NetLogo NW Extension (Beta 1.0.0) — Cheat Sheet

For download and complete documentation, see:
`https://github.com/NetLogo/NW-Extension`

General Primitives

`nw:set-context` *`turtleset linkset`*

Specifies the set of turtles and the set of links that the extension will consider to be the current graph. All the turtles from *`turtleset`* and all the links from *`linkset`* that connect two turtles from turtleset will be included.

`nw:get-context`

Reports the content of the current graph context as a list containing two sublists: the list of turtles that are part of the context and the list of links that are part of the context.

Centrality Primitives

`nw:betweenness-centrality, nw:eigenvector-centrality, nw:closeness-centrality`

These primitives calculate different centrality measures for a turtle. Example:

```
ask turtles [ set size nw:betweenness-centrality ]
```

Distance and Path-Finding Primitives

`nw:distance-to` *`target-turtle`*
`nw:weighted-distance-to` *`target-turtle weight-variable-name`*

Finds the shortest path to the target turtle and reports the total distance for this path, or false if no path exists in the current snapshot. The `nw:distance-to` version of the primitive assumes that each link counts for a distance of one. The `nw:weighted-distance-to` version accepts a *`weight-variable-name`* parameter, which must be a string naming the link variable to use as the weight of each link in distance calculations. The weights cannot be negative numbers.
Example:

```
ask turtle 0 [ show nw:distance-to turtle 2 ]
ask turtle 0 [ show nw:weighted-distance-to turtle 2 "weight" ]
```

`nw:path-to` *`target-turtle`*
`nw:turtles-on-path-to` *`target-turtle`*
`nw:weighted-path-to` *`target-turtle weight-variable-name`*
`nw:turtles-on-weighted-path-to` *`target-turtle weight-variable-name`*

Finds the shortest path to the target turtle and reports the actual path between the source and the target turtle. The `nw:path-to` and `nw:weighted-path-to` variants will report the list of links that constitute the path, while the `nw:turtles-on-path-to` and `nw:turtles-on-weighted-path-to` variants will report the list of turtles along the path, including the source and destination turtles. As with the link distance primitives, the `nw:weighted-path-to` and `nw:turtles-on-weighted-path-to` accept a *`weight-variable-name`* parameter, which must be a string naming the link variable to use as the weight of each link in distance calculations. The weights cannot be negative numbers. If no path exist between the source and the target turtles, all primitives will report an empty list. Examples:

```
ask turtle 0 [ show nw:path-to turtle 2 ]
ask turtle 0 [ show nw:turtles-on-path-to turtle 2 ]
ask turtle 0 [ show nw:weighted-path-to turtle 2 "weight" ]
ask turtle 0 [ show nw:turtles-on-weighted-path-to turtle 2 "weight" ]
```

nw:turtles-in-radius *radius*
nw:turtles-in-out-radius *radius*
nw:turtles-in-in-radius *radius*

Returns the set of turtles within the given distance (number of links followed) of the calling turtle in the current snapshot. The `turtles-in-radius` form works with undirected links. The other two forms work with directed links; out or in specifies whether links are followed in the normal direction (`out`), or in reverse (`in`). Example:

```
ask turtle 0 [ show sort nw:turtles-in-radius 1 ]
```

nw:mean-path-length
nw:mean-weighted-path-length *weight-variable-name*

Reports the average shortest-path length between all distinct pairs of nodes in the current snapshot. If the `nw:mean-weighted-path-length` is used, the distances will be calculated using *`weight-variable-name`*. The weights cannot be negative numbers. Reports false unless paths exist between all pairs. Examples:

```
show nw:mean-path-length
show nw:mean-weighted-path-length "weight"
```

Clusterers and Clique Finder Primitives

nw:bicomponent-clusters

Reports the list of bicomponent clusters in the current network snapshot. The result is reported as a list of agentsets of turtles. One turtle can be a member of more than one bicomponent at once. Example:

```
let clusters nw:bicomponent-clusters
```

nw:weak-component-clusters

Reports the list of "weakly" connected components in the current network snapshot. The result is reported as a list of agentsets of turtles. One turtle cannot be a member of more than one weakly connected component at once. Example:

```
let clusters nw:weak-component-clusters
```

nw:maximal-cliques

A clique is a subset of a network in which every node has a direct link to every other node. A maximal clique is a clique that is not, itself, contained in a bigger clique. The result is reported as a list of agentsets of turtles. One turtle can be a member of more than one maximal clique at once. The primitive uses the Bron–Kerbosch algorithm and only works with undirected links. Example:

```
let cliques nw:maximal-cliques
```

nw:biggest-maximal-cliques

The biggest maximal cliques are, as the name implies, the biggest cliques in the current snapshot. Often, more than one clique are tied for the title of biggest clique, so the result if reported as a list of agentsets. Example:

```
let biggest-clique one-of nw:biggest-maximal-cliques
```

Generator Primitives

nw:generate-preferential-attachment *turtle-breed link-breed nb-nodes [commands]*
nw:generate-random *turtle-breed link-breed nb-nodes connection-prb [commands]*
nw:generate-small-world *turtle-breed link-breed rows cols exp toroidal? [commands]*
nw:generate-lattice-2d *turtle-breed link-breed rows cols exp toroidal? [commands]*
nw:generate-ring *turtle-breed link-breed nb-nodes [commands]*
nw:generate-star *turtle-breed link-breed nb-nodes [commands]*
nw:generate-wheel *turtle-breed link-breed nb-nodes [commands]*
nw:generate-wheel-inward *turtle-breed link-breed nb-nodes [commands]*
nw:generate-wheel-outward *turtle-breed link-breed nb-nodes [commands]*

The generators are amongst the only primitives that do not operate on the current network snapshot. Instead, all of them take a *turtle-breed* and a *link-breed* as inputs and generate a new network using the given breeds. Examples:

```
nw:generate-preferential-attachment turtles links 100 [ set color red ]
nw:generate-random turtles links 100 0.5 [ set color green ]
nw:generate-small-world turtles links 10 10 2.0 false [ set color blue ]
nw:generate-wheel turtles links 100 [ set color yellow ]
```

Import/Export Primitives

nw:save-matrix *file-name*

Saves the current network snapshot to *file-name*, as a text file, in the form of a simple connection matrix. At the moment, nw:save-matrix does not support link weights. Every link is represented as a "1.00" in the connection matrix. Example:

```
nw:save-matrix "matrix.txt"
```

nw:load-matrix *file-name turtle-breed link-breed [commands]*

Generates a new network according to the connection matrix saved in *file-name*, using *turtle-breed* and *link-breed* to create the new turtles and links. Please be aware that the breeds that use use to load the matrix may be different from those that you used when you saved it. Example:

```
nw:load-matrix "matrix.txt" turtles links
```

nw:save-graphml *file-name*

Saves the current snapshot in the GraphML format, including every attribute of the turtles and links included in the snapshot. Example:

```
nw:save-graphml "graph.xml"
```

nw:load-graphml *file-name*

Loads a GraphML file into NetLogo. Tries to assign the attribute values defined in the GraphML file to NetLogo agent variables of the same names (this is not case sensitive). The first one it tries to set is `breed` if it is there, so the turtle or link will get the right breed and, hence, the right breed variables.

```
nw:save-graphml "graph.xml"
```