Stella: Status Update

David Mohr

2014-06-27

Overall State

- Basics are done
- Interesting stuff is not
  - No OO, no structs
- Can run simple, unstructured benchmark
  - Fast, too
- ~2k SLOC
  - 600 SLOC tests

Main work

- Stack -> Register
- Semantics of bytecodes
- Function calls
    - Calling C libraries
- Global variables
- (numpy) arrays

Bumps along the way

llvmpy and LLVM * Error messages are often bad! * Sometimes on errors LLVM simply calls abort() * Unidentified crashes with the default LLVM engine

Avoiding pitfalls

- Use registers, not stack locations
    - Complicates logic
    - Unecessary because of optimization passes

Requirements for compilation

- Based on *types*, not values*
  - Important for semantics, too

Convenience features

- Default values, keyword arguments

```
def f(x, foo=0, bar=2):
    ...
f(5)
f(6, foo=2)
```

Convenience features (2)

Transparent access to C libraries by using cython

- Wrap C function with identical Python names
- In STELLAcall C function
- Needs type annotations (missing cython feature?)

Open issues: SSA dominators

```
while obs_i < K and t < rununtiltime:
    if leg < substrate:
        R = koffp
    else:
        R = kcat
    t += exp(R)
```

(So far) no copying back from STELLA

- Global scalar changes not reflected in Python
- Native "structs" will need this
    - Cost should be acceptable
- Arrays are directly updated (no copying)

Benchmark 1: fibonacci

- Side note: gcc much slower than clang, even at O3
- Elapsed C: *18.13s*
- Elapsed Stella: *10.06s*
- Speed-Up: *1.80*
- (Stella+Compile: 10.09s)

Benchmark 2: 1D one-legged spider walking

- Elapsed C: *57.56s*
- Elapsed Stella: *60.08s*
- Speed-Up: *0.96*
- (Stella+Compile: 60.14s)

Next tasks

- structs
- rewrite benchmark with structs