FFBB Language

Bowen Chen (bc2916)

Jianan Yao (jy3022)

Xiaosheng Chen (xc2561)

Joseph Yang (zy2431)

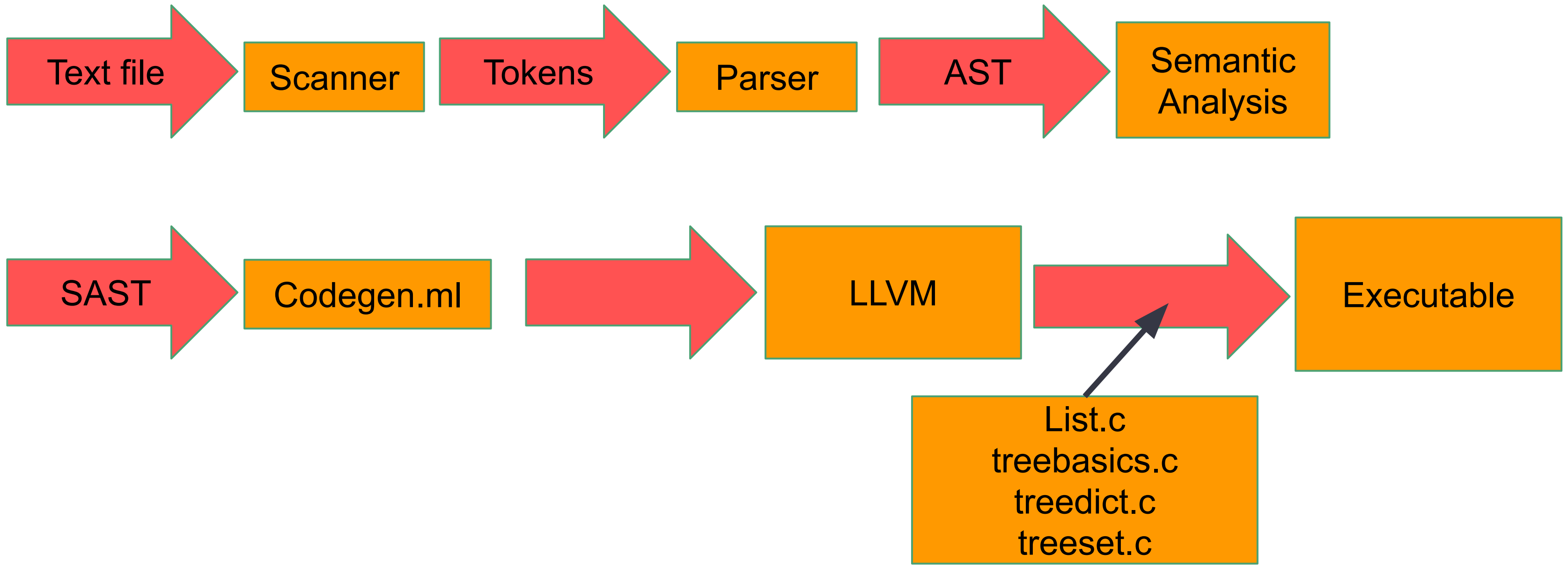
Contents

1. Overview

Motivation

- Based on MicroC, with all the primitive types/operators
- Variable declaration anywhere. Can initialize value at declaration time.
- Primitive type: string
- Data structures: List, Set, Dict
- Function type: func
- Functional programming feature
    - Higher order functions
    - Basic Lambda functions
- for...in… loop

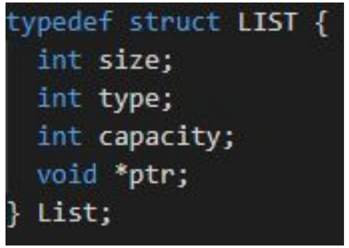
System Architecture



2. Language Features

List

- of a given type: elements all have this type
- List<int> list = [1,2,3,4,5];
- Operations
    - Slice: list[0:-1] return a copy of list
    - append(list, 6);
        - If capacity is not enough, then all original values in the list is copied over to a new memory block with doubled capacity.
    - len(list);
        - Length of the list
    - range(n);
        - Create a list with values 0, 1, 2, …, n-1



```
typedef struct LIST {
  int size;
  int type;
  int capacity;
  void *ptr;
} List;
```

Dictionary

- of two given types: all keys have the first type, all values have the second type
- Dict<int, bool> d;
- Operations
  - dictAdd(d, 10, true)
    - Check if the dictionary d has type Dict<float, bool>, if not, raise an error
    - Adds the key-value pair (10.5, true) to dictionary d
  - dictSize(d)
    - Returns the size of dictionary d
  - dictFind(d, 10)
    - Returns whether the key 10 is in dictionary d
  - dictRemove(d, 10)
    - Try to remove the key 10 from dictionary d
    - No effect if not exist

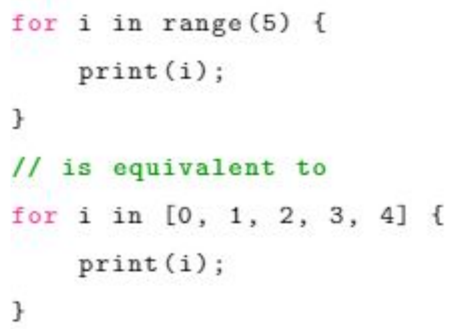
Dictionary

- Operations
  - dictGetBool(d, 10)
  - dictGetInt(d, 10)
  - dictGetFloat(d, 10)
    - Try to get the value associated to key 10 from dictionary d
    - Three different operations depending on the value type of dictionary d
    - If try to execute dictGetBool(d, 10) and dictionary d has type Dict<int, float>, raise an error

Set

- of a given type: elements all have this type
- Set<int> s;
- Operations
    - setAdd(s, 10.5)
        - Check if the set s has type Set<float>, if not, raise an error
        - Adds 10.5 to set s
    - setSize(s)
        - Returns the size of set s
    - setFind(s, i)
        - Returns whether the item i is in set s
    - setRemove(s, i)
        - Try to remove the item i from set s
        - No effect if not exist

for…in…

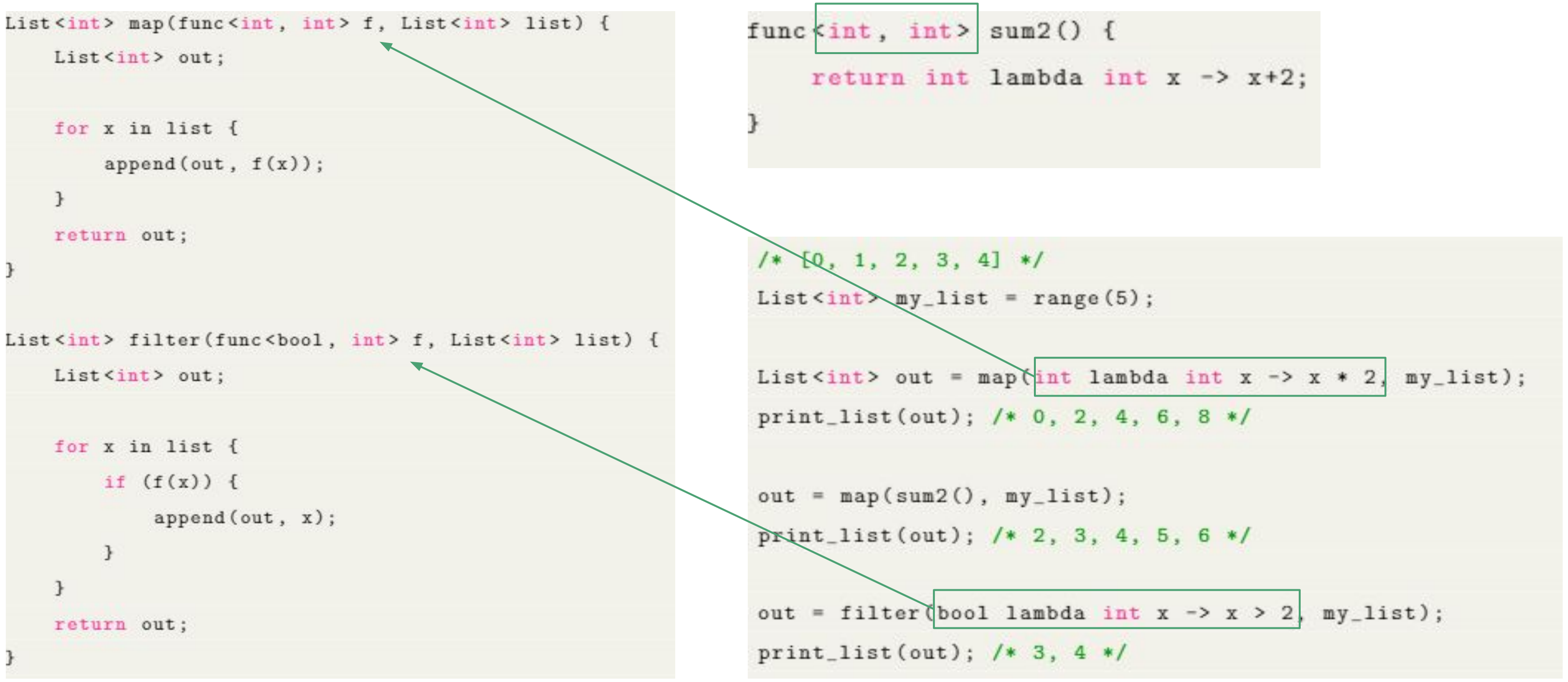
- 

```
for i in range(5) {
    print(i);
}
// is equivalent to
for i in [0, 1, 2, 3, 4] {
    print(i);
}
```

- the type of i depends on the type of list and is inferenced at compile time

Functions

- Function Pointer
  - Of a list of types: first type indicates return type, argument types are the rests.
  - Func<void, int, int>

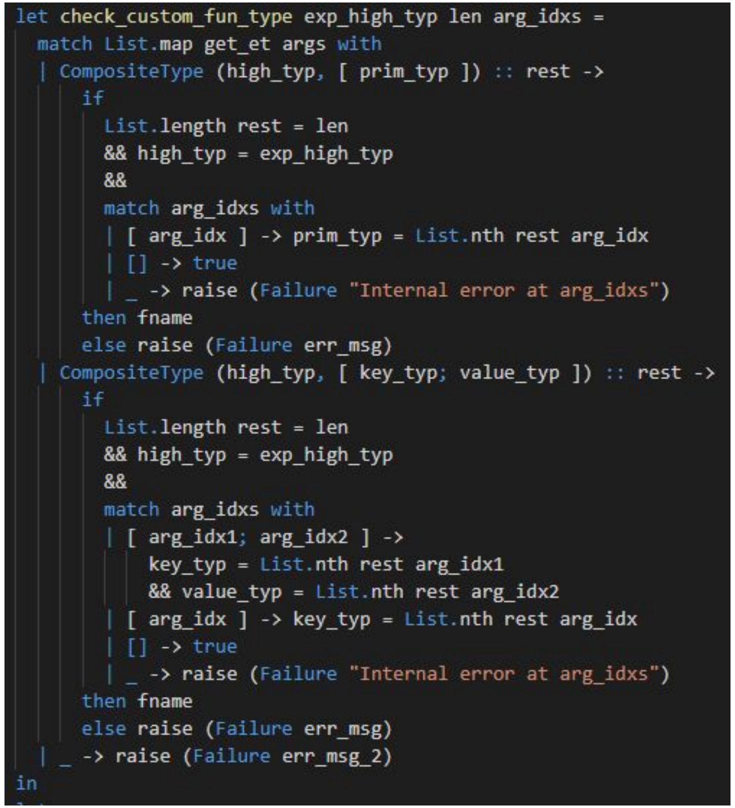


```
List<int> map(func<int, int> f, List<int> list) {
    List<int> out;

    for x in list {
        append(out, f(x));
    }
    return out;
}

List<int> filter(func<bool, int> f, List<int> list) {
    List<int> out;

    for x in list {
        if (f(x)) {
            append(out, x);
        }
    }
    return out;
}
```

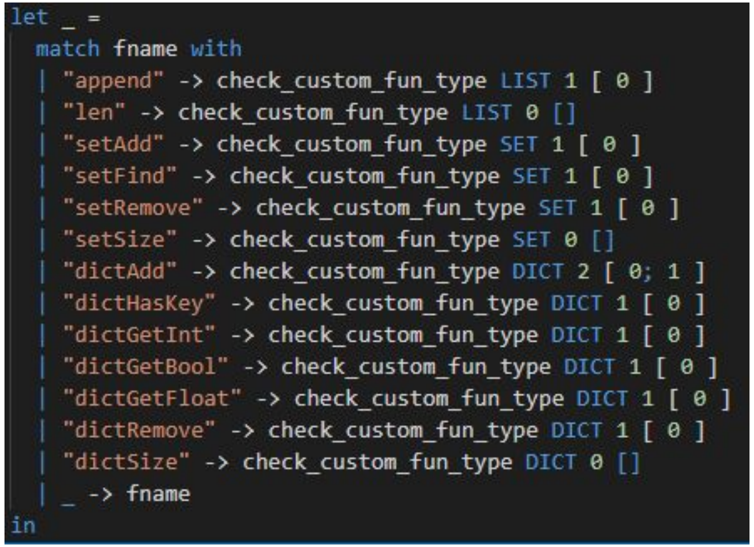
```
func<int, int> sum2() {
    return int lambda int x -> x+2;
}
```

```
/* [0, 1, 2, 3, 4] */
List<int> my_list = range(5);

List<int> out = map(int lambda int x -> x * 2, my_list);
print_list(out); /* 0, 2, 4, 6, 8 */

out = map(sum2(), my_list);
print_list(out); /* 2, 3, 4, 5, 6 */

out = filter(bool lambda int x -> x > 2, my_list);
print_list(out); /* 3, 4 */
```

Semantic Checking



```
let check_custom_fun_type exp_high_typ len arg_idxs =
  match List.map get_et args with
  | CompositeType (high_typ, [ prim_typ ]) :: rest ->
      if
        List.length rest = len
        && high_typ = exp_high_typ
        &&
        match arg_idxs with
        | [ arg_idx ] -> prim_typ = List.nth rest arg_idx
        | [] -> true
        | _ -> raise (Failure "Internal error at arg_idxs")
      then fname
      else raise (Failure err_msg)
  | CompositeType (high_typ, [ key_typ; value_typ ]) :: rest ->
      if
        List.length rest = len
        && high_typ = exp_high_typ
        &&
        match arg_idxs with
        | [ arg_idx1; arg_idx2 ] ->
            key_typ = List.nth rest arg_idx1
            && value_typ = List.nth rest arg_idx2
        | [ arg_idx ] -> key_typ = List.nth rest arg_idx
        | [] -> true
        | _ -> raise (Failure "Internal error at arg_idxs")
      then fname
      else raise (Failure err_msg)
  | _ -> raise (Failure err_msg_2)
in
```



```
let _ =
  match fname with
  | "append" -> check_custom_fun_type LIST 1 [ 0 ]
  | "len" -> check_custom_fun_type LIST 0 []
  | "setAdd" -> check_custom_fun_type SET 1 [ 0 ]
  | "setFind" -> check_custom_fun_type SET 1 [ 0 ]
  | "setRemove" -> check_custom_fun_type SET 1 [ 0 ]
  | "setSize" -> check_custom_fun_type SET 0 []
  | "dictAdd" -> check_custom_fun_type DICT 2 [ 0; 1 ]
  | "dictHasKey" -> check_custom_fun_type DICT 1 [ 0 ]
  | "dictGetInt" -> check_custom_fun_type DICT 1 [ 0 ]
  | "dictGetBool" -> check_custom_fun_type DICT 1 [ 0 ]
  | "dictGetFloat" -> check_custom_fun_type DICT 1 [ 0 ]
  | "dictRemove" -> check_custom_fun_type DICT 1 [ 0 ]
  | "dictSize" -> check_custom_fun_type DICT 0 []
  | _ -> fname
in
```

Check high type, prim typ and args type

3. Testing

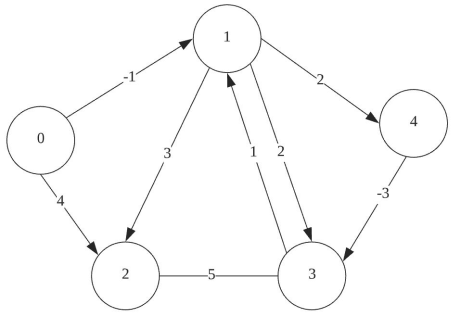
Testing

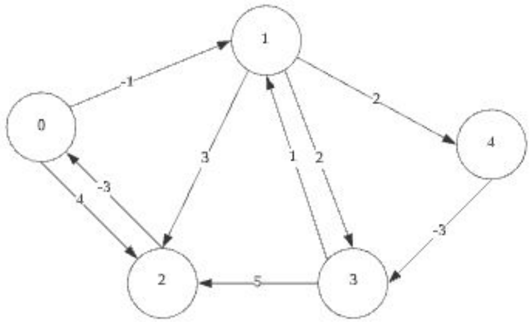
- General testing: shell script
  - `./testall.sh`
  - Automatically run all test suites
- Test cases
  - Inherited from MicroC test suite
  - Added more than 50 test cases
- Stringify statements
  - Implemented string of statements for all AST and SAST types, for clean debugging results

4. Demo

Demo Examples

- Fibonacci
- Quicksort
- Higher Order Functions
- Bellman–Ford Algorithm





Graph with negative cycle

Thanks!