ZIB
ZUSE INSTITUTE BERLIN

GCF

FREIE
UNIVERSITÄT
BERLIN

Vahana.jl Workshop

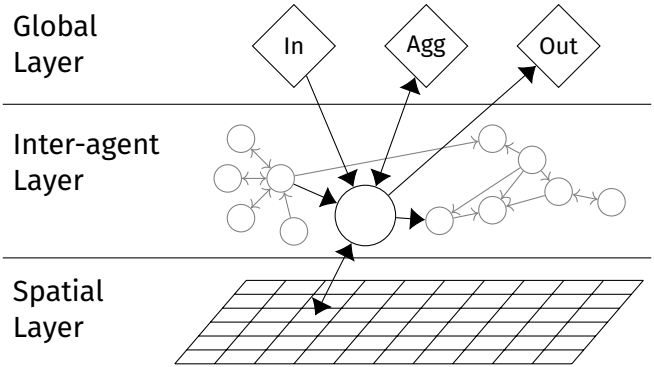
Part III - Adding a spatial component / Performance Tuning

Steffen Fürst

Scaling Complexity Workshop
Berlin, 03.12.2024

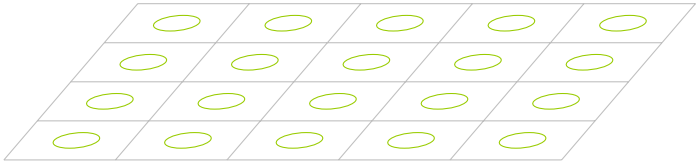
The Global Layer

The global layer conceptually contains **vertices linked to all others**, allowing distribution of parameters and aggregation of agent data, and has also a set of **helper functions**.



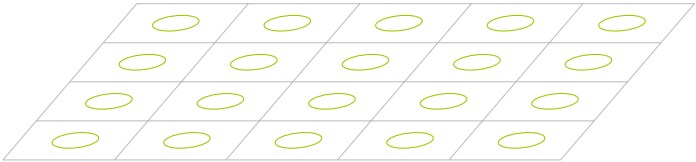
Adding Spatial Dimensions

With the function **add_raster!**, space can be represented as n-dimensional rasters, where each cell becomes a new vertex/agent:



Adding Spatial Dimensions

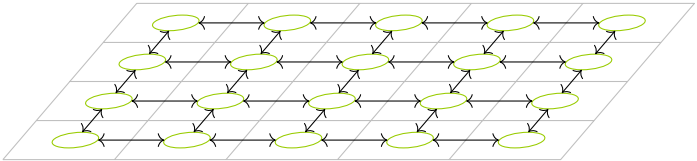
With the function **add_raster!**, space can be represented as n-dimensional rasters, where each cell becomes a new vertex/agent:



Essentially, a raster is a **mapping from spatial coordinates to vertices**, accessible to all agents, with **helper functions** to simplify working with the raster.

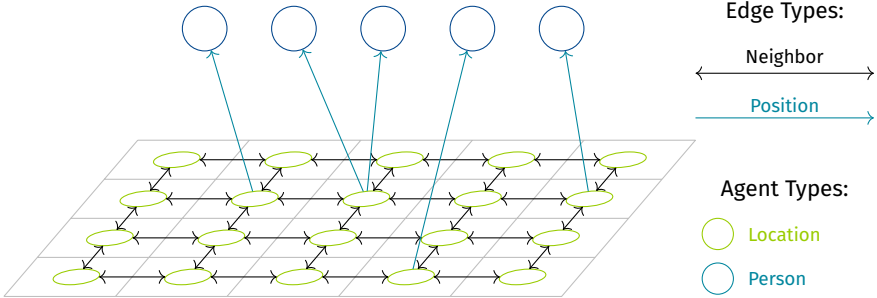
Adding Spatial Dimensions

The **connect_raster_neighbors!** method establish connections between the cells:



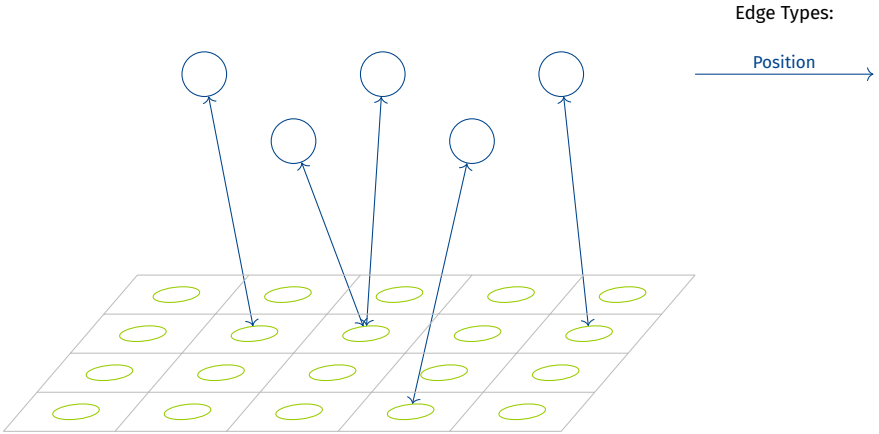
Adding Spatial Dimensions

move_to! creates edge(s) between an agent and the vertex/agent of a cell:



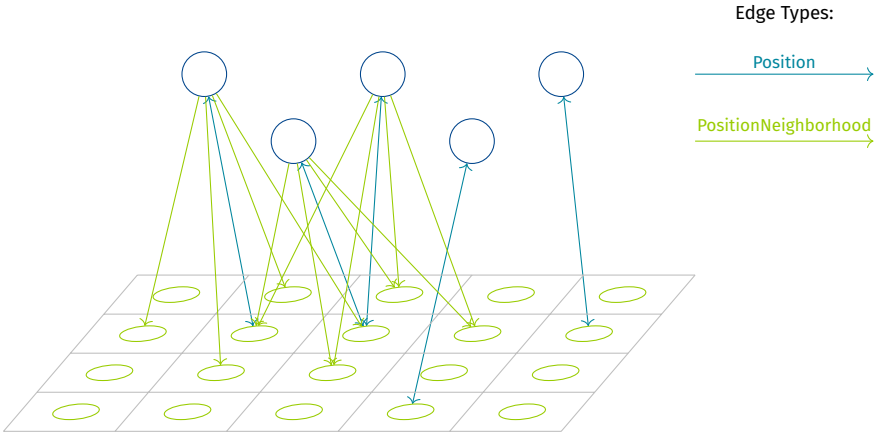
Finding Neighbors

move_to! creates edge(s) between an agent and the node/agent of a cell:



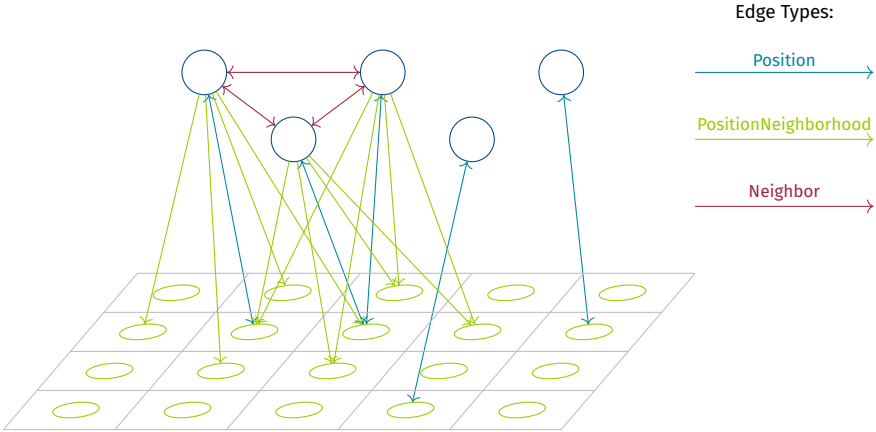
Finding Neighbors

move_to! can also creates edge(s) with the neighborhood of a cell:



Finding Neighbors

The **raster agents** can now **create edges** between neighboring agents:



Type Hints

- The specific code for accessing and managing the model's state, is **generated** during the `create_model` call.
- This code generation process can be influenced by so-called **'hints'**.
- Furthermore, the system takes into account the current mode of **parallelization**.

Edge Hints

Vahana allows users to provide optional arguments to the `register_edgetype!` function, providing additional **properties of an edge type**:

- **IgnoreFrom**: Omits storage of the source vertex's ID
- **Stateless**: Only stores the ID of the source vertex
- **SingleType**: All source vertices are of the same type
- **SingleEdge**: Each agent can be the target of only one edge (of this type).

Edge Hints

Vahana allows users to provide optional arguments to the `register_edgetype!` function, providing additional **properties of an edge type**:

- **IgnoreFrom**: Omits storage of the source vertex's ID
- **Stateless**: Only stores the ID of the source vertex
- **SingleType**: All source vertices are of the same type
- **SingleEdge**: Each agent can be the target of only one edge (of this type).

The **adjancency matrix** representation depends on the edge hints:

Hint	Representation
None	`Dict{UInt64, Vector{Edge{Friend}}}`
Stateless	`Dict{UInt64, Vector{UInt64}}`
SingleType	`Vector{Vector{Edge{Friend}}}`
All four	`Vector{Bool}`

Edge Hint Example: `add_edge!`

Generated code for `add_edge!` in the default case (**no hints** provided):

```
add_edge!(sim::Vahana.var"HK", to::UInt64, edge::Edge{Friend}) =
    push!(get!(Vector{Edge{Friend}}, sim.Friend.write, to), edge)
```

Edge Hint Example: `add_edge!`

Generated code for `add_edge!` in the default case (**no hints** provided):

```
add_edge!(sim::Vahana.var"HK", to::UInt64, edge::Edge{Friend}) =
    push!(get!(Vector{Edge{Friend}}, sim.Friend.write, to), edge)
```

And for the case when **all four** specified **hints** are applied:

```
add_edge!(sim::Vahana.var"HK", to::UInt64, edge::Edge{Friend}) =
    sim.Friend.write[to & (2 ^ 36 - 1)] = true
```