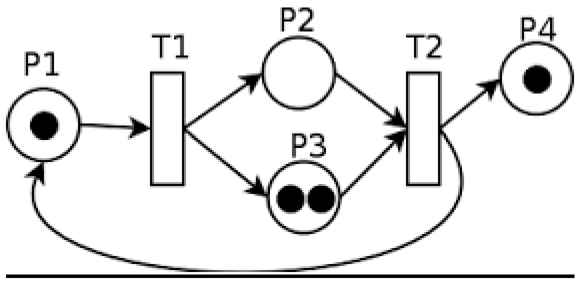
Introduction

This project is meant to help the student to build a compiler for a language dedicated Petri net programming.

The student is required to build all phases of compiler from the mathematical definition of the language until the generation of the lower level code.

The student is required to design and develop a programming language and a compiler for Petri net programming.

What are Petri nets



A Petri net is a directed graph in which the nodes are either transitions (represented as rectangles), or places (represented as circles). A place is connected to one or more transitions and a transition is connected to one or more places.

Nodes sharing identical types are not directly connected. When different types of nodes are connected, they are so through directed arcs. Activities that are performed by the transitions are represented by tokens (solid circles) which reside in places.

An empty place *p* connected to transition *t* disables this transition from being executed. A transition is said to be enabled if and only if there are no empty places connected to it as inputs. A transition executes (or fires) after being enabled and the result of a firing is the removal of tokens from each of its input places and the creation of tokens in each of its output places.

Learning Outcomes

The student will learn to design and develop a programming language from scratch to perform certain purpose. In this project the control of a mobile robot is selected since it would be a problem that is close to their knowledge as engineers.

<u>Deliverables</u>

- **Deliverable 2: Design of the scanners and grammar of the language. (3 marks), Due by week 10**
    - Each Team is required to design the scanner and parser of the given language. In order to do this you need to do the following:
        - Alphabet
        - Language strings
        - Regular expression and finite state automata
        - Regular grammar and Context free grammar (1 mark)
        - The Lex of the scanner (3 marks)
- **Deliverable 3: (4 marks), Due by week 15 :**

  The Parser of the language using Yacc (3 marks).

  Changing the code into low level format (1 mark).
- **Deliverable 4: (4 marks), Due by week 16 :**

  Presentation and final report (individual evaluation)

Example of the proposed language:

```
Begin
Define P1,P2,P3,P4 as Place;
Define T1,T2,T3 as Transition;
P1 is net input;
P4 is net output;
P1 is input to T1,T2;
T3 is connected to T2 via P3; // this means that P3 is output to T2 and input to T3
T3 is connected to T1 via P2; // this means that P3 is output to T1 and input to T3
Mark P1 with 6 Tokens;
Loop
   If T1 is enabled then Fire T1;
   If T2 is enabled then Fire T2;
   Favour T1 over T2;
   Display Marking
Until P4 has 6
End
```

Intermediate Language:

```
Start
Place P1
Place P2
Place P3
Place P4
Transition T1
Transition T2
Transition T3
Input P1
Output P4
Connect P1, T1
Connect P1, T2
Connect T2,P3
Connect P3,T3
Connect T1,P2
Connect P2,T3
Mark P1,6
Loop:
          Enabled T1, Jump1
 Back: Enabled T2, Jump2
          Continue
Jump1 : Fire T1
             Back
Jump2: Fire T2
Continue: Favour T1
Output P1
Output P2
Output P3
Output P4
Contain P4,6 , Loop
Stop
```

The intermediate language is the generated code after compiling the high level code.

The marks for each and every deliverable is divided as follows:

- 50% group work (overall evaluation of work and deliverable quality)
- 50% individual comprehension and understanding of the deliverable.
  So every individual student will be asked about every tiny detail in that deliverable.

Plagiarism

- Please check the students’ handbook for more on information on plagiarism.
- You will be held responsible if someone copies your work - unless you can demonstrate that have taken reasonable precautions against copying.