Programming Paradigms

Control Flow (Part 5)

Prof. Dr. Michael Pradel
Software Lab, University of Stuttgart
Summer 2020
Overview

- Expression Evaluation
- Structured and Unstructured Control Flow
- Selection
- Iteration
- Recursion
Recursion

- Equally powerful as iteration
- Most PLs allow both recursion and iteration
  - **Iteration**: More natural in imperative PLs (because the loop body typically updates variables)
  - **Recursion**: More natural in functional PLs (because the recursive function typically doesn’t update any non-local variables)
Efficiency

Naively written or naively compiled recursive functions: Less efficient than equivalent iterative code

- Reason: New allocation frame for each call
- Example: Compute \( \sum_{low \leq i \leq high} f(i) \) in Scheme

```scheme
(define sum
  (lambda (f low high)
    (if (= low high)
        (f low)
        (+ (f low) (sum f (+ low 1) high))))
)```
Efficiency

Naively written or naively compiled recursive functions: **Less efficient** than equivalent iterative code

- Reason: New allocation frame for each call
- Example: Compute \( \sum_{low \leq i \leq high} f(i) \) in Scheme

```scheme
(define sum
  (lambda (f low high)
    (if (= low high)
        (f low)
        (+ (f low) (sum f (+ low 1) high))))
)
```

Then and else branches
Efficiency

Naively written or naively compiled recursive functions: Less efficient than equivalent iterative code

- Reason: New allocation frame for each call
- Example: Compute \( \sum_{low \leq i \leq high} f(i) \) in Scheme

\[
\text{(define sum}
\begin{align*}
&\quad \text{(lambda (f low high)} \, \\
&\quad \quad \text{(if (= low high)} \, \\
&\quad \quad \quad (f low)} \, \\
&\quad \quad \quad (+ (f low) (sum f (+ low 1) high))))
\end{align*}
\text{)}
\]
Tail Recursion

Recursive call is the last statement before the function returns

- Compiled code can reuse same allocation frame
- Revised example:

```scheme
(define sum
  (lambda (f low high subtotal)
    (if (= low high)
      (+ subtotal (f low))
      (sum f (+ low 1) high (+ subtotal (f low))))))
```
Example: Summation
(sum \ f \ 2 \ 4)
Naive implementation

(sum \ f \ 2 \ 4 \ 0)
Tail-recursive implementation

\[
\begin{align*}
\text{sum} \\
& \text{if } f = f \\
& \text{low } 4 \\
& \text{high } 4 \\
\text{sum} \\
& \text{if } f = f \\
& \text{low } 3 \\
& \text{high } 4 \\
\text{sum} \\
& \text{if } f = f \\
& \text{low } 2 \\
& \text{high } 4
\end{align*}
\]
\rightarrow 3 \text{ allocation frames}

\[
\begin{align*}
\text{sum} \\
& \text{if } f = f \\
& \text{low } 2 \ 3 \ 4 \\
& \text{high } 4 \\
& \text{subtotal } 0
\end{align*}
\]
\rightarrow\text{using a single allocation frame}
Overview

- Expression Evaluation
- Structured and Unstructured Control Flow
- Selection
- Iteration
- Recursion