# Modelling movement for collective adaptive systems with CARMA

Natalia Zoń, Vashti Galpin and Stephen Gilmore

Laboratory for Foundations of Computer Science
University of Edinburgh

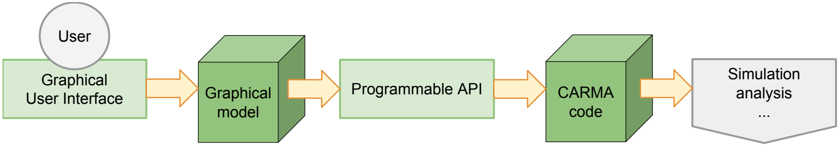
November 1, 2016

## Outline

## Outline

quanticol
www.quanticol.eu

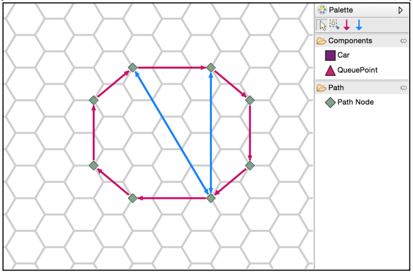
- Space and movement through space play an important role in many collective adaptive systems.
- The CARMA language and its associated software tools can be used to model such systems.
- In particular, a graphical editor for CARMA allows for the specification of spatial structure and generation of templates that can be used in a CARMA model with space.



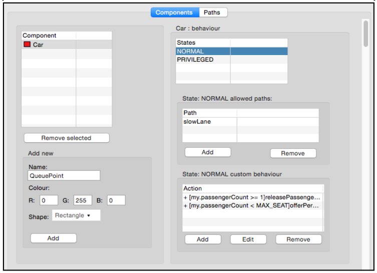
## The CARMA graphical editor

The CARMA Graphical Editor allows the user to specify the structure of movement in a CAS model by laying out graphical symbols on a plane.

$\overbrace{\hspace{6cm}}^{\text{canvas and palette}}$



$\overbrace{\hspace{6cm}}^{\text{controls}}$



| Component |
|---|
| 🟥 Car |

Add new

Name: QueuePoint

Colour:

R: 0 G: 255 B: 0

Shape: Rectangle

Car : behaviour

| States |
|---|
| NORMAL |
| PRIVILEGED |

State: NORMAL allowed paths:

| Path |
|---|
| slowLane |

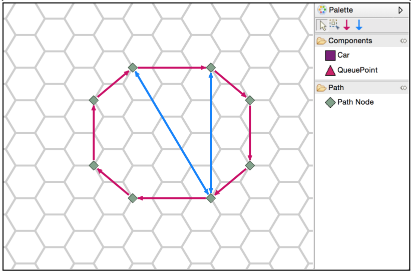
State: NORMAL custom behaviour

| Action |
|---|
| + [my.passengerCount >= 1]releasePassenge... |
| + [my.passengerCount < MAX_SEAT]offerPer... |

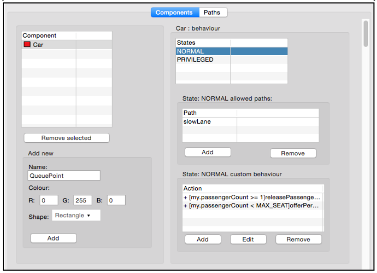
## The CARMA graphical editor

The editor generates CARMA code from the graph which the user has defined. This code generation relieves the user of the burden of creating it themselves.

canvas and palette



controls



| Component |
|---|
| 🟥 Car |

Add new

Name: QueuePoint

Colour:

R: 0 G: 255 B: 0

Shape: Rectangle

Car : behaviour

| States |
|---|
| NORMAL |
| PRIVILEGED |

State: NORMAL allowed paths:

| Path |
|---|
| slowLane |

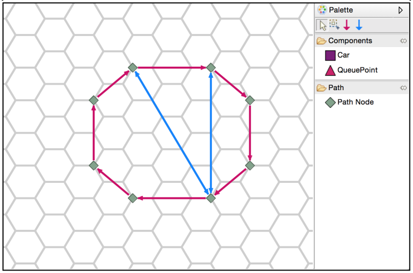
State: NORMAL custom behaviour

| Action |
|---|
| + [my.passengerCount >= 1]releasePassenge... |
| + [my.passengerCount < MAX_SEAT]offerPer... |

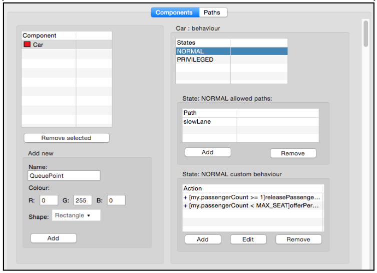
## The CARMA graphical editor

In addition to normal attributes, CARMA components which are defined in this way have a set of distinguished attributes to specify their current location in space.

canvas and palette



controls


Components | Paths

| Component |
|---|
| 🟥 Car |

Remove selected

Add new

Name: QueuePoint

Colour:

R: 0 G: 255 B: 0

Shape: Rectangle ▾

Add

Car : behaviour

| States |
|---|
| NORMAL |
| PRIVILEGED |

State: NORMAL allowed paths:

| Path |
|---|
| slowLane |

Add Remove

State: NORMAL custom behaviour

| Action |
|---|
| + [my.passengerCount >= 1]releasePassenge... |
| + [my.passengerCount < MAX_SEAT]offerPer... |

Add Edit Remove

## Outline

## An introduction to CARMA
### Collective, environment, component, process and store

- A CARMA model consists of a collective $N$ and the environment $\mathcal{E}$ in which it operates, using the syntax $N$ **in** $\mathcal{E}$.
- A collective is either a component $C$ or collectives in parallel $N \parallel N$.
- Each component is either null, **0**, or a combination of behaviour described by a process $P$ and a store of attributes $\gamma$, denoted by $(P, \gamma)$.
- We use function notation to denote store access, thus if $\gamma = \{x \mapsto v\}$ then $\gamma(x) = v$.
- A component refers to its local store with the prefix **my** (similar to **this** in Java) so an update to store the value of $x$ as the new value of **my**.$x$ is written as $\{\mathbf{my}.x \leftarrow x\}$.

## Communication in CARMA

### Unicast and broadcast communication with guards

- Process prefixes are rich and permit actions that provide value-passing unicast and broadcast communication using predicate guards on the attributes in the store of the sending and receiving component.
- Communication between components will only take place if the predicates evaluate to *true*.
- The value *false* indicates that no communication partner is needed.
- Furthermore, attribute values can be updated (probabilistically) on completion of an action.

## Processes in CARMA

### The process calculus sublanguage of CARMA

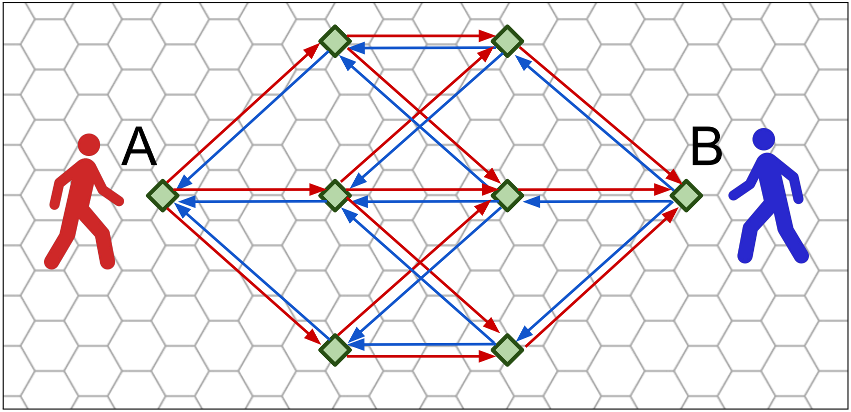
| | | | |
|---|---|---|---|
| $P, Q$ | $::=$ | $\mathbf{nil}$ | — nil process |
| | $\mid$ | $\mathbf{kill}$ | — kill a process |
| | $\mid$ | $act.P$ | — action prefix |
| | $\mid$ | $P + Q$ | — choice |
| | $\mid$ | $P \mid Q$ | — parallel composition |
| | $\mid$ | $[\pi]P$ | — guarded process |
| | $\mid$ | $A \quad (A \triangleq P)$ | — constant definitions |
| $act$ | $::=$ | $\alpha^\star[\pi]\langle\vec{e}\rangle\sigma$ | — broadcast output |
| | $\mid$ | $\alpha^\star[\pi](\vec{x})\sigma$ | — broadcast input |
| | $\mid$ | $\alpha[\pi]\langle\vec{e}\rangle\sigma$ | — unicast output |
| | $\mid$ | $\alpha[\pi](\vec{x})\sigma$ | — unicast input |

## Outline

## Pedestrian movement
### Pedestrians crossing a network of paths

- We consider the example of pedestrians moving over a network of paths.
  - This could be a specific part of a city, a pedestrianised network of lanes, or paths through a large park.
- The defining feature of our example is that there are essentially two groups of pedestrians that start on opposite sides of the network who wish to traverse the paths to get to the side opposite to where they started.

## Pedestrian movement example
### Crossing a network of paths in opposite directions



## Pedestrian behaviour
### Modelled in CARMA

$$Ped \stackrel{\text{def}}{=} \sum_{(i,j)\in V} \left[\textbf{ExistsPath}(P,x,y,i,j)\right] \left(\text{move}_{ij}{}^{\star}[\bot]\langle\rangle\{\textbf{my}.x \leftarrow i, \textbf{my}.y \leftarrow j\}.Ped\right)$$

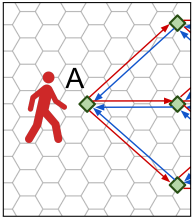
$$+ \left[\textbf{AtGoal}(P,x,y)\right]\left(\text{fin}^{\star}[\bot]\langle\rangle.\textbf{nil}\right)$$

## Pedestrian behaviour

### Modelled in CARMA

$$Ped \stackrel{\text{def}}{=} \sum_{(i,j)\in V} \left[\textbf{ExistsPath}(P,x,y,i,j)\right] \left(\text{move}_{ij}{}^{\star}[\perp]\langle\rangle\{\textbf{my}.x \leftarrow i, \textbf{my}.y \leftarrow j\}.Ped\right)$$

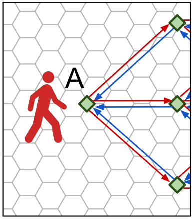
$$+ \left[\textbf{AtGoal}(P,x,y)\right]\left(\text{fin}^{\star}[\perp]\langle\rangle.\textbf{nil}\right)$$



## Pedestrian behaviour
### Modelled in CARMA

$$Ped \overset{\text{def}}{=} \sum_{(i,j)\in V} \left[\mathbf{ExistsPath}(P, x, y, i, j)\right] \left(\text{move}_{ij}{}^{\star}[\bot]\langle\rangle\{\mathbf{my}.x \leftarrow i, \mathbf{my}.y \leftarrow j\}.Ped\right)$$

$$+ \left[\mathbf{AtGoal}(P, x, y)\right]\left(\text{fin}^{\star}[\bot]\langle\rangle.\mathbf{nil}\right)$$

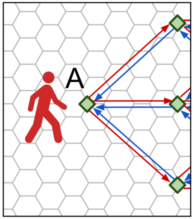


## Pedestrian behaviour

### Modelled in CARMA

$$Ped \overset{\text{def}}{=} \sum_{(i,j)\in V} \left[\textbf{ExistsPath}(P,x,y,i,j)\right] \left(\text{move}_{ij}{}^{\star}[\perp]\langle\rangle\{\textbf{my}.x \leftarrow i, \textbf{my}.y \leftarrow j\}.Ped\right)$$

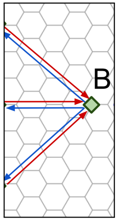
$$+ \left[\textbf{AtGoal}(P,x,y)\right]\left(\text{fin}^{\star}[\perp]\langle\rangle.\textbf{nil}\right)$$



## Pedestrian behaviour
### Modelled in CARMA

$$Ped \overset{\text{def}}{=} \sum_{(i,j)\in V} \left[\textbf{ExistsPath}(P,x,y,i,j)\right] \left(\text{move}_{ij}{}^{\star}[\bot]\langle\rangle\{\textbf{my}.x \leftarrow i, \textbf{my}.y \leftarrow j\}.Ped\right)$$

$$+ \left[\textbf{AtGoal}(P,x,y)\right]\left(\text{fin}^{\star}[\bot]\langle\rangle.\textbf{nil}\right)$$

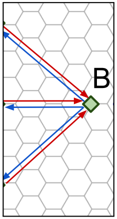


## Pedestrian behaviour

### Modelled in CARMA

$$Ped \overset{\text{def}}{=} \sum_{(i,j)\in V} \left[\mathbf{ExistsPath}(P,x,y,i,j)\right] \left(\mathrm{move}_{ij}{}^{\star}[\bot]\langle\rangle\{\mathbf{my}.x \leftarrow i, \mathbf{my}.y \leftarrow j\}.Ped\right)$$

$$+ \left[\mathbf{AtGoal}(P,x,y)\right]\left(\mathrm{fin}^{\star}[\bot]\langle\rangle.\mathbf{nil}\right)$$



## Pedestrian behaviour
### Modelled in CARMA

$$
\begin{aligned}
Ped \;\stackrel{\text{def}}{=}\; \sum_{(i,j)\in V} & \left[\textbf{ExistsPath}(P, x, y, i, j)\right] \\
& \qquad \left(\text{move}_{ij}{}^{\star}[\bot]\langle\rangle\{\textbf{my}.x \leftarrow i, \textbf{my}.y \leftarrow j\}.Ped\right) \\
& + \left[\textbf{AtGoal}(P, x, y)\right]\left(\text{fin}^{\star}[\bot]\langle\rangle.\textbf{nil}\right)
\end{aligned}
$$

The above code is generated by the CARMA graphical editor from the network graph drawn by the user. The **ExistsPath** function code is also generated by the graphical editor. The termination condition (beginning **AtGoal**) is generated by the graphical editor.

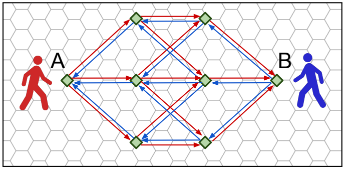
## Rates of movement

... along edges in the graph

A function that is not directly related to the graph structure is **MoveRate**$(P, x, y, i, j, \ldots)$ which determines the rate of movement along a particular edge.

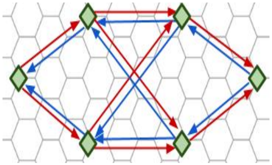
$$\textbf{MoveRate}(P, x, y, i, j, A_{ij}, B_{ij}) = \begin{cases} move_A/(B_{ij}+1) & \text{if } P = A \\ move_B/(A_{ij}+1) & \text{if } P = B \end{cases}$$

where $A_{ij}$ are the number of $A$ pedestrians at the target node and $B_{ij}$ are the number of $B$ pedestrians at the target node, and $move_Q$ is a basic movement rate for each pedestrian type.

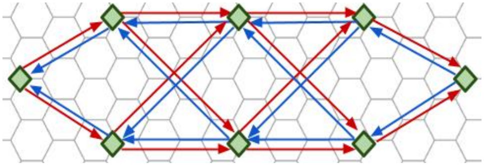


## Path networks

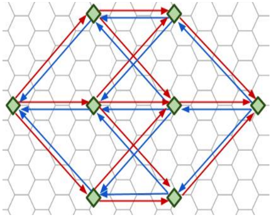
### Four model instances of increasing size and complexity



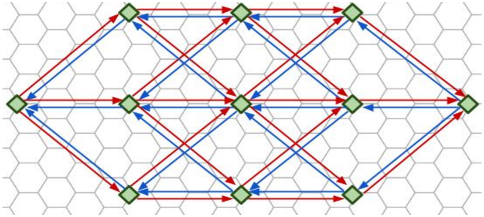
$1 \times 1$



$1 \times 2$



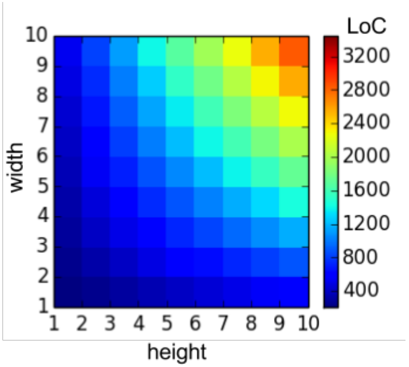
$2 \times 1$



$2 \times 2$

## Code generation for models
### Lines of CARMA model code per model structure

| Model | Nodes | Connections | LoC |
|---|---|---|---|
| 1x1 | 6 | 8 | 208 |
| 1x2 | 8 | 12 | 248 |
| 1x3 | 10 | 16 | 288 |
| 2x1 | 8 | 13 | 258 |
| 2x2 | 11 | 20 | 328 |
| 2x3 | 14 | 27 | 398 |
| 3x1 | 10 | 18 | 308 |
| 3x2 | 14 | 28 | 408 |
| 3x3 | 18 | 38 | 508 |

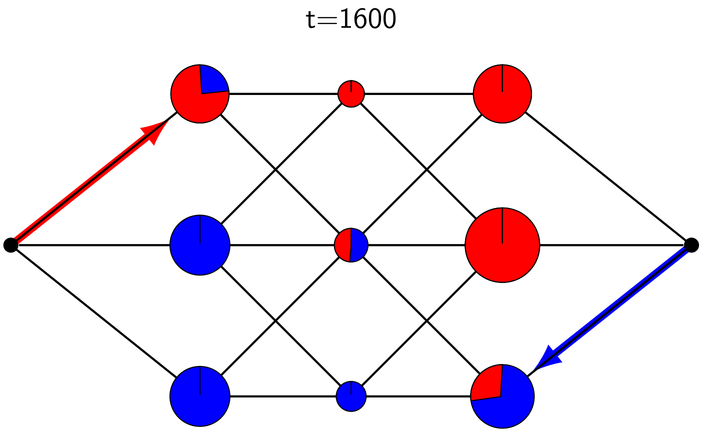


## Design of experiments
### With and without congestion and routing

- We designed a suite of experiments to explore the behaviour of the model.
- To provide a baseline for average travel time we investigated the travel time in the presence of only one type of pedestrian (thereby giving a model which has no congestion).
- Thereafter we investigated the models with congestion in the presence or absence of pedestrian routing.
- When routing is present,
  - only one starting route has a non-zero rate, and
  - the non-zero rate is assigned in order to direct pedestrians away from each other ("keep to the left").

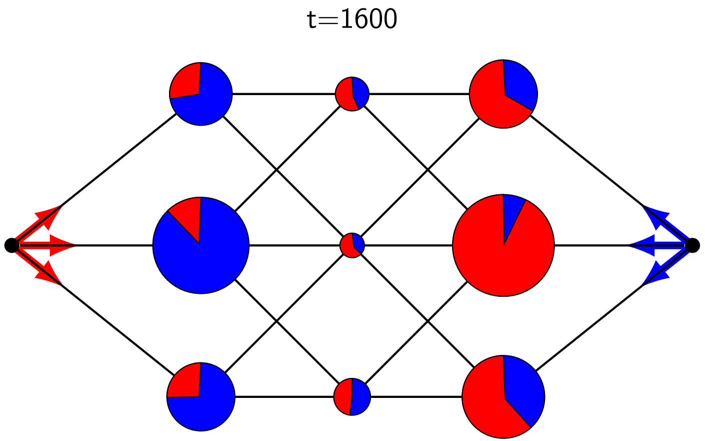
## The 2 $\times$ 2 model *with* routing

### Node size reflects number, pie-chart reflects type



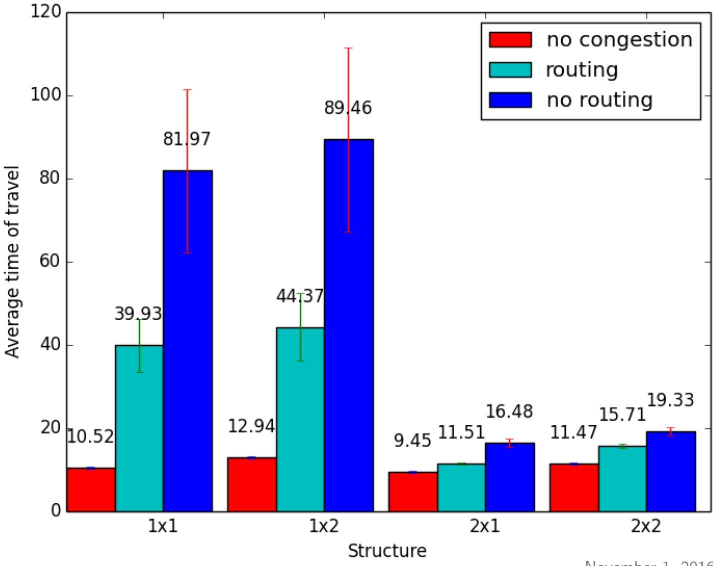
## The 2 × 2 model *without* routing

### Node size reflects number, pie-chart reflects type



## Average travel time results

... from the experiments on structure and network usage



## Outline

## Conclusions

. . . and directions for future work

- We have demonstrated a simple model of pedestrian movement over a number of different graphs, to illustrate the modelling of spatial aspects of collective adaptive systems.
- The CARMA Graphical Editor allowed us to automatically generate the CARMA code for different networks which simplified the task, and allowed our pedestrian components to be generic in nature.
- Our initial experiments have considered situations with and without congestion as well as with and without explicit routing of pedestrians as they enter the network.
    - We are also interested in identifying when the model shows emergent behaviour, in the sense that different groups of pedestrian use different paths through the network in response to environmental cues rather than explicit routing.

## Acknowledgements

- This work is supported by the EU project *QUANTICOL: A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours*, 600708.

Thank you for your attention.