# Supplementary Material for: Composable Languages for Bioinformatics: The NYoSh experiment

Manuele Simi[1,2], Fabien Campagne[1,2]*,

[1] The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Alsaud Institute for Computational Biomedicine, The Weill Cornell Medical College, New York, New York, United States of America. [2] Department of Physiology and Biophysics, The Weill Cornell Medical College, New York, New York, United States of America.

*Correspondence to: Fabien Campagne, fac2003@campagnelab.org.

Table of Contents

```
# This is the only function that aligners need to implement.
# Parameters:
#   $1: a temporary filename
#   $2: the basename that should be used to store the sorted alignment

function plugin_align {
  #sample parameters reading
  OUTPUT=$1
  BASENAME=$2
  #invoke the model through the script generated by the GobyWeb language
  . ${JOB_DIR}/run_model.sh plugin_align ${OUTPUT} ${BASENAME}
}


```

```

export MPS_HOME=${RESOURCES_ARTIFACTS_MPS_DISTRIBUTION}
MPS_LIBS=`cat ${RESOURCES_MPS_JARS_LIST} |awk '{ORS=":"; print $1}'`
NYOSH_SUPPORT_LIBS="$RESOURCES_ARTIFACTS_MPS_SUPPORT_LIBS/*"
CLASSPATH=${MPS_LIBS}${NYOSH_SUPPORT_LIBS}:${JOB_DIR}/plugin.jar:${JOB_DIR}
MODEL=BWAGobyArtifactPlugin
NYOSH_SCRIPTNAME=BWAGobyArtifactScript
CLASSNAME=${MODEL}.${NYOSH_SCRIPTNAME}
java ${PLUGIN_NEED_DEFAULT_JVM_OPTIONS} -classpath ${CLASSPATH} ${CLASSNAME} "$
@"


```

```java
package BWAGobyArtifactPlugin;

/*Generated by MPS */

import org.campagnelab.nyosh.logging.StepsLoggerHelper;

public class ErrorManagementImplementation {
    public void recordStepDone(String actionDescription) {
        StepsLoggingSuccessHandler_ym50rj_(actionDescription);

    }

    public void exception(String actionDescription, int statusCode, Exception exception) {
        StepsLoggingErrorHandler_kysnd9_(actionDescription, statusCode, exception);
    }

    public static void StepsLoggingErrorHandler_kysnd9_(String actionDescription, int statusCode, Excepti
on exception) {
        StepsLoggerHelper.createLogFile();
        StepsLoggerHelper.assertTrue(false, "A step failed");
    }

    public static void StepsLoggingSuccessHandler_ym50rj_(String actionDescription) {
        StepsLoggerHelper.createLogFile();
        StepsLoggerHelper.done(actionDescription, 0);
    }

}

```

```java
package BWAGobyArtifactPlugin;

/*Generated by MPS */

import java.util.Set;
import java.util.HashSet;

import org.campagnelab.nyosh.environment.parsers.Parser;
import org.campagnelab.nyosh.environment.parsers.JVMEnvParser;

import java.util.SortedSet;

import org.campagnelab.nyosh.environment.parsers.ScriptVariable;
import org.campagnelab.nyosh.environment.parsers.GobyWebParser;
import org.campagnelab.nyosh.environment.NYoShRuntimeEnvironment;
import org.apache.commons.io.FilenameUtils;
import org.campagnelab.nyosh.environment.parsers.MapFileParser;
import org.campagnelab.nyosh.exec.CommandAssembler;
import org.campagnelab.nyosh.exec.CommandExecutionPlan;
import org.campagnelab.stepslogger.FileStepsLogger;

import java.io.File;
import java.io.IOException;

import org.apache.log4j.Logger;
import org.apache.log4j.LogManager;

public class BWAGobyArtifactScript {

    private static Set<String> exportedVariables = new HashSet<String>();

    public static void main(String[] arguments) {
        if (arguments.length == 0) {
            arguments = new String[]{"main"};
        }

        // BEFORE_ENTRY_POINT_EXECUTION

        if ("plugin_align".equals(arguments[0])) {

            if (arguments.length == 3) {
                align(arguments[1], arguments[2]);
            } else {
                System.err.println("Invalid number of arguments");
            }
            finish();
            System.exit(0);
        }
        System.err.printf("The entry point %s name was not recognized", arguments[0]);
        finish();
        System.exit(1);
    }

    public static void align(String output, String basename) {
        {
            initializeStepsLogging();
            System.out.println("Executing step: " + "Catch all steps for GobyWeb");
            boolean success_u4s4ck_a0d = false;
            String reason_u4s4ck_a0d = "Catch all steps for GobyWeb";
            Exception exception_a0d = null;
            try {
                {
                    Parser parser_u4s4ck_a0a0a0a5a0a0a0d = new JVMEnvParser();
                    SortedSet<ScriptVariable> variables_u4s4ck_a0a0a0a5a0a0a0d = parser_u4s4ck_a0a0a0a5a0a0
a0d.parseAtRunTime();
                    Parser parser_u4s4ck_b0a0a0f0a0a0a3 = new GobyWebParser();
        SortedSet<ScriptVariable> variables_u4s4ck_b0a0a0f0a0a0a3 = parser_u4s4ck_b0a0a0f0a0a0a3.parseAtRunTime();

        String COLOR_SPACE_OPTION = (NYoShRuntimeEnvironment.getEnvironment().getVariableValue("COLOR_SPACE").equals("true") ?
                "-c" :
                ""
        );
        String BWA_GOBY_EXEC_PATH = NYoShRuntimeEnvironment.getEnvironment().getVariableValue("RESOURCES_ARTIFACTS_BWA_WITH_GOBY_ARTIFACT_EXECUTABLE") + ;
        String ORG = NYoShRuntimeEnvironment.getEnvironment().getVariableValue("ORGANISM").toUpperCase();
        System.out.println("Genome reference id: " + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("GENOME_REFERENCE_ID"));
        String[] genomeInfo = NYoShRuntimeEnvironment.getEnvironment().getVariableValue("GENOME_REFERENCE_ID").toUpperCase().split("\\.");
        String BUILD_NUMBER = "";
        String ENSEMBL_RELEASE = "";
        if (genomeInfo.length == 2) {
            BUILD_NUMBER = genomeInfo[0];
            ENSEMBL_RELEASE = genomeInfo[1];
        } else {
            fail(false, "Invalid genome " + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("GENOME_REFERENCE_ID"), 1);
        }
        String SAMPE_SAMSE_OPTIONS =
                NYoShRuntimeEnvironment.getEnvironment().getVariableValue("PLUGINS_ALIGNER_BWA_GOBY_ARTIFACT_NYOSH_SAMPE_SAMSE_OPTIONS");
        String ALL_OTHER_OPTIONS =
                NYoShRuntimeEnvironment.getEnvironment().getVariableValue("PLUGINS_ALIGNER_BWA_GOBY_ARTIFACT_NYOSH_ALL_OTHER_OPTIONS");
        int BWA_GOBY_NUM_THREADS = 4;
        String SAMPLE_NAME = FilenameUtils.getBaseName(NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE"));
        String PLATFORM_NAME = NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_PLATFORM");
        String READ_GROUPS = "@RG\\tID:1\\tSM:" + SAMPLE_NAME + "\\tPL:" + PLATFORM_NAME + "\\tPU:1";
        String INDEX_DIR_KEY = "RESOURCES_ARTIFACTS_BWA_WITH_GOBY_ARTIFACT_INDEX_" + ORG + "_" + BUILD_NUMBER + "_" + ENSEMBL_RELEASE;
        String INDEX_DIR = NYoShRuntimeEnvironment.getEnvironment().getVariableValue(INDEX_DIR_KEY) + "/index";
        System.out.println("Index directory is: " + INDEX_DIR);
        System.out.println("Loading environment from: "
                + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("RESOURCES_ARTIFACTS_PROTOBUF_CPP_LIBRARIES")
                + "/setup.sh");
        MapFileParser parser_u4s4ck_a91a0a0a5a0a0a0d = new MapFileParser();
        SortedSet<ScriptVariable> variables_u4s4ck_a91a0a0a5a0a0a0d =
                parser_u4s4ck_a91a0a0a5a0a0a0d.parseAtRunTime(String.format("%s/%s",
                        NYoShRuntimeEnvironment.getEnvironment().getVariableValue("RESOURCES_ARTIFACTS_PROTOBUF_CPP_LIBRARIES"),
                        "setup.sh"));
        for (ScriptVariable var : variables_u4s4ck_a91a0a0a5a0a0a0d) {
            exportedVariables.add(var.name);
        }
        System.out.println("Loading environment from: "
                + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("RESOURCES_ARTIFACTS_GOBY_CPP_API_LIBRARIES")
                + "/setup.sh");
        MapFileParser parser_u4s4ck_a12a0a0a5a0a0a0d = new MapFileParser();
        SortedSet<ScriptVariable> variables_u4s4ck_a12a0a0a5a0a0a0d =
                parser_u4s4ck_a12a0a0a5a0a0a0d.parseAtRunTime(String.format("%s/%s",
                NYoShRuntimeEnvironment.getEnvironment().getVariableValue("RESOURCES_ARTIFACTS_GOBY_CPP_API_LIBRARIES"),
                "setup.sh"));
        for (ScriptVariable var : variables_u4s4ck_a12a0a0a5a0a0a0d) {
            exportedVariables.add(var.name);
        }
        String SAI_FILE_0 = String.format("%s%s-0.sai",
                FilenameUtils.getFullPath(NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE")), SAMPLE_NAME);
        String SAI_FILE_1 = String.format("%s%s-1.sai",
                FilenameUtils.getFullPath(NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE")), SAMPLE_NAME);
        {
            StringBuffer commandBuffer = new StringBuffer();
            CommandAssembler assembler = new CommandAssembler();
            assembler.appendCommand("nice " + BWA_GOBY_EXEC_PATH
                    + " aln -w 0 -t " + BWA_GOBY_NUM_THREADS
                    + " " + COLOR_SPACE_OPTION + " -f "
                    + SAI_FILE_0 + " -l "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("INPUT_READ_LENGTH")
                    + " " + ALL_OTHER_OPTIONS + " -x "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                    + " -y "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                    + " " + INDEX_DIR + " " + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE"));
            commandBuffer.append("nice " + BWA_GOBY_EXEC_PATH + " aln -w 0 -t "
                    + BWA_GOBY_NUM_THREADS + " " + COLOR_SPACE_OPTION + " -f "
                    + SAI_FILE_0 + " -l "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("INPUT_READ_LENGTH")
                    + " " + ALL_OTHER_OPTIONS + " -x "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                    + " -y "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                    + " " + INDEX_DIR
                    + " "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE"));
            //  process output according to type
            CommandExecutionPlan plan = null;
            lastExitCode = -1;
            try {
                assembler.setLocalEnvironment(exportedVariables);
                assembler.finishAssembly();
                plan = assembler.getCommandExecutionPlan();
                lastExitCode = plan.run();

            } finally {
                if (plan == null || !(plan.executedCompletely())) {
                    errorManagement.exception("failed executing: " + commandBuffer.toString(), 0, null);
                } else {
                    errorManagement.recordStepDone("successfully executed: " + commandBuffer.toString());
                }
            }
        }

        {
            StringBuffer commandBuffer = new StringBuffer();
            CommandAssembler assembler = new CommandAssembler();
            assembler.appendCommand("nice " + BWA_GOBY_EXEC_PATH
                    + " aln -w 1 -t " + BWA_GOBY_NUM_THREADS
                    + " " + COLOR_SPACE_OPTION + " -f "
                    + SAI_FILE_1 + " -l "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("INPUT_READ_LENGTH")
                    + " " + ALL_OTHER_OPTIONS + " -x "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                    + " -y "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                    + " " + INDEX_DIR + " "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE"));
            commandBuffer.append("nice " + BWA_GOBY_EXEC_PATH
                    + " aln -w 1 -t " + BWA_GOBY_NUM_THREADS + " "
                    + COLOR_SPACE_OPTION + " -f " + SAI_FILE_1
                    + " -l "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("INPUT_READ_LENGTH")
                    + " " + ALL_OTHER_OPTIONS + " -x "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                    + " -y "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                    + " " + INDEX_DIR + " "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE"));
            // process output according to type
            CommandExecutionPlan plan = null;
            lastExitCode = -1;
            try {
                assembler.setLocalEnvironment(exportedVariables);
                assembler.finishAssembly();
                plan = assembler.getCommandExecutionPlan();
                lastExitCode = plan.run();

            } finally {
                if (plan == null || !(plan.executedCompletely())) {
                    errorManagement.exception("failed executing: " + commandBuffer.toString(), 0, null);
                } else {
                    errorManagement.recordStepDone("successfully executed: " + commandBuffer.toString());
                }
            }
        }

        {
            StringBuffer commandBuffer = new StringBuffer();
            CommandAssembler assembler = new CommandAssembler();
            assembler.appendCommand("nice " + BWA_GOBY_EXEC_PATH + " sampe "
                    + COLOR_SPACE_OPTION + " " + SAMPE_SAMSE_OPTIONS + " -F goby -f "
                    + output + " -x "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                    + " -y "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                    + " " + INDEX_DIR + " " + SAI_FILE_0
                    + " " + SAI_FILE_1 + " "
                    + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE")
                    + " -r " + READ_GROUPS);
                    commandBuffer.append("nice " + BWA_GOBY_EXEC_PATH
                            + " sampe " + COLOR_SPACE_OPTION + " "
                            + SAMPE_SAMSE_OPTIONS + " -F goby -f "
                            + output + " -x "
                            + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("START_POSITION")
                            + " -y " + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("END_POSITION")
                            + " " + INDEX_DIR + " " + SAI_FILE_0 + " "
                            + SAI_FILE_1 + " "
                            + NYoShRuntimeEnvironment.getEnvironment().getVariableValue("READS_FILE")
                            + " -r " + READ_GROUPS);
                    //  process output according to type
                    CommandExecutionPlan plan = null;
                    lastExitCode = -1;
                    try {
                        assembler.setLocalEnvironment(exportedVariables);
                        assembler.finishAssembly();
                        plan = assembler.getCommandExecutionPlan();
                        lastExitCode = plan.run();

                    } finally {
                        if (plan == null || !(plan.executedCompletely())) {
                            errorManagement.exception("failed executing: " + commandBuffer.toString(), 0, null);
                        } else {
                            errorManagement.recordStepDone("successfully executed: " + commandBuffer.toString());
                        }
                    }
                }
            }
            success_u4s4ck_a0d = true;
        } catch (Exception e) {
            exception_a0d = e;

        } finally {
            if (!(success_u4s4ck_a0d)) {
                errorManagement.exception("step " + reason_u4s4ck_a0d + " failed.", 0, exception_a0d);
            } else {
                errorManagement.recordStepDone(reason_u4s4ck_a0d);
            }

            try {
                // This was the last step, we need to close the stepslogger:
                _steps.close();

            } catch (Exception e) {
                if (LOG.isInfoEnabled()) {
                    LOG.info("An error occured closing stepslogger", e);
                }
            }
        }
    }
    // end of reduce_step
}

public static void finish() {
}

private static FileStepsLogger _steps;

// declared flag removed
public static void initializeStepsLogging() {
    if (BWAGobyArtifactScript._steps == null) {
        BWAGobyArtifactScript._steps = new FileStepsLogger(new File("./"));
    }
}

public static void fail(boolean mustBeTrue, String reason) {
    fail(mustBeTrue, reason, 1);
}

private static void done(String stepDescription, int statusCode) {
    BWAGobyArtifactScript._steps.step(stepDescription, statusCode);
}

/*package*/
static void fail(boolean mustBeTrue, String reason, int statusCode) {
    if (!(mustBeTrue)) {
        BWAGobyArtifactScript._steps.error(reason);
        try {
            BWAGobyArtifactScript._steps.close();
        } catch (IOException e) {
            //  we tried to close stepslogger. Giving up now.
        }
        System.exit(statusCode);
    }
}

private static int lastExitCode = 0;
private static ErrorManagementImplementation errorManagement = new ErrorManagementImplementati
on();
protected static Logger LOG = LogManager.getLogger(BWAGobyArtifactScript.class);
}

```