We will wait 10 minutes until 10:40 AM for all students to join into the meeting.

We will start the tutorial at **10:40 AM**.

This tutorial will be recorded.

# CS 3SD3 - Concurrent Systems Tutorial 3

Mahdee Jodayree

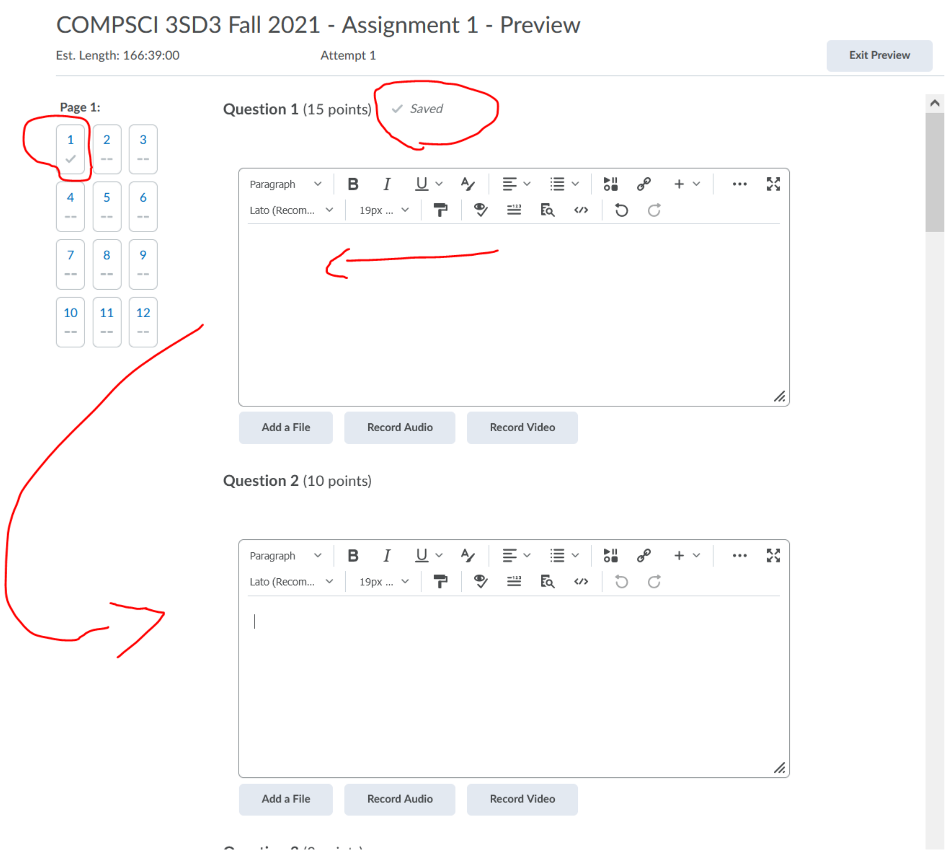
McMaster University

September 28, 2020

# Outline

- Announcements
  - Avenue and submission method.
- Common mistakes when designing a Petri Net
- FSP: Concurrent composition
- Cartesian Product
- Concurrent Composition
- How to convert Petri Nets to LTS
- Reachability graph (cont.);
- Bisimilarity and Equivalent (Semantics)

# Avenue



COMPSCI 3SD3 Fall 2021 - Assignment 1 - Preview

Est. Length: 166:39:00 Attempt 1

# Announcements

- Assignment 1 has been posted and it is due on October 4th, 2021 23:59 PM
- Assignment 1, instruction has been updated.
- Use avenue to submit your assignment.
  - Avenue -> Assessments -> Quizzes -> Assignment 1
- For each assignment you are allowed unlimited number of submission and the lasts submission will count.
- Online compiler for the java questions (recommended not mandatory)
  - https://www.onlinegdb.com/online_c++_compiler
  - .java
  - TA's should be able to copy and past your file to their compiler and read and run your code with any modification
  - PDF files are not acceptable for Java question.
- For each question, there is a rich format text box for every question which also allows
  - copy and paste their solutions directly from Microsoft Word into the text box
  - Attach any image file or pdf file or word file attachment, LTSA file format, …
  - Attach word file or "Please first finish the assignment on your local computer and at the end, copy and paste their solution to each avenue textbox or attach your solution as a PDF file. The **suggested format** for the attached image files is **.png** format (not mandatory).

# Announcements

- Submission format.
  - PDF extensions.
  - For Java, attach your java file only.
  - For LTSA question, either past your LTSA code to the box,
    - Or Attach your LTSA file
    - Or Attach your solution in .txt format (**preferred**)

- PLEASE DO NOT USE THE SUBMIT button every time to SAVE your work. Only use the submit button for your final submission.

- Students can easily create a snapshot of their LTS diagrams by using the **windows snipping tool** (**Grab for Mac**) with these tools students can directly save the image as a .png file and attach it to each textbox at the end after they finalized their solution. Students, who do not have Microsoft Word on their computer, are suggested to use **google document editor** (Google Docs).

- For questions that do not involve using the LTSA tool or producing FSPs, **hand-drawn pictures are allowed**, but a solution should be in image format inserted or attached into the textbox

- (if a student needs to scan a handwritten note and insert it into the textbox is encouraged to use a smartphone app called **CamScanner**).

## Announcements

- Prevent students from not submitting any question.
- It allows multiple TA to mark the same assignments
- Prevents marking errors
- More fair marking.

# Common mistakes when designing a Petri Net

- No initial tokens.
- Place after place.
- Transition after anther transition.
- Cannot move back to initial state.

## FSP: Concurrent composition

Revise:

- An LTS: $P = (S, s, L, \delta)$
- Concurrent composition of two LTS's $P_1 = (S_1, s_1, L_1, \delta_1)$ and $P_2 = (S_2, s_2, L_2, \delta_2)$ is the LTS:

$$(S_1 \times S_2, (s_1, s_2), L_1 \cup L_2, \delta)$$

where

$$((x_1, x_2), a, (y_1, y_2)) \in \delta$$
$$\Leftrightarrow \begin{cases} (x_1, a, y_1) \in \delta_1 \;\wedge\; x_2 = y_2 & \wedge\; \{a\} \in L_1 \setminus L_2 \\ x_1 = y_1 \;\wedge\; (x_2, a, y_2) \in \delta_2 & \wedge\; \{a\} \in L_2 \setminus L_1 \\ (x_1, a, y_1) \in \delta_1 \;\wedge\; (x_2, a, y_2) \in \delta_2 & \wedge\; \{a\} \in L_2 \cap L_1 \end{cases}$$

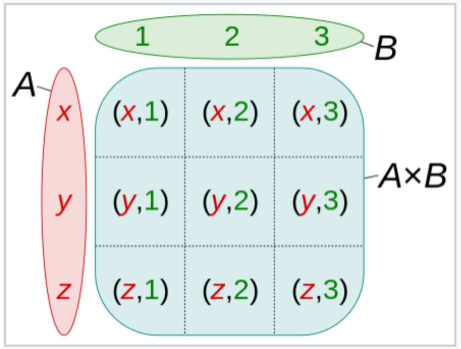
# A Review of Cartesian Products

A Cartesian product is a way to combine sets to get new sets.

Let $S_1 = \{a, b, c\}$

Let $S_2 = \{1, 2\}$

$S_1 \times S_2 = \{(a,1),(a,2),(b,1),(b,2),(c,1),(c,2)\}$



Cartesian product $A \times B$ of the sets $A=\{x,y,z\}$ and $B=\{1,2,3\}$

A three-way Cartesian product:

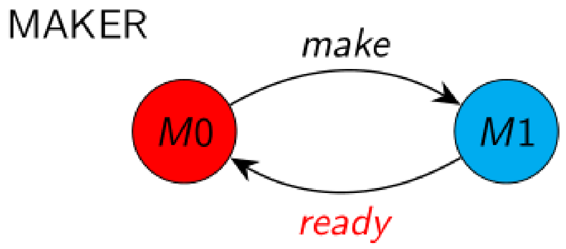
$S_2 \times S_2 \times S_2$
$= \{(1,1,1),(1,1,2),(1,2,1),(1,2,2),(2,1,1),(2,1,2),(2,2,1),(2,2,2)\}$

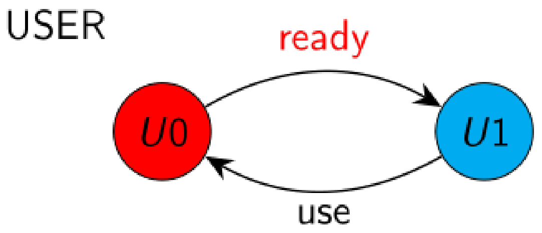
This **not equivalent** to $(S_2 \times S_2) \times S_2$ or $S_2 \times (S_2 \times S_2)$.

## FSP: Concurrent composition

Maker-User example:

```
MAKER = (make → ready → MAKER).
USER = (ready → use → USER).
```





## Concurrent Composition

Maker-User example:

MAKER = (make -> ready -> MAKER).

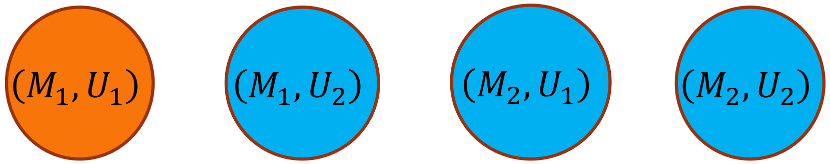
USER = (ready -> use -> USER).

|| MAKER_USER = (MAKER || USER).

Step 1: $S_1 \times S_2, \quad (S_1, S_2)$

$S_1 = (M_1, M_2) \quad S_2 = (U_1, U_2)$

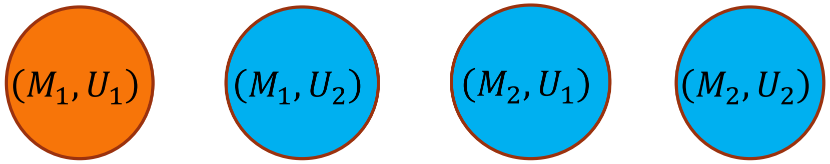
$S_1 \times S_2 = (M_1, U_1), (M_1, U_2), (M_2, U_1), (M_2, U_2)$



## Step 1: Find Cartesian product.

Step 1: $S_1 \times S_2$, $(S_1, S_2)$

In our example, set values are M and U.



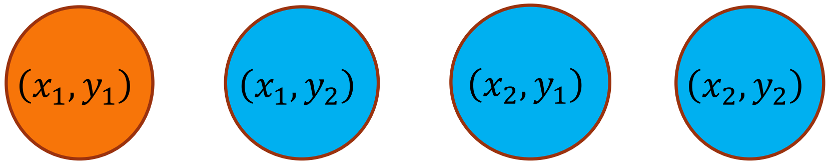
$S_1 = (M_1, M_2)$ $S_2 = (U_1, U_2)$

$S_1 \times S_2 = (M_1, U_1), (M_1, U_2), (M_2, U_1), (M_2, U_2)$

However in the definition the set values are X, Y.

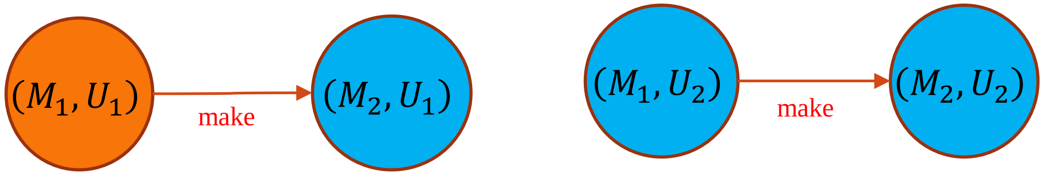
$X_1 = (x_1, x_2)$ $Y_2 = (y_1, y_2)$

$(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$



# Step 2 from the definition

Maker-User example:

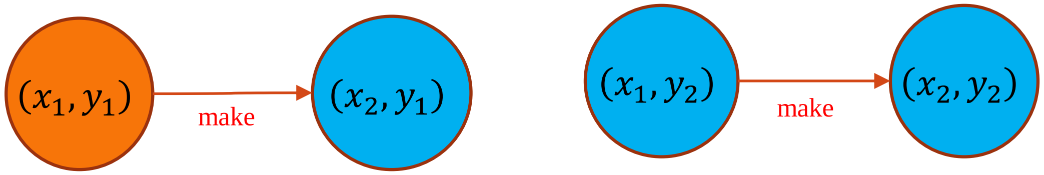


Step 2:

$((x_1, x_2), a, (y_1, y_2)) \in \delta$

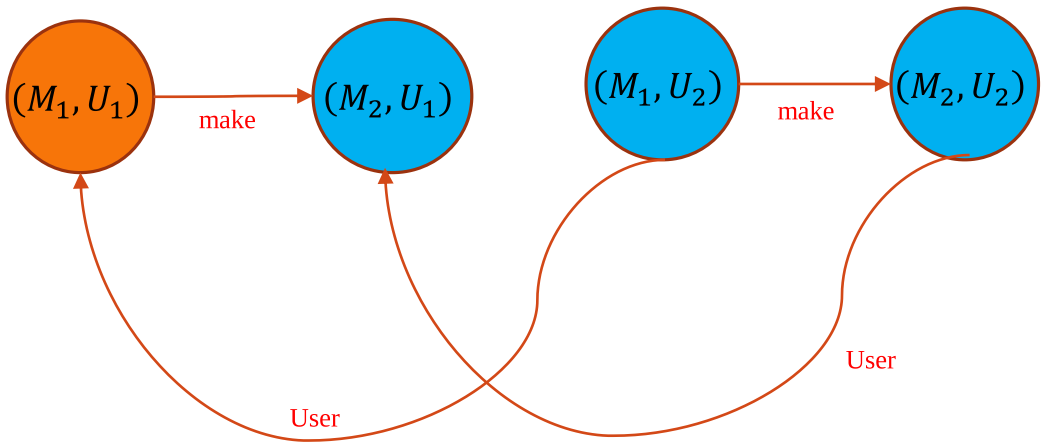
$(x_1, a, y_1) \in \delta_1 \wedge x_2 = y_2 \wedge \{a\} \in L_1 \backslash L_2$

Connect the circles with the same $Y$ values.

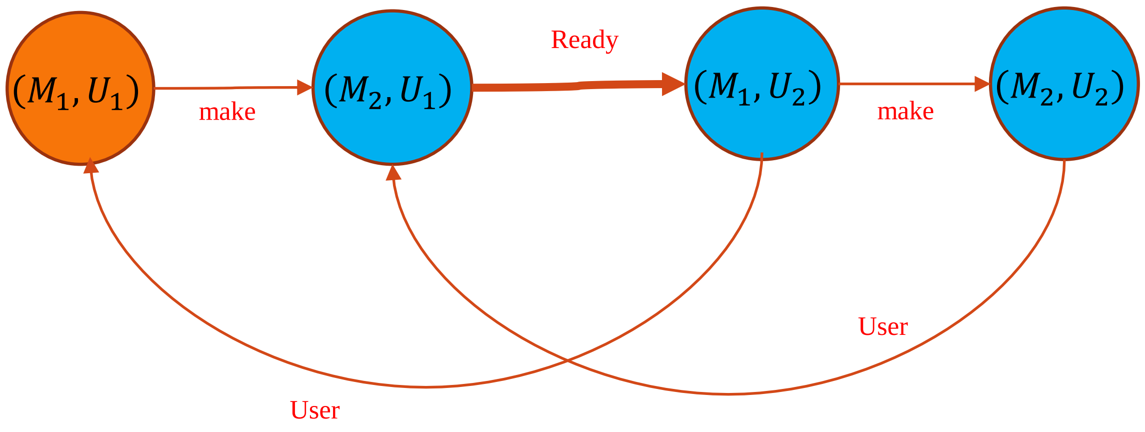


Connect the circles with the same y values

## Step 3 : $x_1 = y_1 \wedge (x_2, \mathrm{a}, y_2) \in \delta_2 \wedge \{a\} \in L_2 \backslash L_1$



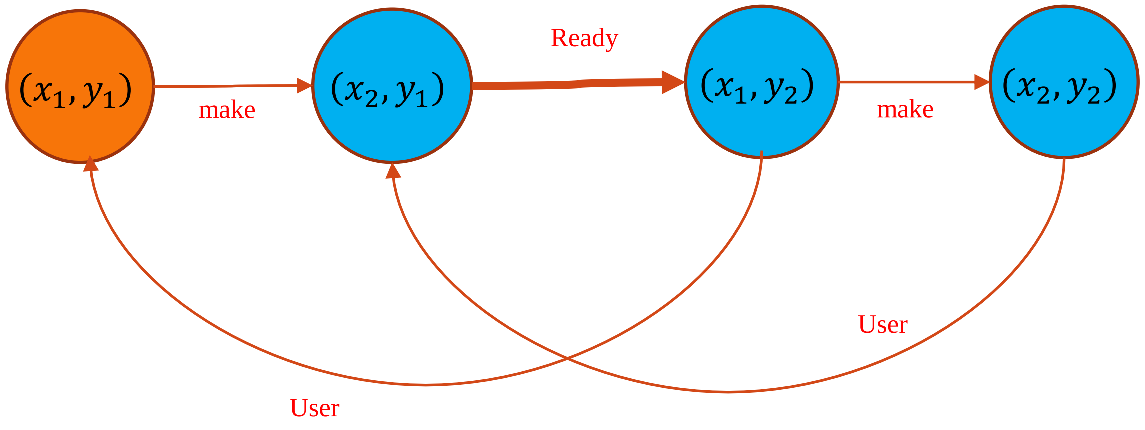
## Step 4 : $(x_1, \text{a}, y_1) \in \delta_1 \wedge (x_2, \text{a}, y_2) \in \delta_2 \wedge \{a\} \in L_2 \cap L_1$


$$\begin{tikzcd}
(M_1, U_1) \arrow[r, "make"'] & (M_2, U_1) \arrow[r, "Ready"] & (M_1, U_2) \arrow[r, "make"'] \arrow[ll, bend left=60, "User"] & (M_2, U_2) \arrow[ll, bend left=60, "User"]
\end{tikzcd}$$

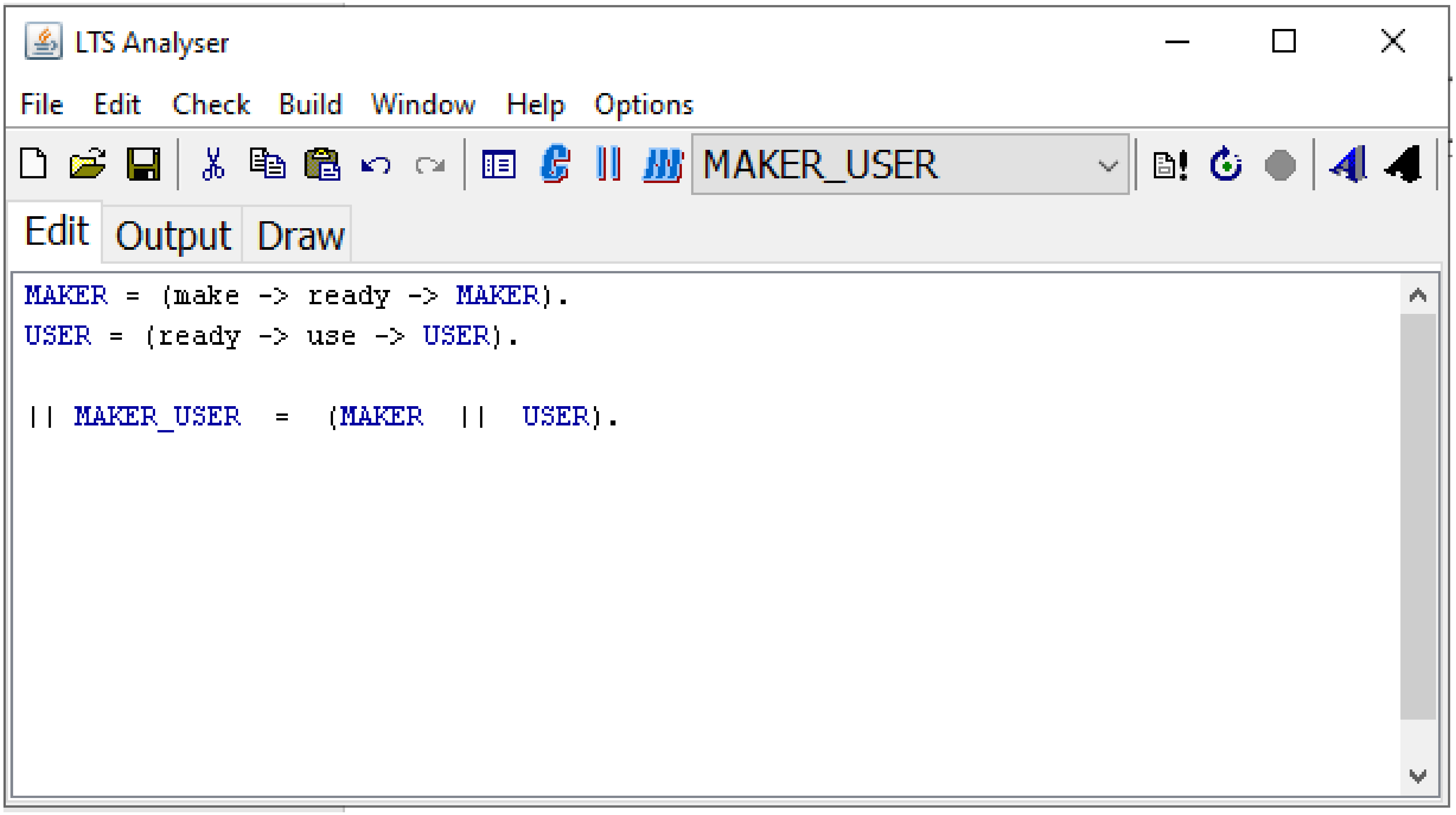
Step 4: Step 4, is for the overlap situation.

To meet both requirements. It means we finally connect circles that their first and second values are not the same.

## Step 4 : $(x_1, \text{a}, y_1) \in \delta_1 \wedge (x_2, \text{a}, y_2) \in \delta_2 \wedge \{a\} \in L_2 \cap L_1$

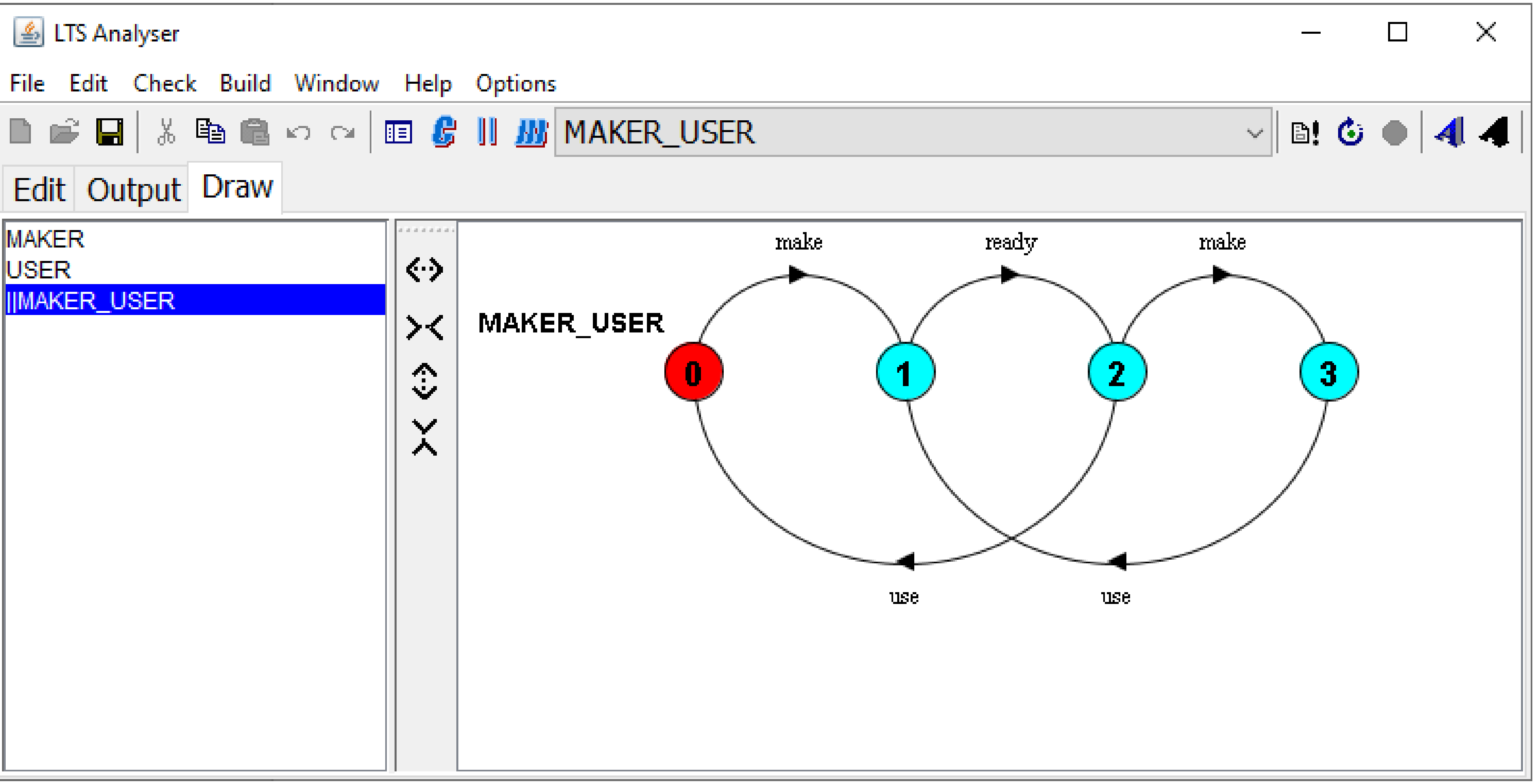


Step 4: This is what it would look like if we had $x_1$ and $y_1$.



```
MAKER = (make -> ready -> MAKER).
USER = (ready -> use -> USER).

|| MAKER_USER  =  (MAKER  ||  USER).
```


LTS Analyser
File
Edit
Check
Build
Window
Help
Options
MAKER_USER
Edit
Output
Draw
MAKER
USER
||MAKER_USER
MAKER_USER
make
ready
make
0
1
2
3
use
use

## FSP: Concurrent composition

Q: How to make the concurrent composition of $P_1, P_2$ and $P_3$?

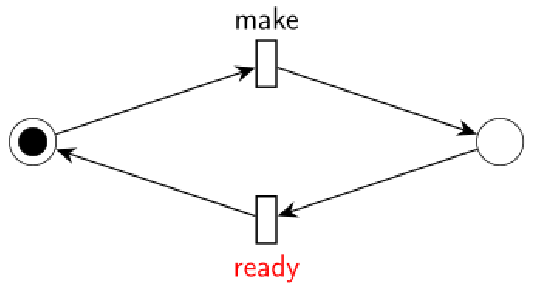
## FSP: Concurrent composition

Q: How to make the concurrent composition of $P_1, P_2$ and $P_3$?
A: Make $P' = P_1 || P_2$, then $P = P' || P_3$

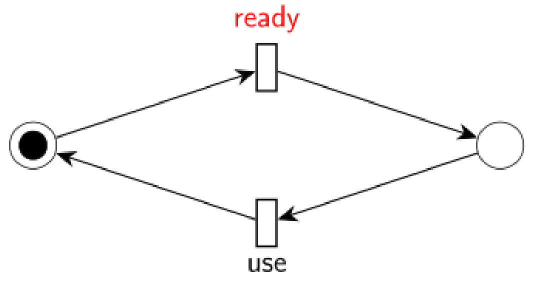
## Elementary Petri Nets: Concurrent composition

Petri nets of processes are “glued” together through the common transition.

Maker:

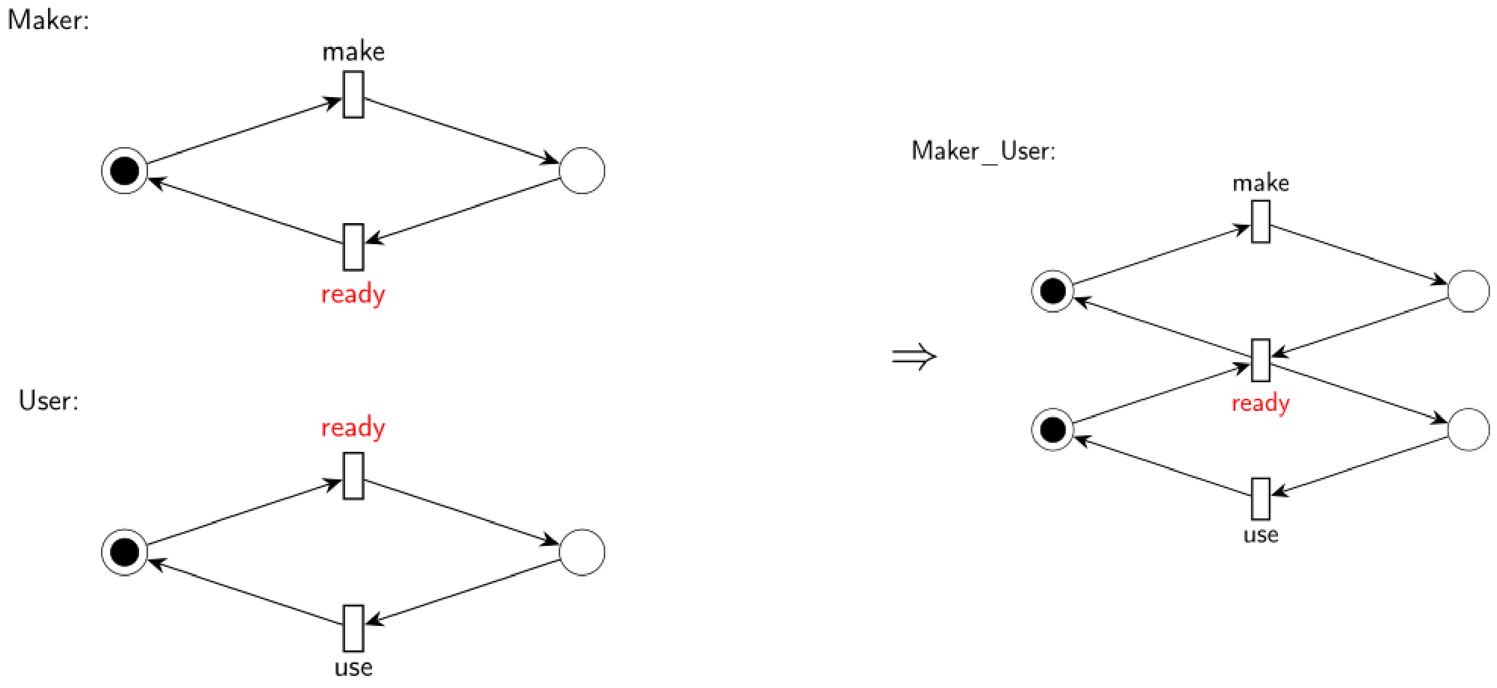


User:



## Elementary Petri Nets: Concurrent composition

Petri nets of processes are "glued" together through the common transition.



## Elementary Petri Nets: Concurrent composition

Q: How to convert Petri Net to LTS?

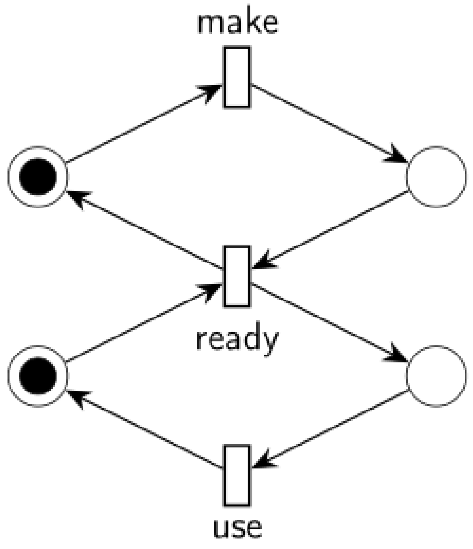
## Elementary Petri Nets: Concurrent composition

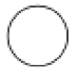
Q: How to convert Petri Net to LTS?
A: By reachability graphs.

## Elementary Petri Nets: Concurrent composition

Reachability graph:

Step 1: Initial state

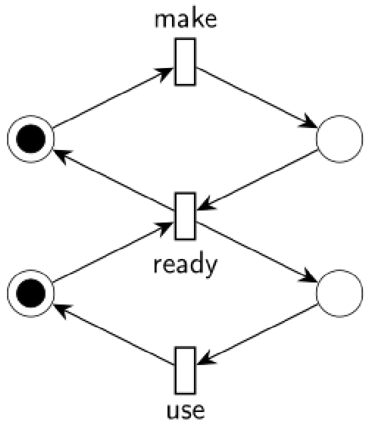


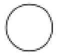


# Elementary Petri Nets: Concurrent composition

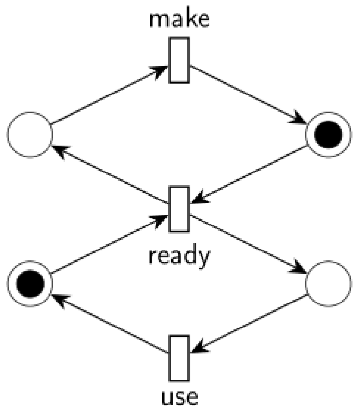
Reachability graph:

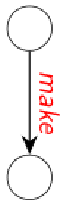
Step 1: Initial state





Step 2: Firing "make"

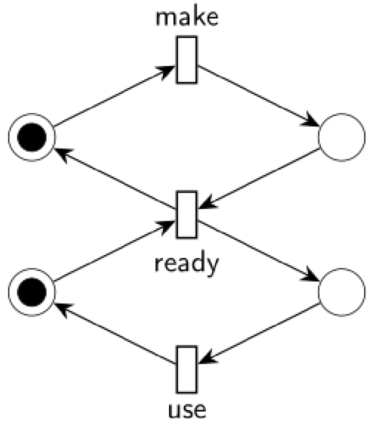


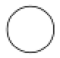


## Elementary Petri Nets: Concurrent composition

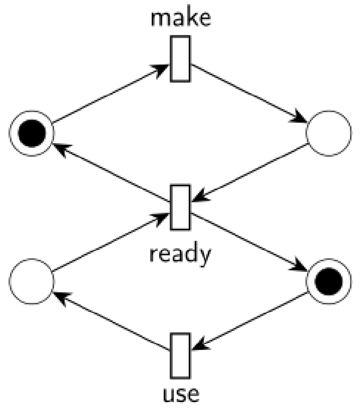
Reachability graph:

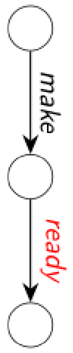
Step 1: Initial state



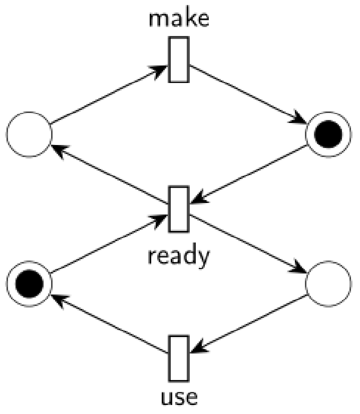


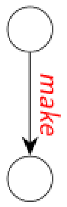
Step 3: Firing "ready"





Step 2: Firing "make"

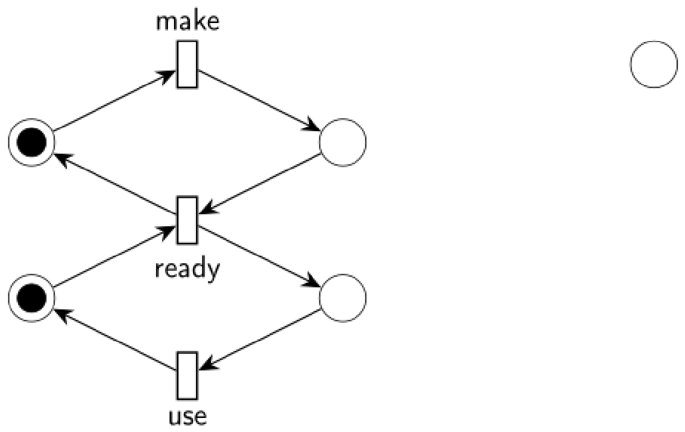




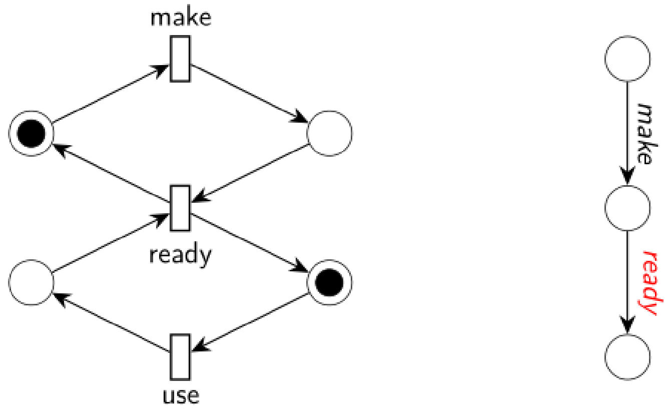
## Elementary Petri Nets: Concurrent composition

Reachability graph:

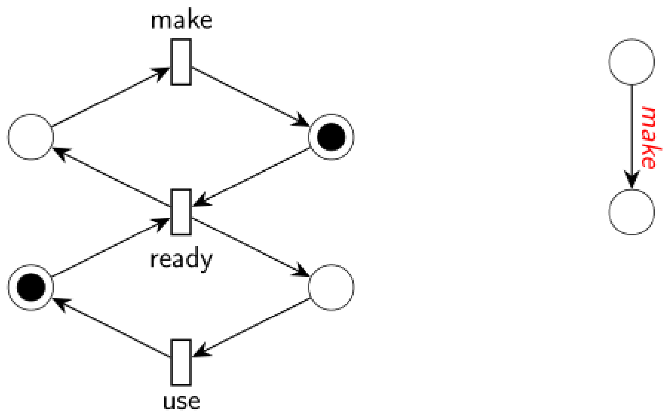
Step 1: Initial state



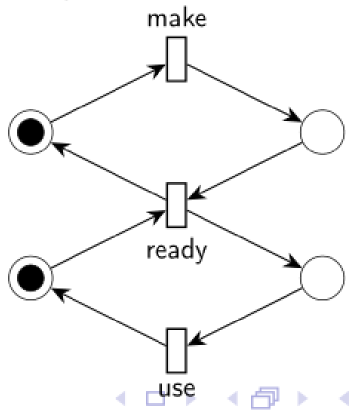
Step 3: Firing "ready"



Step 2: Firing "make"



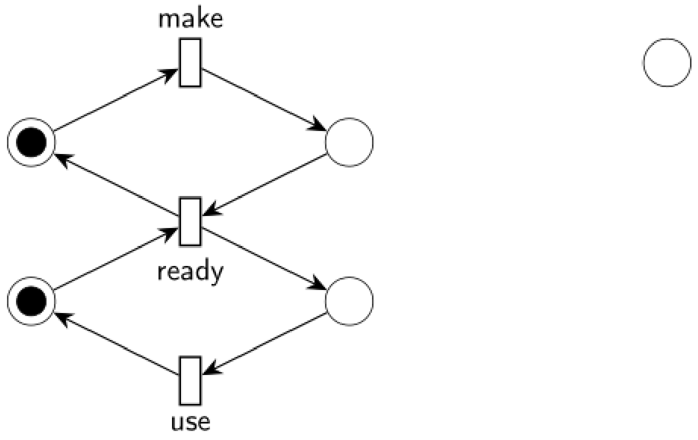
Step 4-1: Firing "use", the LHS is equivalent to that of Step 1



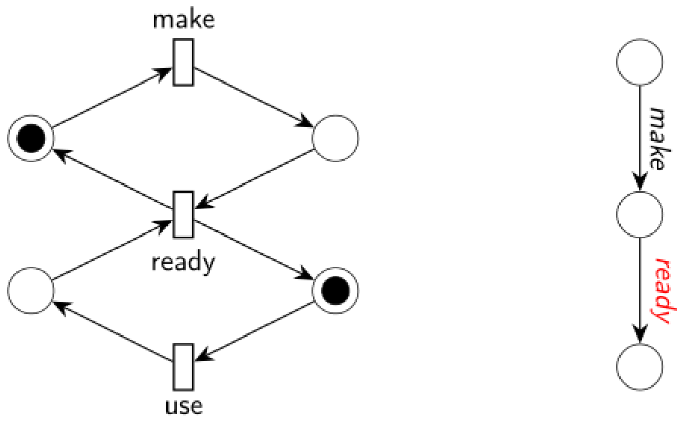
## Elementary Petri Nets: Concurrent composition

Reachability graph:

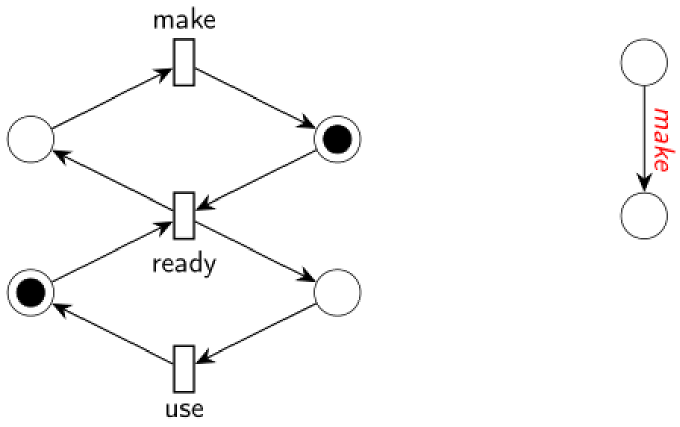
Step 1: Initial state



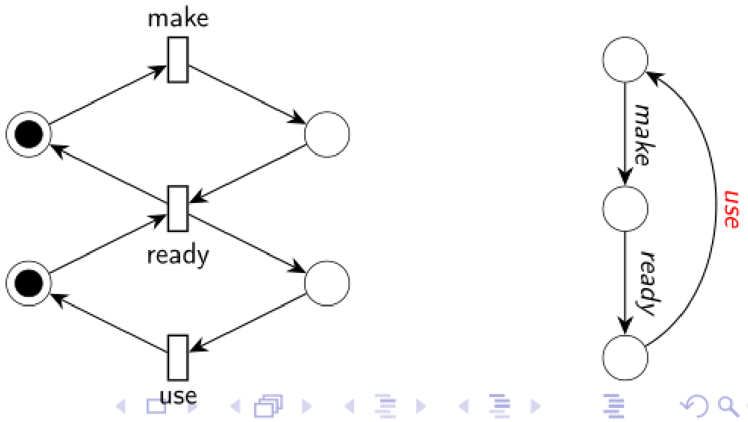
Step 3: Firing "ready"



Step 2: Firing "make"



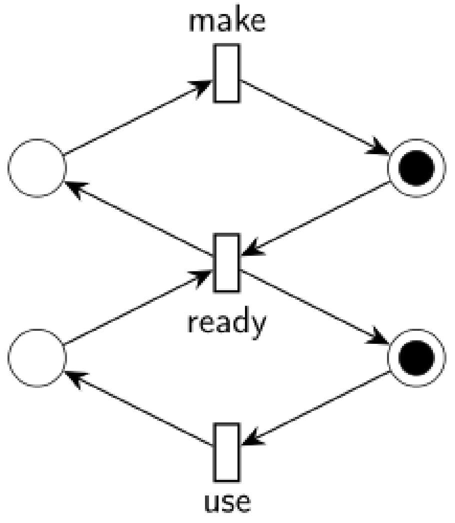
Step 4-1: Firing "use", the LHS is equivalent to that of Step 1

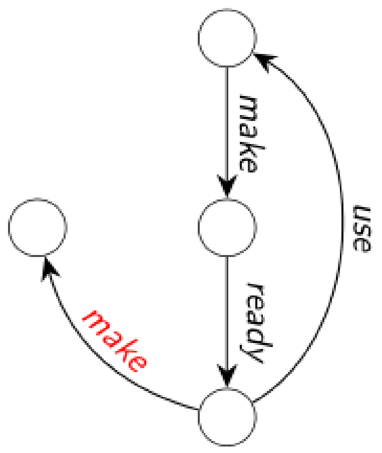


## Elementary Petri Nets: Concurrent composition

Reachability graph (cont.):

Step 4-2: Firing "make"

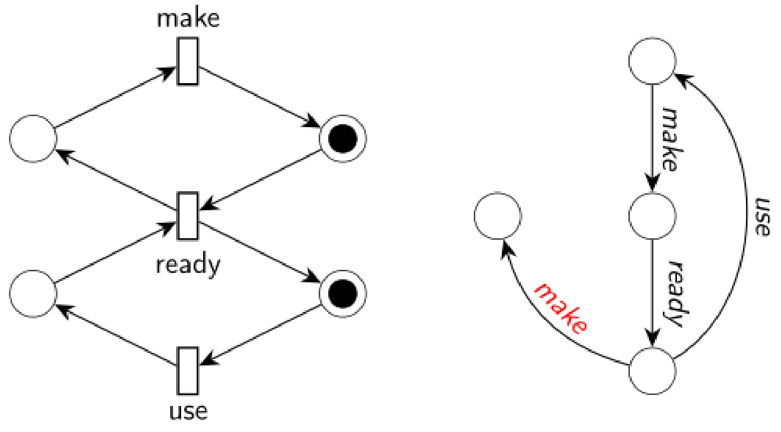




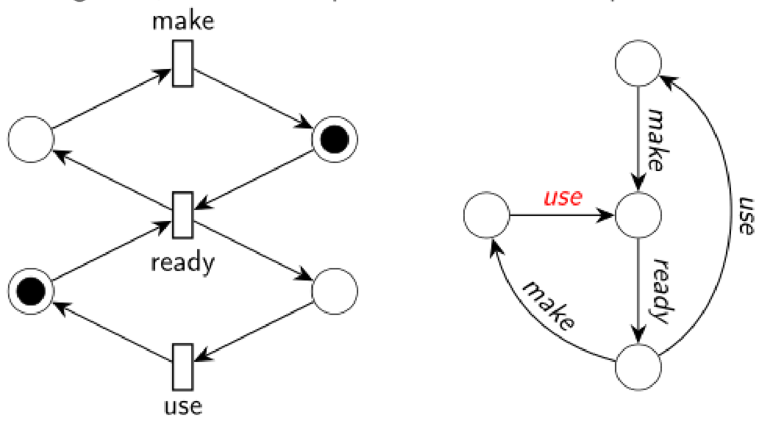
## Elementary Petri Nets: Concurrent composition

Reachability graph (cont.):

Step 4-2: Firing "make"



Step 4-2-1: Firing "use", the LHS is equivalent to that of Step 2



## Semantics

- What is the meaning of

$$P = P_1 \parallel P_2 \parallel \ldots \parallel P_n \ ?$$

- Precise semantics is needed in order to answer *the fundamental question of all models*:
  - ♣ **What does it mean that $P$ and $Q$ are equivalent, written $P \equiv Q$?**
- Obvious properties of equivalence for this model:

$$\begin{aligned} P \equiv Q &\implies (a \to P) \equiv (a \to Q) \\ &\implies (a \to S \mid b \to P) \equiv (a \to S \mid b \to Q) \\ &\implies S \parallel P \equiv S \parallel Q \end{aligned}$$

- Equivalence must preserve model operations, otherwise a model is useless!

## Equivalence of LTS and FSP

**Definition**

Two Labeled Transition Systems $S_1$ and $S_2$ are equivalent if and only if they are *bisimilar*, i.e.

$$S_1 \equiv S_2 \iff S_1 \approx S_2.$$

**Definition**

Two Finite State Processes $P_1$ and $P_2$ are equivalent (or bisimilar) if and only if their Labeled Transition Systems are *equivalent* (or *bisimilar*), i.e.

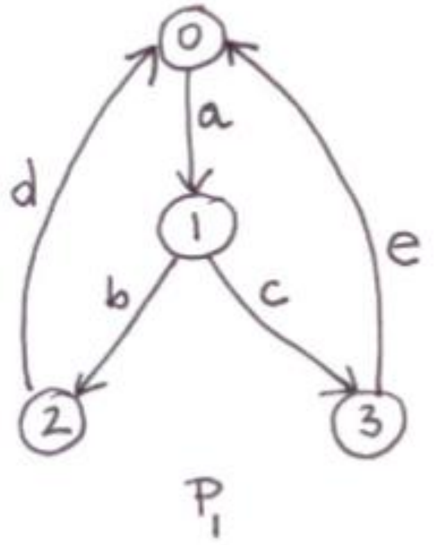
$$P_1 \equiv P_2 \iff LTS(P_1) \equiv LTS(P_2) \iff LTS(P_1) \approx LTS(P_2).$$

- The concept of bisimulation can be defined for FSPs directly, without using LTS, but it will not be discussed in this course.

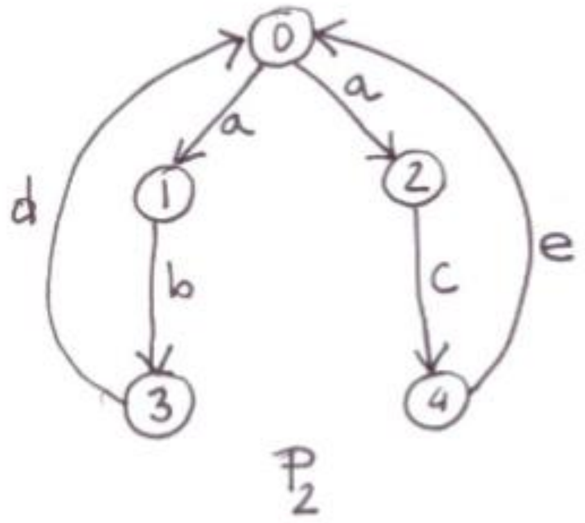
## Example

$P_1 = a \to ((b \to d \to P_1) \mid (c \to e \to P_1))$

$P_2 = (a \to b \to d \to P_2) \mid (a \to c \to e \to P_2))$


$$\begin{tikzcd}
 & 0 \arrow[d, "a"] & \\
 & 1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
2 \arrow[uur, bend left, "d"] & & 3 \arrow[uul, bend right, "e"']
\end{tikzcd}$$

$P_1$


$$\begin{tikzcd}
 & 0 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
1 \arrow[d, "b"] & & 2 \arrow[d, "c"] \\
3 \arrow[uur, bend left, "d"] & & 4 \arrow[uul, bend right, "e"']
\end{tikzcd}$$

$P_2$

- $Traces(P_1) = Traces(P_2) = Prefix((a(bd \cup ce))^*)$

**Prefix** means, all possible traces of a process.

- Let $P$ and $Q$ be Labeled Transition Systems and let $p$ be a state in $P$ and $q$ be a state in $Q$.

**Definition (States bisimilarity)**

We say that the states $p$ and $q$ are bisimilar, $p \approx q$, $\Longleftrightarrow$

whatever action can be executed at $p$ it can also be executed at $q$ and vice versa.

**Definition (LTS bisimilarity)**

We sat that two labeled transition systems $P$ and $Q$ are bisimilar, $P \approx Q$, $\Longleftrightarrow$

each state $p_t$ reachable from the initial state by executing a trace $t$ in $P$, is bisimilar to an appropriate state $q_t$ that is reachable from the initial state by the same trace $t$ in $Q$.

## LTS: bisimilarity

Let $P$ and $Q$ be two different LTS's, and let $p$ be a state in $P$ and $q$ be a state in $Q$.

**Definition: States bisimilarity**

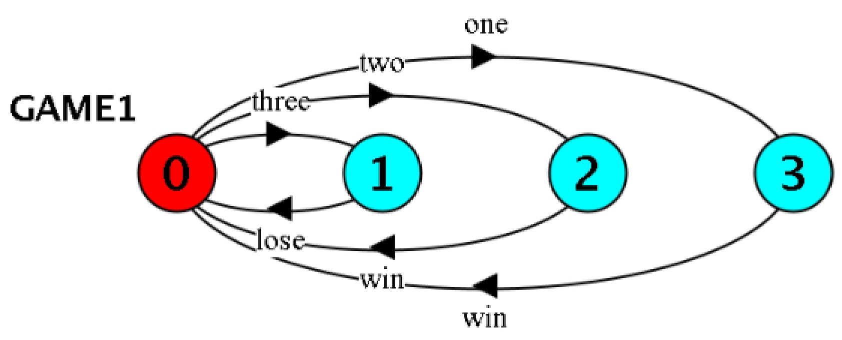
The two states $p$ and $q$ are bisimilar, denoted by $p \approx q$, if and only if whatever action can be executed at $p$ it can also be executed at $q$ and vice versa.

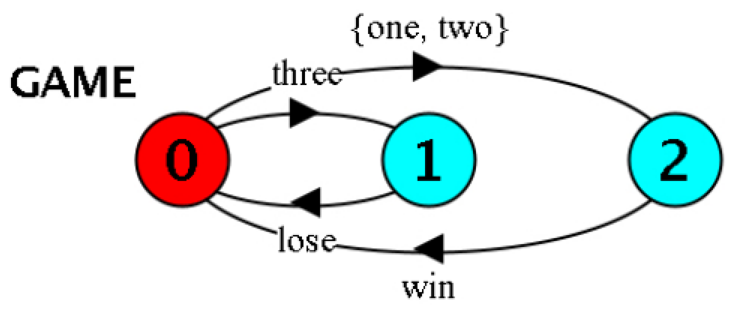
**Definition: LTS's bisimilarity**

The two LTS's $P$ and $Q$ are bisimilar, denoted by $P \approx Q$, if and only if each state $p_t$ reachable from the initial state by executing a trace $t$ in $P$, is bisimilar to an appropriate state $q_t$ that is reachable from the initial state by the same trace $t$ in $Q$.

## LTS: bisimilarity

Example 1: Prove that the two below LTS's are bisimilar:





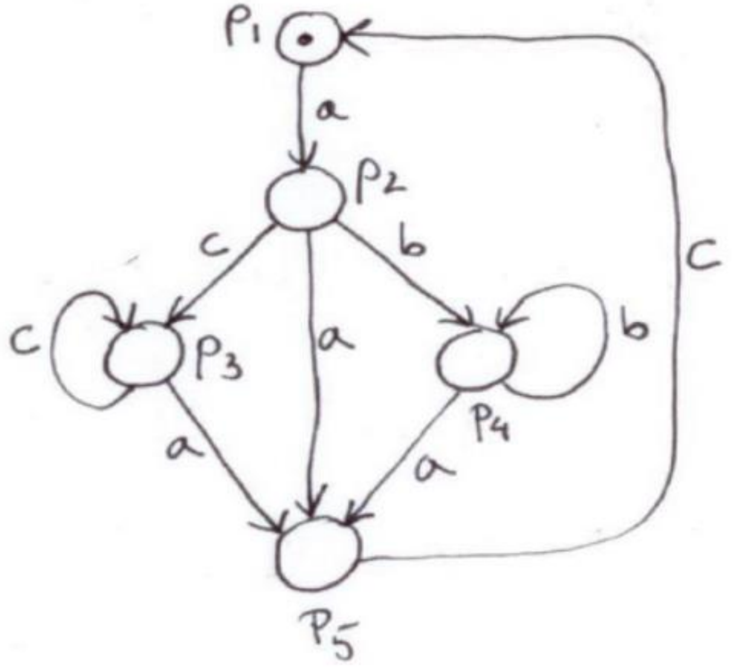
**Definition (States bisimilarity)**

We say that the states $p$ and $q$ are bisimilar, $p \approx q$, $\iff$

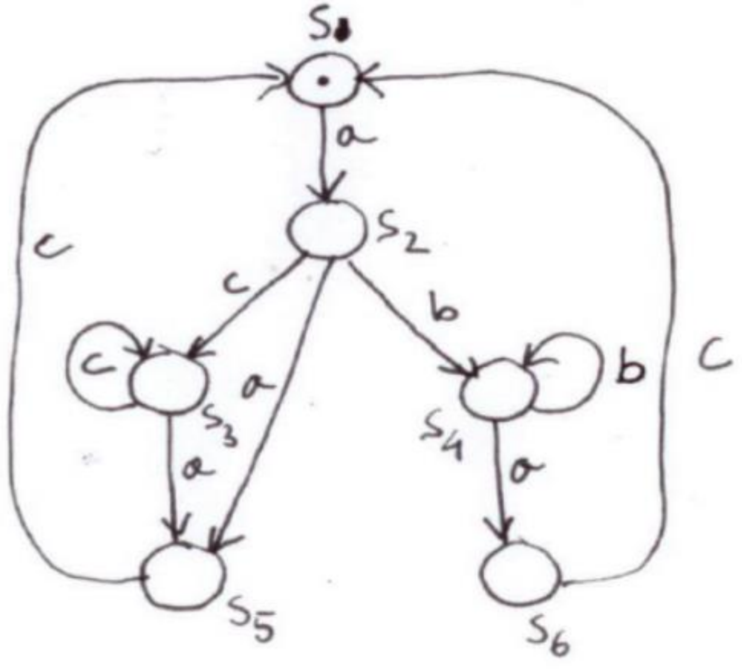
whatever action can be executed at $p$ it can also be executed at $q$ and vice versa.

## LTS: bisimilarity

Example 2: Prove that the two below LTS's are bisimilar:



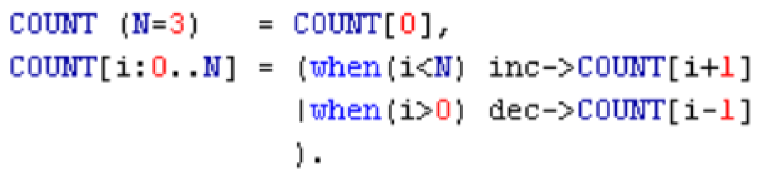
$\mathcal{P}_2$



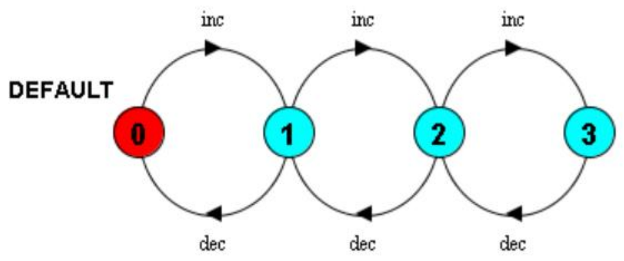
$\mathcal{P}_3$

# FSP guarded actions

- You need to know this for Assignment 1, question 10.
- LTSA examples Chapter 2 → Count
- We are defining i in the second line, LTSA is different from JAVA
- From 0 to 3 (range ), do the following, when i<N,…



```
COUNT (N=3)    = COUNT[0],
COUNT[i:0..N] = (when(i<N) inc->COUNT[i+1]
                |when(i>0) dec->COUNT[i-1]
                ).
```

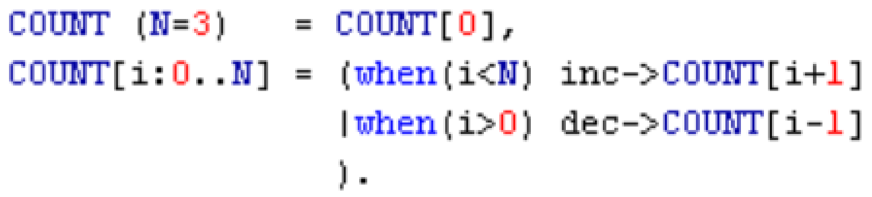


# FSP guarded actions

- ❖There is a another way of writing the code on the right side.
- ❖For the assignment 1, question 10 , use the approach on the left side.

const N=3

count = count [0],



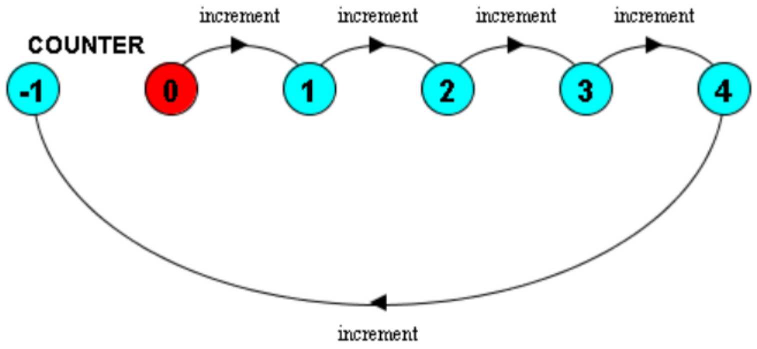
```
COUNT (N=3)    = COUNT[0],
COUNT[i:0..N] = (when(i<N) inc->COUNT[i+1]
                |when(i>0) dec->COUNT[i-1]
                ).
```

## Guarded actions

LTSA examples, Chapter 4 -> counter

```
const N = 4
range T = 0..N

COUNTER = COUNTER[0],
COUNTER[v:T] = (increment ->COUNTER[v+1]).
```



# Any Questions?