Phoenix: PDL and Compile-time Reflection

Drew Hoskins
(ahoskins@microsoft.com)
Agenda

- What is Phoenix?
- PDL
- Visitor Pattern
- Object Cloning
- Compile-Time Reflection
- Phoenix Links
What is Phoenix?

Phoenix is Microsoft’s next-generation, state of the art infrastructure for program analysis and transformation.

Phoenix is used for:
- Compilers
- Tools
- Research
- Instruction
What is Phoenix?

- Example Clients
  - Executable Reader/Writer
  - C++ Compiler backend
  - PreFAST (static analysis)
  - JIT
  - NGEN
  - Many more (tiger, pasm, FxCop, AST, VSInstr)

- Example Compiler Backend Plug-Ins
  - Static Analysis, Arithmetic strength reduction, Analysis Validator, Instrumented Block Count reader, ...
PDL

PDL is a C++/CLI-like language for headers. It evolved from a solution for dual-mode C++ compilation to much more.

Interesting Features:

- Metaprogramming attributes on objects and members
- Partial classes
- Weaving
- Bitpacking
- Dual Mode

We’ll look further into the green ones.
PDL to C++

a.pdl → b.pdl → c.pdl → whole program

Reflection, Weaving

C++ headers

C++ sources

~3.5 MB

~45 MB
PDL - Attributes

- **Class**
  - Layout
  - Align
  - Extensible
  - FreeList
  - **Cloneable**
  - Dynamic
  - Visit
  - Kind

- **Element**
  - Sealed
  - Thunk
  - NoZeroInit
  - NoReinit
  - Ordered
  - List
  - **NoClone**
  - Owner
  - Context
  - CustomFixup
Example: Rectangle

```c++
public ref class Rectangle
 : public class Object
{
public:
    int HorizontalLength;
    int VerticalLength;
};
```
Rectangle: Visitor Weaving

```csharp
public ref class Rectangle;

// Auto-generated
void Rectangle::Accept(Visitor ^ visitor)
{
    visitor->Visit(this);
}

void Visitor::Visit(Rectangle ^ rectangle)
{
    // Default method: Do nothing or throw
}
```
Visitor Pattern

Good

- Decouple unwanted/unpredicted concerns from basic classes.
- Implement distributed algorithm in one place.
- Visitor hierarchy good way to provide small variations on an algorithm.

Bad

- Not extensible cross-assembly.
- Must maintain visitors when adding new classes or fields.
Rectangle: Cloning Visitor

public ref class Rectangle;

// Auto-generated
void CloneVisitor::Visit(Rectangle ^ rect)
{
    if (Map[rect] != NULL) {
        Result = Map[rect];
    } else {
        Rectangle ^ newRect = gcnew Rectangle;
        newRect->HorizontalLength = rect->HorizontalLength;
        newRect->VerticalLength = rect->VerticalLength;
        Result = Map[rect] = newRect;
    }
}
Rectangle: Class Updates

[visit, cloneable]
public ref class Rectangle
 : public class Object
{
 public:

 Color Shade;
 int HorizontalLength;
 int VerticalLength;

};

// Clone Visitor automatically
// updates
Rectangle: Cloning Issues

[visit, cloneable]
public ref class Rectangle :
public class Object
{
public:
    Color Shade;
    int HorizontalLength;
    int VerticalLength;
    // What to do?
    CartesianGrid ^ Grid;
    Coordinate Position;
};
Rectangle: Cloning Issues

[visit, cloneable]
public ref class Rectangle
    : public class Object
{
public:
    Color Shade;
    int HorizontalLength;
    int VerticalLength;
    // Try 1
    [NoClone] CartesianGrid ^ Grid;
    [NoClone]
    Coordinate GridPosition;
};
Rectangle: Cloning Issues

[visit, cloneable]

public ref class Rectangle:
    : public class Object
{
    public:
        Color Shade;
        int HorizontalLength;
        int VerticalLength;

    // Try 2
    [Clone(Owner)] CartesianGrid ^ Grid;
    [Clone(Context="Grid")]
    Coordinate GridPosition;
};
Example: Cloning Phoenix’s Intermediate Representation

An **Operand** might be cloned

- By itself
- As part of an instruction
- As part of a Function

The more context, the more is cloned. “Context” attribute helped share code, and neatly encapsulated a lot of complex logic
Compile-time reflection

- Used to generate object walker
- Walker useful for any algorithms that need to walk all or most of objects’ fields.
  - Cloning
  - Serialization/deserialization
  - Comparison
  - Diagnostic Dumper
Phoenix Links

- Early access RDKs or CDKs available to selected universities or commercial partners; sample projects include AOP, Obfuscation, and Profiling
- Contact phxap@microsoft.com for Academic early access requests, or phxcp@microsoft.com for Commercial early access requests. No NDA must be signed.
- Or see http://research.microsoft.com/phoenix
- Come to the Phoenix discussion on Tuesday!