# Semantic Comparison of Alloy Models

MoDELS 2020 Artifact Evaluation

# Content of Submission

- **readme.pdf** this document
- **paper.pdf** version of paper as reviewed (not yet updated to CR)
- Watch a screencast going over the artifact (ca. 32min)
  - https://youtu.be/JA93sy2oHfo (select 1080p quality)
- **alloy-diff-gui.jar** a limited prototype GUI integrated into Alloy Analyzer
- **alloy-diff/experiments/** folder with packaged binaries and scripts to run experiments
- **alloy-diff/** code of our prototype
  - a fork of the Alloy repository from https://github.com/AlloyTools/org.alloytools.alloy
  - **iAlloy-dataset-master, models-master, platinum-experiment-data** collections of specifications with sources mentioned in the paper
  - **org.alloytools.alloy.diff** the place of our code

# Platforms

- We have developed all code on Windows 10 and Ubuntu 20.04 LTS with Eclipse 2020-06 and OpenJDK 11 (other versions should work)

- We mainly use the SAT solver CryptoMiniSatJNI, which requires
  - a 32Bit Java under Windows 10 (not tested) or
  - a Linux system

# Suggested Steps

1. Quickly browse **readme.pdf** (it repeats in the video)
2. Watch **watchme.mp4** to get an overview of the artifact
3. Play with the GUI prototype **alloy-diff-gui.jar**
4. Inspect scripts and run the experiments under **alloy-diff/experiments/**
    a. this might only work on Linux due to the SAT solver binaries
5. Inspect the code

# Alloy GUI Prototype

- Start the **alloy-diff-gui.jar** (double click or via ‘java -jar alloy-diff-gui.jar’)
  - This might require Java version >= 11
- Load or create two Alloy Models
  - the prototype requires that models to compare are saved into files!
- Click the Compare button and select the comparison.
  - Different cases of this are shown in the video for the paper’s example specifications.

**IMPORTANT:** The GUI prototype is very limited (it uses the first command but a max scope of 3 -- Alloys default).

Example: `run {...} for 10 but 4 X, 2 A, exactly 1 B` will be capped to

`run {...} for 3 but 3 X, 2 A, exactly 1 B`

Run directly from code to use higher scopes.

# Experiments JAR File

We include the binaries used for running the experiments described in the paper:

- java -jar **diff.jar** v1.als v2.als 3 withPred

Will check for an instance of the semantic difference of **v1.als** and **v2.als** for scope **3** including the predicates of the first run command in each file.

- java -jar **diff.jar** v1.als v2.als 8

Will check for an instance of the semantic difference of **v1.als** and **v2.als** for scope **8** ignoring run commands, i.e., based on signatures and facts.

To run a comparative analysis (Tbl. 2) on v1.als use **v1.jar** instead of **diff.jar**

# Experiments Scripts

Two bash scripts are included under **alloy-diff/experiments/*.sh**

- **diff.sh** will execute the evaluation on a single pair of Alloy models
  - with a timeout of 10 minutes
  - writing results to a scope dependent CSV file
    - saves names of files, number of SAT variables, millis seconds of processing
  - storing memory consumption to memory.log
- **runAll.sh** lists all pairs of models and scopes to run the evaluation on
  - ca. 836 pairs per scope
  - from scope 3 to 58 in increments of 5

Running the script might take a few days on a strong computer!

# Tracing Paper to Code

This relates our paper to our code:

- **Sect. 2 Example** Alloy listings
  - /org.alloytools.alloy.diff/misc/paper/*.als
- **Algorithm 1** Signature merge algorithm for models $v1$ and $v2$
  - /org.alloytools.alloy.diff/src/main/java/org/alloytools/alloy/diff/ModuleMerger.java
    - public Collection<Sig> mergeSigs(Module v1, Module v2)
- **Algorithm 2** Merging of fields of common signatures $s1$ and $s2$
  - /org.alloytools.alloy.diff/src/main/java/org/alloytools/alloy/diff/ModuleMerger.java
    - private void mergeField(Sig mergedSig, Field f1, Field f2) // for common fields
    - private void addUniqueField(Sig mergedSig, Field field, boolean inC1) // for unique fields

# Tracing Paper to Code

- **Algorithm 3** Computing inherited fields and sig references
  - /org.alloytools.alloy.diff/src/main/java/org/alloytools/alloy/diff/InheritanceUtil.java
    - public InheritanceUtil(Module m)
- **Algorithm 4** Translation transExpr($e$) of expression $e$
  - /org.alloytools.alloy.diff/src/main/java/org/alloytools/alloy/diff/ModuleMerger.java
    - private Expr replaceSigRefs(Expr expr, List<Decl> names, boolean inV1) // line 745
- **Sect. 6.1 Extension**: run and check commands and scope
  - /org.alloytools.alloy.diff/src/main/java/org/alloytools/alloy/diff/CommandGenerator.java
    - public Command generateDiffCommand(Module v1, Module v2, int scope, boolean withPred, Analysis a)

# Tracing Paper to Code

- **Sect. 7 Implementation and Evaluation**
  - See the scripts in folder experiments/ and the code for creating the data in
    - /org.alloytools.alloy.diff/src/test/java/org/alloytools/alloy/diff/DiffStatsComputer.java
    - /org.alloytools.alloy.diff/src/test/java/org/alloytools/alloy/diff/DiffStatsComputer1.java
- **Sect. 7.2 Validation**
  - A lot of tests in /org.alloytools.alloy.diff/src/test/java and specifically
  - /org.alloytools.alloy.diff/src/test/java/org/alloytools/alloy/diff/ModuleComparisonTest.java

# Badge Application

**Badge 1: Available** (*As long as it is available during the review process, we are good.*)

- We make the artifact available for review
- We are happy to place the artifact in our University's Figshare repository with a DOI etc. after incorporating feedback

**Badge 2: Reusable** (If the data/source code has some logical structure, documentation, etc., we can consider it is reusable.)

- See the documentation of the structure of the code and the paper's listings
- Watch the overview provided in the screencast **watchme.mp4**