Limette User Manual

Karl H. Schäfer

Universität Karlsruhe (TH)
Institut für Algorithmen und Kognitive Systeme
Am Fasanengarten 5, 76131 Karlsruhe, Germany

Phone: ++49 721 6091-472   Fax: ++49 721 6091-413

29 January 1997

1 Terms and Formulae

1.1 Bodies

A Atom $p(T_1, \ldots, T_n)$. An ellipse $p(T_1, \ldots, T_i\boxed{\ldots})$ looks up the actual arity $n \geq i$ of p (which must be unique) and adds variable arguments $v_{i+1}, \ldots, v_n$.

V A variable body corresponds to **call**(V).

B_1 **,@**ν B_2 Conjunction $\wedge_\nu$, $\nu \in \{\mathbf{W}, \mathbf{M}, \mathbf{S}\}$ "B_1 and B_2".

B_1 **;@**ν B_2 Disjunction $\vee_\nu$, $\nu \in \{\mathbf{W}, \mathbf{M}, \mathbf{S}\}$, "$B_1$ or B_2".

not@ν B Constructive negation $\neg_\nu$, $\nu \in \{\mathbf{W}, \mathbf{M}, \mathbf{S}\}$.

I **!** B Universal temporal quantifier $\Box_I$ "always within I".

I **?** B Existential temporal quantifier $\Diamond_I$ "sometime within I".

I k**%** B Temporal frequency operator $\Box_I^k$ "at least k% of the time within I".

B_1 **since**I B_2 Temporal operator $\mathcal{S}_I$ "B_2 sometime in the I-past and B_1 since then".

B_1 **until**I B_2 Temporal operator $\mathcal{U}_I$ "B_2 sometime in the I-future and B_1 until then".

P | B Fuzzy reduction $\downarrow_P$ for $0 < P \leq 1$ "truth value is at least P". Fuzzy intensification $\uparrow_{P^{-1}}$ for $P > 1$ "truth value is P^{-1} fraction".

A \ **{** B_1 $\boxed{\cdot}$ **...** $\boxed{\cdot}$ B_n **}** Fuzzy modifier $A(B_1, \ldots, B_n)$ (like *very* or *some*). User-defined by a predicate $A/n{+}1$ such that $A(\mu, \mu_1, \ldots, \mu_n)$ and $A(\eta, \eta_1, \ldots, \eta_n)$.

A**#** V**:** B Defuzzifier A (like *max* or *avg*). Determines the best value for V (variable list) such that B. User-defined by a predicate $A/2$ such that $A([[\eta_1 \mid T_1], \ldots], T_{\text{best}})$.

V_1 **{**B**}** V_2 *TAB* calculus triple "μ $\{B\}$ η". Either sets μ (V_1 is float) or reads μ (V_1 is variable). Either sets η (V_2 is float) or reads η (V_2 is variable).

\$$S$ B Strategy operator. Evaluates B with the given strategy S: **D**: depth, **B**: breadth, **R**: random, **A**: atomic.

collect B Offline-Filter. Enumerates all derivations of B and optimizes that set for size equivalently by removing subsumed derivations and combining derivations.

event B Same as $\Diamond_{(-\infty,+\infty)}$ "sometime".

event past B Same as $\Diamond_{(-\infty,0]}$ "sometime in the past".

event future B Same as $\Diamond_{[0,+\infty)}$ "sometime in the future".

exists V**:** B Existential quantor $\exists V$ for single variable V or variable list V.

filter B Online-Filter. Removes derivations subsumed by previous derivations.

findall$[L]$ T **-:** B Implicit set $L = \{T \mid B\}$. Creates the set of all solutions of B.

forall V**:** B_1 **-:** B_2 Universal quantor $\forall V(B_1 \rightarrow B_2)$ "all V such that B_1 fulfill B_2".

init B Temporal operator "at the initial instant $0 \in \mathcal{T}$".

next B Same as $\Diamond_{[1,1]}$ "at next time instant".

prev B Same as $\Diamond_{[-1,-1]}$ "at previous time instant".

1.2 Rules

?- B Query B for immediate execution while loading a file of rules.

A Head $p(T_1, \ldots, T_n)$ of a rule.

R **:-** B Nested "if–then" rule "if B then R".

I **!** R Temporally restricted rule "R always within I".

$P \mid R$ Rule of reduced truth "R is at least P-true".

$[T_1$ ::= $T_2]$ Term rewrite rule head "T_1 (pattern) replaceable by T_2".

always R Same as $\Box_{(-\infty,+\infty)}$, "always".

always future R Same as $\Box_{[0,+\infty)}$, "always in the future".

always past R Same as $\Box_{(-\infty,0]}$, "always in the past".

next R Same as $\Box_{[1,1]}$, "at next time instant".

prev R Same as $\Box_{[-1,-1]}$, "at previous time instant".

1.3 Gates

A Atom $p(T_1, \ldots, T_n)$ to open or close.

V Variable atom that represents all atoms of the module.

G_1 **,** G_2 Combination of gates that might share operators.

I **!** G Temporal completeness quantification $\Box_I$ "complete always within I".

$P \mid G$ Fuzzy completeness assertion $\downarrow_P$ "complete for truth values of at least P".

always G Same as $\Box_{(-\infty,+\infty)}$ "always complete".

always future G Same as $\Box_{[0,+\infty)}$ “complete always in the future”.

always past G Same as $\Box_{(-\infty,0]}$ “complete always in the past”.

next G Same as $\Box_{[1,1]}$ “complete at next time instant”.

prev G Same as $\Box_{[-1,-1]}$ “complete at previous time instant”.

1.4 Terms

A Compound term $f(T_1, \ldots, T_n)$ with functor f.

c Symbolic constant $c \in F$.

´ . . . ´ Quoted constant (symbol of arbitrary characters).

V Variable $V \in \mathcal{V}$.

n Integer constant $n \in \mathcal{Z}$.

n_1 **:** n_2 Interval constant $[n_1, n_2] \subseteq \mathcal{Z}$.

mt Empty interval $[c^+, c^-] = \emptyset$ $(c^- < c^+)$.

-i Constant $-\infty$ “minus infinity”.

+i Constant $+\infty$ “plus infinity”.

f Floating-point constant $f \in \mathcal{R}$.

nan Floating-point constant “Not-A-Number”.

[] Empty list.

$[H \mid T]$ Nonempty list with head H and tail T.

$[T_1, T_2, \ldots, T_n]$ Nonempty list of elements T_i $(1 \leq i \leq n)$

”. . .” Character string (list of integer ASCII codes).

$+\ T$ Unary plus.

$-\ T$ Unary minus (arithmetic negation).

$T_1 + T_2$ Arithmetic addition.

$T_1 - T_2$ Arithmetic subtraction.

$T_1 \star T_2$ Arithmetic multiplication.

$T_1\ /\ T_2$ Arithmetic division.

T_1 **^** T_2 Arithmetic power.

`** $E$ **` Inline arithmetic expression for immediate evaluation.

{ T **}** Term that represents a formula (contains operators).

I **!** T Temporal term "value of T always within I".

I **?** T Temporal term "value of T sometime within I".

always T Same as $\Box_{(-\infty,+\infty)}$.

always future T Same as $\Box_{[0,+\infty)}$.

always past T Same as $\Box_{(-\infty,0]}$.

event T Same as $\Diamond_{(-\infty,+\infty)}$.

event future T Same as $\Diamond_{[0,+\infty)}$.

event past T Same as $\Diamond_{(-\infty,0]}$.

next T Same as $\Box_{[1,1]}$ or $\Diamond_{[1,1]}$.

prev T Same as $\Box_{[-1,-1]}$ or $Diamond_{[-1,-1]}$.

2 Commands

add(*file***)** Add all rules contained in the given Limette *file* (or file prefix) to the current module. The *file* will be autoassociated. Inline queries "`?-`" will be executed.

addhost(*host***)** Spawn a Limette server daemon on some *host* machine. Added hosts are ready to schedule remote derivations. Only one daemon per host is allowed.

asserta *rule* Add the given *rule* to the start of the current module.

assertz *rule* Add the given *rule* to the end of the current module.

assoc(*extension***,***shell_command***,***parameters***)** Add an association between the filename postfix *extension* to a *shell_command* like "`cmd %s %s`". The first `%s` is replaced by the default *parameters*, the second `%s` by the filename including the extension. For example, "**`assoc('.gz','gunzip %s %s','-c')`**" tells Limette how to read compressed files. The commands **add**, **import**, **include**, **load** use associated files.

author Print information about the authors of Limette.

cd(*directory***)** Change the current working *directory* (a relative path descriptor).

close *gate* Close all gates subsumed by *gate*, i.e. "$\downarrow_\kappa \Box_S A$" closes all gates "$\downarrow_{\kappa'} \Box_{S'} A\sigma$" where $\kappa \leq \kappa'$ and $S \supseteq S'$. Several gates may be combined using conjunction.

delete(*module****)** Remove the given *module* (or the current module if missing) from the module database including all contained rules.

delete(*predicate***,***arity***)** Remove all those rules from the current module that define the given *predicate* of the given *arity*.

delhost(*host***)** Kill the Limette server daemon on *host* machine. The daemon must have been spawned by **addhost** before.

dlist(*module****)** Print a list of native-language predicates and their arities defined in *module* (or the current module if missing).

dload(*file***,***module****)** Load a native-language module from a shared object *file* (without the "`.so`" extension) and name it *module*. Use *file* if *module* is missing.

dunload(*module***)** Unload a native-language *module* previously loaded by **dload**.

dynamic *predicate/arity* Define *predicate* of *arity* to be a dynamic predicate. A comma-separated list of predicates may be given. Dynamic predicates require open gates (see **open**, **close**) for their derivation.

enter(*module***)** Set *module* to be the current module. If an unloaded module is entered, it is defined to be empty.

factor(*module*⋆**)** Optimize the rules of *module* (current module if missing). Factorization tries to reduce the number of facts equivalently.

gc Perform a global garbage collection which returns allocated but unused memory to the operating system. Limette keeps deallocated memory for immediate reallocation.

get(*setting*⋆**)** Print the current value of some global parameter *setting*. Print all settings and values if *setting* is missing. See **set** for a list of settings and respective values.

glist(*module*⋆**)** List all open gates of *module* (current module if missing). This list must not be identical to the **open** commands, since the gate list is optimized for size.

history Print a list of all previously entered and un**kill**ed queries, together with their query index and their state of execution (ready, stopped, errorneous).

import(*file***)** Load the autoassociated *file* (or file prefix) as a separate module named *file*. This command may only occur in files to recursively load dependent modules.

include(*file***)** Paste the contents of the autoassociated *file* right in the place of the **include** command. Limited recursive inclusion is possible.

kill(*query_index*⋆**)** Remove a query from the **history** list. This also frees the memory allocated for it and its proof. Killed queries can neither be **resume**d nor **restart**ed.

list(*module*⋆**)** Print all rules of the given *module*, prefixed by their respective rule index (useful for **retract**). List the current module if *module* is missing.

list(*predicate***,***arity***)** List all rules defining *predicate* of *arity* inside the current module, prefixed by their rule index.

load(*file***,***parameters*⋆**)** Load a module from the autoassicated *file* (or file prefix). The module will be named *file*. If the module already exists, it will be overwritten. If *parameters* are given, they override the default parameters of the **assoc** command.

memo(*module*⋆**)** Turns on memoizing for all predicates defined in the given module.

memo(*predicate***,***arity***)** Turns on memoizing for the given predicate.

memos(*module*⋆**)** Print the list of all memoized predicates of the given module.

mlist(*module*⋆**)** Print all memo table entries of the given module.

mlist(*predicate***,***arity***)** Print the memo table of the given predicate.

mlookup(*gate*⋆**,***gate*⋆**)** Print all memo table entries such that the initial state matches the first gate (if given, all otherwise). Only print those respective final states that fit to the second gate (if given, all otherwise).

modules Print a sorted list of all nonempty modules (**load**ed or **enter**ed and **assert**ed).

open *gate* Adds *gate* to the gates of the current module. A gate "$\downarrow_\kappa \Box_S A$" allows all atomic queries "$\downarrow_{\kappa'} \Box_{S'} \tilde{\exists}(A\sigma)$" where $\kappa \leq \kappa'$ and $S \supseteq S'$ to be derived. Underivable queries are delayed until they become derivable later or derivation stops.

pack(*module**,*file***)** Save the loaded *module* (the current module if missing) to a *file* (full file name required) in binary format. Packed files can be reloaded much faster than textfiles but cannot be edited. All load commands recognize the binary format.

pipe(*shell_command*,*module****)** Load the *module* (the current module if missing) with the standard output of *shell_command*, which is issued to the operating system for execution. The output must be a Limette file, but the command might have additional side effects or create the output from other data like a preprocessor.

pwd Print the name of the current working directory.

queue(*query_index****)** Print a list of unprocessed queue elements of the indexed query, together with their type and state of execution. This command shows the pending instructions of interrupted queries as well as the delayed instructions refering to dynamic predicates that failed open gates.

quit Leave the shell, stop Limette processes, and return to the operating system.

rememo(*module****)** Clears all memo table entries for predicates of the given module.

rememo(*predicate*,*arity***)** Clears the memo table for the given predicate.

restart(*query_index****)** Enter some indexed query of the **history** list once more without retyping it. If the *query_index* is missing, the recently typed query is used.

resume(*query_index****)** Continue the execution of the indexed query (recent query if *query_index* is missing). Resumable queries must be ready for execution, i.e. either interrupted or tentatively terminated by dynamic predicates.

retract(*module**,*rule_index***)** Remove a single indexed rule inside the given *module* (the current module if missing). The rule indices are given by the **list** command or by the internal **retract** predicate.

rl(*bool***)** Turn the GNU Readline library `on` or `off`. This library allows to edit the command line using Emacs-like keystrokes, and to retrieve previously entered lines using the cursor keys. Without the library, the keyboard directly becomes standard input.

save(*module**,*file***)** Write a *module* (current module if missing) in its current state to the given *file* (full file name required) in text format.

set(*setting*,*value***)** Redefine the *value* of a global parameter *setting*.

alpha(*list***)** The coefficients of the global heuristic function used to measure prospective success of queue elements as a *list* "`[`*tree*`,`*timeset*`,`*subst*`,`*poss*`]`" of four floating-point numbers.

answer(*type***)** The *type* of answer generation:

formula	create answer formula from proof tree
cproof	print entire proof tree
qproof	print partial proof tree of query-related nodes
subst	print answer substitution
success	only report success (`yes` or `no`)

cycles(*bool***)** Turn cycle checking `on` or `off`.

debug(*bool***)** Turn the box-model debugger `on` or `off`.

defcheck(*bool***)** Either report an error if an undefined predicate is encountered (`on`), or simply fail (`off`) since no rule is present.

gates Selects the way of treating dynamic predicates that fail gates:

none	ignore gates
stop	stop derivation with the first failure
delay	delay failed queue elements and proceed

limits(*lower*,*higher***)** Set the *higher* limit that triggers the pruning of queue elements that gain a bad heuristic measure (maximum number of tolerable alternative proofs). Only the *lower* number of elements are left then for hysteresis to avoid high-frequency repetitive pruning.

memo Selects the operation of memoizing. Memoizing reminds results of previous goals and reuses them for subsequent goals subsumed by some reminded goal. If the program changes, the tables must be partially cleared. Automatic memoizing keeps track of necessary changes, while manual memoizing expects the user to clear entries explicitly by executing **rememo** commands.

off	no memoizing (PROLOG compatibility)
manual	memoizing on, manual table management
auto	memoizing on, automatic table management

narrow(*bool***)** Turn the term narrowing engine `on` or `off`. Narrowing refers to term rewrite rules "`::=`" to normalize argument terms of predicates.

nthreads(*integer* Set the number of concurrent tableau engine threads (only available for Multi-Threaded Limette). Up to *integer* many queue elements can be processed simultaneously. The number should be some small multiple of the number of processors locally available.

print(*type***)** Set the method to print terms:

normal	convential look (precedence, quotes on demand)
defaults	name default operator semantics
quotes	quote all functors
functors	use tupled syntax even for ⋆fix operators
internal	visualize the internal term data structure

queue Set the default strategy for the tableau engine:

depth	depth-first strategy
breadth	breadth first strategy
random	random-select strategy

stop(*integer*) Set the number of answers to give for queries without stopping and waiting for interaction (request to continue, stop, or kill). The value `+i` selects an infinite number (never stop, give all answers).

store(*type*) Select the memory deallocation strategy for proof trees. Limette is able to dynamically deallocate nodes of the tree to save memory. The setting *type*=`none` prevents answer generation.

proof	store the entire proof tree
query	only store root nodes related to the query
none	free all nodes as soon as possible

trace(*bool*) Turn the trace protocol `on` or `off`. The protocol emits a line for every derivation step performed. Tracing is the lowest-level of debugging a query.

spy(*predicate*,*arity*) Set a spypoint on *predicate* of *arity*. If the proof encounters that predicate, the box-model debugger is invoked (call, redo, exit, fail ports) and allows interaction (continuation, failure, skip).

spypoints Print a sorted list of all spypoints set by **spy**.

statistics Print a table of memory usage statistics (number of allocations and deallocations, total amount of memory currently allocated). Limette must be compiled in debug mode to support statistics.

unmemo(*module**) Turns off memoizing for all predicates currently defined inside the given module.

unmemo(*predicate*,*arity*) Turns off memoizing for the given predicate.

unspy(*predicate*,*arity*) Remove the spypoint on *predicate* of *arity* from the list of **spypoints** created by **spy**.

version Print a version identifier of the Limette program.

vfree(*variable**) Delete the given shell *variable* to allow reuse. Variables named like "`_Var`" (underscore followed by capital letter) are persistent across command lines. Once they are unified, their value can be referenced in subsequent lines.

vlist Print a sorted list of all shell variables and their current values.

where Print the name of the current module (set by **load** or **enter**).

3 Internal Predicates

! Control predicate “Cut”. Cuts off all alternative derivations of the goals of the same rule left to the cut as well as alternative rules for a predicate. In breadth search strategy the first rule that encounters it cuts off the other, partially executed, rules.

’@=’(_term_,_term_**)** Metapredicate “Equal term order”.

’@<’(_term_,_term_**)** Metapredicate “Smaller than” (term order).

’@=<’(_term_,_term_**)** Metapredicate “Smaller than or equal to” (term order).

’@>’(_term_,_term_**)** Metapredicate “Greater than” (term order).

’@>=’(_term_,_term_**)** Metapredicate “Greater than or equal to” (term order).

’=’(_term_,_term_**)** Unifies the given _term_s.

’=:=’(_term_,_term_**)** Metapredicate “Same value of arithmetic terms”.

’=\=’(_term_,_term_**)** Metapredicate “Different value of arithmetic terms”.

’<’(_term_,_term_**)** Metapredicate “Smaller value of arithmetic terms”.

’<>’(_term_,_term_**)** Constructive inequality of _terms_ $T_1 \neq T_2$. Equivalent to “$\neg(T_1 = T_2)$”.

’=<’(_term_,_term_**)** Metapredicate “Smaller or equal value of arithmetic terms”.

’>’(_term_,_term_**)** Metapredicate “Greater value of arithmetic terms”.

’>=’(_term_,_term_**)** Metapredicate “Greater or equal value of arithmetic terms”.

’:=’(_variable_,_term_**)** Assign _term_ as the value of _variable_ regardless whether _variable_ was already unified with some other value: “$\sigma\ \{v := T\}\ [v \mapsto T] \circ \sigma|_{\mathrm{dom}(\sigma)\setminus\{v\}}$”.

’=..’(_term_,_list_**)** Decompose _term_ $f(T_1,\ldots,T_n)$ into functor f and subterms T_i and unify _list_ with $[f, T_1, \ldots, T_n]$. If _term_ is variable, _list_ is converted into a respective _term_.

’==’(_term_,_term_**)** Metapredicate “Structure Equivalence”. Tests whether both _term_s are identical without unifying them.

’\==’(_term_,_term_**)** Like ’==’, but tests whether both _term_s are not the same.

abort Abort the entire derivation and return to the shell.

abort(_message_**)** Abort the entire derivation with a user-defined error _message_.

append(_file_**)** Set the current output stream to _file_. If _file_ does not exist it is created, otherwise subsequent output is appended to it.

argument(_list_**)** Unifies _list_ with the list of command line arguments (`argc`,`argv`) passed to Limette as a list of constants. Flags "`-flags`" are not considered here.

argument(_type_**,**_constant_**,**_value_**)** Unifies _value_ with the value of the command line flag named _constant_ of some requested _type_ (either **integer**,**float** or **symbol**). Flags are passed to Limette as "`-Cvalue`" where `C` is the _constant_ prepended to the _value_.

asserta(_rule_**)** Add the given _rule_ to the start of the current module. Syntactically invalid _rule_s yield an error message.

asserta(_rule_**,**_integer_**)** Like **asserta**(_rule_), but the rule index associated with the added rule is unified with _integer_.

assertz(_rule_**)** Add the given _rule_ to the end of the current module. Syntactically invalid _rule_s yield an error message.

assertz(_rule_**,**_integer_**)** Like **assertz**(_rule_), but the rule index associated with the added rule is unified with _integer_.

arg(_integer_**,**_compound_**,**_term_**)** Unify _term_ with the subterm indexed by _integer_ of the _compound_ term. If such a subterm does not exist, an error message is generated.

atom(_term_**)** Metapredicate. Tests whether _term_ is an atom (symbolic constant $c \in F$).

atomic(_term_**)** Metapredicate. Tests whether _term_ is atomic (constant, integer or floating-point number, internal object) without unifying it with atoms.

booting Succeeds if Limette is booting. The boot process includes the load of argument modules and contained queries. It stops as soon as the interactive shell is entered.

call(_goal_**)** The argument term _goal_ is interpreted as a goal and executed. Useful to construct goals at runtime. Syntactically invalid _goal_s yield an error message.

cd(_path_**)** Sets the current working directory to _path_. If _path_ is variable, the name of the current working directory is unified with it.

cm(_module_**)** Sets the current module to _module_. If _module_ is variable, the name of the current module is unified with it.

close(_gates_**)** Close the given _gates_, a formula consisting of reduction, universal quantification, conjunction and atoms. Invalid _gates_ yield an error message.

compl(_ifloat_**,**_ofloat_**)** Fuzzy modifier that complements truth: $\|\mathbf{compl}\|(\wp) = 1 - \wp$. If _ofloat_ is variable it is unified with `eps` (inverse operator semantics). If _ifloat_ is variable it is unifed with $1 -$ _ofloat_ (operator semantics).

compound(_term_**)** Metapredicate. Tests whether _term_ is a compound term $f(T_1, \ldots, T_n)$ without unifying it with compound terms.

cp(*float***)** Metapredicate "Current Possibility". Unfies *float* with the minimum required possibility μ of the initial state: "$\mu\ \{\mathbf{cp}(\mu)\}\ \eta$".

cs(*subst***)** Metapredicate "Current Substitution". Unifies *subst* with a term representation (list of equalities $v_i = T_i$ and inequalities $\langle v_{i1}, \ldots, v_{in_i}\rangle \neq \langle T_{i1}, \ldots, T_{in_i}\rangle$) of the substitution σ of the initial state: "$\sigma\ \{\mathbf{cs}(\sigma)\}\ \tau$".

ct(*time***)** Predicate "Current Time". Unifies *time* with a feasible time instant. Acts as if it were defined by the infinite set $\{\circ^t \mathbf{ct}(t) \mid t \in \mathcal{T}\}$ of facts.

ct(*low***,***high***)** Metapredicate "Current Timeset". Unifies $\mathcal{T}_S = [low, high]$ with the initial timeset interval $\mathcal{T}_S$ of the tableau element "$\mathcal{T}_S\ \{\mathbf{ct}(\mathcal{T}_S)\}\ \mathcal{T}_F$".

decode(*internal_term***,***term***)** Calls the decode method of *internal_term* to yield an equivalent conventional term which is unified with *term*.

depth(*term***,***integer***)** Unifies *integer* with the depth of *term*.

di(*low***,***high***)** Metapredicate "During Interval". Acts if it were defined by the infinite set "$\{\Box_{[c^-,c^+]}\ \mathbf{di}(c^-, c^+) \mid c^- \in \mathcal{T},\ c^+ \in \mathcal{T}\}$" of facts.

dp(*float***)** Metapredicate "Depicted Possibility". Acts if it were defined by the inifinite set "$\{\downarrow_\kappa\ \mathbf{dp}(\kappa) \mid 0 < \kappa \leq 1\}$" of facts. The argument *float* must be ground.

ds Metapredicate "Delete Substitution". Overwrites the substitution τ of the final state with the identity substitution: "$\sigma\ \{\mathbf{ds}\}\ \mathrm{id}$".

encode(*module***,***term***,***internal_term***)** Calls the encode method of *module* with a conventional *term* to yield an equivalent *internal_term*. **decode** for opposite direction.

fail Assertion that is always absolutely false.

float(*term***)** Metapredicate. Tests whether *term* is a floating-point number.

fp(*term***,***integer***)** Unifies *integer* with the fingerprint of *term*. Fingerprints $f(T)$ are bitset hashvalues to optimize unification: "$\mathbf{unifyable}(T_1, T_2) \Longrightarrow f(T_1) \cap f(T_2) \neq \emptyset$".

free(*term***)** Set all variables $v \in \mathrm{Vars}(term)$ to free variables regardless of their current value: "$\sigma\ \{\mathbf{free}(v)\}\ \sigma\ |_{\mathrm{dom}(\sigma)\setminus\{v\}}$".

functor(*constant***,***arity***,***term***)** Unifies *term* with the term "$constant(V_1, \ldots, V_{arity})$" (constant or compound term), where V_i are fresh variables. If *constant* is variable, *constant* and *arity* are unified with the functor of *term* and its arity.

gensym(*constant***)** Unifies *constant* with a unique constant never generated before.

infinity(*term***)** Metapredicate. Tests whether *term* represents plus or minus infinity (integer `-i`,`+i` or floating-point `-inf`,`+inf`).

integer(*term***)** Metapredicate. Tests whether *term* is a finite integer constant.

is(*term***,***ground_term***)** The *ground_term* is interpreted as an arithmetic expression and evaluated to a number which is unified with *term*. If *ground_term* contains undefined arithmetic operators or free variables an error occurs.

keyin(*integer***)** Unifies *integer* with the ASCII code of the next character read from the current input stream.

keyout(*integer***)** Writes a character with ASCII code *integer* to the current output stream.

like(*ifloat1***,***ifloat2***,***ofloat***)** Fuzzy modifier that compares truth values for gradual equality: $\|\mathbf{like}\|(\wp_1, \wp_2) = \varphi(\wp_1 - \wp_2)$. If *ofloat* is variable it is unified with `eps` (inverse operator semantics). If *ifloat* is variable it is unifed with the scaled ($\varphi(0) = 1$) Gaussian of the difference of truth values (operator semantics).

limpath(*constant***)** Sets the environment variable `LIMPATH` to be the given *constant*. If *constant* is variable it is unified with the current value of `LIMPATH`.

lookup(*atom***)** Proof the given atomic goal, but only if the results can be retrieved from some memo table directly without subordinate proofs, otherwise fail. Either no solution at all or all feasible solutions are generated.

memo(*module***)** Turn on memoizing for all predicates of the given module.

memo(*predicate/arity***)** Turn on memoizing for the given predicate.

name(*constant***,***string***)** Unifies *string* with the character string of the textual representation of *constant*. If *constant* is variable, *string* is converted into *constant*.

nan(*term***)** Metapredicate. Tests whether *term* is the special floating-point constant "Not-A-Number", which is the result of illegal expressions like "0/0".

nl Starts a new line. Writes a newline character to the output stream.

nonvar(*term***)** Metapredicate. Tests whether *term* is not a free variable.

norm(*term***,***derived_term***)** Normalizes *term* by applying a single unconditional term rewrite rule and unifies *derived_term* with the resulting term.

number(*term***)** Metapredicate. Tests whether *term* is a finite integer (excluding `-i`,`+i`) or a finite floating-point number (excluding `-inf`,`+inf`,`nan`).

number_string(*number***,***string***)** Converts *number* into its textual representation and unifies *string* with that text. If *number* is variable, *string* is converted into a *number* with that representation. Illegal number strings yield an error message.

offline Succeeds if Limette runs in batch mode or as a library.

online Succeeds if Limette runs in interactive mode (not **offline**).

open(*gates***)** Open the given *gates*, a formula consisting of reduction, universal quantification, conjunction and atoms. Invalid *gates* yield an error message.

parse(*constant***,***term***)** Parses the *constant* string as if it syntactically denotes a term and unifies it with the given *term*. If *constant* is variable, it is unified with the textual representation of *term*.

perm(*term***,***perm_term***)** Replaces all variables of *term* with fresh variables and unifies *perm_term* with that permutated term.

preds(*module***,***list***)** Unifies *list* with a list of all predicates in *module*. Each list item is a pair "predicate / arity".

quit Stop the entire Limette interpreter and return to the operating system.

read(*term***)** Reads a term (ending will a full stop) from the input stream and unifies it with the given *term*.

read(*message***,***term***)** Like **read**(*term*), but the given *message* is written to the output stream as a prompt first.

readp(*term***)** Reads a packed term from the input stream, converts it to a conventional term, and unifies it with the given *term*.

rememo(*module***)** Clear the memo tables of all predicates of the given module.

rememo(*predicate*/*arity***)** Clear the memo table of the given predicate.

repeat Iterator predicate defined by the rules "$\{$**repeat**, **repeat** $\leftarrow$ **repeat**$\}$". Always absolutely true for an infinite number of alternative derivations. The query "**repeat** $\wedge\, \mathcal{F}$" enumerates derivations of $\mathcal{F}$ until one succeeds.

retract(*integer***)** Removes a rule from the current module that has the rule index *integer* (returned by **asserta**,**assertz** and shown by the **list** command). If there is no such rule, this predicate is ignored.

see(*file***)** Opens *file* as the current input stream for subsequent **read** operations. The *file* named **input** refers to the keyboard.

seeing(*file***)** Unifies *file* with the name of the current input stream.

seen Closes the current input stream and sets the keyboard (file **input**) as the new current input stream.

send(*file***,***term***)** Send (low-level flushed write) *term* to a fifo-*file*.

sendl(*file***,***list***)** Like **sendq**, but all elements of *list* are written separated by blanks.

sendq(_file_,_term_**)** Like **send**, but appends a full stop and uses buffered writes.

shar(_shared_term_,_term_**)** Unifies _shared_term_ with _term_ where all common subterms of _term_ are shared. Operates logically equivalent to unification '='.

shell(_command_**)** Redirect _command_ to the command shell. It will be executed as if the _command_ was entered interactively. Load commands are not allowed.

si(_low_,_high_**)** Metapredicate "Set Interval". Overwrites the final state $\mathcal{T}_F$ with the given interval $[low, high]$: "$\mathcal{T}_S$ $\{\mathbf{si}(\mathcal{T}_F)\}$ $\mathcal{T}_F$".

size(_term_,_integer_**)** Unifies _integer_ to be the size (number of functors) of _term_.

sleep(_integer_**)** Delay for _integer_ many seconds.

some(_ifloat_,_ofloat_**)** Fuzzy modifier that intensifies truth: $\|\mathbf{some}\|(\wp) = \sqrt{\wp}$. If _ofloat_ is variable it is unified with the square of _ifloat_ (inverse operator semantics). If _ifloat_ is variable it is unifed with the square root of _ofloat_ (operator semantics).

sp(_float_**)** Metapredicate "Set Possibility". Overwrites the asserted possibility η of the final state with _float_: "μ $\{\mathbf{cp}(\eta)\}$ η".

ss(_subst_**)** Metapredicate "Set Substitution". Overwrites the substitution τ of the final state with _subst_. See **cs** for the format of _subst_. "σ $\{\mathbf{ss}(\tau)\}$ τ".

stamp(_term_,_integer_**)** Unifies _integer_ with the timestamp value of the functor of _term_. The timestamp value of logic operators contains the fuzzy semantics index.

stamp(_term_,_integer_,_stamped_term_**)** Unifies _stamped_term_ with _term_ where the toplevel functor is stamped with _integer_. Assigns fuzzy semantics to formulae.

system(_shell_command_,_integer_**)** Executes the _shell_command_ and unifies _integer_ with the error code returned by the operating system.

tab(_integer_**)** Print the given _integer_ number of blanks to the output stream.

tell(_file_**)** Set the current output stream to _file_. If _file_ does not exist it is created, otherwise it is overwritten. The _file_ named **output** refers to the screen.

telling(_file_**)** Unify _file_ with the name of the current output stream set by **tell**.

told Close the current output stream and set the screen (file **output**) as the new current output stream.

true Assertion that is always absolutely true.

unmemo(_module_**)** Turn off memoizing for all predicates of the given module.

unmemo(_predicate/arity_**)** Turn off memoizing for the given predicate.

var(*term***)** Metapredicate. Tests whether *term* is a free variable.

vars(*term***,***list***)** Unifies *list* with a list of all free variables occurring in *term*. Each variable is listed only once.

very(*ifloat***,***ofloat***)** Fuzzy modifier that reduces truth: $\|\mathbf{very}\|(\wp) = \wp^2$. If *ofloat* is variable it is unified with the square root of *ifloat* (inverse operator semantics). If *ifloat* is variable it is unifed with the square of *ofloat* (operator semantics).

window_size(*lines***,***columns***)** Reassigns the window size to the given values. The window size is used to stop text output whenever a page is filled. If the arguments are variable the current window size is unified with them.

write(*term***)** Print *term* without quotes (unless setting **print=quotes**) to the current output stream. The setting **print** determines the print method, the predicate **tell** the output stream. The default stream is the screen named **output**.

writeln(*term***)** Like **write**, but it appends a newline character.

writep(*term***)** Like **write**, but uses packed binary format. Common subterms are replicated, common symbols are shared.

writeps(*term***)** Like **writep**, but all common subterms are identified and represented only once. Such a packed term consumes less memory or disk space.

writeq(*term***)** Like **write**, but functors are quoted if required.