CS 240

Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

November 18, 2011

Patterns of Recursion

Definition (Tail Recursion)

A recursive call is in the tail position if it is the return value of the method. If all calls are in the tail position, a method is said to be tail-recursive.

Example (Length)

Our recursive version of `length` was not tail-recursive.

```
public int length() { return length(head); }

private int length(Node<E> current) {
    if (current == null)
        return 0;
    return 1 + length(current.link);
}
```

Patterns of Recursion

Definition (Tail Recursion)

A recursive call is in the tail position if it is the return value of the method. If all calls are in the tail position, a method is said to be tail-recursive.

Example (Tail-Recursive Length)

```
public int length() { return length(head, 0); }

private int length(Node<E> current, int total) {
    if (current == null)
        return total;
    return length(current.link, total + 1);
}
```

Patterns of Recursion

Definition (Fold)

A fold is a recursive way to replace the "structural" components of a data structure with desired functions and values. Also known as *reduce*, *accumulate*, *compress*, or *inject*.
Folds may either be left-associative or right-associative.

Example (Right Fold)

The linked list `( 1 2 3 )` can be built up by

```
new Node<Integer>(1,
  new Node<Integer>(2,
    new Node<Integer>(3, null)))
```

We can think of a right fold as replacing the `new Node<Integer>`s with a specific function, and `null` with a specific value.

Iterative Folds

Left/right folds over linear sequences (like linked lists) can be understood as the following iterative patterns

Example (Left Fold)

```
E accum = /* initial value */;

for(/* each element from left-to-right */) {
    accum = f(accum, element);
}
```

Iterative Folds

Left/right folds over linear sequences (like linked lists) can be understood as the following iterative patterns

Example (Right Fold)

```
E accum = /* initial value */;

for(/* each element from right-to-left */) {
    accum = f(element, accum);
}
```

Recursive Folds

Recursively, left/right folds over linked lists have the following forms

Example (Right Fold)

```
E some_right_fold(Node<E> xs) {
    if (xs == null) return /* initial value */;
    return f(xs.data, some_right_fold(xs.link));
}
```

Recursive Folds

Recursively, left/right folds over linked lists have the following forms

Example (Left Fold)

```
E some_left_fold(Node<E> xs, E accum) {
    if (xs == null) return accum;
    return some_left_fold(xs.link,
                          f(accum, xs.data));
}
```