Reveal

Heidi Poxon
Manager & Technical Lead, Performance Tools
Cray Inc.

Approach to Adding Parallelism

1. **Identify possible accelerator kernels**
   - Determine where to add additional levels of parallelism
     - Assumes MPI application is functioning correctly on X86
     - Find top serial work-intensive loops (perftools + CCE loop work estimates)
2. **Perform parallel analysis, scoping and vectorization**
   - Split loop work among threads
     - Do parallel analysis and restructuring on targeted high level loops
     - Use CCE loopmark feedback, Reveal loopmark and source browsing
3. **Move to OpenMP and then to OpenACC**
   - Add parallel directives and acceleration extensions
     - Insert OpenMP directives (Reveal scoping assistance)
     - Run on X86 to verify application and check for performance improvements
     - Convert desired OpenMP directives to OpenACC
4. **Analyze performance for further optimizations**

WARNING!!!

- **Nothing comes for free, nothing is automatic**
  - Hybridization of an application is difficult
  - Efficient code requires interaction with the compiler to generate
    - High level OpenMP structures
    - Low level vectorization of major computational areas
- **Performance is also dependent upon the location of the data**
  - CPU: NUMA, first-touch
  - Accelerator: resident or data-sloshing
- **Software such as Cray's Hybrid Programming Environment provides tools to help, but cannot replace the developer's inside knowledge**

Reveal

New code analysis and restructuring assistant…

- **Uses both the performance toolset and CCE's program library functionality to provide static and runtime analysis information**

- **Key Features**
  - Annotated source code with compiler optimization information
    - Feedback on critical dependencies that prevent optimizations
  - Scoping analysis
    - Identify, shared, private and ambiguous arrays
      - Allow user to privatize ambiguous arrays
      - Allow user to override dependency analysis
  - Source code navigation based on performance data collected through CrayPat

How to Use Reveal

Optionally create loop statistics using the Cray performance tools to determine which loops have the most work

Loop Work Estimates

- **Helps identify high-level serial loops to parallelize**
  - Based on runtime analysis, approximates how much work exists within a loop
- **Provides the following statistics**
  - Min, max and average trip counts that can be used to approximate work and help carve up loop on GPU
  - Inclusive time spent in loops
  - Number of times a loop was executed

Collecting Loop Work Estimates

- Load PrgEnv-cray module (must use CCE)
- Load perftools module

- Compile AND link with –h profile_generate
  - cc –h profile_generate –o my_program my_program.c

- Instrument binary for tracing
  - pat_build –w my_program

- Run application

- Create text report with loop statistics
  - pat_report my_program.xf > loops_report

Example Report – Inclusive Loop Time

Table 2: Loop Stats by Function (from -hprofile_generate)

Loop Incl Time Total	Loop Hit	Loop Trips Avg	Loop Trips Min	Loop Trips Max	Function=/.LOOP[.] PE=HIDE
8.995914	100	25	0	25	sweepy_.LOOP.1.li.33
8.995604	2500	25	0	25	sweepy_.LOOP.2.li.34
8.894750	50	25	0	25	sweepz_.LOOP.05.li.49
8.894637	1250	25	0	25	sweepz_.LOOP.06.li.50
4.420629	50	25	0	25	sweepx2_.LOOP.1.li.29
4.420536	1250	25	0	25	sweepx2_.LOOP.2.li.30
4.387534	50	25	0	25	sweepx1_.LOOP.1.li.29
4.387457	1250	25	0	25	sweepx1_.LOOP.2.li.30
2.523214	187500	107	0	107	riemann_.LOOP.2.li.63
1.541299	20062500	12	0	12	riemann_.LOOP.3.li.64
0.863656	1687500	104	0	108	parabola_.LOOP.6.li.67

How to Use Reveal (Continued)

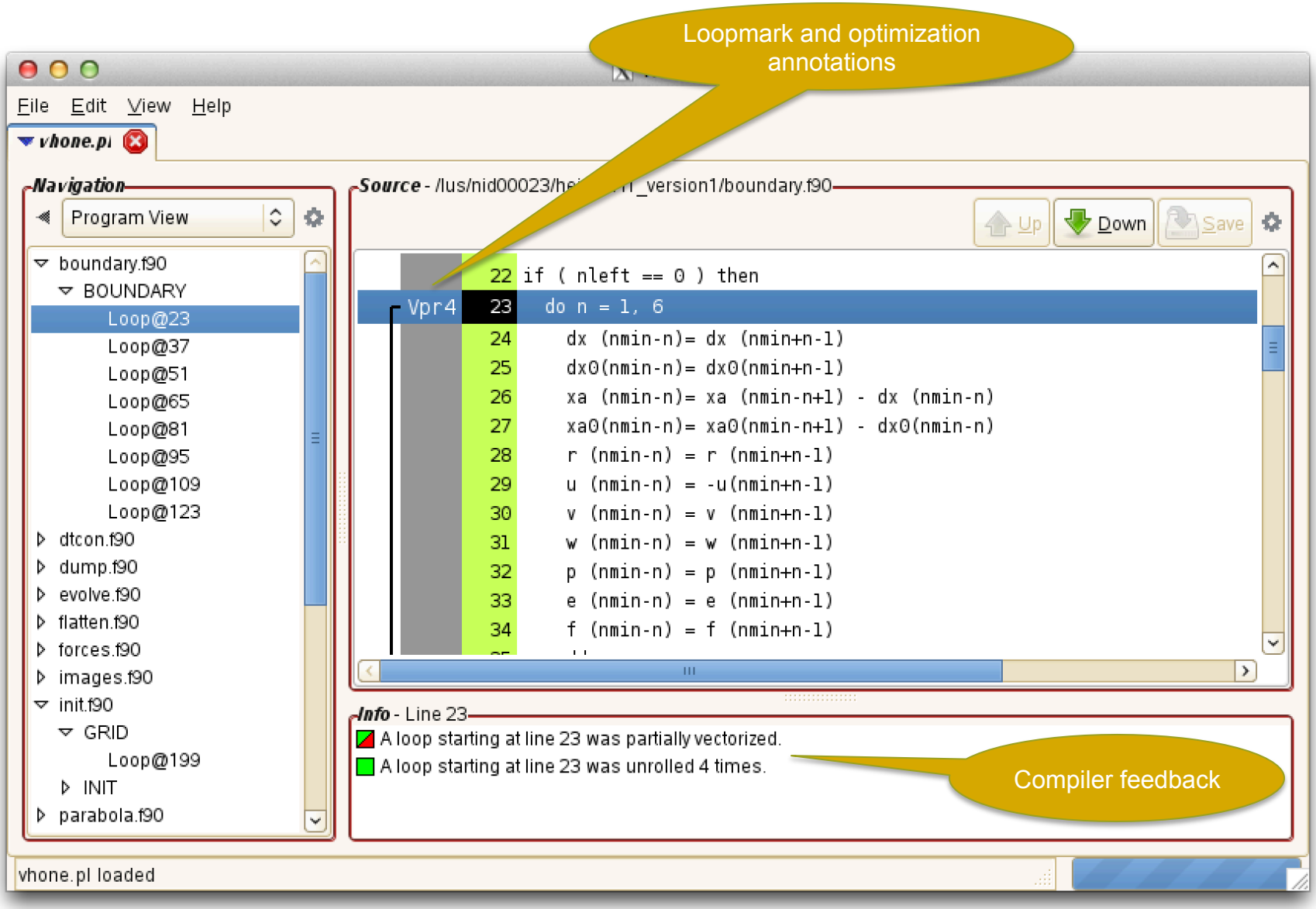
- **Generate a program library for your application with CCE**
  - `> cc -h pl=himeno.pl -hwp himeno.c`
  - `> ftn -h pl=vhone.pl -hwp file1.f90`

Optionally add whole program analysis for more aggressive inlining

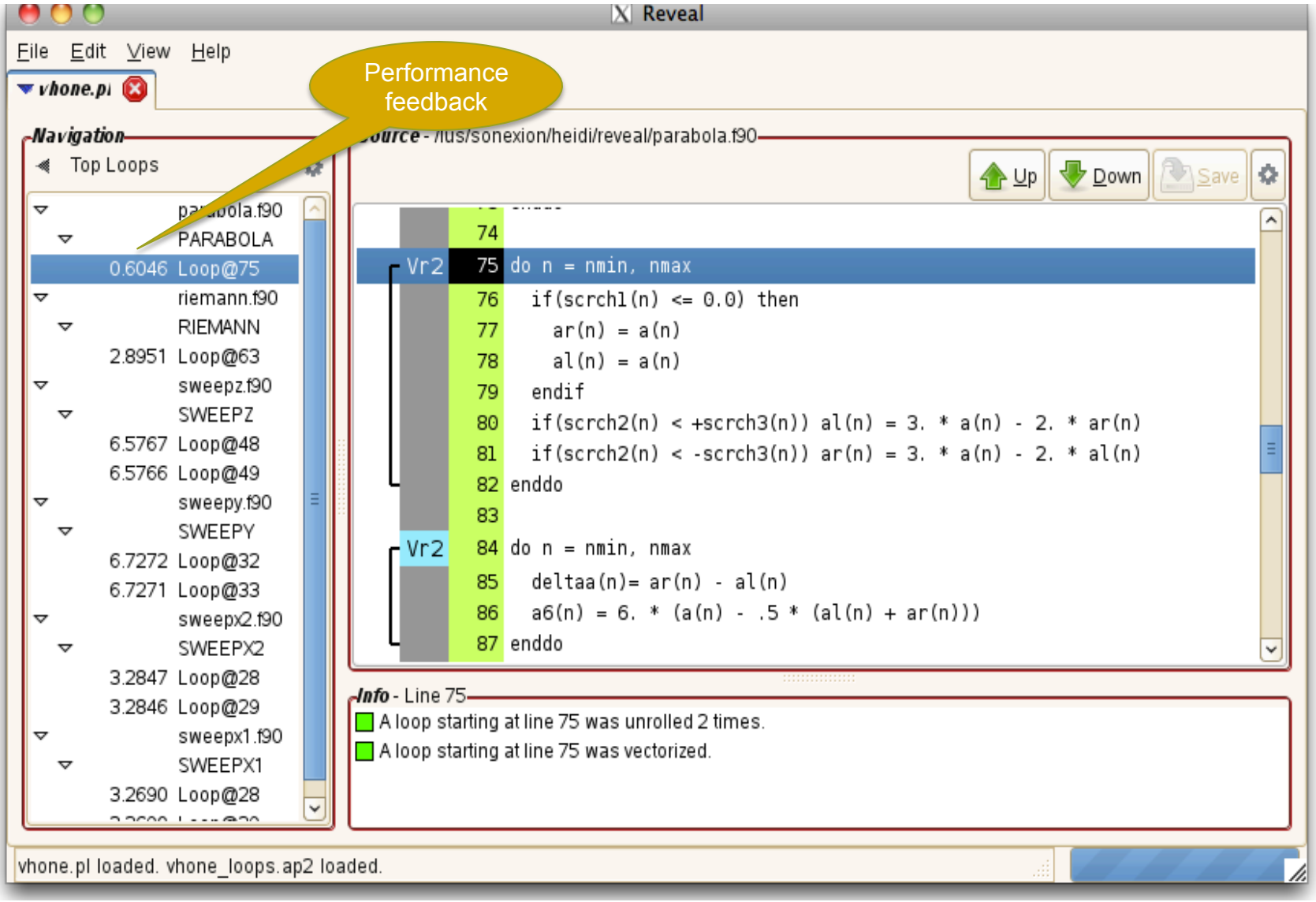
- **Run Reveal**
  - Use with compiler information only (no need to run program):
    - `> reveal vhone.pl`
  - Use with compiler + loop work estimates (include performance data)
    - `> reveal vhone.pl vhone_loops.ap2`


Reveal
File Edit View Help
About Reveal
CRAY
REVEAL
New to Reveal? Click here for Getting Started Info

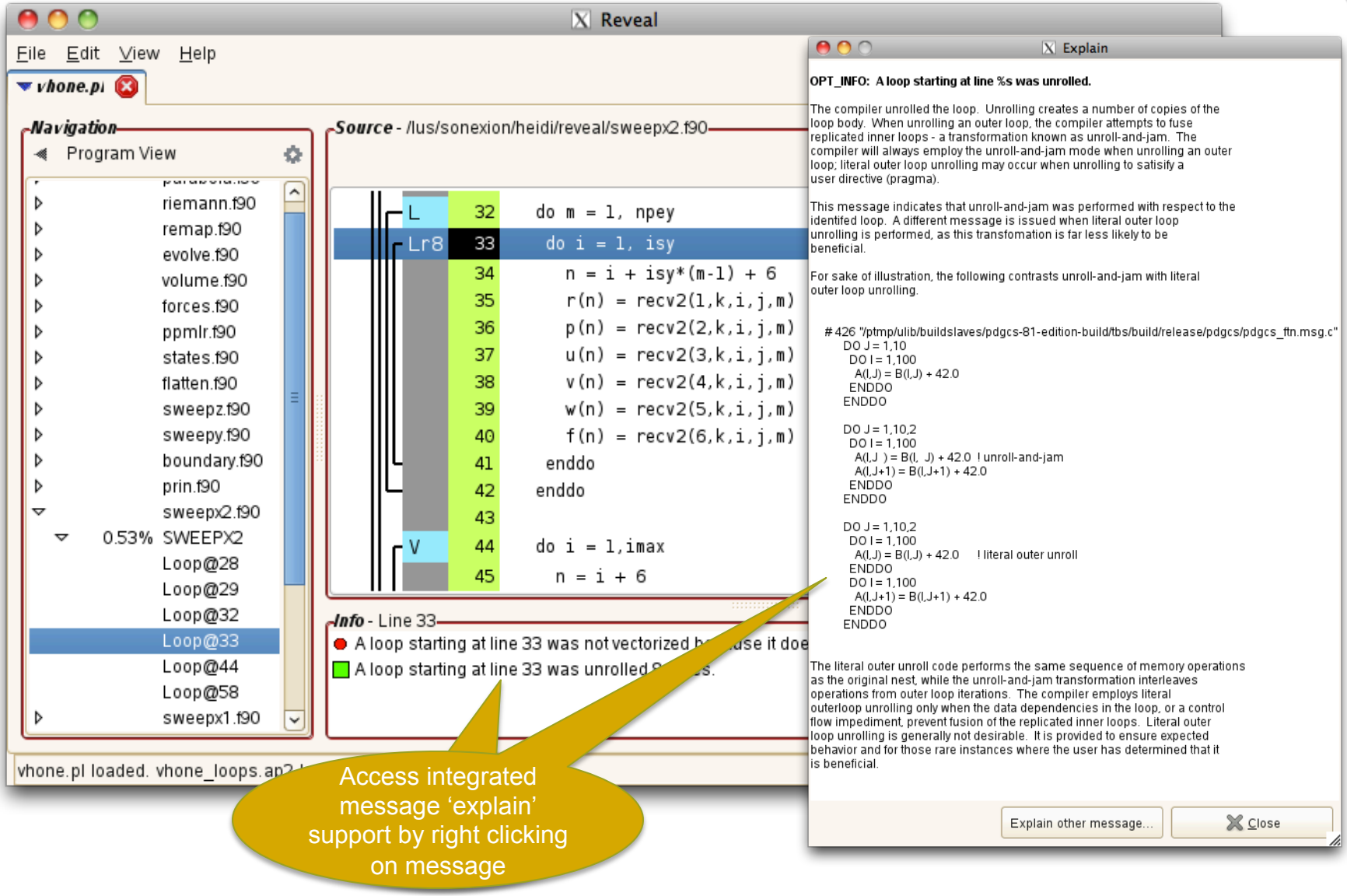
Visualize Loopmark & Compiler Optimizations



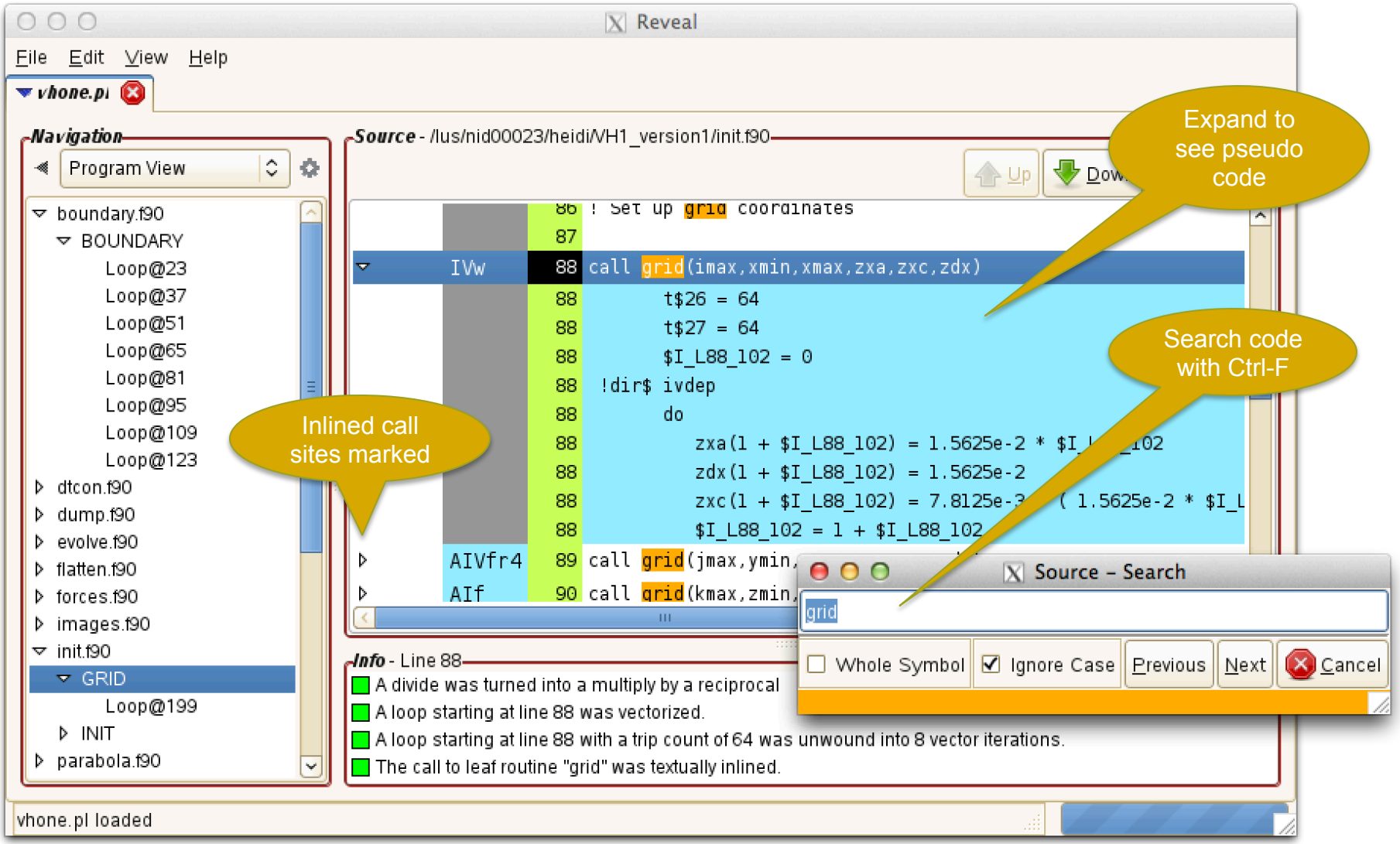
Navigate Source with Performance Information



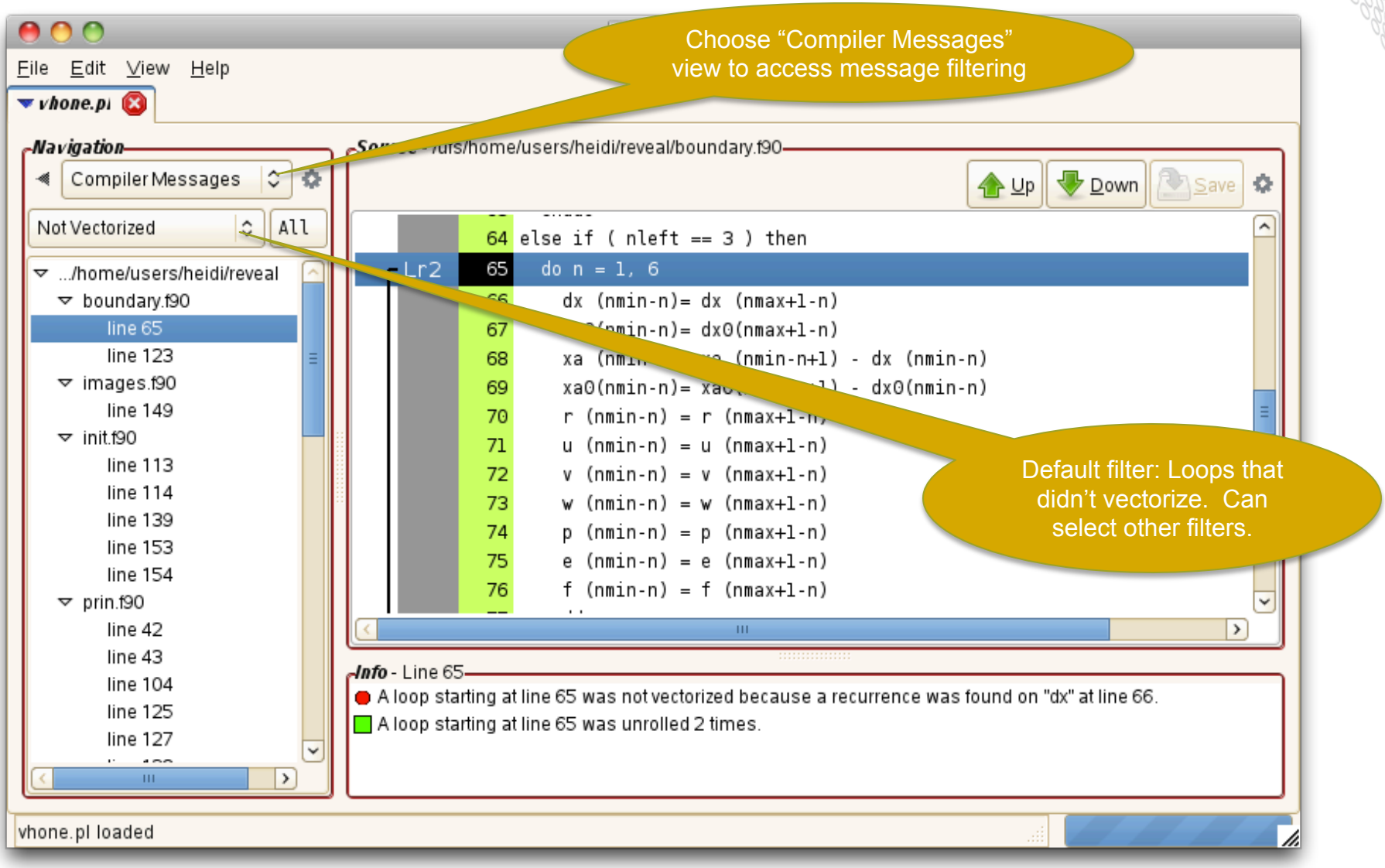
Access CCE's Message Explanations



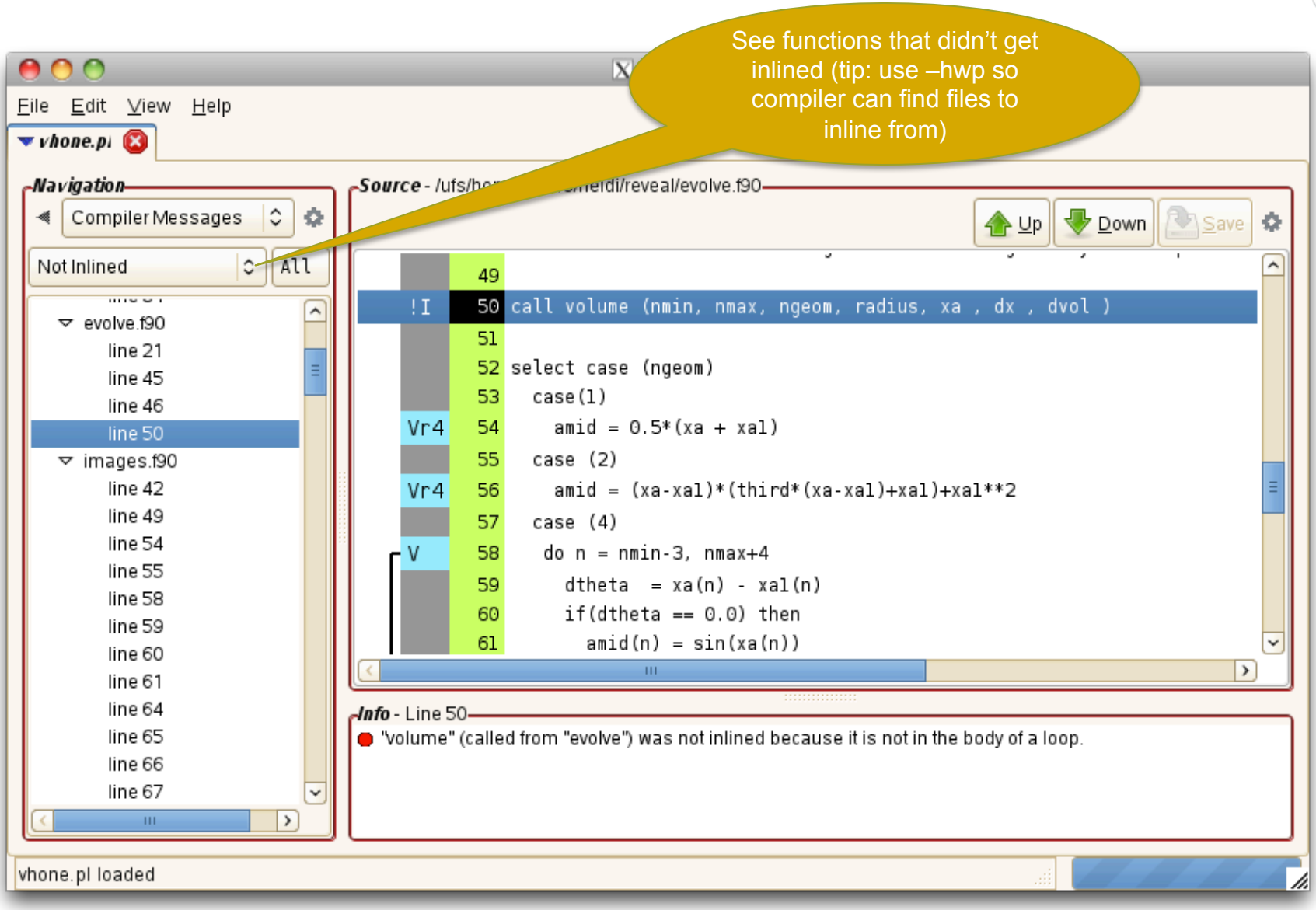
View Pseudo Code for Inlined Functions



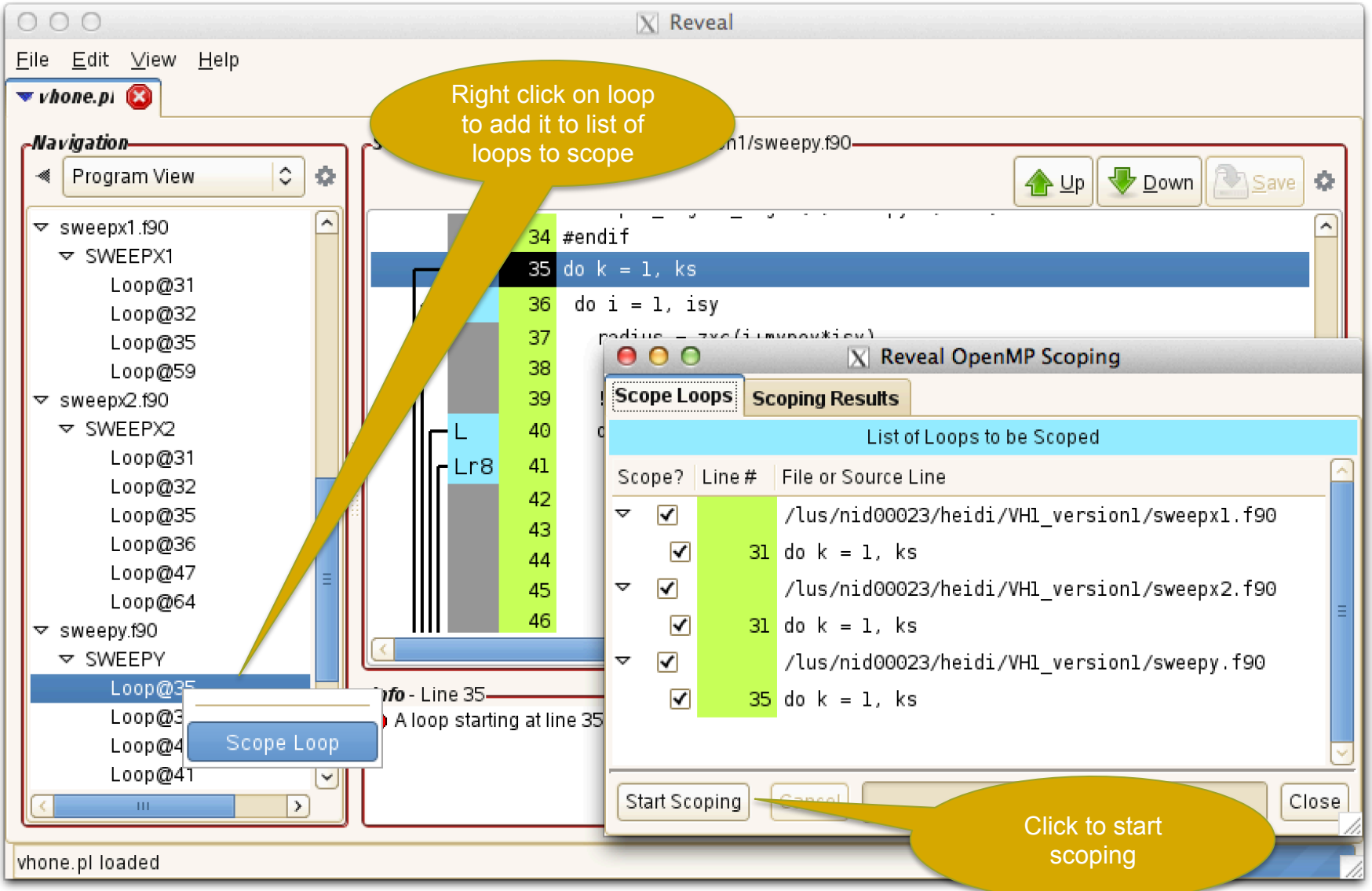
Navigate Code via Compiler Messages



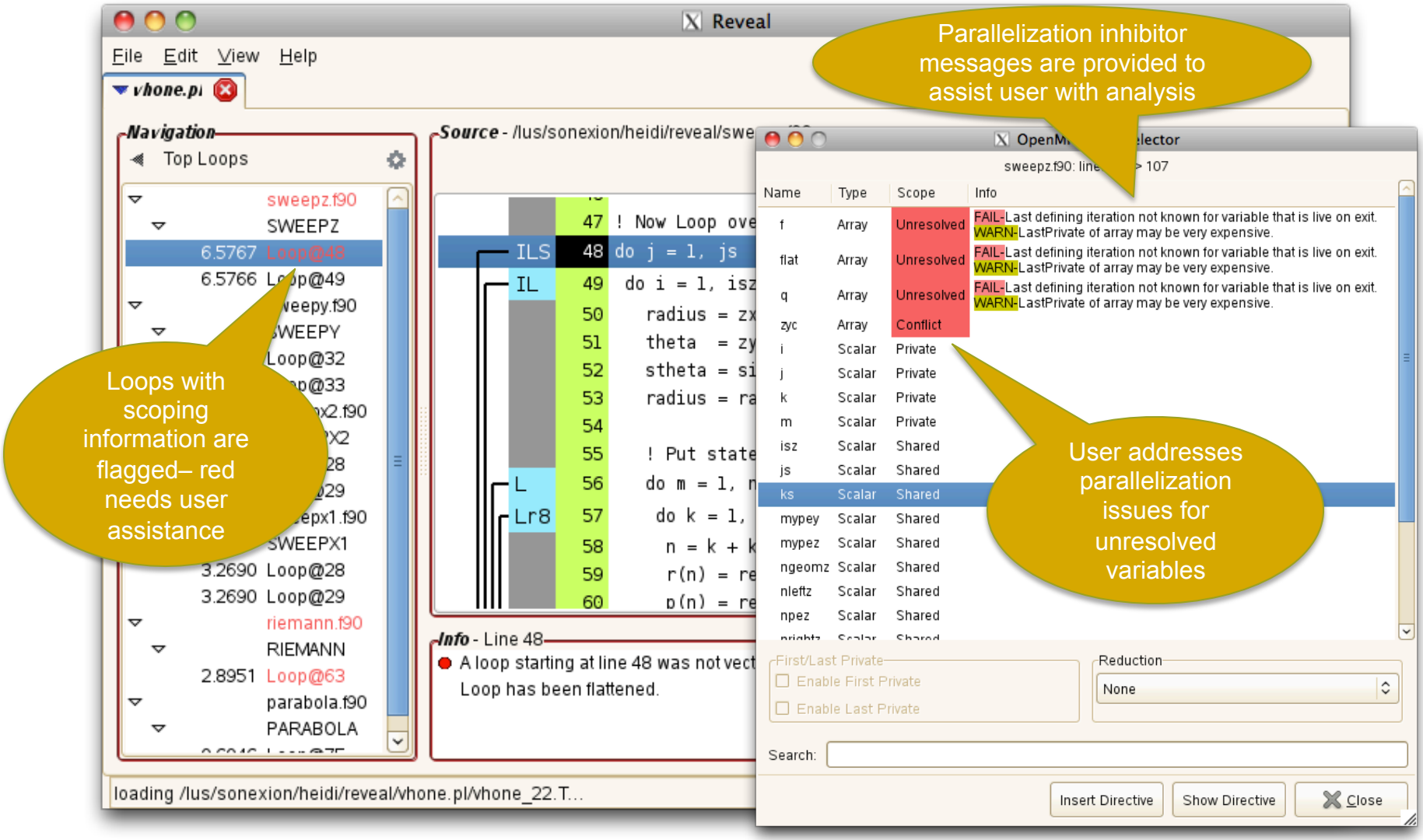
Navigate Code via Compiler Messages (2)



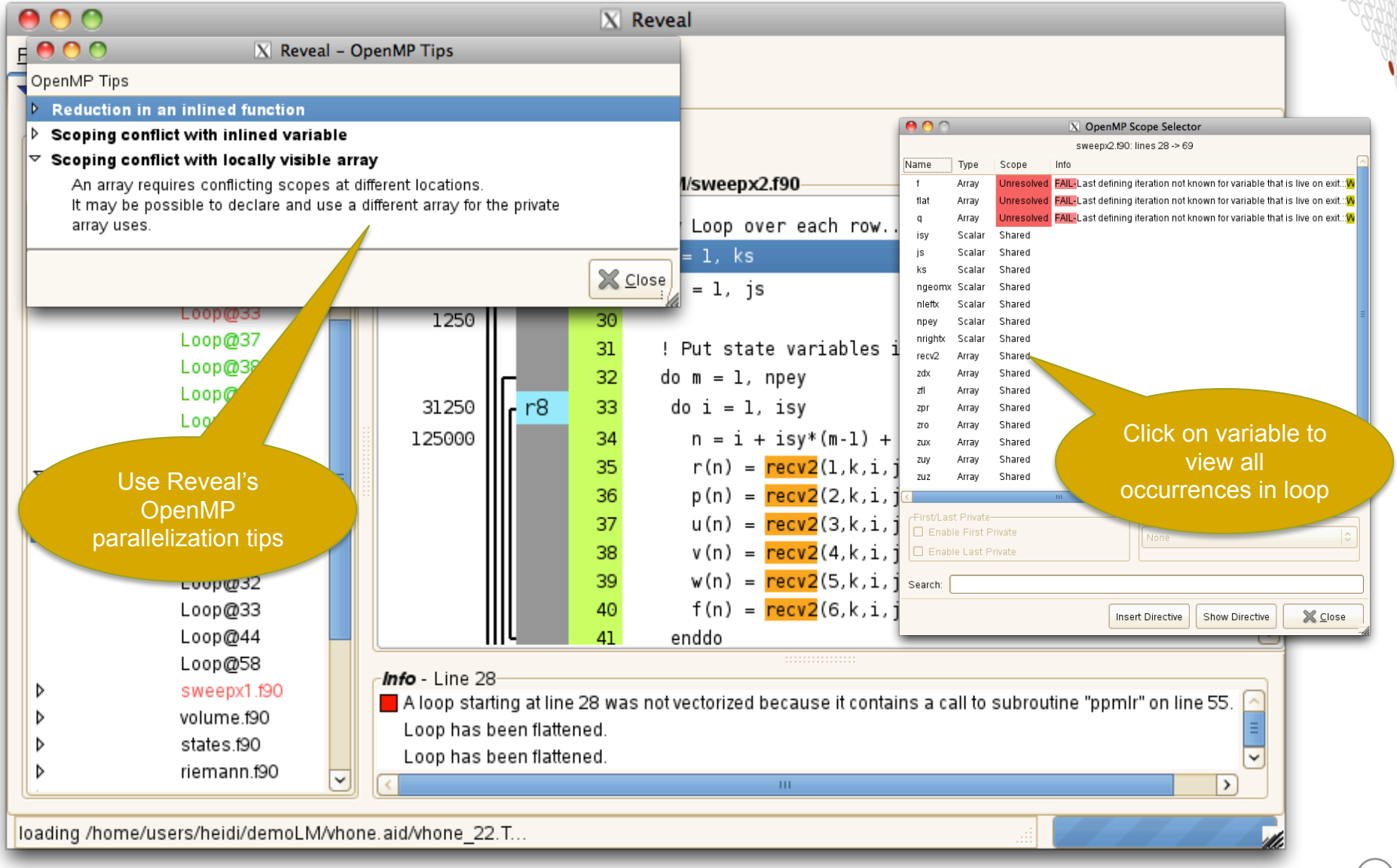
Scoping Assistance - Scope Loop(s)



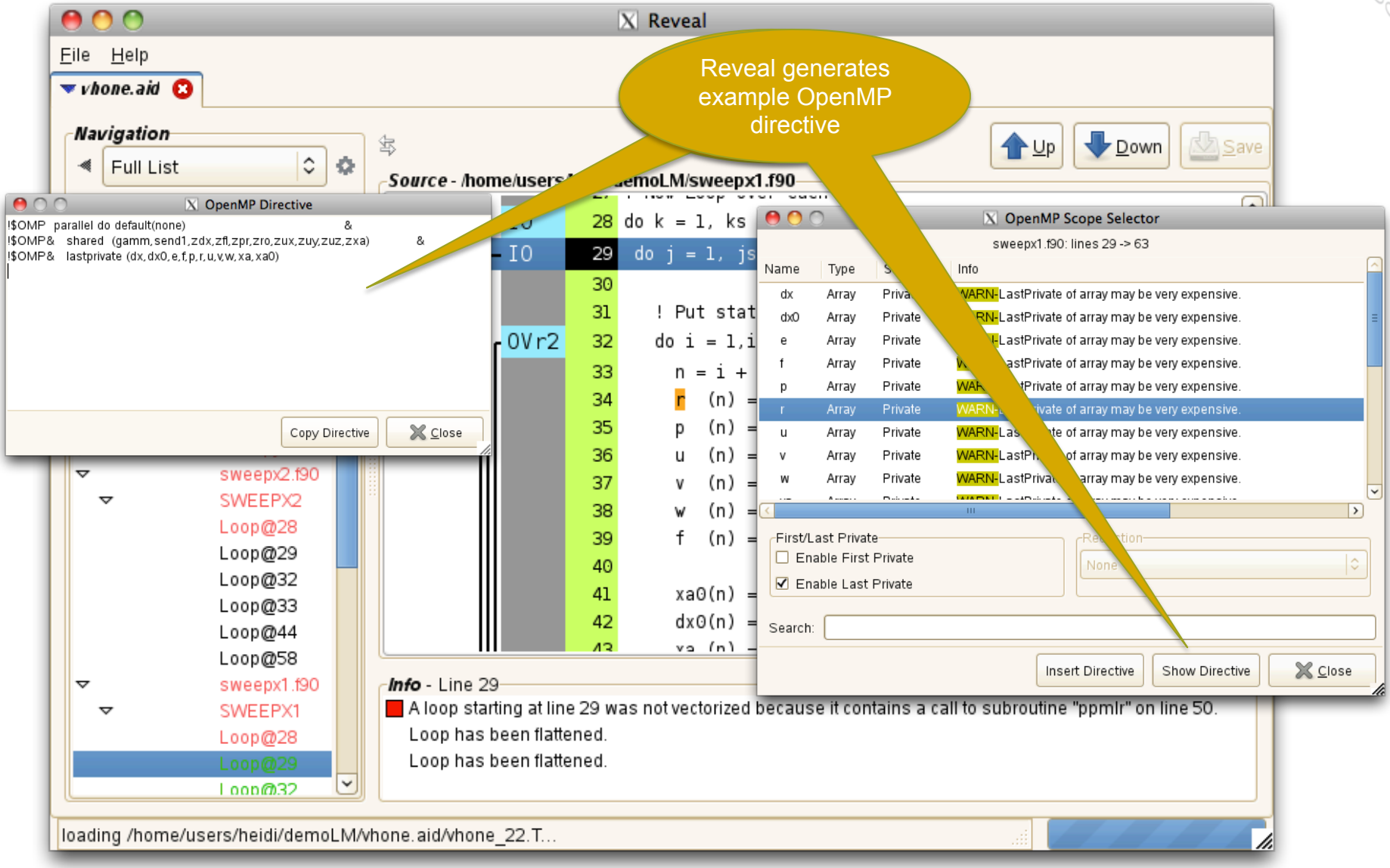
Scoping Assistance – Review Scoping Results



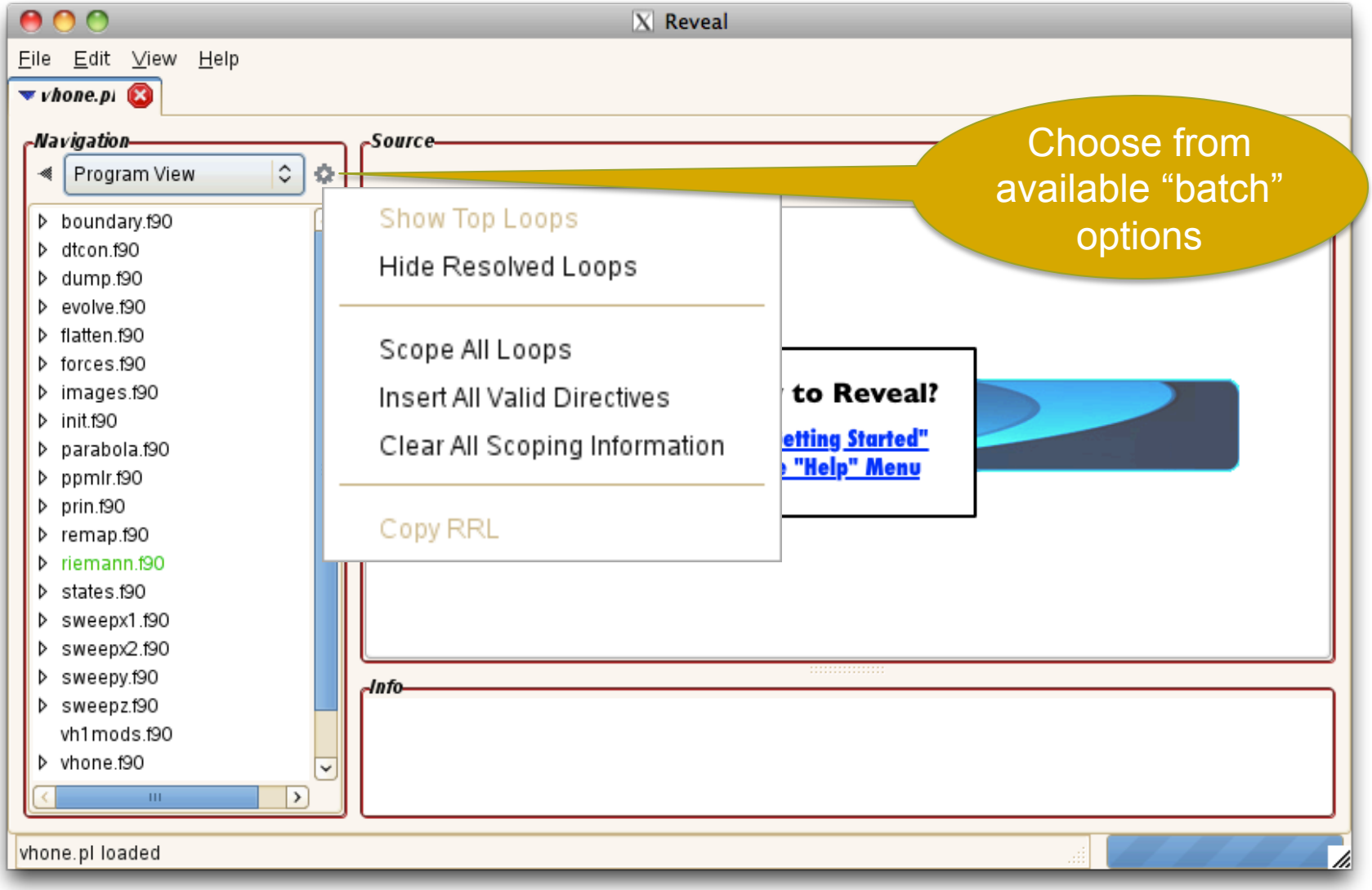
Scoping Assistance – User Resolves Issues



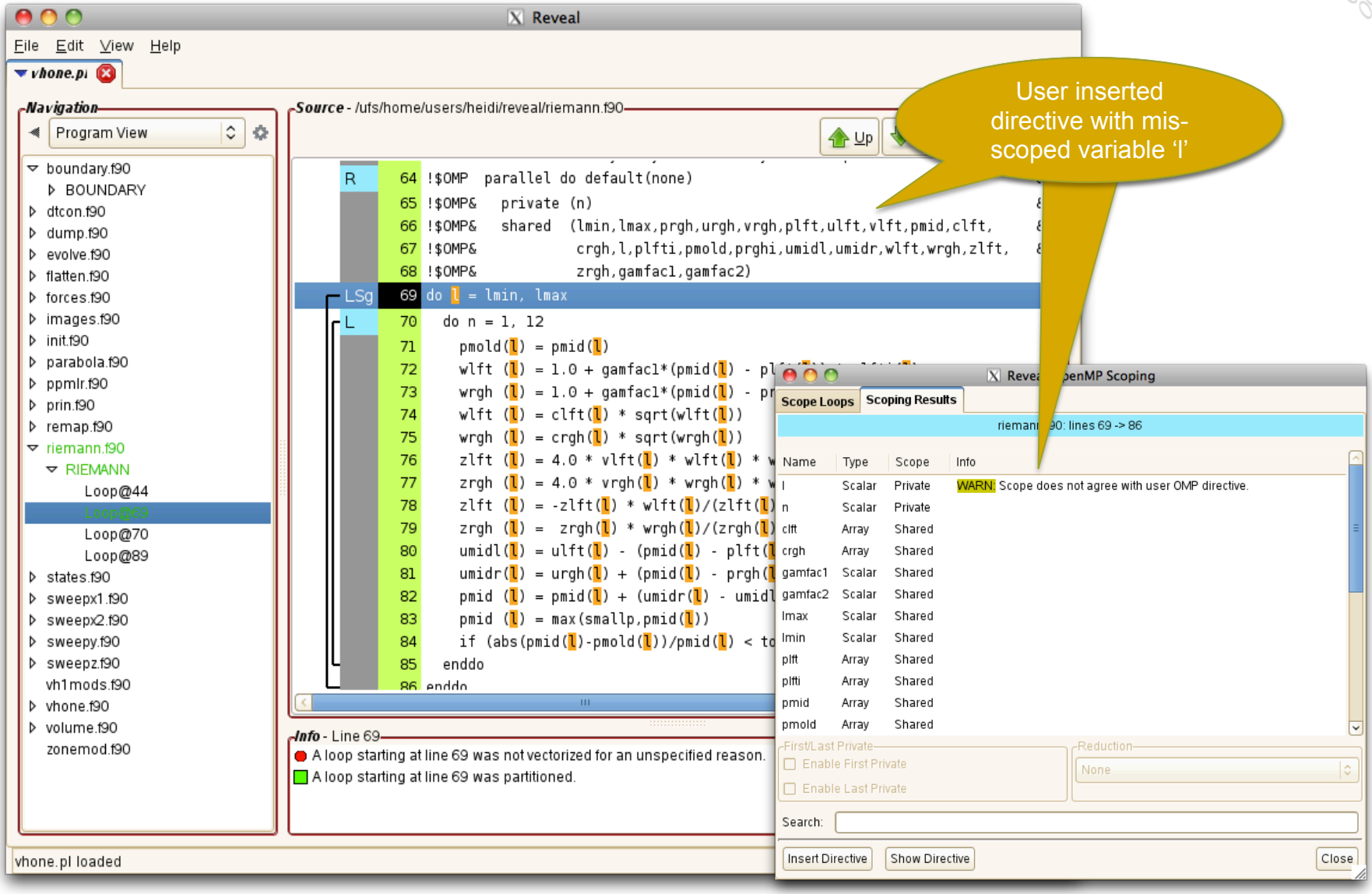
Scoping Assistance – Reveal Generates Directive



Batch Options Available



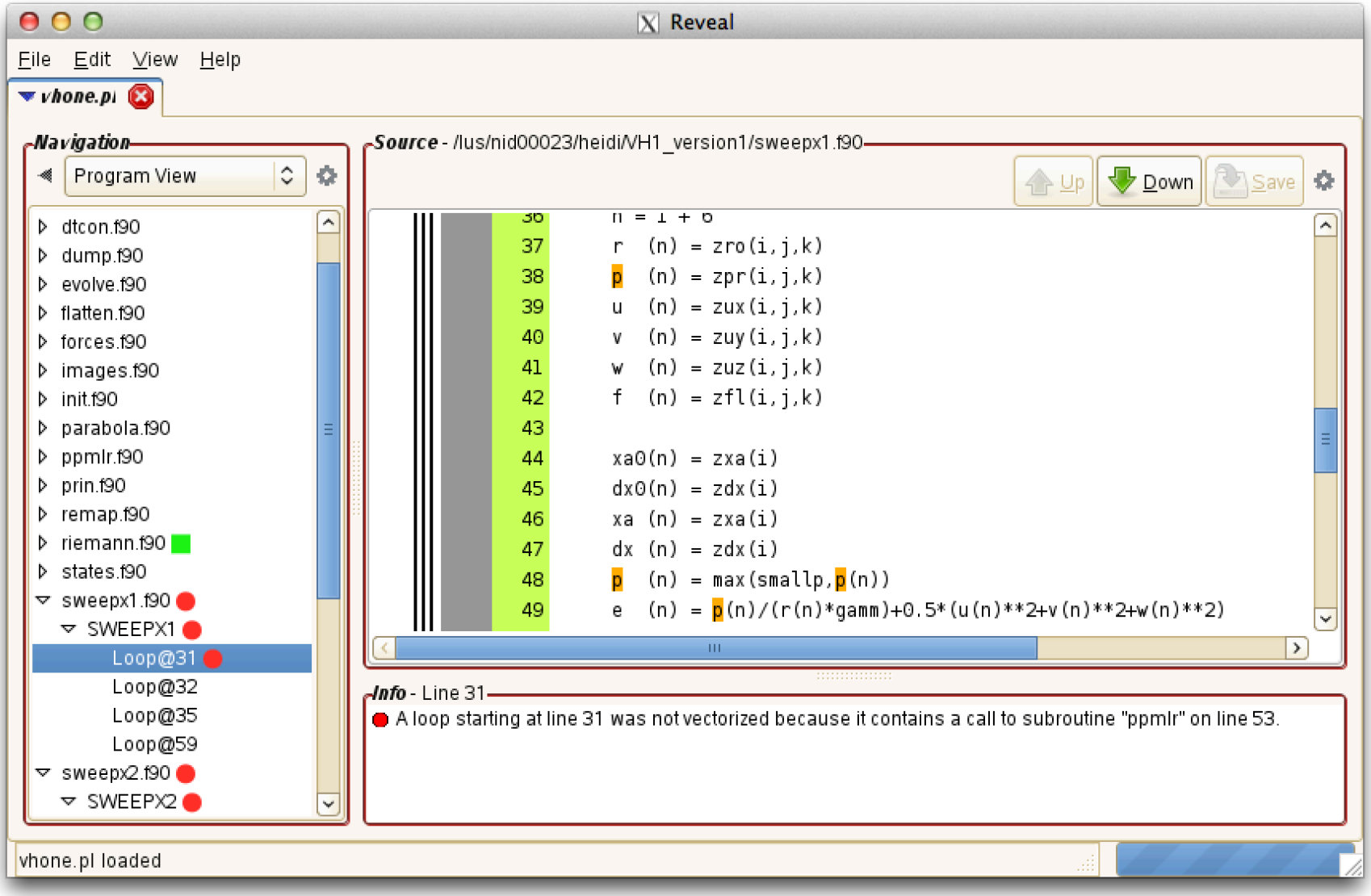
Use Reveal to Validate User Inserted Directives



What's Coming Next…

- **More information on scoped variables**
  - Call out variables included in scoping results due to inlining (shouldn't be included on OMP directive)
  - Call out reductions
- **Navigation panel coloring improvements**
  - Follow Info panel style when highlighting loops that are scoped
- **OpenACC support**
  - Scoping for OpenACC loops
  - OpenACC directive generation
  - Data region building support

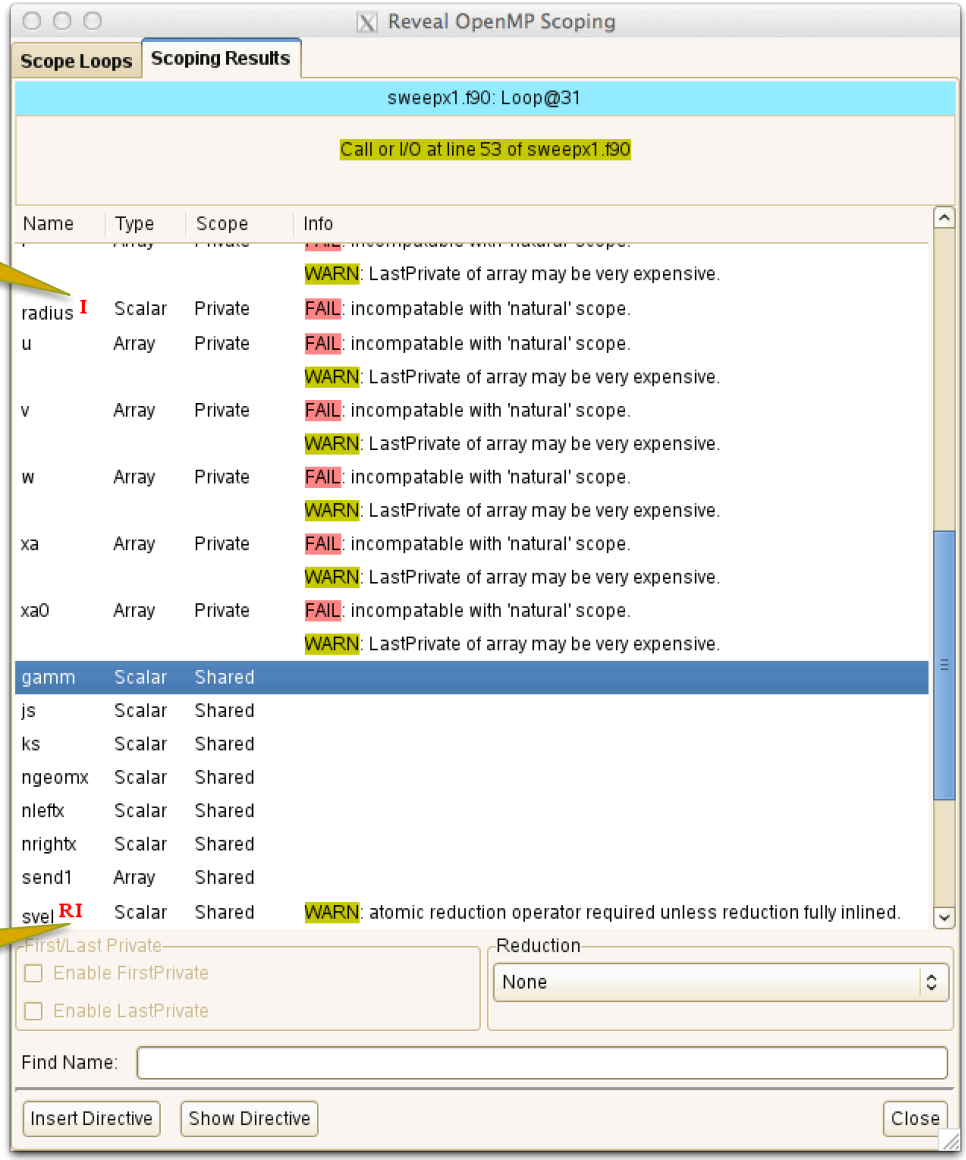
Highlighting Scoped Loops…



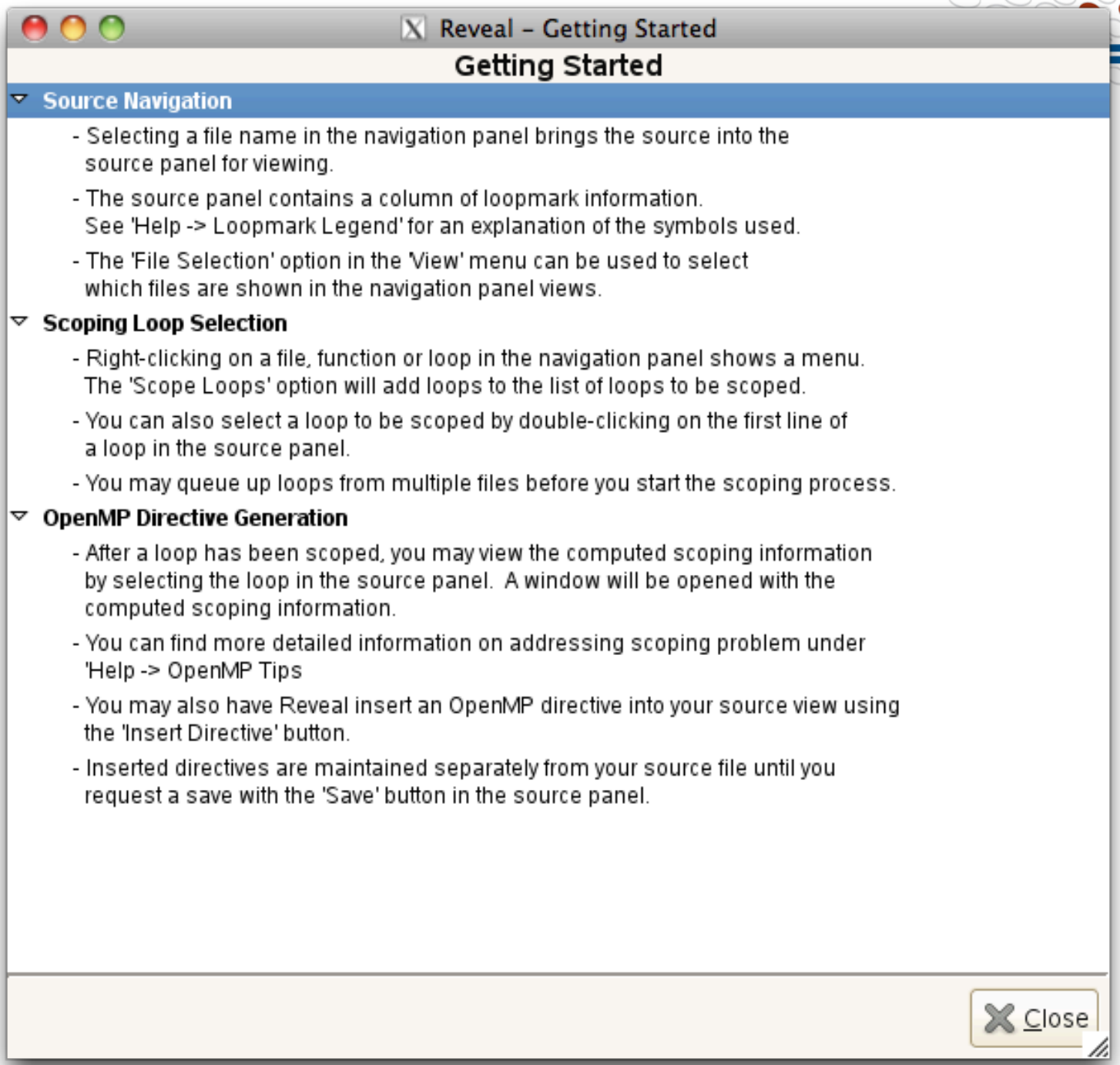
More Information on Scoped Variables

Variable from inlining – hover over 'I' to see what symbol means

Reduction variable from down the call stack - hover over 'RI' to see what symbol means



Questions ?


Reveal – Getting Started
Getting Started
Source Navigation
- Selecting a file name in the navigation panel brings the source into the source panel for viewing.
- The source panel contains a column of loopmark information. See 'Help -> Loopmark Legend' for an explanation of the symbols used.
- The 'File Selection' option in the 'View' menu can be used to select which files are shown in the navigation panel views.
Scoping Loop Selection
- Right-clicking on a file, function or loop in the navigation panel shows a menu. The 'Scope Loops' option will add loops to the list of loops to be scoped.
- You can also select a loop to be scoped by double-clicking on the first line of a loop in the source panel.
- You may queue up loops from multiple files before you start the scoping process.
OpenMP Directive Generation
- After a loop has been scoped, you may view the computed scoping information by selecting the loop in the source panel. A window will be opened with the computed scoping information.
- You can find more detailed information on addressing scoping problem under 'Help -> OpenMP Tips
- You may also have Reveal insert an OpenMP directive into your source view using the 'Insert Directive' button.
- Inserted directives are maintained separately from your source file until you request a save with the 'Save' button in the source panel.
Close