Modifying ROSE Compiler

Adding ALLOCATE clause to parallel directive

Adding directives or clauses to ROSE involves modification to several files. Take the example of adding the **ALLOCATE** clause to **PARALLEL** directive. Our goals at this point are to:

- Ensure that **ALLOCATE** clause is added and the clause can show up on the AST graphs.
- Be able to run source to source translation whereby unparser should be able to re-generate the original source code from the AST graph. This can be achieved by the **identityTranslator** tests as shown later in this documentation.
- NOTE: It is important to take note if the clause will be processed as a variable clause or an expression since that defines how you process the file in unparser and the lexer.

There are a number of files to modify when adding a clause. Since each is detailed, it is better to open each file and locate the ALLOCATE clause to see the type of change. Here below is the list of files affected:

- /src/ROSETTA/astNodeList b/src/ROSETTA/astNodeList
- /src/ROSETTA/src/node.C
- /src/backend/unparser/languageIndependenceSupport/unparseLanguageIndependentConstructs.C
- /src/frontend/SageIII/OmpAttribute.C
- /src/frontend/SageIII/OmpAttribute.h
- /src/frontend/SageIII/astMerge/collectAssociateNodes.C
- /src/frontend/SageIII/nodeBuildFunctionsForAterms.C
- /src/frontend/SageIII/ompAstConstruction.cpp
- /src/frontend/SageIII/omplexer.ll
- /src/frontend/SageIII/ompparser.yy
- /src/midend/programTransformation/ompLowering/omp_lowering.cpp

Modifying ROSE Compiler

Sample Test Program

```
#include "omp.h"
int omp_get_thread_num();
int omp_get_num_threads();
int main(int arg, char *argc[]) {
    int i;
    #pragma omp parallel num_threads(4) allocate(i)
    {
        int thread_id = omp_get_thread_num();
        int num_of_threads = omp_get_num_threads();
    }
    return(0);
}
```

NOTE: It is important to run **make** on $ROSE_BUILD directory after you make a few changes. This will tell you if your changes are successful or have bugs.

Testing and debugging.

1. Modify Makefile.am in the source tree tests/nonsmoke/functional/CompileTests/OpenMP_tests by adding the source code for test. e.g. add `parallel_allocate.c` for testing parallel allocate.
2. Create parallel_allocate.c file in tests/nonsmoke/functional/CompileTests/OpenMP_tests of the source tree
3. In the build tree, tests/nonsmoke/functional/CompileTests/OpenMP_tests, do `make rose_parallel_allocate.c`. If you do `make`, I believe it will test everything.

```
chrisogonas@pivo:~/rose/rose-develop-
build/tests/nonsmoke/functional/CompileTests/OpenMP_tests$ make
rose_parallel_allocate.c
./parseOmp -rose:openmp:ast_only --edg:no_warnings -w -rose:verbose 0
--edg:restrict  -I/home/chrisogonas/rose/rose-
develop/src/frontend/SageIII -c /home/chrisogonas/rose/rose-
develop/tests/nonsmoke/functional/CompileTests/OpenMP_tests/parallel_a
llocate.c
```

Modifying ROSE Compiler

```
lt-parseOmp: /home/chrisogonas/rose/rose-
develop/src/backend/unparser/languageIndependenceSupport/unparseLangua
geIndependentConstructs.C:6703: virtual void
UnparseLanguageIndependentConstructs::unparseOmpVariablesClause(SgOmpC
lause*, SgUnparse_Info&): Assertion `clause != NULL' failed.
Makefile:1903: recipe for target 'rose_parallel_allocate.c' failed
make: *** [rose_parallel_allocate.c] Aborted (core dumped)
make: *** Deleting file 'rose_parallel_allocate.c'
```

1. parseOmp does the identityTranslator thing so you can debug using either parseOmp or identityTranslator, but make sure you pass the same arguments to identityTranslator as parseOmp.
2. For debugging, you need to debug the binary file. The parseOmp and identityTranslator, and lots of other ROSE executable are all scripts, the corresponding binary files are in the .libs folder of where those scripts are in the build tree.

Illustration

```
chrisogonas@pivo:~/rose/rose-develop-
build/tests/nonsmoke/functional/CompileTests/OpenMP_tests$
../../../../../tutorial/identityTranslator -rose:openmp:ast_only --
edg:no_warnings -w -rose:verbose 0 --edg:restrict ~/rose/rose-
develop/tests/nonsmoke/functional/CompileTests/OpenMP_tests/parallel_allocate
.c
lt-identityTranslator: /home/chrisogonas/rose/rose-
develop/src/backend/unparser/languageIndependenceSupport/unparseLanguageIndep
endentConstructs.C:6703: virtual void
UnparseLanguageIndependentConstructs::unparseOmpVariablesClause(SgOmpClause*,
SgUnparse_Info&): Assertion `clause != NULL' failed.
Aborted (core dumped)
```

- You can also run the direct tests regardless of the location of test file location. See example below for file allocateTest.c

$ROSE_BUILD/tutorial/identityTranslator -c allocateTest.c

The above will generate a file with **rose**_ appended to the name as below.

rose_allocateTest.c

Modifying ROSE Compiler

- To generate an AST graph, see illustration below:

$ROSE_BUILD/exampleTranslators/DOTGenerator/dotGeneratorWholeASTGraph -rose:OpenMP:ast_only allocateTest.c

This will generate a file with the _WholeAST.dot appended to the file e.g. for allocateTest.c file, the AST output file generated is named:

allocateTest.c_WholeAST.dot

- Use **GraphViz** program to view the graph from **dotGeneratorWholeASTGraph**. This file can be very large even for small programs.

Note that **$ROSE_BUILD** refers to the ROSE build directory.