Design and implementation of the Meta Casanova 3 Compiler front-end

Jarno Holstein

June 30, 2016

Introduction

In this presentation

- Introduction
- Research questions
- Sub question
- Results
- Conclusion
- Demo

Introduction

Motive

- Video game development
- Casanova
- Meta Casanova
- Front-end of the compiler

Meta Casanova became usefull for more then game development

Research question

Main question

How to develop a maintainable and expandable front-end and type checker for the Meta Casanova programing language?

Research question

Requirements

- Correct
- Maintainable and expandable
- Descriptive error messages

Sub questions

- Syntactic propeties question
- Parser question
- Type system question

Syntactic propeties question

What properties does MC have that the front-end needs to process?

- Few keywords
- Every definition has a declaration
- Features can be devided in groups

Syntactic propeties question

Keywords

MC

- Func
- Data
- TypeFunc
- TypeAlias
- ->
- =>
- #>
- import
- inherit

C#

abstract ; as ; base ; bool ; break ; byte ; case ; catch ; char ; checked ; class ; const ; continue ; decimal ; default ; delegate ; do ; double ; else ; enum ; event ; explicit ; extern ; false ; finally ; fixed ; float ; for ; foreach ; goto ; if ; implicit ; in ; int ; interface ; internal ; is ; lock ; long ; namespace ; new ; null ; object ; operator ; out ; override ; params ; private ; protected ; public ; readonly ; ref ; return ; sbyte ; sealed ; short ; sizeof ; stackalloc ; static

Syntactic propeties question

Devided features

- Data and func
- Dotnet
- lambdas
- TypeAlias
- TypeFunc
- Module

Parser question

How to develop a maintainable and expandable parser for MC?

Parser question

What is a parser

- Takes a sequence of elements
- Builds a data structure
- Detect syntactic errors

Parser question

How to make a parser

- Parser generators
- Parser monad

Parser question

Parser generator

- Is a program
- Syntax description -> parser

Parser question

Parser generator

Pros

- Fast setup
- Fast parsing

Cons

- Difficult to generate helpful error messages

Parser question

What is a parser monad?

A parser monad iterates over a list and builds a output structure.
If the parser fails then instead of the context it will return an error.

Parser question

concat

1. t
2. w
3. o
4. #
5. w
6. o
7. r
8. d
9. s
10. #

ch = get-char
concat ch

1. two#words#

Parser question

check char

1. t
2. w
3. o
4. #
5. w
6. o
7. r
8. d
9. s
10. #

check-char #

1. nope
2. nope
3. nope
4. yep
5. nope
6. nope
7. nope
8. nope
9. nope
10. yep

Parser question

until

1. t
2. w
3. o
4. #
5. w
6. o
7. r
8. d
9. s
10. #

ch = get-char until
check-char #
concat ch

1. two
2. words

Parser question

Error handling within the parser monad

Concat all the errors

1. Slow
2. Get all the errors and thus none

Give only the last viable error

1. Fast
2. possibility to lose error information
3. possibility to get incorrect error information

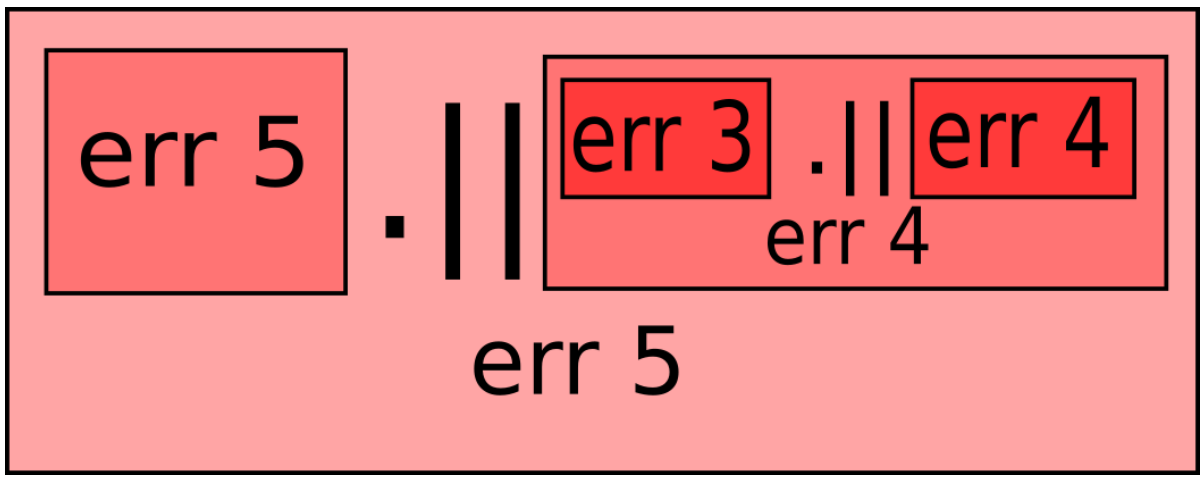
Priority for errors

1. Fast
2. More accurate error information
3. possibility to get incorrect error information

Parser question

Errors with priority

The error with the highest priority is the one that gets returned.



Parser question

parser monad

Pros

- Custom error system
- Powerful parser
- Easy to maintain and expand

Cons

- Slow setup
- Slow parsing

Parser question

Strategy

- Data and func
- Dotnet
- lambdas
- TypeAlias
- TypeFunc
- Module

- Minimal type checker
- DotNet types
- Type inference
- Kind support
- Compile time interpretation
- Complex inheritance system

Parser question

Modules

- Declarations TypeFunc and Alias
- Definitions TypeFunc and Alias
- Declarations Data and Function
- Definitions Rules

Type system question

How to apply type systems to MC?

Type system question

Type checker

- Normalized input and output
- Modular
- Separate type checker for run-time and compile time components

Type system question

Normalization

- Smaller type checker
- Less complexity in data structures

Type system question

Modular

Just like the parser

- Declarations TypeFunc and Alias
- Definitions TypeFunc and Alias
- Declarations Data and Function
- Definitions Rules

Type system question

Separate type checker for run-time and compile time components

- Helps getting the first prototype working faster
- Extra securety on type checking

Detour with the customer

- Request for full Mark 3 compiler
- Will take longer then I have time for
- Customer wants to make a type checker for it
- His type checker is not compatible with the current compiler
- Solution: I make a parser for the customer his type checker

Conclusion

- A parser has been made to process the features of MC
- The parser can give descriptive error messages
- The front-end is designed to be maintainable and expandable
- A type checker was created for the run time features of MC

Demo

The front-end generates a correct data structure for the back-end

Questions