ATF Help

1 Help with Topics

```
module Help (help) where

type HelpTopic = (String, [String])

help_topics :: [HelpTopic]
help_topics =
    [ ("compile",
        [ "Usage:"
        , "    $> atf compile L.atf_lang x1.atf_src ... xN.arf_src"
        , "Description:"
        , "    This command transpiles each of xi.atf_src file into a"
        , "    xi.atf_tgt file, for which both the content and file type"
        , "    (atf_tgt) are specified by L.atf_lang."
        , "Details:"
        , "    L.arg_lang is written in a language-specification format that"
        , "    is immediately compatible with atf. L, the compiled"
        , "    specification, which is then used to parse and transpile each"
        , "    xi.atf_src into a corresponding xi.atf_tgt file." ] )
    , ("help",
        [ "Usage:"
        , "    $> atf help <topic>"
        , "Description:"
        , "    This command prints a helpful illustration description of the"
        , "    usage syntax and execution details about the given topic. For"
        , "    example, I would recommend running `atf help compile`." ] )
    , ("all",
        [ "ATF Commands:"
        , "    help <topic>"
        , "    compile <lang-spec> <source-1> ... <source-N>" ] )
    ]

print_help_topic :: HelpTopic -> IO ()
print_help_topic (topic, content)
    = foldl (>>) (putStr "")
    $ map putStrLn
    $ ("help for \"" ++ topic ++ "\":\n") : content

extract_help_topic :: String -> Maybe HelpTopic
extract_help_topic topic =
    let helper :: [HelpTopic] -> Maybe HelpTopic
        helper [] = Nothing
        helper (h:hs) = if topic == fst h
            then Just h
            else helper hs
    in helper help_topics

help :: String -> IO ()
help topic = case extract_help_topic topic of
```

```
Nothing -> help "all"
Just ht -> print_help_topic ht
```