Enabling Java string-bioinformatics libraries in Kawa Scheme

Mark E. Royer and Sudarshan S. Chawathe

Department of Computer Science
School of Computing and Information Science, University of Maine

Outline

Motivation

Java string-bioinformatics libraries and Kawa Scheme

- Integration of a well-established JVM implementation of Scheme, viz. Kawa
- Cater to existing Java libraries (particularly bioinformatics)
- Java libraries are decorated with special annotations to indicate they should be used in the Kawa environment
- Annotated Java functions are presented in a natural way in the Kawa Scheme environment

The Java Programming Environment

- Java is a **mature** and **robust** programming environment
- Tools and libraries for use in bioinformatics
- Particular libraries of string-bioinformatics algorithms
- Java-based tools provide
  - Strong typing
  - Portability
  - The Java Virtual Machine features (garbage collection, etc.)

Java Environment Prototyping

- Java is **not** convenient for prototyping
- Interactive development is new
- **REPL** (Read-Evaluate-Print-Loop)
  just recently added in Java 9
- The new jshell for Open JDK 11 is **Very good!**

The Scheme Programming Language

- Subset of the Lisp programming language
- Runs on the JVM via Kawa
- Convenient for rapid prototyping
- Scheme is an attractive choice because
  - Well-studied and mature language
  - Enables programming in diverse styles – in particular functional-style programming

Technique

Annotations

- Java string-bioinformatics libraries are decorated with a special annotation that indicates they are to be made available in Kawa
- Annotations in Java are simply tags containing meta-data that begin with the @ symbol

Javadoc Extraction

- At compile time, descriptive information is extracted from the source code and stored in XML files
- XML files are bundled with other artifacts within jar files
- Function descriptive information is extracted from the accompanying Javadoc
- Annotation syntax allows supplying alternative documentation
- Documentation is made available to the Kawa REPL

Functional Designation

- **Deterministic**
  - Value returned by a function depends only on the values of its argument
  - The result cannot depend on any state
  - May **not** depend on any external input from I/O devices
- **Side-effect free**
  - Execution does not cause any observable side effect or output
  - side-effect-free function must not mutate objects or output to I/O devices.

Modification of Existing Libraries

- Existing Java string-bioinformatics libraries
  - **Directly** modified by adorning functions with annotations
  - **Indirectly** supported by wrapping them with the introduction of a language-bridge library

Annotation Tooling

Annotation Processor Interface

- The `Processor` class provides a mechanism to process annotated classes
- Most implementations extends the `AbstractProcessor` class
- Processing takes place using a sequence of rounds
- The user defined processor is specified using the `@AutoService(Processor.class)` annotation

Extracting and Storing Data in XML

- Data is stored in XML using the `JAXB-API`
- For example, Java comment objects are converted into `XML`
- A fully qualified XML file is created for every class that contains `@KawaFunction` annotated methods

Annotation Discovery at Runtime

- Initialization of the `KawaServer` includes searching the annotation processor classpath for annotation XML files
- Function aliasing is performed on the discovered `@KawaFunction` annotated methods
- By default, *long* packaging namespace prefix removed
- The newly created alias is available in the default namespace `kjava`

Kawa-Scheme Annotations

Declaration of Annotation

```
1 @Retention(RUNTIME)
2 @Target(METHOD)
3 public @interface KawaFunction { /* ... */ }
```

1. `@Retention` specifies the annotation is available at runtime
2. The `@Target` indicates that this annotation may only be used with Java methods

@KawaFunction Annotation Expanded

```
1  public @interface KawaFunction {
2
3    String namespace() default "kjava";
4
5    /* Grabs the surrounding Javadoc by default */
6    String description() default "";
7
8    /* Returns the same result given the
9       same argument value(s). */
10   boolean deterministic() default false;
11
12   /* No semantically observable side effect or output */
13   boolean sideEffectFree() default false;
14 }
```

Kawa-Scheme Programming

Kawa Telnet Server

- Create on the Java side when application initializes
- Use the `KawaServer` class to initialize a Kawa telnet server in a separate Java thread.
- Care must be taken to ensure that client model modification does not interfere with native Java library states.

Kawa Client Connection

- **REPL** created from telnet connection (default port 5146)
- **rlwrap** program provides typical shell experience
- **Emacs** is excellent for Scheme programming

Conclusion

Future Work

- Test with a broader set of string-bioinformatics libraries
- Expand on the type interaction of the two language environments
  - Support function **preconditions** and **postconditions**
  - Investigate functional, multi-threaded applications

Summary

- Java libraries decorated with annotations
- Library provides natural functional-style programming environment
- Provide Kawa Scheme integration with existing Java string-bioinformatics libraries