Road Construction Assignment (Erlang)

Advanced Functional Programming (1DL450), 2017-11-10

Henrik Arro (henrik.arro@gmail.com)

This document describes how the Road Construction assigment was implemented and tested.

Implementation

The pieces of road already constructed are represented as intervals `{low, high}`, where the high and low bounds are inclusive. For example, `{2, 5}`, means that the segment starting at 2 and ending at 5 has been constructed. If the total length of road to be built is 10, that means that segments `{1,2}` and `{5,10}` remain.

The implementation of `days/3` uses two helper functions, `add_interval/2` and `is_schedulable/4`.

add_interval/2

The function `add_interval(Interval, Intervals)` adds `Interval` to the list `Intervals`, which is supposed to be sorted by the intervals lower bound, and with no overlapping intervals. The new interval is inserted in the right place in the list, and if possible merged with surrounding intervals.

is_schedulable/4

The name of this function was chosen because the problem at hand is basically a scheduling problem. The function `is_schedulable(Length, Intervals, Start, End)` is true if and only if a "job" of length `Length` can be scheduled into a time window starting at `Start` and ending at `End`, with other jobs already scheduled according to `Intervals`.

Translated to the Road Construction problem, this means that if a given goal length is schedulable, the longest unconstructed road segment (or available time window, if we think of it as job scheduling) is longer than the given goal. This, in turn, means that we have still not reached our goal.

days/5

The function `days(Length, Intervals, Goal, IntervalsSoFar, Day)` tries to schedule `Goal` given `Length` and `IntervalsSoFar`. If this fails, `Day` is the correct answer (since we start a zero and add one day at a time). If it succeeds, the longest unconstructed road segment is longer than `Goal`, so we add the next interval from `Intervals` and try again with `Day + 1`.

days/3

The function `days(Length, Intervals, Goal)` simply calls `days/5` with an empty list of `IntervalsSoFar` and a `Day` of zero.

EUnit Testing

EUnit test cases for `days/3` are taken from the example in the assignment description, plus some other test cases that find corner cases.

There are also test cases for `add_interval/2` and for `is_schedulable/4` where I have tried to find "interesting" cases.

Property-Based Testing

There are two property-based tests for `days/3`, one that checks the result using the function `verify_result/4` and one that checks that the result is the same as the one given by an alternate implementation, `days_alt/3`. There is also a test for `add_interval/2` that checks that the result is sorted and that there are no overlapping intervals.

There are no property-based tests for `is_schedulable/4`. This is partly because I could not think of any simple properties to test, and partly because it is indirectly tested by the test cases for `days/3`. It is also tested using unit tests.

verify_result/4

The function `verify_result(Length, Intervals, Goal, Result)` verifies that `Result` is the correct value of `days(Length, Intervals, Goal)`. It does this by adding the first `Result` intervals from `Intervals`, inverting the result (thereby creating a list of unconstructed road segments), and finally checking that the length of the longest unconstructed road segment is smaller than or equal to `Goal`. To verify that `Result` is the smallest possible, `verify_result/4` also checks that `Result - 1` has a longest unconstructed road segment that is longer than `Goal`.

days_alt/3

The function `days_alt(Length, Intervals, Goal)` is an alternate implementation of `days/3` that uses a simpler but less efficient algorithm. It uses the same helper functions as `verify_result/4` to invert intervals and check the length of the longest unconstructed interval.

Performance Testing

There are two functions available for manual performance testing, `generate_random_intervals/3` and `benchmark/3`. They can be used like this:

```
Intervals = road:generate_random_intervals(1000000, 1000000000, 100000).
road:benchmark(days, [1000000000, Intervals, 100000], 10).
```

On my laptop, the benchmark call above gives a result of about 8000 milliseconds.