Nemo's graph grammar - The beggining of the end of Network's fireman approach

Andrés Laurito

Departamento de Computacón - FCEN -UBA
andy.laurito@gmail.com

Abstract. Something great
"The best way to predict the future is to invent it - Alan Kay"

1 Introduction

In this paper we intend to create a computational model for programming languages running over the Software Defined Network (SDN) paradigm. For this purpose, we will define a graph grammar semantic, with it's own productions. This productions will represent actions that take place in our network, and we will see how this actions creates new behaviour in the model network.
The network will be model as a graph, and it's representation will be obtained by a YAML file, which will have to be delivered by the user of the programming language. We will explain in further details how we manage to create a graph from a YAML file in the following section.
Finally, we will explain how this graph grammar can be used to model NEMO, a network oriented progamming language.

2 Graph grammar production's

We start by defining the most basic grammar production's. These productions are the one's that allows us to modify the network topoloy:

– Node creation

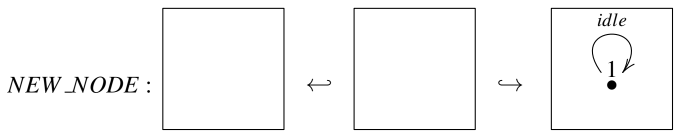


Fig. 1. Node creation

This production creates a new logical node (this means that the node could be either physic (which mean's it's a physical host with an assocciated address) or logic (this

mean’s a group of physical nodes, meanning that the address could be a CIDR).
It is important to remark that the created node is in an idle state, meaning that is not interacting with any other node in the network.

– Service Creation

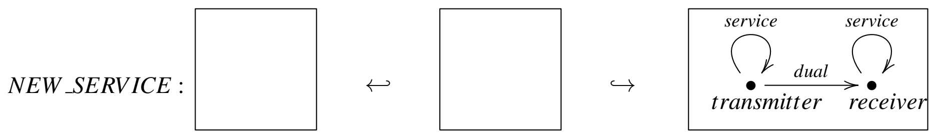


Fig. 2. Service Creation

In this production we define a new service. A service is either some property implemented in nodes, or implemented in another node, wich will be used for generating flows,this means that, when a flow is created, this flow will be of type service. The fact that the main idea is to send and receive information, is represented as dual nodes representing a transmitter and a reicever in the graph.

– Node behaves as some endpiont of communication

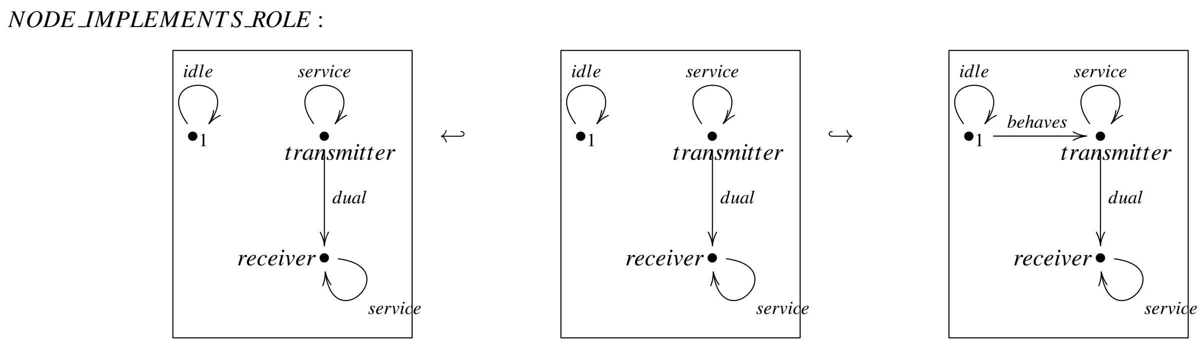


Fig. 3. Node implements role

This production is used to make node’s to behave as one of the endpoints in the communication of services. The node can only behave as one of the endpoint in one service, and it’s allow to implement more than one service. THIS RESTRICTION SHOULD BE DONE BY NEGATIVE APPLICATION

– Link creation

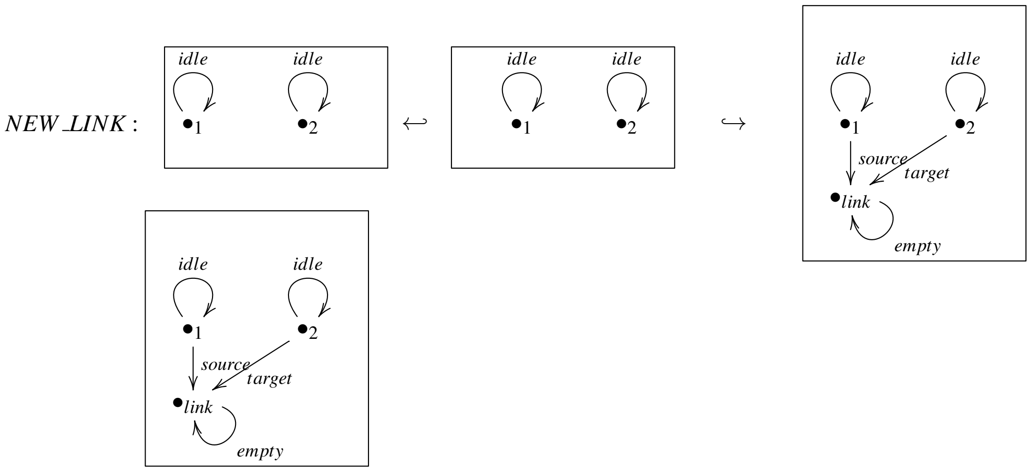


Fig. 4. Link creation

Here we define a new link between nodes 1 and 2. The only condition to apply this production is that both nodes must be in an idle state and neither of them should have an existing link created. This restriction is done by the negative condition showed in L.
Note: Being in an idle state does not mean that they are doing nothing!. An idle state behaves likes a wildcard, meaning that from this state, multiply actions can be applied to the node.

– Link’s property creation

NEW_LINK_PROPERTY :

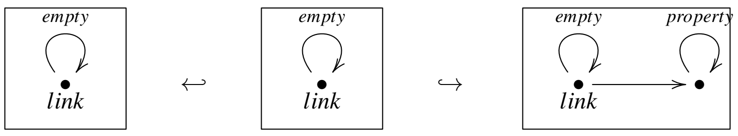


Fig. 5. Link’s property creation

In this production we are allowing definition of link’s property. This properties are going to be perhaps physical properties in physical links (for example, we can de-

fine here bandwith, average latency, perhpahs if it's either ethernet or wifi), and some other properties in logical nodes.
QUESTIONS:
1) What properties could be assigned to a logical link?
2) Can I always define properties in a node? (It really doesn't matter what is going on with that link in the network?. Perphas it happens something similar as nodes, that we need some 'idle' state, for example, an empty state).

So far we have defined productions that which are intended to be use between the interactions with the elements in the network. With the following productions, we will start using the productions listed before, in order to crate interaction between the elements in the network.

– Flow creation

NEW_FLOW :

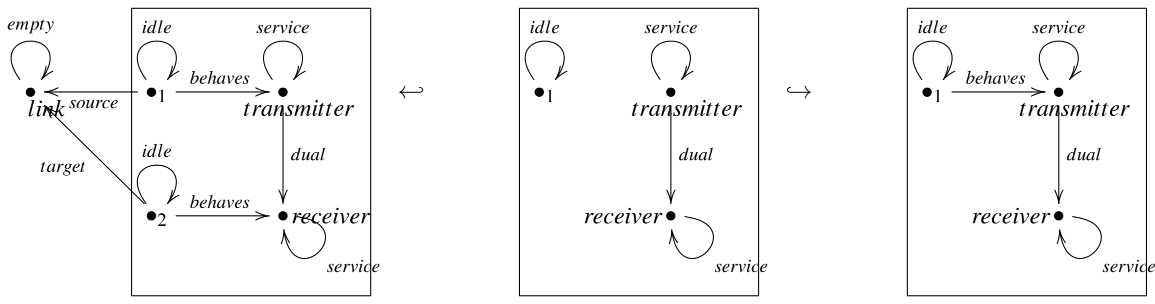


Fig. 6. Flow creation

In order to create a flow, I first need two things:
- I need to have a link between the nodes
- I need to have in both nodes the service which will be the content type of the flow.

Lets go through these rules one more time. The first rule is just telling us that, in order to be able to create a node, I need a connection between this node's (remember that this conection can be either logical or physical).
The second rule is telling us that, in order to create a flow of type X (lets say for example, that I want to create a ssh flow between this nodes), I need to guarantee that both nodes support ssh connections. This is represented as both nodes having a link to the service.
Finally, it's important to notice that a flow is associated betwen nodes and a link (this last one is going to be the place where it's flowing).

THINS THAT HAD TO BE THINK A BIT MORE:
1) Some rules may not be valid in this context ... How do I deny multiple ssh conections betwen same nodes? (Also, this sounds to me like a type condition, since another services may allow this behaviour, for example sql.)
2)

– Flow's properties creation

NEW_FLOW :

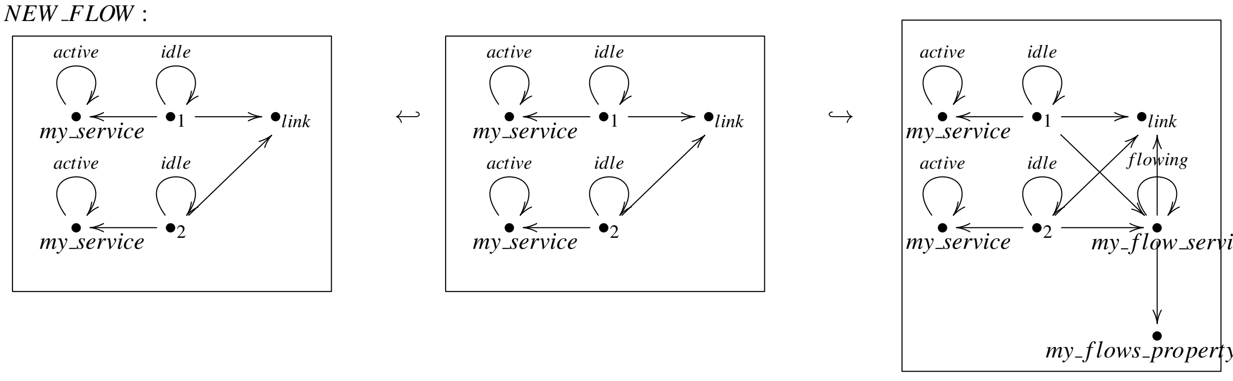


Fig. 7. Flow's property creation

This production allow's the creation of flow's properties.

QUESTIONS:
1) Can this properties be the actions associated in the nemo language ? Perphaps properties can be understood as premises that have to be guaranteed while the flow exist.
2) Some of this properties could be dynamic (this has sense, since flow is the representation of dynamism in the network).

3 Graph instantiation

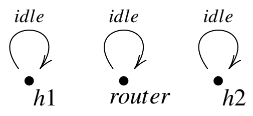
In this section we will focus in giving several examples of the usage of the grammar recently defined. The examples used in this section will be those defined in a previous document (REMEMBER TO REWRITE EXAMPLES HERE)
Lets first defined the first example, a network with two host and a router, which want to start a ssh connection.
There were two problems here:

– Neither of the host had ssh active

– Neither of the host were connected to the router.

Let's model our network, and see how the grammar defined before help us to detect problems in what we are trying to do.
This would be the network represented in our model:

FIRSTEXAMPLE



Having model our network, what we can check is that NEW_FLOW production can't be applied to my network (since there is no morphism between the left side of the production and this model). In this case, if I write a program for the situation described, we will have a compiler error, since there's going to be an instruction that cannot be done, and this is because the production associated with it cannot be executed.