

Albrecht Wöß  
Hanspeter Mössenböck  
Markus Löberbauer  
Johannes Kepler University Linz, Austria  
SystemSoftWare Group

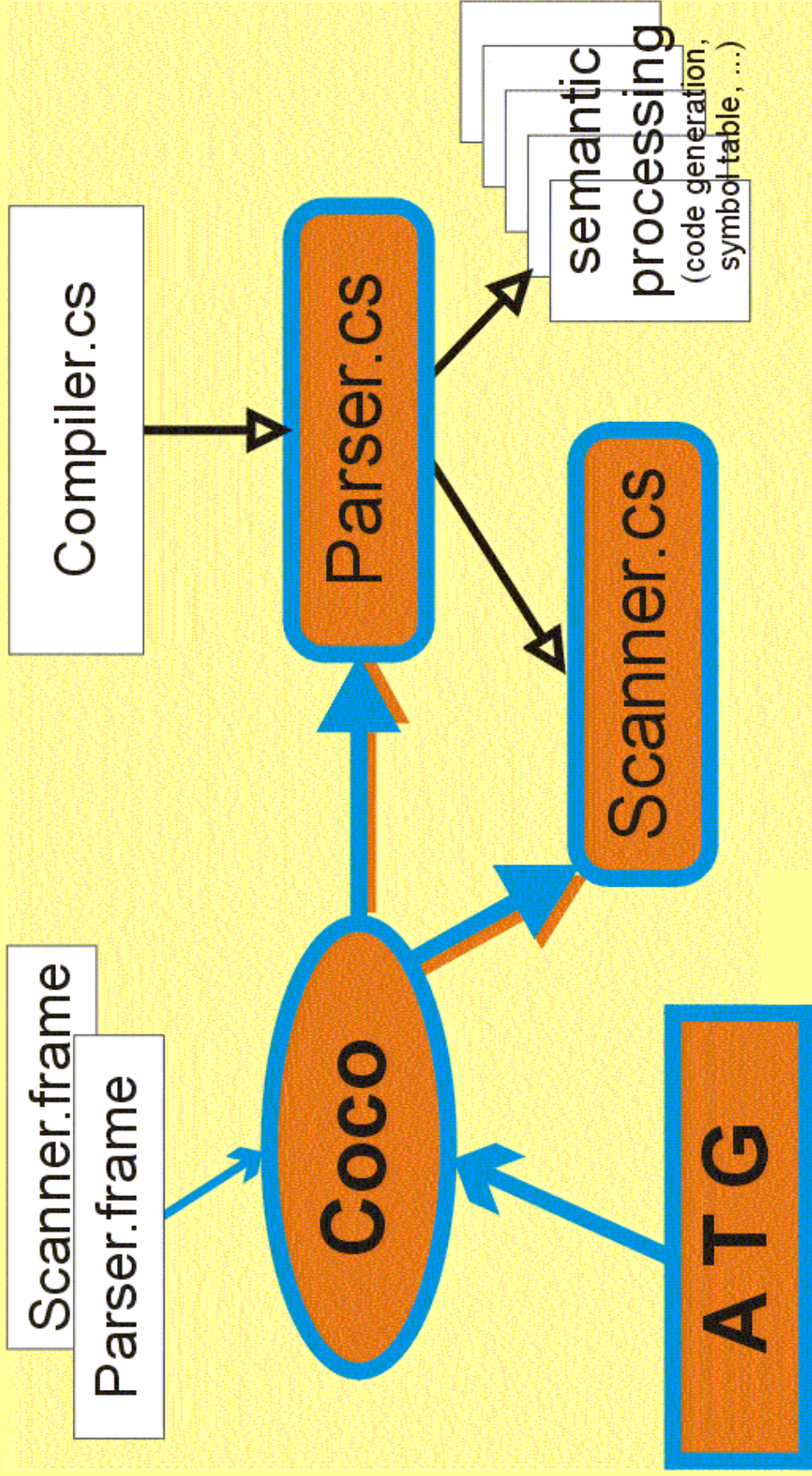
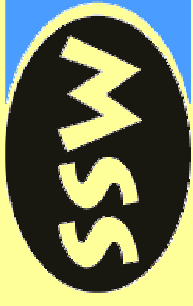
# Compiler Generation Tools for C#

# What's Coco/R? What's the Project?



- *Coco/R*:
  - is an **easy-to-use** compiler generator
  - *Input*: attributed, context-free EBNF grammar
  - *Output*: **scanner & recursive descent parser**
- "*Compiler Generation Tools*" Project:
  - Extension of Coco/R
    - to use Coco/R for non-LL(1) grammars
    - to use semantic information to resolve LL(1) conflicts
  - ATG template for C#
    - provides complete parsing facilities for C# programs
    - add attributes and semantic actions for customized applications

# Architecture of Coco/R



# Coco/R 2.0b - What's new?



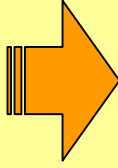
- **LL(1) conflict resolution**
  - new keyword: **IF** ( *boolean expression* )
  - peek functionality in Scanner
- user-defined names for tokens
  - new keyword: **TOKENNAMES**
- user-defined namespace for generated types
  - new keyword: **NAMESPACE**

# NAMESPACE



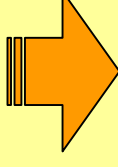
*CSharp.atg*

```
using System.Collections;
NAMESPACE at.jku.ssw.Coco
COMPILER CS
...
```



*Scanner.cs*

```
using System.Collection;
namespace at.jku.ssw.Coco {
    public class Token {...}
    public class Buffer {...}
    public class Scanner {...}
    public class Errors {...}
}
```



*Parser.cs*

```
using System.Collections;
namespace at.jku.ssw.Coco {
    public class Tokens {...}
    public class Parser {...}
}
```

# LL(1) Conflicts



- LL(1) conflicts:
  - explicit alternatives: e.g.  $A = a \ b \mid a \ c.$
  - options: e.g.  $A = [ a \ ] \ a \ b.$
  - iterations: e.g.  $A = \{ a \ } \ a \ b.$
- Conflict resolution:
  - factorization:
    - e.g.  $A = ab \mid ac.$   $\Rightarrow$   $A = a ( b \mid c ).$
  - transform left recursion into iteration:
    - e.g.  $A = Ab \mid c.$   $\Rightarrow$   $A = c \{ b \}.$

# LL(1) Conflicts (cont.)



- Conflict resolution (cont.):
  - but sometimes transformations
    - **do not help**  
IdentList = ident { **" , "** ident } [ **" , "** ].
    - **are undesirable**  
UsingClause = "using" [ ident "=" ] Qualident ";" .  
Qualident = ident { "." ident } .
  - UsingClause = "using" ident ( { "." ident }  
| "=" Qualident  
) ";" .

# Conflict Resolvers



*CSharp.atg*

```
COMPILER CS
...
static bool NotFinalComma () {
    Scanner.StartPeek();
    return la.kind == Tokens.Comma &&
           Scanner.Peek().kind == Tokens.Ident;
}
```

```
...
PRODUCTIONS
...
IdentList { IF (NotFinalComma())
            " , " ident
            [ [ " , " ] .
            ...
```

*in Parser.cs*

```
static void IdentList () {
    Expect(ident);
    while (NotFinalComma()) {
        Expect(" , ");
        Expect(ident);
    }
    if (la.kind == " , ") Get();
}
```



# Peek functionality



*CSharp.atg*

```
COMPILER CS
...
static bool NotFinalComma () {
    Scanner.StartPeek();
    return la.kind == Tokens.Comma &&
           scanner.Peek().kind == Tokens.Ident;
}
...
```

Scanner Interface:

```
public static Token Scan ()
    ... returns next token, moves current token & peek token ahead
public static Token Peek ()
    ... returns next peek token, moves peek token ahead (not current token)
public static Token StartPeek ()
    ... makes sure peeking starts from current token
```

# TOKENNAMES



*CSharp.atg*

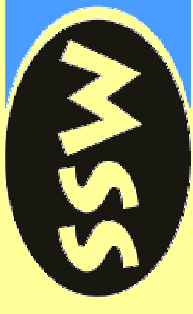
```
COMPILER CS
...
static bool NotFinalComma () {
    Scanner.StartPeek();
    return la.kind == Tokens.Comma &&
           Scanner.Peek().kind == Tokens.Ident;
}
```

```
...
TOKENNAMES
    "," = Comma
    ident = Ident
    "(" = LPar
    ")" = RPar
    ...
```

*in Parser.cs*

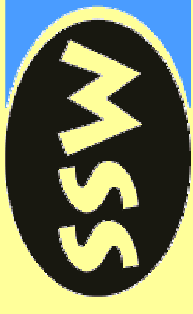
```
public class Tokens () {
    public const int Comma = 2;
    public const int Ident = 1;
    public const int LPar = 3;
    public const int RPar = 4;
    ...
}
```

# Applications (on-going projects)



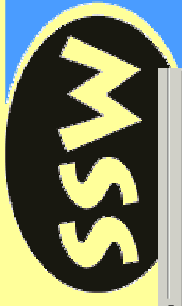
- White-Box-Testing Tool
- CodeDOM-Parser (System.CodeDom.Compiler.CodeParser)
- C#-Source-Code-Analyzer
  - gathers static information about a program
    - Lines of Code (with / without comments)
    - Number of instructions, selections, loops, method calls, ...
    - Complexity measures: McCabe, Rechenberg, ...
    - ...
- Online-C#-Syntaxhighlighter (produces HTML)
  - at <http://dotnet.jku.at/applications/cs2html>
  - major update coming soon!

# White-Box-Testing Tool



- supports testing by measuring and visualizing **path coverage** of test runs
- client/server architecture via HTTP
  - client: WBT tool
  - server: instrumented program under test
- instruments C# source code (*with Coco/R generated parser*) to log covered paths
- visualizes the paths in
  - a control flow graph
  - the source code

# WBT Tool (highlighted path in CFG)



SourceView | Text | Tree | Browser

**T** if **F** | **T** Balance **F** | **T** if **F** | **T** switch **F** | **T** Balance **F** | **T** switch **F** | **T** Balance **F** | **T** for **F** | **T** foreach **F** | **T** while **F**

**T** if **F** | **T** switch **F** | **T** Balance **F** | **T** switch **F** | **T** Balance **F** | **T** switch **F** | **T** Balance **F** | **T** switch **F**

Tool'n'Info

- Hallo.Halovet.Halovet0[118]; 7; 10
- Hallo.Halovet.Halovet1[173]; 11; 14
- Hallo.Halovet.Man2[611]; 41; 22

<input type="checkbox"/>	00000000	(0)
<input type="checkbox"/>	00000001	(1)
<input type="checkbox"/>	00000010	(2)
<input type="checkbox"/>	00000011	(3)
<input type="checkbox"/>	00000100	(4)
<input type="checkbox"/>	00000101	(5)
<input type="checkbox"/>	00000110	(6)
<input type="checkbox"/>	00000111	(7)
<input type="checkbox"/>	00001000	(8)
<input type="checkbox"/>	00001001	(9)
<input type="checkbox"/>	00001010	(10)
<input type="checkbox"/>	00001011	(11)
<input type="checkbox"/>	00001000	(16)
<input type="checkbox"/>	00001001	(17)
<input type="checkbox"/>	00001010	(18)
<input type="checkbox"/>	00001011	(19)
<input type="checkbox"/>	00010000	(32)
<input type="checkbox"/>	00010001	(33)
<input type="checkbox"/>	00010010	(34)
<input type="checkbox"/>	00010011	(35)
<input type="checkbox"/>	00010100	(36)
<input type="checkbox"/>	00010101	(37)
<input type="checkbox"/>	00010110	(38)
<input type="checkbox"/>	00010111	(39)
<input type="checkbox"/>	00100000	(64)
<input type="checkbox"/>	00100001	(65)
<input type="checkbox"/>	00100010	(66)
<input type="checkbox"/>	00100011	(67)
<input type="checkbox"/>	00100100	(68)
<input type="checkbox"/>	00100101	(69)
<input type="checkbox"/>	00100110	(70)
<input type="checkbox"/>	00100111	(71)
<input type="checkbox"/>	00100100	(72)
<input checked="" type="checkbox"/>	00100101	(73)
<input type="checkbox"/>	00100110	(74)
<input type="checkbox"/>	00100111	(75)
<input type="checkbox"/>	00101000	(80)
<input type="checkbox"/>	00101001	(81)
<input type="checkbox"/>	00101010	(82)
<input type="checkbox"/>	00101011	(83)
<input type="checkbox"/>	00110000	(96)
<input type="checkbox"/>	00110001	(97)
<input type="checkbox"/>	00110010	(98)
<input type="checkbox"/>	00110011	(99)
<input type="checkbox"/>	00110010	(100)
<input type="checkbox"/>	00110011	(101)

# WBT Tool (highlighted path in code)



SourceView

Text Tree Browser

```
public int Hallo() {  
    if (a && b) {  
        Console.WriteLine("Hallo Welt!");  
    }  
    else if (a || d) {  
    }  
    if (b && d) {  
    }  
    }  
    switch (i) {  
        case 1 : case 2 : for (int j=0;  
        case 3 : break;  
        case 4 : break;  
        default : break;  
    }  
    foreach(char c in "test") {  
        Console.WriteLine(c);  
    }  
    while (i<23) {  
        i++;  
    }  
    return i;  
}
```

Methods 'n' Paths

- Hallo.HelloWelt.HelloWelt(0)[119]: 7, 10
- Hallo.HelloWelt.Test(1)[182]: 12, 15
- Hallo.HelloWelt.HelloWelt(2)[270]: 20, 14
- Hallo.HelloWelt.Main(3)[703]: 50, 22

- 100000111 (263)
- 100001000 (264)
- 100001001 (265)
- 100001010 (266)
- 100001011 (267)
- 100010000 (272)
- 100010001 (273)
- 100010010 (274)
- 100010011 (275)
- 100100000 (288)
- 100100001 (289)
- 100100010 (290)
- 100100011 (291)
- 100100100 (292)
- 100100101 (293)
- 100100110 (294)
- 100100111 (295)
- 101000000 (320)
- 101000001 (321)
- 101000010 (322)
- 101000011 (323)

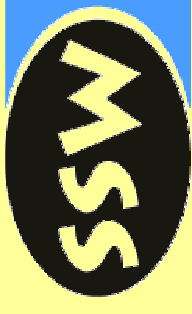
14

# Plans for WBT Tool



- use CodeDOM-Tree
  - as model for visualization
  - to do instrumentation there
- improve path visualization
- redesign GUI
- single process (instead of client/server architecture) to
  - increase speed
  - simplify usage

# Links



- Main Project Site:  
<http://dotnet.jku.at/Projects/Rotor>
- Rotor Community Site Project "cocotools":  
<http://cocotools.ssc.li.net>
- Questions, comments, ... ?

Thanks to  
Werner Vogels & Co.



# Screenshot: cocotools.sscli.net

## ROTOR

Shared Source Common Language Infrastructure

User:

Password:

[Register](#) | [Login help](#)

[My pages](#)

[Projects](#)

[Community](#)

[Projects](#) > [cocotools](#)

### Project tools

- [Project home](#)
- [Membership](#)
- [Mailing lists](#)
- [Source code](#)
- [Issue tracking](#)
- [File sharing](#)
- [News](#)

### How do I...

- [Get help?](#)

### Site search

## Project home

If you were [registered](#) and [logged in](#), you could join this project.

### Summary

Compiler Generation Tools for C#

### Categories

None

### Owner(s)

[alwoess](#)

### Description

We offer recursive descent compiler generator (Coco/R) with an extension to allow LL(1)-conflict resolution and a ready-made attributed grammar template for C# that only needs customization to create individual C# compiler tools. We are also working on a couple of - hopefully - useful tools ourselves.

This is one of the [Rotor Funded Projects](#).

Find more information at [our project page](#).

POWERED BY

**COLLABNET™**

[Site FAQ](#) | [Feedback](#) | [Terms of use](#) | [Developer tools](#)

Copyright © 1999-2003 CollabNet, Inc. CollabNet and SourceCast are trademarks of CollabNet, Inc. All other trademarks or registered trademarks are the property of their respective holders.