S6: Trees

CS1101S AY20/21 Sem 1

Studio 2D

Lee Wei Min

Contents

- Higher order list processing
- Trees

Info

- Not core:
  - Programming language processing
  - CPS

Map

Mapping means applying a given function f element-wise to a given list xs.

The result is a list consisting of the results of applying f to each element of xs.

```
function map(fun, xs) {
    return is_null(xs)
        ? null
        : pair(fun(head(xs)),
               map(fun, tail(xs)));
}
```

Accumulate

```
function list_sum(xs) { // programmed "by hand"
    return is_null(xs)
        ? 0
        : head(xs) + sum(tail(xs));
}

function accumulate(f, initial, xs) {
  return is_null(xs)
    ? initial
    : f(head(xs), accumulate(f,initial,tail(xs)));
}
function list_sum(xs) {  // using accumulate
    return accumulate( (x, y) => x + y, 0, xs);
}
```

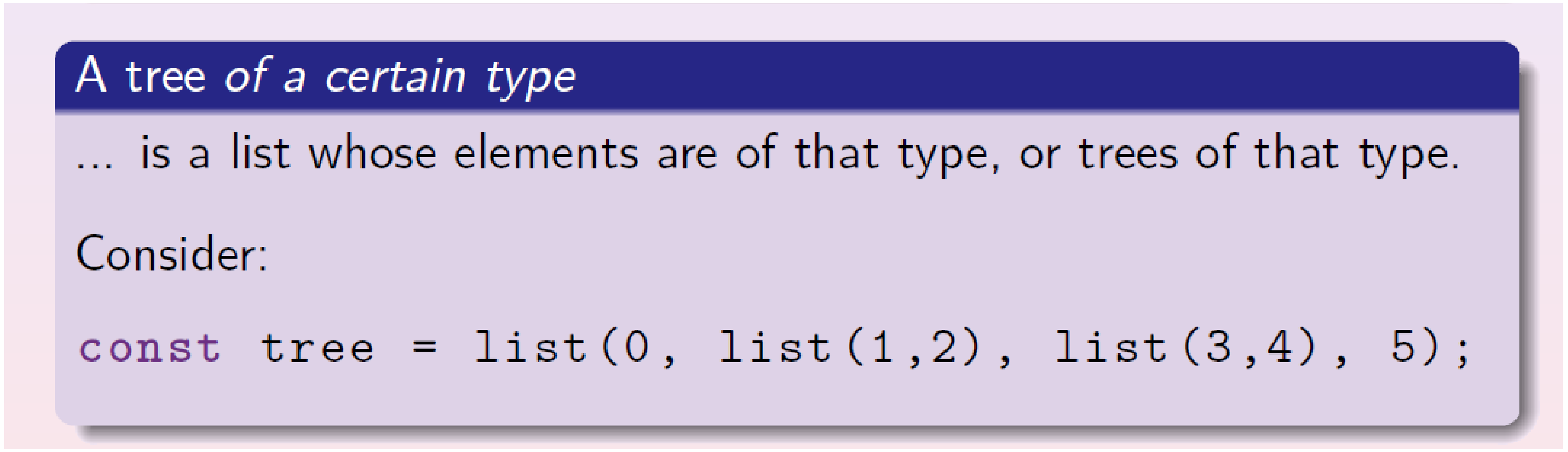
Filter

Problem: take only even elements of list of Numbers

```
filter(x => x % 2 === 0, list(1, 2, 3, 4, 5, 6));

function filter(pred, xs) {
  return is_null(xs)
         ? xs
         : pred(head(xs))
           ? pair(head(xs),
                  filter(pred, tail(xs)))
           : filter(pred, tail(xs));
}
```

Trees



A tree *of a certain type*

... is a list whose elements are of that type, or trees of that type.

Consider:

```
const tree = list(0, list(1,2), list(3,4), 5);
```

Trees

```
function scale_tree(tree, factor) {
    return map(sub_tree =>
                   ! is_list(sub_tree)
                   ? factor * sub_tree
                   : scale_tree(sub_tree, factor),
               tree);
}
```

Trees: Map

```
function map_tree(f, tree) {
    return map(sub_tree =>
                   ! is_list(sub_tree)
                   ? f(sub_tree)
                   : map_tree(f, sub_tree),
               tree);
}
```

Trees: Counting

```
function count_data_items(tree) {
  return is_null(tree)
         ? 0
         : ( is_list(head(tree))
             ? count_data_items(head(tree))
             : 1 )
           +
           count_data_items(tail(tree));
}
```