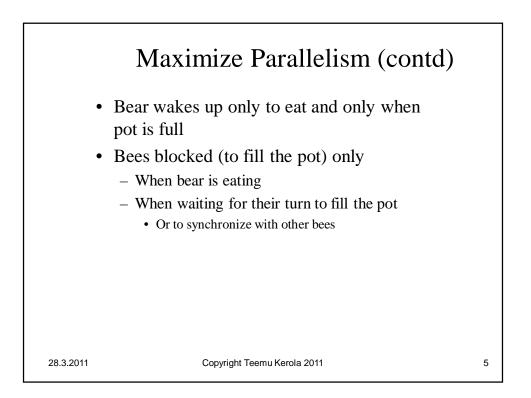
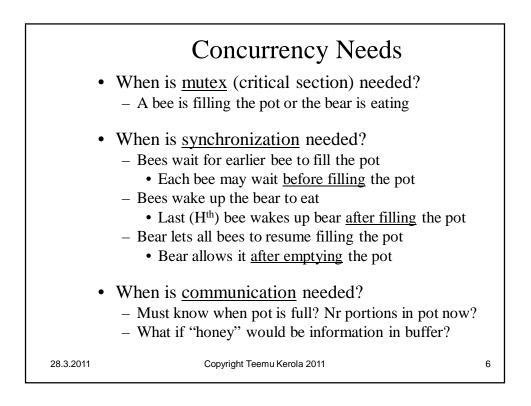
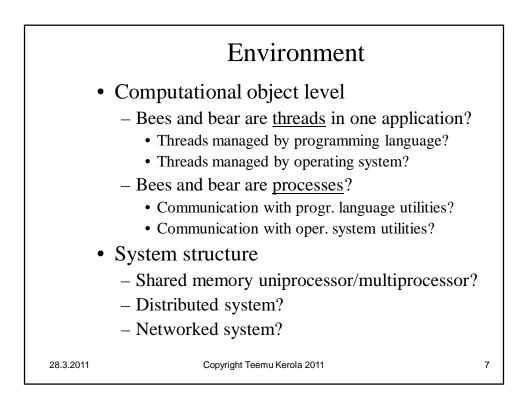
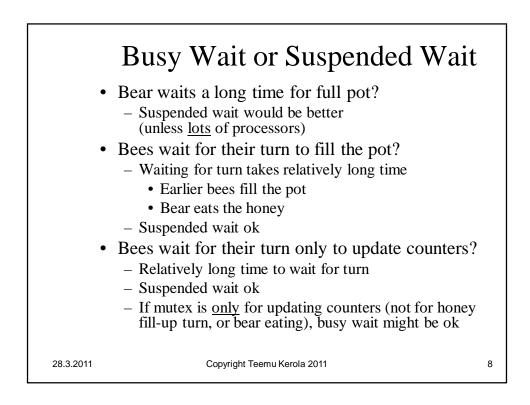


	Maximize Parallelism	
•	All bees concurrently active, no unnecessary blocking	
•	Bees compete only when filling up the pot	
	<ul> <li>Must wake up bear when H portions of honey in pot</li> </ul>	
	– Must fill up the pot one bee at a time	
	• Is this important or could we modify specs?	
	• How big is the mouth of the pot?	
	<ul> <li>Competing just to update the counter would be more efficient?</li> </ul>	
	• Is waking up the bear part of critical section?	
	– What is the real critical section?	
28.3.2011	Copyright Teemu Kerola 2011	4



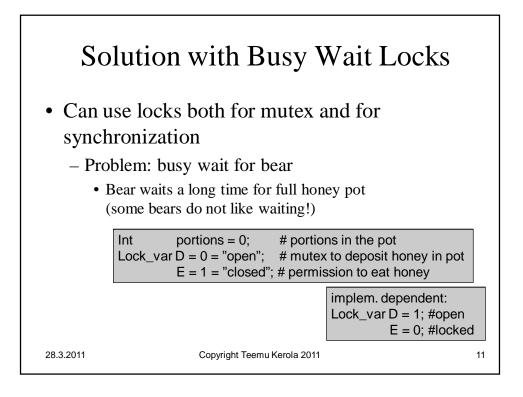




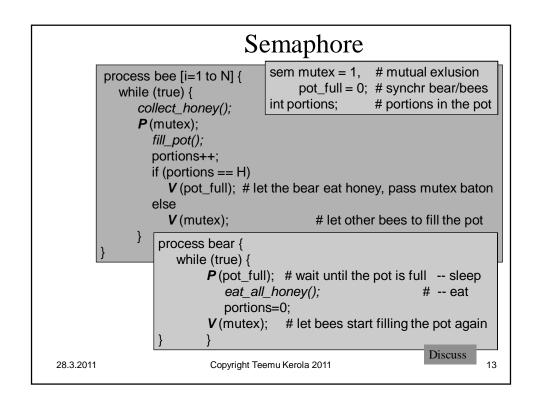


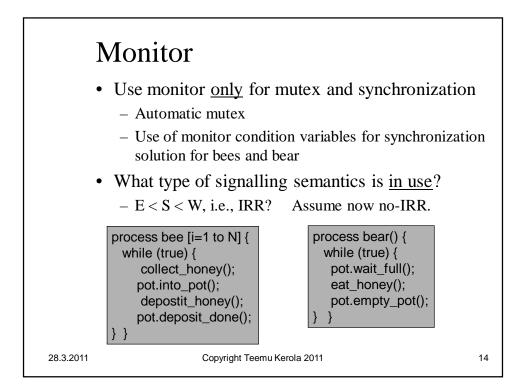
	<b>Evaluate Solutions</b>	
•	Does it work correctly?	
	- Mutex ok, no deadlock, no starvation	
•	Does it allow for maximum parallelism?	
	- Minimally small critical sections	
	- Could bees fill up the jar in parallel?	
•	Is this optimal solution?	
	– Overall processing time? Overall communication time?	
	– Processor utilization? Memory usage?	
	– Response time? Investments/return ratio?	
•	Is this solution good for current problem/environment?	
	<ul> <li>Bees and bear are threads in Java application in 4-processor system running Linux?</li> </ul>	
	- There are 20000 bees, collecting honey takes 15 min, depositing one portion in pot takes 10 sec, 5000 portions fill the pot, and bear eats the honey in pot in 10 minutes?	
28.3.2011	Copyright Teemu Kerola 2011 9	

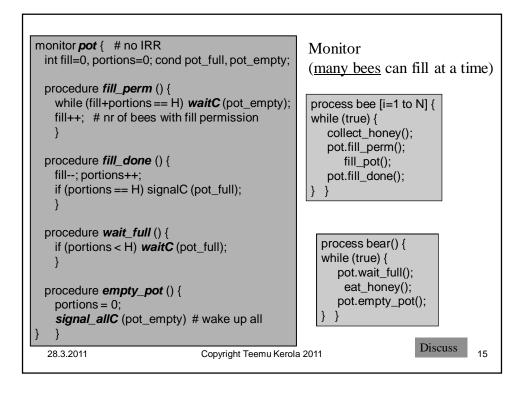
00.0.0011		40
28.3.2011	Copyright Teemu Kerola 2011	10

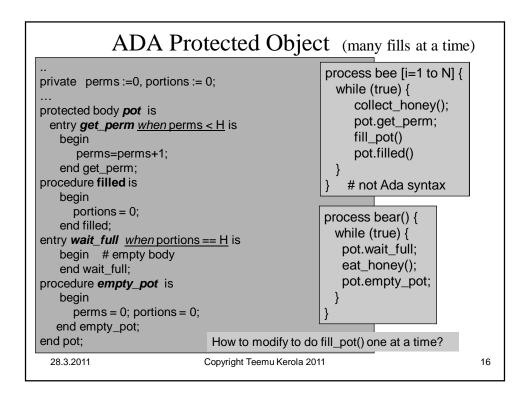


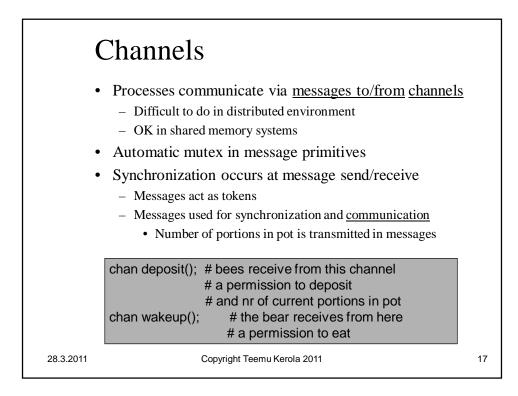
Solution with Locks (contd)				
process bee [i=1 to N] () { while (true) {	Int portions = 0; # portions in the pot Lock_var D = 0; # mutex to deposit honey in pot E = 1; # permission to eat honey			
<pre>collect_honey(); lock (D); # only one be portions++; fill_pot(); if (portions == H) unloc else unlock (D) # let r</pre>	<i>ck</i> (E); # wakeup bear, keep lock			
} } 28.3.2011	process bear () {         while (true) {         lock (E);       # busy-wait, hopefully OK?         eat_honey();         portions = 0;         unlock (D);       # let next bee deposit honey         }       }         Copyright Teemu Kerola 2011       12			











	Channels chan deposit();	
portions+ fill_pot (); if (portion	{ oney (); deposit_perm, portions); # only one bee advances at a tim	
	<pre>send (deposit_perm, 0); # let first bee deposit honey while (true) {     receive (wakeup, dummy);     eat_honey ();     send (deposit_perm, 0); # reset portions to 0 } }</pre>	
	How to modify to do fill_pot() in paralle	
28.3.2011	Copyright Teemu Kerola 2011	18

