Compost

September 20, 2023

```
val name_email_map : (string * string) list =
[("Roger Burtonpatel", "roger.burtonpatel@tufts.edu");
 ("Randy Dang", "randy.dang@tufts.edu");
 ("Jasper Geer", "jasper.geer@tufts.edu");
 ("Jackson Warhover", "jackson.warhover@tufts.edu")]
```

1 Introduction

Here, introduce our readers to the Compost language and give a brief overview on how it may be used.

2 Compost Features

List out 2-3 interesting features of the language.

3 Code Example

```
(define-datatype list   ;; example datatype definition
    [cons   ;; the ‘cons‘ tag is both the name of the function that introduces
            ;; a value of this variant and the tag used during pattern matching
        (: car int)     ;; the tag is followed by type annotations of the form
                        ;; (: <field-name> <type>)
        (: cdr list)
    ]
    [nil]
)

(: concat (-> (list list) list))    ;; a top-level type annotation
                                                ;; giving a type for ‘concat‘
(define concat (xxs ys)
    (match xxs  ;; a pattern match. ‘xxs‘ is now considered out-of-scope
        [(cons x xs)    ;; a pattern. this deallocates ‘xxs‘’s top level ‘cons‘
                        ;; and binds ‘x‘ to its ‘car‘ and xs to its ‘cdr‘
            (cons x (concat xs ys)) ;; ‘cons‘ is used to introduce a ‘(list a)‘,
        ]                           ;; corresponding to an allocation
        [(nil) ys]      ;; in this branch, the ‘nil‘ is deallocated,
                        ;; resulting in the complete destruction of ‘xxs‘
    )
)

(: filterge (-> (int list) list)
(define filter (n xxs)
    (match xxs  ;; a pattern match. ‘xxs‘ is now considered out-of-scope
        [(cons x xs)    ;; deallocates ‘xxs‘ top level ‘cons‘ and binds
                        ;; ‘x‘ to its ‘car‘ and ‘xs‘ to its ‘cdr‘
            (if (>= x n)    ;; x is a primitive value,
                            ;; so does NOT go out-of-scope
                (cons x (filterge n xs))    ;; we allocate a ‘cons‘
                (filter p xs)
            )
        ]
        [(nil) (nil)]
    )
)

(: filterlt (-> (int list) list)
(define filter (n xxs)
    (match xxs  ;; a pattern match. ‘xxs‘ is now considered out-of-scope
        [(cons x xs)    ;; deallocates ‘xxs‘ top level ‘cons‘ and binds
                        ;; ‘x‘ to its ‘car‘ and ‘xs‘ to its ‘cdr‘
            (if (< x n)     ;; x is a primitive value,
```

```
                            ;; so does NOT go out-of-scope
                (cons x (filterlt n xs))    ;; we allocate a `cons`
                (filter p xs)
            )
        ]
        [(nil) (nil)]
    )
)

(: quicksort (-> list list))
(define quicksort (xxs)
    (match xxs  ;; match expression moves `xxs` goes out of scope
        [(nil)  ;; in this match, we deallocate a `nil`
            (nil)   ;; here, we allocate a new `nil`
        ]
        [(cons x xs)    ;; in this match, we deallocate a `cons`,
                        ;; bind its `car` to `x` and bind its `cdr`
                        ;; to `xs`
            (let*
                (
                    [lesser
                        (filterlt
                            (x (dup xs))  ;; `dup` creates a deep copy of
                                          ;; `xs` so it does NOT go out-of-scope
                        )
                    ]
                    [greater (filterge (x xs))]   ;; `xs` is consumed here and thus
                                                  ;; goes out-of-scope
                )
                (concat
                    (quicksort lesser)
                    (cons p (quicksort greater))    ;; allocate a `cons`
                )
            )
        ]
    )
)
```

Note: Some example proposals interleave features with mini-code examples of that feature. We may want to consider that.

4 Additional Sections

Additional sections may include unique challenges for our language, background information, etc.