Vivian Do
CSC 4330
Dec 04 2023

Homework 9

1. (10 points). The composition of Figure 2 has 6 states — how many many *traces* does it have?

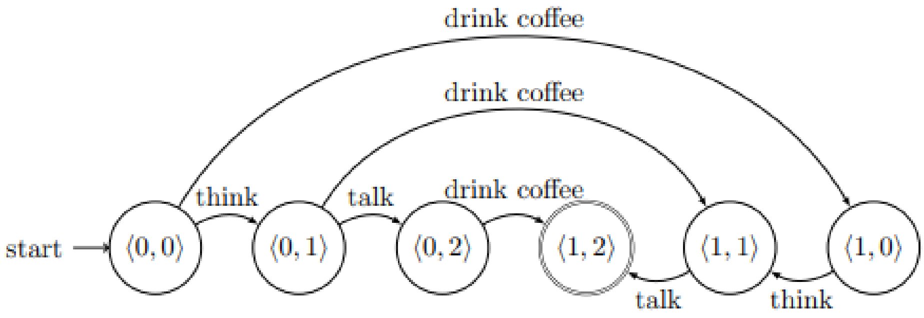


Figure 2: The *composition* of the two concurrent processes of Figure 1

Traces: 3 (or 8 ??)

think ⟶ talk ⟶ drink coffee	:	(0, 0) ⟶ (0, 1) ⟶ (0, 2) ⟶ (1, 2)
think ⟶ drink coffee ⟶ talk	:	(0, 0) ⟶ (0, 1) ⟶ (1, 1) ⟶ (1, 2)
drink coffee ⟶ think ⟶ talk	:	(0, 0) ⟶ (1, 0) ⟶ (1, 1) ⟶ (1, 2)
talk ⟶ drink coffee	:	(0, 1) ⟶ (0, 2) ⟶ (1, 2)
drink coffee ⟶ talk	:	(0, 1) ⟶ (1, 1) ⟶ (1, 2)
drink coffee	:	(0, 2) ⟶ (1, 2)
think ⟶ talk	:	(1, 0) ⟶ (1, 1) ⟶ (1, 2)
talk	:	(1, 1) ⟶ (1, 2)

* Note: I can't tell if traces are the ones that start at (0,0) or all of the possible paths from the figure…

2. (30 points). Suppose we were to add a third (concurrent) process of "eat cookie" to our situation, as depicted in Figure 3 — draw out the resulting composition of the three concurrent processes "drink coffee", "think → talk" and "eat cookie" (you may do this on paper and take a picture).

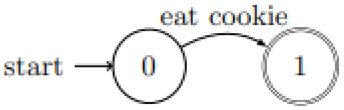
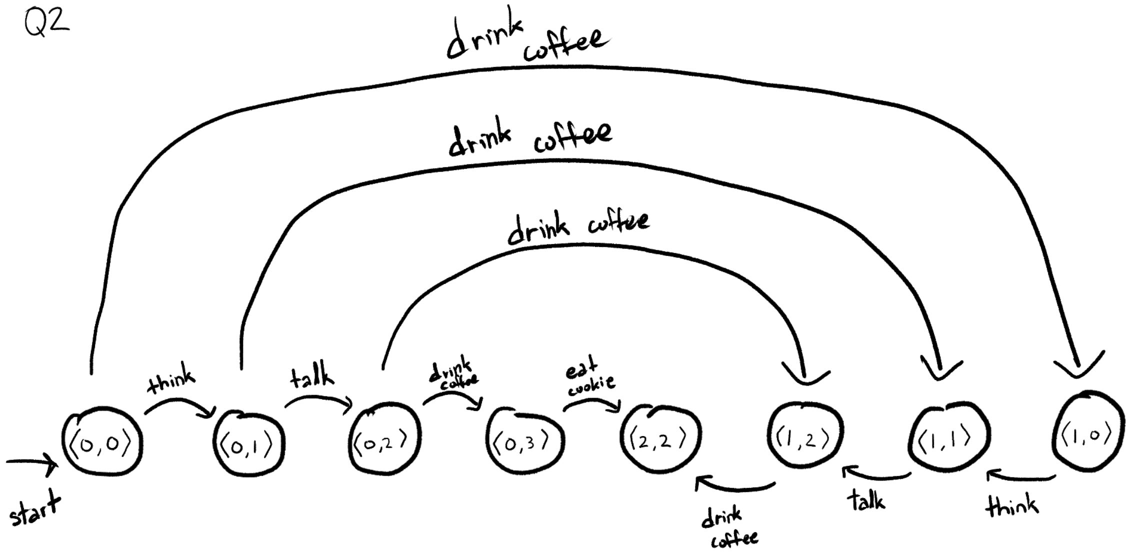


Figure 3: Another possible (concurrent) process among those of Figure 1

HW#9

Q2



3. a. (10 points). How many states does the composition of (2.) have?

b. (10 points). How many *traces* does it have?

a) States: 8

(0, 0) (0, 1) (0, 2) (0, 3) (1, 0) (1, 1) (1, 2) (2, 2)

b) Traces: 4 (or 13 ??)

think → talk → drink coffee → eat cookie	:	(0, 0) → (0, 1) → (0, 2) → (0, 3) → (2, 2)
think → talk → drink coffee → drink coffee	:	(0, 0) → (0, 1) → (1, 2) → (1, 2) → (2, 2)
think → drink coffee → talk → drink coffee	:	(0, 0) → (0, 1) → (1, 1) → (1, 2) → (2, 2)
drink coffee → think → talk → drink coffee	:	(0, 0) → (1, 0) → (1, 1) → (1, 2) → (2, 2)
talk → drink coffee → eat cookie	:	(0, 1) → (0, 2) → (0, 3) → (2, 2)
talk → drink coffee → drink coffee	:	(0, 1) → (0, 2) → (1, 2) → (2, 2)
drink coffee → talk → drink coffee	:	(0, 1) → (1, 1) → (1, 2) → (2, 2)
drink coffee → eat cookie	:	(0, 2) → (0, 3) → (2, 2)
drink coffee → drink coffee	:	(0, 2) → (1, 2) → (2, 2)
eat cookie	:	(0, 3) → (2, 2)
think → talk → drink coffee	:	(1, 0) → (1, 1) → (1, 2) → (2, 2)
talk → drink coffee	:	(1, 1) → (1, 2) → (2, 2)
drink coffee	:	(1, 2) → (2, 2)

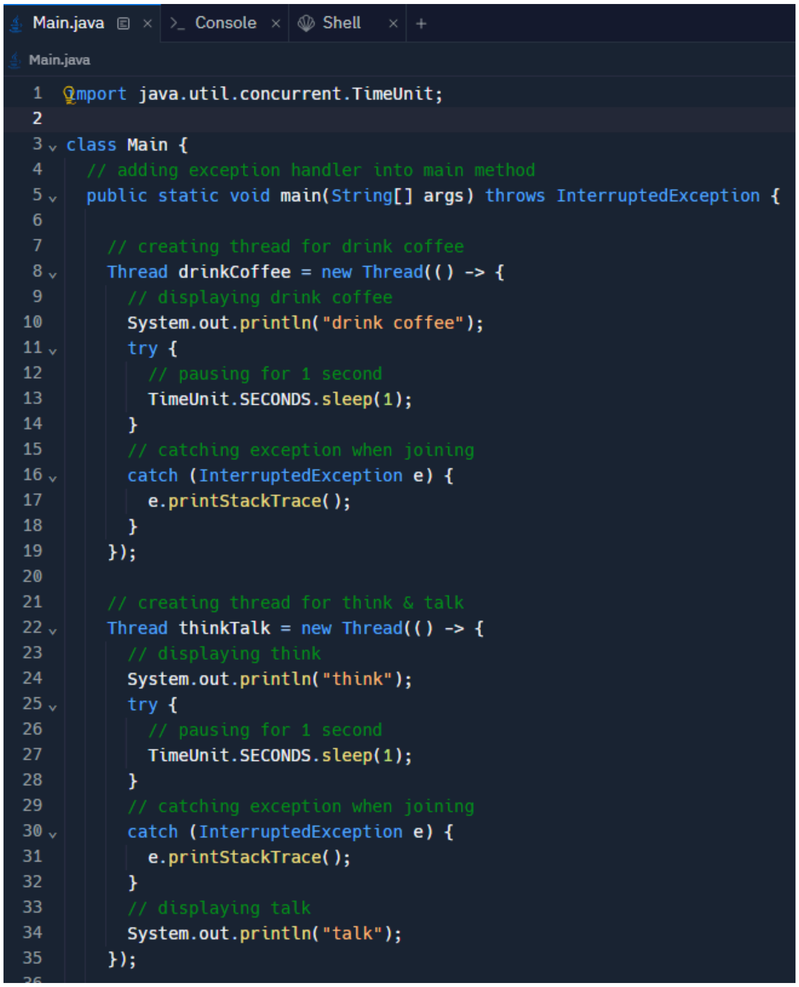
4. (10 points). In the above, we had the three concurrent processes p_1, p_2 and p_3, each with the number $s_1 = 2$, $s_2 = 3$ and $s_3 = 2$ of states, respectively.

In general, for a set of n concurrent processes $p_1, p_2, \dots p_n$, each with the number $s_1, s_2, \dots, s_n$ of states, respectively, how many (composite) states would the composition of these n processes have?

Composite States: s1 * s2 * … * sn

5. (30 points). Model this situation of three concurrent processes with Java threads, in a similar way as that of running several countdown timers concurrently[1]. That is, the "drink coffee" process will pause for a bit, then simply output "drink coffee" and exit (the "eat cookie" process will be similar), while the "think → talk" process will pause for a bit, output "think", pause for another bit, then output "talk" and exit. You may reuse the countdown timer code of footnote 1 for your purposes — it will behave analogously, in that your code will output sequences like: drink coffee → think → eat cookie → talk, *i.e.*, *traces* of the composition of (2.)

My Code:

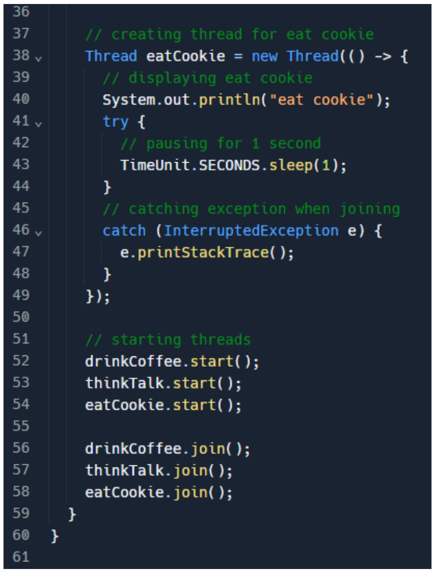


```java
import java.util.concurrent.TimeUnit;

class Main {
  // adding exception handler into main method
  public static void main(String[] args) throws InterruptedException {

    // creating thread for drink coffee
    Thread drinkCoffee = new Thread(() -> {
      // displaying drink coffee
      System.out.println("drink coffee");
      try {
        // pausing for 1 second
        TimeUnit.SECONDS.sleep(1);
      }
      // catching exception when joining
      catch (InterruptedException e) {
        e.printStackTrace();
      }
    });

    // creating thread for think & talk
    Thread thinkTalk = new Thread(() -> {
      // displaying think
      System.out.println("think");
      try {
        // pausing for 1 second
        TimeUnit.SECONDS.sleep(1);
      }
      // catching exception when joining
      catch (InterruptedException e) {
        e.printStackTrace();
      }
      // displaying talk
      System.out.println("talk");
    });
```



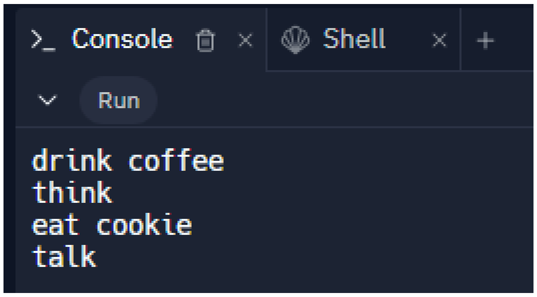
```java

    // creating thread for eat cookie
    Thread eatCookie = new Thread(() -> {
      // displaying eat cookie
      System.out.println("eat cookie");
      try {
        // pausing for 1 second
        TimeUnit.SECONDS.sleep(1);
      }
      // catching exception when joining
      catch (InterruptedException e) {
        e.printStackTrace();
      }
    });

    // starting threads
    drinkCoffee.start();
    thinkTalk.start();
    eatCookie.start();

    drinkCoffee.join();
    thinkTalk.join();
    eatCookie.join();
  }
}
```

My Output:



```
drink coffee
think
eat cookie
talk
```

* Note: I made up my own code to get a similar output sequence as stated in the problem.