McGill University
School of Computer Science
COMP 520

Wig Compiler Report

Report No. 2013-12

Furkan Tufekci

December 6, 2013

www.cs.mcgill.ca

Table of Contents

1 Introduction

1.1 Clarifications

Have you discovered any unclear points in the WIG language definition? How have you chosen to resolve these?

I didn't really find any unclear points in the WIG language definition.

-the tuples. When combining, does the assigned tuple need to have the same definition

1.2 Restrictions

Have you deliberately made any restrictions in your version of WIG? What was your motivation? What are the implications?

I didn't make any restrictions in my version of WIG.

1.3 Extensions

Have you made any extensions to your version of WIG? What was your motivation? What are the implications?

I didn't made any extensions in my version of WIG.

1.4 Implementation Status

What is the status of your WIG implementation? Have all your proposed features been implemented? Have they been tested? Do they work?

All the proposed features are implemented. They have been tested and they work.

2 Parsing and Abstract Syntax Trees

2.1 The Grammar

Give the full grammar for your version of the WIG language by listing your bison input with all actions removed.

2.2 Using the flex or SableCC Tool

Discuss any interesting points in your flex implementation of the scanner. What are your token kinds? Did you use start conditions? How and why?

2.3 Using the bison or SableCC Tool

Discuss any interesting points in your bison implementation of the parser. How did you make bison accept your grammar?

2.4 Abstract Syntax Trees

Present the structure of your abstract syntax trees. If you used SableCC, discuss how you modified the tree.

2.5 Desugaring

Do you resolve any syntactic sugar during parsing? How and why?
- int a,b,c; => int a; int b; int c;

2.6 Weeding

Do you weed unwanted parse trees? How and why?

2.7 Testing

How have you tested this phase? Does it work?
- checks that print(parse(print(parse(code)))) = print(parse(code))

3 Symbol Tables

3.1 Scope Rules
Describe the scope rules of your language.

3.2 Symbol Data
Describe the contents of symbol table entries.

3.3 Algorithm
Describe your algorithm for building symbol tables and checking scope rules.

3.4 Testing
How have you tested this phase? Does it work?

4 Type Checking

4.1 Types
Describe the types supported by your language.

4.2 Type Rules
Describe the type rules of your language.

4.3 Algorithm
Describe your algorithm for checking type rules.

4.4 Testing
How have you tested this phase? Does it work?

5 Resource Computation

5.1 Resources

Describe the resources that you compute.

5.2 Algorithm

Describe your algorithm for computing resources.

5.3 Testing

How have you tested this phase? Does it work?

6 Code Generation

6.1 Strategy
Describe the overall strategy for generating code for a service.

6.2 Code Templates
Describe the code templates for your language constructs.

6.3 Algorithm
Describe your algorithm for generating code.

6.4 Runtime System
Describe the runtime system that is used by the services you generate.

6.5 Sample Code
Show the complete code generated for the service tiny.wig.

6.6 Testing
How have you tested this phase? Does it work?

7 Availability and Group Dynamics

7.1 Manual

Describe how to compile and install services.

7.2 Demo Site

Give the URL for a web site that contains demos of services that you have generated.

7.3 Division of Group Duties

Explain who worked on what parts of the compiler and how you divided the work. Reflect on your group work experience and describe what went well, what could have been better, and what you learned.

8 Conclusions and Future Work

8.1 Conclusions

First, provide a brief summary of the report. Next, describe the main things that you learned in the course of this work, and attempt to draw new conclusions, even if they are just “soft” experience-based ones. Detail the things you learned that you did not expect to learn.

8.2 Future Work

What features does the WIG language need, and how could they be provided? How could your compiler be even better?

8.3 Course Improvements

What changes to the course would you recommend for future years?

8.4 Goodbye

What plans do you have, if any, to continue in the field of compiler research and development? Any parting words?