Advanced Programming

Exam 2018

Silvan Robert Adrian

November 4, 2018

Contents

1 Question 1.1: Utility functions

The Code for this task is attached in the appendix A.1.

1.1 Version

2 Question 1.2: Parsing appm databases

2.1 Choice of parser library

I implemented the Parser for appm in parsec, mostly out of this reason:

- Better Error handling compared to ReadP
- I do have more experience with Parsec then ReadP

2.2 Transform Grammar

The existing grammar has some ambiguities, like allowing many names, version etc. which now transformed to only allow once

```
Database ::= \epsilon
```

3 Question 2: Earls of Ravnica

3.1 Solution

3.2 Implementation

The earls of Ravnica can be seen as a state machine for which I chose to use gen_statem. The following states exist:

- Under Configuration
- Active
- Shutting down

A Code Listing

A.1 Question 1.1: handout/appm/src/Utils.hs

```haskell
module Utils where

-- Any auxiliary code to be shared by Parser, Solver, or tests
-- should be placed here.

import Defs

instance Ord Version where
  (<=) (V[]) _ = False
  (<=) (V(_:_)) (V []) = True
```

```haskell
  (<=) (V[VN v1int v1str])  (V[VN v2int v2str]) =
    if checkVersion v1int v2int v1str v2str then True else False
  (<=) (V(VN _ _:xs))  (V(VN _ _:ys)) = V(xs) <= V(ys)

checkVersion :: Int -> Int -> String -> String -> Bool
checkVersion a b c d = a <= b && (c <= d || length(c) <= length(d))

merge :: Constrs -> Constrs -> Maybe Constrs
merge [] [] = Just []
merge c1 [] = Just c1
merge [] c2 = Just c2
-- merge ((pname1,(bool1, miv1, mxv1)):xs) ((pname2,(bool2,
↪  miv2,mxv2)):ys) =
--                                                        if miv1 <=
↪  (miv2 <= mxv1)  then
--                                                            if miv1
↪  <= (mxv1 <= mxv2) then
--
↪  Just [(pname2, (bool1,miv1, mxv2))]
--                                                            else
--
↪  Nothing
--                                                        else
--
↪  Nothing
merge (c1) (c2) = Just (c1 ++ c2)
```

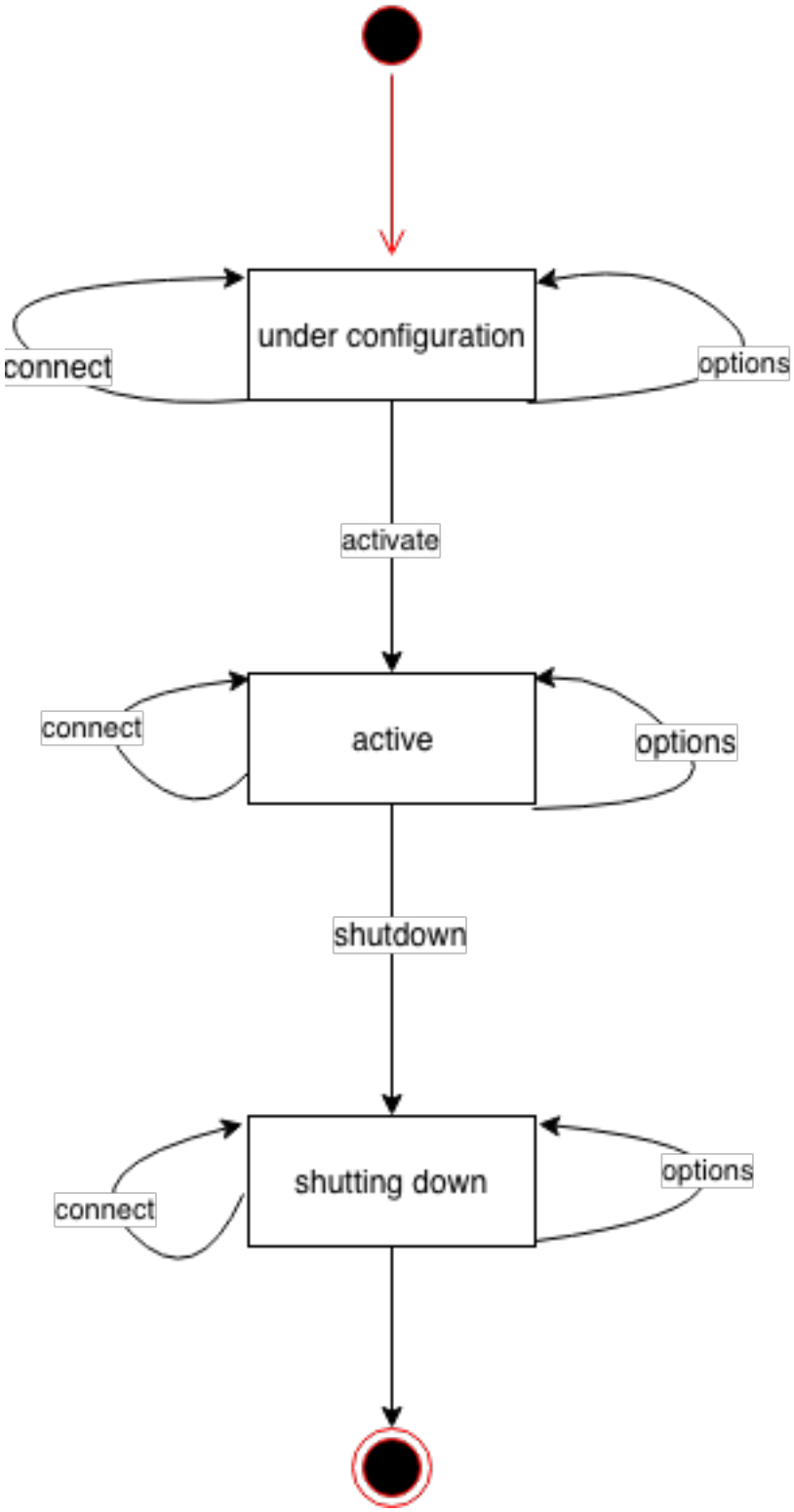

Figure 1: Simple State machine diagramm