Verification of complex systems in Stainless

Romain Ruetschi

Version 0.1
December 2017

Abstract

TODO

Master Thesis Project under the supervision of
Prof. Viktor Kuncak & Dr. Jad Hamza
Lab for Automated Reasoning and Analysis LARA - EPFL

Contents

1 Introduction

TODO

Related Works

TODO

2 Motivation

TODO

3 Verifiying Actor Systems

3.1 The Actor Model

TODO

3.1.1 Message

In our framework, messages are modelled as concrete subclasses of the abstract class `Msg`.

3.1.2 Actor Reference

An actor can be referenced by its `ActorRef`.

3.1.3 Message in Flight

In-flight messages are represented as a product type of the destination `ActorRef` and the `Msg`.

3.1.4 Behavior

A behavior specifies both the current state of an actor, and how this one should process the next incoming message. In our framework, these are modelled as a subclass of the abstract class `Behavior`, which defines a single abstract method `processMsg`, to be overriden for each defined behavior.

```
abstract class Msg

abstract class ActorRef {
  def !(msg: Msg)(implicit ctx: ActorContext): Unit = {
    ctx.send(this, msg)
  }
}

case class Packet(dest: ActorRef, payload: Msg)

case class ActorContext(
  self: ActorRef,
  var toSend: List[Packet]
) {
  def send(to: ActorRef, msg: Msg): Unit = {
    toSend = toSend :+ Packet(to, msg)
  }
}

abstract class Behavior {
  def processMsg(msg: Msg)(implicit ctx: ActorContext): Behavior
}
```

Listing 1: Behavior in PureScala

3.1.5 Actor Context

As mentioned above, when a message is delivered to an actor, the latter is provided with a context, which holds a reference to itself, as well as a list of `Packet`s to send.

3.1.6 Transition

```
case class Transition(
  from: ActorRef,
  to: ActorRef,
  msg: Msg,
  newBehavior: Behavior,
  toSend: List[Packet]
)
```

Listing 2: Transition in PureScala

3.1.7 Actor System

```
case class ActorSystem(
  behaviors: CMap[ActorRef, Behavior],
  inboxes: CMap[(ActorRef, ActorRef), List[Msg]],
  trace: List[Transition]
) {
  def step(from: ActorRef, to: ActorRef): ActorSystem = /* ... */
}
```

Listing 3: Actor System in PureScala

3.2 Proving Invariants

TODO

3.3 Reasoning About Traces

TODO

4 Biparty Communication Protocols

TODO

5 Conclusion

TODO

6 Future Works

TODO