Slicing and Functional Programming

Simon Thompson

University of Kent

Haskell and Erlang

- Pure language …
- … but has monads
- Strongly typed
- Lazy evaluation
- Sequential core

- Impure language …
- … but single assignment
- Weakly typed
- Strict evaluation
- Concurrent

Concurrency in Erlang

```
-module(echo).
-export([go/0, loop/0]).

go() ->
  Pid = spawn(echo, loop, []),
  Pid ! {self(), hello},
  receive
    {Pid, Msg} ->
      io:format("~w~n",[Msg])
  end,
  Pid ! stop.
```

```
loop() ->
  receive
    {From, Msg} ->
      From ! {self(), Msg},
      loop();
    stop ->
      true
  end.
```

Laziness in Haskell

- Arguments are only evaluated when needed.

```
if True  t f = t
if False t f = f

if (2>3) ⊥ 4 ⇝ 4
```

- Arguments are only evaluated to the extent that is needed for evaluation to proceed.

```
sft (a:b:xs) = a+b
ones = 1:ones

sft ones ⇝ 2
```

Slicing

```
parseMessage :: MessageList -> (Message, MessageList)
parseMessage [] = ([], [])
parseMessage xs = (takeWhile (/= '&') (tail ys),
                   dropWhile (/= '&') (tail ys) )
                  where
                  ys = dropWhile (/= '&') xs

parseMsgL :: MessageList -> Message
parseMsgL [] = []
parseMsgL xs = takeWhile (/= '&')
                 (tail (dropWhile (/= '&') xs))
```

Clone detection

- Search for common generalisation.
- What about insertion, deletion or permutation of statements?
- Can slicing help?

```
Clone 2. This code appears 4 times:
/Users/simonthompson/Desktop/small/brchcp_vig_calls_SUITE_copy.erl:2193.4-2197.71:
   new_fun(SidMux, fun precond_no_dsp_create_mod_tdm/1,
        fun precond_create_mux_video/2,
        fun precond_no_dsp_create_ip_video/1)
/Users/simonthompson/Desktop/small/brchcp_vig_calls_SUITE_copy.erl:2179.4-2186.70:
   new_fun(SidMux,
        fun precond_no_dsp_create_mod_tdm_inactive/1,
        fun precond_create_mux_audio/2,
        fun precond_no_dsp_create_ip_audio_inactive/1)
/Users/simonthompson/Desktop/small/brchcp_vig_calls_SUITE_copy.erl:2168.4-2172.71:
   new_fun(SidMux, fun precond_no_dsp_create_mod_tdm/1,
        fun precond_create_mux_audio/2,
        fun precond_no_dsp_create_ip_audio/1)
/Users/simonthompson/Desktop/small/brchcp_vig_calls_SUITE_copy.erl:2204.4-2208.71:
   new_fun(SidMux, fun precond_no_dsp_create_mod_tdm/1,
        fun precond_create_mux_audio/2,
        fun precond_no_dsp_create_ip_audio/1)

The cloned expression/function after generalisation:

new_fun(SidMux, NewVar_1, NewVar_2, NewVar_3) ->
    {ChBlade,_SbBlade} = reserve_all_sb(),
    {SidTdm,LocalData,IntCepTdm} = NewVar_1(ChBlade),
    IntCepMux = precond_no_dsp_create_mux_223(SidMux,LocalData),
    {IntCepMuxAudio,SidMuxAudio} = NewVar_2(SidMux,LocalData),
    {SidAudio,IntCepAudio} = NewVar_3(LocalData),
    {SidTdm, IntCepTdm, IntCepMux, SidAudio, SidMuxAudio,
     IntCepAudio, IntCepMuxAudio, LocalData}.
```

Ideas

- Slicing for debugging functional programs.
- Slicing components of complex structures.
- … and others … ?