Dr. Jozo J. Dujmović

PROLOG REFERENCE CARD

Program manipulation

consult(filename).	Load program filename.pl (or filename.pro)
[filename].	Same as consult (short form of consult)
[f1, f2, …].	Consult f1, then consult f2, etc.
make.	Reload all modified files (files edited after consulting)
listing.	Listing of all predicates
listing(p).	Listing of the specific predicate p
halt.	Terminate the program and exit

Constants

atom	Symbol
123	Integer
12.3	Real number
‘string’	Sring

Relations and equalities

X = Y	X is equal to Y if X and Y match (for arbitrary X,Y)
X \= Y	Succeeds if X and Y do not match
X = = Y	X is literally equal to Y (if X and Y are identical)
X \= = Y	Succeeds if X is not literally equal to Y
Expr1 =:= Expr2	Succeeds Expr1 and Expr2 evaluate to the same value
Expr1 =\= Expr2	Succeeds Expr1 and Expr2 evaluate to different values
Expr1 > Expr2	Succeeds Expr1 > Expr2 (expressions are evaluated)
Expr1 < Expr2	Succeeds Expr1 < Expr2 (expressions are evaluated)
Expr1 >= Expr2	Succeeds Expr1 >= Expr2 (expressions are evaluated)
Expr1 =< Expr2	Succeeds Expr1 <= Expr2 (expressions are evaluated)
X is Expr	X matches the value of expression Expr (assignment)

Facts

predicate(a,b,c).	A relationship between a, b, and c

Rules

:-	the rule definition symbol (“if”)
len[[_|T], N) :- len(T,K), N is 1+K.	Rule for the length of list
order(A, B) :- A=<B.	A and B are in order if A is less than or equal to B

List operations

[ ]	Empty list
[a, b, c]	List with three elements
[H | T]	Separation of head (H) and tail (T)
[H | _]	List with head H and an anonymous tail
[X, Y, Z | T]	First three elements X, Y, Z, and tail T
is_list(L).	Succeeds if L is a list
length(L, Len).	The length of list L is Len
member(E, L) .	E is an element of list L
append([1,2], [3,4], L).	L = [1,2,3,4] ; flexible append of two lists
delete([1, e, 2, e, 3], e, L).	L = [1, 2, 3]
select(e, [1, e, 2, e, 3], L).	L = [1, 2, e, 3] remove one instance of element from list
select(e, L, [1, 2, 3]).	L = [e, 1, 2, 3] insert one instance of element in list
nth0(Ind, L, E).	Addressing vector L[0],L[1],…: creates E=L[Ind]
nth1(Ind, L, E).	Addressing vector L[1],L[2],…: creates E=L[Ind]
last(L, E).	Unify E with the last element of list L
permutation(L1, L2).	L2 is a permutation of L1 and vice versa
flatten([1,[2,[3,4],5],6], L).	L = [1, 2, 3, 4, 5, 6] ; makes L from sublist elements
sumlist (List, Sum).	Sum is the sum of all elements in the List
numlist(MinInt, MaxInt, List).	Make integer list List = [MinInt, MinInt+1, … , Maxint]
reverse(List, RevList).	Reverse List and create the reversed list RevList
sort([7,3,5,3], X).	X = [3,5,7] non flexible sort with elimination of duplicates
msort([7,3,5,3], X).	X = [3,3,5,7] non flexible sort of all elements
merge(L1, L2, L12).	Merge sorted L1 and L2 yielding sorted L12 (all duplicates are included)

Set operations

is_set(S).	Succeeds if S is a set (a list without duplicates)
list_to_set(L, S).	Makes set-list S by eliminating duplicates from L
union(Set1, Set2, U).	U = Set1 ∪ Set2
intersection(Set1, Set2, I).	I = Set1 ∩ Set2
subtract(Set, Del, Res).	Res = Set \ Del (remove all elements of set Del from Set)
subset(Subset, Set).	Succeeds if Subset ⊆ Set
merge_set(S1, S2, S12).	Merge sorted sets S1 and S2 yielding sorted S12 = S1∪S2 (all duplicates in S12 are eliminated)

Control and search

findall(X, prog(a,b,X), Xlist).	Make Xlist that includes all X solutions from prog(a,b,X)
forall(member(E,List), E>0).	Check that all members of List are positive
a, b, !, c, d.	Cut: prevent backtracking from c to b (and a)
a, b, c, ! .	Single satisfaction of goals a, b, and c
fail	Always fail (and cause backtracking)
repeat	Always succeed (backtracking restart point)

Input/Output

read(X).	Read X from keyboard
write(X).	Display X on the screen
tab(N).	Display N spaces
nl	New line

Library functions

//	Integer division
mod	Modulus
rem	Remainder of division (float_fractional_part)
abs, sign, min, max	Min and max of two variables only
random(N)	Random integer 0 <= random <N
round	Nearest integer
truncate(X)	Integer part of X
float_integer_part(X)	Same as truncate
float_fractional_part(X)	Fractional part of X
floor(X), ceiling(X)	Nearest integers <X and >X (X is an expression)
>> , << , \/ , /\ , xor , \	Bitwise shr, shl, or, and, xor, not
sqrt, **, ^, exp, log, log10	Base**expo same as Base^expo
sin, cos, tan, asin, acos, atan	Trigonometric function
pi, e	Constants 3.141593 and 2.718282

Recognizers

var(X), nonvar(X)	Succeeds if X is (is not) a variable
integer, float, number, string	Recognition of basic data types
atom, atomic(X)	X is atom, string, integer or float

Database operations

assert(car(ford)).	Add car(ford) as the last fact of the car predicate
assertz(car(ford)).	Same as assert
asserta(car(ford)).	Add car(ford) as the first fact of the car predicate
retract(car(ford)).	Remove the fact car(ford)
retractall(car(_)).	Remove all car facts from the database

Note: International Organization for Standardization (ISO, http://www.iso.org) has Prolog standard: **ISO/IEC 13211-1** that was published in 1995. Prolog implementations try to support the standard. This is sometimes only partially achieved. For a more complete description of Prolog built-in predicates see http://pauillac.inria.fr/~deransar/prolog/docs.html.