Bill J. Ellis
February 2, 2005

Tweaking the SPADE Simplifier

Contents

*Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe ϵ-baked ideas, where $1 \geq \epsilon \geq 0$. (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

1 Introduction

In Hum 2 changes to the SPADE Simplifier were proposed. Here these changes are implemented and documented.

2 Getting started

2.1 Install

Install the SPARK development environment as per: `http://wallis.praxis-cs.co.uk/~spark/dev_win.html`. Install the SPARK tools, to set up dynamic link libraries (DLL) files associated with the SPARK toolset. Enter the .../*sparkdev*/*spade*/*simplifier*/*src* directory and type: `make`.

2.2 Experiment

It is confusing to have multiple versions of the SPARK tools on the same system. Thus the installed SPARK tools are overwritten with newly built versions. This ensures that at any given time only the version under development is available for execution. Note that this approach also ensures that the newly built code is executed within the full and correctly configured SPARK environment. For this purpose the following tiny script was created:

```
make
cp spadesimp.exe /cygdrive/c/Praxis/Simplifier/
cp spadesimp.psv /cygdrive/c/Praxis/Simplifier/
```

3 Inside the SPADE Simplifier

3.1 Main

The main predicate in the SPADE Simplifier is: `save_simp`, located in `buildsimp.pl`.

3.2 Output the VCs with preserved numbers (writevc.pl)

Internally the SPADE Simplifier continues to use the original numbering scheme held in the VCG file. It is only at the very final stage of VCs being written to the SIV file that the new numbers are inserted. This is the key predicate:

```
process_next_conclusion('$DONE') :-
        nhn(0),
        !,
        write_unit_conc_part.
process_next_conclusion('$DONE') :-
        nl,
        !.
process_next_conclusion(C) :-
        get_next_nhn(N),
        write_next_conclusion(N, C),
        !.
```

The predicate `process_next_conclusion(C)` is called by iterating over the original numbers through modifications to the asserted predicate `hn(N)`. Thus, at this predicate, doing `hn(N)` will unify `N` with the current number. It is necessary to continue to call `get_next_nhn(N)` counting conclusions added in the asserted predicate `nhn(M)`, so that the test `nhn(0)` remains valid. Thus the fix is:

```
process_next_conclusion('$DONE') :-
        nhn(0),
        !,
        write_unit_conc_part.
process_next_conclusion('$DONE') :-
        nl,
        !.
process_next_conclusion(C) :-
```

```
        get_next_nhn(DUMMY),
        hn(N),
        write_next_conclusion(N, C),
        !.
```

3.3 Add new switch (getdcldat.pl)

The SPADE Simplifier accepts a number of switches on the command line. The SPADE Simplifier will be configured so that the number preserving feature is selectable by setting a switch accordingly.

3.3.1 Describing the switch

The following describes the switch being implemented. It may be inserted verbatim into the SPADE Simplifier manual. The SPADE Simplifier is extended to accept a new simple qualifier switch:

> `simple_qualifier = ... | /norenum`
>
> ...
>
> The "/norenum" qualifier may be used to tell the SPADE Simplifier to preserve the conclusion numbers as seen in the .vcg file for those conclusions outputted in the .siv file. The default behaviour is to renumber the conclusions in the .siv file.

3.3.2 Switch clashes

The SPADE Simplifier is aware of the available switches and uses this to narrow the search for switches at the earliest opportunity[1]. Wanting to minimise changes, the new code will continue this policy. The full collection of existing switches are:

```
command_line = filename { qualifier_part }
qualifier_part = simple_qualifier | log_qualifier | choices_qualifier | limit_qualifier
simple_qualifier = /nolog | /rename | /nowrap | /noecho | /plain
log_qualifier = /log= filename
choices_qualifier = choices_name [ = chosen_units ]
choices_name = /nosimplification | /nostandardisation | /norule_substitution |
               /nocontradiction_hunt | /nosubstitution_elimination
chosen_units = ( range { , range } )
range = integer [ - integer ]
limit_qualifier = limit_name = integer
limit_name = /complexity_limit | /depth_limit | /inference_limit | /memory_limit
```

The new switches and their clashes are:

```
/norenum (/nore) - /nolog (/nol)
 ***                **
                   /nowrap (/now)
                    **
                   /noecho  (/now)
                    **
                   /nosimplification (/nosi)    - /nostandardisation (/nost)
                    **                             ***
                                                  /nosubstitution_elimination (/nosu)
                                                   ***
                   /nocontradiction_hunt (/noc)
                    **
                   /norule_substitution (/noru) <-CHANGE (used to be /nor)
                    ***

/renum (/renu)   - /rename (/rena) <-CHANGE (used to be /r)
 ***                ***
```

[1]This seems confusing. Why not match the entire switch and display a warning with no matches. Hard coding a narrowing of the search for such a small state space seems both unnecessary and complicated.

3.3.3 Code for new switch

```
%"/norenum" = [47, 110, 111, 114, 101, 110, 117, 109]
process_dcl_qualifier([47,110,111,114,101|REST]) :-             /* "/nore" -- should be NORENUM */
        gen_append(REST, _, "num"),
        !,
        (
           do_not_renumber_conclusions,
           !,
           scream('/norenum qualifier appears more than once on command-line')
        ;
           do_renumber_conclusions,
           !,
           scream('/norenum and /renum both appear on command-line')
        ;
           true
        ),
        !,
        assert(do_not_renumber_conclusions),
        !.

%"/renum" = [47, 114, 101, 110, 117, 109]
process_dcl_qualifier([47,114,101,110,117|REST]) :-             /* "/renu" -- should be RENUM */
        gen_append(REST, _, "m"),
        !,
        (
           do_renumber_conclusions,
           !,
           scream('/renum qualifier appears more than once on command-line')
        ;
           do_not_renumber_conclusions,
           !,
           scream('/norenum and /renum both appear on command-line')
        ;
           true
        ),
        !,
        assert(do_renumber_conclusions),
        !.
```

3.3.4 Change code for some switches

```
%"/norule_substitution" = [47, 110, 111, 114, 117, 108, 101, 95, 115, 117, 98, 115, 116, 105,
%                          116, 117, 116, 105, 111, 110]
process_dcl_qualifier([47,110,111,114,117|REST]) :- /* "/noru" -- should be NORULE_SUBSTITUTION */
        get_arg_numbers(REST, FRONT, ARG_LIST),
        gen_append(FRONT, _, "le_substitution"),
        !,
        deal_with_list_qualifier(rule_substitution, ARG_LIST),
        !.

%"/rename" = [47, 114, 101, 110, 97, 109, 101]
process_dcl_qualifier([47,114, 101, 110, 97|REST]) :-           /* "/rena" -- should be RENAME */
        gen_append(REST, _, "me"),
        !,
        (
           rename_output_file,
           !,
           scream('/rename qualifier appears more than once on command-line')
        ;
```

```
                true
        ),
        !,
        assert(rename_output_file),
        !.
```

3.4 Connect switch to renumber code (writevc.pl)

Where `do_not_renumber_conclusions` is set, do not renumber the conclusions. In all other cases do renumber the conclusions. Note that the test `\+ do_not_renumber_conclusions,` might have been excluded. However, this helps ensure a more sensible result if the predicate containing `do_not_renumber_conclusions` fails and backtracking begins.

```
process_next_conclusion('$DONE') :-
        nhn(0),
        !,
        write_unit_conc_part.

process_next_conclusion('$DONE') :-
        nl,
        !.

%Do not renumber conclusions.
process_next_conclusion(C) :-
        do_not_renumber_conclusions,
        get_next_nhn(DUMMY),
        hn(N),
        write_next_conclusion(N, C),
        !.

%Do renumber conclusions. (This is the default)
process_next_conclusion(C) :-
        \+ do_not_renumber_conclusions,
        get_next_nhn(N),
        write_next_conclusion(N, C),
        !.
```

4 Executive summary

Changes to the SPADE Simplifier are described, implementing a switch that controls whether or not the SPADE Simplifier will renumber conclusion numbers in its output. The default behaviour will be for the SPADE Simplifier to renumber conclusions, as it does at the moment.

A All SPADE Simplifier changes

A.1 aritheval.pl

A.2 attribs.pl

A.3 buildsimp.pl

A.4 cipher.pl

A.5 datecheck.pl

A.6 deduction.pl

A.7 getdcldat.pl

```
255,260c255,262
< process_dcl_qualifier([47,110,111,114|REST]) :-        /* "/nor" -- should be NORULE_SUBSTITUTION */
<         get_arg_numbers(REST, FRONT, ARG_LIST),
<         gen_append(FRONT, _, "ule_substitution"),
<         !,
<         deal_with_list_qualifier(rule_substitution, ARG_LIST),
<         !.
---
> %"/norule_substitution" = [47, 110, 111, 114, 117, 108, 101, 95, 115, 117, 98, 115, 116, 105,
> %                           116, 117, 116, 105, 111, 110]
> process_dcl_qualifier([47,110,111,114,117|REST]) :- /* "/noru" -- should be NORULE_SUBSTITUTION */
>         get_arg_numbers(REST, FRONT, ARG_LIST),
>         gen_append(FRONT, _, "le_substitution"),
>         !,
>         deal_with_list_qualifier(rule_substitution, ARG_LIST),
>         !.
280,292c282,295
< process_dcl_qualifier([47,114|REST]) :-                        /* "/r" -- should be RENAME */
<         gen_append(REST, _, "ename"),
<         !,
<         (
<            rename_output_file,
<            !,
<            scream('/rename qualifier appears more than once on command-line')
<         ;
<            true
<         ),
<         !,
<         assert(rename_output_file),
<         !.
---
> %"/rename" = [47, 114, 101, 110, 97, 109, 101]
> process_dcl_qualifier([47,114, 101, 110, 97|REST]) :-           /* "/rena" -- should be RENAME */
>         gen_append(REST, _, "me"),
>         !,
>         (
>            rename_output_file,
>            !,
>            scream('/rename qualifier appears more than once on command-line')
>         ;
>            true
>         ),
>         !,
>         assert(rename_output_file),
>         !.
339a343
>
356a361,399
>
> %"/norenum" = [47, 110, 111, 114, 101, 110, 117, 109]
> process_dcl_qualifier([47,110,111,114,101|REST]) :-               /* "/nore" -- should be NORENUM */
>         gen_append(REST, _, "num"),
>         !,
>         (
>            do_not_renumber_conclusions,
>            !,
>            scream('/norenum qualifier appears more than once on command-line')
>         ;
>            do_renumber_conclusions,
>            !,
>            scream('/norenum and /renum both appear on command-line')
>         ;
>            true
>         ),
>         !,
>         assert(do_not_renumber_conclusions),
>         !.
>
> %"/renum" = [47, 114, 101, 110, 117, 109]
> process_dcl_qualifier([47,114,101,110,117|REST]) :-               /* "/renu" -- should be RENUM */
>         gen_append(REST, _, "m"),
>         !,
>         (
>            do_renumber_conclusions,
>            !,
>            scream('/renum qualifier appears more than once on command-line')
>         ;
>            do_not_renumber_conclusions,
>            !,
```

```
>           scream('/norenum and /renum both appear on command-line')
>        ;
>           true
>        ),
>        !,
>        assert(do_renumber_conclusions),
>        !.
>
```

A.8 getlicence.pl

A.9 getlicence_vax.pl

A.10 inferenc2.pl

A.11 initvals.pl

A.12 interrupt.pl

A.13 loaddecs.pl

A.14 loadvcg.pl

A.15 make_vax.pl

A.16 makelog.pl

A.17 newded.pl

A.18 opdeclar.pl

A.19 optionsfreedemo.pl

A.20 optionsr.pl

A.21 quant.pl

A.22 records2.pl

A.23 rename.pl

A.24 simp.pl

A.25 simplify.pl

A.26 simpvc.pl

A.27 spysimp.pl

A.28 standard.pl

A.29 typecheck5.pl

A.30 utilities.pl

A.31 writevc.pl

```
345d344
<
347,349c346,349
<         nhn(0),
<         !,
<         write_unit_conc_part.
---
>         nhn(0),
>         !,
>         write_unit_conc_part.
>
351,352c351,354
<         nl,
<         !.
---
>         nl,
>         !.
>
> %Do not renumber conclusions.
354,356c356,360
<         get_next_nhn(N),
<         write_next_conclusion(N, C),
<         !.
---
>         do_not_renumber_conclusions,
>         get_next_nhn(DUMMY),
>         hn(N),
>         write_next_conclusion(N, C),
>         !.
357a362,367
> %Do renumber conclusions. (This is the default)
> process_next_conclusion(C) :-
>         \+ do_not_renumber_conclusions,
>         get_next_nhn(N),
>         write_next_conclusion(N, C),
>         !.
```