Thesis Progress Report 4

Aytar Akdemir
150170115

December 16, 2021

1 Naessens' Formalization Methodology

Formalization is a collection of specifications that will define a system. While implementing a system, the rules of formalization will be used. Defining the formal specifications, comes after the requirements analysis and before the implementation. For the most abstract formalism, A Flow Graph is utilised.

In [Cite naessens], Naessens divides conceptual design to three phases:

- Order constraints and multiplicity constraints
- Authentication and authorization constraints
- Control measures

Flow chart are used for order constraints and Petri nets are used for the authorization constraints. A Linkability Graph is used for modelling control measures.

In the article, they transform the Petri Net into a Flow Graph and the Linkability Graph into a Petri Net.

1.1 Order and Multiplicity Constraints

Order and Multiplicity Constraints are basically the rules on the order of the actions that can be performed and the number of times those actions can be performed. For example a user cannot remove a file before creating that file first. As an example of the multiplicity constraint, file creation can be performed multiple times denoted by "*"; file deletion can only be performed once on a file, which is denoted by "1".

An example of a flow graph is given below.

1.2 Petri Net Representation

In Petri Net, each right is represented with a token. Tokens are stored in Places and each Action has Places before and after it. If there is at least one token in the Places that

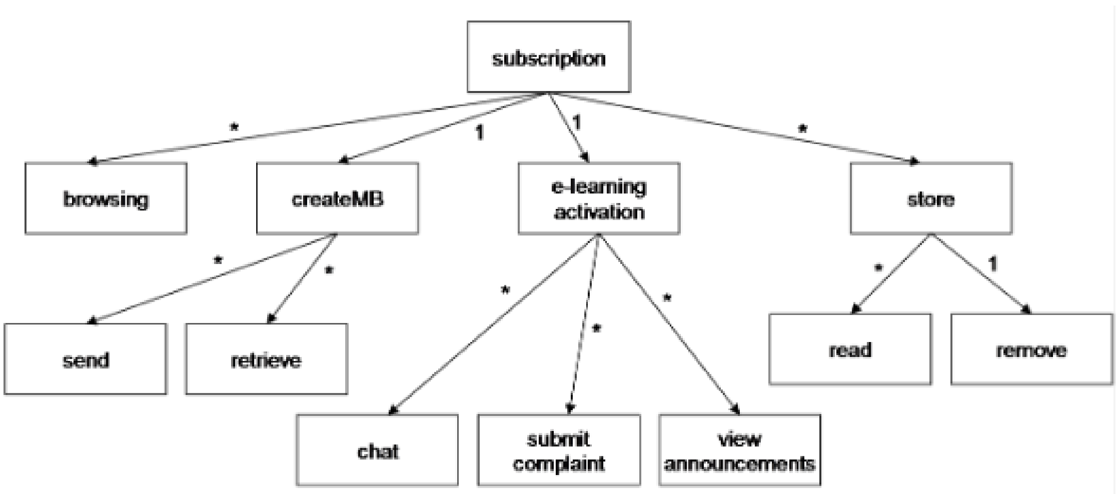
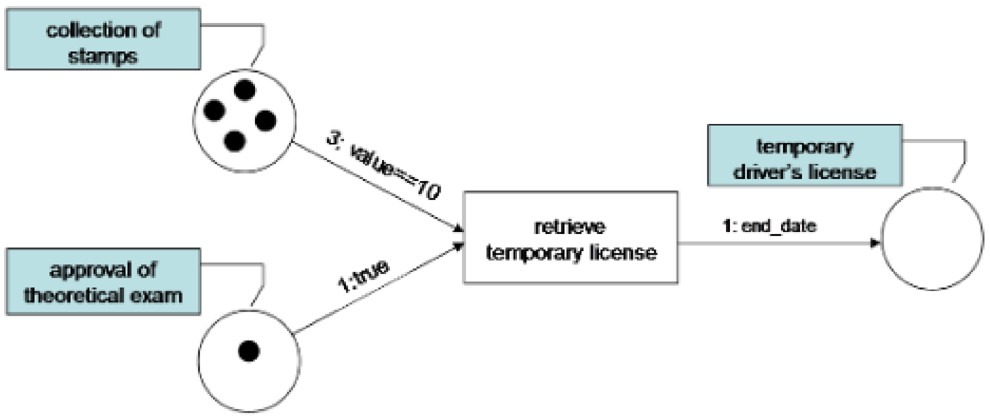


Figure 1: Order and Multiplicity Constraints

point to the Action, Action can be performed and the next Place is marked with a token. Inputs to an Action might have a natural number and a condition property:

- Condition can further constrain if a token is accepted by the Action.
- The natural number is the minimum number of tokens accepted.

The output has a natural number, the specified amount of rights will be placed to the next Place. Initializers are also assigned to outputs, which initializes the attributes of the rights. The condition on the output is the additional constraints that are not tied to the inputs.


collection of stamps
3: value==10
temporary driver's license
retrieve temporary license
1: end_date
approval of theoretical exam
1:true
Figure 2: Petri nets

If the place after the action doesn't need to have a token, i.e. just prove it has the tokens to obtain a right, the token can be returned to the place before the action. After the transition, the state is unchanged even though the action has been performed.

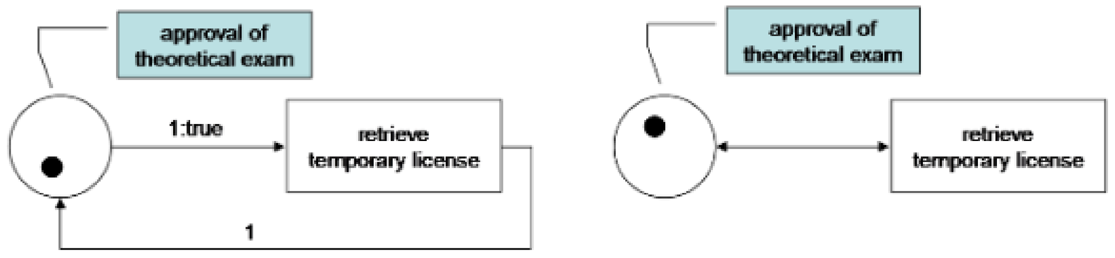

approval of theoretical exam
1:true
retrieve temporary license
1
approval of theoretical exam
retrieve temporary license

Figure 3: Unchanged place

The tokens attributes might change after the action. After the action, a mutator changes the value of the token. It is possible to have more than one transition for different conditions.

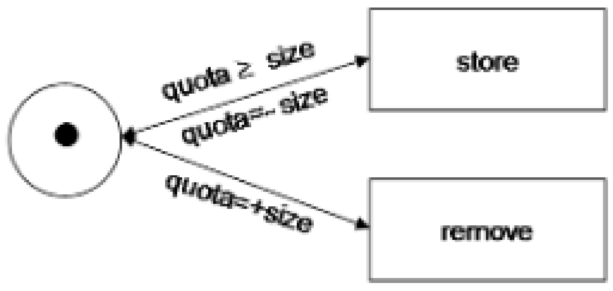


Figure 4: Conditions for different states

1.3 Applying Petri Net to Flow Chart

Multiplicity of an edge has two cases to be simulated.

- multiplicity(e)=n. Multiplicity is limited. Action1 initializes n rights. n tokens will be placed at the Place1. Every time the next action, Action2, is performed, a token is submitted to the next place, Place2, and subtracted from the Place1.
- multiplicity(e)=*. Multiplicity is unlimited. Action1 initializes 1 right. 1 token will be at the Place1. A double sided arc is used between Place1 and the Action2. Therefore, token at the Place1 will be put back in place everytime Action2 is performed (as in Figure 4).

1.4 Linkability Graphs

Linkability graphs show the links between the data. In a case of attack, linkability case can be used to trace the criminal content back to the attacker.

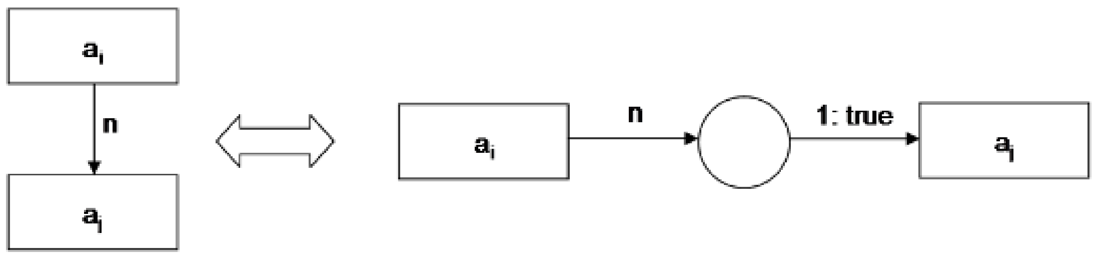


Figure 5: Limited multiplicity

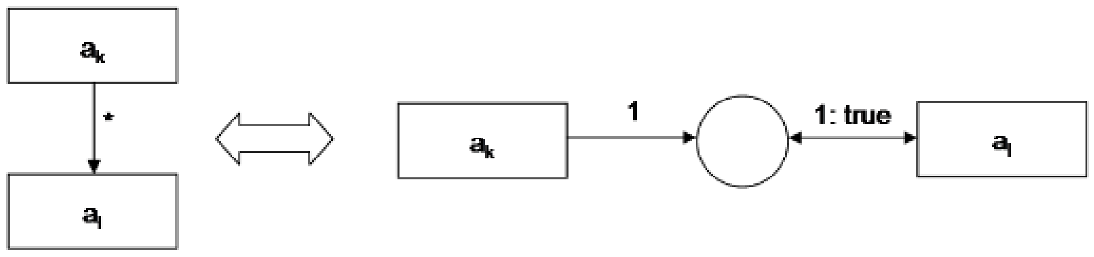


Figure 6: Unlimited multiplicity

In a linkability graph, a node might represent:

- Actions
- Rights
- Environmental variables

Each edge represents a connection between two units. Linkability graohs are undirected. An edge l has three properties:

- cardinalityRatio(l): Maximum number of unit instances this unit can be linked directly. 1:1 or 1:N
- linkCond(l): Conditions that must be fulfilled to reveal the link.
- linkBy(l): Returns the set of subjects that are required to reveal the link.

We have three queries we can use and combine in order to reach to the information we want.

- LinkabilityGraph(Graph, SetofSubjects): Returns the nodes that can be linked by a set of subjects. Returns a subgraph.
- LinkabilityGraph(Graph, SetofConditions): Returns the nodes that can be linked when the set of conditions is fulfilled. Returns a subgraph.
- existsLink(Graph, [Action1, Action2]): Returns whether two actions are linkable.