defs.pl

Tim Menzies
Lane Deptartment of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26506
tim@menzies.us

Contents

XXXX change thing to TERM XXXX change this to FUNC XXXX change def to names

1 Accessor

Been at this one for years.

2 Header

2.1 Operators */

```
:- op(800, xfy, with).

:- op(799,fx,(‘)).
:- op(700, xfx,   :=).
:- op(1,   fx,    in).
:- op(2,  xfx,  next).
:- op(1,   fx,   the).
:- op(1,   fx,   our). /*
```

2.2 Flags */

```
:- X=(names/2),   dynamic(X),discontiguous(X),multifile(X).
:- X=(term/3), dynamic(X),discontiguous(X),multifile(X).
:- index(term(1,1,0)). /*
```

3 Body

3.1 Inside a `term` */

If i’ve done this right, this should be the only place where we can find an explicit referece to `term/3`. */

```
names(meta,[identity,functor,arity,values,fields]).

term2Meta(This,term(Id,This,Vs),term(Id,meta,[Id,This,Size,Vs,Fs])) :-
       names(This,Fs),
       length(Fs,Size),
       length(Vs,Size).

in(This,Term, Term) :-
       Term=term(_Id,This,_Values),
       \+ illegal(This,_),
       term2Meta(This,Term,_). /*
```

3.2 Helper predicates */

```
at(X)   :- at(X,_,_).
at(X,Y) :- at(X,_,Y).

at(F/V0/V)   --> at(F,V0,V).
at(F := V)   --> at(F/_/V).
at(F=V)      --> at(F/V/V).
at(F is N)   --> at(F/_/V),   {V is N}.
at(F+N)      --> at(F/V0/V),  {V is V0+N}.
at(+F)       --> at(F/V0/V),  {V is V0+1}.
at(-F)       --> at(F/V0/V),  {V is V0-1}.
at(F >= V)   --> at(F/V1/V1), {V1 >= V}.
at(F >  V)   --> at(F/V1/V1), {V1 >  V}.
at(F <  V)   --> at(F/V1/V1), {V1 <  V}.
at(F =< V)   --> at(F/V1/V1), {V1 =< V}.
at(F \= V)   --> at(F/V1/V1), {V1 \= V}.
at(call(X))  --> {X}.
at(‘X)       --> {wrapper(X,Y)}, at(Y).
at(X with Y) --> at(X),at(Y).
at(in X)     --> in(X).  /*
```

3.3 Worker predicates

Here’s were fields are found/changed. */

```
at(our X,Old,New,Term0,Term) :-
       term2Meta(This,Term0,Meta0),
       term2Meta(This,Term, Meta),
       at(the X,Old,New,Meta0,Meta).

at(the Field,Old,New,term(Id,This,Before),term(Id,This,After)) :-
       \+ illegal(This,the Field),
       names(This,Fields),
       at1(Fields,Field,Old,New,Before,After).

at1([Field|_],Field,Old,New,[Old|Rest],[New|Rest]).
at1([_|Fields],Field,Old,New,[H|T0],[H|T1]) :-
       at1(Fields,Field,Old,New,T0,T1).  /*
```

3.4 The wrapper */

```
wrapper(X,Out) :-
   wrap(X,Before,[],After,[],Goal),
   append(Before,[call(Goal)|After],Temp),
   l2w(Temp,Out).

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(the X,           the X ).
wrap0(the next X, the next 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(the X,[the X=Y|B],B,A,A,Y).
wrap1(the next X,B,B,[the X:=Y|A],A,Y).

l2w([A,B|C],(A with D)) :- l2w([B|C],D).
l2w([A],A). /*
```

3.5 Error Handler */

```
illegal(T,F) :-
       aboutTerm(T,GT,PT),
       aboutTerm(F,GF,PF),
       \+ legal(GT,GF,T,F),
       write('% E> '),
       illegal1('badness in "~w" of "~w"\n',[PF,PT]).

illegal1(Err,Args) :-
       (source_location(Path,Line),
       file_base_name(Path,File)
       -> format('~w, line ~w: ',[File,Line])
       ;  true),
       format(Err,Args).

aboutTerm(X,0,(?)) :- var(X).
aboutTerm(X,1,X) :- nonvar(X).

legal(0,_,_,_).
legal(1,0,T,_)     :- names(T,_).
legal(1,1,T,the F) :- names(T,Fs), member(F,Fs).
```