Compost: Language Reference Manual

October 18, 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

Briefly explain our language and the LRM.

2 Notation Conventions

Describe how to read notation in manual (how we are formatting prose vs. code vs. grammar stuff)

3 Syntax and Semantics

Describe syntax and semantics of each "thing" in our language, including a full-fledged grammar.

⟨*def*⟩ ::= (`val` ⟨*name*⟩ ⟨*exp*⟩)
| (`define` ⟨*name*⟩ ({⟨*name*⟩}) ⟨*exp*⟩)
| (`datatype` ⟨*name*⟩ ({⟨*variant*⟩}))
| (`:` ⟨*name*⟩ ⟨*type*⟩)
| (`use` ⟨*filename*⟩)

⟨*variant*⟩ ::= (⟨*name*⟩ ({⟨*type*⟩}))

⟨*type*⟩ ::= (`->` ({⟨*type*⟩}) ⟨*type*⟩)
| `int`
| `char`
| `bool`
| `unit`
| `sym`
| ⟨*name*⟩

⟨*expr*⟩ ::= ⟨*literal*⟩
  | ⟨*name*⟩
  | (`case` ⟨*expr*⟩ ({⟨*case-branch*⟩}))
  | (`if` ⟨*expr*⟩ ⟨*expr*⟩ ⟨*expr*⟩)
  | (`begin` {⟨*expr*⟩})
  | (⟨*expr*⟩ {⟨*expr*⟩})
  | (`let` ({(⟨*name*⟩ ⟨*expr*⟩)}) ⟨*expr*⟩)

⟨*case-branch*⟩ ::= (⟨*pattern*⟩ ⟨*expr*⟩)

⟨*pattern*⟩ ::= (⟨*name*⟩ ({⟨*name*⟩ | _}))
  | _

⟨*literal*⟩ ::= ⟨*int-lit*⟩
  | ⟨*sym-lit*⟩
  | ⟨*bool-lit*⟩
  | `unit`

⟨*int-lit*⟩ ::= token composed only of digits, possibly prefixed with a `+` or `-`.

⟨*bool-lit*⟩ ::= `true` | `false`

⟨*sym-lit*⟩ ::= `'`{⟨*sym-char*⟩}`'`

⟨*sym-char*⟩ ::= any unicode code point other than `'` unless escaped with a \.

⟨*name*⟩ ::= any token that is not an *int-lit*, does not contain a `'` or bracket, and is not one of the reserved words shown in typewriter font

4 Standard Library

Describe what we plan to include in a standard library (initial basis in our case?).