UNIVERSITY OF AMSTERDAM
Informatics Institute
Systems and Networking Lab
Parallel Computing Systems Group

Compiler Construction
2019/20
Dr. Clemens Grelck
February 4, 2020

Assignment 1

Elementary Programming in CiviC
Familiarisation with Compiler Framework

The first part of this assignment is supposed to familiarise yourself with the CiviC model language and to provide you with a test suite of example programs for your own CiviC compiler. The second part is intended to deepen your understanding of the compiler construction framework to be used throughout the course.

Assignment 1.1: CiviC Core Programming

Implement a CiviC module `core.cvc` that exports the following functions:

- `gcd(`*a*`,`*b*`)`
  returns the greatest common denominator of *a* and *b*;
- `fac(`*n*`)`
  returns the factorial of *n*;
- `fib(`*n*`)`
  returns the *n*-th Fibonacci number;
- `isprime(`*n*`)`
  returns `true` if *n* is a prime number and `false` otherwise.

Assignment 1.2: CiviC Nested Functions and I/O

Implement a CiviC module `coreio.cvc` that exports the following functions:

- `fibs(`*n*`)`
  print first *n* Fibonacci numbers;
- `primes(`*n*`)`
  print first *n* prime numbers.

The function `fibs` must make use of the function `fib` from the `core.cvc` module. In contrast, the function `primes` shall have a clone of function `isprime` as a nested local function definition.

Assignment 1.3: CiviC Arrays

Implement a CiviC module `array.cvc` that exports the following functions:

- `printIntVec( int[`*n*`] `*vec*`)`
  `printFloatVec( float[`*n*`] `*vec*`)`
  print *vec* to stdout;
- `printIntMat( int[`*m*`,`*n*`] `*mat*`)`
  `printFloatMat( float[`*m*`,`*n*`] `*mat*`)`
  print *mat* to stdout;
- `scanIntVec( int[`*n*`] `*vec*`)`
  `scanFloatVec( float[`*n*`] `*vec*`)`
  scan *vec* from stdin;
- `scanIntMat( int[`*m*`,`*n*`] `*mat*`)`
  `scanFloatMat( float[`*m*`,`*n*`] `*mat*`)`
  scan *mat* from stdin;
- `matMul( float[`*m*`,`*n*`] `*a*`, float[`*o*`,`*p*`] `*b*`, float[`*q*`,`*l*`] `*c*`)`
  multiply two floating point matrices *a* and *b* and store result in *c*;
- `queens( bool[`*m*`,`*n*`] `*a*`)`
  solve the well known 8-Queens problem (bonus challenge).

Note:

In the absence of characters and character strings in CiviC, your formatting options are very limited. Make the best out of it.

Note:

All above CiviC modules must not export a `main` function. For testing purposes write separate modules that do contain `main` functions with suitable test code and submit them alongside.

Assignment 1.4: Familiarisation with Compiler Construction Framework

Implement a compiler traversal that counts the number of occurrences of each of the five arithmetic operators. The `info` structure shall be used to collect the information. The use of global variables is not permitted.

Unlike in the toy traversal provided, the information shall be stored in the root node of the syntax tree at the end of the traversal. For this purpose add a dedicated root node (named *module*) to your syntax tree. This root node shall contain a sequence of statements (as already given) plus the corresponding attributes to store the inferred information.

Extend the pretty print traversal such that it produces appropriate output for the new root node, including the inferred information.

Add the traversal as an additional compiler phase.

Assignment 1.5: Familiarisation with Compiler Construction Framework

Implement a compiler traversal that counts the number of occurrences of each identifier (left or right hand side). Again, the `info` structure shall be used to collect the information, and the use of global variables is not permitted.

In contrast to Assignment 1.4 you do not a-priori know the number of different counters needed, which makes this assignment truly different.

Instead of a fixed set of counters, make use of the lookup table (or map) provided with the framework. These lookup tables can store associations between character strings (or pointers) and heap-allocated objects, e.g. counters.

Print the inferred information at the end of the traversal, i.e. inside the root node handler. Do not store the lookup table in the abstract syntax tree.

Add the traversal as an additional compiler phase.

Assignment due date: Friday, February 14, 2020