Contents

List of Figures

Chapter 1

To begin

Abstract

TEX4LOG is a simple set of macros written in LaTEX allowing for a simple documentation scheme for Prolog.

1.1 Preamble: Naive Bayes Classifiers

A *Naive Bayes classifier* extracts statistics from a table of data, and uses those stats to generate probabilities that new examples fall into different class.

For example, suppose we have a relation in Figure 1.1 showing some mapping between attributes and a special class attribute called *type*. One thing we might try to do with this information is to guess the type of some new example based on this prior information of the examples seen to date. For example, what is the *type* of the following new example:

$$IF\ make = ford\ \wedge\ size = medium\ THEN\ type = ?$$

To accomplish this task, the frequency F with which attribute values appear within certain classes is first computed in Figure 1.2. Next, these frequencies of some range r in class C_i from attribute A_j is turned into the *class frequency ratios* $R(A_i = v|C_i)$. Some product of these ratios will become the probability that an new example falls into a class.

Since any zero entry in a product makes the whole product zero, the F entries labelled $\{abcd\}$ in Figure 1.2 are a problem. We solve this problem using the standard kludge: replace zero entries with a very small number by initializing all counts 1 instead of 0. Assuming this kludge, then, the frequency counts from Figure 1.2 are shown in Figure 1.3.

Each of the entries in Figure 1.3 is a measurement conditional on some class. For example, the $\frac{2}{9}$ for `suv`'s $Make = Ford$ is denoted $R(Make = Ford|suv) = \frac{2}{9}$. Now the likelihood of our example falling into each class is the product of these conditional frequency for that class. In the Naive Bayesian framework, unknown values $F(Make = ?|Class_i)$ are just ignored. So, the likelihood L of our example being an SUV

$$\begin{aligned} likelihood(SUV) &= \frac{2}{5} * \frac{2}{5} * \frac{5}{13} = 0.0615385 \\ likelihood(coupe) &= \frac{2}{9} * \frac{2}{9} * \frac{9}{13} = 0.011396 \end{aligned}$$

Each liklihood is converted into a probability by normalizing them with respect to the sum of all the likelihoods; i.e.

$$\begin{aligned} Prob(SUV) &= \frac{0.0615385}{(0.0615385 + 0.011396)} = 84\% \\ Prob(coup) &= \frac{0.011396}{(0.0615385 + 0.011396)} = 16\% \end{aligned}$$

That is, if $Make = ford$ and $Size = medium$, then it is most likely that we are looking at an SUV.

$Rel =$

attributes			class
make	size	hifi	type
Mitsubishi	small	yes	coup
Mitsubishi	medium	no	suv
Toyota	small	yes	coup
Toyota	large	no	coup
Toyota	large	no	suv
Benz	small	yes	coup
Benz	large	no	suv
BMW	small	yes	coup
BMW	medium	yes	coup
Ford	small	yes	coup
Ford	large	no	suv
Honda	small	no	coup

Figure 1.1: A log of car types.

1.2 Introductions

The following package is a SWI-Prolog system [6]. Prolog is a useful language for rapdily building systems [1–5]. This TEX4LOG description follows my standard(ish) pattern:

- Shell
- Knowledge base
- An appendix with acknowledgements, references, and licensing details.

The shell divides up as follows:

- Initializations:
  - Operator definitions (must be first).
  - Flags (these can usually go just before the start-up actions but, for safety's sake, we place them at the front).
  - hooks (into the Prolog reader)
  - hacks (shameful things we'd rather hide).
- The actual system code.
- Library code which, ideally, should be good for more than just this application).
- Start-up code (must be loaded into Prolog last).

$F =$

	attributes					class
MAKE	coup : suv	SIZE	coup : suv	HIFI	coup : suv	coup : suv
Mitsubishi	1 : 1	Small	$6 : 0^a$	yes	$6 : 0^b$	8 : 4
Toyota	2 : 1	Medium	1 : 1	no	2 : 4	
Benz	1 : 1	Large	1 : 3			
BMW	$2 : 0^c$					
Ford	1 : 1					
Honda	$1 : 0^d$					

Figure 1.2: Frequency counts from Figure 1.1.

$R =$

	attributes					class
MAKE	coup : suv	SIZE	coup : suv	HIFI	coup : suv	coup : suv
Mitsubishi	$\frac{2}{9} : \frac{2}{5}$	Small	$\frac{7}{9} : \frac{1}{5}^\dagger$	yes	$\frac{7}{9} : \frac{1}{5}^\dagger$	$\frac{9}{13} : \frac{5}{13}$
Toyota	$\frac{2}{9} : \frac{2}{5}$	Medium	$\frac{7}{9} : \frac{2}{5}$	no	$\frac{7}{9} : \frac{5}{5}$	
Benz	$\frac{2}{9} : \frac{1}{4}$	Large	$\frac{2}{9} : \frac{4}{5}$			
BMW	$\frac{3}{9} : \frac{1}{5}^\dagger$					
Ford	$\frac{2}{9} : \frac{2}{5}$					
Honda	$\frac{2}{9} : \frac{1}{5}^\dagger$					

Figure 1.3: Class frequency ratios counts from Figure 1.2 (all counts initialized to one).

1.3 Initializations

1.3.1 Operators

```
:- op(1001,xfx, the).
:- op(1001,xfx, a  ).
:- op(999 ,fx,  *  ).
:- op(700,xfx, :=).
:- op(1199 ,xfx, of ).
:- op(700,xfx, <- ).
:- op(1 ,fx, (?) ).
:- op(1 ,fx, (!) ).
```

1.3.2 Flags

```
:- multifile     option/2, meta/9, oo/5, get/3,zap/3,commit/4.
:- dynamic       option/2, meta/9, oo/5, context/1.
:- discontiguous option/2, meta/9, oo/5, get/3,zap/3,commit/4.
```

1.3.3 Hooks

```
term_expansion((W of X --> Y),Z) :- defmethod((W of X --> Y ),Z).
term_expansion((W of X),      Z) :- defmethod((W of X --> []),Z).
term_expansion((X the Rel),Z) :- dd(Rel,X,Z).
term_expansion((X a   Rel),Z) :- eg(Rel,X,Z).
term_expansion(A=B,       []) :- set(A=B).

goal_expansion(*(X,Y,Z),W) :-methodCall(*(X,Y,Z),W).
```

1.3.4 Hacks

Shown here are some dark secrets of the Prolog wizards. If you don't yet understand the following code, then you don't need to know it. Trust us, we are knowledge engineers.

```
goal_expansion(is(A,B,C,C),            is(A,B)).
goal_expansion(=(A,B,C,C),              =(A,B)).
goal_expansion(>=(A,B,C,C),            >=(A,B)).
goal_expansion(>(A,B,C,C),              >(A,B)).
goal_expansion(<(A,B,C,C),              <(A,B)).
goal_expansion(<=(A,B,C,C),            <=(A,B)).
```

1.4 Library code

1.4.1 Does a goal have only 1 matching clause?

```
clause1(X,Y) :- singleton(X), clause(X,Y).

singleton(X) :-
    Sym='$singleton_',
    flag(Sym,_,0),
    \+ singleton1(Sym,X),
    flag(Sym,1,1).

singleton1(Sym,X) :- clause(X,_),flag(Sym,N,N+1),N > 1,!.
```

1.4.2 Does a goal have only 1 way to succeed?

```
only(X) :-
    Sym='$only_',
    flag(Sym,_,0),
    \+ only1(Sym,X),
    flag(Sym,1,1).

only1(Sym,X) :- X, flag(Sym,N,N+1),N > 1,!.

solo(X) :- only(X), X.
```

1.4.3 Configuration Control

```
[] := []     :- !.
[H0|T0] := [H|T] :- !, H0 := H, T0 := T.
X := Y       :-  option(X,Z),!, Y=Z.
X := _       :-  !, barph(missingOption(X)).
?X           :- atomic(X), X := 1.

set(X=Y) :- retractall(option(X,_)), assert(option(X,Y)).

commandLine :-
    current_prolog_flag(argv, Argv),
    append(_, [--|Args], Argv), !,
    concat_atom(Args, ' ', SingleArg),
    term_to_atom(Term, SingleArg),
    c2l(Term,List),
    forall(member(One,List), set(One)).
commandLine.
```

1.4.4 Demo support code

Catches the output from some predicate X and saves it a file X.spy. The command:

```
\SRC{X.spy}{Caption}
```

includes the generated file into the LaTeX document.

The code demos/1 deletes any old output and runs some goal twice: once to trap it to a file and once to show the results on the screen.

```
demos(G) :-
    sformat(Out,'~w.spy',G),
    (exists_file(Out) -> delete_file(Out) ; true),
    tell(Out),
    format('% output from '':- demos(~w).''\n\n',G),
        T1 is cputime,
    ignore(forall(G,true)),
    T2 is (cputime - T1),
    format('\n% runtime = ~w sec(s)\n',[T2]),
        told,
    format('% output from '':- demos(~w).''\n',G),
        ignore(forall(G,true)),
    format('\n% runtime = ~w sec(s)',[T2]).
```

1.4.5 Ordered counted key value pairs

```
inc([],Key,[Key=1]).
inc([Key0=Value0|T],Key,Add) :-
        compare(Order,Key0,Key),
        inc(Order,Key0=Value0,T, Key,Add).

inc(<,H,     T,    X,[H       |Add]) :- inc(T,X,Add).
inc(=,Key=X, T,  Key,[Key=Y   |T])   :- Y is X + 1.
inc(>,H,     T,  Key,[Key=1,H |T]).

n([Key0=V|T],Key,X) :- compare(Ord,Key0,Key), n(Ord,V,T,Key,X).

n(=,V,_,_  ,V).
n(<,    _,T,Key,X) :- n(T,Key,X).
```

Tagged pairs

```
less1([H|T],H,T).
less1([_|T],Out,Rest) :- less1(T,Out,Rest).

key(L0,K,V0,V,[K=V|L]) :- less1(L0,K=V0,L).
```

1.4.6 Dump a Whole File to the Screen

```
chars(F) :- see(F), get_byte(X), ignore(chars1(X)), seen.

chars1(-1) :- !.
chars1(X)  :- put(X), get_byte(Y), chars1(Y).
```

1.4.7 License

```
hello :-  % ◀
    [program,version,copyright,motto,copywho]:=[N,V,Y,M,C],
    format('~s version ~s\n Copyright (C) ~s by ~s\n',
          [N,V,Y,C]),
    format(' "~s"\n\n~s ~s ',[M,N,V]),
    chars('nowarranty.txt'). % see §.1.1

warranty :-
    [program,copyright,copywho]:=[P, Y,C],
    format('~s by ~s\n Copyright (C) ~s\n\n',[P,C,Y]),
    chars('warranty.txt'),nl. % see §.1.2

conditions :-
    chars('conditions.txt'),nl. % see §.1.3
```

1.4.8 Miscalleanous

Pretty print a list of terms.

```
portrays(L) :- portrays(L,_,_).

portrays([],_,_).
portrays([H|T],F0,A0) :-
    functor(H,F,A),
    (F0=F,A0=A
        -> portray_clause(H),
           portrays(T,F0,A0)
         ; nl,portray_clause(H),
           portrays(T,F,A)).
```

Other stuff.

```
times(N,G,Out) :-
    T1 is cputime, repeats(N,true),
    T2 is cputime, repeats(N,G),
    T3 is cputime, Out is (T3-T2-(T2-T1))/N.

repeats(N0,G) :-
    N is N0,
    forall(between(1,N,_),G).

c2l((X,Y),[X|Z]) :- !,c2l(Y,Z).
c2l(X,[X]).

l2c([W,X|Y],(W,Z)) :- l2c([X|Y],Z).
l2c([X],X).

mostC2l((X,Y),[X|Z]) :- !,mostC2l(Y,Z).
mostC2l(_,[]).

sneak(X) :- load_files(X,[silent(true),if(changed)]).

spit(N1,N2,X) :- (0 is N1 mod N2 -> spit(X) ; true).

spit(X) :- ?verbose,!,write(user,X),flush_output(user).
spit(_).

barph(X) :- format('%W> ~p\n',X),fail.

barphln(X) :-
    here(File,Line),
    format('%W> ~p@~p : ~p\n',[File,Line,X]),
        fail.

here(File,Line) :-
        source_location(Path,Line),
        file_base_name(Path,File).
```

1.4.9 Defining methods

```
defmethod((W of X --> Y),[Z|Meta]) :-
    getContext(X,C),
    expandInContext(C,(W-->Y),Z0),
        tidy(Z0,Z),
    metaMethod(Z,C,Meta).

getContext(X,B) :- o([rel0_ = X,blank_ = B]),!.
getContext(X,_) :- barphln(X is unknown).

expandInContext(C,(W-->Y),Z) :-
    retractall(context(_)),
    assert(context(C)),
    expand_term((W --> Y),Z),
    retractall(context(_)).

metaMethod((X :- _),C,Ind) :- !, metaMethod1(X,C,Ind).
metaMethod(X,C,Ind) :- metaMethod1(X,C,Ind).

metaMethod1(X,C,[(:- index(Index)),(:- discontiguous F/A)]) :-
    functor(X,F,A),
    (A=2
        -> Index=.. [F,1,0],
           arg(1,X,C)
        ;  functor(Term,F,A),
       A1 is A - 1,
       arg(A1,X,C),
       Term =.. [F|L],
       append([1|Rest],[1,0],L),
       zeros(Rest),
       Index =.. [F|L]).

zeros([]).
zeros([0|T]) :- zeros(T).
```

Remove stray trues.

```
tidy(A,C) :-
    tidy1(A,B),
    (B = (Head :- true) -> C=Head ;  C=B).

tidy1(A,C) :- once(tidy2(A,C)).

tidy2(A,            A) :- var(A).
tidy2((A,B),    (A,TB)) :- var(A), tidy1(B,TB).
tidy2((A,B),    (TA,B)) :- var(B), tidy1(A,TA).
tidy2(((A,B),C),     R) :- tidy1((A,B,C), R).
tidy2((true,A),      R) :- tidy1(A,R).
tidy2((A,true),      R) :- tidy1(A,R).
tidy2((A;B),   (TA;TB)) :- tidy1(A,TA), tidy1(B,TB).
tidy2((A->B), (TA->TB)) :- tidy1(A,TA), tidy1(B,TB).
tidy2(not(A),  not(TA)) :- tidy1(A,TA).
tidy2((A :- B), R) :-
    tidy1(B,TB),(TB=true-> R=A; R=(A:-TB)).
tidy2((A,B),  R) :-
    tidy1(A,TA), tidy1(B,TB),(TB=true -> R=TA; R=(TA,TB)).
tidy2(A,A).
```

1.4.10 Wrapper

Cool stuff

```
methodCall(*(X,Y,Z),W) :-
    context(Y),
    wrapper(X,Y,Z,W).
```

```
wrapper(X,A,B,Out) :-
    wrap(X,Before,[],After,[],Goal),
    append(Before,[Goal|After],Temp),
        adds2vars(Temp,A,B,Out).

adds2vars([X0],A,B,X) :- add2vars(X0,A,B,X).
adds2vars([X0,Y|Z],A,B,(X,Rest)) :-
    add2vars(X0,A,C,X),
    adds2vars([Y|Z],C,B,Rest).

add2vars(oo(X,Y,Z),A,B,oo(A,X,Y,Z,B)) :- !.
add2vars(X,A,A,X).
```

```
wrap(X,B0,B,A0,A,Y) :-
    once(wrap0(X,Z)),
    wrap1(Z,B0,B,A0,A,Y).

wrap0(X,        leaf(X) ) :- var(X).
wrap0(X,        leaf(X) ) :- atomic(X).
wrap0([],    leaf(true) ).
wrap0([H|T],      [H|T] ).
wrap0(?X,            ?X  ).
wrap0(!X,            !X  ).
wrap0(X,        term(X) ).

wrap1(leaf(X),     B, B, A, A, X).
wrap1([H0|T0],     B0,B, A0,A, [H|T]) :-
    wrap(H0,   B0,B1,A0,A1,H),
    wrap(T0,   B1,B, A1,A, T).
wrap1(term(X),     B0,B, A0,A, Y) :-
    X =.. L0,
    wrap(L0,   B0,B,A0,A,L),
    Y =.. L.
wrap1(?X, [oo(X,Y,Y)|B],B,A,  A,Y).
wrap1(!X, B,B,[oo(X,_,Y)|A],A,Y).
```

1.4.11 Accessors

Usage

```
o(Com) :- o(Com,X), X.
o(Com,X) :- o(Com,X,_).

o([],X,X).
o([H|T],X,Y)  :- o(H,X,Z), o(T,Z,Y).
o(F <- V,X,Y)   :- oo(X,F,_,V,Y).
o(F+V,X,Y)    :- oo(X,F,L,[V|L],Y).
o(F=V,X,X)    :- oo(X,F,V,V,X).

goal_expansion(o(Com),     (o(Com,_,X),X)).
goal_expansion(o(Com,X),   o(Com,X,_)).
goal_expansion(o([H],X,Y), o(H,X,Y)) :-
    nonvar(H).
goal_expansion(o([H1,H2|T],X,Y), (o(H1,X,Z),o([H2|T],Z,Y))) :-
    nonvar(H1).

goal_expansion(o(X,Y,Z),Body) :-
    clause1(o(X,Y,Z),Body).

goal_expansion(oo(X,F,V0,V,Y),_) :-
    \+ oo(X,F,V0,V,Y),
    barphln(unknown(F)).

% this works, but i dont trust it.
goal_expansion(oo(X,F,V0,V,Y),true) :-
    nonvar(F), % fail, % try un-commenting this to see if it is worth it.
    solo(oo(X,F,V0,V,Y)).
```

Field details

Each field has annotations that indicate:

1. If Prolog should index on a particular field.

2. A fields name and default falue.

3. Some rule that defines the valid range of a field.

```
detail(+X:R=D,1,      X,   R,   D).
detail(+X:R,  1,      X,   R,   _).
detail(+X=D,  1,      X,   any, D).
detail(+X,    1,      X,   any, _).
detail(X:R=D, 0,      X,   R,   D).
detail(X:R,   0,      X,   R,   _).
detail(X=D,   0,      X,   any, D).
detail(X,     0,      X,   any, _).
```

These details are stored in a `meta/9` fact which we can manipulate as follows:

```
oo(meta(A,B,C,D,E,F,G,H,I),X,Y,Z,Out) :- % ◄....................... 277
    meta_(X,Y,Z,meta(A,B,C,D,E,F,G,H,I),Out).

meta_(rel0_, X,Y,meta(X,B,C,D,E,F,G,H,I),meta(Y,B,C,D,E,F,G,H,I)).
meta_(arity_,X,Y,meta(A,X,C,D,E,F,G,H,I),meta(A,Y,C,D,E,F,G,H,I)).
meta_(rel_   ,X,Y,meta(A,B,X,D,E,F,G,H,I),meta(A,B,Y,D,E,F,G,H,I)).
meta_(index_,X,Y,meta(A,B,C,X,E,F,G,H,I),meta(A,B,C,Y,E,F,G,H,I)).
meta_(names_,X,Y,meta(A,B,C,D,X,F,G,H,I),meta(A,B,C,D,Y,F,G,H,I)).
meta_(rules_,X,Y,meta(A,B,C,D,E,X,G,H,I),meta(A,B,C,D,E,Y,G,H,I)).
meta_(inits_,X,Y,meta(A,B,C,D,E,F,X,H,I),meta(A,B,C,D,E,F,Y,H,I)).
meta_(blank_,X,Y,meta(A,B,C,D,E,F,G,X,I),meta(A,B,C,D,E,F,G,Y,I)).
meta_(vars_, X,Y,meta(A,B,C,D,E,F,G,H,X),meta(A,B,C,D,E,F,G,H,Y)).
```

Core engine

Code for being able to access and changed named fields within a term.

```
dd(Rel0,FieldsC,All) :-
    dd1s(new(Rel0,FieldsC),Meta),
291 bagof(One, Meta^dd2(Meta,One), All).

dd1s(new(Rel0,Fields0),Out) :-
294 atom_concat(Rel0,'_',Rel),
    c2l(Fields0,Fields1),
    reverse([+id_|Fields1], [_|Fields]),
297 length(Fields,Arity),
    functor(Blank,Rel0,Arity),
    Blank =.. [_|Vars],
300 dd1(Fields,
            meta(Rel0,Arity,Rel,[],[],[],[],Blank,Vars),
         Out).
303
dd1([],X,X).
dd1([H|T]) -->
306 detail(H,I,N,R,D),
    o([index_ +I, names_ +N,rules_ +R,inits_ +D]),
    dd1(T).
```

dd2/2

Reset the counter for this relation to zero.

```
dd2(X,(:- print(reseting(Rel0)),nl,flag(Rel0,_,0))) :- o(rel0_ =Rel0,X).
```

Create an index on the indexed fields.

```
dd2(X,(:- index(Index))) :-
    o([rel0_ =Rel0,index_ =Index0],X),
312 Index =.. [Rel0|Index0].
```

Note that assertions in this relation may or may not exist at a particular time.

```
dd2(X,(:- dynamic R /A)) :- o([rel0_ =R,arity_ =A],X).
```

Automatically generate arity five accessors, just like the ones written manually at line 277.

```
dd2(Meta,Out) :-
    o([names_ =Names,rel0_ =Rel0,rel_ =Rel,arity_ =Arity],Meta),
316 nth0(Pos,Names,Name),
    length(Before,Pos),
    functor(Term0, Rel0,Arity), Term0 =.. [_|L0],
319 functor(Term,  Rel0,Arity), Term  =.. [_|L1],
    append(Before,[Old|After],L0),
    append(Before,[New|After],L1),
322 Out =.. [Rel,Name,Old,New,Term0,Term].
```

Define a *bridge* predicate that calls the arity five accessor that is relevant to a particular term.

```
dd2(X,(oo(T0,Com,V0,V,T) :- Body)) :-
    o([rel0_ =Rel0,rel_ =Rel,arity_ =Arity],X),
325 functor(T0,Rel0,Arity),
    Body =.. [Rel,Com,V0,V,T0,T].
```

Finish up the meta-level fact.

```
dd2(X,Y) :-
    o([rel0_ =Rel0,inits_ =Inits0],X),
329 Inits =.. [Rel0|Inits0],
    o(inits_ <- Inits,X,Y).
```

Write our terms with names fields very succinctly.

```
dd2(X,(portray(Term) :- write(Rel/ Arity))) :-
    o([rel0_ =Rel,arity_ =Arity],X),
333 functor(Term,Rel,Arity).
```

Singlton accessors with arity 3 can be expanded to accessor 5

```
dd2(X,(goal_expansion(H,Body)  :- clause1(H,Body))) :-
    o(rel_ =Rel,X),
336 H =.. [Rel,_,_,_].
```

Singleton accessors with arity 5 can be evaluated, then replaced with `true`.

```
dd2(X,(goal_expansion(H,true) :- clause1(H,true))) :-
    o(rel_ =Rel,X),
339 functor(H,Rel,5).
```

egdd.spy

```
% output from ':- demos(egdd).'

:-flag(eg, A, 0).
:-index(eg(1, 1, 0, 1)).
:-dynamic eg/4.

eg_(id_, A, B, eg(A, C, D, E), eg(B, C, D, E)).
eg_(deptNo, A, B, eg(C, A, D, E), eg(C, B, D, E)).
eg_(name, A, B, eg(C, D, A, E), eg(C, D, B, E)).
eg_(age, A, B, eg(C, D, E, A), eg(C, D, E, B)).

oo(eg(A, B, C, D), E, F) :-
    eg_(E, eg(A, B, C, D), F).

meta(eg, 4, eg_, [1, 1, 0, 1],
        [id_, deptNo, name, age],
        [any, num, [x, y, z], num],
        eg(A, B, C, 1),
        eg(D, E, F, G)
).

portray(eg(A, B, C, D)) :-
    write(eg/4).
goal_expansion(eg_(A, B, C), D) :-
    clause1(eg_(A, B, C), D).
goal_expansion(eg_(A, B, C, D, E), true) :-
    clause1(eg_(A, B, C, D, E), true).

% runtime = 0.0100144 sec(s)
```

Figure 1.4:

Match a fact in the global db.

```
dd2(X,(get(Rel,Vars,Term) :- Term)) :-
    indexVars(X,Rel,Vars,Term).
```

Zap a fact in the global db

```
dd2(X,(zap(Rel,Vars,Term) :- retractall(Term))) :-
    indexVars(X,Rel,Vars,Term).
```

Make a fresh fact in the global db.

```
dd2(X,(commit(Rel,Vars,Y,Z) :- retractall(Y),assert(Z))) :-
    indexVars(X,Rel,Vars,Y),
346 indexVars(X,Rel,Vars,Z).
```

Create a term with the index variables pre-matched.

```
indexVars(X,Rel,Ind,Term) :-
    o([rel0_ = Rel, arity_ = Arity, index_ = Nums],X),
349 functor(Term,Rel,Arity),
    Term =.. [Rel|Args],
    indexVars1(Nums,Args,Ind).
352
indexVars1([],_,[]).
indexVars1([1|T],[X|Args],[X|Ind]) :- indexVars1(T,Args,Ind).
355 indexVars1([0|T],[_|Args],   Ind ) :- indexVars1(T,Args,Ind).
```

Demos

```
egdd :- % for output, see Figure 1.4
    expand_term(
358    (+deptNo : num
        ,name   : [x,y,z]
        ,+age   : num = 1
361     ,are the eg),
       X),
    portrays(X).
```

Generating an instance

see line ??

```
eg(Rel,FieldsC,Out) :-
    mostC2l(FieldsC,FieldsL),
395 flag(Rel,M,M+1),
    N is M + 1,
    spit(N,50,0),
398 Datum =.. [Rel,N|FieldsL],
    (okDatum(Rel,Datum) -> Out = Datum ; Out=[]).

401 okDatum(X,B) :- o([rel0_ = X,blank_ = B]),!.
okDatum(_,_) :- barphln(badness).
```

1.4.12 Statistics

Gaussians

```
stale=0, mean:num=0, n:num=0, sd:num=0,sum:num=0
,sumSquared:num=0, are the gaussian.

add(X) of gaussian -->
    * !stale=1,
    * !n is ?n + 1,
        * !sum is ?sum+X,
    * !sumSquared is ?sumSquared + X*X.

refresh of gaussian --> * ?stale=0,!.
refresh of gaussian -->
    * !mean is ?sum/ ?n,
    * ?sd is sqrt( ?sumSquared -( ?sum^2/ ?n)/( ?n-1)),
    * !stale=0.

sd(Sd) of gaussian -->
    refresh,
    * print(1),
    * ?sd = Sd.

egDefMethod :-
    SRC= (refresh of gaussian -->
            * !mean is ?sum/ ?n,
            * !sd is sqrt( ?sumSquared -( ?sum^2/ ?n)/( ?n-1)),
            * !stale=0),
    defmethod(SRC,Out),
    portrays(Out).
```

Gaussians

```
bins=[], n: num=0, are the histogram.
add(X) of histogram -->  * inc(?bins,X,!bins).
        % better prolog stuff and hunt for ?x in ordinary
%stuff
```

1.5 Main

```
%  go :-  relation := R, classSymbol := Goal,
 %        o([rel0_ = R,blank_ = B, names_ = Names]).

go :-   relation:=Rel, go(Rel).

go(Rel) :-
    setup(Rel,Names,Goal,Wme0),
    flag(Rel,Max,Max),
    go1(Max,Names,Goal,Rel,Wme0,_).

setup(Rel,Names,Goal,Wme) :-
    classSymbol := Goal,
    o([rel0_ =Rel,rules_ =Rules,names_= Names]),
    nth1(Pos,Names,Goal),
    nth1(Pos,Rules,Classes),
    maplist(setup1(Rel,Names),Classes,Wme).

setup1(Rel,Names,Class,Class=Term) :-
    o([rel0_ =Rel, blank_ = Term]),
    setup2(Names,Term).

setup2([],_).
setup2([Name|Names],Term) :-
        o([rel0_ = histogram,inits_ = H]),
    o(Name=H,Term,Term),
    setup2(Names,Term).

go1(0,_,_,_,W,W).
go1(Id,Names,Goal,Rel,W0,W) -->
    Id > 0,
    one(Rel,Id,Goal,One,Vars,Class),
    Id0 is Id - 1,
    less1(W0,Class=Counts0,W1),
    counts(Names,Vars,Counts0,Counts),
    go1(Id0,Names,Goal,Rel,[Class=Counts|W1],W).

one(Rel,Id,Goal,One,Vars,Class) :-
  o([rel0_ = Rel,blank_ = One, vars_ = Vars]),
  o([id_ =Id,Goal=Class], One,One),
  One.

counts([],_) --> [].
counts([Name|Names],[F|Fs]) --> count(Name,F), counts(Names,Fs).

count(Name,F,W0,W) :- true.

%count1(Name,Class) :-
```

1.6 The NB system

If an application stands on some library, then there is some hope that the library may be useful elsewhere even if the application is not. So, the first rule of Timm: a good application is an empty application; i.e. is just a place where supposedly reusable components work together for a while.

1.6.1 Loading

```
namesData :- names,data.

names :- relation:=R, names(R).
data  :- relation:=R, data(R).

names(X) :- atom_concat(X,'.names',Y), [Y].
data(X)  :- atom_concat(X,'.data', Y), [Y].
```

1.7 Start up actions

```
:- ['defaults.nbc' % see §??
   ,'config.nbc'   % see §??
    ].

:- commandLine.
:- ?verbose -> hello ; true.
:- namesData.
```

Acknowledgements

This research was conducted at West Virginia University; the University of British Columbia, Vancouver, Canada; and at the Jet Propulsion Laboratory, California Institute of Technology under NASA contract NCC2-0979. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

Bibliography

[1] I. Bratko. *Prolog Programming for Artificial Intelligence. (third edition)*. Addison-Wesley, 2001.

[2] P. Deransart, A. Ed-Dbali, and L. Cervoni. *Prolog: The Standard*. Sprunger, 1996.

[3] T. Menzies. Domain-specific knowledge representations. *AI Expert*, Summer 1989.

[4] R. O'Keefe. *The Craft of Prolog*. MIT Press, 1990.

[5] L. Sterling and E. Shapiro. *The Art of Prolog (second edition)*. MIT Press, 1994.

[6] J. Wielemaker. *SWI-Prolog*. Available from `http://swi.psy.uva.nl/projects/xpce/SWI-Prolog.html`.

.1 License

This software is distributed under the GNU General Public License. Routines to display that license are shown at line 93.

.1.1 nowarranty.txt

NBC comes with ABSOLUTELY NO WARRANTY: for more details type 'warranty'.

This is free software, and you are welcome to redistribute it under certain conditions: for more details, type 'conditions'.

.1.2 warranty.txt

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 (see http://www.gnu.org/copyleft/gpl.html or type 'conditions').

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, US.

.1.3 conditions.txt

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The 'Program', below, refers to any such program or work, and a 'work based on the Program' means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term 'modification'.) Each licensee is addressed as 'you'.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms

and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and 'any later version', you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS