

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



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:

[Assignment 1](#)

Due Dates:

Monday, October 4, 23:59

Tutorial Notes:

[Tutorial Slides 1](#)

Lecture Dates:

September 14 10:30 AM

Live lectures.

Prerecorded lectures.

[Video Link.](#)

Information related to Teams.

The screenshot displays the Microsoft Teams interface. On the left sidebar, the 'General' channel is selected and circled in red. The main area shows the 'Files' tab, also circled in red, with a red arrow pointing to it from the top right. Below the tab bar, a table lists documents in the 'General' channel. Red arrows point to the 'Lecture Slides' and 'Recordings' rows.

Name	Modified	Modified By
Images	September 10	Jatin Chowdhary
Java Installation for Windows	5 days ago	Mahdee Jodayree
Lecture Slides	3 days ago	Mahdee Jodayree
Recordings	September 7	Mark Hutchison
Tutorial Slides	3 days ago	Mahdee Jodayree

Information related to Teams.

The screenshot displays the Microsoft Teams interface for a team named 'COMPSCI 3SD3 Concurrent ...'. The left sidebar shows the 'Channels' section with 'General' selected. The main area shows the 'Files' tab, which is circled in red with an arrow pointing to it. Below the 'Files' tab, the 'Recordings' section is visible, listing several video recordings.

Channels

- General

Files

Documents > General > Recordings

Name	Modified
CS3SD3 - September 13th - Lecture.mp4	September 13
CS3SD3 - September 7th - Lecture.mp4	September 9
CS3SD3 - September 9th - Lecture.mp4	September 9
Meeting in _General_-20210914_133039-M...	6 days ago
Meeting in _General_-20210916_123015-M...	4 days ago
Tutorial 1 2021 - Tuesday, September 14th, ...	3 days ago

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.

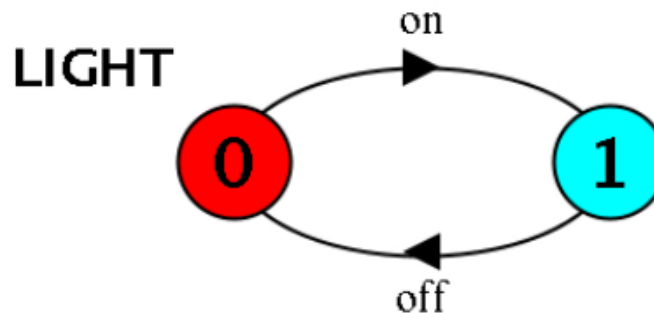


Figure: An LTS of a light

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 *Finite State Processes*.
- FSP is an algebra modelling processes.
- Every FSP description has a corresponding LTS.

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

Figure: An FSP description of a light

Not recommended

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

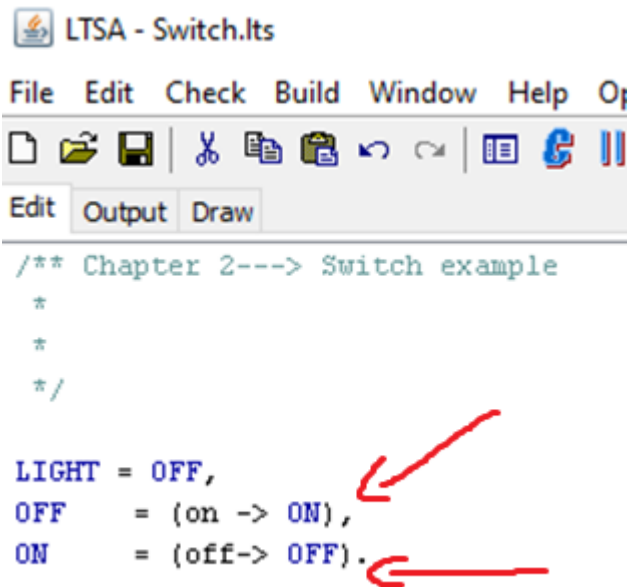
`LIGHT = OFF,`
`OFF = (on -> ON),`
`ON = (off -> LIGHT).`

Recommended way

For assignments

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

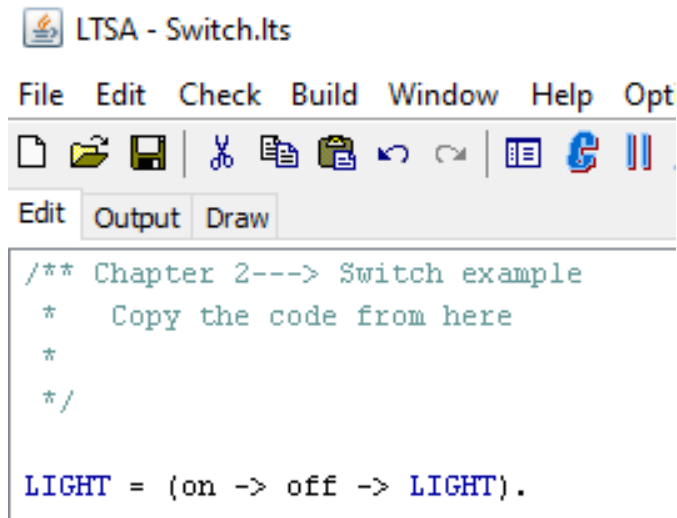
Chapter 2 -> Switch example



```
LTSA - Switch.lts
File Edit Check Build Window Help Op
Edit Output Draw
/** Chapter 2---> Switch example
 *
 *
 */

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

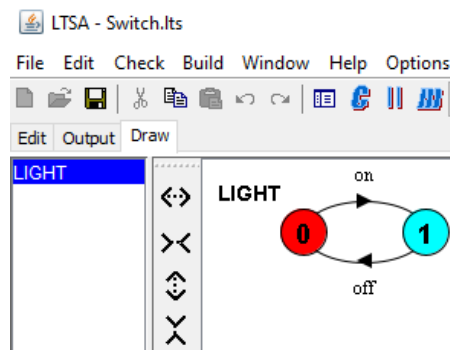
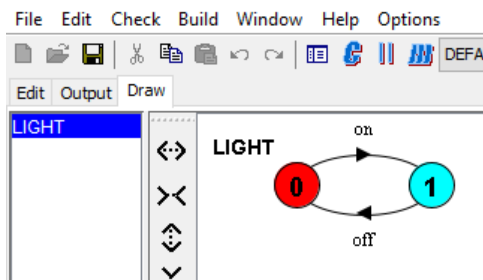
Red arrows point to the initial state definition and the transition definitions.



```
LTSA - Switch.lts
File Edit Check Build Window Help Opt
Edit Output Draw
/** Chapter 2---> Switch example
 *   Copy the code from here
 *
 */

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

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



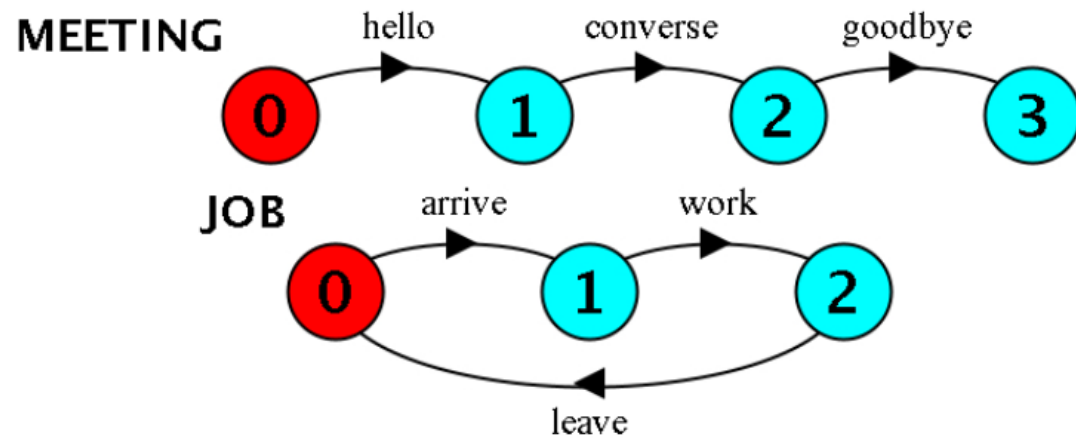
```
LIGHT = OFF,
OFF    = (on -> ON),
ON     = (off-> OFF).
```

Example of LTS to FSP:

Labelled Transition System to Finite State Processes

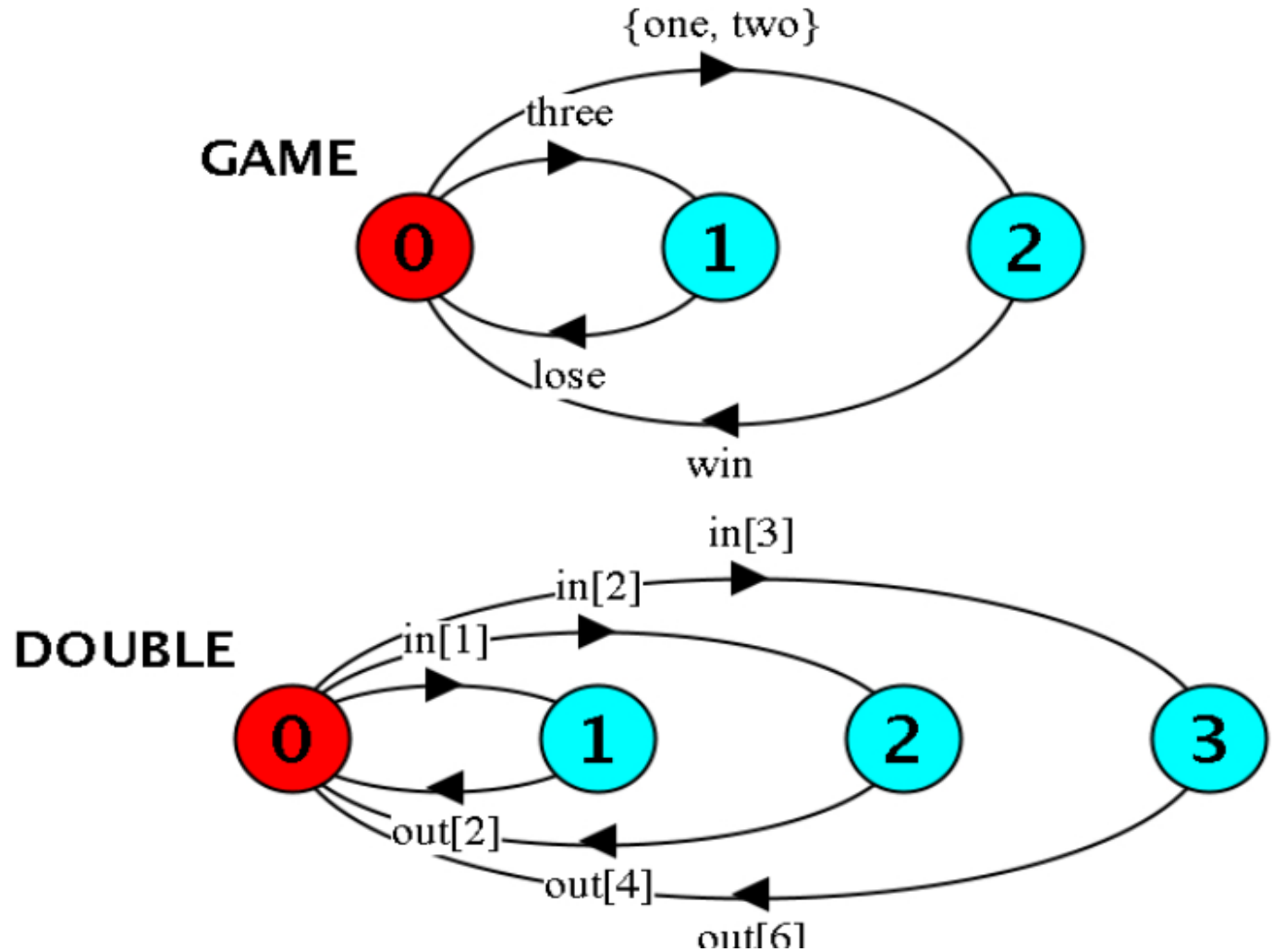
Some examples:

- LTS to FSP:



Some examples:

- LTS to FSP (cont.):



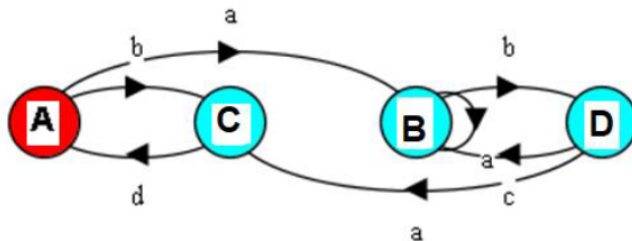
Hints for your assignment

LTSA Tool (commas and dots.)

- When do we use the comma?

- EX. The description for A is not over until the last line.

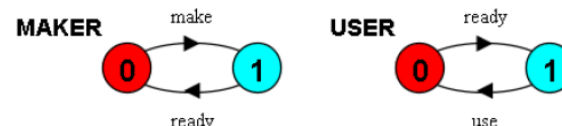
```
P1 = A,  
A= (a->B | b->C) ,  
B= (a->B | b->D) ,  
C= (d->A) ,  
D= (a->C | c->B) .
```



- 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

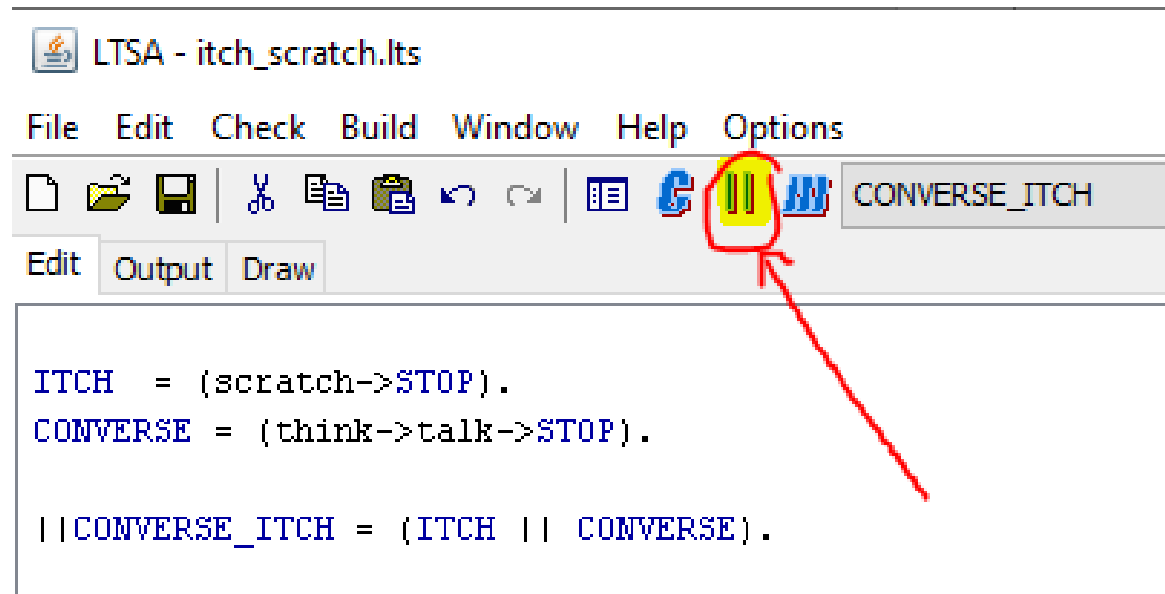
```
think->talk->scratch  
think->scratch->talk  
scratch->think->talk
```

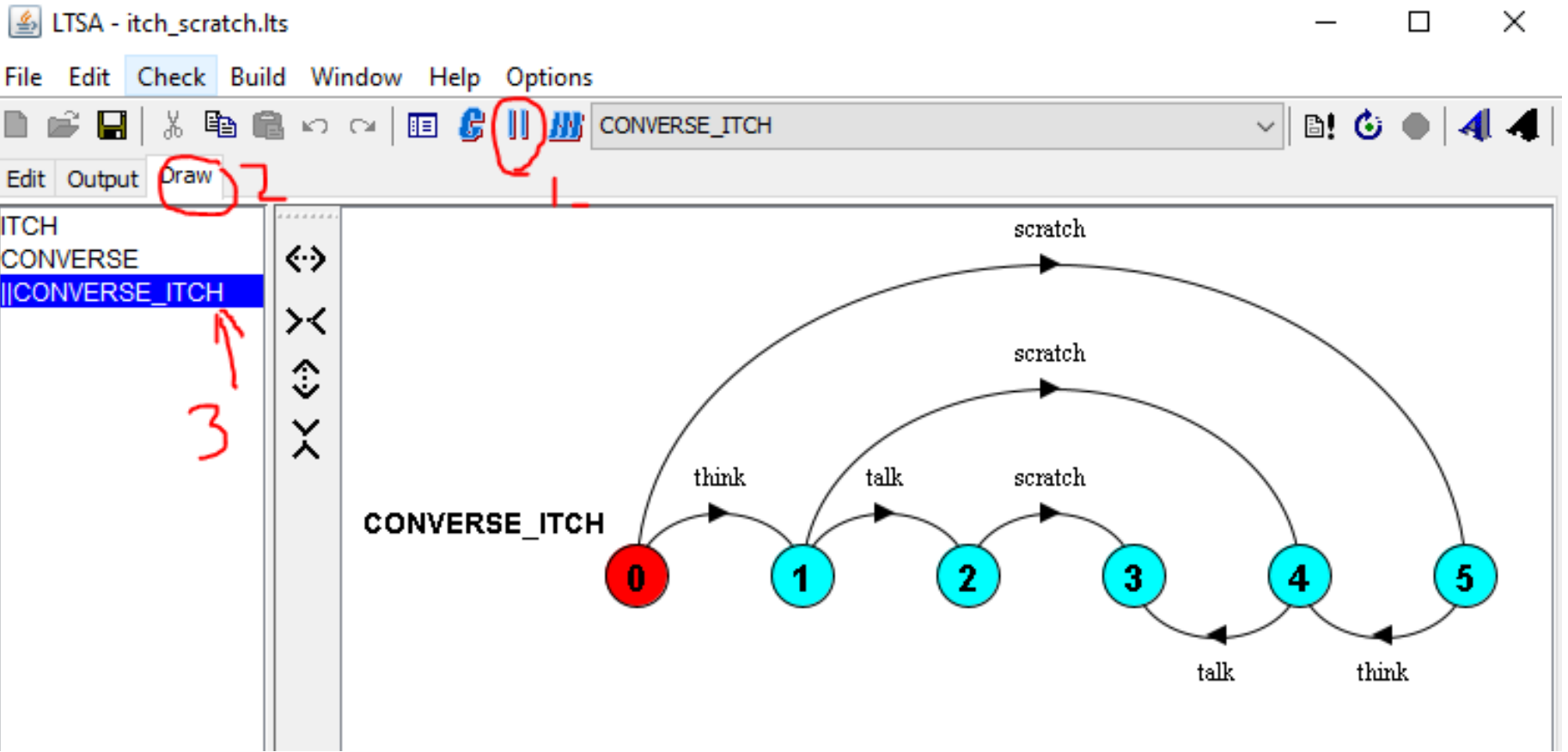
Possible traces as
a result of action
interleaving.

Composition in LTSA

❖ Chapter 3 - > itch_scratch

❖ In LTSA tool, in order to draw a composition, you must use the **Compose** button instead of ~~Compile~~ button

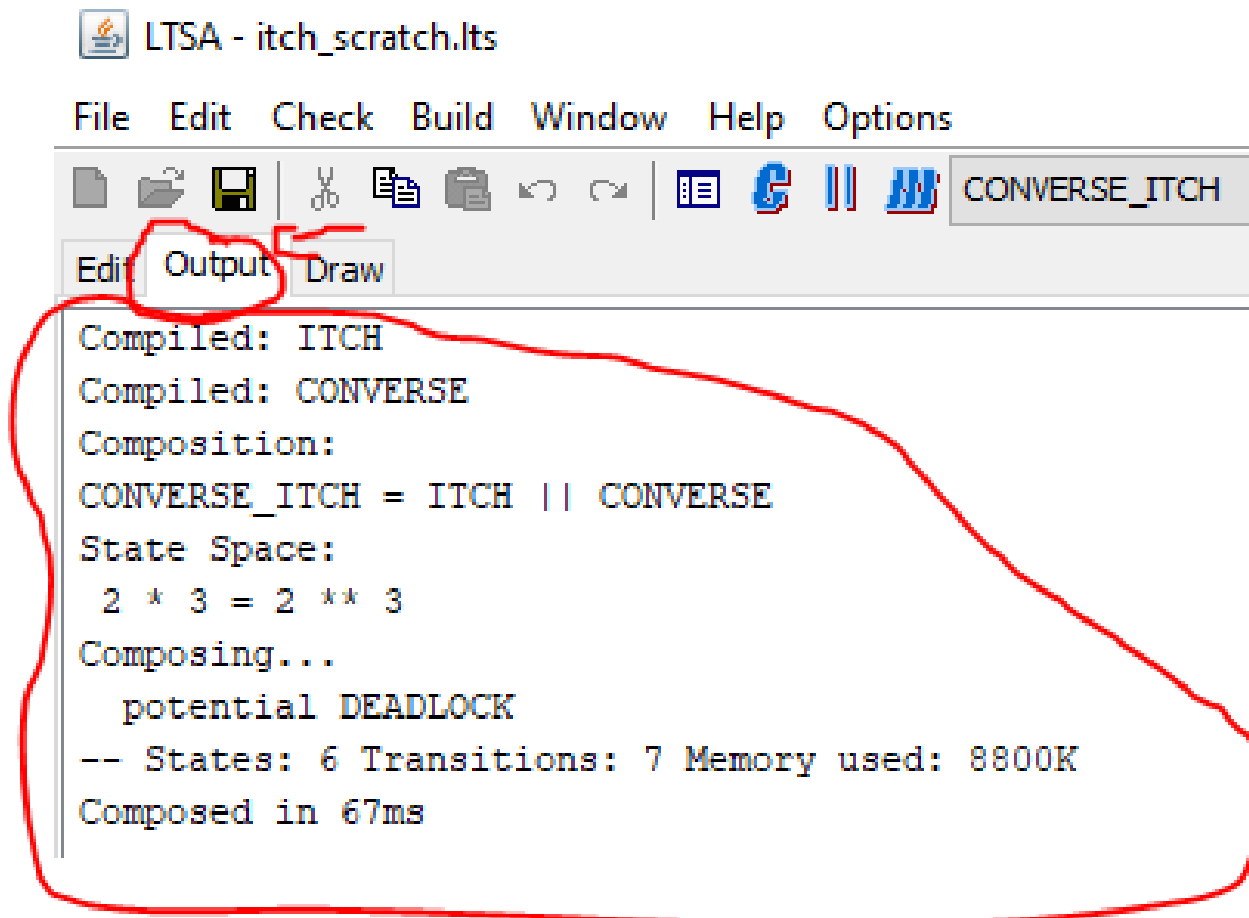




Watch out for any possible errors.

❖ No errors here

❖ Some times you must compile first. (1 percent of the time).



The screenshot shows the LTSA - itch_scratch.lts application window. The menu bar includes File, Edit, Check, Build, Window, Help, and Options. The toolbar contains icons for file operations and a button labeled CONVERSE_ITCH. The 'Output' tab is selected and circled in red. The output text is as follows:

```
Compiled: ITCH
Compiled: CONVERSE
Composition:
CONVERSE_ITCH = ITCH || CONVERSE
State Space:
  2 * 3 = 2 ** 3
Composing...
  potential DEADLOCK
-- States: 6 Transitions: 7 Memory used: 8800K
Composed in 67ms
```

Next Tutorial

❖ Next Tutorial we will learn to draw composition by hand

Adding comment in LTSA tool

The screenshot shows the LTSA - Coin.Its application window. The menu bar includes File, Edit, Check, Build, Window, Help, and Options. The toolbar contains icons for file operations and a dropdown menu set to 'DEFAULT'. The 'Edit' tab is active, showing the following code:

```
/** You can add multiple line of comments by adding /** to the beginning of the code
    and star with slash to the end of the comment

*/

COIN = (toss -> heads -> COIN
       |toss -> tails -> COIN
       ).

menu RUN = {toss}
//menu RUN = {toss}
```

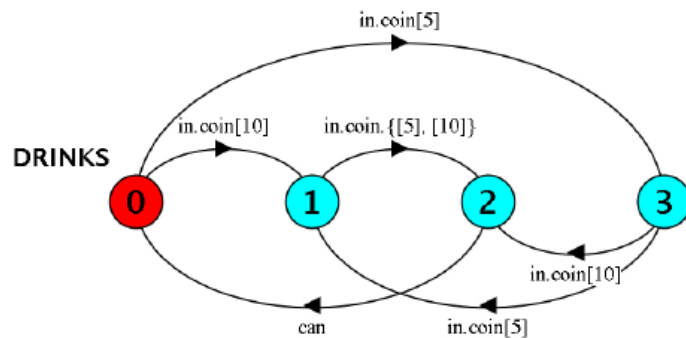
Below the code, a comment example is shown: `// You can add 1 line of comment by adding two slashes`. Red annotations highlight the comment syntax: an arrow points to the opening `/**`, another points to the closing `*/`, and a third points to the `//` in the single-line comment. The phrases 'multiple line of comments' and 'two slashes' are underlined in the original image.

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).
```

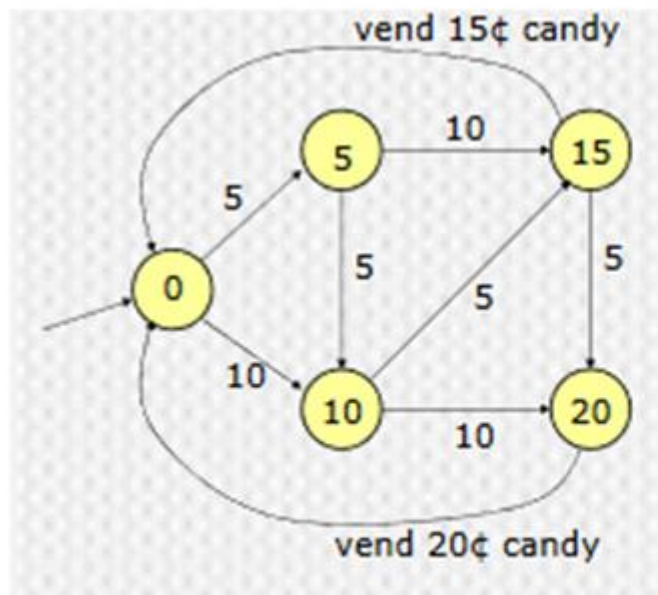
Petri Nets

- ❖ 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).

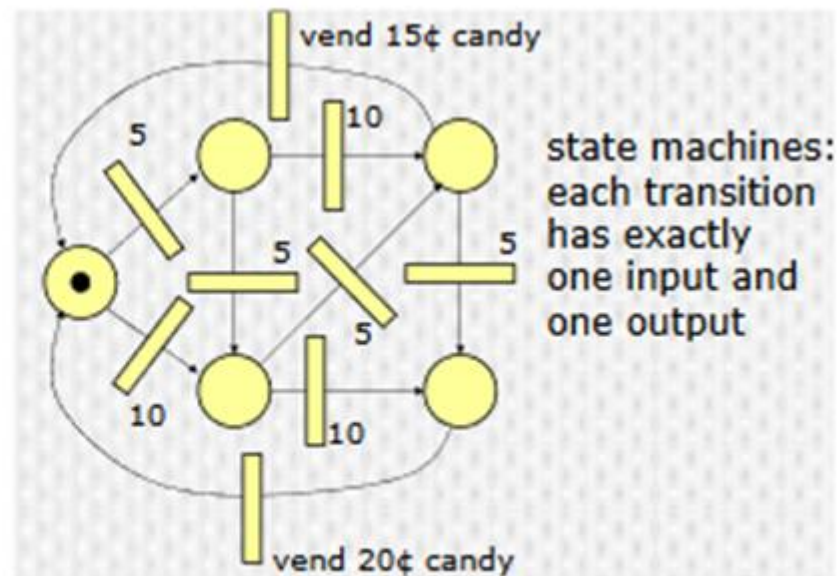
From State Machines to Elementary Petri Nets

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 in some places



Finite State Machine



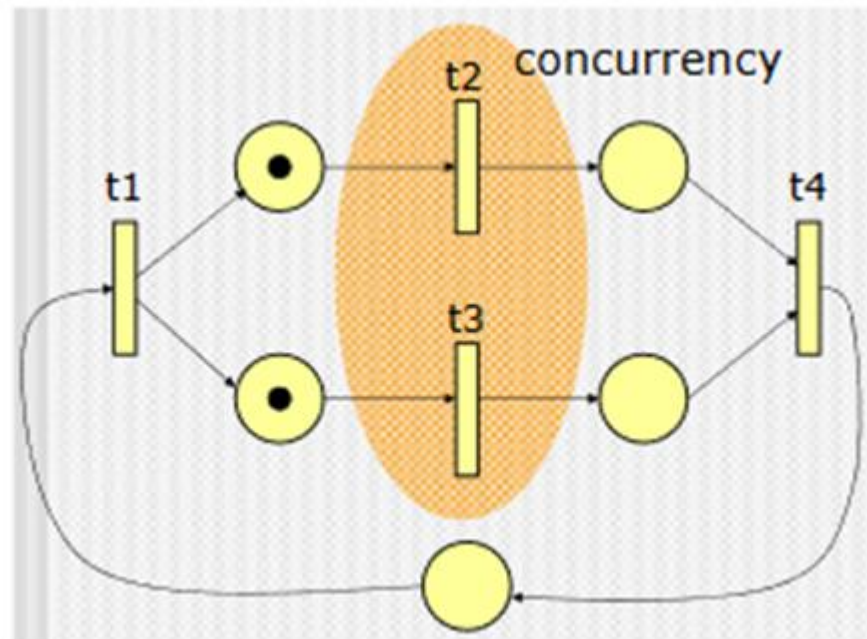
state machines:
each transition
has exactly
one input and
one output

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 in some places

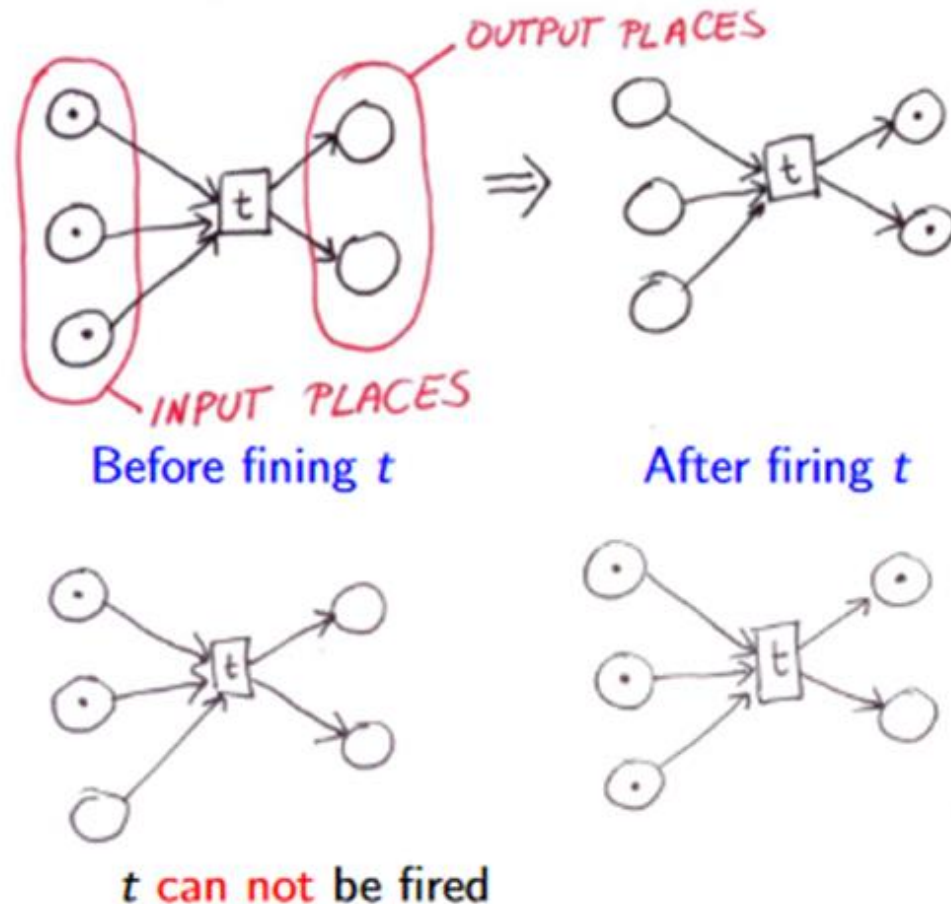


Firing rules for elementary Petri Nets

- ❖ 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.



Common **mistakes** when designing a Petri Nets

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

Any Questions?
