Lecture 11: Synthesis with Stochastic Search

Yu Feng
Spring 2021

Summary of previous lecture

- 2nd homework is due today
- Synthesis with machine learning

Performance

- Many systems where code performance matters
  - Compute-bound
  - Repeatedly executed
- Scientific computing
- Graphics
- Low-latency server code
- Encryption/decryption

Superoptimization

Superoptimization

- Generate the optimal implementation
  - Massalin [ASPLOS'87]
    - Enumerate all possible straight-line programs
- Stoke, Schkufza [ASPLOS'13]
  - Random enumeration instead of exhaustive
  - Transforms programs with loops

```
[r8:rdi] = rsi * [ecx:edx] + r8 + rdi
```

```
1  # gcc -O3
2
3  .L0:
4    movq rsi, r9
5    movl ecx, ecx
6    shrq 32, rsi
7    andl 0xffffffff, r9d
8    movq rcx, rax
9    movl edx, edx
10   imulq r9, rax
11   imulq rdx, r9
12   imulq rsi, rdx
13   imulq rsi, rcx
14   addq rdx, rax
15   jae .L2
16   movabsq 0x100000000, rdx
17   addq rdx, rcx
18 .L2:
19   movq rax, rsi
20   movq rax, rdx
21   shrq 32, rsi
22   salq 32, rdx
23   addq rsi, rcx
24   addq r9, rdx
25   adcq 0, rcx
26   addq r8, rdx
27   adcq 0, rcx
28   addq rdi, rdx
29   adcq 0, rcx
30   movq rcx, r8
31   movq rdx, rdi
```

```
1  # STOKE
2
3  .L0:
4    shlq 32, rcx
5    movl edx, edx
6    xorq rdx, rcx
7    movq rcx, rax
8    mulq rsi
9    addq r8, rdi
10   adcq 0, rdx
11   addq rdi, rax
12   adcq 0, rdx
13   movq rdx, r8
14   movq rax, rdi
```

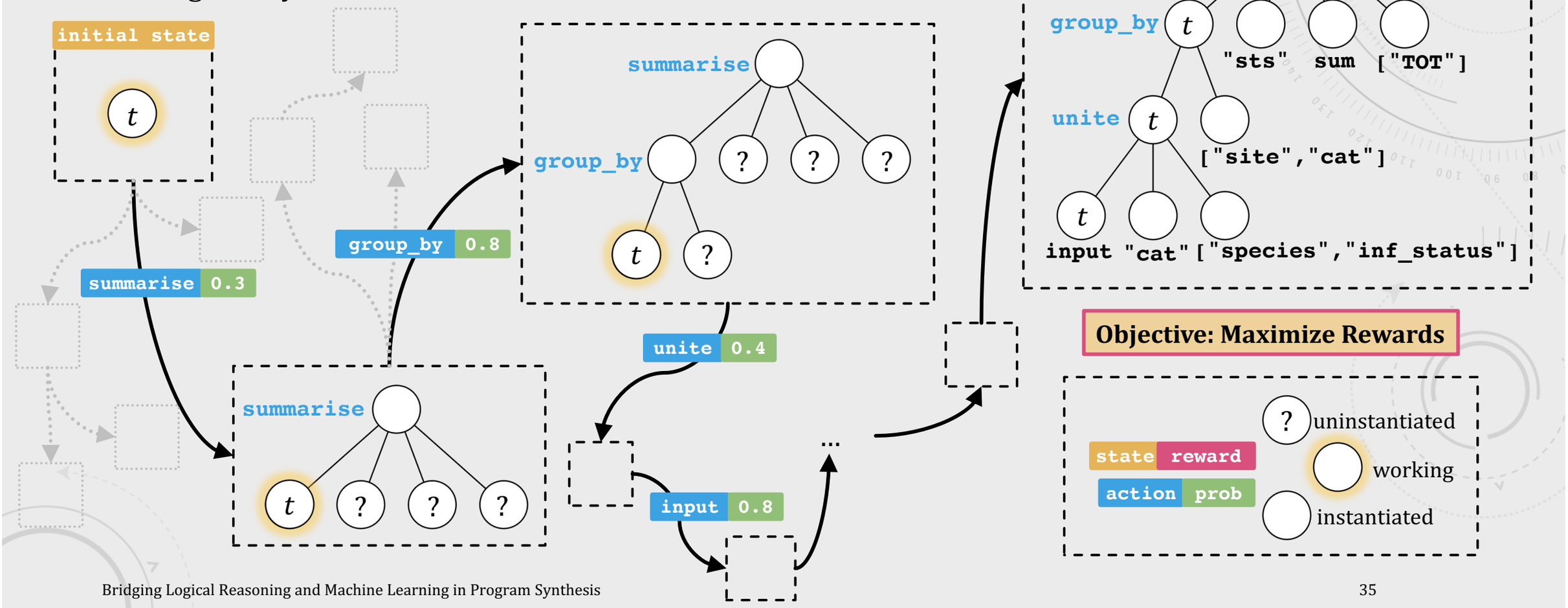
Figure 1. Montgomery multiplication kernel from the OpenSSL big number library, compiled by `gcc -O3` (left) and STOKE (right). The STOKE code is 16 lines shorter, 1.6x faster, and slightly faster than expert handwritten assembly.

The goal is to find a sequence of x86 instructions that are close to optimal for a particular task

https://github.com/StanfordPL/stoke

Markov Chains

- Program Synthesis as Markov Decision Process



Cost function

$$c(\mathcal{R};\mathcal{T}) = \mathrm{eq}(\mathcal{R};\mathcal{T}) + \mathrm{perf}(\mathcal{R};\mathcal{T}) \tag{2}$$

$$\mathcal{R}' = \arg\min_{r} \left(\mathrm{perf}(r;\mathcal{T}) \,\middle|\, \mathrm{eq}(r;\mathcal{T}) = 0 \right) \tag{3}$$

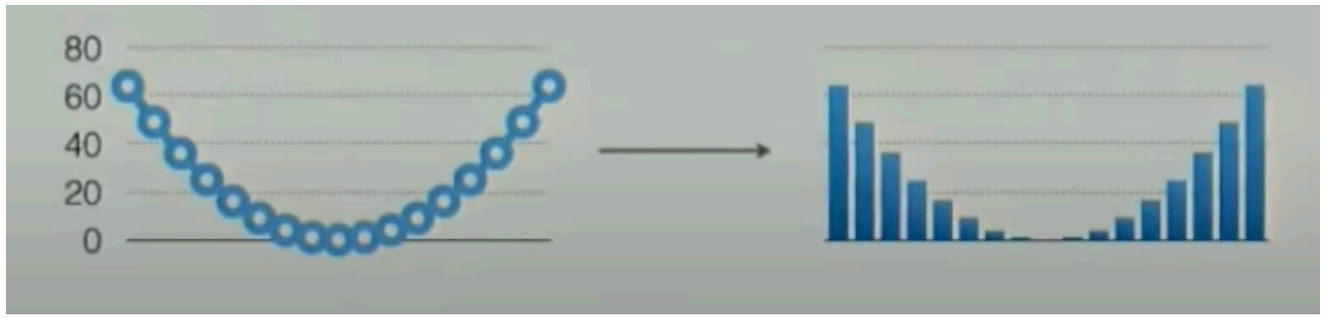
The transformation correctness term, eq(·), measures the similarity of two functions. The term is zero if and only if the two functions are equal. The performance improvement term, perf(·), quantifies the performance improvement of a rewrite with respect to the target. Depending on the application, this term could reflect code size, expected runtime, number of disk accesses, power consumption, or any other measure of resource usage.

Markov chain Monte Carlo (MCMC) sampling

- Select an initial program
- Repeat (millions to billions of times)
  - Propose a random change and evaluate cost
  - If (decreased) {accept}
  - If (increased) {may accept}

MCMC sampling

- Guarantees: Draw samples in proportion to their value; higher value points are sampled more frequently

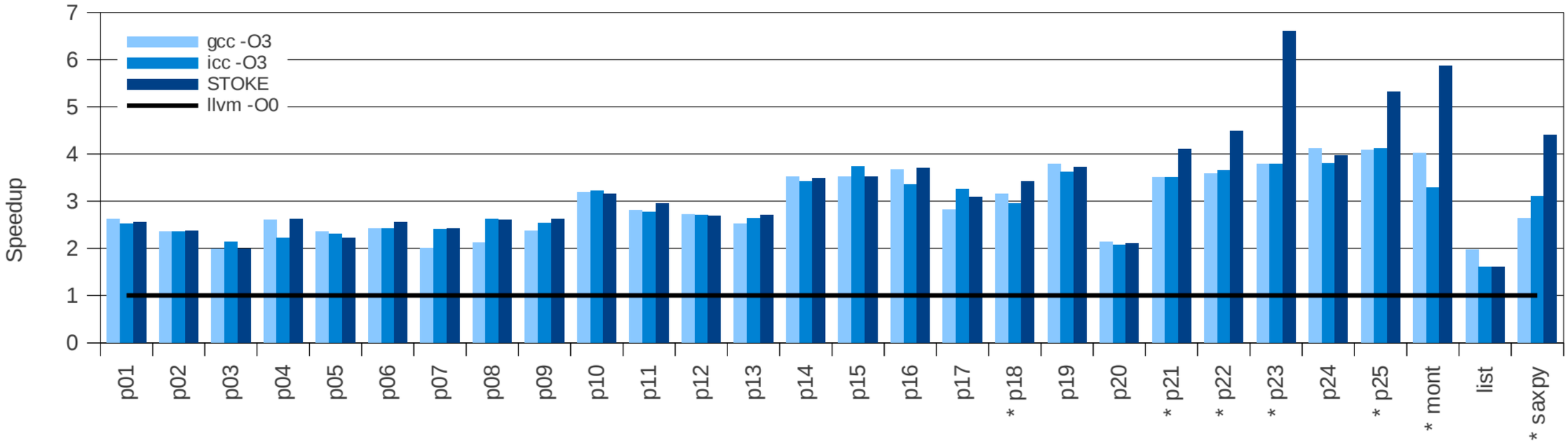


MCMC sampling

- Intelligent “hill climbing” method, robust against local minima, desirable limiting properties (Metropolis–Hastings)

- Nothing prevents the use of other search methods. Offers a tradeoff between performance and complexity

Results



Average speedup over llvm -O0 for benchmark kernels

TODOs by next lecture

- Proposal will be due on Wed
- Start to work on your final report/project! (20%)