

Nemerle

Michał Moskal

Computer Science Institute, University of Wrocław, Poland
Second ROTOR Workshop 2005, Redmond, WA

September 20, 2005

What's that?

- high-level, statically typed programming language
 - object-oriented, functional, imperative
 - type inference
 - pattern matching
 - Turing-complete macros – compiler plugins extending the language
- since the beginning designed for the .NET

Why a new language, from a user perspective?

- combining object-oriented and functional programming
 - familiar, C#-like, object-oriented top-level program structure (classes, interfaces)
 - methods can be implemented in a functional style
 - easy access to imperative features
 - start: looks like ML, has a subset of C# features + functional stuff
 - end: looks like C#, includes C# + functional stuff + ...
- macros!

Why a new language, from a computer scientist perspective?

- experimentation
 - meta-programming system
 - type inference algorithms
 - programming language design
 - .NET code generation using S.R.E. API
 - stress testing generics

And now something completely different:

Sample time!

Sample 1.0

```
class Hello
{
    public static Main () : void
    {
        System.Console.Write ("Hello world!\n");
    }
}
```

Sample 1.1

```
System.Console.WriteLine ("Hello world!\n");
```

Sample 2.0

```
class Factorial {  
    public static factorial (x : int) : int {  
        def loop (acc : int, x : int) : int {  
            if (x <= 1) acc  
            else loop (acc * x, x - 1)  
        }  
  
        loop (1, x)  
    }  
  
    public static Main () : void  
    {  
        System.Console.WriteLine (factorial (10));  
    }  
}
```

Sample 2.1

```
class Factorial {  
    public static factorial (x : int) : int {  
        def loop (acc, x) {  
            if (x <= 1) acc  
            else loop (acc * x, x - 1)  
        }  
  
        loop (1, x)  
    }  
  
    public static Main () : void  
    {  
        System.Console.WriteLine (factorial (10));  
    }  
}
```

Sample 3.0

```
[Record]
class Foo {
    my_value : int;
    public Barize () : void
    { System.Console.Write ($ "Foo ($my_value), "
                            "times 42 = $(my_value * 42)\n");
    }
}

class Qux { public Barize () : void { } }

// Main starts here
def call_barize (x) { x.Barize () }
def our_list = [Foo (1), Foo (2), Foo (3)];
foreach (e in our_list)
    call_barize (e);
```

Sample 3.1

```
[Record]
class Foo {
  my_value : int;
  public Barize () : void
  { System.Console.Write ($ "Foo ($my_value), "
                          "times 42 = $(my_value * 42)\n");
  }
}

class Qux { public Barize () : void { } }

// Main starts here
def call_barize (x) { x.Barize () }
def our_list = [Foo (1), Foo (2), Foo (3)];
// foreach (e in our_list)
//   call_barize (e);
```

Macros

- written in Nemerle
- dynamically loaded compiler modules (no connection with CPP!)
- transform, generate and analyse programs
- can extend syntax of the language
- can interact with type inference
- work on syntax trees of expressions and types
- can read external files, query database etc.

Uses of macros

- specialized sublanguages (printf, scanf, regular expressions, SQL, XML, XPath)
- generating trees from other trees (serialization, specialization of code)
- domain specific optimization
- assertion system
- automatic creation of repeatable class level patterns
- *Aspect Oriented Programming*

Example use of macro

This macro checks syntax and type validity of a query at compile-time (by connecting to a database). It creates code, which uses `SqlParameter` to pass value of *myval* to `SqlCommand` securely.

```
def myval = "Kate";  
sqlloop ("SELECT * FROM employee WHERE"  
        "  firstname = $myval", dbcon)  
{  
    printf ("%s %s\n", firstname, lastname)  
}
```

Compiler

- bootstrapping
- generates and consumes generics
- 0.9.0 release, for August CTP and Mono 1.1.9

Projects using Nemerle

- Sioux – HTTP/application server (founded from the grant)
- cs2n – C# to Nemerle converter (founded from the grant)
- nemish – Nemerle Interactive Shell
- Asper IDE/editor
- RiDL parsing/lexing tools
- NAnt build system plugin
- CodeDom generator (ASP.NET support)
- Code Completion Engine
- IDE integration (VS.NET, #D, MD)

Impact and community

- .NET runtime issues:
 - several SRE issues reported to MSDN Product Feedback and one serious issue with static field initialization in generic classes
 - even more issues reported to the Mono team (the runtime is much less mature)
- mailing list (70 subscribers)
- web forum (recently started)
- issue tracker – 500 issues total, 50 still open
- wiki-based webpage – external documentation writers/fixers
- online course will start around October 1st
- completed regular course at our Institute

Performance results

- interested in performance since compiler bootstrap
- tail calls
 - the `tail.` prefix makes the call go much slower
 - use jumps where possible
 - make `tail.` generation default off
- delegates vs functional objects – delegates are much slower
- boxing vs generics – generics seems faster, but hard to compare

TODO

- 1.0 stable release approaching
- more IDE integration
- more community building
- more static analysis

More info

<http://nemerle.org/>

Questions?

Thank you!

Thank you for your attention!