

Online appendix for the paper  
*Evaluation of the Implementation of an Abstract  
 Interpretation Algorithm using Tabled CLP* \*

published in *Theory and Practice of Logic Programming*

JOAQUÍN ARIAS and MANUEL CARRO

*IMDEA Software Institute and Universidad Politécnica de Madrid*  
 joaquin.arias@imdea.org,alumnos.upm.es}, manuel.carro@imdea.org,upm.es}

*submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003*

### Appendix A PLAI Algorithm Implementation Using TCLP

In this appendix we include the code corresponding to the reimplementaion of PLAI using TCLP. It is not expected to be used to understand the code (we did not add any facility or improve its functionality), but rather to compare the code length and complexity with that of the original PLAI in CiaoPP, which we include in Appendix B. Therefore, we have removed the comments that appear in the original files. The files with comments can be accessed at <http://www.cliplab.org/papers/tclp-plai-iclp2019>.

```

1  /*          Copyright (C)1990-2019 UPM-CLIP          */
2
3  :- module(fixpo_plai_tabling,
4          [
5            query/8,
6            init_fixpoint/0,
7            cleanup_fixpoint/1,
8            entry_to_exit/9
9          ],
10         [assertions, datafacts]).
11
12 % Ciao library
13 :- use_module(engine(io_basic)).
14
15 :- use_module(library(aggregates), [bagof/3, (^)/2]).
16 :- use_module(library(lists), [member/2, append/3]).
17 :- use_module(library(terms_vars), [varset/2]).
18 :- use_module(library(terms_check)).
19 :- use_module(library(sets), [merge/3, ord_subtract/3]).
20 :- use_module(library(sort), [sort/2]).
21 :- use_module(library(messages)).
22 :- use_module(library(write)).
23
24 % CiaoPP library
25 :- use_module(ciaopp(preprocess_flags), [current_pp_flag/2, set_pp_flag/2]).
26

```

\* Work partially supported by EIT Digital (<https://eitdigital.eu>), MINECO project TIN2015-67522-C3-1-R (TRACES), and Comunidad de Madrid project S2018/TCS-4339 BLOQUES-CM co-funded by EIE Funds of the European Union.

```

27 % Plai library
28 :- use_module(ciaopp(plai/fixpo_ops), [inexistent/2, variable/2, bottom/1,
29     singleton/2, fixpoint_id_reuse_prev/5, fixpoint_id/1, fixp_id/1,
30     each_abs_sort/3,
31     % each_concrete/4,
32     each_extend/6, each_project/6, each_exit_to_prime/8, each_unknown_call/4,
33     each_body_succ_builtin/12, body_succ_meta/7, reduce_equivalent/3,
34     each_apply_trusted/7, widen_succ/4, decide_memo/6, clause_applies/2,
35     abs_subset_/3]).
36
37 :- use_module(ciaopp(plai/domains)).
38 :- use_module(ciaopp(plai/trace_fixp), [fixpoint_trace/7, cleanup/0]).
39 :- use_module(ciaopp(plai/plai_db),
40     [ complete/7, memo_call/5, memo_table/6, cleanup_plai_db/1, patch_parents/6 ]).
41 :- use_module(ciaopp(plai/psets), [update_if_member_idlist/3]).
42 :- use_module(ciaopp(plai/re_analysis), [erase_previous_memo_tables_and_parents/4]).
43 :- use_module(ciaopp(plai/transform), [body_info0/4, trans_clause/3]).
44 :- use_module(ciaopp(plai/apply_assertions_old),
45     [ apply_trusted0/7,
46     cleanup_trusts/1 ]).
47
48 :- doc(author, "Joaquin Arias").
49
50 :- doc(module, "This module adapts the implementation of the top-down
51     fixpoint algorithm of PLAI using TCLP with aggregates and an
52     extension that also checks call entailment.").
53
54 init_fixpoint.
55
56 cleanup_fixpoint(_AbsInt).
57
58 %------%
59 % call_to_success(+,+,+,+,+,+) %
60 %------%
61
62 call_to_success(SgKey, Call, Proj, Sg, Sv, AbsInt, Succ) :-
63     call_to_success_fixpoint(SgKey, Sg, st(Sv, Call, Proj, AbsInt, Prime)),
64     each_extend(Sg, Prime, AbsInt, Sv, Call, Succ).
65
66 %%%%%%%%%%% TCLP interface %%%%%%%%%%%
67 :- use_package(tclp_aggregate).
68 :- table call_to_success_fixpoint(_,_, abst_lub).
69
70 call_entail(abst_lub, st(V,_,ProjA,AbsInt,_), st(V,_,ProjB,AbsInt,_)) :-
71     identical_abstract(AbsInt, ProjA, ProjB), !.
72
73 answer_entail(abst_lub, st(V,_,_,AbsInt,PrimeAs), st(V,_,_,AbsInt,PrimeBs), 1) :-
74     singleton(PrimeA, PrimeAs),
75     singleton(PrimeB, PrimeBs),
76     less_or_equal(AbsInt, PrimeA, PrimeB), !.
77
78 answer_join(abst_lub, st(V,_,_,AbsInt,PrimeAs), st(V,_,_,AbsInt,PrimeBs),
79     st(V,_,_,AbsInt,PrimeNews)) :-
80     singleton(PrimeA, PrimeAs),
81     singleton(PrimeB, PrimeBs),
82     singleton(PrimeNew, PrimeNews),
83     compute_lub(AbsInt, [PrimeA, PrimeB], PrimeNew), !.
84
85 apply_answer(abst_lub, st(V,_,_,Ab,A), st(V,_,_,Ab,B)) :- A = B.
86
87 call_to_success_fixpoint(SgKey, Sg, st(Sv, Call, Proj, AbsInt, Primes)) :-
88     trans_clause(SgKey,_, Clause),
89     do_nr_cl(Clause, Sg, Sv, Call, Proj, AbsInt, Primes).
90 call_to_success_fixpoint(SgKey, Sg, st(Sv, _Call, Proj, AbsInt, Primes)) :-
91     \+ trans_clause(SgKey,_,_),
92     apply_trusted0(Proj, SgKey, Sg, Sv, AbsInt, _ClId, Prime),

```

```

93     singleton(Prime,Primes).
94
95 do_nr_cl(Clause,Sg,Sv,Call,Proj,AbsInt,Primes):-
96     Clause = clause(Head,Vars_u,K,Body),
97     clause_applies(Head,Sg), !,
98     varset(Head,Hv),
99     sort(Vars_u,Vars),
100    ord_subtract(Vars,Hv,Fv),
101    process_body(Body,K,AbsInt,Sg,Hv,Fv,Vars_u,Head,Sv,Call,
102                Proj,Primes,_Id).
103 do_nr_cl(_Clause,_Sg,_Sv,_Call,_Proj,_AbsInt,[[]]).
104
105 process_body(Body,K,AbsInt,Sg,Hv,_Fv,_Head,Sv,Call,Proj,LPrime,_Id):-
106     Body = g(_,[],'$built'(_,true,_),'true/0',true), !,
107     singleton(Prime,LPrime),
108     call_to_success_fact(AbsInt,Sg,Hv,Head,K,Sv,Call,Proj,Prime,_Succ).
109 process_body(Body,K,AbsInt,Sg,Hv,Fv,Vars_u,Head,Sv,_Proj,Prime,Id):-
110     call_to_entry(AbsInt,Sv,Sg,Hv,Head,K,Fv,Proj,Entry,ExtraInfo),
111     singleton(Entry,LEntry),
112     entry_to_exit(Body,K,LEntry,Exit,[],_,Vars_u,AbsInt,Id),
113     each_exit_to_prime(Exit,AbsInt,Sg,Hv,Head,Sv,ExtraInfo,Prime).
114
115 %------%
116 % entry_to_exit(+,+,+,-,-,+,+,+) %
117 %------%
118
119 entry_to_exit((Sg,Rest),K,Call,Exit,OldList,NewList,Vars_u,AbsInt,NewN):- !,
120     body_succ(Call,Sg,Succ,OldList,IntList,Vars_u,AbsInt,K,NewN,_),
121     entry_to_exit(Rest,K,Succ,Exit,IntList,NewList,Vars_u,AbsInt,NewN).
122 entry_to_exit(true,_Call,Call,List,List,_,_):- !.
123 entry_to_exit(Sg,Key,Call,Exit,OldList,NewList,Vars_u,AbsInt,NewN):-
124     body_succ(Call,Sg,Exit,OldList,NewList,Vars_u,AbsInt,Key,NewN,_),
125     true.
126
127 body_succ(Call,_Atom,Succ,List,List,_HvFv_u,_AbsInt,_ClId,_ParentId,no):-
128     bottom(Call), !,
129     Succ = Call.
130 body_succ(Call,Atom,Succ,List,NewList,HvFv_u,AbsInt,ClId,ParentId,Id):-
131     Atom=g(Key,Sv,Info,SgKey,Sg),
132     body_succ_(Info,SgKey,Sg,Sv,HvFv_u,Call,Succ,List,NewList,AbsInt,
133                ClId,Key,ParentId,Id).
134
135 body_succ_(Info,SgKey,Sg,Sv,HFv,Call,Succ,L,NewL,AbsInt,ClId,Key,PIId,Id):-
136     Info = [_|_], !,
137     split_combined_domain(AbsInt,Call,Calls,Domains),
138     map_body_succ(Info,SgKey,Sg,Sv,HFv,Calls,Succs,L,NewL,Domains,
139                  ClId,Key,PIId,Id),
140     split_combined_domain(AbsInt,Succ,Succs,Domains).
141 body_succ_(Info,SgKey,Sg,Sv,HFv,Call,Succ,L,NewL,AbsInt,ClId,Key,PIId,Id):-
142     body_succ0(Info,SgKey,Sg,Sv,HFv,Call,Succ,L,NewL,AbsInt,
143                ClId,Key,PIId,Id).
144
145 map_body_succ([],_SgKey,_Sg,_Sv,_HFv,[],[],L,L,[],_ClId,_Key,_PIId,no).
146 map_body_succ([I|Info],SgKey,Sg,Sv,HFv,[Call|Calls],[Succ|Succs],L,NewL,
147              [AbsInt|Domains],ClId,Key,PIId,Id):-
148     body_succ0(I,SgKey,Sg,Sv,HFv,Call,Succ,L,_NewL,AbsInt,
149                ClId,Key,PIId,Id), !,
150     map_body_succ(Info,SgKey,Sg,Sv,HFv,Calls,Succs,L,NewL,Domains,
151                  ClId,Key,PIId,Id).
152
153 body_succ0('$var',SgKey,Sg,_Sv_u,HvFv_u,Calls,Succs,List0,List,AbsInt,
154            _ClId,F,_N,_Id):-
155     !,
156     (Calls=[Call],
157      concrete(AbsInt,Sg,Call,Concretes),
158      concretes_to_body(Concretes,SgKey,AbsInt,B)

```

```

159     -> meta_call(B,HvFv_u,Calls,[],Succs,List0,List,AbsInt,_ClId,_Id,_Ids)
160     ; List=List0,
161     each_unknown_call(Calls,AbsInt,[Sg],Succs) % Sg is a variable
162     ).
163 body_succ0('$meta'(T,B,_),SgKey,Sg,Sv_u,HvFv_u,Call,Succ,List0,List,AbsInt,
164     _ClId,_F,_N,_Id):-
165     !,
166     meta_call(B,HvFv_u,Call,[],Exits,List0,List,AbsInt,ClId,Id,_Ids),
167     ( body_succ_meta(T,AbsInt,Sv_u,HvFv_u,Call,Exits,Succ) ->
168     true
169     ; % for the trusts, if any:
170     varset(Sg,Sv_r), % Sv_u contains extra vars (from meta-term)
171     % which will confuse apply_trusted
172     body_succ0(nr,SgKey,Sg,Sv_r,HvFv_u,Call,Succ,[],_List,AbsInt,
173     _ClId,_F,_N,_Id0)
174     ).
175 body_succ0('$built'(T,Tg,Vs),SgKey,Sg,Sv_u,HvFv_u,Call,Succ,List0,List,AbsInt,
176     _ClId,_F,_N,_Id):-
177     !,
178     List=List0,
179     sort(Sv_u,Sv),
180     each_body_succ_builtin_(Call,AbsInt,T,Tg,Vs,SgKey,Sg,Sv,HvFv_u,Succ).
181 body_succ0(_RFlag,SgKey,Sg,Sv_u,HvFv_u,Call,Succ,_List0,_List,AbsInt,
182     _ClId,_F,_N,_Id):-
183     sort(Sv_u,Sv),
184     each_call_to_success(Call,SgKey,Sg,Sv,HvFv_u,AbsInt,Succ).
185
186 %% predicate adapted from fixpo_ops
187 each_body_succ_builtin_([],_T,_Tg,_Sg,_Sv,_HvFv_u,[]).
188 each_body_succ_builtin_([Call|Calls],AbsInt,T,Tg,Vs,SgKey,Sg,Sv,HvFv_u,[Succ|Succs]):-
189     project(AbsInt,Sg,Sv,HvFv_u,Call,Proj),
190     body_succ_builtin(T,AbsInt,Tg,Vs,Sv,HvFv_u,Call,Proj,Succ),!, %% Doamin call
191     each_body_succ_builtin_tabling_(Calls,AbsInt,T,Tg,Vs,SgKey,Sg,Sv,HvFv_u,Succs).
192
193 each_call_to_success([Call],SgKey,Sg,Sv,HvFv_u,AbsInt,Succ):-
194     !,
195     project(AbsInt,Sg,Sv,HvFv_u,Call,Proj),
196     call_to_success(SgKey,Call,Proj,Sg,Sv,AbsInt,Succ).
197
198 each_call_to_success(LCall,SgKey,Sg,Sv,HvFv_u,AbsInt,LSucc):-
199     each_call_to_success0(LCall,SgKey,Sg,Sv,HvFv_u,AbsInt,
200     LSucc).
201
202 each_call_to_success0([],_SgK,_Sg,_Sv,_HvFv,_AbsInt,[]).
203 each_call_to_success0([Call|LCall],SgKey,Sg,Sv,HvFv_u,AbsInt,
204     LSucc):-
205     project(AbsInt,Sg,Sv,HvFv_u,Call,Proj),
206     call_to_success(SgKey,Call,Proj,Sg,Sv,AbsInt,LSucc0),
207     append(LSucc0,LSucc1,LSucc),
208     each_call_to_success0(LCall,SgKey,Sg,Sv,HvFv_u,AbsInt,
209     LSucc1).
210
211 meta_call([],_HvFv_u,Call,[],Call,List,List,_AbsInt,_ClId,_Id,[]).
212 meta_call([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,ClId,Id,Ids):-
213     meta_call_([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,ClId,Id,Ids).
214 meta_call_([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,ClId,Id,Ids):-
215     meta_call_body(Body,ClId,Call,Succ1,L0,L1,HvFv_u,AbsInt,Id,Ids0),
216     widen_succ(AbsInt,Succ0,Succ1,Succ2),
217     append(Succ0,Succ1,Succ2),
218     append(Ids0,Ids1,Ids),
219     meta_call_(Bodies,HvFv_u,Call,Succ2,Succ,L1,List,AbsInt,ClId,Id,Ids1).
220 meta_call_([],_HvFv_u,_Call,Succ,Succ,List,List,_AbsInt,_ClId,_Id,[]).
221
222 meta_call_body((Sg,Rest),K,Call,Exit,OldList,NewList,Vars_u,AbsInt,PIId,CIDs):-
223     !,
224     CIDs=[Id|Ids],

```

```

225     body_succ(Call,Sg,Succ,OldList,IntList,Vars_u,AbsInt,K,PId,Id),
226     meta_call_body(Rest,K,Succ,Exit,IntList,NewList,Vars_u,AbsInt,PId,Ids).
227 meta_call_body(true,_,Call,Call,List,List,_,_,_,[no]):-!.
228 meta_call_body(Sg,Key,Call,Exit,OldList,NewList,Vars_u,AbsInt,PId,[Id]):-
229     body_succ(Call,Sg,Exit,OldList,NewList,Vars_u,AbsInt,Key,PId,Id).
230
231 concretes_to_body([],_SgKey,_AbsInt,[]).
232 concretes_to_body([Sg|Sgs],SgKey,AbsInt,[B|Bs]):-
233     body_info0(Sg:SgKey,[],AbsInt,B),
234     concretes_to_body(Sgs,SgKey,AbsInt,Bs).
235
236 %------%
237 % query(+,+,+,+,+,+,+,+) %
238 %------%
239
240 :- doc(query(AbsInt,QKey,Query,Qv,RFlag,N,Call,Succ),
241     "The success pattern of @var{Query} with @var{Call} is
242     @var{Succ} in the analysis domain @var{AbsInt}. The predicate
243     called is identified by @var{QKey}. The goal @var{Query} has
244     variables @var{Qv}.").
245
246 query(AbsInt,QKey,Query,Qv,_RFlag,_N,Call,Succ):-
247     project(AbsInt,Query,Qv,Qv,Call,Proj),
248     call_to_success(QKey,Call,Proj,Query,Qv,AbsInt,Succ),!.
249
250 query(_AbsInt,_QKey,_Query,_Qv,_RFlag,_N,_Call,_Succ):-
251     % should never happen, but...
252     error_message("SOMETHING HAS FAILED!").

```

## Appendix B PLAI Algorithm Implementation in Ciao Prolog

We include here the Ciao Prolog implementation of PLAI. As mentioned before, we have removed the comments from the file since the goal of this appendix it to make it easier for the reader to compare the Ciao Prolog code w.r.t. the code using TCLP, which we include in Appendix A. The original version is available at <http://www.cliplab.org/papers/tclp-plai-iclp2019>.

```

1  /*          Copyright (C)1990-2019 UPM-CLIP          */
2
3  :- module(fixpo_plai_with_comments,
4     [ query/8,
5       init_fixpoint/0,
6       cleanup_fixpoint/1,
7       entry_to_exit/9
8     ],
9     [assertions, datafacts]).
10
11 % Ciao library
12 :- use_module(library(agggregates), [bagof/3, (^)/2]).
13 :- use_module(library(lists), [member/2, append/3]).
14 :- use_module(library(terms_vars), [varset/2]).
15 :- use_module(library(sets), [merge/3, ord_subtract/3]).
16 :- use_module(library(sort), [sort/2]).
17 :- use_module(library(messages)).
18
19 % CiaoPP library
20 :- use_module(ciaoopp(preprocess_flags), [current_pp_flag/2, set_pp_flag/2]).
21
22 % Plai library
23 :- use_module(ciaoopp(plai/fixpo_ops), [inexistent/2, variable/2, bottom/1,
24     singleton/2, fixpoint_id_reuse_prev/5, fixpoint_id/1, fixp_id/1,

```

```

25     each_abs_sort/3,
26     each_extend/6, each_project/6, each_exit_to_prime/8, each_unknown_call/4,
27     each_body_succ_builtin/12, body_succ_meta/7, reduce_equivalent/3,
28     each_apply_trusted/7, widen_succ/4, decide_memo/6, clause_applies/2,
29     abs_subset_/3]).
30
31 :- use_module(ciaopp(plai/domains)).
32 :- use_module(ciaopp(plai/trace_fixp), [fixpoint_trace/7, cleanup/0]).
33 :- use_module(ciaopp(plai/plai_db),
34     [ complete/7, memo_call/5, memo_table/6, cleanup_plai_db/1, patch_parents/6 ]).
35 :- use_module(ciaopp(plai/psets), [update_if_member_idlist/3]).
36 :- use_module(ciaopp(plai/re_analysis), [erase_previous_memo_tables_and_parents/4]).
37 :- use_module(ciaopp(plai/transform), [body_info0/4, trans_clause/3]).
38 :- use_module(ciaopp(plai/apply_assertions_old),
39     [ apply_trusted0/7,
40       cleanup_trusts/1 ]).
41
42 :- doc(author, "Kalyan Muthukumar").
43 :- doc(author, "Maria Garcia de la Banda").
44 :- doc(author, "Francisco Bueno").
45
46 :- doc(module, "This module implements the top-down fixpoint
47     algorithm of PLAI, both in its mono-variant and multi-variant
48     on successes versions. It is always multi-variant on calls.
49     The algorithm is parametric on the particular analysis domain.").
50
51
52 :- data '$depend_list'/3.
53 :- data ch_id/2.
54
55 :- data approx/6.
56 :- data fixpoint/6.
57 :- data fixpoint_variant/6.
58 :- data approx_variant/7.
59
60 init_fixpoint:-
61     retractall_fact(approx(_,_,_,_,_)),
62     retractall_fact(fixpoint(_,_,_,_,_)),
63     retractall_fact('$depend_list'(_,_,_)),
64     retractall_fact(ch_id(_,_)),
65     retractall_fact(fixpoint_variant(_,_,_,_,_)),
66     retractall_fact(approx_variant(_,_,_,_,_)),
67     trace_fixp:cleanup.
68
69 cleanup_fixpoint(AbsInt):-
70     cleanup_plai_db(AbsInt),
71     cleanup_trusts(AbsInt),
72     retractall_fact(fixp_id(_)),
73     asserta_fact(fixp_id(0)), % there is no way to recover this
74     init_fixpoint.         % if several analyses coexist!
75
76 approx_to_completes(AbsInt):-
77     current_fact(approx(SgKey,Sg,Proj,Prime,Pid,Fs),Ref),
78     asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Pid,Fs)),
79     erase(Ref),
80     fail.
81 approx_to_completes(AbsInt):-
82     current_fact(approx_variant(_Id,Pid,SgKey,Sg,Proj,Prime,Fs),Ref),
83     asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Pid,Fs)),
84     erase(Ref),
85     fail.
86 approx_to_completes(_AbsInt).
87
88
89 %-----%
90 % call_to_success(+,+,+,+,+,+,+,-,+,+,+) %

```

```

91  %-----%
92
93  call_to_success(_RFlag,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id):-
94      % ClId = number identifying the clause?... for an entry point is 0...
95      % F = program point of the call. clauseId+0 for an entry call
96      current_fact(complete(SgKey,AbsInt,Subg,Proj1,Prime1,Id,Fs),R),
97      identical_proj(AbsInt,Sg,Proj,Subg,Proj1), !,
98      patch_parents(R,complete(SgKey,AbsInt,Subg,Proj1,Prime1,Id,Ps),F,N,Ps,Fs),
99      List = [],
100     each_abs_sort(Prime1,AbsInt,Prime),
101     each_extend(Sg,Prime,AbsInt,Sv,Call,Succ).
102  call_to_success(r,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id) :-
103     current_fact(approx(SgKey,Subg,Proj1,Prime1,Id,Fs),Ref),
104     identical_proj(AbsInt,Sg,Proj,Subg,Proj1), !,
105     each_abs_sort(Prime1,AbsInt,TempPrime),
106     current_fact('$depend_list'(Id,SgKey,IdList)),
107     call_to_success_approx(SgKey,Subg,Call,Proj,Proj1,Sg,Sv,AbsInt,F,N,Fs,
108                          Id,Ref,IdList,Prime1,TempPrime,List,Prime),
109     each_extend(Sg,Prime,AbsInt,Sv,Call,Succ).
110  call_to_success(r,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id) :-
111     current_fact(fixpoint(SgKey,Subg,Proj1,Prime1,Id,Fs),Ref),
112     identical_proj(AbsInt,Sg,Proj,Subg,Proj1), !,
113     patch_parents(Ref,fixpoint(SgKey,Subg,Proj1,Prime1,Id,Ps),F,N,Ps,Fs),
114     current_fact(ch_id(Id,Num)),
115     List = [Id/Num],
116     each_abs_sort(Prime1,AbsInt,Prime),
117     each_extend(Sg,Prime,AbsInt,Sv,Call,Succ).
118  call_to_success(_RFlag,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id):-
119     current_pp_flag(variants,on),
120     current_fact(complete(SgKey,AbsInt,Subg,Proj1,Prime1,_Id1,_Fs),_R),
121     identical_proj_1(AbsInt,Sg,Proj,Subg,Proj1,Prime1,Prime2), !,
122     format("call to success tipe _RFlag SgKey",[]),
123     ( current_pp_flag(reuse_fixp_id,on) ->
124       fixpoint_id_reuse_prev(SgKey,AbsInt,Sg,Proj,Id)
125     ;
126       fixpoint_id(Id)
127     ),
128     each_abs_sort(Prime2,AbsInt,Prime),
129     List = [],
130     asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Id,[(F,N)])),
131     each_extend(Sg,Prime,AbsInt,Sv,Call,Succ).
132  call_to_success(r,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id) :-
133     current_pp_flag(variants,on),
134     current_fact(approx(SgKey,Subg,Proj1,Prime1,Id1,Fs),Ref),
135     identical_proj_1(AbsInt,Sg,Proj,Subg,Proj1,Prime1,Prime2), !,
136     each_abs_sort(Prime2,AbsInt,TempPrime),
137     current_fact('$depend_list'(Id1,SgKey,IdList)),
138     call_to_success_approx_variant(SgKey,Subg,Call,Proj,Proj1,Sg,Sv,AbsInt,F,N,Fs,
139                                   Id,Id1,Ref,IdList,Prime1,TempPrime,List,Prime),
140     each_extend(Sg,Prime,AbsInt,Sv,Call,Succ).
141  call_to_success(r,SgKey,Call,Proj,Sg,Sv,AbsInt,_ClId,Succ,List,F,N,Id) :-
142     current_pp_flag(variants,on),
143     current_fact(fixpoint(SgKey,Subg,Proj1,Prime1,Id1,_Fs),_Ref),
144     identical_proj_1(AbsInt,Sg,Proj,Subg,Proj1,Prime1,Prime2), !,
145     (
146       current_fact(fixpoint_variant(Id1,Id,SgKey,Sgv,Projv,Fsv),Refv),
147       identical_proj(AbsInt,Sg,Proj,Sgv,Projv) ->
148       patch_parents(Refv,fixpoint_variant(Id1,Id,SgKey,Sgv,Projv,Ps),F,N,Ps,Fsv)
149     ;
150     (
151       current_pp_flag(reuse_fixp_id,on) ->
152       fixpoint_id_reuse_prev(SgKey,AbsInt,Sg,Proj,Id)
153     ;
154       fixpoint_id(Id)
155     ),
156     asserta_fact(fixpoint_variant(Id1,Id,SgKey,Sg,Proj,[(F,N)]))

```

```

157     ),
158     each_abs_sort(Prime2, AbsInt, Prime),
159     current_fact(ch_id(Id1, Num)),
160     List = [Id1/Num],
161     each_extend(Sg, Prime, AbsInt, Sv, Call, Succ).
162 call_to_success(r, SgKey, Call, Proj, Sg, Sv, AbsInt, _ClId, Succ, List, F, N, Id) :-
163     init_fixpoint0(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, [(F, N)], Id, List, Prime),
164     each_extend(Sg, Prime, AbsInt, Sv, Call, Succ).
165 call_to_success(nr, SgKey, Call, Proj, Sg, Sv, AbsInt, ClId, Succ, [], F, N, Id) :-
166     ( current_pp_flag(reuse_fixp_id, on) ->
167         fixpoint_id_reuse_prev(SgKey, AbsInt, Sg, Proj, Id)
168     ;
169         fixpoint_id(Id)
170     ),
171     proj_to_prime_nr(SgKey, Sg, Sv, Call, Proj, AbsInt, ClId, Prime, Id),
172     asserta_fact(complete(SgKey, AbsInt, Sg, Proj, Prime, Id, [(F, N)])),
173     each_extend(Sg, Prime, AbsInt, Sv, Call, Succ).
174
175 call_to_success_approx(SgKey, Subg, _Call, Proj, Proj1, Sg, _Sv, _AbsInt, F, N, Fs,
176     Id, Ref, IdList, Prime1, TempPrime, List, Prime) :-
177     not_modified(IdList), !,
178     patch_parents(Ref, approx(SgKey, Subg, Proj1, Prime1, Id, Ps), F, N, Ps, Fs),
179     Prime = TempPrime,
180     List = IdList.
181 call_to_success_approx(SgKey, _Subg, Call, Proj, _Proj1, Sg, Sv, AbsInt, F, N, Fs,
182     Id, Ref, _IdList, _Prime1, TempPrime, List, Prime) :-
183     erase(Ref),
184     init_fixpoint_(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, Fs, Id,
185         TempPrime, List, Prime).
186
187 aproxs_to_fixpoint_variant(Id) :-
188     current_fact(approx_variant(Id, Idv, SgKey, Sgv, Projv, _Primev, Fs), Ref), !,
189     erase(Ref),
190     asserta_fact(fixpoint_variant(Id, Idv, SgKey, Sgv, Projv, Fs)),
191     aproxs_to_fixpoint_variant(Id).
192 aproxs_to_fixpoint_variant(_).
193
194
195 call_to_success_approx_variant(SgKey, _Subg, _Call, Proj, _Proj1, Sg, _Sv, AbsInt, F, N, _Fs,
196     Id, Id1, _Ref, IdList, _Prime1, TempPrime, List, Prime) :-
197     not_modified(IdList), !,
198     (
199         current_fact(approx_variant(Id1, Id, SgKey, Sgv, Projv, Primev, Fsv), Refv),
200         identical_proj(AbsInt, Sg, Proj, Sgv, Projv) ->
201         patch_parents(Refv, approx_variant(Id1, Id, SgKey, Sgv, Projv, Primev, Ps), F, N, Ps, Fsv)
202     ;
203         (
204             current_pp_flag(reuse_fixp_id, on) ->
205             fixpoint_id_reuse_prev(SgKey, AbsInt, Sg, Proj, Id)
206         ;
207             fixpoint_id(Id)
208         ),
209         asserta_fact(approx_variant(Id1, Id, SgKey, Sg, Proj, TempPrime, [(F, N)]))
210     ),
211     Prime = TempPrime,
212     List = IdList.
213 call_to_success_approx_variant(SgKey, Subg, Call, Proj, Proj1, Sg, Sv, AbsInt, F, N, Fs,
214     Id, Id1, Ref, _IdList, Prime1, _TempPrime, List, Prime) :-
215     (
216         current_fact(approx_variant(Id1, Id, SgKey, Sgv, Projv, _Primev, Fsv), Refv),
217         identical_proj(AbsInt, Sg, Proj, Sgv, Projv) ->
218         erase(Refv),
219         ( member((F, N), Fsv) -> NewFs = Fsv ; NewFs = [(F, N) | Fsv] %)
220     ;
221         (
222             current_pp_flag(reuse_fixp_id, on) ->

```

```

223         fixpoint_id_reuse_prev(SgKey, AbsInt, Sg, Proj, Id)
224     ;
225     fixpoint_id(Id)
226 ),
227     NewFs = [(F, N)]
228 ),
229     aproxs_to_fixpoint_variant(Id1),
230     erase(Ref),
231     asserta_fact(fixpoint_variant(Id1, Id, SgKey, Sg, Proj, NewFs)),
232     varset(Subg, Subv),
233     init_fixpoint_(SgKey, Call, Proj1, Subg, Subv, AbsInt, F, N, Fs, Id1,
234                   Prime1, List, Prime0),
235     each_exit_to_prime(Prime0, AbsInt, Sg, Subv, Subg, Sv, (no, Proj), Prime).
236
237 init_fixpoint0(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, Fs, Id, List, Prime):-
238     init_fixpoint2(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, Fs, Id, List, Prime).
239
240 init_fixpoint1(SgKey, _Call, Proj, Sg, _Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
241     current_fact(complete(SgKey, AbsInt, Subg, Proj1, Prime1, Id, Fs), R),
242     identical_proj(AbsInt, Sg, Proj, Subg, Proj1), !,
243     patch_parents(R, complete(SgKey, AbsInt, Subg, Proj1, Prime1, Id, Ps), F, N, Ps, Fs),
244     List = [],
245     each_abs_sort(Prime1, AbsInt, Prime).
246 init_fixpoint1(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
247     current_fact(approx(SgKey, Subg, Proj1, Prime1, Id, Fs), Ref),
248     identical_proj(AbsInt, Sg, Proj, Subg, Proj1), !,
249     each_abs_sort(Prime1, AbsInt, TempPrime),
250     current_fact('$depend_list'(Id, SgKey, IdList)),
251     call_to_success_approx(SgKey, Subg, Call, Proj, Proj1, Sg, Sv, AbsInt, F, N, Fs,
252                           Id, Ref, IdList, Prime1, TempPrime, List, Prime).
253 init_fixpoint1(SgKey, _, Proj, Sg, _Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
254     current_fact(fixpoint(SgKey, Subg, Proj1, Prime1, Id, Fs), Ref),
255     identical_proj(AbsInt, Sg, Proj, Subg, Proj1), !,
256     patch_parents(Ref, fixpoint(SgKey, Subg, Proj1, Prime1, Id, Ps), F, N, Ps, Fs),
257     current_fact(ch_id(Id, Num)),
258     List = [Id/Num],
259     each_abs_sort(Prime1, AbsInt, Prime).
260 init_fixpoint1(SgKey, _Call, Proj, Sg, _Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
261     current_pp_flag(variants, on),
262     current_fact(complete(SgKey, AbsInt, Subg, Proj1, Prime1, _Id1, _Fs), _R),
263     identical_proj_1(AbsInt, Sg, Proj, Subg, Proj1, Prime1, Prime2), !,
264     ( current_pp_flag(reuse_fixp_id, on) ->
265       fixpoint_id_reuse_prev(SgKey, AbsInt, Sg, Proj, Id)
266     ;
267       fixpoint_id(Id)
268     ),
269     each_abs_sort(Prime2, AbsInt, Prime),
270     List = [],
271     asserta_fact(complete(SgKey, AbsInt, Sg, Proj, Prime, Id, [(F, N)])).
272 init_fixpoint1(SgKey, Call, Proj, Sg, Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
273     current_pp_flag(variants, on),
274     current_fact(approx(SgKey, Subg, Proj1, Prime1, Id1, Fs), Ref),
275     identical_proj_1(AbsInt, Sg, Proj, Subg, Proj1, Prime1, Prime2), !,
276     each_abs_sort(Prime2, AbsInt, TempPrime),
277     current_fact('$depend_list'(Id1, SgKey, IdList)),
278     call_to_success_approx_variant(SgKey, Subg, Call, Proj, Proj1, Sg, Sv, AbsInt, F, N, Fs,
279                                   Id, Id1, Ref, IdList, Prime1, TempPrime, List, Prime).
280 init_fixpoint1(SgKey, _, Proj, Sg, _Sv, AbsInt, F, N, _Fs0, Id, List, Prime):-
281     current_pp_flag(variants, on),
282     current_fact(fixpoint(SgKey, Subg, Proj1, Prime1, Id1, _Fs), _Ref),
283     identical_proj_1(AbsInt, Sg, Proj, Subg, Proj1, Prime1, Prime2), !,
284     (
285       current_fact(fixpoint_variant(Id1, Id, SgKey, Sgv, Projv, Fsv), Refv),
286       identical_proj(AbsInt, Sg, Proj, Sgv, Projv) ->
287       patch_parents(Refv, fixpoint_variant(Id1, Id, SgKey, Sgv, Projv, Ps), F, N, Ps, Fsv)
288     ;

```

```

289         (
290             current_pp_flag(reuse_fixp_id,on) ->
291             fixpoint_id_reuse_prev(SgKey,AbsInt,Sg,Proj,Id)
292         );
293         fixpoint_id(Id)
294     ),
295     asserta_fact(fixpoint_variant(Id1,Id,SgKey,Sg,Proj,[(F,N)]))
296 ),
297     each_abs_sort(Prime2,AbsInt,Prime),
298     current_fact(ch_id(Id1,Num)),
299     List = [Id1/Num].
300 init_fixpoint1(SgKey,Call,Proj,Sg,Sv,AbsInt,F,N,Fs,Id,List,Prime):-
301     init_fixpoint2(SgKey,Call,Proj,Sg,Sv,AbsInt,F,N,Fs,Id,List,Prime).
302
303 init_fixpoint2(SgKey,Call,Proj,Sg,Sv,AbsInt,F,N,Fs,Id,List,Prime):-
304     ( current_pp_flag(reuse_fixp_id,on) ->
305         fixpoint_id_reuse_prev(SgKey,AbsInt,Sg,Proj,Id)
306     );
307     fixpoint_id(Id)
308 ),
309     asserta_fact(ch_id(Id,1)),
310     proj_to_prime_r(SgKey,Sg,Sv,Call,Proj,AbsInt,TempPrime,Id),
311     init_fixpoint_(SgKey,Call,Proj,Sg,Sv,AbsInt,F,N,Fs,Id,
312         TempPrime,List,Prime).
313
314 init_fixpoint_(SgKey,Call,Proj,Sg,Sv,AbsInt,F,N,Fs,Id,Prime0,List,Prime):-
315     normalize_asub0(AbsInt,Prime0,TempPrime),
316     asserta_fact(fixpoint(SgKey,Sg,Proj,TempPrime,Id,Fs)),
317     bagof(X, X^(trans_clause(SgKey,r,X)),Clauses),!,
318     fixpoint_compute(Clauses,SgKey,Sg,Sv,Call,Proj,
319         AbsInt,_LEntry,TempPrime,Prime1,Id,TempList),
320     each_apply_trusted(Proj,SgKey,Sg,Sv,AbsInt,Prime1,Prime),
321     current_fact(fixpoint(SgKey,Sg,_,_,Id,Fs2),Ref),
322     erase(Ref),
323     ( current_fact('$depend_list'(Id,SgKey,_),RefDep) ->
324         erase(RefDep)
325     ); true
326 ),
327     update_if_member_idlist(TempList,Id,AddList),
328     ( member((F,N),Fs2) -> NewFs = Fs2 ; NewFs = [(F,N)|Fs2] ),
329     decide_approx(AddList,Id,NewFs,AbsInt,SgKey,Sg,Proj,Prime),
330     List = AddList.
331
332 widen_call(AbsInt,SgKey,Sg,F1,Id0,Proj1,Proj):-
333     ( current_pp_flag(widencall,off) -> fail ; true ),
334     widen_call0(AbsInt,SgKey,Sg,F1,Id0,[Id0],Proj1,Proj), !.
335
336 widen_call0(AbsInt,SgKey,Sg,F1,Id0,Ids,Proj1,Proj):-
337     widen_call1(AbsInt,SgKey,Sg,F1,Id0,Ids,Proj1,Proj).
338 widen_call0(AbsInt,SgKey,Sg,F1,Id0,Ids,Proj1,Proj):-
339     current_pp_flag(widencall,com_child),
340     widen_call2(AbsInt,SgKey,Sg,F1,Id0,Ids,Proj1,Proj).
341
342 widen_call1(AbsInt,SgKey,Sg,F1,Id0,Ids,Proj1,Proj):-
343     current_fact(fixpoint(SgKey0,Sg0,Proj0,_Prime0,Id0,Fs0)),
344     ( SgKey=SgKey0,
345         % same program point:
346         member((F1,_NewId0),Fs0)
347     -> Sg0=Sg,
348         abs_sort(AbsInt,Proj0,Proj0_s),
349         abs_sort(AbsInt,Proj1,Proj1_s),
350         widencall(AbsInt,Proj0_s,Proj1_s,Proj)
351     ; % continue with the parents:
352         member((_F1,NewId0),Fs0),
353         \+ member(NewId0,Ids),
354         widen_call1(AbsInt,SgKey,Sg,F1,NewId0,[NewId0|Ids],Proj1,Proj)

```

```

355     ).
356
357 widen_call2(AbsInt,SgKey,Sg,F1,_Id,_Ids,Proj1,Proj):-
358     current_fact(complete(SgKey,AbsInt,Sg0,Proj0,_Prime0,_Id0,Fs0)),
359     member((F1,Id0),Fs0),
360     Sg0=Sg,
361     same_fixpoint_ancestor(Id0,[Id0],AbsInt),
362     abs_sort(AbsInt,Proj0,Proj0_s),
363     abs_sort(AbsInt,Proj1,Proj1_s),
364     widencall(AbsInt,Proj0_s,Proj1_s,Proj).
365
366 same_fixpoint_ancestor(Id0,_Ids,_AbsInt):-
367     current_fact(fixpoint(_SgKey0,_Sg0,_Proj0,_Prime0,Id0,_Fs0)),!.
368 same_fixpoint_ancestor(Id0,_Ids,_AbsInt):-
369     current_fact(approx(_SgKey0,_Sg0,_Proj0,_Prime0,Id0,_Fs0)),!.
370 same_fixpoint_ancestor(Id0,Ids,AbsInt):-
371     current_fact(complete(_SgKey0,AbsInt,_Sg0,_Proj0,_Prime0,Id0,Fs0)),
372     member((_F1,Id),Fs0),
373     \+ member(Id,Ids),
374     same_fixpoint_ancestor(Id,[Id|Ids],AbsInt).
375
376 fixpoint_variants_update(Id,AbsInt,Sg,Prime):-
377     current_fact(fixpoint_variant(Id,Idv,SgKey,Sgv,Projv,Fs),Ref),!,
378     erase(Ref),
379     varset(Sg,Hv),
380     varset(Sgv,Hvv),
381     each_exit_to_prime(Prime,AbsInt,Sgv,Hv,Sg,Hvv,(no,Projv),Prime2),
382     asserta_fact(complete(SgKey,AbsInt,Sgv,Projv,Prime2,Idv,Fs)),
383     fixpoint_variants_update(Id,AbsInt,Sg,Prime).
384 fixpoint_variants_update(_,_,_,_).
385
386 approx_variants_update(Id,AbsInt,Sg,Prime):-
387     current_fact(fixpoint_variant(Id,Idv,SgKey,Sgv,Projv,Fs),Ref),!,
388     erase(Ref),
389     varset(Sg,Hv),
390     varset(Sgv,Hvv),
391     each_exit_to_prime(Prime,AbsInt,Sgv,Hv,Sg,Hvv,(no,Projv),Prime2),
392     asserta_fact(approx_variant(Id,Idv,SgKey,Sgv,Projv,Prime2,Fs)),
393     approx_variants_update(Id,AbsInt,Sg,Prime).
394 approx_variants_update(_,_,_,_).
395
396 decide_approx([],Id,Fs,AbsInt,SgKey,Sg,Proj,Prime):-!,
397     current_fact(ch_id(Id,_),Ref3),
398     erase(Ref3),
399     % Not needed for correctness: only book-keeping
400     % update_depend_list_approx(Id,AbsInt),
401     asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Id,Fs)),
402     (
403         current_pp_flag(variants,on) ->
404         each_abs_sort(Prime,AbsInt,Prime_s),
405         fixpoint_variants_update(Id,AbsInt,Sg,Prime_s)
406     ;
407         true
408     ).
409 decide_approx(AddList,Id,Fs,_AbsInt,SgKey,Sg,Proj,Prime):-
410     asserta_fact('$depend_list'(Id,SgKey,AddList)),
411     asserta_fact(approx(SgKey,Sg,Proj,Prime,Id,Fs,_)),
412     (
413         current_pp_flag(variants,on) ->
414         each_abs_sort(Prime,AbsInt,Prime_s),
415         approx_variants_update(Id,AbsInt,Sg,Prime_s)
416     ;
417         true
418     ).
419
420 not_modified([]).

```

```

421 not_modified([Id/N|List]):-
422     current_fact(ch_id(Id,N)), !,
423     not_modified(List).
424
425 proj_to_prime_nr(SgKey,Sg,Sv,Call,Proj,AbsInt,_CId,LPrime,Id) :-
426     bagof(X, X^(trans_clause(SgKey,nr,X)),Clauses), !,
427     proj_to_prime(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LPrime1,Id),
428     compute_clauses_lub(AbsInt,Proj,LPrime1,LPrime).
429 proj_to_prime_nr(SgKey,Sg,Sv,_Call,Proj,AbsInt,CId,LPrime,_Id) :-
430     apply_trusted0(Proj,SgKey,Sg,Sv,AbsInt,CId,Prime), !,
431     singleton(Prime,LPrime).
432 proj_to_prime_nr(_SgKey,Sg,Sv,Call,_Proj,AbsInt,_CId,LSucc,_Id) :-
433     % In Java programs, mode and type information is known for any method.
434     % Therefore, in case of a method with unavailable code we can still
435     % infer useful information.
436     ( current_pp_flag(prog_lang,java) ->
437       unknown_call(AbsInt,Sg,Sv,Call,Succ),
438       singleton(Succ,LSucc)
439     ;
440     fail
441     ).
442 proj_to_prime_nr(SgKey,_Sg,_Sv,_Call,_Proj,_AbsInt,CId,Bot,_Id) :-
443     bottom(Bot),
444     inexistent(SgKey,CId).
445
446 proj_to_prime_r(SgKey,Sg,Sv,Call,Proj,AbsInt,Prime,Id) :-
447     bagof(X, X^(trans_clause(SgKey,nr,X)),Clauses), !,
448     proj_to_prime(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,Prime,Id).
449 proj_to_prime_r(_SgKey,_Sg,_Sv,_Call,_Proj,_AbsInt,Bot,_Id):-
450     bottom(Bot).
451
452 proj_to_prime(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,Prime,Id) :-
453     proj_to_prime_loop(Clauses,Sg,Sv,Call,Proj,AbsInt,ListPrime0,Id),
454     reduce_equivalent(ListPrime0,AbsInt,ListPrime1),
455     each_apply_trusted(Proj,SgKey,Sg,Sv,AbsInt,ListPrime1,Prime).
456
457 proj_to_prime_loop([],_,_,_,_,[],_).
458 proj_to_prime_loop([Clause|Rest],Sg,Sv,Call,Proj,AbsInt,Primes,Id):-
459     do_nr_cl(Clause,Sg,Sv,Call,Proj,AbsInt,Primes,TailPrimes,Id),!,
460     proj_to_prime_loop(Rest,Sg,Sv,Call,Proj,AbsInt,TailPrimes,Id).
461
462 do_nr_cl(Clause,Sg,Sv,Call,Proj,AbsInt,Primes,TailPrimes,Id):-
463     Clause = clause(Head,Vars_u,K,Body),
464     clause_applies(Head,Sg), !,
465     varset(Head,Hv),
466     sort(Vars_u,Vars),
467     ord_subtract(Vars,Hv,Fv),
468     process_body(Body,K,AbsInt,Sg,Hv,Fv,Vars_u,Head,Sv,Call,
469                 Proj,LPrime,Id),
470     append_(LPrime,TailPrimes,Primes).
471 do_nr_cl(_Clause,_Sg,_Sv,_Call,_Proj,_AbsInt,Primes,Primes,_Id).
472
473 append_([Prime],TailPrimes,Primes):- !, Primes=[Prime|TailPrimes].
474 append_(LPrime,TailPrimes,Primes):- append(LPrime,TailPrimes,Primes).
475
476 process_body(Body,K,AbsInt,Sg,Hv,Fv,_Head,Sv,Call,Proj,LPrime,Id):-
477     Body = g(_,[],'$built'(_,true,_),'true/0',true), !,
478     Help=(Sv,Sg,Hv,Fv,AbsInt),
479     singleton(Prime,LPrime),
480     call_to_success_fact(AbsInt,Sg,Hv,Head,K,Sv,Call,Proj,Prime,_Succ),
481     ( current_pp_flag(fact_info,on) ->
482       call_to_entry(AbsInt,Sv,Sg,Hv,Head,K,[],Prime,Exit,_),
483       decide_memo(AbsInt,K,Id,no,Hv,[Exit])
484     ;
485     true
486     ).

```

```

487 process_body(Body,K,AbsInt,Sg,Hv,Fv,Vars_u,Head,Sv,_,Proj,Prime,Id):-
488   call_to_entry(AbsInt,Sv,Sg,Hv,Head,K,Fv,Proj,Entry,ExtraInfo),
489   singleton(Entry,LEntry),
490   entry_to_exit(Body,K,LEntry,Exit,[],_,Vars_u,AbsInt,Id),
491   each_exit_to_prime(Exit,AbsInt,Sg,Hv,Head,Sv,ExtraInfo,Prime).
492
493 fixpoint_compute(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
494   Prime0,Prime,Id,List) :-
495   fixpoint_compute_(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
496   Prime0,Prime1,Id,List),
497   compute_clauses_lub(AbsInt,Proj,Prime1,Prime).
498
499 fixpoint_compute_(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
500   TempPrime,Prime,Id,List) :-
501   compute(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
502   TempPrime,Prime1,Id,[],NewList,Flag),
503   fixpoint(NewList,Flag,Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
504   Prime1,Prime,Id,List), !.
505
506 fixpoint([],_,_,_,_,_,_,_,_,Prime1,Prime,_,List):- !,
507   Prime = Prime1,
508   List = [].
509 fixpoint(NewList,Flag,_,_,_,_,_,_,_,Prime1,Prime,_,List):-
510   var(Flag),!,
511   Prime = Prime1,
512   List = NewList.
513 fixpoint(,_,_ ,Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,Prime1,Prime,Id,List):-
514   fixpoint_compute_(Clauses,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,
515   Prime1,Prime,Id,List).
516
517 % some domains need normalization to perform the widening:
518 normalize_asub0(AbsInt,Prime0,Prime):-
519   current_pp_flag(widen,on), !,
520   normalize_asub(AbsInt,Prime0,Prime).
521 normalize_asub0(_AbsInt,Prime,Prime).
522
523 compute([],_,_,_,_,_,_,_,_,Prime,Prime,_,List,List,_) .
524 compute([Clause|Rest],SgKey,Sg,Sv,Call,Proj,AbsInt,[EntryInf|LEntryInf],
525   TempPrime,Prime,Id,List,NewList,Flag) :-
526   do_r_cl(Clause,SgKey,Sg,Sv,Proj,AbsInt,EntryInf,Id,List,IntList,
527   TempPrime,NewPrime,Flag),
528   compute(Rest,SgKey,Sg,Sv,Call,Proj,AbsInt,LEntryInf,NewPrime,Prime,
529   Id,IntList,NewList,Flag).
530
531 do_r_cl(Clause,SgKey,Sg,Sv,Proj,AbsInt,EntryInf,Id,OldL,List,TempPrime,
532   NewPrime,Flag):-
533   Clause=clause(Head,Vars_u,K,Body),
534   clause_applies(Head,Sg), !,
535   erase_previous_memo_tables_and_parents(Body,AbsInt,K,Id),
536   varset(Head,Hv),
537   reuse_entry(EntryInf,Vars_u,AbsInt,Sv,Sg,Hv,Head,K,Proj,Entry,ExtraInfo),
538   singleton(Entry,LEntry),
539   entry_to_exit(Body,K,LEntry,Exit,OldL,List,Vars_u,AbsInt,Id),
540   each_exit_to_prime(Exit,AbsInt,Sg,Hv,Head,Sv,ExtraInfo,Prime1),
541   widen_succ(AbsInt,TempPrime,Prime1,NewPrime),
542   decide_flag(AbsInt,TempPrime,NewPrime,SgKey,Sg,Id,Proj,Flag).
543
544 do_r_cl(,_ ,_,_,_,_,_,_,_,_,List,List,Prime,Prime,_) .
545
546 widen_succ_off(AbsInt,Prime0,Prime1,LPrime) :-
547   current_pp_flag(multi_success,on), !,
548   reduce_equivalent([Prime0,Prime1],AbsInt,LPrime).
549 widen_succ_off(AbsInt,Prime0,Prime1,Prime):-
550   singleton(P0,Prime0),
551   singleton(P1,Prime1),
552   singleton(P,Prime),

```

```

553     compute_lub(AbsInt, [P0, P1], P).
554
555 reuse_entry(EntryInf, Vars_u, AbsInt, Sv, Sg, Hv, Head, K, Proj, Entry, ExtraInfo) :-
556     var(EntryInf), !,
557     sort(Vars_u, Vars),
558     ord_subtract(Vars, Hv, Fv),
559     call_to_entry(AbsInt, Sv, Sg, Hv, Head, K, Fv, Proj, Entry, ExtraInfo),
560     EntryInf = (Entry, ExtraInfo).
561 reuse_entry(EntryInf, _Vars_u, _AbsInt, _Sv, _Sg, _Hv, _Head, _K, _Proj, Entry, ExtraInfo) :-
562     EntryInf = (Entry, ExtraInfo).
563
564 decide_flag(AbsInt, TempPrime, NewPrime, _SgKey, _Sg, _Id, _Proj, _Flag) :-
565     abs_subset_(NewPrime, AbsInt, TempPrime), !.
566 decide_flag(_AbsInt, TempPrime, NewPrime, SgKey, Sg, Id, Proj, Flag) :-
567     Flag = notend,
568     merge_(NewPrime, TempPrime, LPrime),
569     current_fact(fixpoint(SgKey, Sg, _, _, Id, Fs), Ref),
570     erase(Ref),
571     asserta_fact(fixpoint(SgKey, Sg, Proj, LPrime, Id, Fs)),
572     current_fact(ch_id(Id, Num), Ref3),
573     erase(Ref3),
574     Num1 is Num+1,
575     asserta_fact(ch_id(Id, Num1)).
576
577 merge_( [NewPrime], _TempPrime, LPrime) :- !, LPrime = [NewPrime].
578 merge_(NewPrime, TempPrime, LPrime) :-
579     merge(NewPrime, TempPrime, LPrime).
580
581 %------%
582 % entry_to_exit(+,+,+,-,+,-,+ ,+,+) %
583 %------%
584
585 entry_to_exit((Sg, Rest), K, Call, Exit, OldList, NewList, Vars_u, AbsInt, NewN) :- !,
586     body_succ(Call, Sg, Succ, OldList, IntList, Vars_u, AbsInt, K, NewN, _),
587     entry_to_exit(Rest, K, Succ, Exit, IntList, NewList, Vars_u, AbsInt, NewN).
588 entry_to_exit(true, _, Call, Call, List, List, _, _, _) :- !.
589 entry_to_exit(Sg, Key, Call, Exit, OldList, NewList, Vars_u, AbsInt, NewN) :-
590     body_succ(Call, Sg, Exit, OldList, NewList, Vars_u, AbsInt, Key, NewN, _),
591     decide_memo(AbsInt, Key, NewN, no, Vars_u, Exit), !.
592
593 body_succ(Call, Atom, Succ, List, List, HvFv_u, AbsInt, _ClId, ParentId, no) :-
594     bottom(Call), !,
595     Succ = Call,
596     Atom = g(Key, _Av, _I, _SgKey, _Sg),
597     asserta_fact(memo_table(Key, AbsInt, ParentId, no, HvFv_u, Succ)).
598 body_succ(Call, Atom, Succ, List, NewList, HvFv_u, AbsInt, ClId, ParentId, Id) :-
599     Atom = g(Key, Sv, Info, SgKey, Sg),
600     body_succ(Info, SgKey, Sg, Sv, HvFv_u, Call, Succ, List, NewList, AbsInt,
601             ClId, Key, ParentId, Id),
602     decide_memo(AbsInt, Key, ParentId, Id, HvFv_u, Call).
603
604 body_succ(Info, SgKey, Sg, Sv, HFv, Call, Succ, L, NewL, AbsInt, ClId, Key, PId, Id) :-
605     Info = [_|_], !,
606     split_combined_domain(AbsInt, Call, Calls, Domains),
607     map_body_succ(Info, SgKey, Sg, Sv, HFv, Calls, Succs, L, NewL, Domains,
608                 ClId, Key, PId, Id),
609     split_combined_domain(AbsInt, Succ, Succs, Domains).
610 body_succ(Info, SgKey, Sg, Sv, HFv, Call, Succ, L, NewL, AbsInt, ClId, Key, PId, Id) :-
611     body_succ0(Info, SgKey, Sg, Sv, HFv, Call, Succ, L, NewL, AbsInt,
612              ClId, Key, PId, Id).
613
614 map_body_succ([], _SgKey, _Sg, _Sv, _HFv, [], [], L, L, [], _ClId, _Key, _PId, no).
615 map_body_succ([I|Info], SgKey, Sg, Sv, HFv, [Call|Calls], [Succ|Succs], L, NewL,
616             [AbsInt|Domains], ClId, Key, PId, Id) :-
617     body_succ0(I, SgKey, Sg, Sv, HFv, Call, Succ, L, NewL, AbsInt,
618             ClId, Key, PId, Id), !,

```

```

619     map_body_succ(Info,SgKey,Sg,Sv,HFv,Calls,Succs,L,NewL,Domains,
620                  ClId,Key,PIId,Id) .
621
622 body_succ0('$var',SgKey,Sg,_Sv_u,HvFv_u,Calls,Succs,List0,List,AbsInt,
623            ClId,F,_N,Id):-
624     !,
625     ( Calls=[Call],
626       concrete(AbsInt,Sg,Call,Concretes),
627       concretes_to_body(Concretes,SgKey,AbsInt,B)
628     -> fixpoint_id(Id),
629       meta_call(B,HvFv_u,Calls,[],Succs,List0,List,AbsInt,ClId,Id,Ids),
630       assertz_fact(memo_call(F,Id,AbsInt,Concretes,Ids))
631     ; Id=no,
632       List=List0,
633       variable(F,ClId),
634       each_unknown_call(Calls,AbsInt,[Sg],Succs) % Sg is a variable
635     ).
636 body_succ0('$meta'(T,B,_),SgKey,Sg,Sv_u,HvFv_u,Call,Succ,List0,List,AbsInt,
637            ClId,F,N,Id):-
638     !,
639     ( current_pp_flag(reuse_fixp_id,on) ->
640       ( Call=[C]
641         -> sort(Sv_u,Sv),
642           project(AbsInt,Sg,Sv,HvFv_u,C,Proj),
643           fixpoint_id_reuse_prev(SgKey,AbsInt,Sg,Proj,Id)
644         ; true
645       )
646     ;
647       fixpoint_id(Id)
648     ),
649     meta_call(B,HvFv_u,Call,[],Exits,List0,List,AbsInt,ClId,Id,_Ids),
650     ( body_succ_meta(T,AbsInt,Sv_u,HvFv_u,Call,Exits,Succ) ->
651       ( Call=[C] ->
652         sort(Sv_u,Sv),
653         project(AbsInt,Sg,Sv,HvFv_u,C,Proj),
654         each_project(Exits,AbsInt,Sg,Sv,HvFv_u,Prime),
655         asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Id,[(F,N)]))
656       ; true
657     )
658     ; % for the trusts, if any:
659       varset(Sg,Sv_r), % Sv_u contains extra vars (from meta-term)
660         % which will confuse apply_trusted
661       body_succ0(nr,SgKey,Sg,Sv_r,HvFv_u,Call,Succ,[],_List,AbsInt,
662                ClId,F,N,Id0),
663       retract_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Id0,Ps)),
664       asserta_fact(complete(SgKey,AbsInt,Sg,Proj,Prime,Id,Ps))
665     ).
666 body_succ0('$built'(T,Tg,Vs),SgKey,Sg,Sv_u,HvFv_u,Call,Succ,List0,List,AbsInt,
667            _ClId,F,N,Id):-
668     !,
669     Id=no,
670     List=List0,
671     sort(Sv_u,Sv),
672     each_body_succ_builtin(Call,AbsInt,T,Tg,Vs,SgKey,Sg,Sv,HvFv_u,F,N,Succ) .
673 body_succ0(RFlag,SgKey,Sg,Sv_u,HvFv_u,Call,Succ,List0,List,AbsInt,
674            ClId,F,N,Id):-
675     sort(Sv_u,Sv),
676     each_call_to_success(Call,RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,ClId,
677                          Succ,List0,List,F,N,Id) .
678
679 each_call_to_success([Call],RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,ClId,Succ,L0,L,
680                     F,N,Id):-
681     !,
682     project(AbsInt,Sg,Sv,HvFv_u,Call,Proj),
683     call_to_success(RFlag,SgKey,Call,Proj,Sg,Sv,AbsInt,ClId,Succ,L1,F,N,Id),
684

```

```

685     merge(L1,L0,L).
686 each_call_to_success(LCall,RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,C1Id,LSucc,L0,L,
687     F,N,Id):-
688     each_call_to_success0(LCall,RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,C1Id,
689     LSucc,L0,L,F,N,Id).
690
691 each_call_to_success0([],_Flag,_SgK,_Sg,_Sv,_HvFv,_AbsInt,_,[],L,L,_F,_N,_NN).
692 each_call_to_success0([Call|LCall],RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,C1Id,
693     LSucc,L0,L,F,N,NewN):-
694     project(AbsInt,Sg,Sv,HvFv_u,Call,Proj),
695     call_to_success(RFlag,SgKey,Call,Proj,Sg,Sv,AbsInt,C1Id,LSucc0,L1,F,N,_),
696     merge(L0,L1,L2),
697     append(LSucc0,LSucc1,LSucc),
698     each_call_to_success0(LCall,RFlag,SgKey,Sg,Sv,HvFv_u,AbsInt,C1Id,
699     LSucc1,L2,L,F,N,NewN).
700
701 meta_call([],_HvFv_u,Call,[],Call,List,List,_AbsInt,_C1Id,_Id,[]).
702 meta_call([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,C1Id,Id,Ids):-
703     meta_call_([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,C1Id,Id,Ids).
704
705 meta_call_([Body|Bodies],HvFv_u,Call,Succ0,Succ,L0,List,AbsInt,C1Id,Id,Ids):-
706     meta_call_body(Body,C1Id,Call,Succ1,L0,L1,HvFv_u,AbsInt,Id,Ids0),
707     widen_succ(AbsInt,Succ0,Succ1,Succ2),
708     append(Succ0,Succ1,Succ2),
709     append(Ids0,Ids1,Ids),
710     meta_call_(Bodies,HvFv_u,Call,Succ2,Succ,L1,List,AbsInt,C1Id,Id,Ids1).
711 meta_call_([],_HvFv_u,_Call,Succ,Succ,List,List,_AbsInt,_C1Id,_Id,[]).
712
713 meta_call_body((Sg,Rest),K,Call,Exit,OldList,NewList,Vars_u,AbsInt,PId,CIDs):-
714     !,
715     CIDs=[Id|Ids],
716     body_succ(Call,Sg,Succ,OldList,IntList,Vars_u,AbsInt,K,PId,Id),
717     meta_call_body(Rest,K,Succ,Exit,IntList,NewList,Vars_u,AbsInt,PId,Ids).
718 meta_call_body(true,_,Call,Call,List,List,_,_,_,[no]):-!.
719 meta_call_body(Sg,Key,Call,Exit,OldList,NewList,Vars_u,AbsInt,PId,[Id]):-
720     body_succ(Call,Sg,Exit,OldList,NewList,Vars_u,AbsInt,Key,PId,Id).
721
722 concretes_to_body([],_SgKey,_AbsInt,[]).
723 concretes_to_body([Sg|Sgs],SgKey,AbsInt,[B|Bs]):-
724     body_info0(Sg:SgKey,[],AbsInt,B),
725     concretes_to_body(Sgs,SgKey,AbsInt,Bs).
726
727 %-----%
728 % query(+,+,+,+,+,+,+) %
729 %-----%
730
731 :- doc(query(AbsInt,QKey,Query,Qv,RFlag,N,Call,Succ),
732     "The success pattern of @var{Query} with @var{Call} is
733     @var{Succ} in the analysis domain @var{AbsInt}. The predicate
734     called is identified by @var{QKey}, and @var{RFlag} says if it
735     is recursive or not. The goal @var{Query} has variables @var{Qv},
736     and the call pattern is uniquely identified by @var{N}.").
737
738 query(AbsInt,QKey,Query,Qv,RFlag,N,Call,Succ):-
739     project(AbsInt,Query,Qv,Qv,Call,Proj),
740     call_to_success(RFlag,QKey,Call,Proj,Query,Qv,AbsInt,0,Succ,_,N,0,Id),!,
741     approx_to_completes(AbsInt).
742
743 query(_AbsInt,_QKey,_Query,_Qv,_RFlag,_N,_Call,_Succ):-
744     % should never happen, but...
745     error_message("SOMETHING HAS FAILED!").

```