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 2

Mahdee Jodayree

McMaster University

September 21, 2021

Outline

- Announcements.
- More details on materials were covered last week.
- Review of important terms from the lecture notes.
  - LTS, FSP and LTSA.
- Petri Nets.
- Reading suggestion.

Tutorials schedule

- Tuesdays (Mahdee)
  - Tutorials are recorded on teams, and everyone in the class can access and watch them.
  - Online for now.
  - Can also ask questions.
- Wednesdays (Jatin)
  - Will answer questions
  - Live but not recorded.
- Fridays (Mahdee)
  - Will answer questions
  - Live but not recorded.

Assignment 1

McMaster University
Computing & Software

Concurrent Systems
COMP SCI 3SD3

Term 1, Fall 2021

- Course Information
- Communications
- Course Material
- Term Marks
- Academic Dishonesty

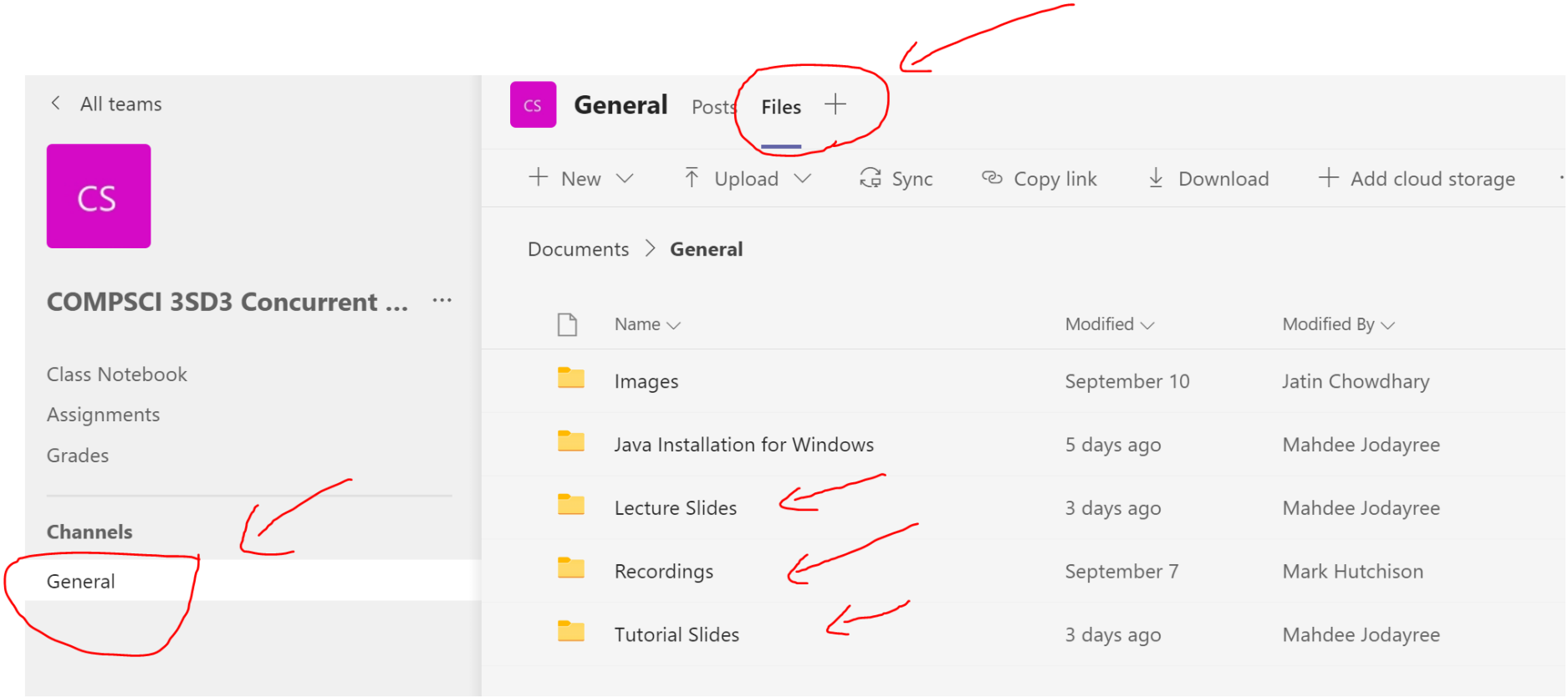
Course Material

- Petri Nets - book
- LTSA Tool Direct Download Link
- Coloured Petri Nets (Textbook)

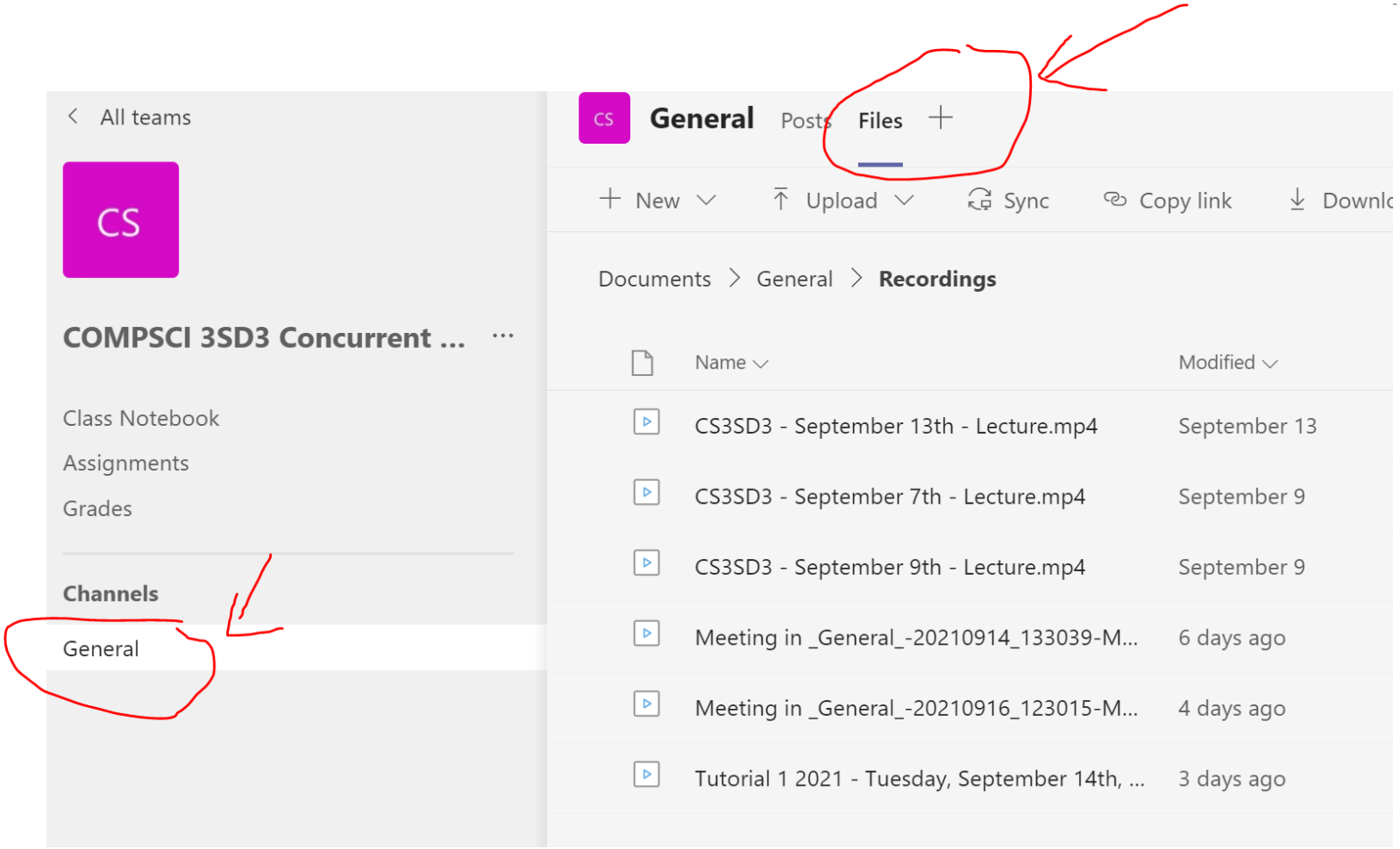
Assignments:	Due Dates:
Assignment 1	Monday, October 4, 23:59

Tutorial Notes:	Lecture Dates:	Live lectures.	Prerecorded lectures.
Tutorial Slides 1	September 14 10:30 AM	Video Link.	

Information related to Teams.



Information related to Teams.



Benefits

You can access the material much easier on your **phone** and your **table,** and on your **pc**.

You can also download the videos.

- If you do not have a stable internet connection.

If you download any videos on windows.

- It will go to your windows default download folder, located in
- **C:\Users\$your username$\Downloads**
- Start menu, type downloads.

LTS, FSP and LTSA

- LTS means Labelled Transition System, $P = (S, s, L, \delta)$ where:

S: set of states,
$s \in S$: initial states,
L: set of action labels,
$\delta = S \times L \times S$: transition function.

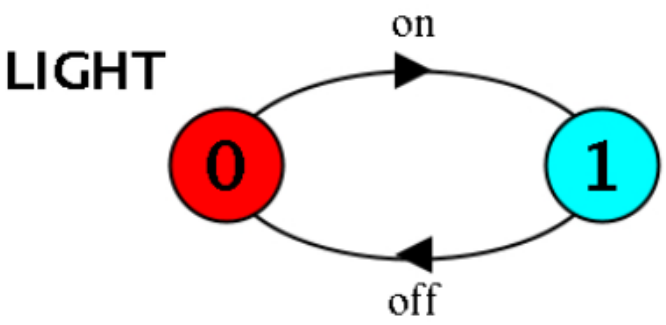


Figure: An LTS of a light

LTS, FSP and LTSA

Labelled Transition Systems (LTS)

- To analyze, display and animate behaviour
- Java threads
- Automata, state machines

- FSP means Finite State Processes
  - Another name for FSP is CSP (Communicating Sequential Processes), Processes in Process Algebras
  - To model processes as sequences of actions

LTS, FSP and LTSA

- FSP means *F*inite *S*tate *P*rocesses.
- FSP is an algebra modelling processes.
- Every FSP description has a corresponding LTS.

```
LIGHT = (on -> off -> LIGHT).
```

Not recommended

Figure: An FSP description of a light

- Basic syntax:
  - Each process ends by a full-period.
  - There could be several sub-processes within a process and separated by commas

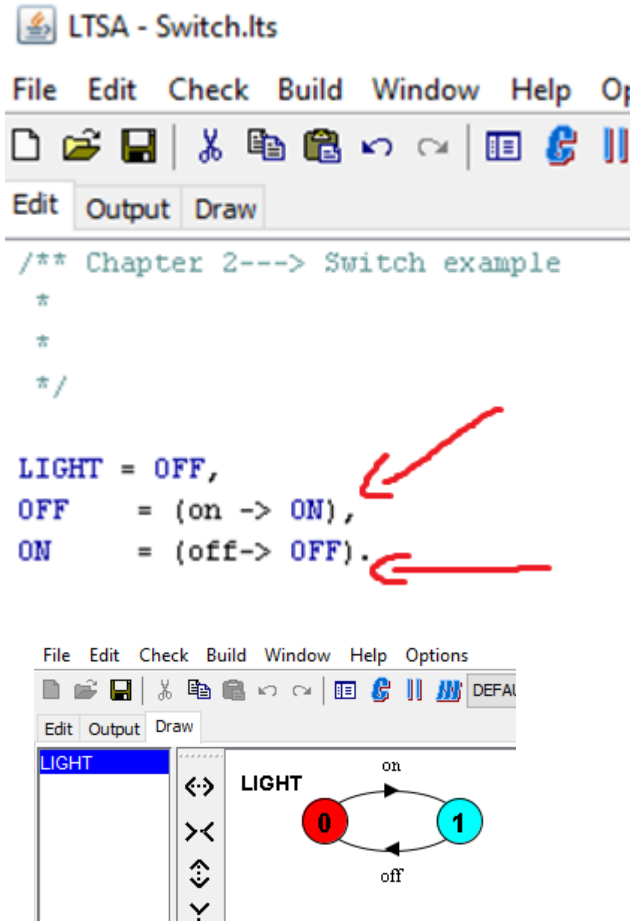
```
LIGHT = OFF,
OFF = (on -> ON),
ON = (off -> LIGHT).
```

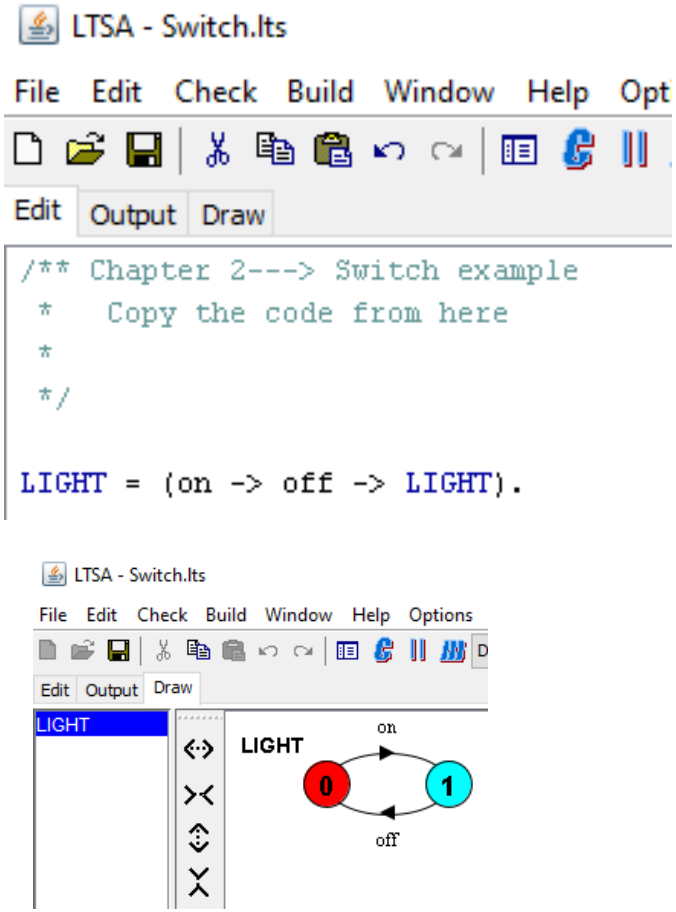
Recommended way

For assignments

Figure: An FSP description of a light with multiple sup-processes

Chapter 2 -> Switch example





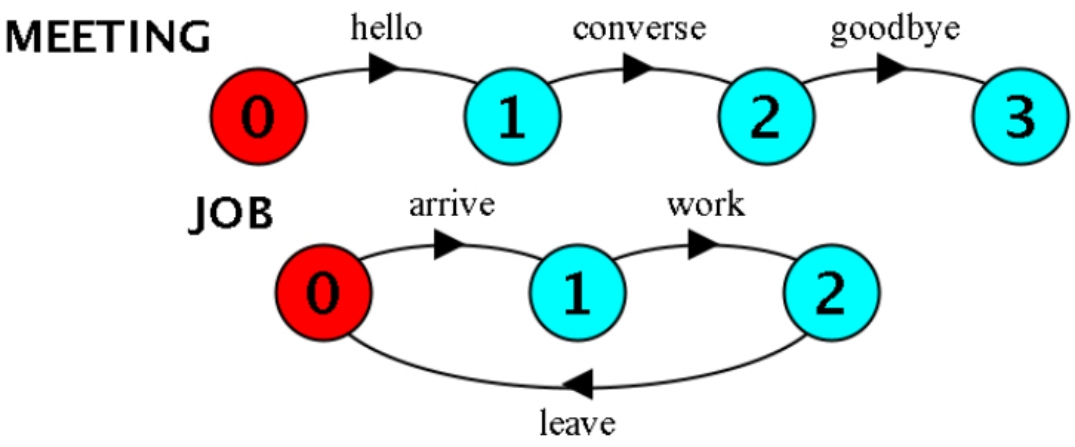

```
/** Chapter 2---> Switch example
 *   Copy the code from here
 */

LIGHT = OFF,
OFF    = (on -> ON),
ON     = (off-> OFF).
```

Example of LTS to FSP:

Labelled Transition System to Finite State Processes

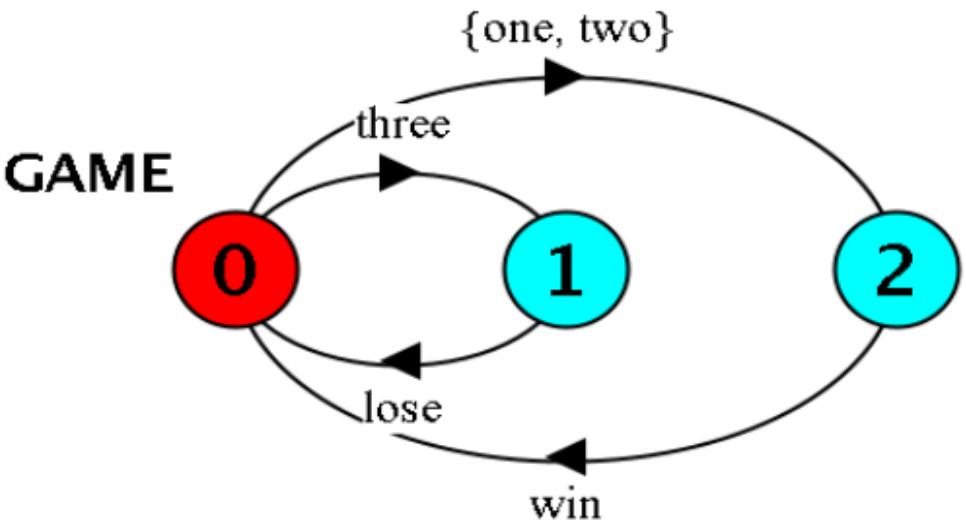
Some examples:

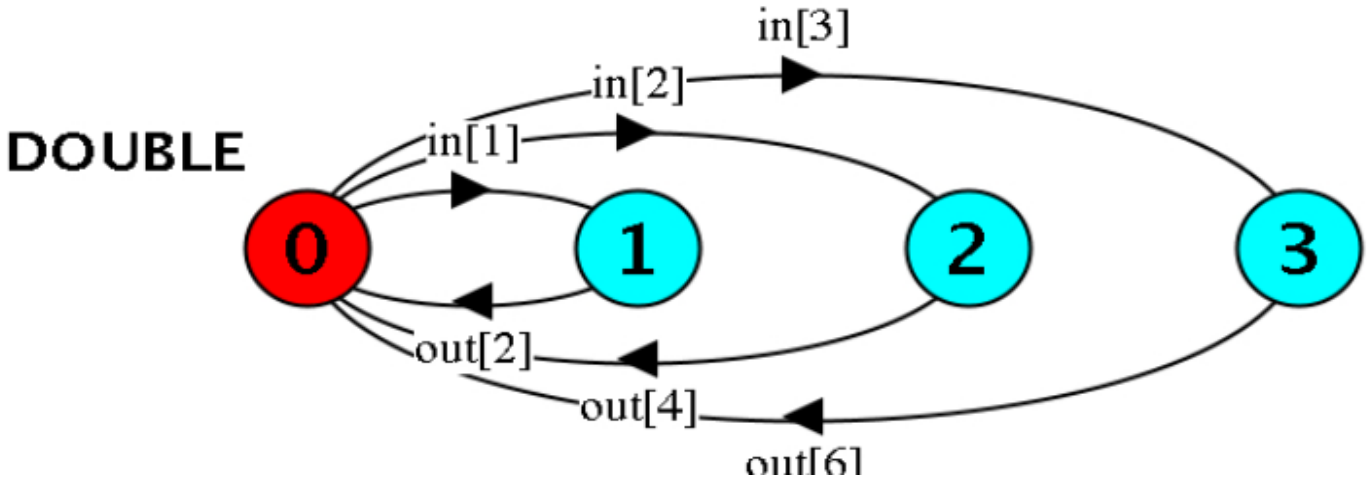
- LTS to FSP:



Some examples:

- LTS to FSP (cont.):



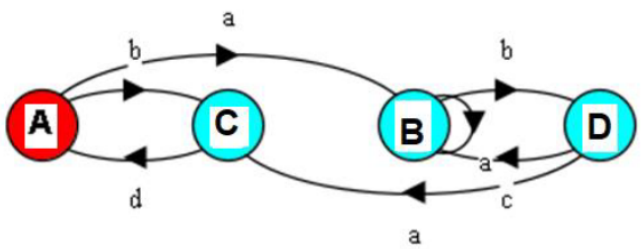


Hints for your assignment

LTSA Tool (commas and dots.)

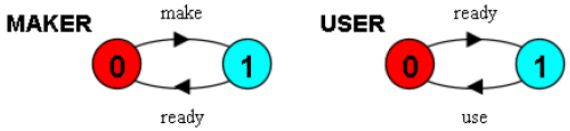
- When do we use the comma?
  - Ex. The description for A is not over until the last line.

```
P1 = A,
A= (a->B|b->C),
B= (a->B|b->D),
C= (d->A),
D= (a->C|c->B).
```



- When do we use the dot?
  - The description for Maker is over in the first line.

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



What is composition

- Parallel composition - action interleaving
- P and Q are processes then (P||Q) represents the concurrent execution of P and Q. The operator || is the parallel composition opera
- | is know as pipe or bar and || is double bar.
- Example from lecture notes which can be found in Chapter 3 - > itch_scratch

```
ITCH  = (scratch->STOP).
CONVERSE = (think->talk->STOP).

||CONVERSE_ITCH = (ITCH || CONVERSE).
```

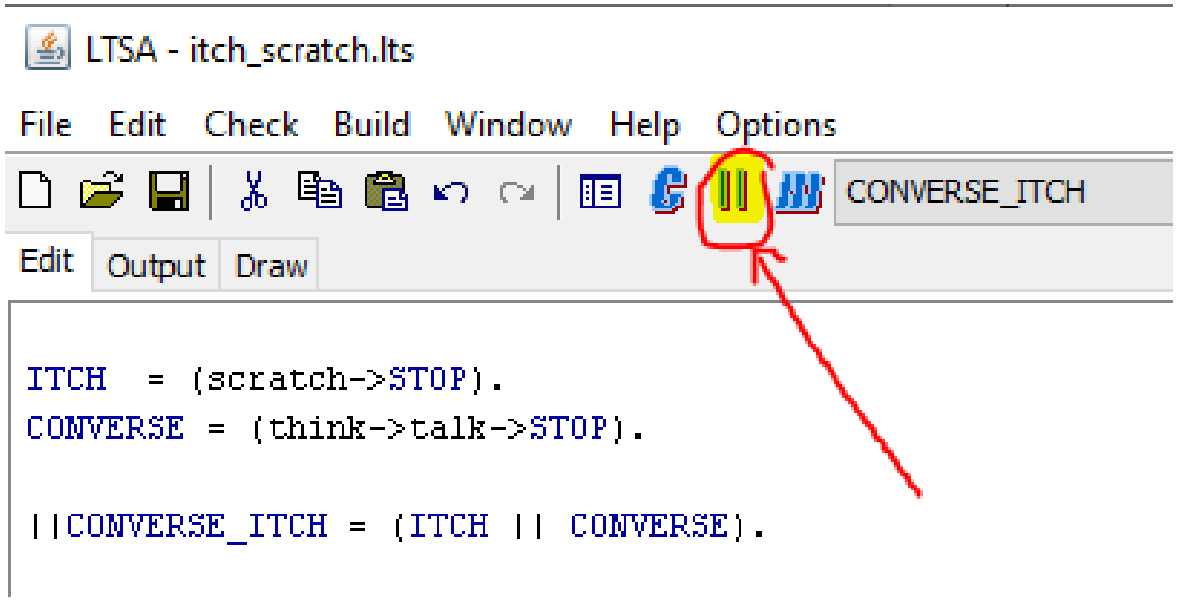
Disjoint alphabets

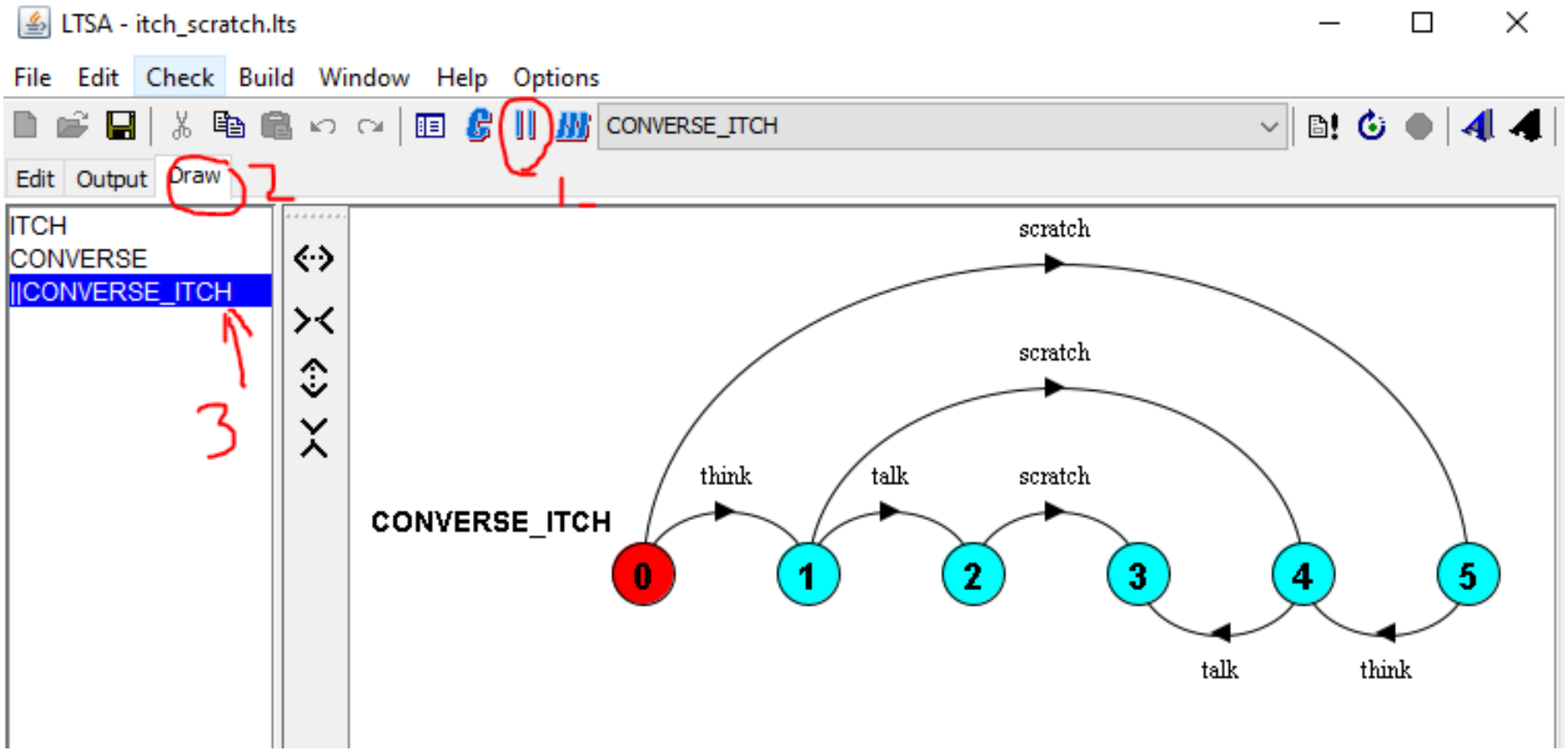
```
think→talk→scratch
think→scratch→talk
scratch→think→talk
```

Possible traces as a result of action interleaving.

Composition in LTSA

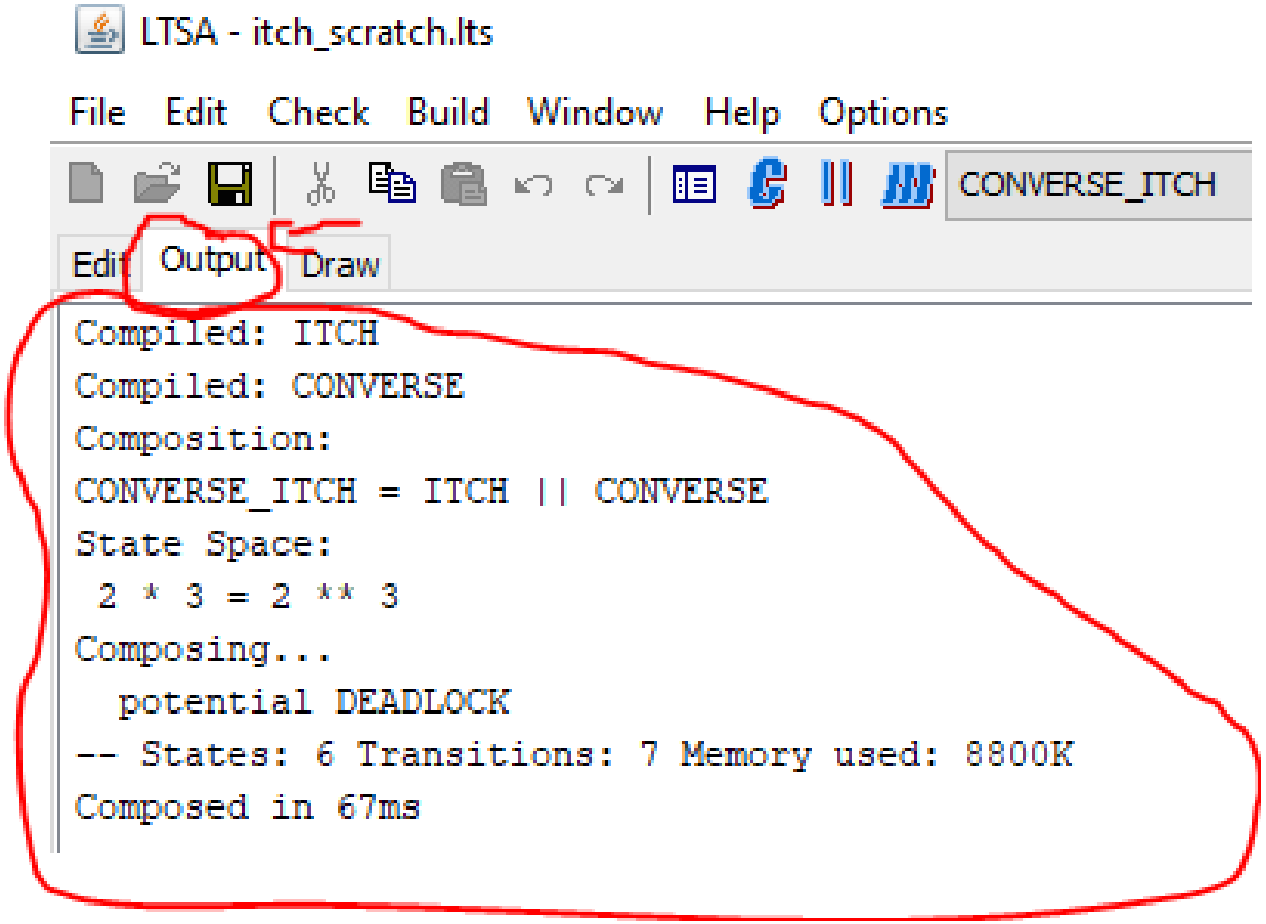
- Chapter 3 - > itch_scratch
- In LTSA tool, in order to draw a composition, you must use the use the **Compose** button instead of ~~**Compile button**~~




LTSA - itch_scratch.lts
File
Edit
Check
Build
Window
Help
Options
CONVERSE_ITCH
Edit
Output
Draw
ITCH
CONVERSE
||CONVERSE_ITCH
1
2
3
CONVERSE_ITCH
scratch
scratch
think
talk
scratch
0
1
2
3
4
5
talk
think

Watch out for any possible errors.

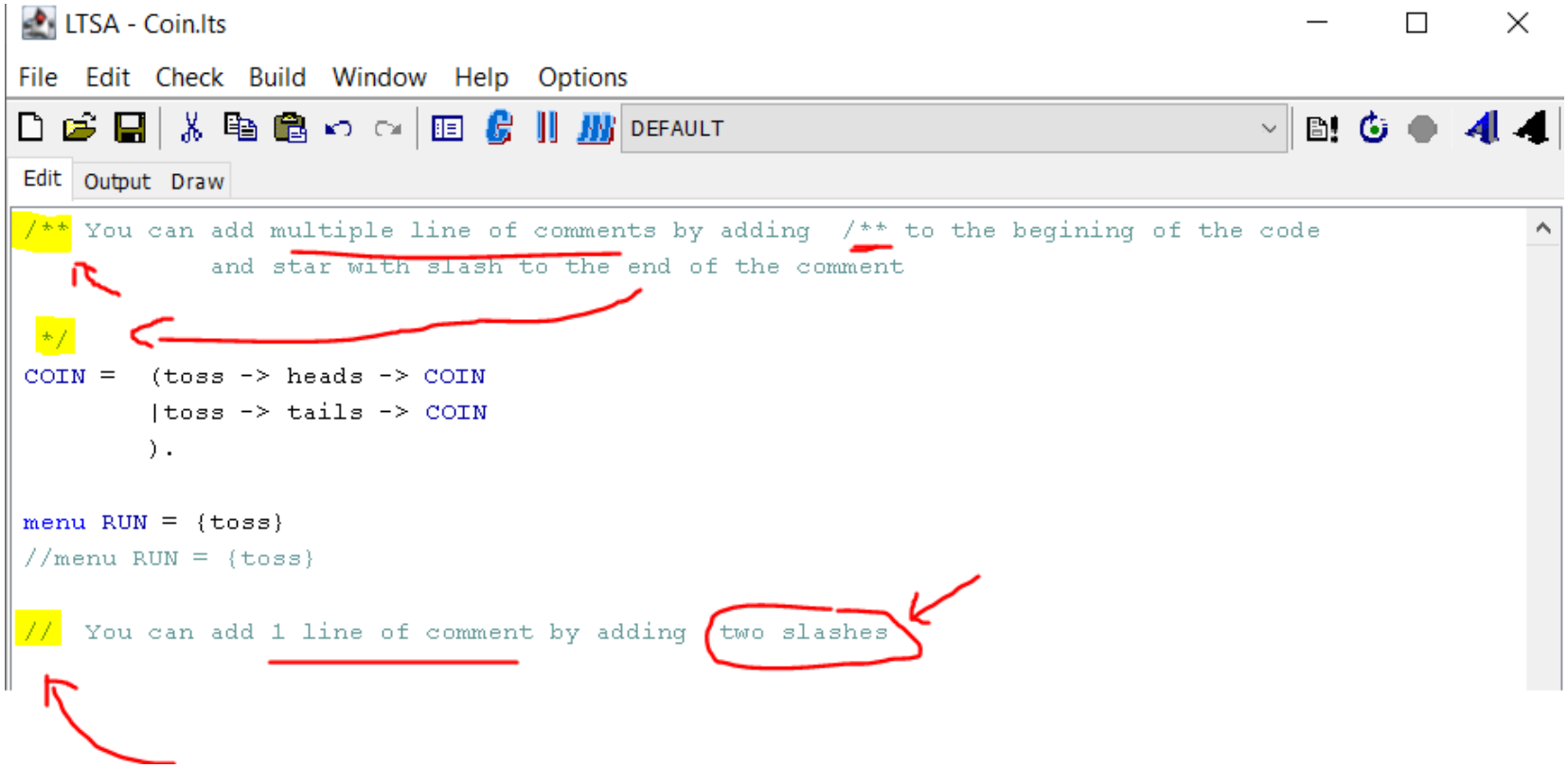
- No errors here
- Some times you must compile first. (1 percent of the time).



Next Tutorial

- Next Tutorial we will learn to draw composition by hand

Adding comment in LTSA tool

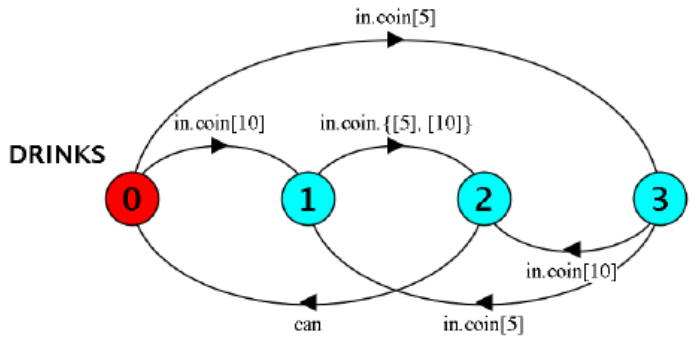


Hint for you assignment

- In your assignment, if you were given a similar question, you can use this code and modify it.
- This example was also used in the lecture 3 notes, slide 4 to describe Petri nets.

Some examples:

- A vending machine charges 15p for a can of Sugarola. It accepts coins of 5p and 10p and does not give change. Model the machine using FSP.




```
DRINKS = CREDIT[0],

CREDIT[0] = (in.coin[5]   -> CREDIT[5]
            |in.coin[10]  -> CREDIT[10]),

CREDIT[5] = (in.coin[5]   -> CREDIT[10]
            |in.coin[10]  -> CAN),

CREDIT[10]= (in.coin[5]   -> CAN
            |in.coin[10]  -> CAN),

CAN  = (can -> DRINKS).
```

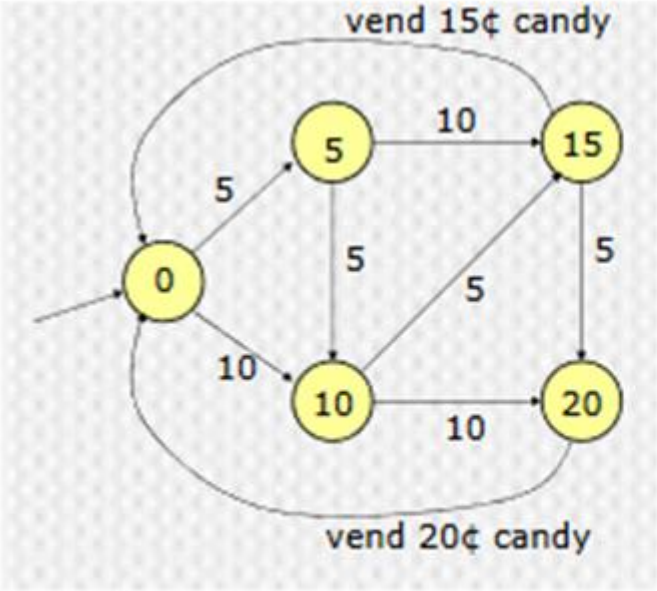
Petri Nets

- A Petri net, also known as a place/transition (PT) net,
  - is one of several mathematical modeling languages for the description of **distributed systems**.
  - It is a class of discrete event dynamic system.
  - A Petri net is a directed bipartite graph, in which
  - The nodes represent transitions.
  - Events that may occur represented by bars and places .
  - Conditions, represented by circles.
  - The directed arcs describe which places are pre- and/or post conditions for which transitions (signified by arrows).

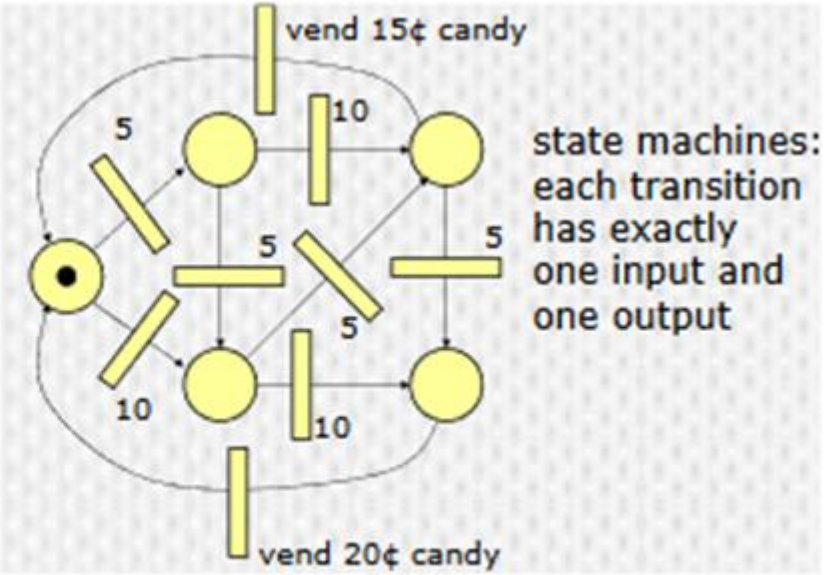
From State Machines to Elementary Petri Nets

Elementary Petri Nets are directed bipartite graphs with

- *places*, represented by circles or ovals (represent some type of resource)
- *transitions*, represented by rectangles or lines (consume and produce resources)
- *arcs* (from places to transitions or transitions to places)
- *tokens*, placed in places
- *initial marking*, tokens is some places



Finite State Machine

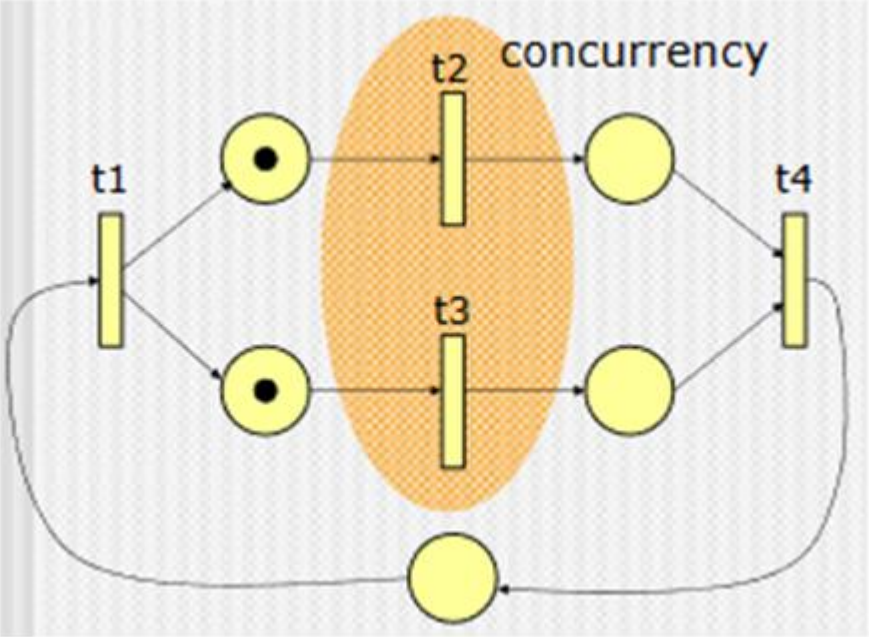


Petri Net

Modeling Concurrency

Elementary Petri Nets are directed bipartite graphs with

- *places*, represented by circles or ovals (represent some type of resource)
- *transitions*, represented by rectangles or lines (consume and produce resources)
- *arcs* (from places to transitions or transitions to places)
- *tokens*, placed in places
- *initial marking*, tokens is some places

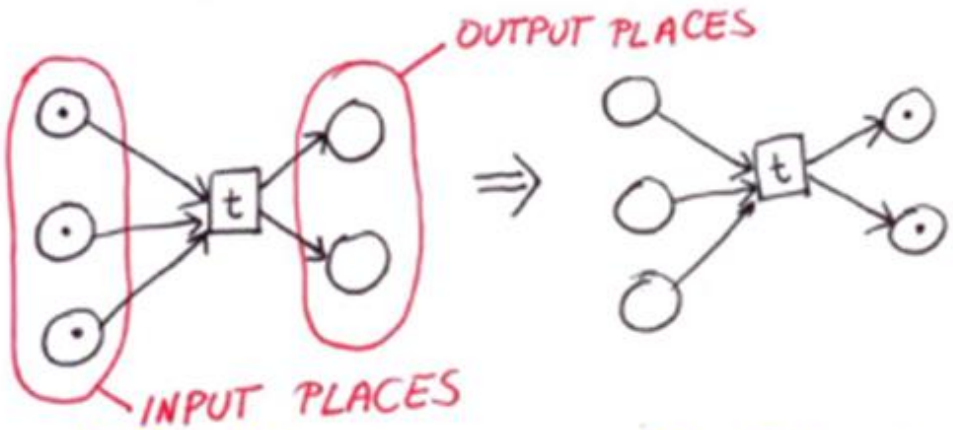


Firing rules for elementary Petri Nets

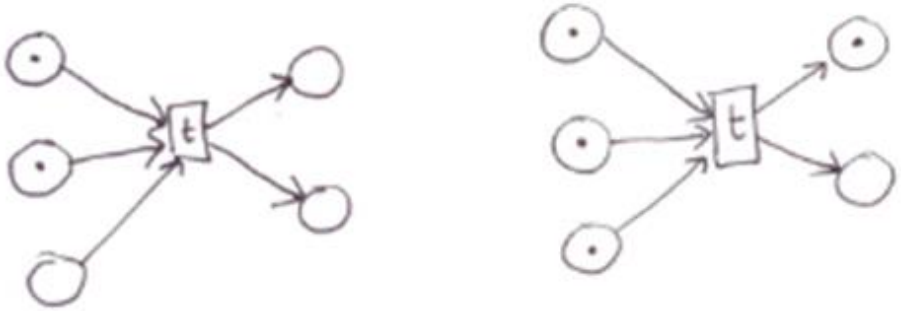
- A transition t can be fired if and only if it has token in all its input places and all its output places are empty.
- After firing, all its input places become empty and all its output places contain tokens.

Firing Rules for Elementary Petri Nets

- A transition t can be *fired* if and only if it has tokens in all its input places and all output places are empty.
- After firing t, all its input places become empty and all its output places contain tokens.



Before fining t After firing t



t can not be fired

Common mistakes when designing a Petri Nets

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

Any Questions?