Innopolis University

Jolie Programming language

Type Checker

Daniel Elambo Atonge

SummerInternShip2019

Intro to Jolie, the language for MicroServices

A Service-Oriented Programming Language

Jolie is perfect for fast prototyping.

Latest version: 1.8.2

The basic building blocks of software are not objects or functions, but rather services that can always be relocated and replicated as needed.

Service-Oriented Programming

3 Commandments:

1. Everything is a **service**
2. A service is an application that offers **operations**
3. A service can **invoke** another service by calling one of its operations.

There are great online resources to learn Jolie. Take a look at the official docs at: http://docs.jolie-lang.org

Jolie is the first programming language written for Microservices. Distribution and reusability are achieved by design.

Typing statements in Jolie: Case study If-statement

Ideally, we want this for Jolie's typing system

```
1   include "console.iol"
2   main
3   {
4       //number:void  str:void
5       if (condition) {
6           number = 12 //number:int  str:void
7       } else {
8           str = "twelve" //str:string number:void
9       }
10      //number : void|int  str : void|string
11  }
```

Basic Datatypes

bool: booleans

int: integers

long: long integers

double: double-precision float

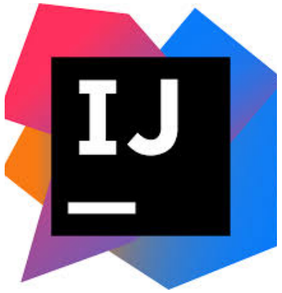
string: strings

raw: byte arrays

void: the empty type

Jolie is a dynamically typed language. Jolies also supports the "any" basic type, a value that can be any basic type

Helpful Tools



Z3 :- Satisfiability modulo theory prover

Intellij :- IDE holding code base

Java :- The language on which the Jolie interpreter is built

Atom :- A hackable text editor for the 21st Century

Understanding Z3

```
(declare-const x Int)
(declare-const y Int)
(declare-fun f (Int) Int)
(declare-fun a () Int) ; a is a constant
(declare-const b Int)
(push)
(assert (= (+ x y) 10))
(assert (= (+ x (* 2 y)) 20))
(check-sat)
(pop) ; remove the two assertions
(assert (> a 20))
(assert (> b a))
(assert (= (f 10) 1))
(check-sat)
```

Z3 is a state-of-the art theorem prover from Microsoft Research. It can be used to check the satisfiability of logical formulas over one or more theories.

Z3 is a low level tool. It is best used as a component in the context of other tools that require solving logical formulas. Try Z3 at https://rise4fun.com/Z3/tutorial

Project Structure

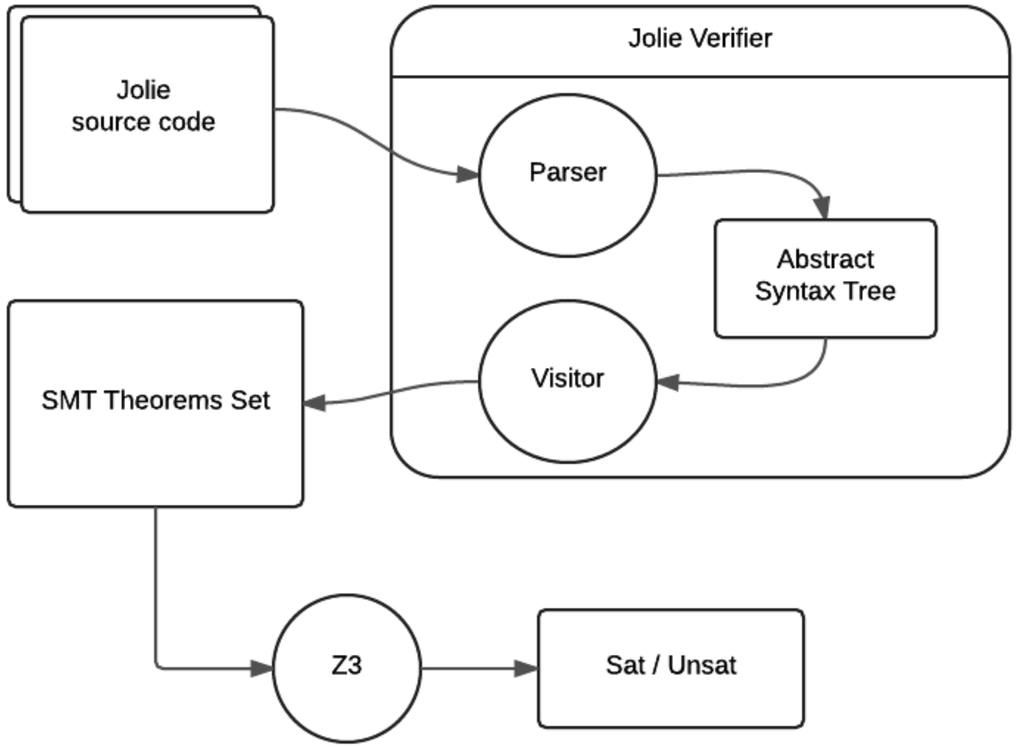


Figure from: **Towards Static Type-checking for Jolie**
https://www.researchgate.net/publication/313898767_Towards_Static_Type-checking_for_Jolie

Indentified shortcomings

Weak Visibility of Variables

This essentially affects a majority of Jolie's language constructs that use "{" and "}"

Example of such construct are:

+ If-else-statements
+ parallel-statements
+ sequential-statements
+ while-loops and more ...

Previous Implementation

```
1        include "console.iol"
2        main
3         {
4           //x:void
5                if (condition) {
6                   x = 5   //x:int
7                };
8           //x:void|int
9                x = "a"    //x:string
10        }
```

Z3 output: sat

However, this should not be allowed if we want Jolie to be typed

As a case-study, we consider if-else-statements

Proposed Solution

Enforce visibility rules among variables and their type consistency among various code block segments

Method and Data-Structures:

+ Inbuilt LinkedList in Java
+ Control structures
+ Looping structures
+ Other data manipulations

Improved Implementation

```
include "console.iol"
main
 {
   //x:void
        if (condition) {
           x = 5    //x:int
        };
   //x:void|int
        x = "a"     //x:string
 }
```

Z3 output: **unsat** 

According to Nielsen's thesis,
if-else-statements should be typed as:

$$\frac{\Gamma \vdash e{:}bool \quad \Gamma \vdash_{\mathrm{B}} B_1 \triangleright \Gamma' \quad \Gamma \vdash_{\mathrm{B}} B_2 \triangleright \Gamma'}{\Gamma \vdash_{\mathrm{B}} \texttt{if}(e)\ B_1\ \texttt{else}\ \ B_2 \triangleright \Gamma'}$$

Future Work

Ideally, we want this for Jolie's typing system and more

```
1    include "console.iol"
2    main
3    {
4        //number:void  str:void
5        if (condition) {
6            number = 12 //number:int  str:void
7        } else {
8            str = "twelve" //str:string number:void
9        }
10       //number : void|int  str : void|string
11   }
```

Introducing new types

+ choice-types
+ sum-types

(Implementing them in a satisfiability modulo theory solver e.g Z3)

Updating Nielsen's thesis

Identify ineffectively implemented or unimplemented rules from Nielsen's thesis and Improving them

Typing other layers

choice-types also known as untagged unions have been implemented in the recent releases of Jolie

References

The Jolie official Documentation: http://jolie-lang.org

Z3 Tutorials: https://rise4fun.com/Z3/tutorial

Verified type checker for Jolie programming language: https://arxiv.org/pdf/1703.05186.pdf

A Type System for the Jolie Language. Julie Meinicke Nielsen. Master's thesis, Technical University of Denmark, 2013.

Any Questions?

Thanks for listening