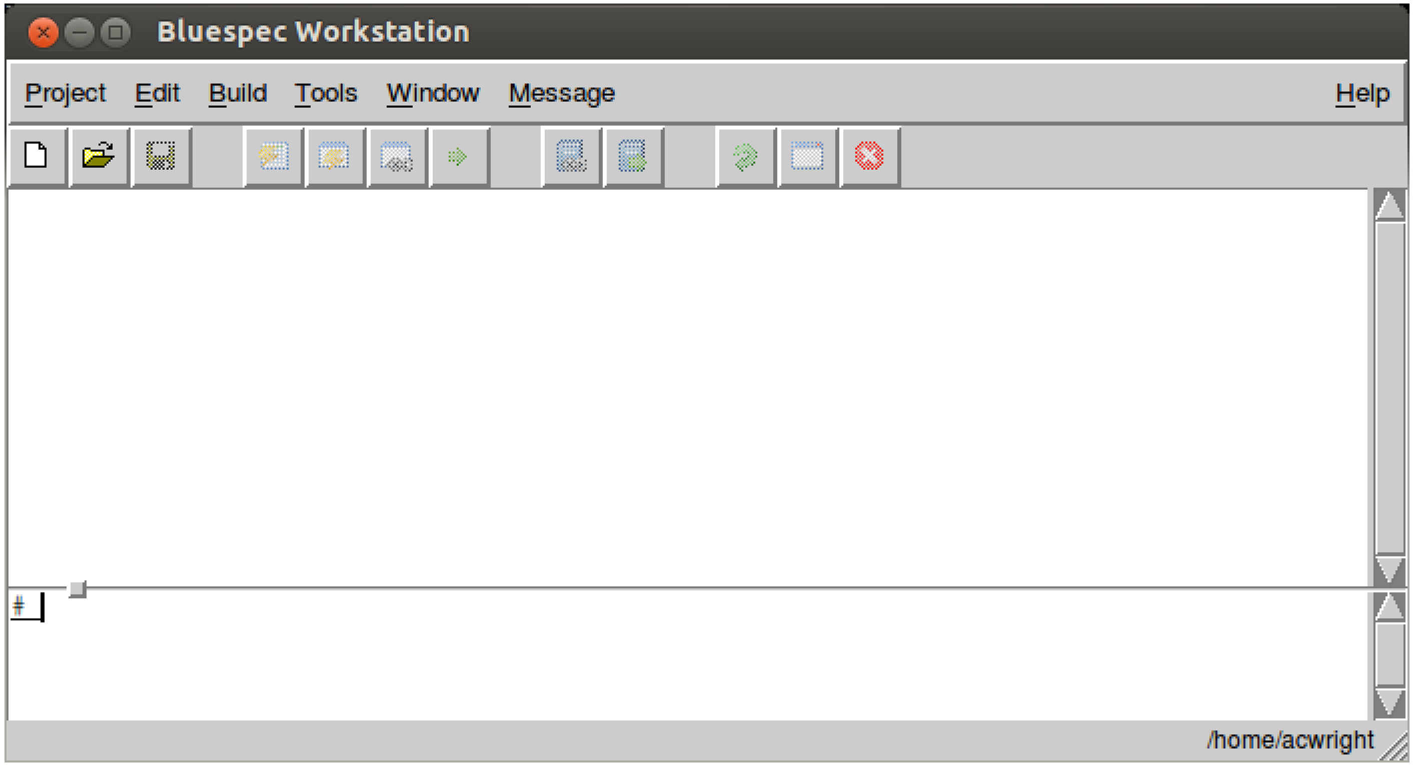
How to do scheduling analysis in the Bluespec Gui



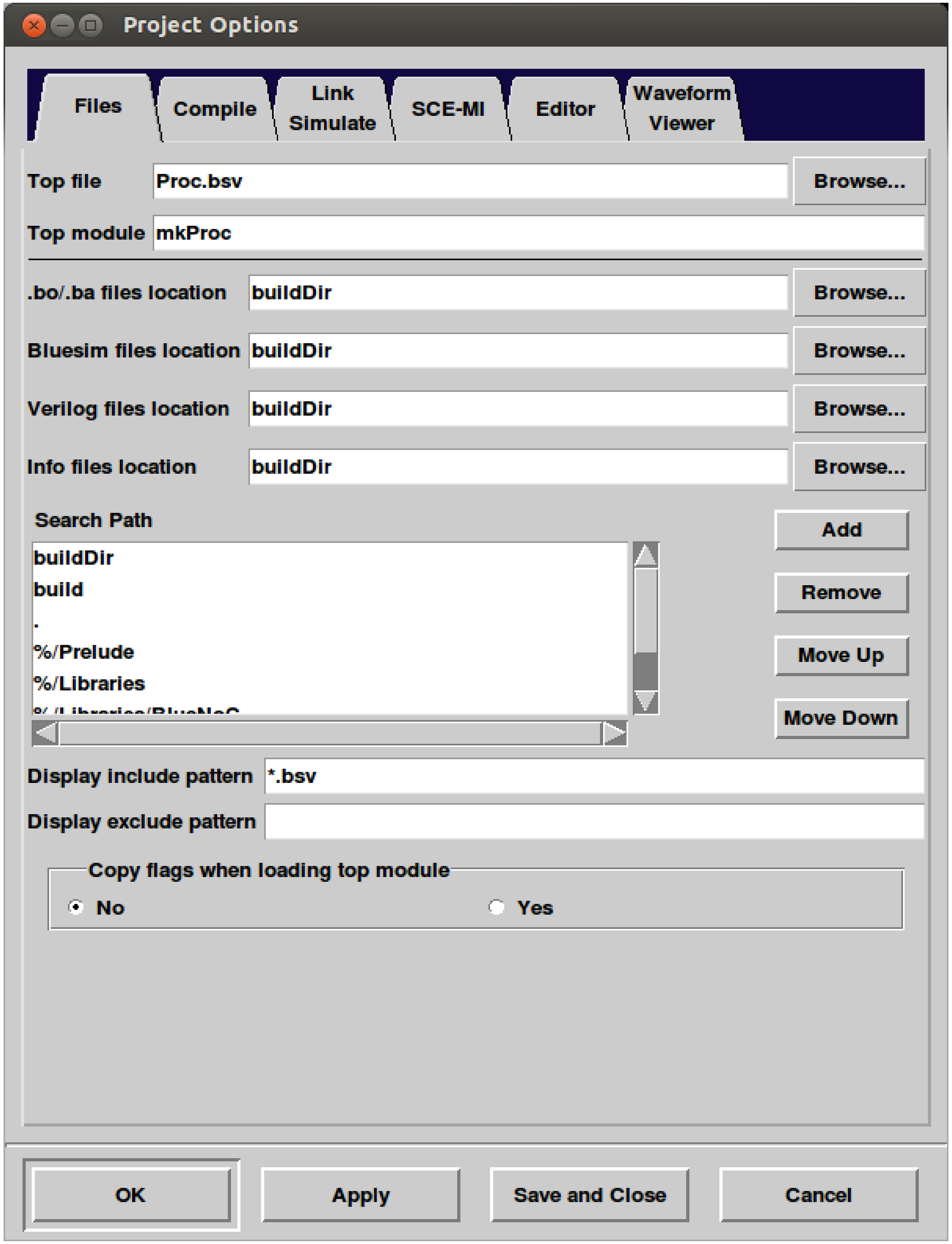
Open the Bluespec GUI in your working folder with the command “bluespec”

Create a new project

Top file is where mkProc is defined

Put buildDir as a temporary folder for bsc to output to

Add /mit/6.s195/common-lib to the Search Path


Project Options

Files | Compile | Link Simulate | SCE-MI | Editor | Waveform Viewer

Top file: Proc.bsv

Top module: mkProc

.bo/.ba files location: buildDir

Bluesim files location: buildDir

Verilog files location: buildDir

Info files location: buildDir

Search Path

```
buildDir
build
.
%/Prelude
%/Libraries
```

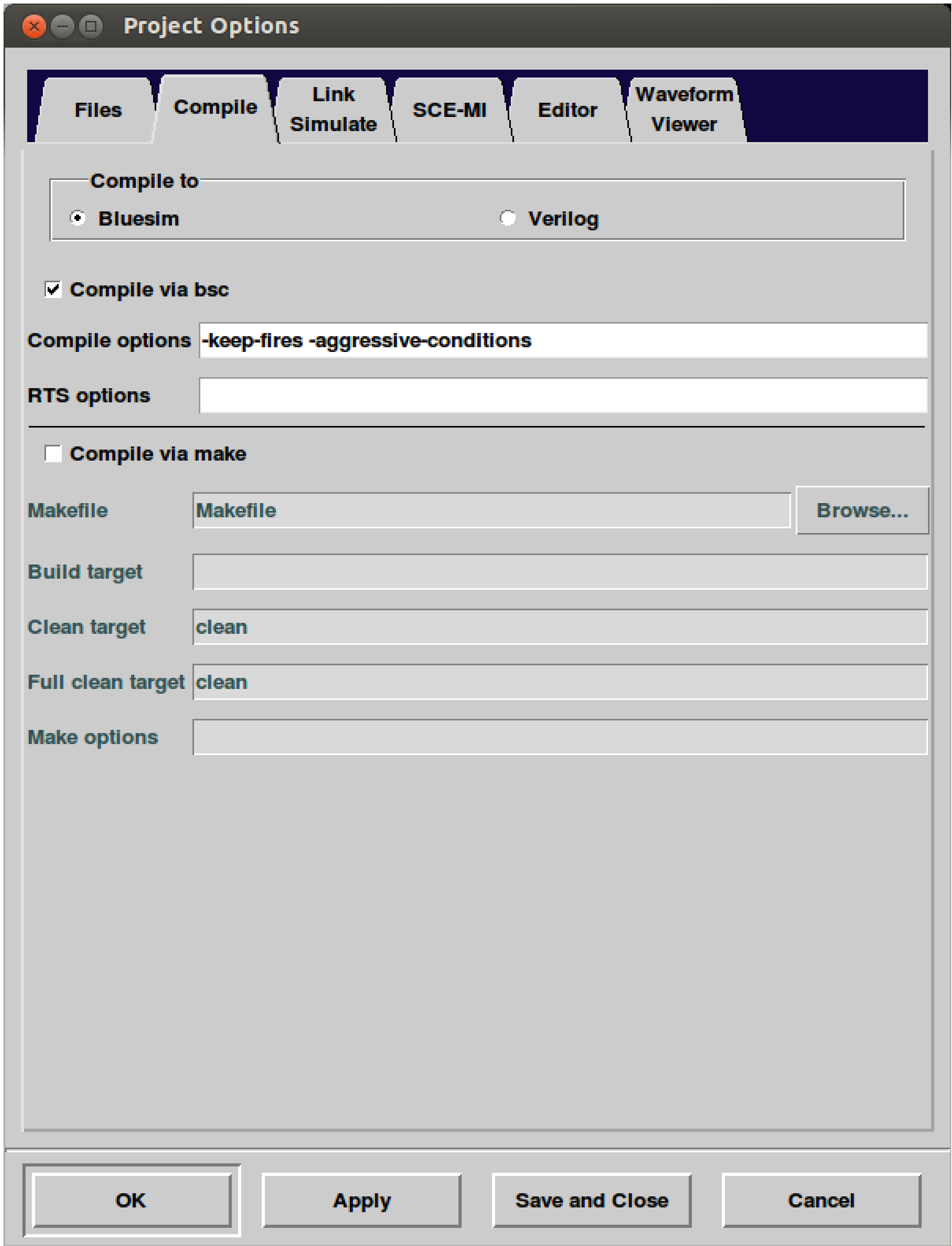
Display include pattern: *.bsv

Display exclude pattern:

Copy flags when loading top module: [✓ No] Yes

On the compile tab:

Add -aggressive-conditions flag


Project Options

Compile to
[✓ Bluesim] Verilog

[✓] Compile via bsc

Compile options: -keep-fires -aggressive-conditions

RTS options:

[ ] Compile via make

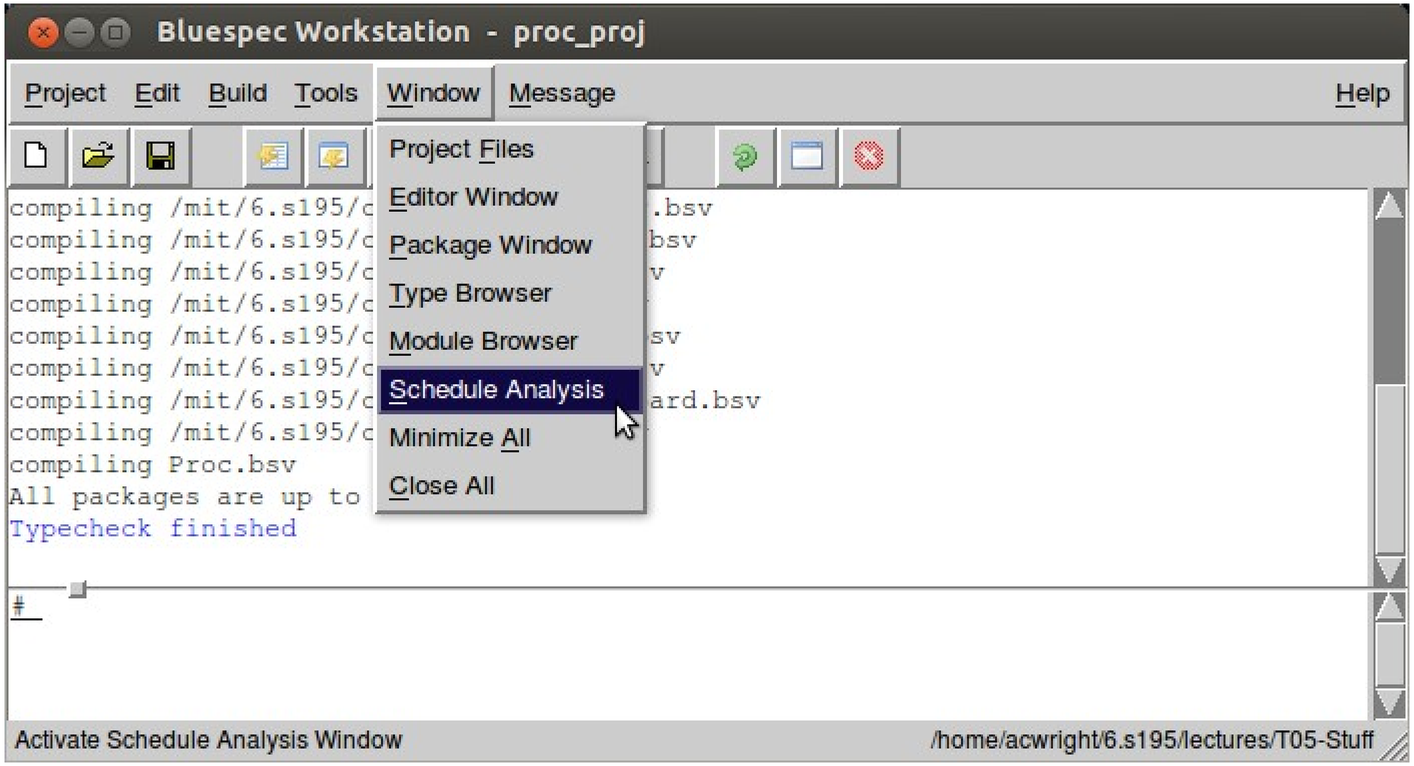
Makefile: Makefile

Build target:

Clean target: clean

Full clean target: clean

Make options:

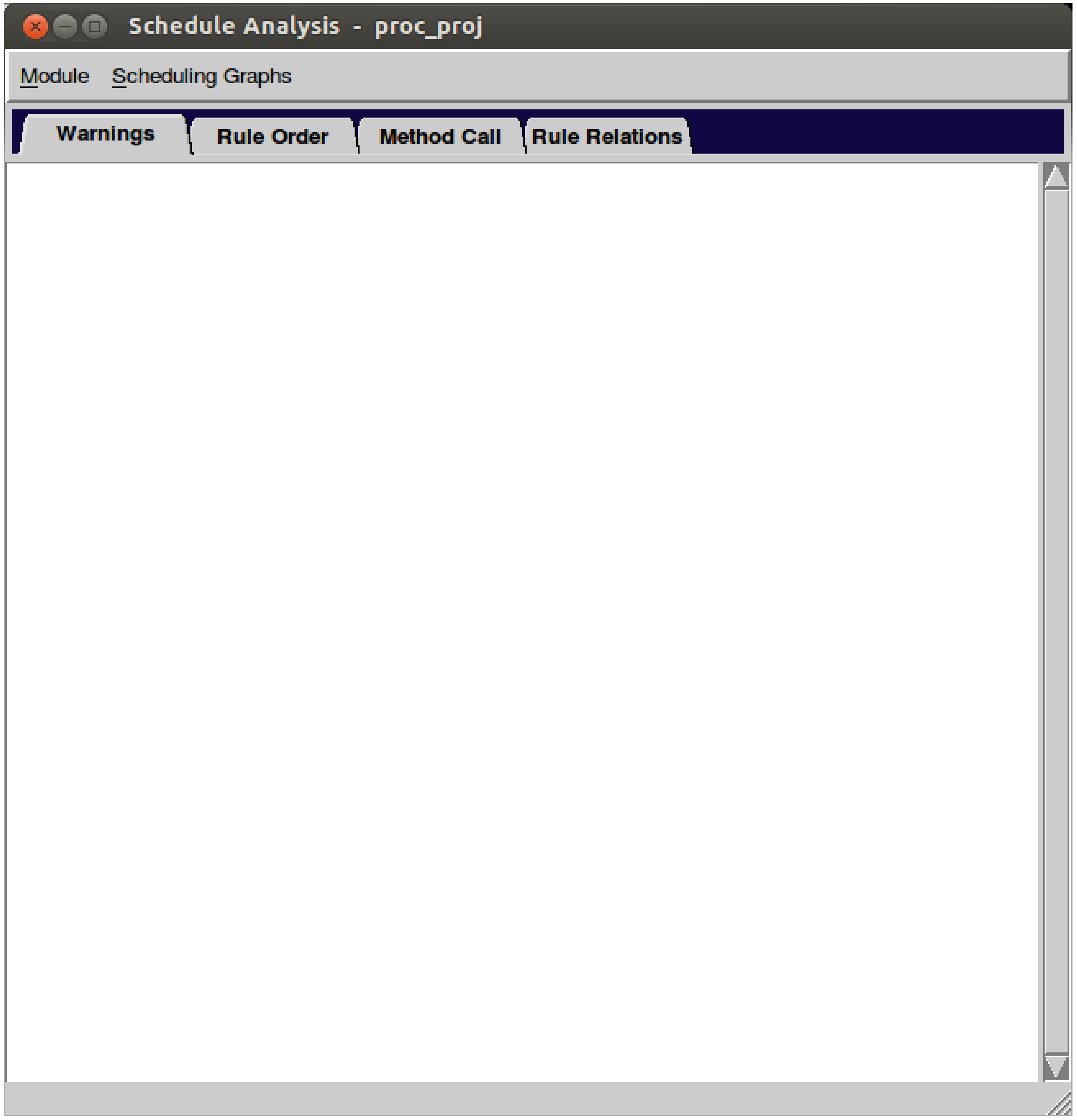


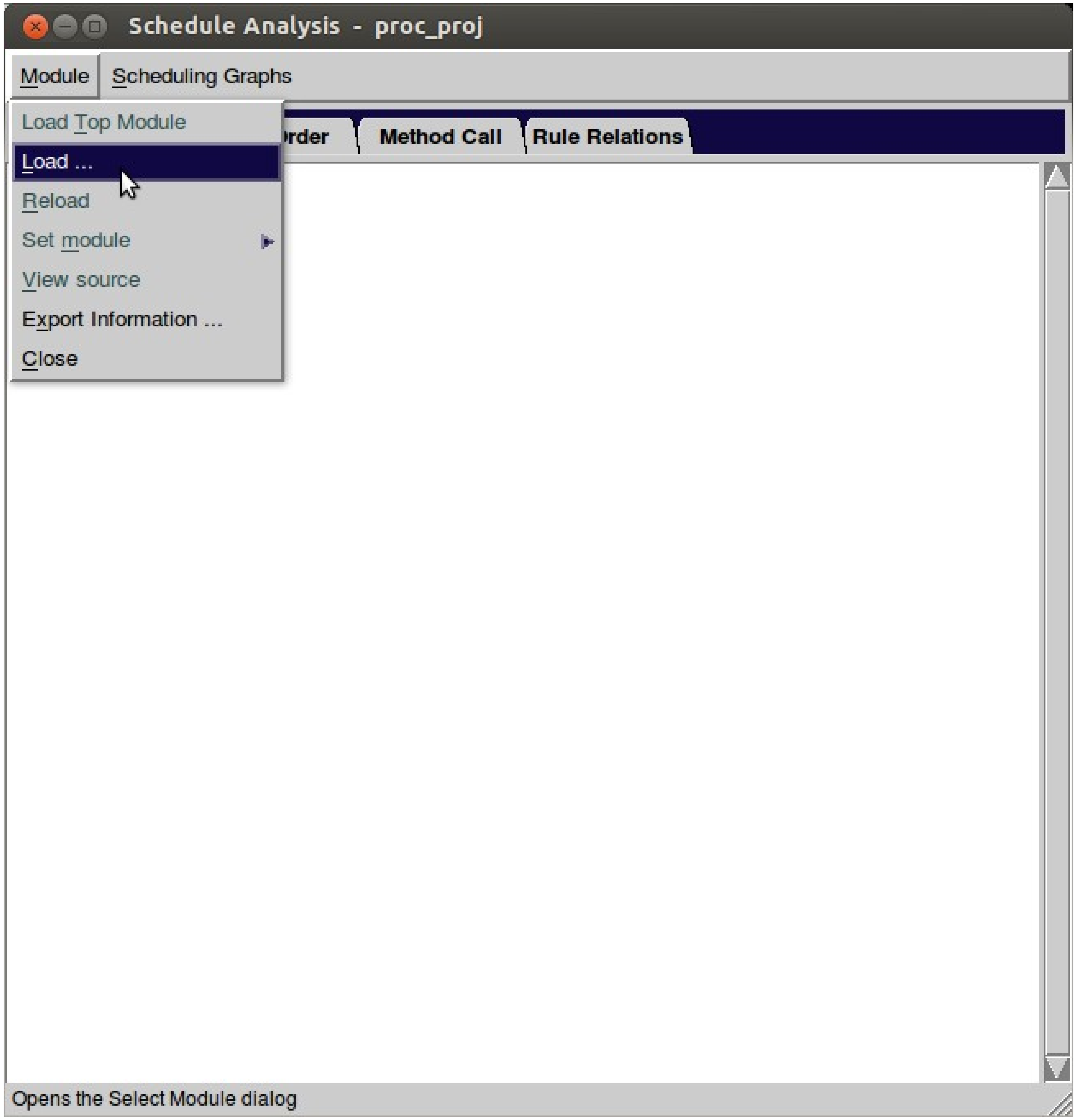
```
compiling /mit/6.s195/c                              .bsv
compiling /mit/6.s195/c                              bsv
compiling /mit/6.s195/c                              v
compiling /mit/6.s195/c
compiling /mit/6.s195/c                              sv
compiling /mit/6.s195/c                              v
compiling /mit/6.s195/c                              ard.bsv
compiling /mit/6.s195/c
compiling Proc.bsv
All packages are up to
Typecheck finished
```

```
#
```

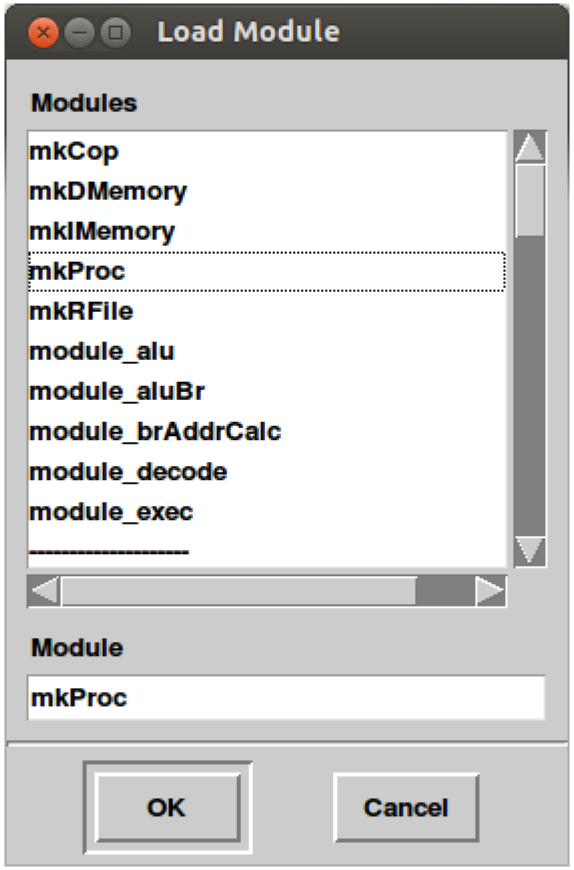
Activate Schedule Analysis Window   /home/acwright/6.s195/lectures/T05-Stuff

After you save the project, you can compile the project from the build menu. After compiling, you can open the “Schedule Analysis” window


Schedule Analysis - proc_proj
Module
Scheduling Graphs
Warnings
Rule Order
Method Call
Rule Relations



Opens the Select Module dialog



mkCop
mkDMemory
mkIMemory
mkProc
mkRFile
module_alu
module_aluBr
module_brAddrCalc
module_decode
module_exec

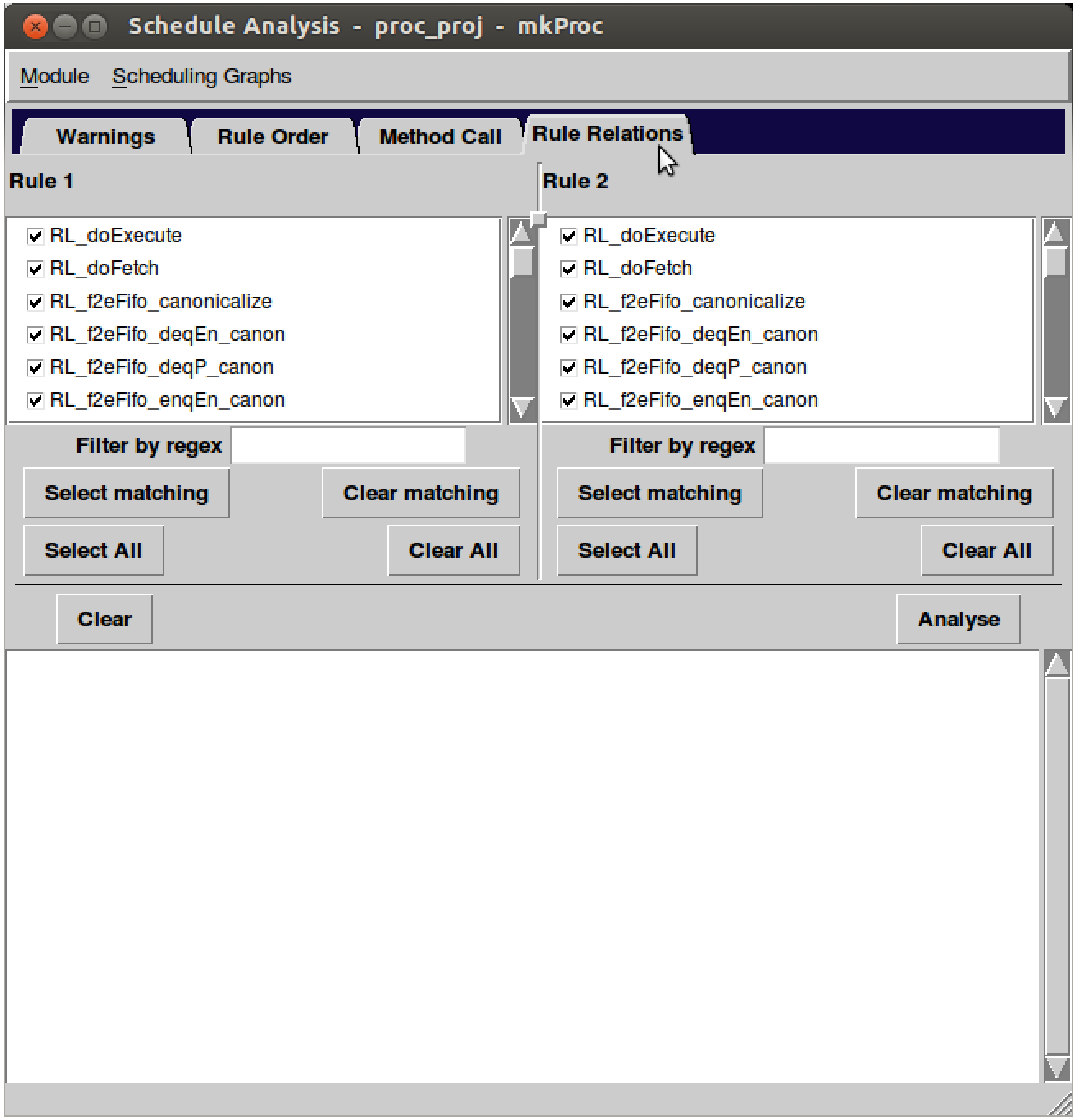
mkProc

Select mkProc and press OK

Go to the Rule Relations tab

Select only RL_doExecute and RL_doFetch from both lists

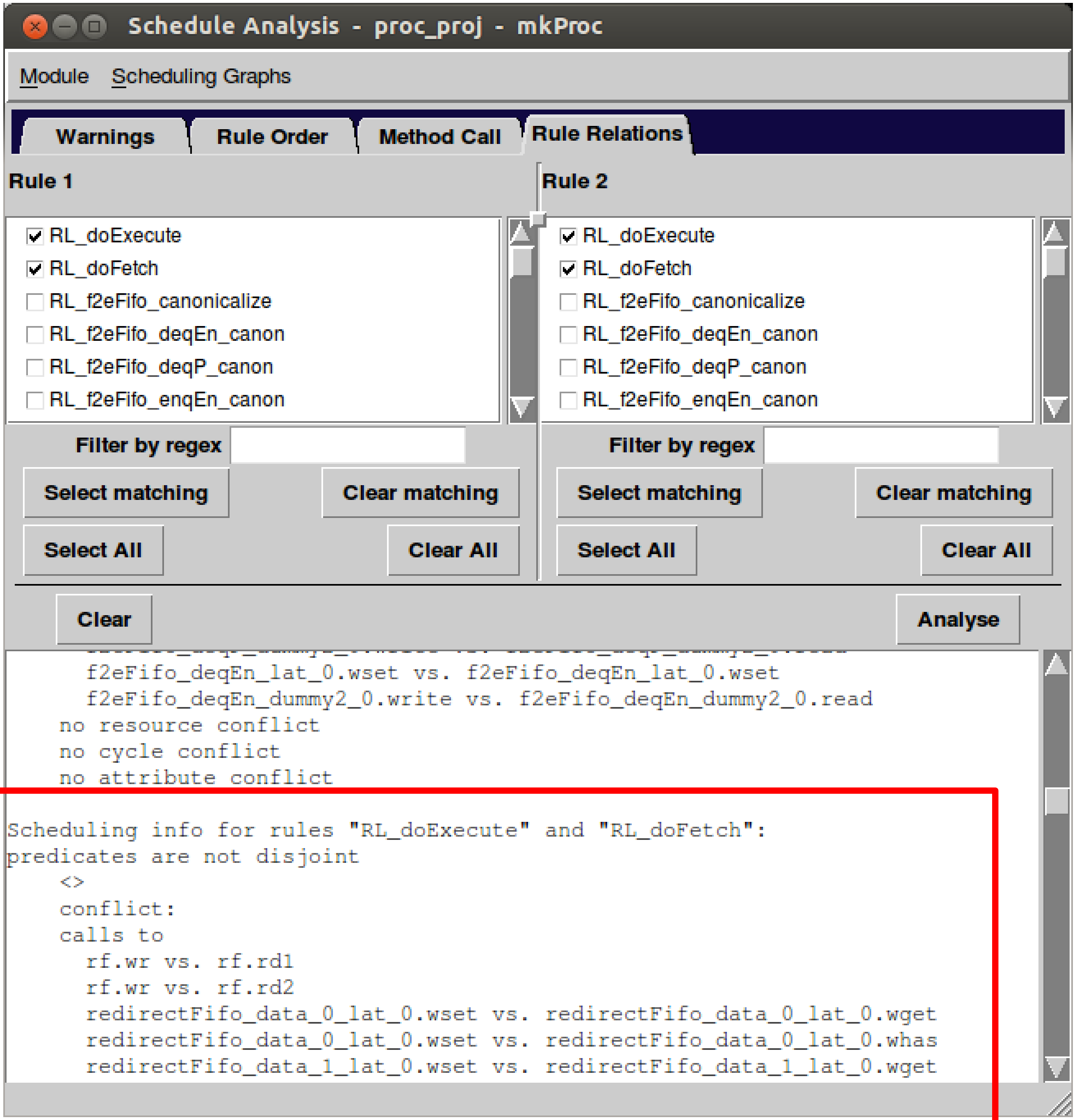
Click Analyse


Schedule Analysis - proc_proj - mkProc

Module   Scheduling Graphs

Warnings | Rule Order | Method Call | Rule Relations

Rule 1	Rule 2
☑ RL_doExecute	☑ RL_doExecute
☑ RL_doFetch	☑ RL_doFetch
☑ RL_f2eFifo_canonicalize	☑ RL_f2eFifo_canonicalize
☑ RL_f2eFifo_deqEn_canon	☑ RL_f2eFifo_deqEn_canon
☑ RL_f2eFifo_deqP_canon	☑ RL_f2eFifo_deqP_canon
☑ RL_f2eFifo_enqEn_canon	☑ RL_f2eFifo_enqEn_canon



Rule 1:

- [✓] RL_doExecute
- [✓] RL_doFetch
- [ ] RL_f2eFifo_canonicalize
- [ ] RL_f2eFifo_deqEn_canon
- [ ] RL_f2eFifo_deqP_canon
- [ ] RL_f2eFifo_enqEn_canon

Rule 2:

- [✓] RL_doExecute
- [✓] RL_doFetch
- [ ] RL_f2eFifo_canonicalize
- [ ] RL_f2eFifo_deqEn_canon
- [ ] RL_f2eFifo_deqP_canon
- [ ] RL_f2eFifo_enqEn_canon

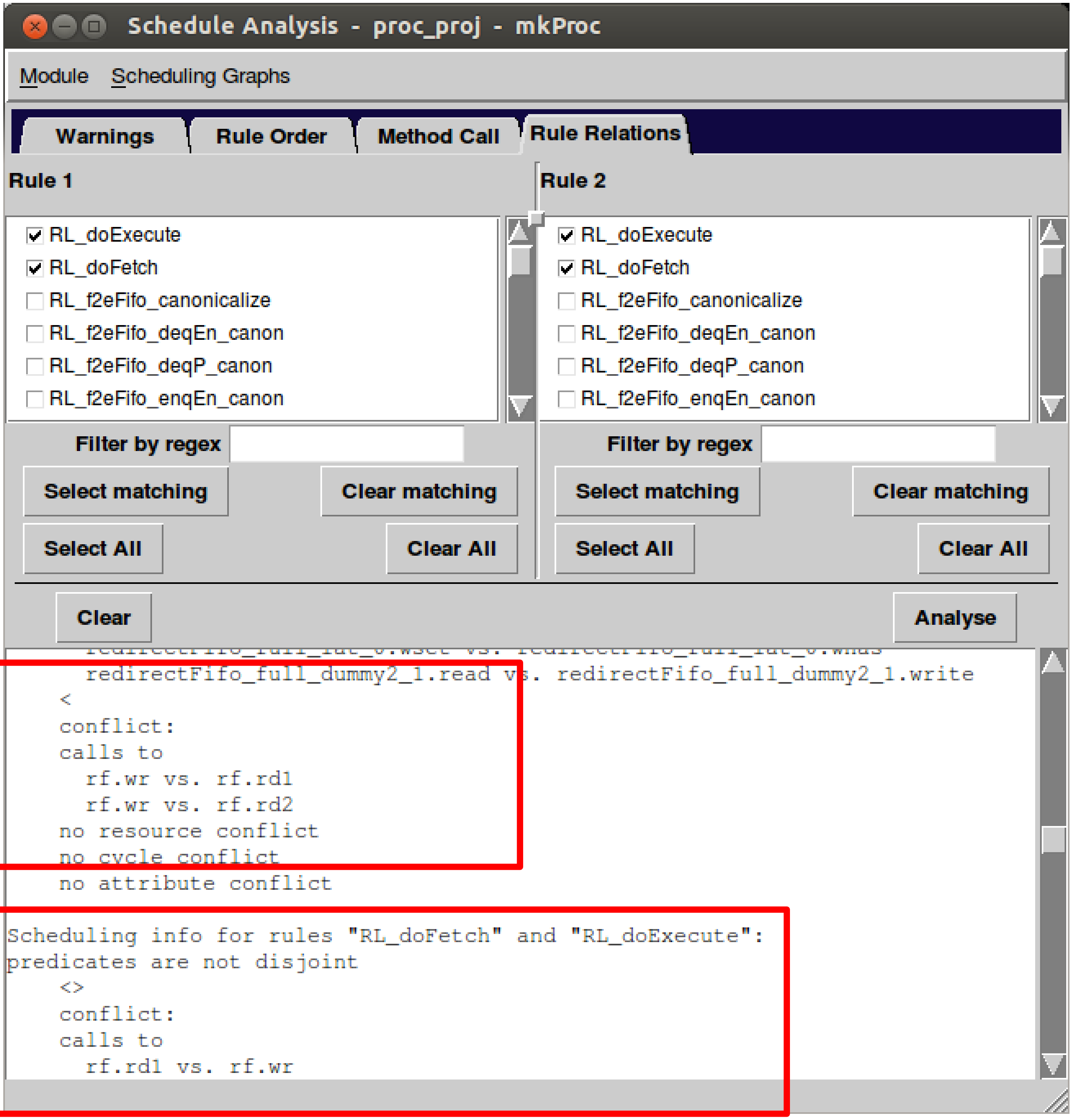
```
      f2eFifo_deqEn_lat_0.wset vs. f2eFifo_deqEn_lat_0.wset
      f2eFifo_deqEn_dummy2_0.write vs. f2eFifo_deqEn_dummy2_0.read
    no resource conflict
    no cycle conflict
    no attribute conflict

Scheduling info for rules "RL_doExecute" and "RL_doFetch":
predicates are not disjoint
    <>
    conflict:
    calls to
      rf.wr vs. rf.rd1
      rf.wr vs. rf.rd2
      redirectFifo_data_0_lat_0.wset vs. redirectFifo_data_0_lat_0.wget
      redirectFifo_data_0_lat_0.wset vs. redirectFifo_data_0_lat_0.whas
      redirectFifo_data_1_lat_0.wset vs. redirectFifo_data_1_lat_0.wget
```

The “<> conflicts:” section shows methods in each rule that are not conflict free with each other. This does not mean the two rules can't fire together, it just means the rules aren't conflict-free.

The “< conflicts:” section shows methods in each rule that prevent rule 1 from firing before rule 2. doExecute can't be before doFetch

Now to look at the two rules in the opposite order



Rule 1:
- [✓] RL_doExecute
- [✓] RL_doFetch
- [ ] RL_f2eFifo_canonicalize
- [ ] RL_f2eFifo_deqEn_canon
- [ ] RL_f2eFifo_deqP_canon
- [ ] RL_f2eFifo_enqEn_canon

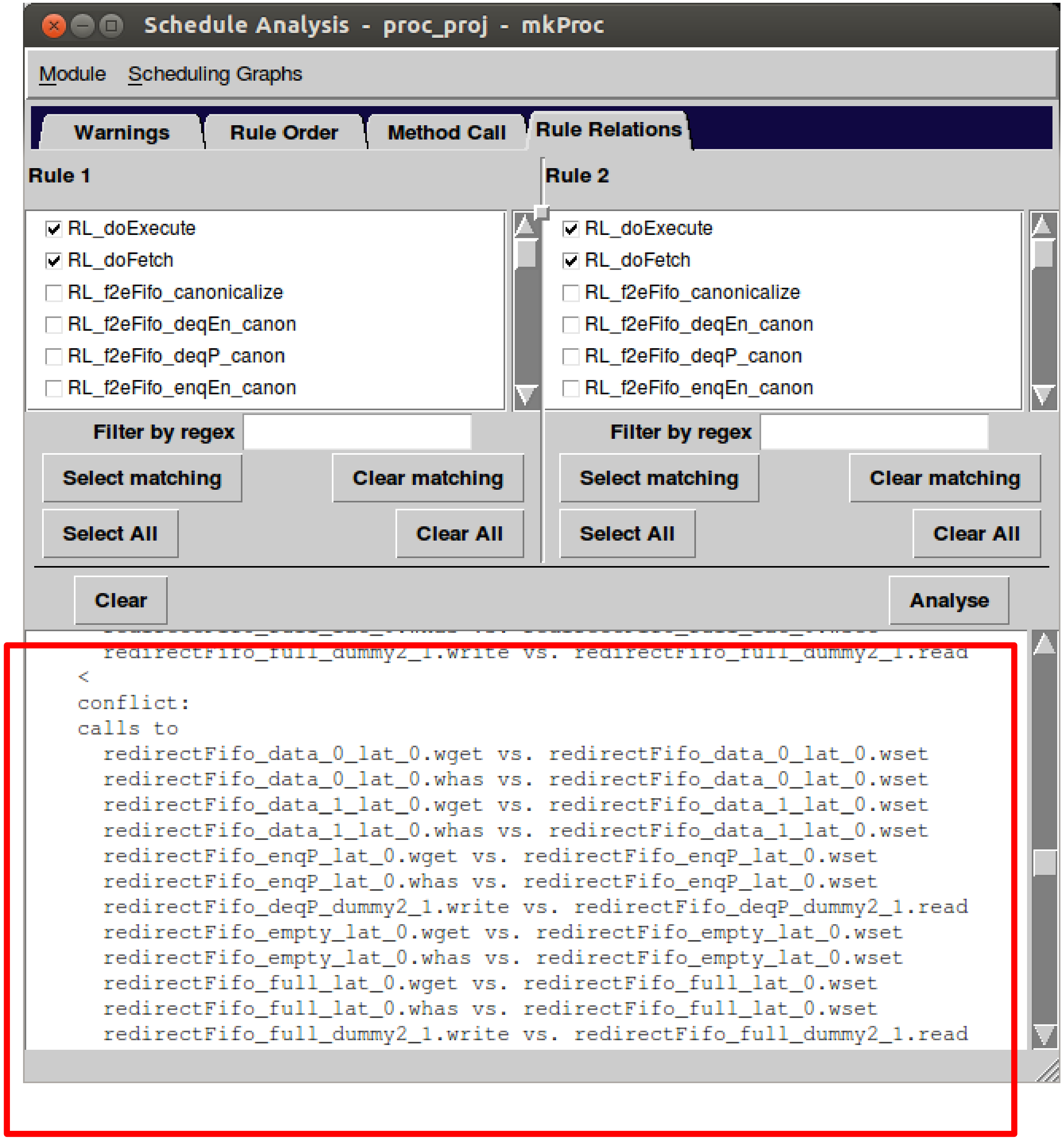
Rule 2:
- [✓] RL_doExecute
- [✓] RL_doFetch
- [ ] RL_f2eFifo_canonicalize
- [ ] RL_f2eFifo_deqEn_canon
- [ ] RL_f2eFifo_deqP_canon
- [ ] RL_f2eFifo_enqEn_canon

```
      redirectFifo_full_lat_0.wset vs. redirectFifo_full_lat_0.whas
      redirectFifo_full_dummy2_1.read vs. redirectFifo_full_dummy2_1.write
    <
    conflict:
    calls to
      rf.wr vs. rf.rd1
      rf.wr vs. rf.rd2
    no resource conflict
    no cycle conflict
    no attribute conflict

Scheduling info for rules "RL_doFetch" and "RL_doExecute":
predicates are not disjoint
    <>
    conflict:
    calls to
      rf.rd1 vs. rf.wr
```

More conflicts. DoFetch can't occur before doExecute.

Therefore doFetch and doExecute can't fire in the same cycle



Rule 1

- ☑ RL_doExecute
- ☑ RL_doFetch
- ☐ RL_f2eFifo_canonicalize
- ☐ RL_f2eFifo_deqEn_canon
- ☐ RL_f2eFifo_deqP_canon
- ☐ RL_f2eFifo_enqEn_canon

Rule 2

- ☑ RL_doExecute
- ☑ RL_doFetch
- ☐ RL_f2eFifo_canonicalize
- ☐ RL_f2eFifo_deqEn_canon
- ☐ RL_f2eFifo_deqP_canon
- ☐ RL_f2eFifo_enqEn_canon

```
    redirectFifo_full_dummy2_1.write vs. redirectFifo_full_dummy2_1.read
  <
  conflict:
  calls to
    redirectFifo_data_0_lat_0.wget vs. redirectFifo_data_0_lat_0.wset
    redirectFifo_data_0_lat_0.whas vs. redirectFifo_data_0_lat_0.wset
    redirectFifo_data_1_lat_0.wget vs. redirectFifo_data_1_lat_0.wset
    redirectFifo_data_1_lat_0.whas vs. redirectFifo_data_1_lat_0.wset
    redirectFifo_enqP_lat_0.wget vs. redirectFifo_enqP_lat_0.wset
    redirectFifo_enqP_lat_0.whas vs. redirectFifo_enqP_lat_0.wset
    redirectFifo_deqP_dummy2_1.write vs. redirectFifo_deqP_dummy2_1.read
    redirectFifo_empty_lat_0.wget vs. redirectFifo_empty_lat_0.wset
    redirectFifo_empty_lat_0.whas vs. redirectFifo_empty_lat_0.wset
    redirectFifo_full_lat_0.wget vs. redirectFifo_full_lat_0.wset
    redirectFifo_full_lat_0.whas vs. redirectFifo_full_lat_0.wset
    redirectFifo_full_dummy2_1.write vs. redirectFifo_full_dummy2_1.read
```