Clang Template Specialization Resugaring

LLVM Dev Meeting 2022

GSoC 2022

Matheus Izvekov
<mizvekov@gmail.com>

Mentors:

Vassil Vassilev <v.g.vassilev@gmail.com>

https://compiler-research.org/

Richard Smith <richard@metafoo.co.uk>

Problem

Template instantiations are not affected by *type sugar* in the template arguments.

Infamous example:

```
template <class T> auto foo(T) -> T;
int x = foo(std::string("hello"));
```

Produces diagnostic:

```
error: no viable conversion from 'std::basic_string<char>' to 'int'
```

Proposal

When performing member access on template specializations, propagate the type sugar in the specialization arguments into the accessed type.

```
template <class T> struct Foo {
  using type = T;
};

using Int = int;

using bar = typename Foo<Int>::type; // Propagate 'Int' into 'type'.
```

The Basics of Resugaring

We leverage the information provided by the Subst* node.

When we encounter a member access into a template specialization, such as:

```
Foo<Int>::type
```

We take note that `Int` was used as the argument to the parameter `T` of the `Foo` class template.

Transforming

When traversing the AST for `type`, replace any *Subst* nodes referring into `T` with the argument used to name the specialization, `Int`:

```
TypeAlias 'type'
`-TypedefType 'Int' sugar
  |-TypeAlias 'Int'
  `-BuiltinType 'int'
```

Constraints

We want the implementation to impose few impediments to it's use:

- Avoid different modes: Always perform resugaring.
- Avoid introducing new specific AST nodes.
- No changes to matchers or other APIs.
- Reasonable cost and no surprises.

Status of the Implementation

We have the 'resugar' Clang branch on compiler explorer.

Functionality has been quite advanced for a while.

Many type sugar preservation changes / improvements have been merged into Clang.

Most of the work now is on performance aspects.

We are aiming for an always enabled, eager resugaring.

Performance

For most workloads, difference in performance is within noise levels.

We have taken some benchmarks on llvm-compile-time-tracker:

NewPM-O3	Change
kimwitu++	+0.89%
Bullet	+0.21%
tramp3d-v4	+1.27%
7zip	+1.40%
geomean	+0.39%

No changes: sqlite3, consumer-typeset, mafft, ClamAV, lencod, SPASS

Conclusion

We are almost there!
Last few patches are up for review, but some work remains on enablers.

Bugs and missing functionality:

- Resugaring is not working well in libstdc++, but does in libc++.
- We can't resugar over accesses to base classes yet.
- More work on enablers and performance.

The End