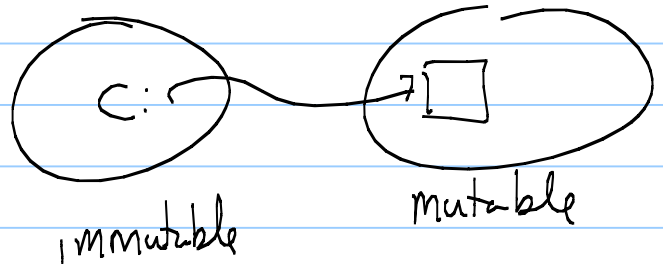
Lecture 11

Note Title 2/14/2008

Exam — Feb 28th

{NewCell X C}

X ↑ value, C ↑ name



operations:

X = @C bind X to current value of C

C := X make cell C contain X

{Exchange C X Y} atomically bind X to current value of C and give Y as C's new content

Assignment: (1) implement := and @ in terms of Exchange

(2) Exercise 6.4 — implement ports in terms of cells

Ch. 8

a ; b ; c // d ; e ; f

4^2 - states

number of possible execution sequences

6 steps 1 2 3 4 5 6

$$\frac{6!}{3!3!} = \frac{720}{36} = 20$$

$$\frac{20!}{10!}$$

Hardware support for critical section entry

Test-and-set
compare-and-swap
exchange

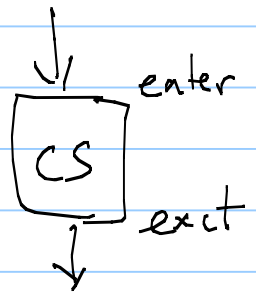
} can be used to implement CS entry easily -
7 • bounded overtaking

↳ why not - because CS requiring busy waiting are typically very small

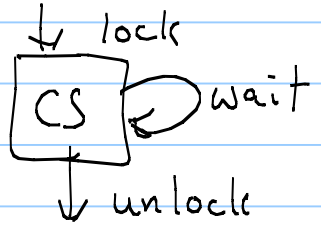
Non-blocking concurrency — specific problems

Herlihy

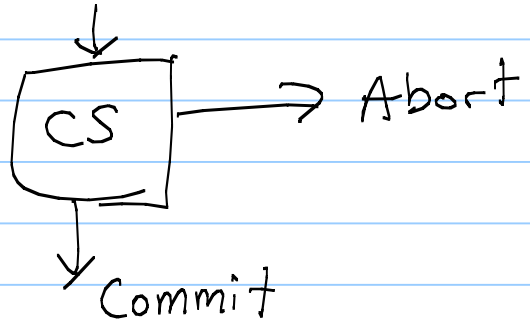
Lock



Monitors



Transactions



Locks

```
fun {SLock}
   Token = {NewCell unit}
   proc {Lock P}
      Old New in
         {Exchange Token Old New}
         {Wait Old}
         try {P} finally New = unit end
      end
   in
      'lock'('lock':Lock)
   end
   'lock'('lock':MyLock) = {SLock}
      {MyLock CScode}
```

Old = @Token
Token := New

DANGER!
Assumes P dies only when Data Structure is in a consistent state.

P must not exit w/exception if inconsistent

Re-entrant lock - same thread can pass thru a lock it already holds.

DANGER!

When original P makes any call that could result in re-entering it must make sure the DS is consistent!

To avoid deadlock (if non-reentrant locks):

Design an outside entry point Pe takes the lock and internal entry point that doesn't

Monitors — Per Brinch-Hansen — Anthony Hoare

Abstract Type & Synchronizing Mech

When you call an operation of the ADT a lock is acquired automatically.

Also a new operation wait() is introduced:

(join a waiting queue; release the lock; go to sleep... re acquire lock...)

notify (removes 1 process from wait queue and makes it runnable.

---

if (! what I want) wait() ←

while (! what I want) wait()