

DESTROY *name* Remove a library object from TPS (the object will remain stored in the library).

FETCH *name type* Make a library object available in TPS. Will create a new TPS object if EXPERTFLAG is set to T, otherwise will create a weak label for the new library object.

FIND-PROVABLE *backup*
Look for gwffs with a certain provability status.

RESTORE-MASTERINDEX
Restore library master index. Normally this need not be done by the user as it is done automatically when TPS is first entered. However, if the contents of the library may have been changed from outside of TPS (e.g. by a text editor) since TPS was started, then this command will re-initialize the library index.

RETRIEVE-FILE *file*
Make all objects in a library file available in TPS. Objects in a file are retrieved in the same order as they are stored in the file.

8.4. Editing

ADD-KEYWORD *keyword defn*
Add a keyword to the keywords.rec file in your default directory. This must be done before the keyword can be used anywhere else in the library.

CHANGE-KEYWORDS *name*
Change the keywords attribute of a stored library object. NOTE: not all keywords can be changed. TPS may modify your list of keywords -- for example, if you specify FIRST-ORDER for a problem that is higher-order, TPS will change it.

CHANGE-PROVABILITY *name*
Change the PROVABILITY attribute of a stored gwff.

COPY-LIBFILE *oldfile newfile*
Copy a file of library objects. The source file will be found among the directories in DEFAULT-LIB-DIR and BACKUP-LIB-DIR (the user will be prompted if more than one such file exists, and also if there is a choice of directories for the new file). Needed objects are not copied.

COPY-LIBOBJECT *name type filename*
Copy an object from some specified directory to the default directory. Does not copy the needed objects.

DELETE *names type*
Delete an object from the library.

FIX-MODES Change all references to obsolete flags into the appropriate new flag setting, for every mode in your library directory. You only need to do this once. You will be prompted before anything is changed, and you should probably keep a backup copy of your old library in case disaster strikes! THE CODE FOR THIS COMMAND SHOULD BE REWRITTEN FOR EACH RELEVANT CHANGE TO THE TPS FLAGS. At the minute, it's set up to remove references to REWRITE-DEFNS-EAGER, REWRITE-EQUAL-EXT and REWRITE-ONLY-EXT, which have been removed, and to reset REWRITE-DEFNS and REWRITE-EQUALITIES to appropriate values. It also puts LAST-MODE-NAME at the head of all settings for RECORDFLAGS.

INSERT *name type* Insert an item in the library. The INSERT command can be used to modify existing entries in the library. All the items will be replaced by whatever you write (or kept the same if you use the default) except for "additional remarks"; what you specify here will be added to whatever is already there. If you don't want to add additional remarks, respond with <space><return>. Use your favorite editor to make any changes within the existing comment.

MOVE-LIBFILE *oldfile newfile*
Move a file of library objects. The source file will be found among the directories in DEFAULT-LIB-DIR and BACKUP-LIB-DIR (the user will be prompted if more than one such file exists, and also if there is a choice of directories for the new file). Needed objects are not moved.

MOVE-LIBOBJECT *name type filename*
Move an object from one file to another within the same directory. This command will also move a list of objects (either all of the same type, or all of type NIL, into a single named file in the same directory.

REFORMAT *file* Reformat the specified file. Will attempt to load all the objects in a given file and then to rewrite that file in the standard library format. This can be useful if you manually edit your library files a lot and they've started to look a little disorganised. To reformat all files in your directories, use SPRING-CLEAN.

REINDEX *file reformat*
Reindex and reformat the specified file --- i.e. reconstruct the entries in the library master index relating to the objects in a particular file (you should only need this if you've been manually editing the libindex.rec file and have accidentally lost some entries...), and then attempt to load and rewrite the file. To reindex all files in your directories, use SPRING-CLEAN. If you get an error because of parsing problems, try again but answer no to "Reformat?" (it is not possible to format a file without parsing it).

RENAME-OBJECT *name type newname*
Change the name of a library object. Does not move the object or alter it in any other way.

SHOW-KEYWORDS
List all of the current acceptable keywords for the library.

SORT *file head* Sort the specified file into alphabetical order, except for the given list of objects which are put at the head of the file (if they were originally in the file). This command reads in the entire file and then rewrites it; it will incidentally also catch any parsing errors.

SPRING-CLEAN *reindex reformat sort delete*
Will do its best to reindex, reformat and/or sort every file in the default library directory. If your files are a real mess, you might consider using emacs to get rid of the worst of the problems before using SPRING-CLEAN. It will also delete any file in the directory that doesn't belong there Generally this means everything except .lib and libindex.rec files; you will be asked for confirmation before each file is deleted. If you get an error because of parsing problems, try again but answer no to "Reformat?" and "Sort?" (it is not possible to reformat or sort a file that cannot be parsed). Better yet, delete the unparseable entry and try again.

UPDATE-KEYWORDS
For each library entry, update the keywords field to include all of those keywords that can be determined automatically. Any other keywords will be left untouched. If you answer NO to the question about checking existing keywords, then this command will just attempt to fill in keywords for those objects which have none. If you answer YES, keywords will be generated for all of the objects (but existing user-defined keywords will not be overwritten).

This command will almost certainly crash if it discovers any untypable definitions, missing needed-objects, circular definitions, misprints, etc... in your library. This probably won't damage your library, but you might want to make a backup of all your files before you call this, just in case...

9. Review Commands

The internal name of this category is REVIEWCMD. A review command can be defined using DEFREVIEW. Allowable properties are: ARGTYPES, ARGNAMES, ARGHELP, DEFAULTFNS, MAINFNS, CLOSEFNS, MHELP.

9.1. Top Levels

LEAVE Leave REVIEW to the next enclosing top level.

9.2. Flags

CHANGED-FLAGS *omit*
List all those flags whose current value is not the default value.

DESCRIBE *flag* Describe a flag.

DESCRIBE* *subjectlist*
List all flags under the subjects requested alongwith their descriptions.

KEY *phrase subjectlist search-names*
Look for a key phrase in the help strings (or just the names) of flags of given subjects. See also SEARCH, at the main top level.

LIST *subjectlist* List all flags in the given subjects with their current value.

SET *flag flag-value* Directly set the value of a flag.

SETFLAG *flag* Set the value of a flag after examining it.

SETFLAGS1 *fvlist* Simultaneously sets multiple flags of the form ((FLAG1 . VALUE1) (FLAG2 . VALUE2)...) (the dots may be omitted); intended for use when cutting and pasting records from library or bug files. The opening and closing parentheses must be supplied.

SETFLAGS2 *whole*
Simultaneously sets multiple flags of the form "FLAG1: VALUE1 FLAG2: VALUE2 ...". Intended for use when cutting and pasting records from library or bug files. User must provide double quotes before and after pasting the record, and each flag and value pair should be separated by a newline. Flag-names containing double quotes must be set separately. This command cannot handle such cases.

SUBJECTS Print a list of currently defined subjects for REVIEW.

UPDATE *subjectlist* Update all the flags concerning the given subjects. ! will leave the remaining flags unchanged.

9.3. Modes

ADD-FLAG-TO-MODE *mode flag*
Add a flag to a mode. The flag will be added with its current setting. If the flag is already present, its value in the mode will be changed to its current setting.

COMPARE-MODES *mode1 mode2*
Compare two different modes; print a list of the values on which they differ.

COPY-MODE *oldname newname*
Make a copy of a mode, with a new name. To delete the old mode from memory, use DESTROY.

MODE *mode* Set a group of flags by switching to a mode.

REMOVE-FLAG-FROM-MODE *mode flag*
Delete a flag from a mode. If the flag is not present in the mode, this command will do nothing.

9.4. Unclassified

UNIF-DEPTHS Turn off all the MAX-SUBSTS checking in unification, and use only the flags MAX-SEARCH-DEPTH, MAX-UTREE-DEPTH and MIN-QUICK-DEPTH.

UNIF-NODEPTHS Turn off all the depth checking in unification, and set the MAX-SUBSTS-VAR and MAX-SUBSTS-QUICK flags.

10. Flag Or Parameters

The internal name of this category is FLAG. A flag or parameter can be defined using DEFFLAG%. Allowable properties are: FLAGTYPE, DEFAULT, CHANGE-FN, SUBJECTS, MHELP.

10.1. Top Levels

MT-DUPS-PER-QUANT
: The maximum number of times that each individual quantifier may be duplicated in the MATINGSTREE search procedures. This flag is overridden by NUM-OF-DUPS, which governs the maximum total number of duplications of all quantifiers in the matingstree search.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects ETREES, MTREE-TOP. The default value is INFINITY.

PROOFW-ACTIVE
: If T, active lines of the current proof are printed in the Current Subproof window, if this window exists.It takes values of type BOOLEAN and belongs to subjects PRINTING, OTL-VARS. The default value is T.

PROOFW-ACTIVE+NOS
: If T, active lines of the current proof are printed in the Current Subproof & Line Numbers window, if this window exists.It takes values of type BOOLEAN and belongs to subjects PRINTING, OTL-VARS. The default value is T.

PROOFW-ACTIVE+NOS-HEIGHT
: Controls the initial height of the Current Subproof & Line Numbers window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 24.

PROOFW-ACTIVE+NOS-WIDTH
: Controls the initial width of the Current Subproof & Line Numbers window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 80.

PROOFW-ACTIVE-HEIGHT
: Controls the initial height of the Current Subproof window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 24.

PROOFW-ACTIVE-WIDTH
: Controls the initial width of the Current Subproof window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 80.

PROOFW-ALL
: If T, entire proof so far is printed in the Complete Proof window, if this window exists.It takes values of type BOOLEAN and belongs to subjects PRINTING, OTL-VARS. The default value is T.

PROOFW-ALL-HEIGHT
: Controls the initial height of the Complete Proof window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 24.

PROOFW-ALL-WIDTH
: Controls the initial width of the Complete Proof window.It takes values of type POSINTEGER and belongs to subjects PRINTING, OTL-VARS. The default value is 80.

10.2. Style

STYLE
: The style of the terminal output device.It takes values of type DEV-STYLE and belongs to subjects PRINTING. The default value is GENERIC.

10.3. Review

ALPHA-LOWER-FLAG
: If T, output from ? will be made more readable (alphabetised, smaller left margin, mostly lower case) If NIL, output is in the old style (non-alphabetised, large left margin, mostly block capitals).It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

LAST-MODE-NAME
: LAST-MODE-NAME contains the name of the last MODE used. There is no point in the user's altering its value, since TPS only ever writes to it, and never reads from it.It takes values of type STRING and belongs to subjects MATING-SEARCH. The default value is "".

10.4. Modes

SUPPRESS-FLAGS

If T, will suppress the printing of any flags in SUPPRESS-FLAGS-LIST by the HELP MODE, COMPARE-MODES, LIST, DESCRIBE*, UPDATE and CHANGED-FLAGS commandsIt takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

SUPPRESS-FLAGS-LIST

If SUPPRESS-FLAGS is T, these flags will not be printed. SUPPRESS-FLAGS-LIST itself is always suppressed, because it's very large.It takes values of type TPSFLAGLIST and belongs to subjects PRINTING. The default value is ().

10.5. Help

SHOW-ALL-PACKAGES

Determines whether ENVIRONMENT will show symbols in all packages or merely accessible symbols.It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is NIL.

10.6. Starting and Finishing

COMPLETION-OPTIONS

If T, then the user will be offered a choice between multiple completions of a command. Also, the commands offered will come from the current top level, the main top level and the flags. If NIL, command completion will try first the current top level, then the main top level, and then the flags, and will fail if the first of these which contains any completions also contains multiple completions. For example (when T) <1>displ&

3 matching commands or flags have been found. 1) DISPLAYFILE 2) DISPLAY-TIME 3) DISPLAYWFF 4) None of these. Input a number between 1 and 4: [1]>

(when NIL) <2>displ& TPS error while reading. Multiple completions for DISPL: DISPLAYFILE DISPLAY-TIME It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is T.

HISTORY-SIZE Maximum number of commands to save. If NIL, all commands will be saved.It takes values of type NULL-OR-POSINTEGER and belongs to subjects MAINTAIN, OTL-VARS. The default value is 25.

10.7. OTL Object

AUTO-GENERATE-HYPS

If T, hypotheses for lines computed and filled in automatically, if NIL, the user will be asked for confirmation for each set of hypotheses.It takes values of type BOOLEAN and belongs to subjects OUTLINE. The default value is T.

CLEANUP-RULEC

If T, cleanup-same works on lines with multiple-line justifications.It takes values of type BOOLEAN and belongs to subjects OTL-VARS. The default value is T.

CLEANUP-SAME If NIL, identical lines are not replaced when doing CLEANUP.It takes values of type BOOLEAN and belongs to subjects OTL-VARS. The default value is T.

DEFAULT-WFFEQ

The name of the functions which checks for equality of wffs.It takes values of type SYMBOL and belongs to subjects OUTLINE. The default value is WFFEQ-AB.

PRINT-DOTS If nil, ... are not printed before a plan line.It takes values of type BOOLEAN and belongs to subjects OTL-VARS. The default value is T.

PRINTLINEFLAG If nil, lines in the proof outline are not printed.It takes values of type BOOLEAN and belongs to subjects OTL-VARS. The default value is T.

SHORT-HELP If T, only the rule specification will be shown when asking for help on a rule, and the command format of a command will not be shown.It takes values of type BOOLEAN and belongs to subjects OTL-VARS. The default value is NIL.

10.8. Printing

PRINT-COMBINED-EGENS
: When set to t, the commands PBRIEF and EXPLAIN will combine lines which are a sequence of existential generalizations and print a single line.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

PRINT-COMBINED-UGENS
: When set to t, the commands PBRIEF and EXPLAIN will combine lines which are a sequence of universal generalizations and print a single line.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

PRINT-COMBINED-UIS
: When set to t, the commands PBRIEF and EXPLAIN will combine lines which are a sequence of universal instantiations and print a single line.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

PRINT-UNTIL-UI-OR-EGEN
: When set to t, the commands PBRIEF and EXPLAIN will continue to print beyond the depth specified until a line justified by UI or Egen is encountered. The intuition is that these are the real choice points in the proof. When set to nil, PBRIEF and EXPLAIN print only to the depth specified.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

10.9. Printing

ALLSCOPEFLAG
: If T, all brackets will be printed; no implicit scoping is assumed.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

ATOMVALFLAG
: If T, the name of every atom will be printed below its value.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

BLANK-LINES-INSERTED
: Number of blank lines printed in the proofwindows between different stages of each proof.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 24.

CHARSIZE
: Should be one of MIN, MED or MAX. Determines the size of characters used by Proofwindows and Editor Windows. Currently, MIN and MED are the same size.It takes values of type SYMBOL and belongs to subjects PRINTING, EDITOR. The default value is MED.

DISPLAYWFF
: If T, formulas are printed on separate lines.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

ELIM-DEFNS
: When printing a wff, first instantiate all of the definitions and lambda-normalize. This instantiation will ignore REWRITE-DEFNS, but will use the current setting of REWRITE-EQUALITIES. It's best to leave this at NIL (i.e. off), since output with it set to T can be confusing.It takes values of type BOOLEAN and belongs to subjects LIBRARY, PRINTING. The default value is NIL.

FILLINEFLAG
: If NIL, every argument of an associative infix operator will have a separate line.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

FIRST-ORDER-PRINT-MODE
: If T, formulas are printed so they can be parsed when FIRST-ORDER-MODE-PARSE is set to T.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

FLUSHLEFTFLAG
: If T, no line of a pretty-printed formula will be indented.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

LEFTMARGIN
: The global left margin of the terminal in characters.It takes values of type INTEGER+ and belongs to subjects PRINTING. The default value is 0.

LOCALLEFTFLAG
: If T, arguments of infix operators start in the same column as the operator.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

PPWFFLAG
: If T, formulas will generally be pretty-printed (except for the editor).It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

PRINTDEPTH
: If 0, all printing will be done to arbitrary recursive depth, if n > 0 subformulas of depth n will be replaced by '&'.It takes values of type INTEGER+ and belongs to subjects PRINTING. The default value is 0.

PRINTTYPES
: If NIL, type symbols will never be printed.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

PRINTTYPES-ALL
: This flag only applies when the flag PRINTTYPES is T. If PRINTTYPES-ALL is NIL, type symbols will be printed only on the first occurrence of a variable name. If it is T, type symbols will be printed on every occurrence of a variable name.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `NIL`.

RETAIN-INITIAL-TYPE
: If T, type property is inherited from the previous occurrence (if any) of the logical symbols. Else, it is modified whenever the parser encounters a fresh occurrence.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

RIGHTMARGIN
: The global right margin of the terminal in characters.It takes values of type INTEGER+ and belongs to subjects `PRINTING`. The default value is `79`.

SCOPE
: If T, all wffs will be enclosed in square brackets.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `NIL`.

SLIDES-PREAMBLE
: The preamble that is printed into the first lines of all the Scribe slides files produced by TPS. See also SCRIBE-PREAMBLE.It takes values of type STRING and belongs to subjects `PRINTING`. The default value is `""`.

USE-DOT
: If T, formulas are printed using Church's dot notation. If NIL, only brackets will be used.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

USE-INTERNAL-PRINT-MODE
: If T, the internally-defined modes SCRIBE-OTL, TEX-OTL and TEX-1-OTL will be used for printing Scribe and TeX output. (See the help message for TEX-MIMIC-SCRIBE for help on the difference between the last two.) These are usually good enough, but if you want to use a custom-defined flag setting, then set this flag to NIL to override the internal modes.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

10.10. Internal for Printing

INFIX-NOTATION
: If T, infix notation can be used for connectives and abbreviations which have an INFIX property. If NIL, infix notation is disallowed. (Note: If you set this to NIL, library objects saved with infix notation will become unreadable.)It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

10.11. TeX

IN-TEX-MATH-MODE
: If T, $'s will not be printed around wffs in style TeX.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `NIL`.

LATEX-EMULATION
: If T, all of the printing commands that produce TeX output will produce output suitable for LaTeX instead. See LATEX-PREAMBLE, LATEX-POSTAMBLE.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `NIL`.

PAGELENGTH
: Number of lines on an output page. Used by printing routines to determine where to break output.It takes values of type POSINTEGER and belongs to subjects `PRINTING`. The default value is `55`.

TEX-MIMIC-SCRIBE
: If T, TEXPROOF will give a good-looking tex output. If NIL, TEXPROOF cannot break formulas in terms of the connectives in it. So the output is a little bit ugly. Change the flag into NIL only when you cannot get a good-looking output by setting it to T.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

10.12. X Windows

USE-WINDOW-STYLE
: If T, uses the style given by WINDOW-STYLE for output to windows other than the main one. If NIL, windows will all be in the style given by STYLE.It takes values of type BOOLEAN and belongs to subjects `PRINTING`. The default value is `T`.

WINDOW-STYLE
: The style of output that will be used in all the windows besides the main one, if USE-WINDOW-STYLE is T. Ignored if USE-WINDOW-STYLE is NIL.It takes values of type DEV-STYLE and belongs to subjects

PRINTING. The default value is XTERM.

10.13. Weak Labels

PRINT-WEAK If T, weak labels are printed, otherwise they wff the represent will be printed.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is T.

10.14. Flavors of Labels

MAKE-WFFOPS-LABELS
If T, meta labels are created by the parser, if NIL, wffops are evaluated at parse-time.It takes values of type BOOLEAN and belongs to subjects PARSING. The default value is NIL.

META-LABEL-NAME
The prefix for names of meta labels (from wffops).It takes values of type SYMBOL and belongs to subjects INTERNAL-NAMES. The default value is ML.

PRINT-META If T, meta labels are printed, otherwise the wffop they represent will be printed.It takes values of type BOOLEAN and belongs to subjects PRINTING. The default value is NIL.

10.15. Saving Work

SAVE-INTERVAL Interval of file-write of saved commands.It takes values of type INTEGER+ and belongs to subjects SAVING-WORK. The default value is 5.

SAVE-WORK-ON-START-UP
If T, work is saved automatically whenever TPS3 is started. It takes values of type BOOLEAN and belongs to subjects SAVING-WORK. The default value is NIL.

SAVE-WORK-P If T, work is saved automatically.It takes values of type BOOLEAN and belongs to subjects SAVING-WORK. The default value is T.

10.16. Recording

PRINTEDTFILE The name of the file in which wffs are recorded.It takes values of type FILESPEC and belongs to subjects PRINTING, EDITOR. The default value is "edt.mss".

PRINTEDTFLAG If T, editor operations are recorded into open transcript files.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is NIL.

PRINTEDTFLAG-SLIDES
If T, editor operations are recorded in slides style. This flag has no effect unless PRINTEDTFLAG is T.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is NIL.

PRINTEDTOPS The function or name of the function which test whether the result of a particular edop should be written to a file.It takes values of type ANYTHING and belongs to subjects PRINTING, EDITOR. The default value is ALWAYS-TRUE.

PRINTMATEFILE The name of the file in which mateops are recorded. This has not yet been implemented, although one can record remarks (only) into the file.It takes values of type FILESPEC and belongs to subjects PRINTING, MATING-SEARCH. The default value is "mate.mss".

PRINTMATEFLAG
If T, mating-search operations are recorded into open transcript files. Not currently implemented.It takes values of type BOOLEAN and belongs to subjects PRINTING, MATING-SEARCH. The default value is NIL.

PRINTMATEFLAG-SLIDES
If T, mating-search operations are recorded in slides style. This flag has no effect unless PRINTMATEFLAG is T. (In fact, it has no effect even if PRINTMATEFLAG is T, since it hasn't been implemented.)It takes values of type BOOLEAN and belongs to subjects PRINTING, MATING-SEARCH. The default value is NIL.

PRINTMATEOPS The function or name of the function which test whether the result of a particular mateop should be written to a file. This has not been implemented.It takes values of type ANYTHING and belongs to subjects PRINTING, MATING-SEARCH. The default value is ALWAYS-TRUE.

10.17. Printing Proofs into Files

LATEX-POSTAMBLE
: The standard way in which TPS will end a TeX file when LATEX-EMULATION is T.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

LATEX-PREAMBLE
: The preamble that is printed into the beginning of all TeX files produced by TPS when LATEX-EMULATION is T.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

SCRIBE-LINE-WIDTH
: Width of a proofline in characters.It takes values of type INTEGER+ and belongs to subjects OTL-VARS. The default value is 75.

SCRIBE-POSTAMBLE
: The postamble that is printed into all Scribe files immediately before they are closed by TPS. See SCRIBE-PREAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

SCRIBE-PREAMBLE
: The preamble that is printed into the first lines of all the Scribe files produced by TPS, except those that are in SLIDES style. See also SLIDES-PREAMBLE, TEX-PREAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

TEX-1-POSTAMBLE
: Another TeX postamble, used when TEX-MIMIC-SCRIBE is T. See TEX-POSTAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

TEX-1-PREAMBLE
: Another TeX preamble, used when TEX-MIMIC-SCRIBE is T. See TEX-PREAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

TEX-LINE-WIDTH
: width of a proofline in characters.It takes values of type INTEGER+ and belongs to subjects OTL-VARS. The default value is 75.

TEX-POSTAMBLE
: The standard way in which TPS will end a TeX file. See TEX-PREAMBLE, TEX-1-POSTAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

TEX-PREAMBLE The preamble that is printed into the beginning of all TeX files produced by TPS. See also VPFORM-TEX-PREAMBLE, TEX-1-PREAMBLE, TEX-POSTAMBLE.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

TPSTEX The pathname of the tps.tex file on your system. Should be initialised by the tps3.ini file.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

VPDTEX The pathname of the vpd.tex file on your system. Should be initialised by the tps3.ini file.It takes values of type STRING and belongs to subjects PRINTING. The default value is "".

10.18. Proof Outline

PRINT-COMMENTS
: If T, print the comments attached to lines and proofs. See LINE-COMMENT and PROOF-COMMENT.It takes values of type BOOLEAN and belongs to subjects OUTLINE, PRINTING. The default value is T.

SLIDES-TURNSTILE-INDENT
: Number of columns (from leftmargin) that turnstile should be indented when making slides. Compare TURNSTILE-INDENT. This flag and SLIDES-TURNSTYLE-INDENT are synonymous.It takes values of type INTEGER+ and belongs to subjects OTL-VARS. The default value is 4.

SLIDES-TURNSTYLE-INDENT
: Number of columns (from leftmargin) that turnstile should be indented when making slides. Compare TURNSTYLE-INDENT. This flag and SLIDES-TURNSTILE-INDENT are synonymous.It takes values of type INTEGER+ and belongs to subjects OTL-VARS. The default value is 4.

SUPPORT-NUMBERS
: This has three possible settings: GAP: new support lines will be put in the gap between the current planned line and the previous line, whatever it is. PLAN: new support lines will be put immediately after the previous (lower-numbered) planned line, if there is one (and as for NIL if there isn't). NIL (or anything else): new support lines will be put in whatever seems to be a sensible place.

This flag may well be useless (although non-NIL values will force it to do the right thing, TPS will probably do the right thing anyway).It takes values of type SYMBOL and belongs to subjects OUTLINE, OTL-VARS. The default value is

NIL.

TURNSTILE-INDENT
Number of columns (from leftmargin) that turnstile should be indented when writing proofs in a SCRIBE file. Notice that slides use a different flag, SLIDES-TURNSTILE-INDENT. This flag and TURNSTYLE-INDENT are synonymous.It takes values of type INTEGER+ and belongs to subjects OTL-VARS, PRINTING. The default value is 13.

TURNSTILE-INDENT-AUTO
Decides how turnstiles are printed in proofs. This flag works in all styles other than TEX; in particular, it works in XTERM, GENERIC, SCRIBE and SLIDES styles. There are four possible settings: FIX : put the turnstile in the column indicated by TURNSTYLE-INDENT (or SLIDES-TURNSTYLE-INDENT, in style SLIDES). MIN : print the turnstile as far to the left as possible while still having it in the same column on every line. (If this puts it off the right margin, then this will default to the same behaviour as FIX.) COMPRESS : similar to VARY, but also removes spaces at other points in the proof (e.g. around dots, and between line numbers and hypotheses). VARY : print the turnstile one space after the hypotheses in each line (so it will move from line to line).It takes values of type INDENTATION and belongs to subjects OTL-VARS, PRINTING. The default value is MIN.

TURNSTYLE-INDENT
Number of columns (from leftmargin) that turnstile should be indented when writing proofs in a SCRIBE file or on the screen. Notice that slides use a different flag, SLIDES-TURNSTYLE-INDENT. This flag and TURNSTILE-INDENT are synonymous.It takes values of type INTEGER+ and belongs to subjects PRINTING, OTL-VARS. The default value is 13.

TURNSTYLE-INDENT-AUTO
Decides how turnstiles are printed in proofs. This flag works in all styles other than TEX; in particular, it works in XTERM, GENERIC, SCRIBE and SLIDES styles. There are four possible settings: FIX : put the turnstile in the column indicated by TURNSTYLE-INDENT (or SLIDES-TURNSTYLE-INDENT, in style SLIDES). MIN : print the turnstile as far to the left as possible while still having it in the same column on every line. (If this puts it off the right margin, then this will default to the same behaviour as FIX.) COMPRESS : similar to VARY, but also removes spaces at other points in the proof (e.g. around dots, and between line numbers and hypotheses). VARY : print the turnstile one space after the hypotheses in each line (so it will move from line to line).It takes values of type INDENTATION and belongs to subjects PRINTING, OTL-VARS. The default value is MIN.

10.19. Expansion Trees

ADD-TRUTH When set to IF-NEEDED, tests whether the etree has any path of length 1; if it does, then adds a conjunct TRUTH to the vpform. When set to T, it will always add this conjunct. When set to NIL, it will never add this conjunct. (When TRUTHVALUES-HACK is NIL, it will also add a conjunct NOT FALSEHOOD).It takes values of type SYMBOL and belongs to subjects MATING-SEARCH, ETREES. The default value is IF-NEEDED.

DUPLICATION-STRATEGY
The name of a duplication strategy. Currently, either DUP-ALL or DUP-OUTER. Only applies to MS88.It takes values of type SYMBOL and belongs to subjects MS88, MATING-SEARCH. The default value is DUP-OUTER.

DUPLICATION-STRATEGY-PFD
The name of a duplication strategy for path-focused procedures. It may have either of two values: DUP-INNER and DUP-OUTER. DUP-INNER means inner quantifiers get duplicated before outer ones, while DUP-OUTER means vice versa.It takes values of type SYMBOL and belongs to subjects MS93-1, MS92-9, MS91-7, MS90-9, MS90-3, MATING-SEARCH. The default value is DUP-INNER.

ECONJ-NAME Prefix for labels associated with conjunction nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is CONJ.

EDISJ-NAME Prefix for labels associated with disjunction nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is DISJ.

EMPTY-DUP-INFO-NAME
Prefix for labels associated with empty-dup-info nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is EMP.

EPROOF-NAME Prefix for names of expansion proofs.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is EPR.

EXPANSION-NAME
Prefix for labels associated with expansion nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is EXP.

FALSE-NAME Prefix for labels associated with FALSEHOOD nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is FALSE.

IMP-NAME Prefix for labels associated with implication nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is IMP.

INITIAL-BKTRACK-LIMIT
Initial backtrack limit. If a mating exceeds this limit, a new mating will be started, and the limit incremented. If the value of the flag is set to INFINITY, then this will never happen.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is INFINITY.

LEAF-NAME Prefix for labels associated with leaf nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is LEAF.

MATING-NAME Prefix for names of matings.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is MAT.

MATINGSTREE-NAME
Prefix for labels associated with nodes in a matingstree.It takes values of type SYMBOL and belongs to subjects ETREES, MTREE-TOP. The default value is MSTREE.

MAX-DUP-PATHS
Any universal jform which has more than MAX-DUP-PATHS paths below it cannot get duplicated during search process.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects MS93-1, MS92-9, MS88, MS89, MS91-6, MS91-7, MS90-9, MS90-3, MATING-SEARCH. The default value is INFINITY.

MIN-QUANTIFIER-SCOPE
When this flag is T, the scope of quantifiers is minimized before starting expansion proofs. If an eproof is found with this flag set to T, during the translation of the eproof to an ND proof RULEQ is called to fill the gap between the theorem as originally stated and its min-quantifier-scope version.It takes values of type BOOLEAN and belongs to subjects MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH, ETREES. The default value is NIL.

NEG-NAME Prefix for labels associated with negation nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is NEG.

PRINT-DEEP T will print the deep formula of an expansion or selection node, NIL will print the shallow formula, both only if PRINT-NODENAMES is NIL.It takes values of type BOOLEAN and belongs to subjects ETREES, PRINTING. The default value is T.

PRINT-NODENAMES
T will print the names of expansion and selection nodes, NIL will print either the deep or shallow formula of the node. (see the flag PRINT-DEEP).It takes values of type BOOLEAN and belongs to subjects ETREES, PRINTING. The default value is T.

RENUMBER-LEAVES
If this flag is T, copies of leafN will be numbered leafN.1, leafN.2, etc. If the flag is NIL, they will be given the next available number, as determined by an internal counter.It takes values of type BOOLEAN and belongs to subjects JFORMS. The default value is T.

REWRITE-DEFNS A list whose first element is one of NONE, EAGER, LAZY1 and DUAL, and whose other (optional) elements are lists whose first element is one of these four options and whose other elements are the names of definitions. The first element is the default behaviour for rewriting definitions, and the other lists are lists of exceptions to this default, with a different behaviour specified. NONE: do not rewrite this definition at all. EAGER: rewrite all of these definitions, in one big step, as soon as possible. LAZY1: rewrite these, one step at a time, when there are no more EAGER rewrites to do. DUAL: as LAZY1, but rewrite these abbreviations A to a conjunction of A and A, and then deepen only one of these conjuncts. (e.g. TRANSITIVE p becomes TRANSITIVE p AND FORALL x y z . [pxy AND pyz] IMPLIES pxz LAZY2: synonym for DUAL.

For example: the value (EAGER) would be interpreted as "Rewrite every definition in one step."

(DUAL (EAGER TRANSITIVE) (NONE INJECTIVE SURJECTIVE)) would be interpreted as "Rewrite TRANSITIVE whenever it appears. Don’t ever rewrite INJECTIVE or SURJECTIVE. Rewrite every other definition in the DUAL way."It takes values of type REWRITE-DEFNS-LIST and belongs to subjects MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, IMPORTANT, MATING-SEARCH. The default value is (EAGER).

REWRITE-NAME Prefix for labels associated with rewrite nodes.It takes values of type SYMBOL and belongs to subjects ETREES. The default value is REW.

SELECTION-NAME
Prefix for labels associated with selection nodes (in a non-skolem etree).It takes values of type SYMBOL and belongs to subjects ETREES. The default value is SEL.

SHOW-SKOLEM When true, skolem terms are shown when a wff containing them is printed, otherwise a parameter is printed instead.It takes values of type BOOLEAN and belongs to subjects The default value is NIL.

SKOLEM-DEFAULT

Default method for skolemizing, in which wffs of the form EXISTS y . M are replaced by M(g(...)). There are three possible ways to do this: SK1 is the original method due to Skolem, where the Skolem constants g take as arguments all the x such that FORALL x occurs in the wff and EXISTS y . M is in its scope. SK3 is the method in which the arguments of g are the free variables of EXISTS y . M. NIL means don't Skolemize at all; use selection nodes instead.It takes values of type SYMBOL and belongs to subjects `MS98-1`, `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `MATING-SEARCH`. The default value is `SK1`.

SKOLEM-SELECTION-NAME
Prefix for labels associated with selection nodes (in a skolem etree).It takes values of type SYMBOL and belongs to subjects `ETREES`. The default value is `SKOL`.

TOTAL-NUM-OF-DUPS
Max number of duplications allowed at any time during a search using path-focused duplication. Compare NUM-OF-DUPS. This flag will be ignored if set to NIL. THE IMPLEMENTATION OF THIS IS BUGGY; setting it to NIL is safest.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS90-3`, `MATING-SEARCH`, `IMPORTANT`. The default value is `NIL`.

TRUE-NAME Prefix for labels associated with TRUTH nodes.It takes values of type SYMBOL and belongs to subjects `ETREES`. The default value is `TRUE`.

TRUTHVALUES-HACK
When this flag is T, leaves of truthvalues will not deepened into an empty disjunction or an empty conjunction. this allows us to deal with truthvalues in formulas, especially, higher-order formulas. In order to deal with truthvalues in definitions, such as NULLSET, the definitions containing falsehood should be rewritten. Please put new definitions containing falsehood into truthvalues-hack-updatelist so that they can be rewritten appropriately.It takes values of type BOOLEAN and belongs to subjects `MATING-SEARCH`, `ETREES`. The default value is `NIL`.

10.20. Mtree Operations

DEFAULT-OB If DEEPEST, the default next obligation is found by depth-first search of the obtree, if HIGHEST it is found by breadth-first-search, if D-SMALLEST then the deepest of the set of smallest obligations (i.e. the set of all obligations with the fewest possible literals) is chosen, if H-SMALLEST then the highest of this set is chosen.It takes values of type OBDEFAULT and belongs to subjects `ETREES`, `MTREE-TOP`. The default value is `D-SMALLEST`.

MT-DEFAULT-OB-MATE
Determines how ADD-CONN chooses the default obligation for the second literal of the given pair (it is possible that this literal will occur several times on the path, in several different obligations). Options are: LOWEST : Chooses the obligation which lies lowest (i.e. furthest from the root) HIGHEST : Chooses the obligation nearest to the root (but not the root). HI-LO : Finds the obligation which occurs lowest; this obligation was first added at some point in the matingstree. Then chooses the highest obligation which was added at the same point in the matingstree.It takes values of type SYMBOL and belongs to subjects `MTREE-TOP`. The default value is `LOWEST`.

10.21. Mtree Auto

MT-SUBSUMPTION-CHECK
If SAME-CONNS or T, will check whether the node about to be added is duplicated elsewhere in the tree, and will reject it if it is. (This will use the SAME-TAG function described below, and then do a more thorough check if the tags match.)

If SUBSET-CONNS, will check whether the connections at the node about to be added are a subset of those at some other node. (This is only really useful in MT94-11, where all possible new nodes are added, breadth-first, to the tree. It is probably too restrictive for the other mtree searches.)

If SAME-TAG will check whether the tag (an integer generated from the list of connections) is the same as any other existing tag, and will reject it if it is. See TAG-CONN-FN and TAG-LIST-FN. (Note that most tag functions can produce the same tag for different matings, so this may reject connections unneccessarily.)

If NIL, will turn off subsumption checking altogether.It takes values of type MT-SUBSUMPTION and belongs to subjects `MTREE-TOP`. The default value is `SAME-CONNS`.

MT94-12-TRIGGER
If the current obligation contains fewer than MT94-12-TRIGGER literals, MT94-12 will behave in the same way as MT94-11 If it contains MT94-12-TRIGGER or more, MT94-12 will choose a literal with as few mates as possible. There are two extrema: infinity means that the least branch will only be chosen if the obligation is as big as the initial obligation; 0 means that the least branch will always be chosen.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `ETREES`, `MTREE-TOP`. The default value is `INFINITY`.

MTREE-FILTER-DUPS

If T, will not add the same link to a mating twice on the same branch of a matingstree during automatic search. If NIL, will add it as many times as it wants to.It takes values of type BOOLEAN and belongs to subjects `ETREES`, `MTREE-TOP`. The default value is `T`.

MTREE-STOP-IMMEDIATELY

If T, will stop an automatic search as soon as a closed node is found. If NIL, will continue to generate whatever level of the tree it was working on, and will check for closed nodes when it finishes.It takes values of type BOOLEAN and belongs to subjects `ETREES`, `MTREE-TOP`. The default value is `T`.

TAG-CONN-FN Determines how the tag (a number attached to each mating) is calculated. Should be the name of a function which, given a connection, will generate an integer from it. See MT-SUBSUMPTION-CHECK and TAG-MATING-FN.

Current settings are TAG-CONN-QUICK, which uses TPS's internal number for the connection. (Actually, it uses (1 + this number), so as to avoid multiplying by one.) TAG-CONN-LEAFNO, which multiplies the integer parts of the two leaf names in the given connection.It takes values of type SYMBOL and belongs to subjects `MTREE-TOP`. The default value is `TAG-CONN-LEAFNO`.

TAG-MATING-FN Determines how the tags for each connection are combined to produce a tag for the entire mating. Should be the name of a function which, given two integers, will generate a third integer. See MT-SUBSUMPTION-CHECK and TAG-MATING-FN.

Current settings are MULTIPLY-TAG-LIST, which simply multiplies the numbers together.It takes values of type SYMBOL and belongs to subjects `MTREE-TOP`. The default value is `MULTIPLY-TAG-LIST`.

10.22. Mating search

DEFAULT-EXPAND

Used with DEFAULT-MATE to determine a setting for DEFAULT-MS. Combinations marked N/A will result in DEFAULT-MS being set to NIL. Notice that for otree and oset searches, the actual primsubs generated will depend on the setting of PRIMSUB-METHOD. Takes values: none, ms98-1, otree and oset. The value MS98-1 is an exceptional setting used for both this flag and DEFAULT-MATE to denote the MS98-1 procedure. Changes DEFAULT-MS as follows: DEFAULT-EXPAND: | NONE | OTREE | OSET | ==========+=========+=========+========+ DEFAULT-MATE: NPFD | MS88 | MS89 | MS91-6 | ----------+---------+---------+--------+ NPFD-1 | MS92-9 | MS93-1 | N/A | ----------+---------+---------+--------+ PFD | MS90-3 | MS90-9 | MS91-7 | ----------+---------+---------+--------+ MTREE | MT94-11 | N/A | N/A | ----------+---------+---------+--------+ MTREE-1 | MT94-12 | N/A | N/A | ----------+---------+---------+--------+ MTREE-2 | MT95-1 | N/A | N/A | ----------+---------+---------+--------+It takes values of type SYMBOL and belongs to subjects `MTREE-TOP`, `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `MATING-SEARCH`. The default value is `OTREE`.

DEFAULT-MATE Used with DEFAULT-EXPAND to determine a setting for DEFAULT-MS. Combinations marked N/A will result in DEFAULT-MS being set to NIL. (Notice that for otree and oset searches, the actual primsubs generated will depend on the setting of PRIMSUB-METHOD.) Takes values: ms98-1, npfd, npfd-1, pfd, mtree, mtree-1 and mtree-2. The value MS98-1 is an exceptional setting used for both this flag and DEFAULT-EXPAND to denote the MS98-1 procedure. Changes DEFAULT-MS as follows: DEFAULT-EXPAND: | NONE | OTREE | OSET | ==========+=========+=========+========+ DEFAULT-MATE: NPFD | MS88 | MS89 | MS91-6 | ----------+---------+---------+--------+ NPFD-1 | MS92-9 | MS93-1 | N/A | ----------+---------+---------+--------+ PFD | MS90-3 | MS90-9 | MS91-7 | ----------+---------+---------+--------+ MTREE | MT94-11 | N/A | N/A | ----------+---------+---------+--------+ MTREE-1 | MT94-12 | N/A | N/A | ----------+---------+---------+--------+ MTREE-2 | MT95-1 | N/A | N/A | ----------+---------+---------+--------+It takes values of type SYMBOL and belongs to subjects `MS98-1`, `MTREE-TOP`, `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `MATING-SEARCH`. The default value is `PFD`.

DEFAULT-MS The default mating search procedure to be used when the GO command is invoked. This will be changed if you set the DEFAULT-MATE and DEFAULT-EXPAND flags (they may also change DEFAULT-MS to NIL, if you pick a non-existent combination -- see the help messages for those flags). Conversely, setting DEFAULT-MS will set the values of DEFAULT-MATE and DEFAULT-EXPAND, as follows: (Notice that for otree and oset searches, the actual primsubs generated will depend on the setting of PRIMSUB-METHOD.) DEFAULT-EXPAND: | NONE | OTREE | OSET | ==========+=========+=========+========+ DEFAULT-MATE: NPFD | MS88 | MS89 | MS91-6 | ----------+---------+---------+--------+ NPFD-1 | MS92-9 | MS93-1 | N/A | ----------+---------+---------+--------+ PFD | MS90-3 | MS90-9 | MS91-7 | ----------+---------+---------+--------+ MTREE | MT94-11 | N/A | N/A | ----------+---------+---------+--------+ MTREE-1 | MT94-12 | N/A | N/A | ----------+---------+---------+--------+ MTREE-2 | MT95-1 | N/A | N/A | ----------+---------+---------+--------+

Possible values are MS88, MS89, MS90-3, MS90-9, MS91-6, MS91-7, MS92-9, MS93-1, MT94-11, MT94-12, MT95-1 and

MS98-1.It takes values of type SEARCHTYPE and belongs to subjects IMPORTANT, MTREE-TOP, MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is MS90-3.

INTERRUPT-ENABLE
When true, allows user to interrupt mating search by typing a <RETURN>; otherwise mating search will continue until it succeeds or is aborted by a CTRL-G. You may want to set this flag to nil if you are going to have input commands (examples to run, etc.) read in from a file.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is T.

MATING-VERBOSE
Should be one of SILENT, MIN, MED, or MAX. Determines the amount of information given about the current mating process.It takes values of type VERBOSE and belongs to subjects MATING-SEARCH. The default value is MED.

MONITORFLAG The monitor is switched on if this flag is T and off if it is NIL. This flag is set by the command MONITOR, and unset by the command NOMONITOR (and may of course also be set manually).It takes values of type BOOLEAN and belongs to subjects MATING-SEARCH. The default value is NIL.

NEW-MATING-AFTER-DUP
This flag affects the way a complete mating is constructed after duplication. If nil, mating search attempts to extend only those matings which were inextensible earlier. Otherwise, it starts constructing new matings.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is NIL.

QUERY-USER Has the following effects according to its value: T : User will be queried by the mating search process as to whether a duplication of variables should occur, unification depth should be increased, etc. NIL : The mating search process will take some action that makes sense. QUERY-JFORMS : The mating search process will stop after printing each vpform and ask whether to search on this vpform or to generate another. (Note: in MS90-3, this is pointless, since the vpform never changes.) SHOW-JFORMS : Like QUERY-JFORMS, but automatically answers no to each question (and hence never actually proceeds with a search). QUERY-SLISTS : In the TEST top level, stops after each setting of the flags and asks whether to search with those settings.It takes values of type QUERYTYPE and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is NIL.

REC-MS-FILE If true, mating search events are recorded in file named by flag rec-ms-filename. This only works for npfd procedures MS88, MS89 and MS91-6.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is NIL.

REC-MS-FILENAME
Name of file in which mating search events are recorded. (See REC-MS-FILE.)It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "mating.rec".

USE-DIY When T, proof lines which are proven by DIY, DIY-L or UNIFORM-SEARCH-L will not be translated into natural deduction style, but will instead be justified in a single step, as "Automatic" from the support lines. A comment will be added to the relevant line of the proof showing the time taken and the mode used for the automatic proof.

Obviously, ND proofs containing justifications of this sort cannot be translated by NAT-ETREE.It takes values of type BOOLEAN and belongs to subjects OTL-VARS, TACTICS, MATING-SEARCH, ETR-NAT. The default value is NIL.

USE-FAST-PROP-SEARCH
If T, will attempt to use the path-focused fast propositional theorem prover on all problems, before switching to the usual default mating-search if this fails. If NIL, will use the default mating-search only.It takes values of type BOOLEAN and belongs to subjects MATING-SEARCH. The default value is T.

10.23. MS88 search procedure

ADDED-CONN-ENABLED
If NIL, recording events of type ADDED-CONN is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

CONSIDERED-CONN-ENABLED
If NIL, recording events of type CONSIDERED-CONN is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

DUP-ALLOWED If T mating search duplicates quantifiers whenever necessary.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is T.

DUPE-ENABLED If NIL, recording events of type DUPE is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

DUPE-VAR-ENABLED
If NIL, recording events of type DUPE-VAR is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

EXCLUDING-GC-TIME
If T, we can use the function get-net-internal-run-time to exclude the gc time in recordings. Otherwise, get-net-internal-run-time is the same as get-internal-run-time. The value of the flag should not be changed. This is a nominal flag, whose value does not affect the system at all except telling users the message above. Check the flags SEARCH-TIME-LIMIT and MAX-SEARCH-LIMIT to get more information.It takes values of type BOOLEAN and belongs to subjects MATING-SEARCH, SYSTEM. The default value is NIL.

FIRST-ORDER-MODE-MS
If T first-order unification is called during mating search, else higher-order unification is used. TPS changes the value of this flag to T when it is called by DIY to work on a first-order problem, but not when it is called from MATE.It takes values of type BOOLEAN and belongs to subjects MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is NIL.

INCOMP-MATING-ENABLED
If NIL, recording events of type INCOMP-MATING is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

MATE-FFPAIR Controls whether to consider a pair of literals with flexible heads as a potential connection. The MS controller will locally modify it under certain conditions; in particular, it will always be set locally to T in the following cases, among others: a) for first-order problems (when FIRST-ORDER-MODE-MS is T). b) when a mating is removed because it is incompatible with the etree. c) when using the interactive command ADD-CONN.It takes values of type BOOLEAN and belongs to subjects MS91-6, MS89, MS88, MATING-SEARCH. The default value is NIL.

MATE-SUBSUMED-TEST-ENABLED
If NIL, recording events of type MATE-SUBSUMED-TEST is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

MATE-SUBSUMED-TRUE-ENABLED
If NIL, recording events of type MATE-SUBSUMED-TRUE is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

MATING-CHANGED-ENABLED
If NIL, recording events of type MATING-CHANGED is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

MS-INIT-PATH If NIL MS considers the current path when a new mating is started. Otherwise, starts from the beginning in the natural ordering on paths in a jform.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is NIL.

MS-SPLIT If T mating search attempts to split the proof.It takes values of type BOOLEAN and belongs to subjects MS91-6, MS89, MS88, MATING-SEARCH. The default value is T.

OCCURS-CHECK This flag is not effective unless FIRST-ORDER-MODE-MS is T. If its value is T, occurs check in first-order unification is postponed till a mating is complete.It takes values of type BOOLEAN and belongs to subjects MS91-6, MS89, MS88, MATING-SEARCH. The default value is T.

PRIM-QUANTIFIER
When NIL, primitive substitutions containing new quantifiers will not be applied.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH, PRIMSUBS. The default value is T.

PRIMSUB-ENABLED
If NIL, recording events of type PRIMSUB is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

PROP-STRATEGY
This flag is only used in PROPOSITIONAL proof search, which can be one of (1) allow-duplicates (2) hash-table (3) pushnew (1) Adds CONNECTION to the mating even though it might already be in the mating. In case of (2) and (3) adds CONNECTION to the mating only if it is not already in the mating. (2) uses HASH-TABLE to determine this. (3) uses CLISP macro PUSHNEW to determine this.It takes values of type SYMBOL and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH. The default value is ALLOW-DUPLICATES.

REMOVED-CONN-ENABLED
If NIL, recording events of type REMOVED-CONN is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

SEARCH-COMPLETE-PATHS
: Not yet implemented. If NIL paths are generated only to a length until a connection can be located on it. Otherwise full paths are generated.It takes values of type BOOLEAN and belongs to subjects MS91-6, MS89, MS88, MATING-SEARCH. The default value is NIL.

START-TIME-ENABLED
: If NIL, recording events of type START-TIME is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

STOP-TIME-ENABLED
: If NIL, recording events of type STOP-TIME is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

TIMING-NAMED
: If T, the labels printed by display-time will be shortened to allow room for the name of the current dproof, if there is one. If NIL, then they won't. Abbreviations used are: PRE - preprocessing, MS - mating search, U - unification, PPR - postprocessing, MRG - merging, TRA - translation, PRT - printing.It takes values of type BOOLEAN and belongs to subjects MATING-SEARCH, SYSTEM. The default value is NIL.

UNIF-SUBSUMED-TEST-ENABLED
: If NIL, recording events of type UNIF-SUBSUMED-TEST is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

UNIF-SUBSUMED-TRUE-ENABLED
: If NIL, recording events of type UNIF-SUBSUMED-TRUE is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

10.24. MS89 search procedure

MAX-SEARCH-LIMIT
: If integer-valued, is an upper limit on the TOTAL amount of time (in seconds) which can be spent on searching for a proof in any particular option. If null, then search time is unbounded. The flag is not affected by Garbage Collecting time whenever the value of the flag excluding-gc-time is T. Please read the help message for EXCLUDING-GC-TIME for more information.It takes values of type NULL-OR-POSINTEGER and belongs to subjects MS93-1, MS91-7, MS91-6, MS90-9, MS89, IMPORTANT, MATING-SEARCH. The default value is NIL.

RANK-EPROOF-FN
: The name of a function which should take as its single argument an incomplete expansion proof, and return a nonnegative integer ranking the proof's likelihood of success, with 0 meaning no success (so don't try), and, otherwise, the better the likelihood, the lower the returned value. The only currently defined value for this flag is NUM-VPATHS-RANKING.It takes values of type SYMBOL and belongs to subjects MS93-1, MS90-9, MS89, MATING-SEARCH. The default value is NUM-VPATHS-RANKING.

SEARCH-TIME-LIMIT
: If integer-valued, is an upper limit on the CONTINUAL amount of time (in seconds) which can be spent on searching for a proof in any particular option. If null, then an ad hoc bound is used by the search procedure. The flag is not affected by Garbage Collecting time whenever the value of the flag excluding-gc-time is T. Please read the help message for EXCLUDING-GC-TIME for more information.It takes values of type NULL-OR-POSINTEGER and belongs to subjects MS93-1, MS91-7, MS91-6, MS90-9, MS89, IMPORTANT, MATING-SEARCH. The default value is NIL.

10.25. MS90-3 search procedure

MAX-MATES
: Max number of mates for a literal. If the search attempts to add a mate that would exceed this limit, then this connection is not added. Copies of a literal created by path-focused duplication are regarded as the same when computing this number. Set MAX-MATES to INFINITY to allow an unlimited number of mates for any literal.It takes values of type POSINTEGER-OR-INFINITY and belongs to subjects IMPORTANT, MS98-1, MS93-1, MS92-9, MS88, MS89, MS91-6, MS91-7, MS90-9, MS90-3, MATING-SEARCH. The default value is 2.

MIN-QUANT-ETREE
: Only affects path-focused search procedures. When this flag is T, the scope of quantifiers is minimized in primsubs appearing in the expansion proof after searching is done and before the propositional proof checker starts. This allows the corresponding instantiation terms in the ND proof to be in non-prenex form, often giving more readable proofs.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS90-9, MS90-3, MATING-SEARCH, ETREES. The default value is T.

MS90-3-DUP-STRATEGY
: 1 to select any combination of duplications (2 1 3 1 is allowed), any thing else to select duplications in non

decreasing order only. (2 1 3 1 is not allowed, but 1 1 2 3 is allowed.)It takes values of type INTEGER+ and belongs to subjects MS93-1, MS92-9, MS91-7, MS90-9, MS90-3, MS89, MATING-SEARCH. The default value is 1.

NUM-FRPAIRS The match routine considers at most NUM-FRPAIRS frpairs, before selecting a frpair. However, if it finds a pair that has at most 1 substitution, it will automatically select this pair. Applies to UN90 only.It takes values of type INTEGER+ and belongs to subjects MS93-1, MS92-9, MS91-7, MS90-9, MS90-3, UNIFICATION. The default value is 5.

PRINT-MATING-COUNTER
Prints the current mating after this many iterations in the top level ms90-3 search. Applicable only for path-focused duplication search proceduresIt takes values of type INTEGER+ and belongs to subjects MS91-7, MS90-9, MS90-3, MATING-SEARCH. The default value is 300000.

SHOW-TIME When true, print the time taken by MS90-3 and MS90-9.It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS90-9, MS90-3, MATING-SEARCH. The default value is T.

10.26. MS91-6 and MS91-7 search procedures

MS91-INTERLEAVE
In MS91-*, primitive substitutions are generated by NAME-PRIM, and they are applied to the master eproof before the search mechanism chooses particular parts of that eproof (and hence particular substitutions) to try and prove.

If MS91-INTERLEAVE is NIL, all of the substitutions generated by NAME-PRIM are applied at once, and then the search mechanism chooses among them, probably in the order in which they were generated. The process of applying them to the eproof can take a very long time.

If MS91-INTERLEAVE is an integer n, we take n primsubs at a time for each variable which has primsubs, and apply only those to the eproof. Once we have searched through those (to be specific, once we decide to generate new options), we take the next n primsubs for each variable and apply them, and so on. This is much quicker, and has the advantage of not having to work through every primsub for the first variable before starting work on the next variable.

If MS91-INTERLEAVE is non-NIL, and NEW-OPTION-SET-LIMIT is greater than MS91-INTERLEAVE * (# of vars that have primsubs), then TPS will reduce NEW-OPTION-SET-LIMIT. This ensures that single substitutions are generated before multiple substitutions.It takes values of type NULL-OR-POSINTEGER and belongs to subjects PRIMSUBS, MS91-7, MS91-6. The default value is 5.

MS91-PREFER-SMALLER
When T, smaller option-sets will be preferred to any larger ones.It takes values of type BOOLEAN and belongs to subjects MS91-7, MS91-6. The default value is T.

MS91-TIME-BY-VPATHS
When T, the amount of time given by SEARCH-TIME-LIMIT and MAX-SEARCH-LIMIT will be multiplied by the number of vertical paths through the vpform and then divided by the number of paths through the initial vpform (so the first vpform will get SEARCH-TIME-LIMIT seconds, and if the next has twice as many paths it will get twice as many seconds, and so on...). When NIL, every option set will get the same search time. This flag only applies in MS91 procedures.It takes values of type BOOLEAN and belongs to subjects MS91-7, MS91-6. The default value is NIL.

MS91-WEIGHT-LIMIT-RANGE
New option-sets, when constructed, will be accepted if their weights lie in the range [current weight limit, current weight limit + MS91-WEIGHT-LIMIT-RANGE]. Hence increasing this value means that more option-sets will be acceptable during the creation stage. If this range is very small, there is a risk that no option sets at all will be accepted and the search will waste time recreating these sets with a higher current weight limit. If it is too large, then there is a risk that high-weighted sets will be considered before lower-weighted ones. Note: option sets of weight INFINITY will never be accepted, no matter what.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects MS91-7, MS91-6. The default value is 1.

NEW-OPTION-SET-LIMIT
The maximum number of new option-sets that can be created at any one time. See MS91-INTERLEAVE.It takes values of type POSINTEGER and belongs to subjects MS91-7, MS91-6. The default value is 20.

OPTIONS-GENERATE-ARG
The argument used by the function given in the flag OPTIONS-GENERATE-FN. If this argument is INFINITY then new options will never be generated. See the help message for OPTIONS-GENERATE-FN.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects MS91-7, MS91-6. The default value is 75.

OPTIONS-GENERATE-FN
This is the function for deciding when to add new options to the list from which option sets are generated. This is only called when new option sets are being generated, so if you are generating large numbers of options sets at a

time then you might not see an effect until some time after your given criterion is satisfied. (Check the value of NEW-OPTION-SETS-LIMIT if this seems to be the case.) The argument for this function is in the flag OPTIONS-GENERATE-ARG, and the function to update that argument is in the flag OPTIONS-GENERATE-UPDATE. The options are: * ADD-OPTIONS-ORIGINAL generates new options when over OPTIONS-GENERATE-ARG percent of the possible option sets have been used, and each option appears in at least one option set. * ADD-OPTIONS-COUNT generates new options when more than OPTIONS-GENERATE-ARG different option sets have been tried. * ADD-OPTIONS-WEIGHT generates new options when the lower end of the acceptable weight bracket for a new option set exceeds OPTIONS-GENERATE-ARG. * ADD-OPTIONS-SUBS generates new options when the number of substitutions and duplications in the next option set (i.e. its SIMPLEST-WEIGHT-B) exceeds OPTIONS-GENERATE-ARG. If OPTIONS-GENERATE-ARG is INFINITY, no new options are ever generated.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `ADD-OPTIONS-ORIGINAL`.

OPTIONS-GENERATE-UPDATE

The function used to update the value of the flag OPTIONS-GENERATE-ARG. Current possibilites are: * IDENT-ARG leaves the value unchanged. * DOUBLE-ARG doubles the value. * SQUARE-ARG squares the value. * INF-ARG makes the value INFINITY. Note that a value of INFINITY means that new options will never be generated.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `IDENT-ARG`.

OPTIONS-VERBOSE

If T, will output extra information about the options being considered.It takes values of type BOOLEAN and belongs to subjects `MS91-7`, `MS91-6`. The default value is `NIL`.

PENALTY-FOR-EACH-PRIMSUB

Used in computing weight-b in MS91 search procedures. Should be a nonnegative integer or the symbol INFINITY, and will be the amount of penalty given for using each primitive substitution. See WEIGHT-B-FN.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `MS91-7`, `MS91-6`. The default value is `3`.

PENALTY-FOR-MULTIPLE-PRIMSUBS

Used in computing weight-b in MS91 search procedures. Should be a nonnegative integer or the symbol INFINITY, and will be the amount of penalty given for using more than one primitive substitution for a single variable. See WEIGHT-B-FN.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `MS91-7`, `MS91-6`. The default value is `5`.

PENALTY-FOR-MULTIPLE-SUBS

Used in computing weight-b in MS91 search procedures. Should be a nonnegative integer or the symbol INFINITY, and will be the amount of penalty given for using more than one substitution for a single variable. See WEIGHT-B-FN.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `MS91-7`, `MS91-6`. The default value is `5`.

PENALTY-FOR-ORDINARY-DUP

Used in computing weight-b in MS91 search procedures. Should be a nonnegative integer or the symbol INFINITY, and will be the amount of penalty given for each duplicate copy of a quantifier which is not used by a primitive substitution. See WEIGHT-B-FN.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `MS91-7`, `MS91-6`. The default value is `INFINITY`.

RECONSIDER-FN A function that should take a weight as argument and return a value to be used as a new weight after the associated option set runs out of time. Currently, the predefined functions are INF-WEIGHT SQUARE-WEIGHT, DOUBLE-WEIGHT and INCREMENT-WEIGHT (which, respectively, make reconsidering an old option set impossible, very unlikely, quite likely and probable). INCREMENT-WEIGHT actually adds 10 to the weight of an option set, as adding 1 is insignificant under most circumstances.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `DOUBLE-WEIGHT`.

WEIGHT-A-COEFFICIENT

Coefficient to be used in multiplying weight-a of options in the option-set of which we are computing weight-d. See WEIGHT-A-FN. The total weight of a set of options is the weight-a of each option plus the weight-b of the set plus the weight-c of the set.It takes values of type INTEGER+ and belongs to subjects `MS91-7`, `MS91-6`. The default value is `0`.

WEIGHT-A-FN A function that should take an option as argument and return a value to be used as its weight-a. Currently, the only such predefined function is EXPANSION-LEVEL-WEIGHT-A, which returns the expansion level of the option to be used as a weight. The total weight of a set of options is the weight-a of each option plus the weight-b of the set plus the weight-c of the set.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `EXPANSION-LEVEL-WEIGHT-A`.

WEIGHT-B-COEFFICIENT

Coefficient to be used in multiplying weight-b of option/option-subset pairs for the option-set of which we are computing weight-d. See WEIGHT-B-FN. The total weight of a set of options is the weight-a of each option plus

the weight-b of the set plus the weight-c of the set.It takes values of type INTEGER+ and belongs to subjects `MS91-7`, `MS91-6`. The default value is `1`.

WEIGHT-B-FN A function that should take an option set and return a value to be used as its weight-b. Currently, the only such predefined functions are: * SIMPLE-WEIGHT-B-FN, which returns the sum of the penalties for the primsubs, multiple subs and duplications used in the option set (see the flags PENALTY-FOR-EACH-PRIMSUB, PENALTY-FOR-MULTIPLE-PRIMSUBS and PENALTY-FOR-MULTIPLE-SUBS for more information), * ALL-PENALTIES-FN which is much the same as SIMPLE-WEIGHT-B-FN but also adds a penalty for extra duplications given by the PENALTY-FOR-ORDINARY-DUP flag, and * SIMPLEST-WEIGHT-B-FN, which returns 1 for the original option set and adds 1 for each primsub or duplication (the idea is to set the coefficients of weight-a and weight-c to zero while using SIMPLEST-WEIGHT-B-FN). The total weight of a set of options is the weight-a of each option plus the weight-b of the set plus the weight-c of the set.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `SIMPLEST-WEIGHT-B-FN`.

WEIGHT-C-COEFFICIENT
Coefficient to be used in multiplying weight-c of options in the option-set of which we are computing weight-d. See WEIGHT-C-FN. The total weight of a set of options is the weight-a of each option plus the weight-b of the set plus the weight-c of the set.It takes values of type INTEGER+ and belongs to subjects `MS91-7`, `MS91-6`. The default value is `0`.

WEIGHT-C-FN A function that should take an list of options as argument and return a value to be used as its weight-c. Currently, the only such predefined functions are OPTION-SET-NUM-VPATHS, which returns the number of vertical paths through the relevant etree, and OPTION-SET-NUM-LEAVES, which returns the number of leaves in the relevant etree. The total weight of a set of options is the weight-a of each option plus the weight-b of the set plus the weight-c of the set.It takes values of type SYMBOL and belongs to subjects `MS91-7`, `MS91-6`. The default value is `OPTION-SET-NUM-LEAVES`.

10.27. MS98-1 search procedure

BREAK-AT-QUANTIFIERS
Applies only to quantifiers which cannot be duplicated later in the search. If T, then fragments will be broken so as not to contain any quantifiers; if NIL, fragments may contain quantifiers of the sort specified.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

FF-DELAY If T, delay unifying f-f pairs for single connections, and unify them in context when some f-r pairs are added. If NIL, unify them as usual.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

HPATH-THRESHOLD
If NIL, break on major conjunctions. If n, break at conjunctions and also on disjunctions having more than n hpaths.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS98-1`. The default value is `1`.

MAXIMIZE-FIRST
For each component which is being extended, do not create any new components which exceed MAX-MATES 1 until there are no other ways to extend the component.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-BASE-PRIM
If T, we allow the search to begin with a fragment which is part of a primitive substitution. If NIL, we always choose a fragment which is outside the primitive substitutions (if possible).It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-DUP-PRIMSUBS
When T, MS98-DUP duplicates variables which have primsubs; when NIL, it doesn’tIt takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-FRAGMENT-ORDER
The order in which the fragments are considered. This principally affects which fragment will become the starting point of the search, and which of the touched but not blocked fragments will be blocked next. 0 : consider the number of ways to block the given fragment. 1 : consider the number of ways that the results for 0 might be extended (i.e. look ahead two steps in the search process) 2 : as for 1, but then weight in favour of ground fragments (i.e. those containing no variables).It takes values of type INTEGER+ and belongs to subjects `MS98-1`. The default value is `1`.

MS98-INIT Before doing ms98-1 search: If 0, do nothing. If 1, duplicate all outer quantifiers NUM-OF-DUPS times. If 2, apply primsubs and duplicate all outer quantifiers NUM-OF-DUPS times.It takes values of type INTEGER+ and belongs to subjects `MS98-1`. The default value is `0`.

MS98-MAX-PRIMS
The maximum number of primsubs allowed in any component.It takes values of type NULL-OR-POSINTEGER

and belongs to subjects `MS98-1`. The default value is `1`.

MS98-MEASURE Determines the measure which is used on components. If 0, count the components blocked and then weight heavily against the situation described by MS98-VALID-PAIR. If 1, the standard measure using the # of components blocked and touched If 2, as for 1 but also take account of the number of dups If 3, just count the number of components blocked If 4, as for 2 but also count the no of matings for the smallest component touched If 5, multiply the no of matings for the smallest touched by the number of subs. If 6, use the ratio of blocked to touched components and the ratio of the number of blocked components to the number of connections. If 7, prefer matings where positive leaves are mated to negative leaves and vice versa. If 8, use the ratio of blocked to touched components. If 9, favour large components satisfying max-mates 1. If 10, do as for 9 and then weight heavily against the situation described by MS98-VALID-PAIR.It takes values of type INTEGER+ and belongs to subjects `MS98-1`. The default value is `0`.

MS98-NUM-OF-DUPS
If NIL, we can use every duplication that's present. If some positive integer n, we reject any component using more than n of the duplications.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-PRIMSUB-COUNT
The maximum number of primsubs to be applied each set variable in the expansion tree.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS98-1`. The default value is `3`.

MS98-REW-PRIMSUBS
When T, MS98-DUP does primsubs for Leibniz variables which have become rewrites; when NIL, it doesn'tIt takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-REWRITE-DEPTH
When attempting to rewrite one term into another, the maximum number of steps of rewriting that are allowed.It takes values of type POSINTEGER and belongs to subjects `MS98-1`. The default value is `2`.

MS98-REWRITE-MODEL
If T, ask the user for a model of the rewrite rules to help slim down the unification tree.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-REWRITE-PRUNE
If T, delete any unifiers which are duplicates modulo rewriting (this can be slow). If NIL, don't.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `T`.

MS98-REWRITE-SIZE
The maximum size of a (lambda-normalized) term that can be produced by rewriting, measured as the number of nodes in the parse tree of that term. NIL means that there is no maximum.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-REWRITE-UNIF
When a rewrite rule can introduce a new variable, this flag governs the size of the allowed substitutions for that variable. Essentially, this is a special case of MAX-SUBSTS-VAR.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-REWRITES When T, make all of the global equalities into rewrites.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-STORE When T, store all the results of merging unification DAGs in the main hashtable. When NIL, just store the results from the first round of merging.It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `T`.

MS98-UNIF-HACK
If T, do not introduce new constants during unification. (NOTE: This is a hack; we *do* need to introduce new constants, in general, but in most cases we needn't bother.)It takes values of type BOOLEAN and belongs to subjects `MS98-1`. The default value is `NIL`.

MS98-VALID-PAIR
Given two disjuncts X OR Y and A OR B, this flag determines when we are allowed to make a component containing connections X-A and Y-B (assuming they're unifiable connections). The higher the number, the more stringent (and more time-consuming) the test; any correct mating is guaranteed to pass any of these tests: 1: MAX-MATES is not 1. 2: As for 1, plus we require an extra mate for each of X,Y,A and B. 3: As for 2, plus we require that all of these new mates be pairwise compatible with each other. 4: As for 3, plus we require that all of these new mates be simultaneously compatible with each other. 5: As for 4, plus we return all possible sets of new mates and add them into the component immediately.

3,4 and 5 are only applicable to higher-order searches.

There is an extra value, 0, which rejects any such connections even if max-mates is not 1. This results in an incomplete search, but is often acceptable.It takes values of type INTEGER+ and belongs to subjects `MS98-1`. The default value is `2`.

MS98-VERBOSE If T, print extra information during MS98-1 search.It takes values of type BOOLEAN and belongs to subjects

MS98-1. The default value is NIL.

10.28. Proof Translation

ETREE-NAT-VERBOSE

Should be a list of print-functions (see the help message for PRINT-FUNCTION), which will be executed after each tactic during ETREE-NAT.It takes values of type PRINT-FUNCTION-LIST and belongs to subjects PRINTING, ETR-NAT. The default value is (PRFW-PALL PRFW-^P PRFW-^PN ^PN).

MERGE-MINIMIZE-MATING

If T, merging will attempt to minimize the mating by removing any unnecessary connections. If NIL, it won't. T will sometimes produce a more readable ND proof, but can also take a very long time.It takes values of type BOOLEAN and belongs to subjects MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, ETR-NAT, ETREES, MATING-SEARCH. The default value is T.

NATREE-DEBUG No more help available. Sorry.It takes values of type BOOLEAN and belongs to subjects The default value is NIL.

REMOVE-LEIBNIZ

If TRUE, selection parameters corresponding to Leibniz equality definitions will be removed from expansion proofs during merging (cf. Pfenning's thesis, theorem 138).It takes values of type BOOLEAN and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, ETR-NAT, ETREES, MATING-SEARCH. The default value is T.

10.29. Unification

APPLY-MATCH Heuristic to decide the pair that should be given to match. UN88 procedures: APPLY-MATCH-ALL-FRDPAIRS applies match to all flexible-rigid pairs and chooses whichever will have fewest substitutions. APPLY-MATCH-ALL-FRDPAIRS-MSV does the same, but also checks for MAX-SUBSTS-VAR violations at the same time. APPLY-MATCH-MAX-SUBSTS applies match to whichever flexible-rigid pair is closest to exceeding the bound in MAX-SUBSTS-VAR. If it finds one with a unique substitution, it uses that. APPLY-MATCH-MIN-SUBSTS is like the above, but chooses the pair which is farthest from the MAX-SUBSTS-VAR bound. APPLY-MATCH-MOST-CONSTS applies match to whichever flex-rigid pair contains the most constant symbols. (The last two of these are all but useless; both of the SUBSTS versions will be disastrous if MAX-SUBSTS-VAR is NIL...)

UN90 procedures: This flag is almost always ignored (the default behaviour is much like APPLY-MATCH-ALL-FRDPAIRS, but see NUM-FRPAIRS and COUNTSUBS-FIRST for more details). The exception is if it is APPLY-MATCH-MAX-SUBSTS, in which case it will go for whichever pair is closest to exceeding the MAX-SUBSTS-VAR bound (but will still use NUM-FRPAIRS and COUNTSUBS-FIRST).It takes values of type SYMBOL and belongs to subjects UNIFICATION. The default value is APPLY-MATCH-ALL-FRDPAIRS.

COUNTSUBS-FIRST

if NIL, the substitutions which MATCH generates for each dpair in the unification process are generated and counted, and then MATCH is actually applied to the variable for which this number is smallest; if T, the substitutions are counted before they are generated, and only those which will be applied are actually generated. Applies to UN90 only.It takes values of type BOOLEAN and belongs to subjects UNIFICATION. The default value is NIL.

ETA-RULE If T, eta rule is permitted in the unification package. This can be T or NIL for the UN88 procedure, but it can only be T for the UN90 procedure. (In fact, UN90 ignores this flag.)It takes values of type BOOLEAN and belongs to subjects UNIFICATION. The default value is T.

IMITATION-FIRST

Controls whether imitations are considered before projections during unification procedure UN88. No effect in UN90.It takes values of type BOOLEAN and belongs to subjects UNIFICATION. The default value is T.

LEIBNIZ-SUB-CHECK

When T, check substitutions which are made for Leibniz variables, to ensure that they are relevant in their first argument. When NIL, don't do this.It takes values of type BOOLEAN and belongs to subjects UNIFICATION. The default value is NIL.

MAX-SEARCH-DEPTH

If non nil, search to depth MAX-SEARCH-DEPTH, else search to arbitrary depth. Takes precedence over all other flags that may control the search depth in a unification tree (i.e. no tree is ever generated to a greater depth, although other flags may cause the unification search to stop temporarily at a shallower depth. Used in all search procedures, and in UN88 and UN90. See flag MAX-UTREE-DEPTH also.It takes values of type NULL-OR-POSINTEGER and belongs to subjects IMPORTANT, UNIFICATION. The default value is NIL.

MAX-UTREE-DEPTH

If non-NIL, maximum depth to which unification tree is to be generated. Used only in UN88 procedures. This variable is incremented during mating-search to allow unification tree to grow to greater depth as the search progresses. The unification tree is, however, never searched or generated to a depth greater than MAX-SEARCH-DEPTH provided it is non NIL and a positive integer. One can also consider this variable to be the initial value to which unification trees are generated during mating-search.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `IMPORTANT`, `UNIFICATION`. The default value is `5`.

MIN-QUICK-DEPTH

The minimum depth to which a unification tree should be generated when unification tree is searched only to non branching depth. Setting this flag to 1 has the effect of generating the tree to non branching depth. Applicable only to UN88. MIN-QUICK-DEPTH is used only in the process of checking whether two literals are potential mates. It is used to construct the connection graph. See flag MAX-SEARCH-DEPTH also. See MAX-SUBSTS-QUICK for a different way to achieve a similar effect.It takes values of type NULL-OR-POSINTEGER and belongs to subjects `UNIFICATION`. The default value is `3`.

MS-DIR The director to be used in mating search.It takes values of type SYMBOL and belongs to subjects `MS91-6`, `MS89`, `MS88`, `MATING-SEARCH`. The default value is `QUASI-TPS1`.

MS90-3-QUICK If T, do MS88 quick unification on dpairs in MS90-3. If NIL, don't.It takes values of type BOOLEAN and belongs to subjects `UNIFICATION`, `MS92-9`, `MS93-1`, `MS91-7`, `MS90-9`, `MS90-3`. The default value is `NIL`.

PRUNING If T, the unification routine will prune the tree as it goes. Only works for BREADTH-FIRST and BEST-FIRST unification, and only then in MS88.It takes values of type BOOLEAN and belongs to subjects `UNIFICATION`. The default value is `NIL`.

REDUCE-DOUBLE-NEG

If T double negations are eliminated during lambda contraction at a unification node. This only applies in UN88.It takes values of type BOOLEAN and belongs to subjects `UNIFICATION`. The default value is `T`.

RIGID-PATH-CK If T, apply rigid-path checking when doing unification. If NIL, switch to original unification. Both UN90 and UN88 unification procedures are affected by the flag.It takes values of type BOOLEAN and belongs to subjects `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `UNIFICATION`. The default value is `T`.

STOP-AT-TSN If T the unification algorithm terminates at a terminal success node. Otherwise, it continues generating the tree. This only applies to UN88.It takes values of type BOOLEAN and belongs to subjects `UNIFICATION`. The default value is `T`.

SUBSUMPTION-CHECK

Limited subsumption check should be done during unification when this flag is set. Applies for procedures UN88 and UN90, although it is much more useful in UN88 (UN90 does not generate as many subsumed nodes, and so subsumption-checking tends to be a waste of time). See also SUBSUMPTION-NODES and SUBSUMPTION-DEPTH.It takes values of type BOOLEAN and belongs to subjects `UNIFICATION`. The default value is `NIL`.

SUBSUMPTION-DEPTH

Subsumption checking takes a lot of time, compared to unification, which means that checking a new node may take more time than it could possibly save, particularly if the node is almost at the maximum depth for the unification tree. In the unification tree, new nodes at depth SUBSUMPTION-DEPTH or deeper will not be subsumption-checked; other new nodes will be. Having SUBSUMPTION-DEPTH INFINITY means that all new nodes are subsumption-checked; SUBSUMPTION-DEPTH 0 is just a slower way of turning subsumption-checking off altogether. (You should use SUBSUMPTION-CHECK NIL to do that!) This flag only applies when SUBSUMPTION-CHECK is T. See also SUBSUMPTION-NODES.It takes values of type INTEGER+-OR-INFINITY and belongs to subjects `UNIFICATION`. The default value is `INFINITY`.

SUBSUMPTION-NODES

When SUBSUMPTION-CHECK is T, this flag determines which other nodes should be examined to see if they subsume the new node being considered. The values are as follows, arranged in order with the quickest first: PATH-NODES checks only those nodes on the path from the root to the new node. LEAF-NODES checks only the leaf nodes in the tree. LP-NODES checks leaf nodes and those on the path to the new node. ALL-NODES checks every node in the tree. Some nodes will always be excluded from subsumption checking, regardless of the value of this flag. In particular, two nodes representing different sets of connections will not be compared. This flag only applies to the UN88 procedure; in UN90, if subsumption-checking is used at all, it is implicitly set to ALL-NODES.It takes values of type SYMBOL and belongs to subjects `UNIFICATION`. The default value is `LP-NODES`.

UNI-SEARCH-HEURISTIC

Search strategy used to select the next node in the unification tree. BREADTH-FIRST and DEPTH-FIRST are exactly as they sound; BEST-FIRST takes whichever leaf node has the fewest free variables (and is not already terminal). All of these options work for UN90 (ms90-*, ms91-7, ms92-*); BREADTH-FIRST and BEST-FIRST are the only options for UN88 (ms88, ms89, ms91-6, mtree).It takes values of type SYMBOL and belongs to

subjects UNIFICATION. The default value is BREADTH-FIRST.

UNIFY-VERBOSE Takes values SILENT=NIL, MIN, MED or MAX=T, and governs the amount of output relating to the unification process.It takes values of type VERBOSE and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, MATING-SEARCH, UNIFICATION. The default value is MED.

10.30. Tactics

DEFAULT-TACTIC

The default tactic for ETREE-NAT and USE-TACTIC. See the help messages for these commands for more information.It takes values of type TACTIC-EXP and belongs to subjects TACTICS. The default value is (IDTAC).

TACMODE The default mode for tactics. It takes values of type TACTIC-MODE and belongs to subjects TACTICS. The default value is INTERACTIVE.

TACTIC-VERBOSE

Determines which of the three levels of verbosity will be used: MAX -- prints the message returned by each tactic called, even if it fails. MED -- prints messages only when tactic succeeds. MIN -- prints nothing.It takes values of type SYMBOL and belongs to subjects TACTICS. The default value is MED.

TACUSE The default use for tactics. It takes values of type TACTIC-USE and belongs to subjects TACTICS. The default value is NAT-DED.

10.31. suggestions

GO-INSTRUCTIONS

A list of instructions for GO to decide what to do with suggestions. It is a list of pairs (priority action), action being among DO, ASK, SHOW, FORGET. The default setting ((0 DO) (5 ASK) (9 SHOW) (100 FORGET)) means do suggestions of priority 0, ask me about doing suggestions of priority 5 or less, otherwise just show me suggestions of priority 9 or less and then quit.It takes values of type GO-INSTRUCT and belongs to subjects SUGGESTS. The default value is ((0 DO) (5 ASK) (9 SHOW) (100 FORGET)).

QUIETLY-USE-DEFAULTS

If T, GO will fill in arguments with their defaults without asking for confirmation. If NIL, the command will be executed like any other command issued at the top level.It takes values of type BOOLEAN and belongs to subjects SUGGESTS. The default value is T.

RESOLVE-CONFLICT

If T, always the first of several suggestions is chosen, if NIL, the user will be asked.It takes values of type BOOLEAN and belongs to subjects SUGGESTS. The default value is T.

10.32. Searchlists

TEST-EASIER-IF-HIGH

The list of flags that, if set to high numbers, make mating-search easier. Used by SCALE-UP. "Easier" in this context means "more likely to succeed eventually, although possibly taking longer about it". Compare TEST-FASTER-IF-HIGH; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (MAX-SEARCH-DEPTH SEARCH-TIME-LIMIT NUM-OF-DUPS MAX-UTREE-DEPTH MAX-MATES MAX-SEARCH-LIMIT).

TEST-EASIER-IF-LOW

The list of flags that, if set to low numbers, make mating-search easier. Used by SCALE-UP. "Easier" in this context means "more likely to succeed eventually, although possibly taking longer about it". Compare TEST-FASTER-IF-LOW; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (MIN-QUICK-DEPTH).

TEST-EASIER-IF-NIL

The list of flags that, if set to NIL, make mating-search easier. Used by SCALE-UP. "Easier" in this context means "more likely to succeed eventually, although possibly taking longer about it". Compare TEST-FASTER-IF-NIL; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is ().

TEST-EASIER-IF-T

The list of flags that, if set to T, make mating-search easier. Used by SCALE-UP. "Easier" in this context means "more likely to succeed eventually, although possibly taking longer about it". Compare TEST-FASTER-IF-T; the

list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (ETA-RULE MIN-QUANTIFIER-SCOPE MS-SPLIT).

TEST-FASTER-IF-HIGH
: The list of flags that, if set to high numbers, make mating-search faster. Used by SCALE-DOWN. "Faster" in this context means "if it succeeds at all, it does so more quickly". Compare TEST-EASIER-IF-HIGH; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (MIN-QUICK-DEPTH).

TEST-FASTER-IF-LOW
: The list of flags that, if set to low numbers, make mating-search faster. Used by SCALE-DOWN. "Faster" in this context means "if it succeeds at all, it does so more quickly". Compare TEST-EASIER-IF-LOW; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (MAX-SEARCH-DEPTH SEARCH-TIME-LIMIT NUM-OF-DUPS MAX-UTREE-DEPTH MAX-MATES MAX-SEARCH-LIMIT).

TEST-FASTER-IF-NIL
: The list of flags that, if set to NIL, make mating-search run faster. Used by SCALE-DOWN. "Faster" in this context means "if it succeeds at all, it does so more quickly". Compare TEST-EASIER-IF-NIL; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is ().

TEST-FASTER-IF-T
: The list of flags that, if set to T, make mating-search faster. Used by SCALE-DOWN. "Faster" in this context means "if it succeeds at all, it does so more quickly". Compare TEST-EASIER-IF-T; the list is somewhat debatable, which is why you're allowed to change it.It takes values of type TPSFLAGLIST and belongs to subjects TEST-TOP. The default value is (MIN-QUANTIFIER-SCOPE MS-SPLIT).

TEST-FIX-UNIF-DEPTHS
: If T, then LEAST-SEARCH-DEPTH will be used to fix the unification depths MAX-UTREE-DEPTH and MAX-SEARCH-DEPTH as soon as a search in the TEST top level is successful, and these will not be varied again. Destructively alters the search list, by changing the range of these two flags to a single element.It takes values of type BOOLEAN and belongs to subjects TEST-TOP. The default value is T.

TEST-INCREASE-TIME
: After each unsuccessful search in the test top level, the value of TEST-INITIAL-TIME-LIMIT will be increased by this proportion. (So, e.g., setting this flag to 10 will result in a 10% increase on each attempt; setting it to 100 will double TEST-INITIAL-TIME-LIMIT every time around.) NOTE: After the first successful search, this flag will be set to zero. The change will be permanent, in order to allow CONTINUE to work properly.It takes values of type INTEGER+ and belongs to subjects TEST-TOP. The default value is 0.

TEST-INITIAL-TIME-LIMIT
: The time limit to be used for each individual search. This limit will be increased if it is found to be insufficient. See also the flags TEST-INCREASE-TIME and TEST-REDUCE-TIME. The time referred to will be internal time without counting garbage collection, if possible (see the flag EXCLUDING-GC-TIME).It takes values of type POSINTEGER and belongs to subjects TEST-TOP. The default value is 30.

TEST-MAX-SEARCH-VALUES
: The maximum number of values that will be put in the range of any flag in an automatically-generated searchlist. (In a manually-generated list, you can have as large a range as you like.)It takes values of type POSINTEGER and belongs to subjects TEST-TOP. The default value is 10.

TEST-NEXT-SEARCH-FN
: The name of a function which should take a searchlist and the time taken for the previous attempt as arguments, and should set the flags in the list appropriately for the next search. This function should also return T in *finished-flag* if all settings have been tried. The only values defined so far are: EXHAUSTIVE-SEARCH, which tries all combinations of flags in a searchlist, varying one flag through its entire range before trying the next flag. BREADTH-FIRST-SEARCH, which also tries all combinations of flags, but varies each flag a little at a time. PRESS-DOWN, which is used by the PRESS-DOWN command. PRESS-DOWN-2, which behaves like breadth-first search except that if varying a flag makes the search faster, that flag is then prevented from returning above its original value (the range of each flag is assumed to be ordered; if the range is (A B C D), and setting it to C results in a faster search, it will never again be set to A or B). PUSH-UP, which is used by the PUSH-UP command. PUSH-UP-2, which is like breadth-first search but terminates once a successful mode is discovered; it is used for relaxing an unsuccessful mode until it is successful.It takes values of type SYMBOL and belongs to subjects TEST-TOP. The default value is EXHAUSTIVE-SEARCH.

TEST-REDUCE-TIME
: If T, then TEST-INITIAL-TIME-LIMIT will be reduced every time a faster combination of flags is found. If NIL, then it won't be.It takes values of type BOOLEAN and belongs to subjects TEST-TOP. The default value is T.

TEST-VERBOSE If NIL, suppresses a lot of the output of the test top level.It takes values of type BOOLEAN and belongs to

subjects TEST-TOP. The default value is T.

TESTWIN-HEIGHT
Contains the initial height of the testwindow.It takes values of type POSINTEGER and belongs to subjects PRINTING, TEST-TOP. The default value is 24.

TESTWIN-WIDTH
Contains the initial width of the testwindow.It takes values of type POSINTEGER and belongs to subjects PRINTING, TEST-TOP. The default value is 80.

10.33. Vpforms

LIT-NAME Prefix for labels associated with literals.It takes values of type SYMBOL and belongs to subjects JFORMS. The default value is LIT.

ORDER-COMPONENTS
When T or PATHNUM, the components of a jform node will be rearranged in order of the number of paths which lie below them (go through them). When T-REVERSED or PATHNUM-REVERSED, the components of a jform node will be rearranged in reverse order of the number of paths which lie below them (go through them). When NIL or COMMON, then the jform of the current eproof will not be modified by the mating search; When REVERSE, the order of the components in the jform of current eproof will be reversed; When PREFER-RIGID2, the order of the components in the jform of the current eproof will be sorted in terms of the number of rigid literals in a jform before beginning the mating search. When PREFER-RIGID3, the components in the jform of the current eproof will be sorted as for PREFER-RIGID2, but with preference given to literals that arise from DUAL rewriting.

(PREFER-RIGID1 is still available; it is an obsolete version of PREFER-RIGID2.)It takes values of type ORDERCOM and belongs to subjects MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, IMPORTANT, JFORMS, MATING-SEARCH. The default value is T.

PRINT-LIT-NAME
If the value of this flag is true, labels (instead of wffs associated with literal, or neg-literal) are printed inside the editor.It takes values of type BOOLEAN and belongs to subjects JFORMS. The default value is T.

PRINTVPDFLAG If T, vertical path diagrams are written into the VPD-FILENAME whenever wffs are written into the PRINTEDTFILE. In particular PRINTEDTFLAG must be T, for the automatic writing to take place.It takes values of type BOOLEAN and belongs to subjects EDITOR, JFORMS. The default value is NIL.

VPD-BRIEF The default value for BRIEF when printing VP diagrams in a file. Currently the options are: T = no atom values will show in VP diagram A = atom values but no labels will appear in VP diagram NIL = atom values and labels will show in VP diagram LT = atom values and labels and a legend will show in VP diagram L = labels but no atom values will show in VP diagram, and a legend will show both B = boxed labels and atoms will show in the VP diagram. BT = boxed labels will show in the diagram, and the atom values will be listed below. B and BT only work in TeX format (i.e. with the VPT command).It takes values of type VPFORMAT and belongs to subjects JFORMS. The default value is L.

VPD-FILENAME Default filename when printing VP diagrams in a file.It takes values of type FILESPEC and belongs to subjects JFORMS. The default value is "vpd.vpf".

VPD-LIT-NAME Prefix for labels associated with literals when VP diagrams are created automatically within the editor.It takes values of type SYMBOL and belongs to subjects JFORMS. The default value is V.

VPD-PTYPES If T, print types when printing VP diagrams in a file.It takes values of type BOOLEAN and belongs to subjects JFORMS. The default value is T.

VPD-STYLE The default value for STYLE when printing VP diagrams in a file.It takes values of type VPSTYLE and belongs to subjects JFORMS. The default value is GENERIC.

VPD-VPFPAGE The default value for the width of the page when printing VP diagrams in a file.It takes values of type POSINTEGER and belongs to subjects JFORMS. The default value is 78.

VPFORM-LABELS
In the editor, a value of T for this flag will suppress printing of labels in vpforms; if it is NIL, labels and atom values will be printed. If this flag is set the default value for argument BRIEF will be A. Unless one decides to override the default value, labels will not be printed. This flag has no effect on the editor command VPD, and on the wffop DISPLAY-VPD. To suppress labels when using these commands, please set the flag VPD-BRIEF to A.It takes values of type BOOLEAN and belongs to subjects JFORMS. The default value is NIL.

VPW-HEIGHT Contains the intial height of the vpform window; there is no need to update this if the window is resized after being opened.It takes values of type POSINTEGER and belongs to subjects PRINTING, JFORMS. The default value is 25.

VPW-WIDTH Contains the current width of the vpform window; should be updated by the user if the window is resized after

being opened.It takes values of type POSINTEGER and belongs to subjects PRINTING, JFORMS. The default value is 120.

10.34. Wff Editor

EDPPWFFLAG If T, wffs are always pretty-printed in the formula editor.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is NIL.

EDPRINTDEPTH The depth to which wffs are printed in the formula editor.It takes values of type INTEGER+ and belongs to subjects PRINTING, EDITOR. The default value is 24.

EDWIN-CURRENT
If T, the Current Edwff window is opened to display the current wff being edited when the editor is started.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is T.

EDWIN-CURRENT-HEIGHT
Controls the initial height of the Current Edwff window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 3.

EDWIN-CURRENT-WIDTH
Controls the initial width of the Current Edwff window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 80.

EDWIN-TOP If T, the Top Edwff window is opened to display the entire wff being edited when the editor is started.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is T.

EDWIN-TOP-HEIGHT
Controls the initial height of the Top Edwff window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 3.

EDWIN-TOP-WIDTH
Controls the initial width of the Top Edwff window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 80.

EDWIN-VPFORM If T, the Current Vpform window is opened to display the vpform of the current wff being edited when the editor is started. This flag is ignored in ETPS, where the Vpform window is never opened.It takes values of type BOOLEAN and belongs to subjects PRINTING, EDITOR. The default value is NIL.

EDWIN-VPFORM-HEIGHT
Controls the initial height of the Current Vpform window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 30.

EDWIN-VPFORM-WIDTH
Controls the initial width of the Current Vpform window.It takes values of type POSINTEGER and belongs to subjects PRINTING, EDITOR. The default value is 60.

10.35. wff Primitives

META-BDVAR-NAME
The prefix for names of bound meta variables.It takes values of type SYMBOL and belongs to subjects INTERNAL-NAMES. The default value is BD.

META-VAR-NAME
The prefix for names of meta variables.It takes values of type SYMBOL and belongs to subjects INTERNAL-NAMES. The default value is MV.

REN-VAR-FN The value of this flag is a function to be called when a variable must be renamed automatically. It has three possible settings: REN-VAR-X1 is the standard renaming function. It renames y to y^1, then to y^2, and so on. If there is another variable y, of a different type, it makes no difference. REN-VAR-X11 is much like REN-VAR-X1, except it will avoid creating two variables of the same name at different types (so it tends to produce higher exponents than REN-VAR-X1). REN-VAR-XA renames alphabetically, turning y into ya, then yba, and so on.It takes values of type SYMBOL and belongs to subjects WFF-PRIMS. The default value is REN-VAR-X1.

RENAME-ALL-BD-VARS
When T, all bound variables inside a definition will be renamed before instantiation.It takes values of type BOOLEAN and belongs to subjects WFF-PRIMS. The default value is NIL.

10.36. Wff Parsing

BASE-TYPE If not NIL, it should be the 'default' type for individual variables in a logic system. Typically I (for iota).It takes values of type SYMBOL and belongs to subjects PARSING. The default value is NIL.

FIRST-ORDER-MODE-PARSE
If T, every letter by itself is a symbol for the parser, with the exception of keywords like FORALL, AND etc., which can be in mixed case. If NIL, symbols must be separated by spaces (or brackets, dots, etc.).It takes values of type BOOLEAN and belongs to subjects PARSING. The default value is NIL.

LOWERCASERAISE
If T, lower case characters will be raised to upper case, when read. Has no effect in first-order mode.It takes values of type BOOLEAN and belongs to subjects PARSING. The default value is NIL.

TYPE-IOTA-MODE
If T, type variables are always assumed to be iota.It takes values of type BOOLEAN and belongs to subjects PARSING. The default value is T.

UNTYPED-LAMBDA-CALCULUS
Takes values T or NIL. To set it to T if you want to use the editor to deal with untyped lambda-calculus.It takes values of type BOOLEAN and belongs to subjects EDITOR. The default value is NIL.

10.37. Basic Abbreviations

REWRITE-EQUALITIES
One of the following: NONE: do not rewrite equalities ONLY-EXT: rewrite only those equalities that can be rewritten using extensionality. LEIBNIZ: rewrite all equalities using the Leibniz definition. ALL: rewrite all equalities, to an equivalence for those of type OOO, to the extensional form [lambda f(AB) lambda g(AB) forall x(B) f x = g x] for those of type O(AB)(AB), and to the Leibniz form [lambda x(A) lambda y(A) forall q(OA). q x implies q y] for those of type OAA. LAZY2: As for ALL, but keeping a duplicate leaf as in the LAZY2 setting of the flag REWRITE-DEFNS.It takes values of type REWRITE-DEFNS and belongs to subjects MS98-1, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, IMPORTANT, WFF-PRIMS, MATING-SEARCH. The default value is ALL.

10.38. Lambda-Calculus

LAMBDA-CONV BETA-ETA-TOGETHER means that BETA and ETA rules are used together; BETA-ETA-SEPARATE means BETA and ETA rules are used separately; BETA-ONLY means that onle BETA rule is allowed.It takes values of type SYMBOL and belongs to subjects TACTICS, ETR-NAT, ETREES. The default value is BETA-ETA-TOGETHER.

10.39. Primitive Substitutions

MAX-PRIM-DEPTH
Maximum depth to which primsubs with quantifiers are generated. The types of the quantified variables range over the values in PRIM-BDTYPES. With PRIMSUB-METHOD PR89 : This flag is ignored. Primsubs of the form "exists x . literal" and "forall x . literal" will be generated. With PRIMSUB-METHOD PR93 : At depth 1, a single quantifier is introduced, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula containing (N-1) conjunctions {disjunctions} of (N-2) disjunctions {conjunctions}. With PRIMSUB-METHOD PR95 : At depth 1, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula with between MIN-PRIM-LITS and MAX-PRIM-LITS literals, with all combinations of connectives between them. With PRIMSUB-METHOD PR97 : At depth N>0, we have (N-1) quantifiers ranging over each subformula taken from the etree which contains between MIN-PRIM-LITS and MAX-PRIM-LITS literals. You can see these subformulas by doing ETP from the MATE top level. With PRIMSUB-METHOD PR97A : As in PR97, but all substitutions are in negation normal form. With PRIMSUB-METHOD PR97B : The substitutions from PR97A and PR95 are interleaved. The order is determined firstly by the number of literals, then by the number of quantifiers, and lastly with PR97 substs taking precedence over PR95. With PRIMSUB-METHOD PR97C : If set to N, all primsubs will have < N quantifiers.It takes values of type POSINTEGER and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, PRIMSUBS. The default value is 1.

MAX-PRIM-LITS Maximum no. of literals allowed in a primsub. Does not apply for PRIMSUB-METHOD PR89 or PR93. See the help message for MIN-PRIM-DEPTH, which explains how primsubs are generated.It takes values of type POSINTEGER and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9,

MS90-3, MS89, MS88, PRIMSUBS. The default value is 4.

MIN-PRIM-DEPTH

Minimum depth at which primsubs with quantifiers are generated. The types of the quantified variables range over the values in PRIM-BDTYPES. With PRIMSUB-METHOD PR89 : This flag is ignored. Primsubs of the form "exists x . literal" and "forall x . literal" will be generated. With PRIMSUB-METHOD PR93 : At depth 1, a single quantifier is introduced, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula containing (N-1) conjunctions {disjunctions} of (N-2) disjunctions {conjunctions}. With PRIMSUB-METHOD PR95 : At depth 1, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula with between MIN-PRIM-LITS and MAX-PRIM-LITS literals, with all combinations of connectives between them. With PRIMSUB-METHOD PR97 : At depth N>0, we have (N-1) quantifiers ranging over each subformula taken from the etree which contains between MIN-PRIM-LITS and MAX-PRIM-LITS literals. You can see these subformulas by doing NAME-PRIM from the MATE top level. With PRIMSUB-METHOD PR97A : As in PR97, but all substitutions are in negation normal form. With PRIMSUB-METHOD PR97B : The substitutions from PR97A and PR95 are interleaved. The order is determined firstly by the number of literals, then by the number of quantifiers, and lastly with PR97 substs taking precedence over PR95. With PRIMSUB-METHOD PR97C : If set to N, the number of quantifiers in any primsub will be >= N-1.It takes values of type POSINTEGER and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, PRIMSUBS. The default value is 1.

MIN-PRIM-LITS Minimum no. of literals allowed in a primsub. Does not apply for PRIMSUB-METHOD PR89 or PR93. See the help message for MIN-PRIM-DEPTH, which explains how primsubs are generated.It takes values of type POSINTEGER and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, PRIMSUBS. The default value is 2.

NEG-PRIM-SUB When T, one of the primitive substitutions will introduce negation.It takes values of type BOOLEAN and belongs to subjects PRIMSUBS. The default value is NIL.

PR97C-MAX-ABBREVS

The maximum number of abbreviations that may appear in a PR97C primsub.It takes values of type POSINTEGER and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, PRIMSUBS. The default value is 1.

PR97C-PRENEX If T, PR97C generates substitutions in prenex normal form. If NIL, it doesn't.It takes values of type BOOLEAN and belongs to subjects IMPORTANT, MS93-1, MS92-9, MS91-7, MS91-6, MS90-9, MS90-3, MS89, MS88, PRIMSUBS. The default value is T.

PRIM-BDTYPES List of types of quantified variables used to contruct primitive substitutions. This list will always be used when constructing primitive substitutions interactively, but see the flag PRIM-BDTYPES-AUTO for more information on the types that will be used by automatic search procedures.It takes values of type TYPESYMLIST-NIL and belongs to subjects IMPORTANT, PRIMSUBS. The default value is ("I").

PRIM-BDTYPES-AUTO

Has five possible values: REPLACE, REPLACE-SUB, APPEND, APPEND-SUB and IGNORE. Determines how the procedures that use primitive substitutions handle the flag PRIM-BDTYPES, as follows: REPLACE -- the value of PRIM-BDTYPES will be changed to an automatically-generated list of all the primitive types used in the gwff to be proven. REPLACE-SUB -- as for replace, except that the list will be of all the subtypes of the types that appear in the gwff. APPEND -- the same list is calculated as for REPLACE, but instead of replacing the current setting of PRIM-BDTYPES it will be appended to it. APPEND-SUB -- the same list is calculated as for APPEND, but instead of replacing the current setting of PRIM-BDTYPES it will be appended to it. IGNORE -- no list will be generated, and the user's setting of PRIM-BDTYPES will be left intact.It takes values of type SYMBOL and belongs to subjects IMPORTANT, PRIMSUBS. The default value is REPLACE.

PRIM-PREFIX Prefix for weak labels associated with primitive substitutions.It takes values of type SYMBOL and belongs to subjects PRIMSUBS. The default value is PRIM.

PRIMSUB-METHOD

Takes one of the values PR89, PR93, PR95, PR97, PR97A, PR97B. This determines how primsubs will be generated, in conjunction with MAX-PRIM-DEPTH, MIN-PRIM-DEPTH, MAX-PRIM-LITS and MIN-PRIM-LITS. With PRIMSUB-METHOD PR89 : Primsubs of the form "exists x . literal" and "forall x . literal" will be generated. With PRIMSUB-METHOD PR93 : For all integers from MIN-PRIM-DEPTH to MAX-PRIM-DEPTH: At depth 1, a single quantifier is introduced, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula containing (N-1) conjunctions {disjunctions} of (N-2) disjunctions {conjunctions}. With PRIMSUB-METHOD PR95 : For all integers from MIN-PRIM-DEPTH to MAX-PRIM-DEPTH: At depth 1, as in PR89. At depth N>1, we have (N-1) quantifiers ranging over a formula with between MIN-PRIM-LITS and MAX-PRIM-LITS literals, with all combinations of connectives between them. With PRIMSUB-METHOD PR97 : For all integers from MIN-PRIM-DEPTH to MAX-PRIM-DEPTH: At depth N>0, we have (N-1) quantifiers ranging over each subformula taken from the etree which contains between MIN-PRIM-LITS and MAX-PRIM-LITS literals. You can see these subformulas by doing NAME-PRIM from the MATE top level. (Note: both the instantiated and uninstantiated versions of each definition are used.) With PRIMSUB-METHOD

PR97A : As in PR97, but all substitutions are in negation normal form. With PRIMSUB-METHOD PR97B : The substitutions from PR97A and PR95 are interleaved. The order is determined firstly by the number of literals, then by the number of quantifiers, and lastly with PR97 substs taking precedence over PR95. With PRIMSUB-METHOD PR97C : Using the connectives AND and OR, and the quantifiers EXISTS and FORALL (ranging over variables of types PRIM-BDTYPES), and also using any abbreviations or equalities that occur in the gwff to be proven, primsubs are built up using the bounds given by MIN- and MAX-PRIM-LITS and MIN- and MAX-PRIM-DEPTH. See also PR97C-PRENEX and PR97C-MAX-ABBREVS.It takes values of type SYMBOL and belongs to subjects `IMPORTANT`, `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `PRIMSUBS`. The default value is `PR93`.

10.40. Miscellaneous

REWRITE-EQUIVS

This chooses one of the two ways of constructing an etree from an equivalence A EQUIV B: 1 chooses the option with the fewest vertical paths (positive: A AND B OR ~A AND ~B negative: A IMPLIES B AND B IMPLIES A) 2 chooses the option with the fewest horizontal paths (negative: A AND B OR ~A AND ~B positive: A IMPLIES B AND B IMPLIES A) 3 behaves as for 2 except for the first equivalence it finds, when it behaves as for 1. (This means that a gwff which is a quantified equivalence will produce an etree which can be split.) 4 always chooses A IMPLIES B AND B IMPLIES A 5 always chooses A AND B OR ~A AND ~B Any other setting will behave like 1.

This does not work with MIN-QUANTIFIER-SCOPE T; in that case, etrees will be constructed as in case 1, regardless of the setting of this flag.It takes values of type POSINTEGER and belongs to subjects `MS98-1`, `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `IMPORTANT`, `MATING-SEARCH`. The default value is `1`.

10.41. RuleP

RULEP-MAINFN The main function used for RULEP. Defaults to RULEP-DELUXE, in which case RULEP will find a minimal subset of the support lines which suffices to justify the planned line. If set to RULEP-SIMPLE, RULEP will merely check that the planned line follows from the support lines that are specified by the user.It takes values of type RULEP-MAINFN-TYPE and belongs to subjects `RULES-MOD`. The default value is `RULEP-DELUXE`.

RULEP-WFFEQ The wffop used for testing whether two wffs are equal when checking RULEP and propositional mating search.It takes values of type SYMBOL and belongs to subjects `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `MATING-SEARCH`, `JFORMS`. The default value is `WFFEQ-AB`.

10.42. Skolemizing

NAME-SKOLEM-FN

Name of the functions which names a Skolem function.It takes values of type SYMBOL and belongs to subjects `WFF-PRIMS`. The default value is `NAME-SKOLEM-CAP`.

10.43. Quantifiers

UI-HERBRAND-LIMIT

Maximum number of times to apply ui-herbrand-tac to the same universally-quantified formula.It takes values of type POSINTEGER and belongs to subjects `TACTICS`. The default value is `3`.

10.44. Auxiliary

USE-RULEP When true, indicates that RuleP should be used when possible in translating from expansion proof to natural deduction proof.It takes values of type BOOLEAN and belongs to subjects `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `TACTICS`, `ETR-NAT`, `MATING-SEARCH`. The default value is `T`.

USE-SYMSIMP When true, indicates that symmetric simplification should be used when possible in translating from expansion proof to natural deduction proof. Consult Pfenning's thesis for a description of symmetric simplification.It takes values of type BOOLEAN and belongs to subjects `MS93-1`, `MS92-9`, `MS91-7`, `MS91-6`, `MS90-9`, `MS90-3`, `MS89`, `MS88`, `TACTICS`, `ETR-NAT`, `MATING-SEARCH`. The default value is `T`.

10.45. Events

ADVICE-ASKED-ENABLED
: If NIL, recording events of type ADVICE-ASKED is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

ADVICE-FILE
: The file recording advice.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.advice".

COMMAND-ENABLED
: If NIL, recording events of type COMMAND is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

COMMAND-FILE
: The file recording commands.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.command".

DONE-EXC-ENABLED
: If NIL, recording events of type DONE-EXC is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

ERROR-ENABLED
: If NIL, recording events of type ERROR is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

ERROR-FILE
: The file recording the events of errors.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.error".

EVENT-CYCLE
: The indivisible unit in number of inputs. When WRITE-WHEN for an EVENT is ‘n’, the event info will be written every n * event-cycle inputs. n=0 means don’t write.It takes values of type INTEGER+ and belongs to subjects EVENTS. The default value is 5.

EVENTS-ENABLED
: If nil, all events are disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

INPUT-ERROR-ENABLED
: If NIL, recording events of type INPUT-ERROR is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

INPUT-ERROR-FILE
: The file recording illegal inputs caught by TPS.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.ierror".

PROOF-ACTION-ENABLED
: If NIL, recording events of type PROOF-ACTION is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

PROOF-FILE
: The file recording started and completed proofs.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "/users/andrews/etps3.proof".

QUIET-EVENTS
: If T, no message will be given when events are written.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

RULE-ERROR-ENABLED
: If NIL, recording events of type RULE-ERROR is disabled.It takes values of type BOOLEAN and belongs to subjects EVENTS. The default value is T.

RULE-ERROR-FILE
: The file recording illegal rules caught by TPS.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.rerror".

SCORE-FILE
: The file recording completed exercises.It takes values of type FILESPEC and belongs to subjects EVENTS. The default value is "etps3.scores".

10.46. Grader

CAL-PERCENTAGE
: The program calculates percentage based on total scores if the value of this variable is T.It takes values of type BOOLEAN and belongs to subjects GR-MISC. The default value is NIL.

COURSE-NAME
: Name of the course. Also used as a suffix for various files which are created or modified by the grading package.It takes values of type STRING and belongs to subjects GR-MISC. The default value is "course".

DEFAULT-PENALTY-FN
: Default penalty function for late exercises. The default is no-penalty which doesn’t take any points off.It takes

values of type FUNCTION and belongs to subjects GR-MISC. The default value is NO-PENALTY.

DROP-MIN When calculating totals, the program drops the minimum scores on each of the items in this list.It takes values of type CONSP1 and belongs to subjects GR-MISC. The default value is NIL.

DUE-DATE-FLAG If this flag is nil, the user is not prompted for due dates (in the command ETPS-GRADE) and it's assumed that all exercises were submitted in time.It takes values of type BOOLEAN and belongs to subjects GR-MISC. The default value is T.

ETPS-FILE Name of the file which contains ETPS records.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

GRADE-DIR Name of the directory in which the grader files are to be found, or "" for the directory from which grader was started. This name should end with a backslash, as in : "/usr/teacher/course-grades/". When this flag is changed, all of the other filenames will change with it. Note that in old versions of CMU lisp, the "" option will not work properly.It takes values of type STRING and belongs to subjects GR-FILENAMES. The default value is "".

GRADE-FILE Name of the GRADE-FILE.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

LETTER-GRADE-FILE
Name of the file which will contain letter grades.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

LETTER-GRADE-FLAG
The program creates a separate file containing letter grades if the value of this variable is true.It takes values of type BOOLEAN and belongs to subjects GR-MISC. The default value is T.

NEW-ITEM The list of new items to be calculated when calculating totals. See the manual for more details.It takes values of type CONSP1 and belongs to subjects GR-MISC. The default value is NIL.

OLD-GRADE-FILE
Name of the back-up GRADE-FILE.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

OLD-TOTALS-GRADE-FILE
Name of the back-up TOTALS-GRADE-FILE .It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

PATCH-FILE Name of the file containing changes to the grader core image.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "grader.patch".

PRINT-N-DIGITS The number of digits to be printed after the decimal.It takes values of type INTEGER+ and belongs to subjects GR-MISC. The default value is 0.

STATISTICAL-OPTIONS
List of statistical data to be calculated. Currently the program can calculate mean, median , standard deviation. The default is (-mean- -median- -sdev-).It takes values of type CONSP1 and belongs to subjects GR-MISC. The default value is (-MEAN- -MEDIAN- -SDEV-).

TOTALS-GRADE-FILE
Name of the file which will contain totals.It takes values of type FILESPEC and belongs to subjects GR-FILENAMES. The default value is "".

10.47. Maintenance

COMPILED-EXTENSION
The extension of compiled files in TPS3.It takes values of type STRING and belongs to subjects MAINTAIN. The default value is "fasl".

EXPERTFLAG If T, arbitrary Lisp expression may be evaluated on top levels.It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is NIL.

INIT-DIALOGUE If T, the value of INIT-DIALOGUE-FN will be called on startup after the INI file has been read and the terminal is initialized.It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is NIL.

INIT-DIALOGUE-FN
The value of this flag is a function of no arguments, which will be called after the INI file has been read, if the flag INIT-DIALOGUE is T. It may be used to set the terminal type correctly, load some libraries, if the user wishes, or even decide between expert and non-expert modes. The default function does nothing; the function INIT-DEFINE-MY-DEFAULT-MODE defines a mode called MY-DEFAULT-MODE containing the state of all the system's flags at the point immediately after the INI file is read.It takes values of type ANYTHING and belongs to subjects MAINTAIN. The default value is INIT-DIALOGUE-DEFAULT-FN.

LISP-IMPLEMENTATION-TYPE

Tells what Common Lisp we are running on. Initialized when TPS starts up. Can't be changed.It takes values of type STRING and belongs to subjects SYSTEM. The default value is "".

LOAD-WARN-P If T, warning messages will be printed when redefining TPS-objects while loading a file or fetching library objects.It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is T.

MACHINE-INSTANCE
Tells what particular machine we are running on. Initialized when TPS starts up. Can't be changed.It takes values of type STRING and belongs to subjects SYSTEM. The default value is "".

MACHINE-TYPE Tells what hardware that we are running on. Initialized when TPS starts up. Can't be changed.It takes values of type STRING and belongs to subjects SYSTEM. The default value is "".

NEWS-DIR The directory with the NEWS and NOTE files.It takes values of type DIRSPEC and belongs to subjects MAINTAIN. The default value is "".

READ-LLOAD-SOURCES-P
If T while LLoading, one can later Ledit compiled functions.It takes values of type BOOLEAN and belongs to subjects MAINTAIN. The default value is T.

SAVE-FILE The name of the file in which to save the core-image for TPS3.It takes values of type FILESPEC and belongs to subjects MAINTAIN. The default value is "tps3.exe".

SHORT-SITE-NAME
Tells what site we are running at. Initialized when TPS starts up. Can't be changed.It takes values of type STRING and belongs to subjects SYSTEM. The default value is "".

SOURCE-EXTENSION
The extensions (:type) of source files in TPS3.It takes values of type STRING and belongs to subjects MAINTAIN. The default value is "lisp".

SOURCE-PATH A list of pathnames with source files for TPS3.It takes values of type DIRSPECLIST and belongs to subjects MAINTAIN. The default value is ().

TEST-MODIFY A string which will be evaluated in exactly the same way as an alias. May contain any valid lisp commands, and will be evaluated after setting the mode during tps-test. So, for example, setting it to "(set-flag 'skolem-default nil) (when search-time-limit (setq search-time-limit (* 2 search-time-limit))) (when max-search-limit (setq max-search-limit (* 2 max-search-limit)))" would make tps-test changed SKOLEM-DEFAULT to NIL and double the time limits before each search.It takes values of type STRING and belongs to subjects MAINTAIN. The default value is "".

TEST-THEOREMS
A list of pairs; the first of each pair is the name of a theorem; the second is the name of a mode. Used by the command TPS-TEST. The default setting is a sample list of two standard TPS exercises, both to be run in mode ML (also standard in TPS). If you set this flag yourself, beware of unexported symbols --- which is to say, make sure that the symbols you use are all in the USER package (this is particularly necessary if you are using library theorems which are not yet loaded into TPS, or they may end up interned in the wrong package). If in doubt, put "USER::" before all symbols, thus:

(setq test-theorems '((user::thm30 . user::mode-thm30) (user::x2112 . user::ml)))

You can use the flag TEST-MODIFY to alter modes on the fly as TPS-TEST runs. See TEST-MODIFY for more information.It takes values of type SYMBOLPAIRLIST and belongs to subjects MAINTAIN. The default value is ((X2106 ML) (X2108 ML)).

10.48. Modules

TEXFORMAT HPD for a horizontal path diagram (p.d.) of the positive wff. VPD for a vertical p.d. of the negated wff. VPP (or anything else) for a vertical p.d. of the positive wff.It takes values of type SYMBOL and belongs to subjects JFORMS. The default value is VPP.

VPFORM-TEX-MAGNIFICATION
The magnification factor to use for TeX files containing vpforms. This has two possible settings: if it is lower than 10, then it is used in the form \magnification=\magstepN Roughly, 0 = 10pt, 1 = 12pt, 2 = 14pt, 3 = 17pt, 5 = 25pt.

Otherwise, it is used in the form \magnificationN, in which case 1000 corresponds to "normal size" (12pt), 800 is 80%, 1200 is 120%, and so on.It takes values of type INTEGER+ and belongs to subjects JFORMS. The default value is 1000.

VPFORM-TEX-NEST
Maximal number of boxes to nest in path diagrams for TeX. 0 means not to break into boxes.It takes values of type INTEGER+ and belongs to subjects JFORMS. The default value is 4.

VPFORM-TEX-PREAMBLE

The string to be put at the beginning of a TeX file containing vpforms.It takes values of type STRING and belongs to subjects JFORMS. The default value is "".

10.49. Rules object

BUILD-MATCH If T, <rule>-MATCH functions for use with SUGGEST will be built.It takes values of type BOOLEAN and belongs to subjects RULES-PACK. The default value is T.

HLINE-JUSTIFICATION

The justification for hlines, if TREAT-HLINES-AS-DLINES is NIL.It takes values of type STRING and belongs to subjects RULES-OBJECT. The default value is "Hyp".

TREAT-HLINES-AS-DLINES

If T, hlines may have multiple hypotheses and a justification, if NIL, hlines can only have one hypothesis (itself) and 'Hyps' as justification.It takes values of type BOOLEAN and belongs to subjects RULES-OBJECT. The default value is T.

10.50. Unclassified

DNEG-IMITATION

Determine when to produce imitation terms that contain double negations. Only applies in UN88 when REDUCE-DOUBLE-NEG is T (in UN88 otherwise, it is implicitly set to ALWAYS; in UN90 it is implicitly set to CONST-FLEX). When TPS mates two flexible literals f and g, it adds (f . ~g) as a dpair. Because it may really have needed (g . ~f), we allow imitation terms to contain double negations even if REDUCE-DOUBLE-NEG is T. The options are as follows: ALWAYS always allows double negations to be used. CONST forbids them for dpairs of the form (f . ~G), where G is a constant, but allows them otherwise. FLEX forbids them for (f . ~g) if g was created by a double negation in the first place (this prevents endless cycles), but allows them otherwise. CONST-FLEX forbids them in the two cases for CONST and FLEX, but allows them otherwise. NEVER forbids them outright.It takes values of type SYMBOL and belongs to subjects UNIFICATION. The default value is CONST-FLEX.

MAX-SUBSTS-PROJ

The total number of projection substitutions allowed for any given variable. See also MAX-SUBSTS-VAR and MAX-SUBSTS-PROJ-TOTAL. This applies to higher-order unification (UN88 or UN90) only.It takes values of type NULL-OR-INTEGER and belongs to subjects UNIFICATION. The default value is NIL.

MAX-SUBSTS-PROJ-TOTAL

The total number of projection substitutions allowed for any given dpairset. See also MAX-SUBSTS-VAR and MAX-SUBSTS-PROJ. This applies to higher-order unification (UN88 or UN90) only.It takes values of type NULL-OR-INTEGER and belongs to subjects UNIFICATION. The default value is NIL.

MAX-SUBSTS-QUICK

When NIL, quick unification is governed by the MIN-QUICK-DEPTH flag, and only minimal amounts of MAX-SUBSTS checking are done during quick unification. When MIN-SUBSTS-QUICK is a positive integer, quick unification (i.e. partial unification of a possible connection) is considered as a special case of normal unification, with MAX-SUBSTS-VAR temporarily equal to the value of MAX-SUBSTS-QUICK. When MIN-SUBSTS-QUICK is 0, quick unification goes down as far as it can until it is forced to either branch or violate MAX-SUBSTS-VAR. (This is almost equivalent to MAX-SUBSTS-QUICK NIL and MIN-QUICK-DEPTH 1.)

Note: non-NIL values of MAX-SUBSTS-QUICK only take effect if MAX-SUBSTS-VAR is also non-NIL. In this case, other flags will also be affected, as follows: APPLY-MATCH will be ignored (the matching routine that is used will be a variant of APPLY-MATCH-ALL-FRDPAIRS) COUNTSUBS-FIRST and STOP-AT-TSN will be T. SUBSUMPTION-CHECK, UNIF-COUNTER and UNIF-TRIGGER will be NIL. UNI-SEARCH-HEURISTIC will be BREADTH-FIRST. MIN-QUICK-DEPTH and MAX-UTREE-DEPTH will be ignored.It takes values of type NULL-OR-INTEGER and belongs to subjects MS98-1, IMPORTANT, UNIFICATION. The default value is NIL.

MAX-SUBSTS-VAR

The maximum number of substitutions allowed for any given free variable in a dpairset. This is cumulative (i.e. if an old variable f is replaced by h1, which is in turn replaced by h2, that counts as two substitutions for f). Only projections or imitations are counted; eliminating substitutions are not. See also MAX-SUBSTS-PROJ and MAX-SUBSTS-PROJ-TOTAL. This applies to higher-order unification (UN88 or UN90) only.It takes values of type NULL-OR-INTEGER and belongs to subjects MS98-1, IMPORTANT, UNIFICATION. The default value is NIL.

NUM-OF-DUPS Max number of duplications allowed on any path in search procedures using path-focused duplication. This flag may be set to 0.It takes values of type INTEGER+ and belongs to subjects MS98-1, MS93-1, MS92-9,

MS91-7, MS90-9, MS90-3, MATING-SEARCH, IMPORTANT. The default value is 2.

PRIMSUB-VAR-SELECT
If T, primsubs will only be applied to those variables which occur both negatively and positively as the head variable of some leaves in the current eproof. If NIL, primsubs will be applied to any variable which occurs either negatively or positively or both, anywhereIt takes values of type BOOLEAN and belongs to subjects PRIMSUBS. The default value is T.

UNIF-COUNTER If this flag is non-zero, PP* will be called to print out information about the current unification tree at regular intervals. This flag determines the length of the intervals, measured by the number of calls to the unification procedure. The amount of information is determined by the setting of UNIF-COUNTER-OUTPUT. If the flag is set to 0, this feature will be turned off. This flag only applies in UN88 unification.It takes values of type INTEGER+ and belongs to subjects UNIFICATION. The default value is 0.

UNIF-COUNTER-OUTPUT
See UNIF-COUNTER and UNIF-TRIGGER for the use of this flag. Settings are: 0: Print the entire tree in flat format with details. (PALL) 1: Print the entire tree in flat format without details. (PALL) 2: Print the tree in tree format with subs. (UTREE*) 3: Print the tree in tree format without subs. (UTREE*) 4: Print just the subs and details in flat format. (UTREE) 5: Print just the subs in flat format. (UTREE) 6: Print full details of the last node. (P and PP*) 7: Print some details of the last node. (P and PP) 8: Print the last node and its properties only. 9: Print the statistics for the tree so far. (STATS) 10: Print the average values for STATS, after a mating is found. This flag only applies in UN88 unification.It takes values of type INTEGER+ and belongs to subjects UNIFICATION. The default value is 0.

UNIF-TRIGGER If this flag is non-NIL, PP* will be called to print out information about the current unification tree after certain events (compare UNIF-COUNTER). Settings are: NIL: Print nothing. UTREE-END: Printout whenever a tree has come to an end (either failure or success; NB UNIF-COUNTER-OUTPUT 6 or 7 will not work with this setting.) UTREE-END1: As UTREE-END, but also gives output when quick unification ends a tree without completing it. UTREE-BEGIN: Printout the root node when unification is first called. PROPS-CHANGE: Printout whenever the properties of a node are different from those of its parent. (Best used with UNIF-COUNTER-OUTPUT 6 or 7.) The amount of information is determined by the setting of UNIF-COUNTER-OUTPUT. If the flag is set to NIL, this feature will be turned off. This flag only applies in UN88 unification.It takes values of type SYMBOL and belongs to subjects UNIFICATION. The default value is NIL.

10.51. Library

ADD-SUBDIRECTORIES
When restoring the library index, search the directories in DEFAULT-LIB-DIR and BACKUP-LIB-DIR for subdirectories which also contain library files, and add these to the flags. This flag only works for Allegro, CMU, Kyoto and Lucid Common Lisps.It takes values of type BOOLEAN and belongs to subjects LIBRARY. The default value is T.

BACKUP-LIB-DIR
The list of all backup directories of library files. These should be directories to which the user has read access. No attempt will be made to write to a directory on this list. See also DEFAULT-LIB-DIR and SHOW-ALL-LIBOBJECTS.It takes values of type DIRSPECLIST and belongs to subjects LIBRARY. The default value is ().

DEFAULT-LIB-DIR
The list of writeable directories containing library files. All of the directories in this list ought to be library directories to which the user has write access. See also BACKUP-LIB-DIR and SHOW-ALL-LIBOBJECTS.It takes values of type DIRSPECLIST and belongs to subjects LIBRARY. The default value is ().

DEFAULT-LIBFILE-TYPE
The default value for the extension of library files.It takes values of type STRING and belongs to subjects LIBRARY. The default value is "lib".

DEFAULT-LIBINDEX-TYPE
The default value for the extension of library index files.It takes values of type STRING and belongs to subjects LIBRARY. The default value is "rec".

LIB-KEYWORD-FILE
Name of the file containing acceptable keywords for the library.It takes values of type FILESPEC and belongs to subjects LIBRARY. The default value is "keywords.rec".

LIB-MASTERINDEX-FILE
Name of the file containing index of entries in the library.It takes values of type FILESPEC and belongs to subjects LIBRARY. The default value is "libindex.rec".

RECORDFLAGS List of flags to be saved when using the mateop DATEREC.It takes values of type TPSFLAGLIST and belongs to subjects MATING-SEARCH, LIBRARY. The default value is ().

REMOVE-TRAILING-DIR

If T, the parts of the directory specification that are the same for all library files will be removed before printing. If NIL, the full directory will be printed.It takes values of type BOOLEAN and belongs to subjects `LIBRARY`. The default value is `T`.

SHOW-ALL-LIBOBJECTS

When loading an object, if there are multiple objects of that name and type, when NIL then accept the first object found (searching DEFAULT-LIB-DIR and then BACKUP-LIB-DIR in order). When T, show a list of all the objects and ask the user to choose.It takes values of type BOOLEAN and belongs to subjects `LIBRARY`. The default value is `T`.

10.52. Bugs

DEFAULT-BUG-DIR

If USE-DEFAULT-BUG-DIR is T, this is the default value for the directory where bugs generated by BUG-SAVE will be stored, and the first directory that will be searched by BUG-RESTORE. If USE-DEFAULT-BUG-DIR is NIL, this flag is ignored, and bugs will be saved like normal library objects, in the directories listed in DEFAULT-LIB-DIR.It takes values of type DIRSPEC and belongs to subjects `LIBRARY`. The default value is `""`.

USE-DEFAULT-BUG-DIR

Determines whether or not to use the directory given by DEFAULT-BUG-DIR for saving. If T, bugs are saved to and restored from DEFAULT-BUG-DIR, otherwise they aren't. See DEFAULT-BUG-DIR.It takes values of type BOOLEAN and belongs to subjects `LIBRARY`. The default value is `T`.

Index

Table of Contents