UPC2Promela

Amin M. Alipour
malipour@mtu.edu

Department of Computer Science
Michigan Technological University

August 25, 2010
Outline

Goal

UPC Constructs

Transformations

Tools

Current implementation

Interface

Status
Goal

Main Goal
Model Checking of UPC programs.
- Transforming UPC program $P$ to Promela program $P'$,
- Verification of synchronization skeleton of program $P'$, and
- Visualization of fault-inducing trace of $P'$ in $P$.

Challenges
- Transforming UPC constructs to Promela,
- Memory consistency mode of the program,
- Pointer manipulations
UPC Constructs

UPC is an extension of C, with parallelism. It includes:

**Data management constructs**

- Shared data types:
- Shared data manipulation:

**Inter-Thread Synchronization**

- Standard Constructs
- Library Support

**Collectives**
UPC to Promela Transformation

Generally we use the blocking conditions in Promela to block a thread.

Steps

- Pre-processing of original program
- Parse UPC to build the parse tree
- Transform program
  - Transform statement in UPC to statements in Promela which preserves the semantics,
  - Transform variables in UPC to data variables in Promela s.t. preserves the memory model and data location\(^1\).

---

\(^1\)Data location means conformance between shared, global and local variables in programs
Tools

Tools for model checking the source code

- Major trend is reusing the compiler generated intermediate language.

Potential tools which has been tried

- UNRAVEL: C slicer, it does not keep the Abstract Syntax Tree (AST) in memory.
- MODEX: C static analysis tool which extensive use of structures and pointers, which is hard to comprehend.
- UPC parser of Eclipse: parser of UPC in Java.
- cTool: C parser with OO design. Simpler to comprehend comparing to above tools.

None of them has been documented!
How the current implementation works

cTool

Shared Library cTool

Interface
Project *prj = new Project();
prj->SetPrintOption(debug);
for (i=0; i < nf; i++)
{
    TransUnit *unit = prj->parse(file_list[i], ...);
    if (unit)
    {
        PromelaPrintTraversal pt(std::cout, debug);
        pt.traverse_unit(unit);
        std::cout << std::endl;
    }
    else
    {
        std::cout << "trans unit for " << file_list[i] << " is NULL." << std::endl;
    }
}

File Location: /example/ctdemo.cpp
Basic Classes

▶ Each statement type has its own class like `WhileStatement` all of them inherit from `Statement`
▶ There is an umbrella class `Type` which all data type inherits from it.
▶ Vector of declarations are kept in `decl.h`

Utility classes

▶ Like `PromelaPrintTraversal`

Files Locations: *.h in `/include/ctool`, *.cpp in `/src`
A sample transformation in PromelaPrintTraversal

```cpp
void PromelaPrintTraversal::traverse_while(WhileStemnt *node)
{
    out<<"do\n";
    out << " ::(");
    node->cond->accept(this);
    out << " )->\n";
    block(node->block);
    out<<"\n:: else -> break;\n";
    out<<"od";
}
```
Project Status

Tasks

- Parse the UPC Code
- Augmenting declaration
- Translate it to Promela
- Counter-Example Visualization

Status in August 25, 2010

- Partially
- Not yet!
- C constructs done. Partial UPC constructs.
- Not yet!