Visualization of C++ Template Metaprograms

Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb

Dept. Programming Languages and Compilers
Eötvös Loránd University, Budapest, Hungary

SCAM 2010

1TÁMOP-4.2.1/B-09/1/KMR-2010-0003
C++ Templates

- Different from Java / C# generics
  - Java / C#: type erasure
  - C++: instantiation
- Mainly used for libraries: STL, etc.
- Templates are skeletons, code generated on demand
- Possibility for specialisation
- Recursive templates are ok
C++ Template Metaprogram - example

template <int N>
struct Factorial
{
    enum { Value = N * Factorial<N-1>::Value};
};
template <>
struct Factorial<0>
{
    enum { Value = 1 }; 
};

int main()
{
    std::cout << Factorial<5>::Value;
}

Visualization of C++ Template Metaprograms Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
C++ Template Metaprogram features

- Executed at compilation-time
- Functional paradigm
- Why we used them:
  - optimizations of runtime programs, expression templates
  - static interface checking, concept checking
  - compile-time code adoption, active libraries
  - embedding DSLs
- Turing complete
Motivation

- Metaprogramming is side effect of template construct
- Template syntax is not helpful
- Compiler interprets metaprograms at compilation-time
- No user input, trivial printouts, etc.
- Maintenance is hopeless
Motivation

- Metaprogramming is side effect of template construct
- Template syntax is not helpful
- Compiler interprets metaprograms at compilation-time
- No user input, trivial printouts, etc.
- Maintenance is hopeless

C++ template metaprogram code comprehension tools are essential
Templight

- Lightweight parser using boost wave and spirit
- Instruments template classes/functions injecting begin/end markers
- Markers emit compilation warnings on instantiation
- Collects warnings generating a "stack-trace"
- Post-mortem way
- Take advantage of compiler dependent implementation details (e.g. memoization)
Debugger

- Based on Templight
- GUI is based on QT
- Implements "usual" debugger features:
  - Breakpoints, continue
  - Step in/out/over
  - Locals, watch
- Backward execution
Visualization of C++ Template Metaprograms
Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
Based on Templight

Transform the instantiation chain into a directed graph:
- nodes: types generated from templates
- edges show the instantiation requests

Show corresponding code

Filter out irrelevant nodes

Export to png, jpg etc,
Unruh Example Demonstration
Unruh Example Demonstration
Unruh Example Demonstration
Unruh Example Demonstration

Visualization of C++ Template Metaprograms  Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
Unruh Example Demonstration

Visualization of C++ Template Metaprograms  Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
Unruh Example Demonstration
Unruh Example Demonstration
Unruh Example Demonstration

Visualization of C++ Template Metaprograms
Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
Unruh Example Demonstration

Visualization of C++ Template Metaprograms  Zoltán Borók-Nagy, Viktor Májer, József Mihalicza, Norbert Pataki, Zoltán Porkoláb
Unruh Example Demonstration
Conclusion

- It is hard to understand and maintain C++ template metaprograms
- Visualization of programs is essential
- We have created a basic framework called *Templight*
- We have developed a graphical user interfaced post-mortem debugger
- We have implemented a tool to visualize the C++ template metaprograms as graphs
template <int p, int i>
struct is_prime {
    enum {
        prim = (p==2) ||
               (p%i) &&
               is_prime<(i>2?p:0),i-1>::prim
    };
};

template<>
struct is_prime<0,0> {
    enum {prim=1};
};

template<>
struct is_prime<0,1> {
    enum {prim=1};
};
C++ source is the assembly of template metaprogram.

We have to use high level functional programming languages, like Haskell, to write metaprograms, and generate C++ source.
Controversial

C++ source is the assembly of template metaprogram.

We have to use high level functional programming languages, like Haskell, to write metaprograms, and *generate* C++ source.

Thank you for attention