CS131: Programming Languages

Fall 2015
Week #3

Today

- HW 2 Clarifications
- Java / HW 3

HW2: Naive Parsing of CFGs

- A parser generator
- Submission due: Oct 16, 11:55 pm

parse_prefix: what to return

- parse_prefix should return Some(derivation, suffix)
  - `parse_prefix gram`
  - `parse_prefix gram accept frags`
- What derivation to generate?
  - Prefer a full derivation? No.
  - Prefer first derivation accepted, full or partial

Left Recursion / Blind Alleys

- Don’t need to handle left recursion

```
A -> A | “hello”
```

- DO need to handle blind alleys
    - But not like those in hw1

How do we handle blind alleys?

Not like in HW1. Here, blind alley in the sense that we eagerly matched everything remain with non terminals

- If there is no alternatives for a non terminal:

```
S → A
A → B      (Blind alley rule)
A → "a"
```

HW3

- Due October 23, 11:55pm
- JMM - Java Memory Model
- Measure performance & reliability of various state models
- Create new state models

Concurrency Issues

- Data races
- Deadlocks

Java Memory Model (JMM)

- final
- volatile
- synchronized

Volatile Keyword

- Initial
  - ready = false
  - answer = 11;
- Thread 1

```
(1)  answer = 22;
(2)  ready = true;
```

- Thread 2

```
(3) if (ready)
(4) println (answer);
```

- Volatile keyword ensures
  - Read and write occurs at the main memory
- Before Java 1.5, (4) may print 11 or 22.
- After Java 1.5, (4) always print 22.

Other entities that can change the memory value

- The other peripheral device
  - memory-mapped operation
- Interrupt service routine
- The other threads

Shared Memory & Swap

- An **array** of integers in shared memory
    - Integers between 0 and 127 → represent as byte

```
byte[] arr = new byte[5]; // default values: 0

byte[] arr = {1,2,3,4,5};
```

- **Swap**: decrement one element, increment another
    - Sum stays the same

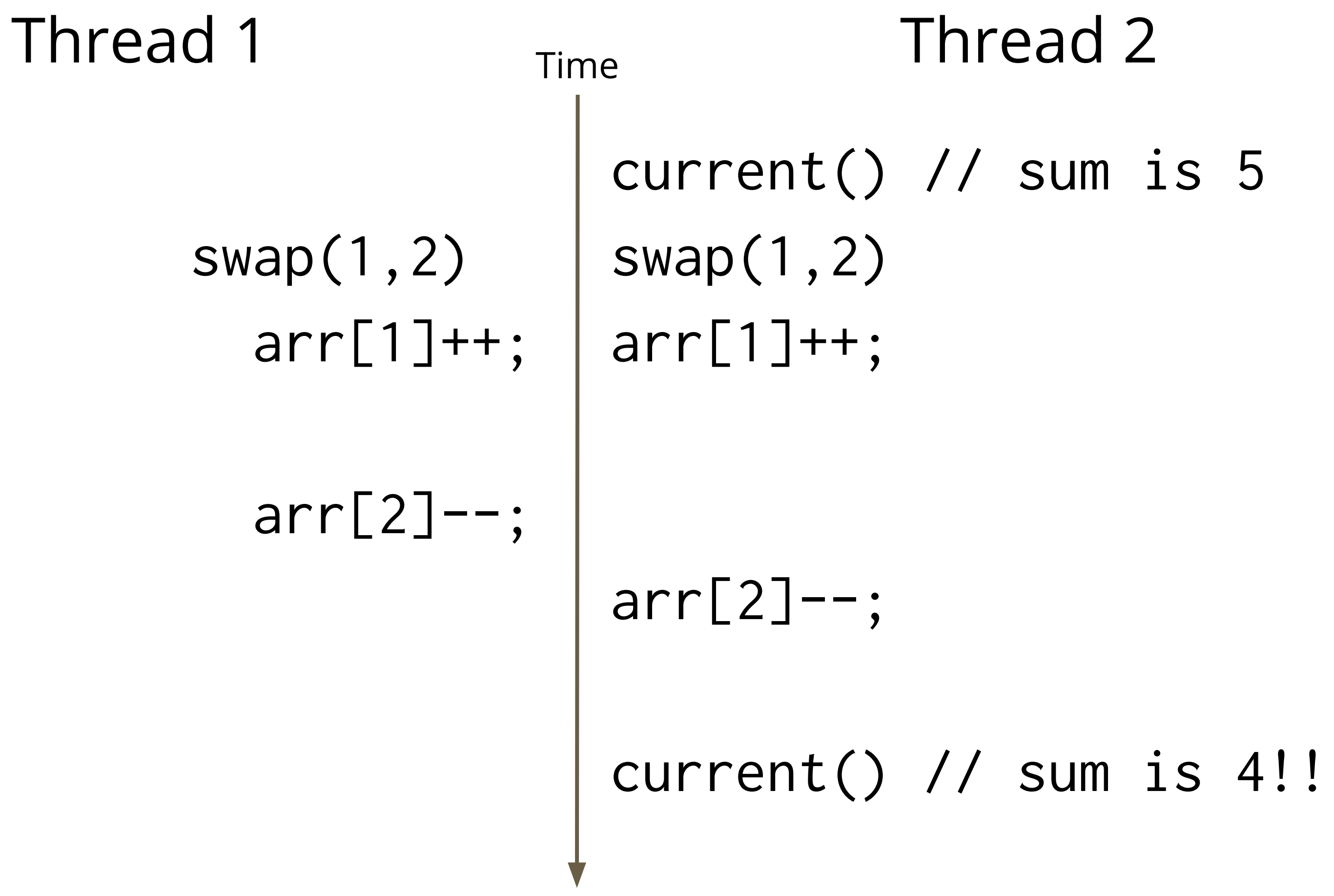
```
arr[0]++; arr[1]--; // arr = {2,1,3,4,5}
```

- Threads will observe a consistent sum
    - Unless there are **data races**

State Model

- Keeps a **reference** to the array in shared memory
- A model is a Java class that implements the `State` interface
    - `public int size()`
        - length of the array
    - `public byte[] current()`
        - See the current state of the array
    - `public boolean swap(int i, int j)`
        - Do the swap operation

Example Data Race



Thread 1	Thread 2
	`current() // sum is 5`
`swap(1,2)`	`swap(1,2)`
`arr[1]++;`	`arr[1]++;`
`arr[2]--;`	
	`arr[2]--;`
	`current() // sum is 4!!`

Expanded Example Data Race

```
           Thread 1            Time           Thread 2
                                 |
                                 |  current() // sum is 5
             load arr[1] -> x    |
                       incr x    |  load arr[1] -> x
            store x -> arr[1]    |
                                 |  incr x
                                 |  store x -> arr[1]
   load/decr/store arr[2]        |
                                 |  load/decr/store arr[2]
                                 v  current() // sum is 4!!
```

This read happens before the other thread’s write!

SynchronizedState

- Makes `swap` a synchronized method
- `public synchronized boolean swap(int i,int j)`
- The first call to `swap` will block all other calls to `swap` until completion

Models to Create

- `UnsynchronizedState` - like `SynchronizedState`, but without the `synchronized` keyword
- `GetNState - volatile array access`
- `BetterSafeState - better than Synchronized in perf`
- `BetterSorryState - even better, but < 100 reliable`

Building & Testing Performance

- javac
- java UnsafeMemory SynchronizedState 8 1000000 10 1 2 3 4 5