Today

- Programming assignment - questions? extension?
- Semaphores in Tuple Space
- Interleavings handout - continued
- Midterm exam: Thursday, October 12th
Semaphores
(TS implementation)

// Semaphore primitives P and V (works for binary and counting sems)
// -- must be implemented over tuples in tuple space

void P(char *s)
{
    in("sem", s:(strlen(s)+1));
}

void V(char *s)
{
    out("sem", s:(strlen(s)+1));
}

So this invocation:
P("empty");

Tries to match a tuple like this:
("sem", "empty")

and this invocation:
V("full");

Will place a tuple in TS that looks like this:
("sem", "full")
Semaphore usage
(binary / counting)

binary initialization:

sem full = 0;
sem empty = 1;

Becomes this in your C-Linda code:

V("empty");

// places a tuple in TS:
// ("sem", "empty")

// do nothing to initialize
// semaphore full = 0...

counting initialization:

sem empty = n;
sem full = 0;

Becomes this in your C-Linda code:

for (i=0, i<n; i++) {
    V("empty");
}

// places n tuples in TS
// that all look like this:
// ("sem", "empty")
Bounded buffer in Tuple Space

// C declaration of a buffer as an array of ints
int buf[n];

// Assignment of three elements to buf
buf[0] = 42;
buf[1] = 43;
buf[2] = 44;

// Equivalent assignment using distributed data
// structure in tuple space...
// Tuples of this form are used:
//
//  ("buf", index, value)
out("buf", 0, 42);
out("buf", 1, 43);
out("buf", 2, 44);

// to access value stored in buf[13]...
int i, value;
i = 13;
rd("buf", 13, ?value);

// to consume same data, change rd() to in()...
in("buf", 13, ?value);
Producer / Consumer
Version 3

Here’s the pseudo code for producer and consumer procs:

```c
//shared variables -- must be implemented in tuple space
int buf[n],
    int front = 0, rear = 0;    // indices to buf
int empty = n, full = 0;    // between producers/consumers
int mutexD = 1,             // between different producers
    mutexF = 1;             // between different consumers

process Producer[i = 1 to M] {
    while (true) {
        // produce data; deposit in buf
        P(empty);
        P(mutexD);
        buf[rear] = data;
        rear = (rear+1)%n;
        V(mutexD);
        V(full);
    }
}

process Consumer[j = 1 to N] {
    while (true) {
        //fetch data from buf; consume it.
        P(full);
        P(mutexF);
        result = buf[front];
        front = (front+1)%n;
        V(mutexF);
        V(empty);
        ...
    }
}
```
Here's how to initialize tuple space with this shared data:

```c
//shared variables -- must be implemented in tuple space
int buf[n],
    front = 0, rear = 0;  // indices to buf
sem empty = n, full = 0;  // between producers/consumers
sem mutexD = 1,             // between different producers
    mutexF = 1;             // between different consumers

// nothing for buf[n] -- until data produced...
out("front", 0);    out("rear", 0);
for (i = 0, i < n, i++) {
    V("empty");
}  // nothing for full -- until producer produces something

V("mutexD");
V("mutexF");
```
Midterm Exam

- Midterm exam: Thu, October 12th
- topics: interleavings, Linda/TS, semaphores
- format: questions, reading/writing C-Linda code