Coco/R

Compiler Compiler for C#, ...

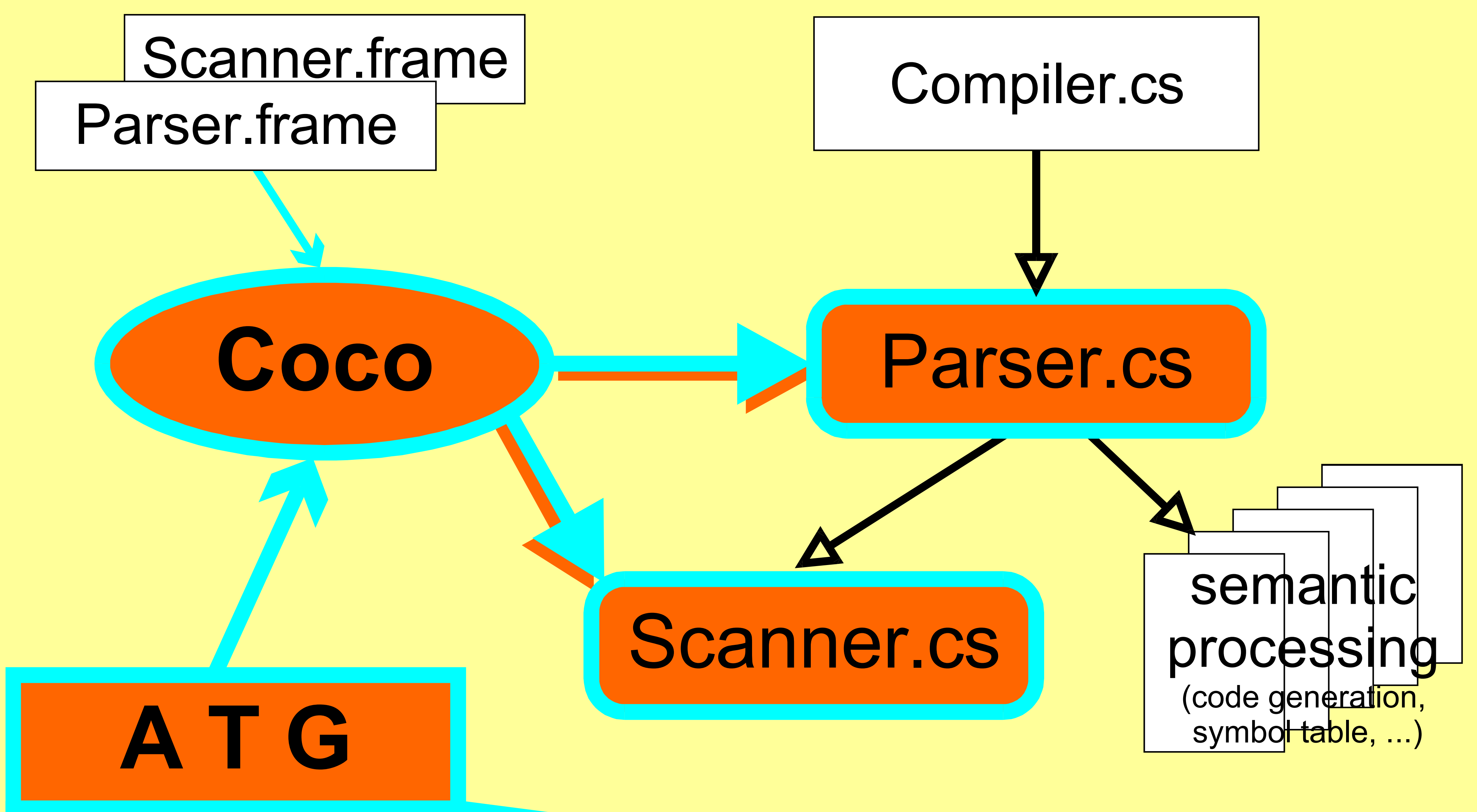
Coco/R is a compiler generator which takes a compiler description in the form of an attributed EBNF grammar (ATG) and generates the scanner & recursive descent parser for the described language.

available at

www.ssw.uni-linz.ac.at/Research/Projects/Coco

Interfaces of Generated Types:

```
class Scanner {
    static void Init(string sourceFile);
    static Token Scan();
}
class Parser {
    static Token token;     // last recognized token
    static Token t          // lookahead token
    static void Parse();
}
class Token {
    int kind;
    string val;
    int pos;
    int line;
    int col;
}
class Buffer {
    static void Fill(string sourceFile);
    static int Read();
    static int Pos;
}
delegate void ErrorProc(int n, int line, int col);
class Errors {
    static int count;
    static ErrorProc SynErr;
    static ErrorProc SemErr;
    static void Exception (string s);
}
```



```
using System;

COMPILER Expression                         arbitrary C# declarations
    public static int value;
                                                scanner specification
CHARACTERS  digit = "0123456789" .
TOKENS      number = digit { digit } .

PRODUCTIONS
Expression = Expr<out value> .

Expr<out int val>                       (. int val1, sign; .)
=   Term<out val>         attributes                      sematic actions
    {  ( "+"                            (. sign = 1; .)
       | "-"                            (. sign = -1; .)
       ) Term<out val1>                 (. val += val1 * sign; .)
    } .

Term<out int val>                       (. int val1; bool mul; .)
=   Factor<out val>
    {  ( "*"                            (. mul = true; .)
       | "/"                            (. mul = false; .)
       ) Factor<out val1>               (. val = (mul) ? val * val1 : val / val1; .)
    } .

Factor<out int val>                     (. int val1; val = 1; .)
=   ["-"                                (.val = -1; .)
    ]  ( number                         (. val *= Convert.ToInt32(token.val); .)
       | "(" Expr<out val1> ")"         (. val *= val1; .)
       ) .

END Expression.
```

Compiler.cs:

```
using System;
namespace Expression {
    public class Compiler {
        public static void Main (string[] args) {
            if (args.Length > 0) {
                Scanner.Init(args[0]);
                Parser.Parse();
                if (Errors.count == 0) Console.WriteLine(Parser.value);
            } else Console.WriteLine("No source file specified\nusage: evaluate <file>");
        }
    }
}
```

```
> Coco.exe Expr.ATG
> csc.exe /out:evaluate.exe Compiler.cs Parser.cs Scanner.cs
```

Valid.expr:

```
-120/10 + 2*(9/3)
```

```
> evaluate Valid.expr
-6
```

Invalid.expr:

```
2 * ((5) + 8
```

```
> evaluate Invalid.expr
-- line 1 col 15: ')' expected
```

Compiler Generation Tools for C#

GOALS

Extension of Coco/R

The Coco-generated parser shall be able to use semantic information in order to resolve LL(1)-conflicts. Thus Coco/R can be applied to non-LL(1)-grammars (such as C# 's).

ATG template for C#

This forms a framework with complete parsing facilities for C# programs. By adding specific attributes and semantic actions to the template concrete applications can be instantiated.

?

GUI for template modifications

By providing a graphical user interface that supports and controls the modifications of the template even less experienced users shall be empowered to create their own compiler tools.

APPLICATIONS

- Instrumenting C# programs by inserting, deleting or rewriting code fragments, thus creating new runtime behavior, e.g. profiling, testing or debugging output.
- Analyzing C# source code to extract arbitrary information, e.g. complexity or style measures, call graphs, …
- Translating C# source code into any arbitrary format, e.g. Java-Bytecode, syntax-highlighted HTML version, …

Any ideas or wishes for conrete applications?
Please visit our project homepage and contact us!

Pilot Project using the Shared Source Implementation of CLI and C#

http://dotnet.jku.at/Projects/Rotor