Course on Domain-Specific (Modelling) Languages

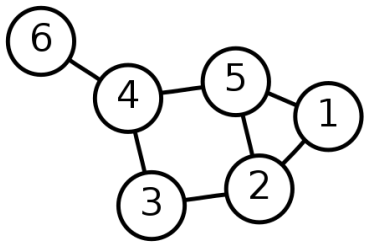
2022/2023
LAB 2 - Exercise - Metamodelling

Vasco Amaral
September 2022

1. A Simple Undirected Labelled Graph Case Study

In graph theory, a graph is a set of related objects. The objects are called vertices and each related pair of vertices is called an edge. We can define graph as an ordered pair G = (V, E) where V is the set of vertices and E id s set of sets of two vertices.

We can draw a diagram as a schematic (visual) representation of a graph. For instance, consider the following Graph G={G,V} where the vertices are V={1,2,3,4,5,6} and the edges are E={{1,2},{1,5},{2,3},{2,5},{3,4},{4,5},{4,6}}. We could represent that example as:



Build the ecore that could be the metamodel of this language.

2. Turing Machines graphical language

The Turing machine was invented in 1936 by Alan Turing and is a mathematical model of computation of an abstract machine that operates on an unlimited tape (extending infinitely in one direction) according to programmed control logic. It can be seen as a finite-state machine in which a transition prints a symbol on the tape. The head of the tape may move to either left or right, allowing the machine to manipulate by reading and writing at will.

It can be defined as a 6-tuple **M= { Q,** $\sum$**,** Γ**, δ, q0, F }**, where:

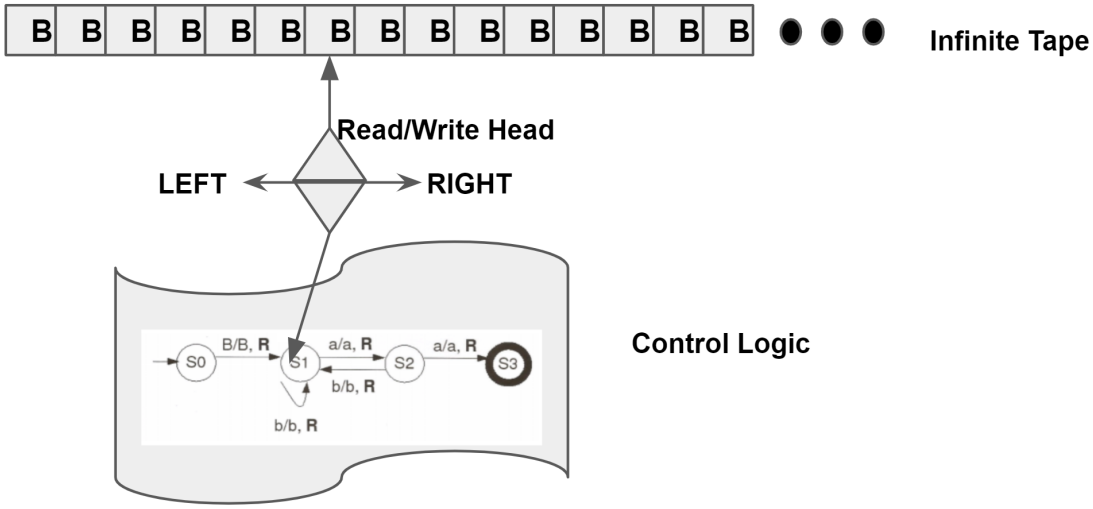
- **Q** is a finite set of states,
- Γ is a finite set called the tape alphabet with a special symbol **B** that represents a blank cell in the tape,
- $\sum$ is a subset of Γ except B,
- **δ** is the transition function (a partial function from Q x Γ to Q x Γ x {L,R}),
- **q0** ∈ Q is the initial/start state,
- **F**⊆ Q is a set of final states.

The transition function of a standard Turing machine with the input alphabet Γ={a,b} U {B} and the states Q = { s0,s1,s2,s3 }, where q0 = s0 and F = { s3 } with the program to accept strings with the following regular expression **(a+b)*aa** can be described as follows:

	δ	B	a	b
>	s0	S1, B, R		
	s1		S1, a, R	S2, a, R
	s2		S3, b, R	S1, b, R
*	s3			

(* identifies a final state, while > tags the start state)

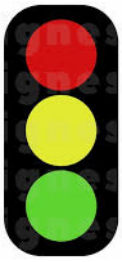
Consider the following visual notation to describe Turing machines



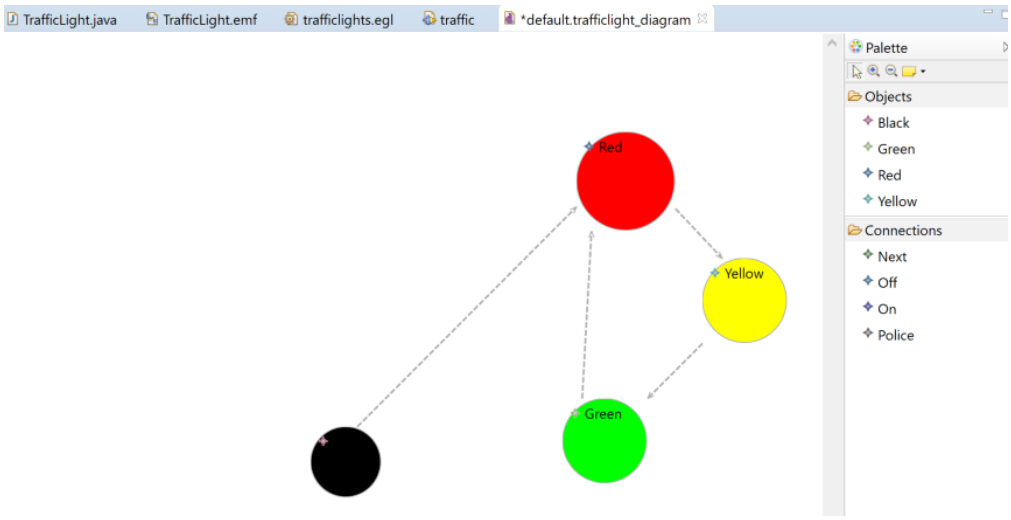
The transition function can be graphically represented by a state diagram:

Build the ecore that could be the metamodel of this language.

3. The Traffic Lights Control language



Traffic lights are a common device in our daily life. We pass by them, unaware that their behaviour varies more widely than we might expect. Different countries have different rules, equipment and standards (have a glimpse at this variety, for instance, in https://en.wikipedia.org/wiki/Traffic_light). We will next design a naive language for describing the behaviour of a traffic light like the one in the figure above.



Build the ecore that could be the metamodel of this language.

4. The Planning Tasks Language - The Model-Driven way

1- Design a language to plan and organise tasks/activities (with duration) based on dependencies between them and the notion of Human and budget resources allocated for each of them.

2- Build a more refined language like the Activity-on-Nodes (AON) based language, with the purpose of refining models of language 1, that furthers the details with concepts of an early start, early finish, late start and late finish. The purpose of the second language is to have a model that contains and makes it easier to calculate the mentioned values, including the concept of a critical path. This language has nodes like the one below, connected by edges to indicate the order (dependencies) between tasks that should be scheduled in a project plan.

Early Start	Duration	Early Finish
Task Name		
Late Start	Float	Late Finish

Acronyms List:

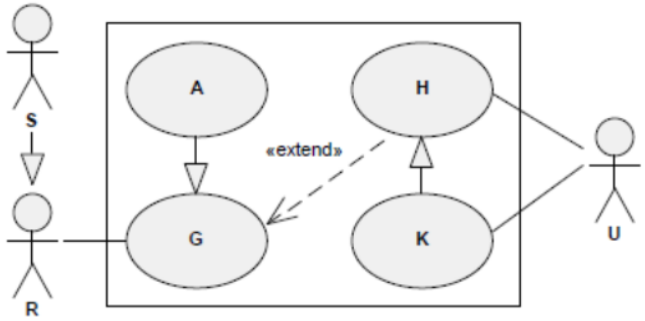
- AON Activity On Node
- EST Earliest Start Time
- EFT Earliest Finish Time
- LST Latest Start Time

- LFT Latest Finish Time
- ETL Epsilon Transformation Language
- EVL Epsilon Validation Language
- EGL Epsilon Generation Language

5. The Use Case Diagrams

[This was a question of the exam in 2019] Build the ecore that could be the metamodel of this language.

Propose a possible metamodel for Use Case Diagrams (as learnt in the "MDS" course). For that, you should use Ecore as your metamodelling language. Please find below an incomplete example of such diagrams.



Please take into account that:

- there is only a system box per Use case containing the use cases;
- there is the <<include>> relationship between use cases;
- both use cases and actors can be <<abstract>>