INCF Task Force for a Standard Language in Neural Network Models

nineML (9ML) version 0.9 User Layer Summary

Proposal v.2 ID:AG-20100708.01

Maintained by Anatoli Gorchetchnikov

July 8, 2010

Abstract

This draft intends to describe our current opinions on User Layer concepts of NineML.

This work currently summarizes the different ideas and propositions elaborated during five Task Force meetings and User Layer group discussions. It might contain some misrepresentations or omissions of important issues and shall be updated during future work.

Changes from version 1:

- added footnotes for some issues raised in Stockholm June meeting
- added brief description of interactions between user layer and simulation software that does not support abstraction layer
- removed random number generator nodes as decided in Stockholm
- added space, region, and layout nodes as a first draft of geometrical concepts (based on NetworkML proposal by Padraig and Robert as well as Stockholm discussions)
- removed recurrent projections that can be described on the population level
- cut out the previous approach to cell positioning within population

1 Introduction

The user layer of NineML is intended to be primarily machine-readable and uses XML syntax. It is designed with a focus on ease of parsing, verification, and automatic model construction. This does not prevent advanced users from editing the user layer XML descriptions by hand, but the primary means for creation of these descriptions is expected to be software tools that will convert GUI- or script-based representations of objects and properties into valid XML.

The user layer provides the syntax for specifying the model and parameters to be used to instantiate the key elements of a spiking neuron network. This includes descriptions of individual elements (cells, synapses, inputs) and theconstructs for describing the grouping of these entities into networks. In addition the user layer defines the syntax for specifying a range of connectivity patterns.

The remainder of this document summarizes the concepts that underlie user layer design. For specific examples of implementing these concepts please refer to a General NineML Description.

2 Nodes and Properties

The basic building block of NineML is called node[1]. In the user layer description node is a reference to an object defined in the core semantics (abstraction layer). Abstraction layer defines the mathematics of

[1] Alternative terms: component, instantiation, entity

the node and is then referred in the user layer. The initial version of the standard allows to put references to external (including user space abstraction layer) definitions.

Each of the nodes has a user-given ID or name and a set of properties that is defined in the core semantics (or externally) and instantiated in the user layer. Since these properties are unique for every definition, the tags that are used for them are predefined by these definitions[2]. The core semantics specification or the external definition documentation shall provide the mapping between mathematical description of the object and the corresponding tags.

User can add short notes to each node description. These notes are intended to provide a specific reference to the research paper page where the node is described and similar kind of information. These notes are not intended to duplicate the core semantics documentation or any other documentation, thus they shall not provide mathematical description and other details of the node implementation. Note can contain text or link to an Internet resource.

As a result, a complete description of a node in the user layer consists of three parts: reference to a definition, a list of properties, and an optional note.

To reduce the size of the resulting network description user can refer to already described node by user-given name instead of providing link to abstraction layer or external definition. In this case the properties of the node that have to be redefined are stated explicitly, the properties that are inherited from the original description are omitted.

If the simulator only supports user layer, then the simulator developers can create mappings directly between the reference to a definition in the user layer description of the node and the intrinsic simulator code that implements the same mathematics.

2.1 External Definitions and Extensibility

The language should be flexible enough to allow representation of concepts that do not yet exist, as it is developed to serve the forefront of research. A simple mechanism to add concepts that are not part of the standard is provided through external (other than abstraction layer based) definitions. It is the choice of a simulator developer to support these definitions during initial stsage of NineML development. It is unclear at this point who will eventually convert these definitions into native NineML definitions, but the maturation of NineML shall eliminate the need to support simulator-specific definitions.

2.2 Property Description

NineML uses predefined tags for properties. These tags are provided by the core semantics (abstraction layer) definition of the parent node. The user can set the value of the property as well as (where applicable) the units of measurements. There is no need to provide a user-given ID to a property because it can always be uniquely identified by the parent node ID and its own tag name. User can add a note to a property in a way similar to adding notes to nodes described above.

There are two kinds of properties: values of the first kind are given to the model by the user and stay fixed, values of a second kind are computed within the model during simulation. For all practical reasons the syntax of the user layer descriptions is identical for both kinds of properties. Furthermore, because NineML does not provide any default values for properties, it is a job of the user to provide initial values for all defined properties. To ensure the integrity of the model NineML requires all initial values to be set in the user layer description. For batch simulations and other modifications any of the values given in the user layer can be overwritten by a simulation setup description, but this is outside of the scope of the current version of NineML.

Some properties can have values drawn from random distribution. In this case user layer description of the property includes a reference to a random distribution node (section 2.3). Other properties might be calculated according to some function dependent on the geometry of the system or some other properties. These can be defined by including inline abstraction layer definitions or MathML. Note that properties that depend on geometry will remain undefined until the network is created.

Issue: Syntax shall be defined to provide compound descriptions of both values and units, for example when relative conductance is defined in mS/cm^2.

[2]Issue raised during June Stockholm meeting whether this shall be the case

2.3 Random Distribution Nodes

This set of nodes allows to define random distributions with corresponding parameters. In includes the properties of the distribution. These nodes allow the reuse of the same distribution multiple times similar to reuse of all other nodes.

2.4 Neuronal Nodes

The description of the neuronal node defines a prototypical neuron which can be reused multiple times within the network. As a consequence, multiple connectivity patterns can be applied to this neuron, which results in a different set of synaptic inputs. Therefore, no description of these inputs shall be provided at the node level, they shall be described at the projection level. Simulation software shall take care of complete construction of each neuron by analyzing both levels of description.

2.5 Plasticity Nodes

Plasticity nodes handle the synaptic weight and its possible modification. Note that synaptic weight is a dimensionless quantity and as such does not include units. This is done to allow same plasticity rules to operate on different types of postsynaptic responses. Synaptic weight defined in the plasticity node determines only the magnitude of the response, the shape and physical properties are defined in the corresponding post-synaptic response node (section 2.6).

2.6 Post-synaptic Response (PSR) Nodes

Post-synaptic response nodes define the effect imposed on the post-synaptic cell dynamics by triggering the synaptic input. This definition includes only the shape of this effect and does not include the exact magnitude of the effect. The magnitude is described separately through plasticity rules and synaptic weights in section 2.5. Note though that the units of the effect are defined here and synaptic weights in plasticity nodes are left dimensionless.

2.7 Connectivity Nodes

Connectivity nodes contain the definition of a connectivity pattern between neurons. Connectivity nodes based on the geometry of source and/or target groups include descriptions of the edge handling. All connectivity nodes have a delay property that sets the axonal delay for the corresponding set of projections.

2.8 Space Nodes

Space nodes define the coordinate systems used to set up the geometry of the network. These nodes can be Cartesian, polar, and maybe some other coordinate systems. More than one space nodes can be defined in the same network, for example Cartesian 3D for cortical cells and polar 2D for retinal space.

Issue: In case we need rotated, translated, or scaled versions of the same coordinate system, shall we define a new space node?

Issue: In case user combines two models how shall the space be handled?

2.9 Region Nodes

Region nodes define a finite region given a coodrinate system. A reference to a corresponding coordinate system as well as boundaries of a region constitute the properties of a region node.

2.10 Layout Nodes

Layout nodes define how neurons or groups of neurons are positioned within a given region of space. Note that both discrede grids and continuous mappings can operate on the same region, so it is possible to create a 3D grid of neurons for one population as well randomly distributed set of neurons from other population within the same region.

3 Grouping

The grouping of objects is of two major types: **constructive** grouping used to **create** objects and **access** grouping used to define the subgroups of **already created** objects for the purposes of connectivity or monitoring. To eliminate access groups that fully replicate corresponding constructive groups any constructive group can be used for the access purposes. Both types of grouping allow hierarchical build of larger groups from smaller groups. Rather than having a deep hierarchy, however, the structure is kept flat by referencing. Each group contains a user-given ID or name. Groups do not contain the description of the elements that are repeated, but only refers to their original description by these IDs. The scoping rules for these names are described in section 5.1.

The rules of construction provided by constructive groups shall not be executed immediately on occurrence of such a group within the description. The process of construction shall proceed from top-most group in the hierarchy down to component groups. This allows the flexibility of defining one constructive group in the description and reusing it multiple times within the network. This also allows the inclusion of developed networks in the larger models as sub-components. The top-most group in the hierarchy is not marked anywhere in the description, it is rather selected by the user through the software interface. This way the user can simulate individual components of the network for debugging and tuning purposes without modifying the rest of the model.

3.1 Constructive Group

This is a two-layered structure including of a set of populations on the lower level and connectivity between these populations together with access groups (section 3.2) on the upper level. Each population describes a collection of *identical* elements. To allow the arbitrary complexity of the model, these elements can be other constructive groups. Both the number of populations and the number of elements within population can be set to one. Processing of a constructive group by a NineML-compliant software shall lead to creation of instances for all elements, define the spatial structure of the group, and provide internal connectivity if necessary.

Populations of the group refer to a prototypical node or constructive group by a user given ID. Different populations can use the same prototype if the user wants to define two populations of same neurons or circuits but with different connectivity or with different positional distribution.

A collection of projection entries within group can describe the internal connectivity within this group according to the rules shown in section 4. In case this connectivity depends on some features described outside of the group it has to be defined in the point of the hierarchy from where both the feature and question and this group are directly accesible by descending the tree.

3.2 Access Group

Access group does not define how the cells or subnetworks are created, but rather how they shall be selected for the purpose of monitoring of the the activity or building the connectivity. Any constructive group declaration also naturally defines an access group of all the elements it contains plus one access group per each of the populations. Constructive group user-given IDs as well as population user-given IDs can be used anywhere access group IDs can, without explicitly defining an access group for each constructive group or its populations. Declaring an access group does not add any objects or nodes to a model specification: it just forms a new, possibly empty, set from those nodes that have already been declared.

3.3 Basics of the Structure, Layouts, and Coordinates

Declaring the existence of a population in a constructive group implicitly introduces a notion of indexing for its elements, so, for example selection rules may use conditions on the indexes to select subsets of cells. Population contains a reference to a layout node to map indices onto coordinates, which can also be used in selection rules for access groups. In general, a layout (see section 2.10) is simply a mapping from indices to generalized positions.

3.4 Selection for Access Group

3.4.1 Selection by a User-given ID

Matching criteria for these selections are user-given IDs that can be found while descending the tree within the parent group of the access group description. When all matches are exhausted, the union of each such binding is constructed and returned.

3.4.2 Selection by a Cell Property

This selection only makes sense for cells that do have certain property defined in their parameter list. If later the model will be changed and a different neuronal node will be used as a prototype, this select statement will become invalid. The user shall be notified of this inconsistency by a simulation software and resolve the problem manually. Furthermore, if the property in question has units of measurements attached to it, the selection shall also include units. Finally, some properties as defined in section 2.2 are variable over the course of each simulation, and if these properties are allowed in for selection that means that the access group will be different on every simulation step.

> **Issue:** Given all these issues it is unclear whether NineML shall allow this kind of selections at all.

3.4.3 Selection by Geometry

This type of selection takes valid positions as arguments. The compliant software shall ensure that both positions are described within the same coordinate system, an error message shall be raised if this is not the case. For topographic projections the position of the source within its group (population) partially determines the position of the target within a different group (population). In this case we might need more complicated coordinate mapping then a simple reduction to a uniform coordinate space (e.g. cells with certain 2D retinal position project to cells with certain 3D coordinates in the cortical sheet).

3.4.4 Selection by Logical Combinations of Criteria

Within each select statement logical operations are possible. For completeness all three (AND, OR, NOT) are allowed.

4 Projections

A projection holds a description of the connectivity between two sets of cells. Along with a user-given ID or name unique inside the hosting group, each projection description contains source and target elements, each of which can reference either individual cell or an access group. In the latter case each projection description leads to creation of a set of projections rather than a single projection.

Projection description also includes references to a plasticity node that controls the synaptic weight (section 2.5) and post-synaptic response node that controls the influence of the input through this projection on the post-synaptic cell dynamics (section 2.6). Both appear in the description of the projection rather than neuron because the same type of neuron can use different synapses when instantiated in different populations, similarly the same postsynaptic response can be used in multiple projections with different plasticity rules. Instead of providing the user with all possible pre-wired combinations, NineML allows user to combine the plasticity→response→neuron chain from a set of small standard components.

Finally, a connectivity pattern or rule shall be set for a projection through reference to a connectivity node.

5 Structure of a User Layer Description

In order to simplify the descriptions themselves and the mechanisms for combining multiple components of the model the user layer description is consisting of multiple files that define various components and contains the syntax to import external files.

5.1 Scope of User Layer Descriptions

In order to avoid the collisions of user-given IDs as well as to provide the user with the ability to redefine some of the imported components the following set of rules is suggested:

- Every identifier is considered unique only within its parent
- To access the identifier a full path from the point of access to this identifier shall be provided.
- Any imported construct can be redefined within the importing description by adding a new definition of the same object. All properties of the imported object are imported from external file and the values of some (or even all) of them are overwritten by the local description.
- If multiple redefinitions of the same entity are found, then the most recent one will be used. The most recent is determined by the order of parsing of the top level model description file. Here the rule shall be enforced that when multiple files are included all nodes are parsed before all groups. This will allow a smooth transition from networks with simple nodes and groups to networks where some nodes and some groups are redefined.
- If several different subfields are redefined in different places of the description, then all of these new definitions will be used. Please note that this is not the optimal way, and the better way to achieve the same effect is to provide all redefinitions in a unique location. Simulation software developers can (but at this point of time are not obliged to) optimize the NineML code that is output by their software by grouping all redefinitions of components of the same node into a single node.