Typescript-on-LLVM

Ratheet Pandya
UNI: rp2707
COMS 4115 H01 (CVN)
Project Proposal

Describe the language that you plan to implement.

I would like to implement a very simplified subset of Typescript (which is statically-typed and compiles down to Javascript) that would work server-side (as opposed to in the browser).

Explain what sorts of programs are meant to be written in your language.

Any basic program that can be run server-side (not in the browser). e.g., computing the Fibonacci sequence, GCD, etc.

Explain the parts of your language and what they do.

The language will support:

- A subset of the Basic Types. Namely: `Boolean`, `Number`, `String`, `Array`, `Tuple`, and `Void`.
- `null` values
- `let` and `const` for declaring scoped names
- String concatenation
- Block scoping via `let`
- Functions
- `while` loops
- `if-else` conditionals
- (nice-to-have) Basic Classes without inheritance - may introduce a new "`struct`" type that is not in the Typescript spec for this. Mainly I would like to support grouping data and functions together. All the bells and whistles of classes are not necessary for this.

The language will not support anything else in the Typescript specification.

Some things worth calling out - it will NOT support:

- Browser builtins: as far as I know, compiling Typescript down to LLVM out-of-the-box would not give access to browser-specific libraries, so it would not be feasible to support JS browser facilities (like `document` or `window`).
    - There is a library that lets you compile LLVM into JS (Emscripten), but it seems like overkill to try to support this for the project.
- The rest of the Basic Types (`Tuple`, `Enum`, `Any`, `Never`, `undefined`):

- The Advanced Types
- Optional parameters, default-initialized parameters
- Interfaces, Class inheritance

Include the source code for an interesting program in your language.

Here's how one could compute GCD (imperative-style) in Typescript-on-LLVM:

```
function gcd(a: number, b: number): number {
  while (a != b) {
    if (a > b) {
      a -= b;
    } else {
      b -= a;
    }
  }
  return a;
}
```

Here's a GCD example using recursion:

```
function gcd(a: number, b: number): number {
  if (a == b) {
    return a;
  }
  if (a > b) {
    return gcd(a - b, b);
  } else {
    return gcd(a, b - a);
  }
}
```

Finally, a more ambitious example using simple `struct`-like classes to build a binary-search tree:

```
class Node {
  value: string;
  left: Node;
  right: Node;
  constructor(v: string, l: Node; r: Node) {
    this.value = v;
  }
}
```

```
let bst = Node('C',
               Node('B',
                    Node('A', null, null),
                    null),
               Node('D',
                    null,
                    Node('E', null, null)));
```