Forge: A Quick Guide

Version 7.2

July 13, 2019

Welcome to Forge! Forge is a language built for teaching formal methods and model checking.

Forge extends the Ocelot package to create first-order relational logic models. Forge differs from Ocelot in that it provides:

- Alloy-like semantics for universe declaration (i.e. implicitly defining a universe of discourse in terms of sigs and bounds)
- A built-in visualizer and evaluator
- Support for KodKod integers (coming soon)
- Built-in semantics for state (coming soon)
- Additional teaching tools like language levels and different interaction modes (coming soon)

1 The Language

To begin working with forge, make sure to begin your file with the line:

```
#lang s-exp forge
```

1.1 Sigs

The primary construct in forge is the sig. A sig represents a type of atom that can appear in the universe of discourse.

To declare a sig, use the declare-sig or declare-one-sig form:

```
(declare-sig name)
(declare-sig name ((relation field-sig ...) ...))
```

Adds a sig called *name* to the universe. If any relations are specified, adds them to the spec. The type of a declared relation is (name -> field-sig0 -> field-sig1 ...).

For example, we might use forge to model a simple directed graph. To do this, we'd specify a sig to represent vertices of the graph and a relation to represent edges.

```
(declare-sig vertex ((edges vertex)))
```

This declaration adds two relations to the spec - the arity 1 relation `vertex`, which contains exactly the atoms of type vertex in the universe, and the arity 2 relation `edges`, which contains some number of pairs of vertices, or is empty.

```
(declare-one-sig name)
(declare-one-sig name ((relation field-sig ...) ...))
```

Like `declare-sig`, but also implicitly bounds the sig such that each instance contains exactly one atom of type *name*.

1.2 Expressions

2 The Visualizer