The Library Modules of Picat

Neng-Fa Zhou and Jonathan Fruhman

December 13, 2012

- X = Y
- X != Y
- X == Y
- X !== Y
- X := Y
- X > Y
- X >= Y
- X < Y
- X =< Y
- X <= Y
- $Term_1$ ++ $Term_2$ = $List$
- [X : I in D,...] = $List$
- L .. U = $List$
- L .. $Step$ .. U = $List$
- -X = Y
- +X = Y
- X + Y = Z
- X - Y = Z
- X * Y = Z
- X / Y = Z
- X // Y = Z
- X div Y = Z
- X /< Y = Z
- X /> Y = Z
- X ** Y = Z
- X mod Y = Z
- X rem Y = Z
- ~X = Y
- X \/ Y = Z
- X /\ Y = Z
- X ^ Y = Z
- X << Y = Z
- X >> Y = Z
- X >>> Y = Z
- Var[$Index_1$,...,$Index_n$]
- $Goal_1$,$Goal_2$
- $Goal_1$;$Goal_2$
- abort
- acyclic_term($Term$)
- append(X,Y,Z) (nondet)
- apply(S,Arg_1,...,Arg_n) = Val
- arity($Term$)
- array($Term$)
- atom($Term$)
- atom_chars(Atm) = $String$
- atom_codes(Atm) = $List$
- atomic($Term$)
- attr_var($Term$)
- avg($List$) = Val
- between($From$,To,X) (nondet)
- call(S,Arg_1,...,Arg_n)
- char($Term$)
- char_code($Char$) = Int
- code_char($Code$) = $Char$
- compare_terms($Term_1$,$Term_2$) = Res
- compound($Term$)
- copy_term($Term_1$) = $Term_2$
- delete($List$,X) = $ResList$
- delete_all($List$,X) = $ResList$
- different_terms($Term_1$,$Term_2$)
- digit($Char$)
- exit
- fail
- findall($Template$,S,Arg_1,...,Arg_n) = $List$
- float($Term$)
- flush
- freeze(X,$Goal$)
- get($MapOrAttrVar$,Key) = Val
- get_global_map() = Map
- get_heap_map() = Map
- ground($Term$)

- halt
- has_key(*MapOrAttrVar*,*Key*)
- hash_code(*Term*) = *Int*
- insert(*List*,*Index*,*Elm*) = *ResList*
- insert_all(*List*,*Index*,*AList*) = *ResList*
- integer(*Term*)
- keys(*MapOrAttrVar*) = *List*
- length(*Compound*) = *Len*
- list(*Term*)
- lowercase(*Char*)
- map(*Term*)
- map_to_list(*Map*) = *List*
- max(*List*) = *Val*
- max(*X*,*Y*) = *Val*
- membchk(*Term*,*List*)
- member(*Term*,*List*) (nondet)
- min(*List*) = *Val*
- min(*X*,*Y*) = *Val*
- name(*Struct*) = *Name*
- new_array(D_1,...,D_n) = *Arr*
- new_list(*N*) = *List*
- new_map(*PairsList*) = *Map*
- new_struct(*Name*,*IntOrList*) = *Struct*
- nonvar(*Term*)
- not *Call*
- number(*Term*)
- number_chars(*Num*) = *String*
- number_codes(*Num*) = *List*
- number_vars(*Term*,N_0) = N_1
- once *Call*
- parse_term(*String*,*Term*,*Vars*,*RString*)
- parse_term(*String*,*Term*,*Vars*)
- parse_term(*String*) = *Term*)
- post_event(*X*,*Event*)
- post_event_any(*X*,*Event*)
- post_event_bound(*X*)
- post_event_dom(*X*,*Event*)
- post_event_ins(*X*)
- print(*Term*)
- printf(*Term*,*Args*...)
- println(*Term*)
- put(*MapOrAttrVar*,*Key*,*Val*)
- read_char(*N*) = *String*
- read_char() = *Val*
- read_int() = *Int*
- read_line() = *String*
- read_real() = *Real*
- read_term() = *Term*
- read_token() = *String*
- read_unicode_char(*N*) = *String*
- read_unicode_char() = *Val*
- real(*Term*)
- remove_dups(*List*) = *ResList*
- repeat (nondet)
- reverse(*List*) = *ResList*
- select(*X*,*List*,*ResList*) (nondet)
- sort(*List*) = *SList*
- sort_down(*List*) = *SList*
- string(*Term*)
- struct(*Term*)
- sublist(*List*,*Start*,*End*) = *SubList*
- subsumes(*Term*$_1$,*Term*$_2$)
- sum(*List*) = *Val*
- throw *E*
- to_binary_string(*Int*) = *String*
- to_codes(*Term*) = *List*
- to_fstring(*Format*,*Term*) = *String*
- to_hex_string(*Int*) = *String*
- to_integer(*Num*) = *Int*
- to_list(*Struct*) = *List*
- to_lowercase(*String*) = *LString*
- to_oct_string(*Int*) = *String*
- to_real(*Num*) = *Real*
- to_string(*Term*) = *String*
- to_uppercase(*String*) = *UString*
- true
- unnumber_vars(*Term*$_1$) = *Term*$_2$
- uppercase(*Char*)

- values(*MapOrAttrVar*) = *List*
- var(*Term*)
- variant(*Term*$_1$,*Term*$_2$)
- vars(*Term*) = *Vars*
- write(*Term*)
- write_byte(*Bytes*)
- writef(*Term*,*Args*...)
- writeln(*Term*)
- zip(*List*$_1$,*List*$_2$,...,*List*$_n$) = *List*

Module math

- abs(*X*) = *Val*
- acos(*X*) = *Val*
- acosh(*X*) = *Val*
- asin(*X*) = *Val*
- asinh(*X*) = *Val*
- atan(*X*) = *Val*
- atan2(*X*,*Y*) = *Val*
- atanh(*X*) = *Val*
- cbrt(*X*) = *Val*
- ceiling(*X*) = *Val*
- cos(*X*) = *Val*
- cosh(*X*) = *Val*
- cot(*X*) = *Val*
- coth(*X*) = *Val*
- csc(*X*) = *Val*
- csch(*X*) = *Val*
- degrees(*Radian*) = *Degree*
- e = 2.71828
- exp(*X*) = *Val*
- floor(*X*) = *Val*
- inf
- log(*X*) = *Val*
- log(B,*X*) = *Val*
- log10(*X*) = *Val*
- log2(*X*) = *Val*
- modf(*X*) = (*FractVal*,*IntVal*)
- ninf
- nthrt(*N*,*X*) = *Val*
- pi = 3.14159
- power(*X*,*Y*) = *Val*
- radians(*Degree*) = *Radian*
- random = *Val*
- random(*Seed*) = *Val*
- randrange(*From*,*Step*,*To*) = *Val*
- randrange(*From*,*To*) = *Val*
- round(*X*) = *Val*
- sec(*X*) = *Val*
- sech(*X*) = *Val*
- sign(*X*) = *Val*
- sin(*X*) = *Val*
- sinh(*X*) = *Val*
- sqrt(*X*) = *Val*
- tan(*X*) = *Val*
- tanh(*X*) = *Val*
- truncate(*X*) = *Val*

Module io

- at_end_of_stream(*FD*)
- close(*FD*)
- dup(*FD*) = *NewFD*
- dup2(*FromFD*,*ToFD*)
- eof
- flush(*FD*)
- fprint(*FD*,*Term*)
- fprintf(*FD*,*Format*,*Args*...)
- fprintln(*FD*,*Term*)
- fread_byte(*FD*) = *Val*
- fread_byte(*FD*,*N*) = *List*
- fread_char(*FD*) = *Val*
- fread_char(*FD*,*N*) = *String*
- fread_file_bytes(*FD*) = *List*
- fread_file_chars(*FD*) = *String*
- fread_int(*FD*) = *Int*
- fread_line(*FD*) = *String*
- fread_real(*FD*) = *Real*
- fread_term(*FD*) = *Term*
- fread_token(*FD*) = *String*
- fread_unicode_char(*FD*) = *Val*
- fread_unicode_char(*FD*,*N*) = *String*

- `freadln(`FD`) = `$String$
- `fwrite(`FD`,`$Term$`)`
- `fwrite_byte(`$Bytes$`)`
- `fwritef(`FD`,`$Format$`,`$Args\ldots$`)`
- `fwriteln(`FD`,`$Term$`)`
- `getpos(`FD`) = `Pos
- `mkfifo(`$Path$`)`
- `mkfifo(`$Path$`,`$Mode$`)`
- `mkpipe() = `FD_Map
- `mktmp() = `FD
- `open(`$Name$`) = `FD
- `open(`$Name$`,`$Mode$`) = `FD
- `peek_byte(`FD`) = `Val
- `peek_char(`FD`) = `Val
- `peek_int(`FD`) = `Int
- `peek_real(`FD`) = `$Real$
- `peek_unicode_char(`FD`) = `Val
- `rewind(`FD`)`
- `seek(`FD`,`$Offset$`,`$From$`)`
- `setpos(`FD`,`Pos`)`
- `sizeof_char() = `$Size$
- `stderr`
- `stdin`
- `stdout`

Module `os`

- `atime(`$Path$`) = `$Time$
- `block_special(`$Path$`)`
- `cd(`$Path$`)`
- `char_special(`$Path$`)`
- `chdir(`$Path$`)`
- `chmod(`$Path$`,`$Mode$`)`
- `cp(`$Path1$`,`$Path2$`)`
- `create(`$Path$`)`
- `create(`$Path$`,`$Mode$`)`
- `ctime(`$Path$`) = `$Time$
- `cwd() = `$Path$
- `dev_id(`$Path$`) = `Int
- `directory(`$Path$`)`
- `directory_exists(`$Path$`)`
- `executable(`$Path$`)`
- `exists(`$Path$`)`
- `fifo(`$Path$`)`
- `file(`$Path$`)`
- `file_base_name(`$Path$`) = `$String$
- `file_directory_name(`$Path$`) = `$String$
- `file_exists(`$Path$`)`
- `file_type(`$Path$`) = `$Term$
- `gid(`$Path$`) = `Int
- `ino(`$Path$`) = `Int
- `link(`$Path$`)`
- `link(`$Path1$`,`$Path2$`)`
- `listdir(`$Path$`) = `$List$
- `listdir(`$Path$`,`$REPattern$`) = `$List$
- `message_queue(`$Path$`)`
- `mkdir(`$Path$`)`
- `mkdir(`$Path$`,`$Mode$`)`
- `mkdirs(`$Path$`)`
- `mkdirs(`$Path$`,`$Mode$`)`
- `mode(`$Path$`) = `$String$
- `mtime(`$Path$`) = `$Time$
- `mv(`$Path1$`,`$Path2$`)`
- `nlink(`$Path$`) = `Int
- `pwd() = `$Path$
- `readable(`$Path$`)`
- `rm(`$Path$`)`
- `rmdir(`$Path$`)`
- `root() = `$Path$
- `semaphore(`$Path$`)`
- `separator() = `Val
- `shared_memory(`$Path$`)`
- `shortcut(`$Path$`)`
- `shortcut(`$Path1$`,`$Path2$`)`
- `size(`$Path$`) = `Int
- `socket(`$Path$`)`
- `uid(`$Path$`) = `Int
- `unlink(`$Path$`)`
- `writable(`$Path$`)`

Modules `cp`**,** `sat`**, and** `mip`

- X `#=` Y
- X `#!=` Y
- X `#>` Y
- X `#>=` Y
- X `#<` Y
- X `#=<` Y
- X `#<=` Y
- `#~`X
- X `#\/` Y
- X `#/\` Y
- X `#^` Y
- X `#=>` Y
- X `#<=>` Y
- $Vars$ `in` Exp
- $Vars$ `notin` Exp
- `all_different(`$FDVars$`)`
- `all_distinct(`$FDVars$`)`
- `assignment(`$FDVars1$`,`$FDVars2$`)`
- `circuit(`$FDVars$`)`
- `count(`V`,`$FDVars$`,`Rel`,`N`)`
- `cumulative(`$Starts$`,`$Durations$`,`$Resources$`,`$Limit$`)`
- `diffn(`$RectangleList$`)`
- `disjunctive_tasks(`$Tasks$`)`
- `element(`I`,`$List$`,`V`)`
- `fd_degree(`$FDVar$`) =` $Degree$
- `fd_disjoint(`$DVar1$`,`$DVar2$`)`
- `fd_dom(`$FDVar$`) =` $List$
- `fd_false(`$FDVar$`,`Elm`)`
- `fd_max(`$FDVar$`) =` Max
- `fd_min(`$FDVar$`) =` Min
- `fd_min_max(`$FDVar$`,`Min`,`Max`)`
- `fd_next(`$FDVar$`,`Elm`) =` $NextElm$
- `fd_prev(`$FDVar$`,`Elm`) =` $PrevElm$
- `fd_set_false(`$FDVar$`,`Elm`)`
- `fd_size(`$FDVar$`) =` $Size$
- `fd_superset(`$DVar1$`,`$DVar2$`)`
- `fd_true(`$FDVar$`,`Elm`)`
- `fd_var(`$Term$`)`
- `global_cardinality(`$List$`,`$Pairs$`)`
- `indomain(`Var`)`
- `indomain_down(`Var`)`
- `lp_in(`$Vars$`,`$LExp$`,`$UExp$`)`
- `neqs(`$NeqList$`)`
- `new_fd_var() =` $FDVar$
- `serialized(`$Starts$`,`$Durations$`)`
- `solve(`$Options$`,`$Vars$`)`
- `solve(`$Vars$`)`
- `subcircuit(`$FDVars$`)`

Module `thread`

- `acquire_mutex(`$Mutex$`)`
- `broadcast_cv(`CV`)`
- `join(`$Thread$`)`
- `new_cv() =` CV
- `new_mutex() =` $Mutex$
- `new_rwlock() =` $RWLock$
- `new_semaphore() =` $Semaphore$
- `new_semaphore(`N`) =` $Semaphore$
- `new_thread(`S`,`Arg_1`,...,`Arg_n`) =` $Thread$
- `p_semaphore(`$Semaphore$`)`
- `rdlock(`$RWLock$`)`
- `release_mutex(`$Mutex$`)`
- `rwunlock(`$RWLock$`)`
- `signal_cv(`CV`)`
- `sleep(`$Milliseconds$`)`
- `start(`$Thread$`)`
- `this_thread() =` $Thread$
- `v_semaphore(`$Semaphore$`)`
- `wait_cv(`CV`,`$Mutex$`)`
- `wrlock(`$RWLock$`)`

Module `timer`

- `get_interval(`$Timer$`) =` $Milliseconds$
- `kill(`$Timer$`)`
- `new_timer(`$Milliseconds$`) =` $Timer$
- `set_interval(`$Timer$`,`$Milliseconds$`)`
- `start(`$Timer$`)`
- `stop(`$Timer$`)`

Module socket

- accept(*FD*) = *Client*
- bind(*FD*,*INet*,*Address*,*Port*)
- bind(*FD*,*Unix*,*Name*)
- close(*FD*)
- connect(*FD*,*INet*,*Address*,*Port*)
- connect(*FD*,*Unix*,*Name*)
- getaddr(*Name*) = *Addr*
- getcanonicalname(*Addr*) = *Name*
- gethostbyaddr(*Addr*) = *Host*
- gethostbyname(*Name*) = *Host*
- getservbyname(*Name*) = *Service*
- getservbyname(*Name*,*Type*) = *Service*
- getservport(*Name*) = *Port*
- getsockopt(*FD*,*Level*,*Option*) = *Value*
- joingroup(*GroupAddress*)
- leavegroup(*GroupAddress*)
- listen(*FD*)
- listen(*FD*,*Backlog*)
- recv(*FD*) = *Message*
- recv(*FD*,*Flags*) = *Message*
- recvfrom(*FD*,*Domain*) = *Message*
- recvfrom(*FD*,*Flags*,*Domain*) = *Message*
- send(*FD*,*Message*) = *NBytes*
- send(*FD*,*Message*,*Flags*) = *NBytes*
- sendto(*FD*,*Message*,*Domain*,*Address*,*Port*) = *NBytes*
- sendto(*FD*,*Message*,*Flags*, *Domain*,*Address*,*Port*) = *NBytes*
- sendto(*FD*,*Message*,*Flags*,*Name*) = *NBytes*
- sendto(*FD*,*Message*,*Name*) = *NBytes*
- setsockopt(*FD*,*Level*,*Option*,*Value*)
- socket(*Domain*,*Type*) = *FD*
- tcp_bind(*FD*,*Address*,*Port*)
- tcp_connect(*FD*,*Address*,*Port*)
- tcp_socket() = *FD*
- udp_socket() = *FD*
- udp_bind(*FD*,*Address*,*Port*)
- unix_bind(*FD*,*Name*)
- unix_connect(*FD*,*Name*)
- unix_socket() = *FD*

Module sys **(imported by default)**

- cl(*File*)
- compile(*File*)
- debug
- execute(*CommandString*) = *Status*
- getenv(*EnvironmentVarNameString*) = *String*
- modules() = *List*
- nodebug
- nospy *Functor*
- nospy
- notrace
- profile(*Goal*)
- profile_src(*File*)
- spy *Functor*
- statistics(*Name*,*Value*) (nondet)
- statistics
- table_get_all(*Goal*) = *List*
- table_get_one(*Goal*)
- trace

Module `datetime`

- `add_days(`$DateTime$`,`$Days$`)` = $DateTime$
- `add_hours(`$DateTime$`,`$Hours$`)` = $DateTime$
- `add_milliseconds(`$DateTime$`,`$MilliSeconds$`)` = $DateTime$
- `add_minutes(`$DateTime$`,`$Minutes$`)` = $DateTime$
- `add_months(`$DateTime$`,`$Months$`)` = $DateTime$
- `add_seconds(`$DateTime$`,`$Seconds$`)` = $DateTime$
- `add_years(`$DateTime$`,`$Years$`)` = $DateTime$
- `compare(`$DateTime$`,`$DateTime$`)` = Res
- `current_datetime()` = $DateTime$
- `day(`$DateTime$`)` = Day
- `day_of_week(`$DateTime$`)` = $Atom$
- `day_of_year(`$DateTime$`)` = Int
- `day_string(`$DateTime$`)` = $String$
- `dt_to_fstring(`$Format$`,`$DateTime$`)` = $String$
- `hour(`$DateTime$`)` = $Hour$
- `is_leap_year(`$DateTime$`)`
- `millisecond(`$DateTime$`)` = $MilliSecond$
- `minute(`$DateTime$`)` = $Minute$
- `month(`$DateTime$`)` = $Month$
- `month_string(`$DateTime$`)` = $String$
- `second(`$DateTime$`)` = $Second$
- `set_day(`$DateTime$`,`Day`)`
- `set_hour(`$DateTime$`,`$Hour$`)`
- `set_millisecond(`$DateTime$`,`$MilliSecond$`)`
- `set_minute(`$DateTime$`,`$Minute$`)`
- `set_month(`$DateTime$`,`$Month$`)`
- `set_second(`$DateTime$`,`$Second$`)`
- `set_year(`$DateTime$`,`$Year$`)`
- `time_string(`$DateTime$`)` = $String$
- `year(`$DateTime$`)` = $Year$