Semantic Code Browsing *

published in Theory and Practice of Logic Programming

ISABEL GARCÍA-CONTRERAS[1] JOSÉ F. MORALES[1] MANUEL V. HERMENEGILDO[1,2]

[1] *IMDEA Software Institute* (*e-mail:* {isabel.garcia, josef.morales, manuel.hermenegildo}@imdea.org)
[2] *School of Computer Science, Technical University of Madrid (UPM)* (*e-mail:* manuel.hermenegildo@upm.es)

submitted May 6, 2016; revised July 8, 2016; accepted July 22, 2016

Appendix A Example code

Sample code found with `math_graph` structure:

```
:- module(named_graphs, [complete_graph/2, cycle_graph/2], []).

:- use_module(library(lists), [append/3]).

complete_graph(N, graph(V,E)) :-
        count(N, V),
        generate_complete_edges(V, E).

generate_complete_edges(V, E) :-
        generate_complete_edges_(V, V, E).

generate_complete_edges_([], _, []).
generate_complete_edges_([V|Vs], AllV, E) :-
        generate_complete_edges_for_vertex(V, AllV, E1),
        append(E1, RestE, E),
        generate_complete_edges_(Vs, AllV, RestE).

generate_complete_edges_for_vertex(_, [], []) :- !.
generate_complete_edges_for_vertex(V, [V|Vs], E) :- !,
        generate_complete_edges_for_vertex(V, Vs, E).
generate_complete_edges_for_vertex(V, [V1|Vs], [(V, V1)|E]) :-
        generate_complete_edges_for_vertex(V, Vs, E).

cycle_graph(N, graph(V,E)) :-
        N = 2, !,
        V = [1,2],
        E = [(1,2),(2,1)].
cycle_graph(N, graph(V,E)) :-
        N > 1,
        count(N, V),
        generate_cycle_edges(V, E).

generate_cycle_edges([V1], [(V1, 1)]) :- !.
generate_cycle_edges([V1, V2|Vs], [(V1, V2)|Edges]) :-
        generate_cycle_edges([V2|Vs], Edges).

count(N, Lst) :-
        count_(1, N, Lst).
count_(I, N, []) :-
        I > N, !.
count_(I, N, [I|L]) :-
        I1 is I+1,
        count_(I1, N, L).
```

Fig. A 1: `named_graphs.pl` (Ciao library)

* This research has received funding from the EU FP7 agreement no 318337, ENTRA, Spanish MINECO TIN2012-39391 *StrongSoft* and TIN2015-67522-C3-1-R *TRACES* projects, and the Madrid M141047003 *N-GREENS* program.

Sample code found with `al_graph` structure:

```
:- module(ugraphs, [add_vertices/3], [assertions,isomodes] ).

:- use_module(library(sets), [ord_union/3]).
:- use_module(library(sort), [sort/2]).

:- pred add_vertices(+Graph1, +Vertices, -Graph2)
#  "Is true if @var{Graph2} is @var{Graph1} with @var{Vertices} added to it.".
add_vertices(Graph0, Vs0, Graph) :-
        sort(Vs0, Vs),
        Vs = Vs0,
        vertex_units(Vs, Graph1),
        graph_union(Graph0, Graph1, Graph).
% ...
```

Fig. A 2: Fragment from `ugraphs.pl` (Ciao library).

Appendix B Algorithms for predicate matching

The algorithms presented in this section are used to decide whether a predicate is proven to match a condition (that condition is checked or false) or that it cannot say anything about that property holding (check).

Algorithm 1 Matching Status of a calls condition for a predicate p

Input: $Analysis(P, D_\alpha, \mathcal{Q}_\alpha)$, $p \in exported(P)$, $C = \texttt{calls}(H, (Pre_1; ...; Pre_n))$
Output: Status of proof

if $\forall \langle H, \lambda^c, \lambda^s \rangle \in Analysis$ *s.t.* $H = p(X_1, ..., X_n), \bigvee_i \lambda^-_{TS(Pre_i,P)} \sqsupseteq \lambda^c$ **then**
    Status = **Checked**
else if $\forall \langle H, \lambda^c, \lambda^s \rangle \in Analysis$ *s.t.* $H = p(X_1, ..., X_n), \bigvee_i \lambda^-_{TS(Pre_i,P)} \sqcap \lambda^c = \bot$ **then**
    Status = **False**
else
    Status = **Check**
end if

Algorithm 2 Matching Status of a success condition for a predicate p

Input: $Analysis(P, D_\alpha, \mathcal{Q}_\alpha)$, $p \in P$, $C = \texttt{success}(H, Pre, Post)$
Output: Status of proof

if $\exists \langle H, \lambda^c, \lambda^s \rangle \in Analysis$ s.t. $H = p(X_1, ..., X_n), \lambda^c = \lambda^+_{TS(Pre,P)}$ **then**
    if $\lambda^s \sqsubseteq \lambda^-_{TS(Post,P)}$ **then**
        Status = **Checked**
    else if $\lambda^s \sqcap \lambda^+_{TS(Post,P)} = \bot$ **then**
        Status = **False**
    else
        Status = **Check**, analysis accurate enough
    end if
else if $\exists \langle H, \lambda^c, \lambda^s \rangle \in Analysis$ s.t. $H = p(X_1, ..., X_n), \lambda^c \sqsupset \lambda^+_{TS(Pre,P)}$ **then**
    if $\lambda^s \sqsubseteq \lambda^-_{TS(Post,P)}$ **then**
        Status = **Checked**
    else if $\lambda^s \sqcap \lambda^+_{TS(Post,P)} = \bot$ **then**
        Status = **False**
    else
        Status = **Check**, Refine analysis
    end if
else
    Status = **Check**, No information for that calls, Refine analysis
end if

Appendix C Additional tables

Table C 1: Analysis statistics from core/lib modules: time(ms) and memory(B) consumption.

Module name	**load time**	**regtype ana time**	**regtype global stack mem**	**shfr ana time**	**shfr global stack mem**	**total analysis time**
dict	480	20	669,312	3,712	772,472	3,732
sets	548	116	1,462,696	1,512	1,923,720	1,628
assrt_write	760	172	1,404,136	1,240	420,392	1,412
sort	544	184	877,104	992	222,288	1,176
optparse_tr	744	32	814,168	1,068	950,272	1,100
translation	516	108	3,415,632	564	471,456	672
exsteps	664	28	990,872	296	1,186,552	324
assrt_write0	724	80	1,030,808	96	271,688	176
assrt_lib_extra	724	108	1,103,072	48	314,680	156
term_list	488	72	867,120	24	160,944	96
civil_registry	508	76	554,032	16	621,504	92
assertions_props	556	44	1,385,760	40	1,722,832	84
pl2wam_tables	484	40	2,887,440	32	3,086,248	72
embedded_tr	832	24	680,736	44	792,152	68
terms	528	44	571,912	12	653,248	56
ceval1	496	48	522,152	4	582,896	52
unittest_base	516	36	630,600	16	740,344	52
ceval2	528	44	522,320	4	583,064	48
errhandle	540	28	620,024	16	747,456	44
goal_trans	484	32	582,088	12	673,952	44
llists	480	24	526,384	12	592,568	36
file_utils	564	20	641,728	12	758,024	32
foreign_compilation	584	28	541,280	4	618,072	32
qsort	484	24	483,552	8	528,312	32
srcdbg	720	4	2,540,064	28	2,570,376	32
meta_props	500	24	496,800	4	535,128	28
strings	532	16	693,304	12	809,928	28
libpaths	552	16	439,304	8	475,040	24
metatypes_tr	468	24	439,216	0	477,136	24
attr_bench	796	16	2,641,584	4	2,737,680	20
between	472	16	438,144	4	467,216	20
iso_char	496	12	599,032	8	679,024	20
length	540	20	437,240	0	456,896	20
phrase_test	512	8	556,144	8	644,392	16
optparse_rt	488	4	456,144	8	501,440	12
relationships	532	8	479,552	4	519,952	12
res_exectime_rt	632	8	480,568	4	495,480	12
resources_tr	476	8	419,248	4	444,992	12
resources_types	484	8	483,864	4	534,696	12
streams	532	8	468,608	4	518,952	12
ttyout	500	8	450,688	4	498,456	12

(continued in next page)

Table C 1: Analysis statistics from core/lib modules: time(ms) and memory(B) consumption. *(continued).*

Module name	**load time**	**regtype ana time**	**regtype global stack mem**	**shfr ana time**	**shfr global stack mem**	**total analysis time**
bundle_params	484	4	2,476,504	4	2,499,576	8
ctrlcclean	524	8	396,168	0	428,456	8
miscprops	460	4	448,888	4	480,056	8
odd	488	4	407,320	4	421,968	8
old_database	492	4	494,040	4	529,896	8
pretty_names	488	4	416,456	4	432,328	8
dict_types	512	4	413,912	0	439,584	4
fastrw	512	0	447,968	4	471,416	4
prf_ticks_rt	636	0	506,008	4	520,416	4
res_nargs_res	524	0	393,080	4	407,560	4
test1	500	4	367,368	0	382,456	4
test4	520	4	372,968	0	384,120	4
assrt_synchk	496	0	375,848	0	384,968	0
c_itf_props	480	0	414,208	0	435,624	0
compressed_bytecode	500	0	367,288	0	376,488	0
doc_flags	512	0	426,312	0	455,768	0
doc_props	520	0	366,272	0	375,344	0
regtypes_tr	484	0	415,608	0	434,712	0
res_litinfo	528	0	498,792	0	526,104	0
runtime_ops_tr	460	0	375,136	0	389,048	0
test2	488	0	367,704	0	382,864	0
unittest_examples	472	0	384,632	0	396,216	0
TOTAL (63)	34,088	1,680	47,436,912	9,924	44,316,888	11,604
AVG	541	26.7	752,967	157	703,443	184

Table C 2: Analysis dump files statistics from core/lib modules.

Module name	**dump size(B)**	**restore time(s)**
assrt_write	566,132	2,440
sort	524,490	1,772
translation	314,227	1,652
assrt_write0	142,058	1,228
assrt_lib_extra	138,689	1,132
assertions_props	142,057	1,084
sets	212,735	1,028
exsteps	257,632	916
term_list	103,583	780
errhandle	51,222	640
attr_bench	47,920	548
terms	59,107	536
phrase_test	37,034	516
file_utils	50,810	484

(continued in next page)

Table C 2: Analysis dump files statistics from core/lib modules.

Module name	**dump size(B)**	**restore time(s)**
embedded_tr	80,129	440
strings	36,279	400
optparse_tr	136,929	384
unittest_base	46,705	356
civil_registry	30,588	328
dict	106,704	308
iso_char	26,702	300
foreign_compilation	23,623	276
goal_trans	44,771	276
llists	22,500	260
ceval2	24,122	248
ceval1	21,995	232
qsort	17,867	200
pl2wam_tables	17,649	184
ttyout	9,631	164
streams	12,383	160
metatypes_tr	12,959	156
meta_props	19,571	148
libpaths	11,676	124
old_database	13,462	112
dict_types	6,807	108
relationships	7,951	108
between	11,756	100
fastrw	6,754	100
ctrlcclean	7,474	96
srcdbg	31,936	92
miscprops	6,056	88
doc_flags	4,678	84
optparse_rt	8,713	84
res_litinfo	6,662	80
resources_tr	8,358	80
bundle_params	6,528	72
c_itf_props	2,474	68
resources_types	3,864	64
length	4,503	60
test2	1,519	52
test1	1,517	48
odd	2,498	44
pretty_names	4,875	44
prf_ticks_rt	2,271	44
res_exectime_rt	2,676	44
runtime_ops_tr	3,204	40
res_nargs_res	2,646	36
compressed_bytecode	782	32
regtypes_tr	884	28
test4	218	24
unittest_examples	58	24

(continued in next page)

Table C 2: Analysis dump files statistics from core/lib modules.

Module name	**dump size(B)**	**restore time(s)**
assrt_synchk	58	20
doc_props	396	20
TOTAL (63)	3,512,057	21,596
AVG	55,747	343