

# Transforming Flowchart into Coloured Petri Nets

Utumporn Gulati

Department of Computer Engineering  
Faculty of Engineering, Chulalongkorn University  
Bangkok, Thailand  
6071035321@student.chula.ac.th

Wiwat Vatanawood

Department of Computer Engineering  
Faculty of Engineering, Chulalongkorn University  
Bangkok, Thailand  
wiwat@chula.ac.th

## ABSTRACT

Flowchart is a common graphical representation of a process or step-by-step solution for a problem simply drawn in the software design stage. A flowchart helps visualize how an algorithm works to solve a problem better than the program source code. However, it is crucial and beneficial to validate the correct behavior of the flowchart automatically using the formal methods. In this paper, we propose a scheme of transforming a basic flowchart into a coloured Petri net so that the designated model could be validated beforehand. A set of mapping rules is proposed to construct the resulting coloured Petri net from a given flowchart. The basic data types, including integer, boolean, and string, are coped in order to simulate the changing of state of variables in the flowchart. The resulting coloured Petri net are simulated and verified to ensure the correctness of its behavior using CPN tools.

## CCS Concepts

• Software and its engineering→Software organization and properties→Software system structures→Software system models→Petri nets • Software and its engineering→Software organization and properties→Software functional properties→Formal methods→Model checking.

## Keywords

Flowchart, coloured Petri nets, Transformation scheme, Formal method

## 1. INTRODUCTION

Flowchart is a graphical representation of a process or step-by-step solution of a problem. It contains a suitably annotated geometric figures connected by flowlines, as to help visualize the design or document a process or algorithm [1]. Typically, flowchart is used in the design stage as to represent the processes of the software system before going to be developed. It eases all stakeholders to clearly understand quite well on the data flows and control flows within the system. However, the supporting tools are scarcely proposed to automatically simulate or verify the behaviors of a flowchart. It should be crucial to ensure the correctness of the flowchart automatically using formal methods. A formal model would be used to verify the system behaviors.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [Permissions@acm.org](mailto:Permissions@acm.org).

ICSEB 2019, December 9–11, 2019, TOKYO, Japan

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-7649-5/19/12...\$15.00

<https://doi.org/10.1145/3374549.3374568>

Many researches proposed the scheme to convert business models or UML models into formal models, enabling the formal verifications. [2] proposed coloured Petri nets to verify the behavior of the model formally. Whilst, [3] proposed a set of mapping rules to convert UML activity diagram into coloured Petri nets with inscription in order to verify the behavior of the model and to ensure the correctness of its design. [4] proposed an approach to transform UML state machine diagram into coloured Petri nets using Isabelle/HOL. Some researches exploited Petri nets [5] as a formal model, such as [6] proposed a set of mapping rules to convert dataflow diagram into Petri nets in order to manage the resources in the system. In [7], the rules were proposed to translate business rules into CPN ML function and completely do the inscription onto the CP-net files.

In this paper, we propose the scheme of transforming a flowchart into a formal model, called coloured Petri net. A coloured Petri net was an extension of the original Petri net to cope with the distinguishing of the types or colours of the tokens and their firing functions. In short, both data flow and control flow of the coloured Petri net are possibly concerned. The resulting coloured Petri net would represent the changing state of all variables specified in the flowchart and only basic data types of variables are covered, such as boolean, integer, string, etc.

The rest of this paper is organized as followed. Section 2 is the backgrounds. Section 3 describes our transforming scheme. Section 4 shows our demonstration and case study. Section 5 is our conclusion.

## 2. BACKGROUND

### 2.1 Flowchart

Flowcharting is a simple graphical method of representing program sequences and algorithms using a standard set of symbols to represent program flow [8]. Flowchart is a kind of diagram that represents the concept, work procedures, or the work process of the system. Flowchart is also a tool that helps show the details of work in each step of the system including decisions, iterations, and results which helps see the direction of the system more clearly. The flowchart shows each step of the work process representing with a square box, which is connected by an arrow to represent workflow sequence. The flowchart is used in the design process and documenting the process of the system, which helps simulate the work process and show what happens in each process of the system. Making it possible to understand the work process and see the problems that may occur such as bottlenecks.

A flowchart is composed of a set of processes, decisions, and connecting arrow. A sample of flowchart element [8] is shown in Figure 1. The description of each element as follows. Terminal element indicates the starting or ending point of the flowchart. Process element indicates a defined operation or set of operations. Decision element indicates the conditional operation that

determines which one of the two parts the program will take. Flowline element indicates the process's order of operation. Connector element indicates the point that connected from another part of the chart.

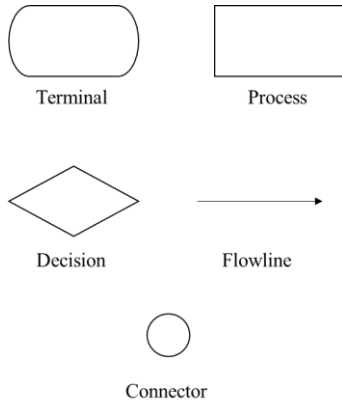


Figure 1. A Sample of flowchart element [9].

## 2.2 Coloured Petri Nets (CPN)

CPN [2] is a graphical oriented language that very popular in not only to simulate the behavior of the system, but also to verify the correctness of the system as well. CPN is the Petri nets combines with the standard ML language [10] to improve the ability of the basic Petri nets.

## 2.3 Inscription in CPN Tools

CPN tools [11] allows us to insert conditions on each part of the model called inscription so that the developed model would be complied with specified conditions. In this paper, we focus only two types of the inscription which are place inscription and arc inscription as follows.

Place inscription is the condition that could be defined on each place. There are three parts of inscriptions that are associated with a particular place. The first one is required as the colour set inscription of a place  $p$ , denoted by  $C(p)$ . The second one is optional called the place name inscription. The third is also optional called initial marking. The sample of place inscription as shown in Figure 2

Arc inscription is the condition that could be defined on each arc. An arc inscription is a CPN ML expression that would be evaluated to a multiset or a single element. An arc inscription is a CPN ML expression  $E(a)$  on arc  $a$ . The CPN ML expression could be if-else condition in order to select the appropriate data flow through the specified arc. The sample of arc inscription as shown in Figure 3

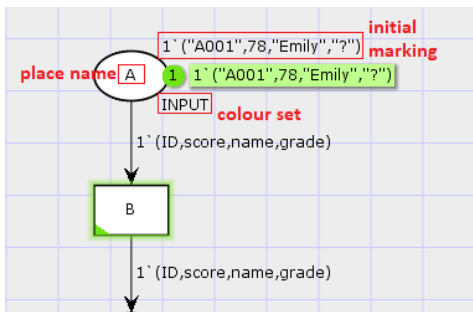


Figure 2. The sample of place inscription

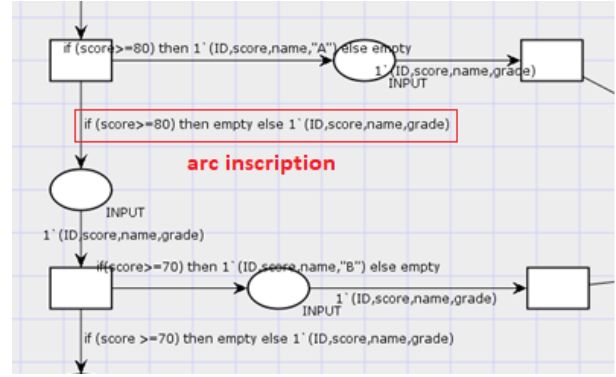


Figure 3. The sample of arc inscription

## 3. OUR MODEL TRANSFORMING APPROACH

In this section, we present the overview of our model transforming approach as shown in Figure 4. The XML file of the flowchart is imported and extracted to the elements of flowchart. And we also propose transformation rules to map the elements of flowchart into the elements of CPN. The result of CPN model is successfully generated and verified by CPN tools.

### Definition 1: Flowchart

A flowchart is an 8-tuple  $FC = (T, P, D, C, CO, F, TS, TE)$  such that

$T$  is a set of terminal nodes

$P$  is a set of process nodes

$D$  is a set of decision nodes

$C$  is a set of conditions

$F$  is a set of flowlines

$CO$  is a set of connector nodes

$TS \subseteq T$  is a set of start nodes

$TE \subseteq T$  is a set of end nodes

In our approach, the process attached to the start node would be the variable declaration statements with basic data types, such as STRING, INT, BOOLEAN.

### Definition 2: CPN

A coloured Petri net is a 9-tuple  $CPN = (PL, TR, A, \Sigma, V, M0, IN, INP, INA)$  such that

$PL$  is a finite set of places

$TR$  is a finite set of transitions

$A$  is a finite set of directed arcs

$\Sigma$  is a finite set of non-empty colour sets

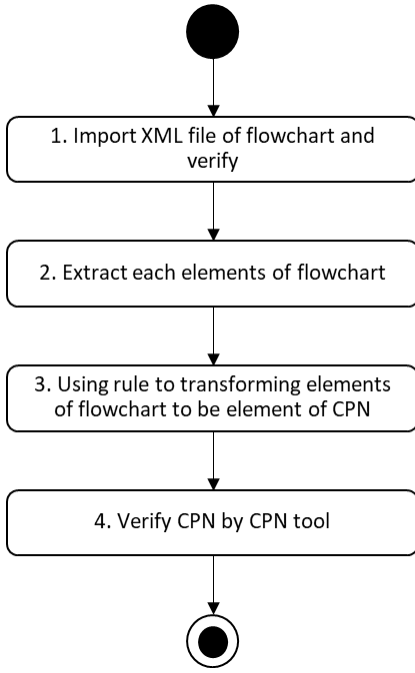
$V$  is a finite set of variables

$M0$  is the initial marking

$IN$  is a set of CPN inscription

$INP \subseteq IN$  is a set of place inscription

$INA \subseteq IN$  is a set of arc inscription



**Figure 4. The overview of our Transforming Approach**

### 3.1 Import Flowchart Diagram

Flowchart diagram that represents the processes of the system needs to be prepared in well-formed XML format. So, we can extract element of flowchart from the XML structure.

### 3.2 Extract Element of Flowchart

After the flowchart diagram in XML format is imported from previous step. In this step, we intend to extract the flowchart diagram written in XML format into a set of flowchart elements. Each element is considered to match the set of elements of workflow found in Table 1.

### 3.3 Translate the Elements of Flowchart into CPN Elements

In this step, we propose a set of mapping rules to transform the elements of flowchart extracted into the portions of CPN model. Our mapping rules are visually shown in Table 1, and described as follows.

#### 3.3.1 Mapping Start Node

A given flowchart begins with a start node  $ts \in TS$  and the attached process node  $p_1 \in P$  which do only the declaration of variables in the flowchart. These two nodes are mapped into two places  $pl_1, pl_2 \in PL$  and a transition  $tr_1 \in TR$  in between, connecting with two arcs  $a_1, a_2 \in A$ , as shown in Table 1. The process node  $p_1$  would be considered on all variables, data types, and their initial values, in order to generate the CPN variable set  $V$ . The data types would guide us to generate CPN colour set  $colset \in \Sigma$ . This colset would be attached as the inscription to every place  $pl \in P$  in CPN. The initial values of all variables defined in flowchart process node  $p_1$  would be mapped into the initial marking  $m \in M0$ . Every particular CPN arc  $a_i \in A$  would be attached with its relevant inscription  $ina_i \in INA$  and be ready for the CPN firing operation.

#### 3.3.2 Mapping Process Node

Our process node mapping rule would be divided into two parts as follows.

Part 1: Process node preceded by a decision node

A process node  $p \in P$ , preceded by a decision node  $d \in D$ , would be mapped into an arc  $a \in A$  and a place  $pl \in PL$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. This arc  $a$  would be attached with the arc inscription  $ina \in INA$  extracted, from the condition  $c \in C$  of the flowchart decision node  $d$ .

Part 2: Process node preceded by a process node

A process node  $p \in P$ , preceded by a previous process node  $q \in P$ , would be mapped into a transition  $tr \in TR$  and a place  $pl \in PL$ , connecting with arcs  $a_1, a_2 \in A$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. These arc arcs  $a_1, a_2$  would be attached with the arc inscription  $ina_1, ina_2 \in INA$  extracted, from the statement of the flowchart process node  $q$ .

#### 3.3.3 Mapping Decision Node

Our decision node mapping rule would be divided into two parts as follows.

Part 1: Decision node preceded by a process node

A decision node  $d \in D$ , preceded by a process node  $p \in P$ , would be mapped into an arc  $a \in A$  and a transition  $tr \in TR$ . This arc  $a$  would be attached with the arc inscription  $ina \in INA$  extracted, from the statement of the flowchart process node  $p$ .

Part 2: Decision node preceded by a decision node

A decision node  $d \in D$ , preceded by a previous decision node  $e \in D$ , would be mapped into a place  $pl \in PL$  and a transition  $tr \in TR$ , connecting with arcs  $a_1, a_2 \in A$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. These arc arcs  $a_1, a_2$  would be attached with the arc inscription  $ina_1, ina_2 \in INA$  extracted, from the condition  $c \in C$  of the flowchart previous decision node  $e$ .

#### 3.3.4 Mapping Connector Node

Connector node mapping rule can be divided into three parts such that

Part 1: Connector node preceded by multiple process nodes

A connector node  $co \in CO$ , preceded by multiple process nodes  $p_1, p_2, p_3, \dots \in P$ , would be mapped into multiple arcs  $a_1, a_2, a_3, \dots \in A$  and multiple transitions  $tr_1, tr_2, tr_3, \dots \in TR$  as shown in Table 1. These arcs  $a_i$  would be attached with the arc inscriptions  $ina_1, ina_2, ina_3, \dots \in INA$  extracted, from the statements of the flowchart process nodes  $p_i$ .

Part 2: Connector node followed by a process node

A connector node  $co \in CO$ , followed by a process node  $p \in P$ , would be mapped into multiple arcs  $a_1, a_2, a_3, \dots \in A$  and a place  $pl \in PL$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. These arcs  $a_i$  would be attached with the arc inscriptions  $ina_i \in INA$  extracted, from the statements of the flowchart process nodes preceded it.

Part 3: Connector node followed by a decision node

A connector node  $co \in CO$ , followed by a decision node  $d \in D$ , No CPN elements are generated for this part. However, the condition  $c \in C$  of the decision node  $d$  would be extracted for the arc inscriptions.

### 3.3.5 Mapping End Node

Our end node mapping rule would be divided into three parts as follows.

Part 1: End node preceded by a connector node

An end node  $te \in TE$ , preceded by a connector node  $co \in CO$ , would be mapped into multiple arcs  $a_1, a_2, a_3, \dots \in A$  and a place  $pl \in PL$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. These arcs  $a_i$  would be attached with the arc inscriptions  $ina_i \in INA$  extracted, from the statement of the flowchart process node preceded it.

Part 2: End node preceded by a decision node

An end node  $te \in TE$ , preceded by a decision node  $d \in D$ , would be mapped into an arc  $a \in A$  and a place  $pl \in PL$ . The place  $pl$  would be attached with the place inscription as a colour set  $colset \in \Sigma$  which has been created earlier. This arcs  $a$  would be attached with the arc inscriptions  $ina \in INA$  extracted, from the condition  $c \in C$  of the flowchart decision node  $d$ .

Part 3: End node preceded by a process node

An end node  $te \in TE$ , preceded by a decision node  $d \in D$ , needs to map into a place, and there is a place  $pl \in PL$  that already generated by another part of rule, so that nothing must be generated for this particular part of rule.

### 3.4 Verify the CPN Model Result with CPN tools

After the resulting CPN model for the given flowchart is generated, we verify the correctness of the desirable properties using CPN tools [11]. The resulting CPN could simulate the critical paths of the desirable behaviors of the original flowchart.

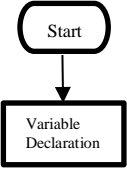
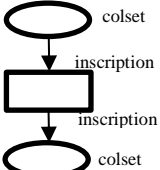
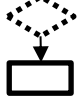
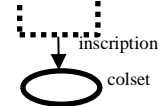
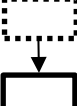
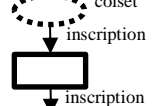
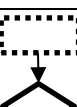
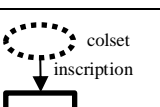

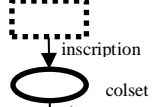
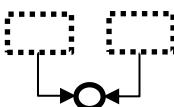
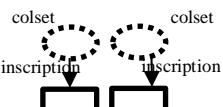
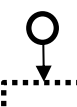
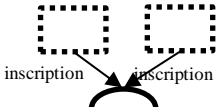

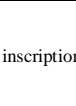
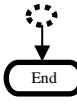
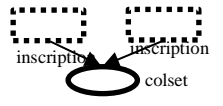
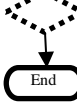
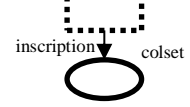
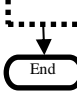

## 4. DEMONSTRATION AND CASE STUDY

In this section, we demonstrate our model transforming approach as mentioned earlier. The original flowchart of a score grading process is used as the case study, as shown Figure 5. The score grading process flowchart begins with the start node  $ts \in TS$  which is followed by a declaration process node  $P_1 \in P$ . All variables are initialized in this process node  $P_1$ . The score grading flowchart classifies the grade of A if the score is greater or equal than 80 and the grade of B if the score is greater or equal than 70 but less than 80. The flowchart shows the grading of C, D, and F as shown in Figure 5.

Firstly, we extract element of flowchart as following:

- One of start node:  $ts$  and the following declaration process node:  $P_1$
- Five of process nodes:  $P_2, P_3, P_4, P_5, P_6$
- Four of decision nodes:  $D_1, D_2, D_3, D_4$
- Four of conditions:  $C_1, C_2, C_3, C_4$ ,
- One of connector nodes:  $CO_1$
- Sixteen of flowlines:  $F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9, F_{10}, F_{11}, F_{12}, F_{13}, F_{14}, F_{15}, F_{16}$
- One of end node:  $te$

**Table 1. Mapping rule for transforming flowchart into CPN.**

No.	Name	Flowchart	CPN
1	Start		
2	Process		
			
3	Decision		
			
4	Connector		
			
			
5	End		
			
			

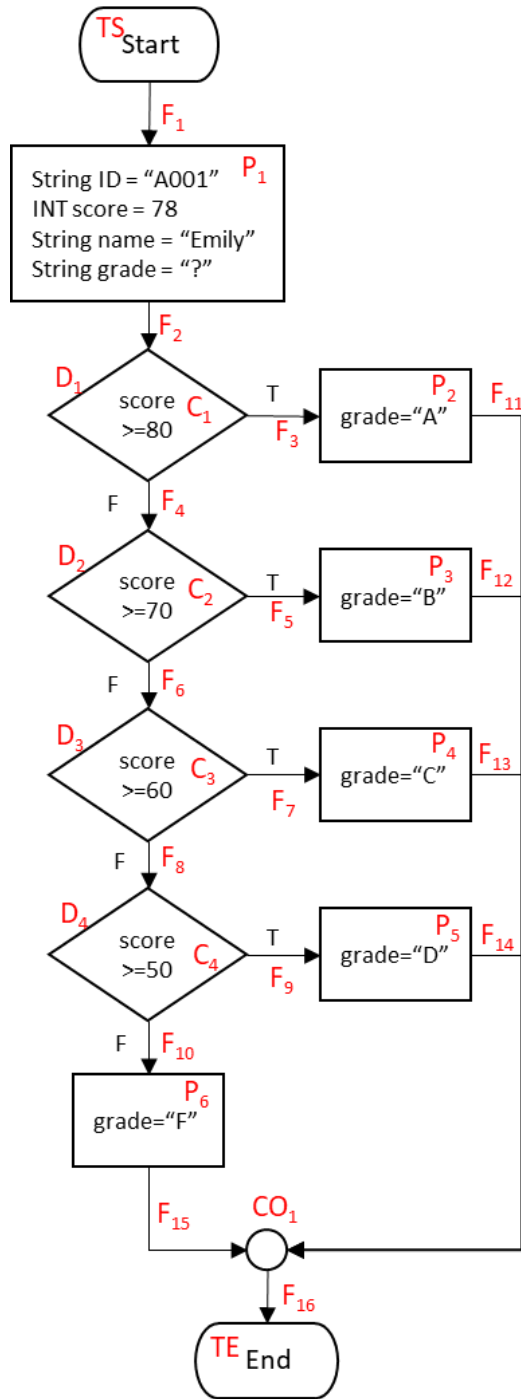


Figure 5. Sample of flowchart

Secondly, we use mapping rules shown in Table 1 to transform flowchart into CPN model by these following steps.

Step1: Colour set in CPN will be generated from all data types declared in process node  $P_i$ . In case there are more than one data types, product colour set will be generated as shown in Figure 6.

Step2: Variables in CPN will be generated from all variable declared in process node  $P_i$  as shown in Figure 7.

Step3: The flowchart start node  $ts$  and the following declaration process node  $P_i$  are mapped into the CPN first place  $pl1 \in PL$ , with the initial marking extracted from declaration process node  $P_i$ . The CPN transition  $tr1 \in TR$  and a CPN place  $pl2 \in PL$  are drawn. The arc inscriptions  $ina1, ina2 \in INA$  of the arcs  $a1, a2 \in A$  are generated. Following the mapping rules, the rest of the flowchart nodes would be mapped into the final resulting CPN as shown in Figure 8. The CPN is simulated using CPN tools and the score grading of B is shown finally for “Emily” with score = 78.

```

▼Declarations
  ▶Standard priorities
  ▼Standard declarations
    ▶colset UNIT
    ▶colset BOOL
    ▼colset INT = int;
    ▼var score: INT;
    ▶colset INTINF
    ▶colset TIME
    ▶colset REAL
    ▼colset STRING = string;
    ▼var name, grade, ID: STRING;
    ▼colset INPUT = product STRING*INT*STRING*STRING;

```

Figure 6. Colour set generated in CPN

```

▼Declarations
  ▶Standard priorities
  ▼Standard declarations
    ▶colset UNIT
    ▶colset BOOL
    ▼colset INT = int;
    ▼var score: INT;
    ▶colset INTINF
    ▶colset TIME
    ▶colset REAL
    ▼colset STRING = string;
    ▼var name, grade, ID: STRING;
    ▼colset INPUT = product STRING*INT*STRING*STRING;

```

Figure 7. Variable generated in CPN

Thirdly, after the final resulting CPN is generated, we use CPN tools to verify the correctness of our final result. By using CPN tools, we can simulate the system step-by-step to ensure the system that going to be developed is working correctly. CPN tools also allow us to use the simulation to find the unreachable path or the deadlock. So, if there is any incorreced, we can fix it before going to the development process.

## 5. CONCLUSION

In this paper, we intend to demonstrate how to automatically verify the behaviors of a given flowchart using formal method. A scheme of transforming a basic flowchart into coloured Petri net is proposed. A set of mapping rules is illustrated in Table 1, to cope with the transformation of basic flowchart elements into CPN elements. We provide the guides to label the essential inscriptions on places, transitions, and arcs, in order to make the resulting CPN complete and ready to be simulated by CPN tools.

Without the details of the CPN inscription, the CPN is just called CPN skeleton. We demonstrate a case study and verify the correct behaviors of the resulting CPN model using CPN tool.

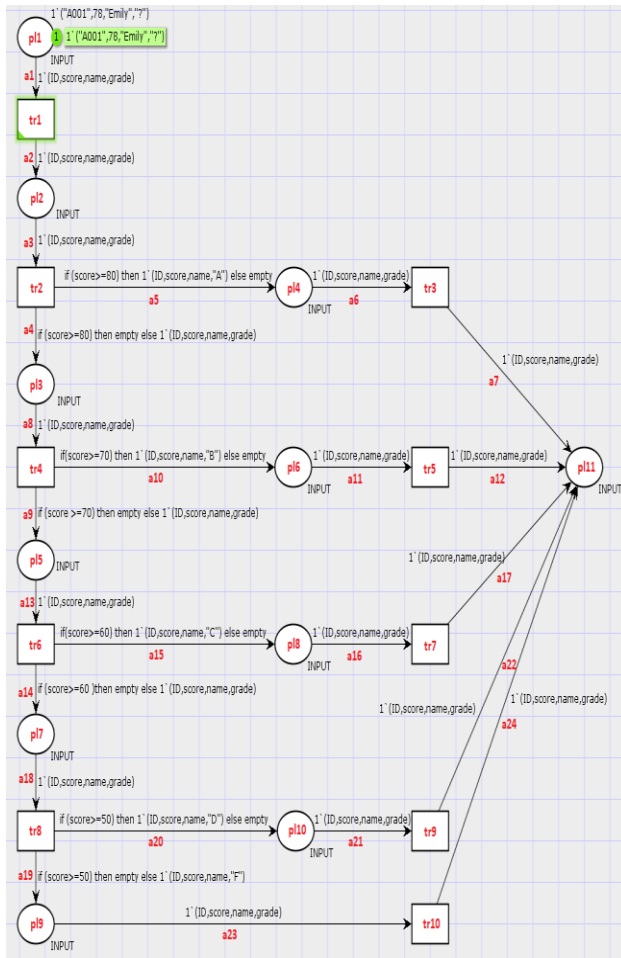


Figure 8. A grading process in CPN

## 6. REFERENCES

- [1] ISO/IEC 2382:2015, Information technology -- Vocabulary <https://www.iso.org/> , Last visited: 17th June 2019
- [2] K. Jensen and L. M. Kristensen, Coloured Petri Nets: Modelling and Validation of Concurrent Systems, 1st ed., Springer Publishing Company, Incorporated., 2009.
- [3] N. Maneerat, W. Vatanawood, Translation UML Activity Diagram into Colored Petri Net with Inscription, in 2016 International Joint Conference on Computer Science and Software Engineering (JCSSE)
- [4] S. Meghzili, A. Chaoui, M. Strecker, E. Kerkouche, On the Verification of UML State Machine Diagrams to Colored Petri Nets Transformation Using Isabelle\_HOL, in 2017 IEEE International Conference on Information Reuse and Integration
- [5] James L. Peterson. Petri net theory and the modeling of systems. Prentice-Hall, Inc., Englewood Cliffs N.J., 1981.
- [6] JI Rocha, L Gomes, OP Dias, Dataflow Model Property Verification Using Petri net Translation Techniques, in 2011 IEEE International Conference on Industrial Informatics, 783-788
- [7] J. Deesukying, W. Vatanawood, Generating of Business Rules for Coloured Petri Nets, in 2016 ICIS
- [8] Harley R. Myler, Fundamentals of Engineering Programming with C and Fortran, Cambridge University Press, 1998
- [9] Ned Chapin, Flowcharting With the ANSI Standard: A Tutorial Computing Surveys, InfoSci Inc., Menlo Park, California, Vol 2, No. 2, June 1970
- [10] M. CPN, An extension of standard ML, ed.
- [11] CPN Tool Official site <http://cpntools.org/> , Last visited: 17th June 2019.