Team 11

Alireza Bahremand
Cecilia La Place
Joshua Hewlett
Paul Horton

Language: AH-J

This language is modeled after day-to-day communication methods.

- Inspired by the in-class communication between Dr. Bansal and Xiangyu
- Reformatted to create an epistolary language
- Strongly typed to enforce security & maintainability of the language.

This language is best for those with an understanding of:

- English grammar
- Email/letter writing

Grammar

AH-J follows a pattern-like structure that is representative of the English language.

Commands are portrayed as common English-phrase-directives.

Declarations are conveyed as statements separated by periods.

Structure	AH-J Syntax
Program opening block,	Salutations Xiangyu,
Open body code block	Would you mind doing the following:
Declaration Command Command	Create the variable n. Assign the integer n to the value of 1. Assign the integer n to the value of n + 3.
Close body code block	Thank you.
Program closing block.	Sincerely, Ajay Bansal

Grammar

The grammar of AH-J is comprised of 2 entities, *declarations* & *commands*.

- Declarations initialize variables for the runtime environment.

Commands perform logical & arithmetic operations on variables.

- Commands include conditional statements, while loops, expressions, and nested blocks of code.

AH-J Syntax	Code Comparison

Salutations Xiangyu,
  Would you mind doing the following:
  Create the variable soft.
  Assign the boolean foft to the value of 0.
  Should it be the case foft EQUALS 0 please
  Would you mind doing the following:
  Assign the integer soft to the value of 4 * 3.
  Thank you.
  otherwise
  Would you mind doing the following:
  Please reply with the value of foft.
  Thank you.
  that is all.
  Thank you.
Sincerely, Ajay Bansal
	Salutations Xiangyu,
	{
	  body {
	  var soft;
	  var foft = 0;
	  if (foft == 0) {
	  soft = 4 * 3;
	}
	  else {
	  print(foft);
	}
	  }
	}

Compiler: Parser

Using Prolog, AH-J is first tokenized by checking for

- Spaces
- Newlines

AH-J is then parsed using Prolog's DCG functionality in order to ensure

- Enforcement of type security
- Syntax validation

The parse tree is then generated in the event of successful compilation.

DCG

```
program(t_prog(K)) --> ["Salutations", "Xiangyu,"],
  list(K),  ["Sincerely,", "Ajay", "Bansal"].

list(t_list(D, C)) -->
  ["Would", "you", "mind", "doing", "the", "following:"],
  declaration(D), block_command(C), ["Thank", "you"], [","];
  ["Would", "you", "mind", "doing", "the", "following:"],
  declaration(D), command(C), [","], ["Thank", "you"], [","].

list(t_list(C)) -->
  ["Would", "you", "mind", "doing", "the", "following:"],
  block_command(C), [","], ["Thank", "you"], [","];
  ["Would", "you", "mind", "doing", "the", "following:"],
  command(C), [","], ["Thank", "you"], [","].
```

Intermediate Code: Parse Tree

AH-J Syntax	Parse Tree
Salutations Xiangyu, Would you mind doing the following: Assign the integer c to the value of 7. So long as NOT c EQUALS 5 please Would you mind doing the following: Assign the integer c to the value of c - 1. Thank you. your iterations are appreciated. Thank you. Sincerely, Ajay Bansal	t_prog(t_list(t_command(t_id(c),t_term(t_factor(t_num(7))), t_block_cmnd(t_while(t_exp_not(t_exp_eq(t_term(t_factor(t_id(c))),t_term(t_factor(t_num(5))))), t_list(t_command(t_id(c),t_minus(t_term(t_factor(t_id(c))),t_term(t_factor(t_num(1))))))))))

Runtime: Interpreter

The interpreter for AH-J is written in Prolog and takes the parse tree as input and uses the parameters of each node of the parse tree to perform the desired functionality in Prolog.

The control structure of the final program is determined by the sequence in the parse tree.

Conditional blocks, such as while and if, are accomplished by evaluating boolean expressions and executing the correct data path.

Environment variables are collected through the execution and are output at the end.

Screenshot

interpreter.pl | testCase4.ahj | testCase5.ahj

```
Salutations Xiangyu,
    Would you mind doing the following:

        Create the variable flomo.
        Create the variable pomo.
        Create the variable somo.
        Assign the boolean flomo to the value of 0.
        Assign the integer pomo to the value of 1233.

        Should it be the case flomo EQUALS 1 please
            Would you mind doing the following:
                Assign the integer somo to the value of 4 * 3 - 2 + 3 / 2.
                Please reply with the value of somo.
            Thank you.
        otherwise
            Would you mind doing the following:
                Should it be the case 4 + 2 EQUALS 6 please
                    Would you mind doing the following:      You, 7 hours ago • Nested
                        Assign the integer somo to the value of 8 * 4 + 4 * 2 / 1.
                        Please reply with the value of somo.
                    Thank you.
                otherwise
                    Would you mind doing the following:
                        Please reply with the value of pomo.
                    Thank you.
                that is all.
            Thank you.
        that is all.

    Thank you.
Sincerely, Ajay Bansal
```

1: bash

```
┌─[Reza][~/Desktop/College/Senior/SER502/Project/SER502-Spring
2018-Team-11/doc/TestPrograms]
└──╼ ../../src/ahj.sh -f testCase5.ahj -k
../../src/ahj.sh: line 102: kill: (30441) - No such process

Program output:
"somo"=40
[(somo,40),(pomo,1233),(flomo,0)]
┌─[Reza][~/Desktop/College/Senior/SER502/Project/SER502-Spring
2018-Team-11/doc/TestPrograms]
└──╼ ls ahjc/
_out
testCase5.ahj.betterparse
testCase5.ahj.output
testCase5.ahj.parse
testCase5.ahj.tokens
┌─[Reza][~/Desktop/College/Senior/SER502/Project/SER502-Spring
2018-Team-11/doc/TestPrograms]
└──╼ cat ahjc/testCase5.ahj.parse
t_prog(t_list(t_decl(t_id(flomo),t_decl(t_id(pomo),t_decl(t_id
(somo)))),t_command(t_id(flomo),t_term(t_factor(t_num(0))),t_c
ommand(t_id(pomo),t_term(t_factor(t_num(1233))),t_block_cmnd(t
_if(t_exp_eq(t_term(t_factor(t_id(flomo))),t_term(t_factor(t_n
um(1)))),t_list(t_command(t_id(somo),t_minus(t_mult(t_factor(t
_num(4)),t_term(t_factor(t_num(3)))),t_plus(t_term(t_factor(t_
num(2))),t_div(t_factor(t_num(3)),t_term(t_factor(t_num(2)))))
),t_command(t_print(t_id(somo))))),t_list(t_block_cmnd(t_if(t_
exp_eq(t_plus(t_term(t_factor(t_num(4))),t_term(t_factor(t_num
(2)))),t_term(t_factor(t_num(6)))),t_list(t_command(t_id(somo)
,t_plus(t_mult(t_factor(t_num(8)),t_term(t_factor(t_num(4)))),
t_mult(t_factor(t_num(4)),t_div(t_factor(t_num(2)),t_term(t_fa
ctor(t_num(1))))))),t_command(t_print(t_id(somo))))),t_list(t_c
ommand(t_print(t_id(pomo))))))))))))
┌─[Reza][~/Desktop/College/Senior/SER502/Project/SER502-Spring
2018-Team-11/doc/TestPrograms]
└──╼
```

Compiling language for parsing and interpreting