Studying Smart Cities as Collective Adaptive Systems

Jane Hillston
LFCS, University of Edinburgh

15th March 2017

Bio-Inspiration!

Performance Modelling of Computer Systems

⇓

Systems Biology of Intracellular Processes

⇓

Ecological Systems as Collective Adaptive Systems

⇓

Smart Cities as Collective Adaptive Systems

Outline

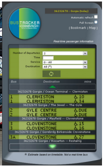
Outline

The Informatic Environment

Robin Milner coined the term of informatics environment, in which pervasive computing elements are embedded in the human environment, invisibly providing services and responding to requirements.

Such systems underpin the current trend towards smart cities.

These systems are developed on the basis that information flows within the system, from the users to the service provider and from the service provider to the user, creating a dynamic ecosystem.



Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

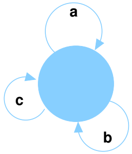
Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

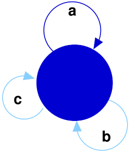


Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

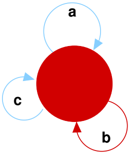


Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

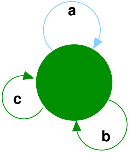


Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.

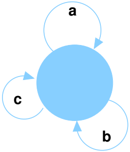


Quantitative Modelling

The pervasive and embedded nature of the informatics environment means it is imperative that we are able to reason about their behaviour before deployment.

Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.



Performance Modelling: Motivation

Capacity planning

- How many buses do I need to maintain service at peak time in a smart urban transport system?

Performance Modelling: Motivation

System Configuration

- What capacity do I need at bike stations to minimise the movement of bikes by truck?

Performance Modelling: Motivation

System Tuning

- What strategy can I use to maintain supply-demand balance within a smart electricity grid?

Performance Modelling

The size and complexity of real systems makes the direct construction of discrete state models costly and error-prone.

Instead models are constructed using formal modelling techniques enhanced with information about timing and probability, such as stochastic Petri nets and stochastic process algebras.

From these high-level system descriptions the underlying mathematical model (Continuous Time Markov Chain (CTMC)) can be automatically generated.

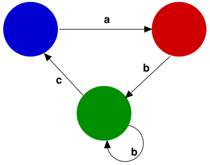
Stochastic Process Algebra

- Stochastic process algebras are simple system description languages where the focus is on components that engage in activities.
- Activities have a name and a stochastic rate and a small set of language constructs determine which activities are possible in each state.
- Every expression in the language can be used to generate a CTMC for quantitative analysis.

Stochastic Process Algebra

- Stochastic process algebras are simple system description languages where the focus is on components that engage in activities.
- Activities have a name and a stochastic rate and a small set of language constructs determine which activities are possible in each state.
- Every expression in the language can be used to generate a CTMC for quantitative analysis.

Process algebra model —SOS rules→ Labelled transition system



CTMC **Q**

J.Hillston, A Compositional Approach to Performance Modelling, CUP, 1995

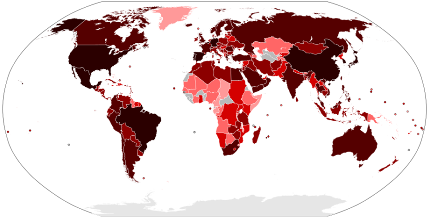
Collective Systems

We are surrounded by examples of collective systems:

Collective Systems

We are surrounded by examples of collective systems:

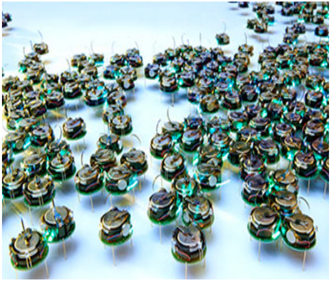
in the man-made world....



Collective Systems

We are surrounded by examples of collective systems:

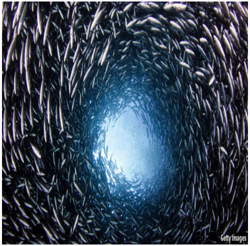
in the man-made world....



Collective Systems

We are surrounded by examples of collective systems:

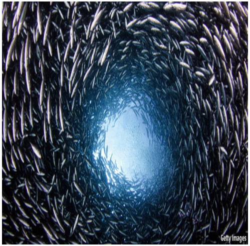
.... and in the natural world.



Collective Systems

We are surrounded by examples of collective systems:

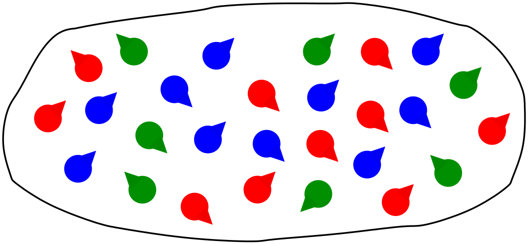
.... and in the natural world.



Most of these systems are also adaptive to their environment

Collective Adaptive Systems

From a computer science perspective these systems can be viewed as being made up of a large number of interacting entities.

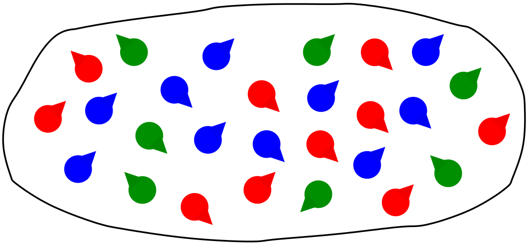


Each entity may have its own properties, objectives and actions.

At the system level these combine to create collective behaviour.

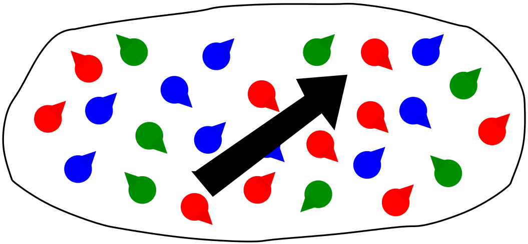
Collective Adaptive Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



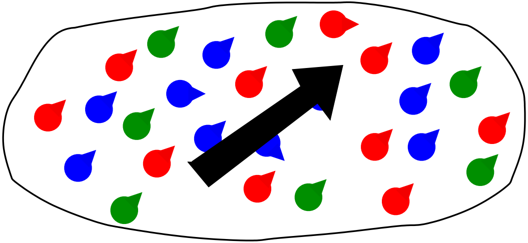
Collective Adaptive Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



Collective Adaptive Systems

The behaviour of the system is thus dependent on the behaviour of the individual entities.



And the behaviour of the individuals will be influenced by the state of the overall system, leading to autonomous adaptation.

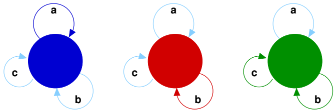
Bio-inspiration

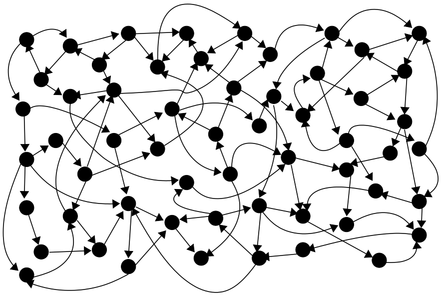
What can we learn from the way that the CAS in nature have been modelled to understand their behaviour, in order to build formal modelling frameworks for engineered CAS that allow us to reason about their behaviour before they are deployed?

Outline

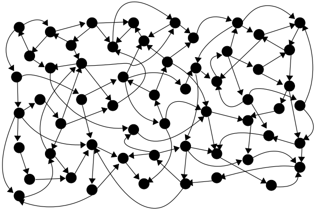
Solving discrete state models

Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.





Solving discrete state models



When the size of the state space is not too large they are amenable to numerical solution (linear algebra) to determine a steady state or transient probability distribution.

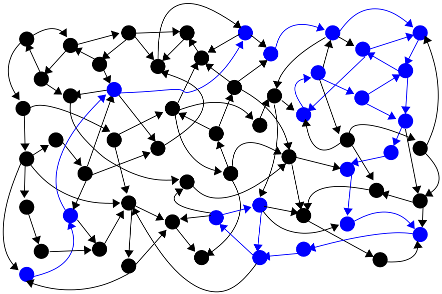
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

$$\pi(t) = (\pi_1(t), \pi_2(t), \ldots, \pi_N(t))$$

$$\pi(\infty)Q = 0$$

Solving discrete state models

Alternatively they may be studied using stochastic simulation. Each run generates a single trajectory through the state space. Many runs are needed in order to obtain average behaviours.



State space explosion

As the size of the state space becomes large it becomes infeasible to carry out numerical solution and extremely time-consuming to conduct stochastic simulation.

Modelling collective behaviour

- A key feature of collective systems is the existence of populations of entities who share certain characteristics.
- In disciplines such as ecology and cellular biology, large scale discrete systems are routines treated as if they were continuous.
- For example, in protein interactions concentrations are modelled rather than counts of molecules; in SIR models the proportion of the population that are infected, is modelled rather than numbers of individuals.
- Whilst this shift from discrete to continuous is often made informally, it can have a sound mathematical basis.

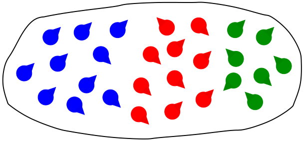
T.G.Kurtz, *Approximation of Population Processes*, SIAM 1981

The Fluid Approximation Alternative

Analogously in the formal setting, we can shift attention from the individual entities to the populations, and then consider the average behaviour within a population.

The Fluid Approximation Alternative

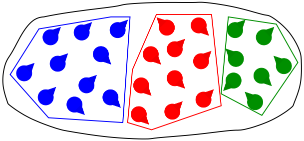
Analogously in the formal setting, we can shift attention from the individual entities to the populations, and then consider the average behaviour within a population.



Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space.

The Fluid Approximation Alternative

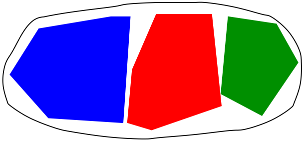
Analogously in the formal setting, we can shift attention from the individual entities to the populations, and then consider the average behaviour within a population.



Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space.

The Fluid Approximation Alternative

Analogously in the formal setting, we can shift attention from the individual entities to the populations, and then consider the average behaviour within a population.

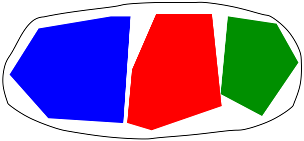


Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space.

We no longer regard the components as individuals, instead focussing on the proportion of the population exhibiting certain behaviours.

The Fluid Approximation Alternative

Analogously in the formal setting, we can shift attention from the individual entities to the populations, and then consider the average behaviour within a population.



Ceasing to distinguish between instances of components we form an aggregation or counting abstraction to reduce the state space.

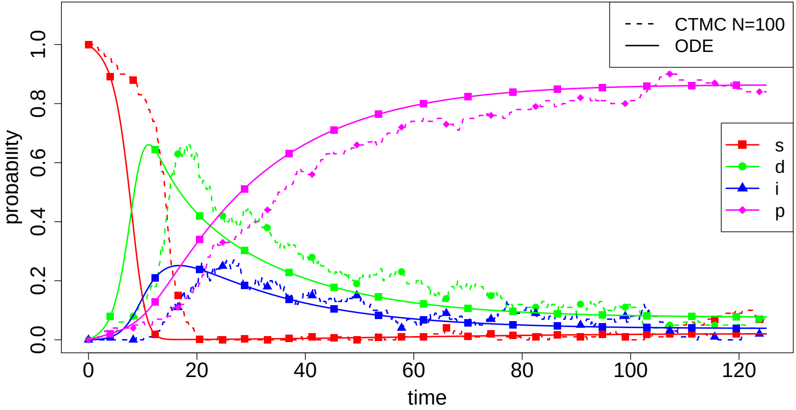
We no longer regard the components as individuals, instead focussing on the proportion of the population exhibiting certain behaviours.

Furthermore we make a continuous or fluid approximation of how the proportions vary over time.

M.Tribastone, S.Gilmore and J.Hillston. Scalable Differential Analysis of Process Algebra Models. IEEE TSE 2012.

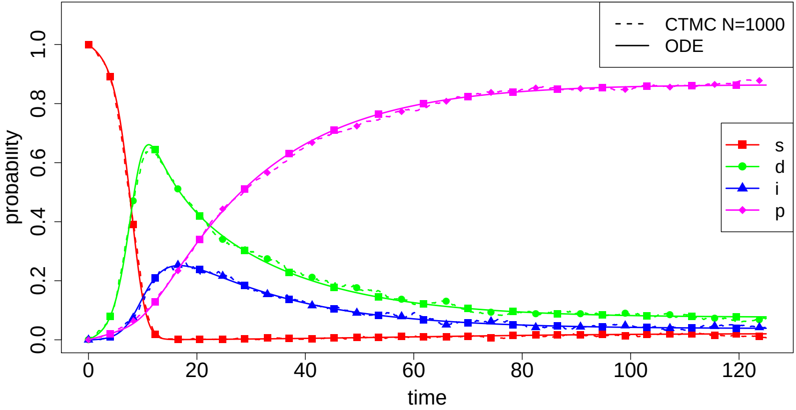
Illustrative trajectories

Limit fluid ODE and single stochastic trajectory of a network epidemics example for $N = 100$



Illustrative trajectories

Limit fluid ODE and single stochastic trajectory of a network epidemics example for $N = 1000$



Challenges for modelling CAS

The work over the last decade demonstrates a solid basic framework for modelling systems with collective behaviour but there remain a number of challenges:

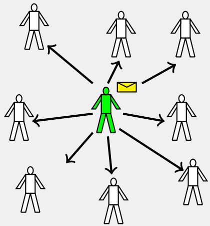
- Richer forms of interaction
- The influence of space on behaviour
- Capturing adaptivity

Interaction patterns in CAS

Typically, CAS exhibit two kinds of interaction pattern:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading: 1-to-many

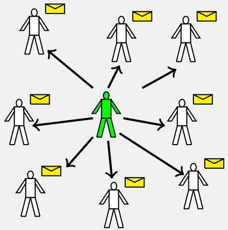


Interaction patterns in CAS

Typically, CAS exhibit two kinds of interaction pattern:

1. **Spreading**: one agent spreads relevant information to a given group of other agents

Spreading: 1-to-many

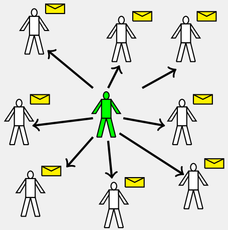


Interaction patterns in CAS

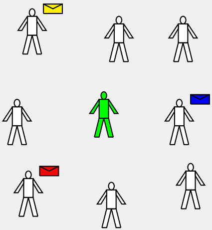
Typically, CAS exhibit two kinds of interaction pattern:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading: 1-to-many



Collecting: 1-to-1

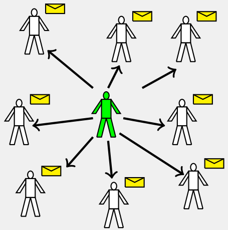


Interaction patterns in CAS

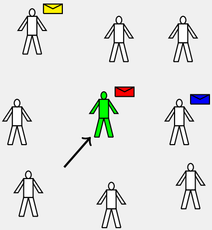
Typically, CAS exhibit two kinds of interaction pattern:

1. **Spreading**: one agent spreads relevant information to a given group of other agents
2. **Collecting**: one agent changes its behaviour according to data collected from one agent belonging to a given group of agents.

Spreading: 1-to-many



Collecting: 1-to-1



Modelling space

Location and movement play an important role within many CAS, especially smart cities.

We can encode space into the behaviour of the actions of components (e.g. using different names in different locations) and so distinguishing the same component in different locations, but this only captures space implicitly.

It would be preferable to model space explicitly but this poses significant challenges both for model expression and model solution.

Moreover this is difficult for scalable analysis which is often based on an assumption that all components are co-located.

Capturing adaptivity

Existing process algebras, tend to work with a fixed set of actions for each entity type.

Some stochastic process algebras allow the rate of activity to be dependent on the state of the system.

But for truly adaptive systems there should also be some way to identify the goal or objective of entity in addition to its behaviour.

Outline

A new language for CAS

CARMA (Collective Adaptive Resource-sharing Markovian Agents), is a new language stochastic process algebra-based language for CAS which handles:

1. The behaviours of agents and their interactions;
2. The global knowledge of the system and that of its agents;
3. The environment where agents operate...

A new language for CAS

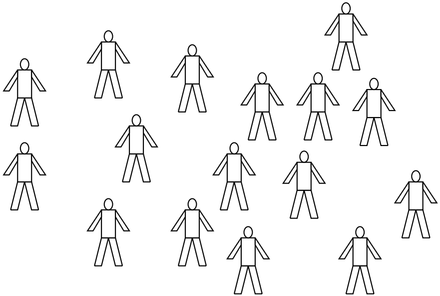
CARMA (Collective Adaptive Resource-sharing Markovian Agents), is a new language stochastic process algebra-based language for CAS which handles:

1. The behaviours of agents and their interactions;
2. The global knowledge of the system and that of its agents;
3. The environment where agents operate...
   - taking into account open ended-ness and adaptation;
   - taking into account resources, locations and visibility/reachability issues.

M.Loreti et al. CARMA: Collective Adaptive Resource-sharing Markovian Agents. QAPL 2015.

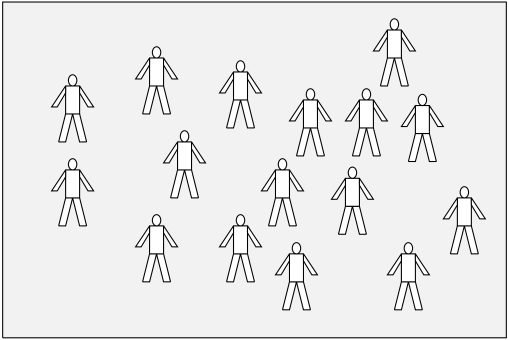
CAS: CARMA perspective

Collective



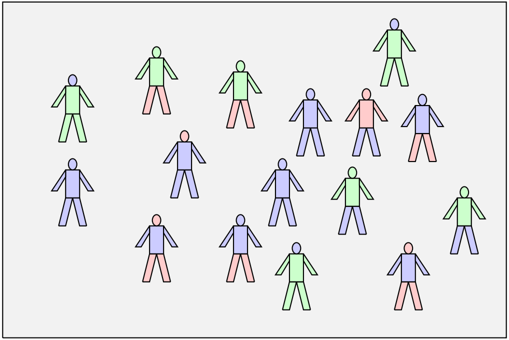
CAS: CARMA perspective

Collective **Environment**



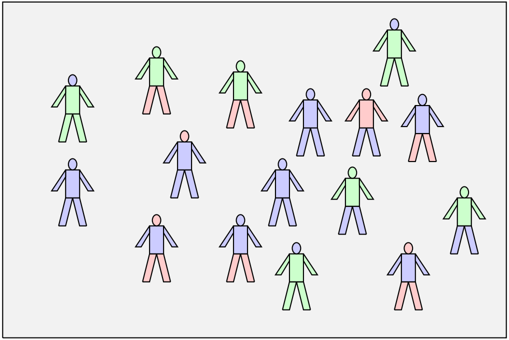
CAS: CARMA perspective

Collective **Environment** **Attributes**



CAS: CARMA perspective

Collective **Environment** **Attributes**



Processes are referenced via their attributes.

Components

Agents in CARMA are defined as components C of the form (P, γ) where...

- P is a process, representing agent behaviour;
- γ is a store, modelling agent knowledge.

Components

Agents in CARMA are defined as components C of the form (P, γ) where. . .

- P is a process, representing agent behaviour;
- γ is a store, modelling agent knowledge.

The participants of an interaction are identified via predicates. . .

- the counterpart of a communication is selected according its properties
- both sender and receiver can filter messages using predicates, choosing who they are willing to communicate with

Interaction primitives

We term this attribute based communication and we support several forms:. . .

Interaction primitives

We term this attribute based communication and we support several forms:. . .

- **Broadcast output**: a message is sent to all the components satisfying a predicate π;
- **Broadcast input**: a process is willing to receive a broadcast message from a component satisfying a predicate π;

Interaction primitives

We term this attribute based communication and we support several forms:. . .

- **Broadcast output**: a message is sent to all the components satisfying a predicate π;
- **Broadcast input**: a process is willing to receive a broadcast message from a component satisfying a predicate π;
- **Unicast output**: a message is sent to one of the components satisfying a predicate π;
- **Unicast input**: a process is willing to receive a message from a component satisfying a predicate π.

Interaction primitives

We term this attribute based communication and we support several forms:...

- **Broadcast output**: a message is sent to all the components satisfying a predicate π;
- **Broadcast input**: a process is willing to receive a broadcast message from a component satisfying a predicate π;
- **Unicast output**: a message is sent to one of the components satisfying a predicate π;
- **Unicast input**: a process is willing to receive a message from a component satisfying a predicate π.

The execution of an action takes an exponentially distributed time; the rate of each action is determined by the environment.

Interaction primitives

Syntax

$$
\begin{array}{lcll}
act & ::= & \alpha^{\star}[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Broadcast output} \\
 & \mid & \alpha^{\star}[\pi](\overrightarrow{x})\sigma & \text{Broadcast input} \\
 & \mid & \alpha[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Unicast output} \\
 & \mid & \alpha[\pi](\overrightarrow{x})\sigma & \text{Unicast input}
\end{array}
$$

Interaction primitives

Syntax

$$
\begin{array}{lcll}
act & ::= & \alpha^{\star}[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Broadcast output} \\
 & \mid & \alpha^{\star}[\pi](\overrightarrow{x})\sigma & \text{Broadcast input} \\
 & \mid & \alpha[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Unicast output} \\
 & \mid & \alpha[\pi](\overrightarrow{x})\sigma & \text{Unicast input}
\end{array}
$$

- α is an action type;

Interaction primitives

Syntax

act	$::=$	$\alpha^{\star}[\pi]\langle\overrightarrow{e}\rangle\sigma$	Broadcast output
	$\mid$	$\alpha^{\star}[\pi](\overrightarrow{x})\sigma$	Broadcast input
	$\mid$	$\alpha[\pi]\langle\overrightarrow{e}\rangle\sigma$	Unicast output
	$\mid$	$\alpha[\pi](\overrightarrow{x})\sigma$	Unicast input

- α is an action type;
- π is a predicate;

Interaction primitives

Syntax

$$
\begin{array}{llll}
act & ::= & \alpha^{\star}[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Broadcast output} \\
 & \mid & \alpha^{\star}[\pi](\overrightarrow{x})\sigma & \text{Broadcast input} \\
 & \mid & \alpha[\pi]\langle\overrightarrow{e}\rangle\sigma & \text{Unicast output} \\
 & \mid & \alpha[\pi](\overrightarrow{x})\sigma & \text{Unicast input}
\end{array}
$$

- α is an action type;
- π is a predicate;
- σ is the effect of the action on the store.

Updating the store

After the execution of an action, a process can update the component store:

- updates are instantaneous
- σ is a function mapping attribute γ to a probability distribution over possible values (the deterministic distribution in most cases)

$$\mathsf{move}^{\star}[\pi]\langle v\rangle\{x := x + U(-1,+1)\}$$

More on synchronisation

Predicates regulating broadcast/unicast inputs can refer also to the received values.

More on synchronisation

Predicates regulating broadcast/unicast inputs can refer also to the received values.

Example:

A value greater than 0 is expected from a component with a *trust_level* less than 3:

$$\alpha^{\star}[(x > 0) \wedge (trust\_level < 3)](x)\sigma.P$$

Examples of interactions. . .

Broadcast synchronisation:

$$
\begin{aligned}
&(\ \mathrm{stop}^{\star}[\mathrm{bl} < 5\%]\langle v\rangle\sigma_1.P\ ,\{role = \text{“}master\text{”}\})\ \|\\
&\quad(\ \mathrm{stop}^{\star}[\mathrm{role} = \text{“}master\text{”}](x)\sigma_2\ .Q_1\ ,\{\mathrm{bl} = 4\%\})\ \|\\
&\qquad(\ \mathrm{stop}^{\star}[\mathrm{role} = \text{“}super\text{”}](x)\sigma_3.Q_2\ ,\{\mathrm{bl} = 2\%\})\ \|\\
&\qquad\qquad(\ \mathrm{stop}^{\star}[\top](x)\sigma_4.Q_3\ ,\{\mathrm{bl} = 2\%\})
\end{aligned}
$$

Examples of interactions...

Broadcast synchronisation:

$$
\begin{array}{l}
(\ \mathsf{stop}^{\star}[\mathsf{bl} < 5\%]\langle v\rangle\sigma_1.P\ ,\{role = \text{“}master\text{”}\})\ \|\\
\quad (\ \mathsf{stop}^{\star}[\mathsf{role} = \text{“}master\text{”}](x)\sigma_2\ .Q_1\ ,\{\mathsf{bl} = 4\%\})\ \|\\
\qquad (\ \mathsf{stop}^{\star}[\mathsf{role} = \text{“}super\text{”}](x)\sigma_3.Q_2\ ,\{\mathsf{bl} = 2\%\})\ \|\\
\qquad\qquad (\ \mathsf{stop}^{\star}[\top](x)\sigma_4.Q_3\ ,\{\mathsf{bl} = 2\%\})
\end{array}
$$

Examples of interactions...

Broadcast synchronisation:

$$
\begin{array}{l}
(\ \mathrm{stop}^{\star}[\mathrm{bl} < 5\%]\langle v\rangle\sigma_1.P\ ,\{role = \text{``}master\text{''}\})\ \|\\
\quad(\ \mathrm{stop}^{\star}[\mathrm{role} = \text{``}master\text{''}](x)\sigma_2\ .Q_1\ ,\{\mathrm{bl} = 4\%\})\ \|\\
\qquad(\ \mathrm{stop}^{\star}[\mathrm{role} = \text{``}super\text{''}](x)\sigma_3.Q_2\ ,\{\mathrm{bl} = 2\%\})\ \|\\
\qquad\qquad(\ \mathrm{stop}^{\star}[\top](x)\sigma_4.Q_3\ ,\{\mathrm{bl} = 2\%\})
\end{array}
$$

Examples of interactions...

Broadcast synchronisation:

$$
\begin{aligned}
&(\mathsf{stop}^\star[\mathsf{bl} < 5\%]\langle v\rangle\sigma_1.P \ , \{\mathit{role} = \text{``}\mathit{master}\text{''}\}) \parallel \\
&\quad (\mathsf{stop}^\star[\mathsf{role} = \text{``}\mathit{master}\text{''}](x)\sigma_2 \ .Q_1 \ , \{\mathsf{bl} = 4\%\}) \parallel \\
&\qquad (\mathsf{stop}^\star[\mathsf{role} = \text{``}\mathit{super}\text{''}](x)\sigma_3.Q_2 \ , \{\mathsf{bl} = 2\%\}) \parallel \\
&\qquad\qquad (\mathsf{stop}^\star[\top](x)\sigma_4.Q_3 \ , \{\mathsf{bl} = 2\%\})
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{aligned}
&(P, \sigma_1(\{\mathit{role} = \text{``}\mathit{master}\text{''}\})) \parallel \\
&\quad (Q_1[v/x], \sigma_2(\{\mathsf{bl} = 4\%\})) \parallel \\
&\qquad (\mathsf{stop}^\star[\mathsf{role} = \text{``}\mathit{super}\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel \\
&\qquad\qquad (Q_3[v/x], \sigma_4(\{\mathsf{bl} = 2\%\}))
\end{aligned}
$$

Examples of interactions...

Broadcast synchronisation:

$$
\begin{aligned}
&(\mathsf{stop}^{\star}[\mathsf{bl} < 5\%]\langle v\rangle\sigma_1.P, \{role = \text{``}master\text{''}\}) \parallel \\
&\quad (\mathsf{stop}^{\star}[\mathsf{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\mathsf{bl} = 45\%\}) \parallel \\
&\qquad (\mathsf{stop}^{\star}[\mathsf{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel \\
&\qquad\qquad (\mathsf{stop}^{\star}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 25\%\})
\end{aligned}
$$

Examples of interactions...

Broadcast synchronisation:

$$(\mathsf{stop}^{\star}[\mathsf{bl} < 5\%]\langle v\rangle\sigma_1.P, \{role = \text{“}master\text{”}\}) \parallel$$
$$(\mathsf{stop}^{\star}[\mathsf{role} = \text{“}master\text{”}](x)\sigma_2.Q_1, \{\mathsf{bl} = 45\%\}) \parallel$$
$$(\mathsf{stop}^{\star}[\mathsf{role} = \text{“}super\text{”}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel$$
$$(\mathsf{stop}^{\star}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 25\%\})$$

Examples of interactions...

Broadcast synchronisation:

$$(\text{stop}^{\star}[\text{bl} < 5\%]\langle v\rangle\sigma_1.P, \{role = \text{``}master\text{''}\}) \parallel$$
$$(\text{stop}^{\star}[\text{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\text{bl} = 45\%\}) \parallel$$
$$(\text{stop}^{\star}[\text{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\text{bl} = 2\%\}) \parallel$$
$$(\text{stop}^{\star}[\top](x)\sigma_4.Q_3, \{\text{bl} = 25\%\})$$

$$\Downarrow$$

$$(P, \sigma_1(\{role = \text{``}master\text{''}\})) \parallel$$
$$(\text{stop}^{\star}[\text{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\text{bl} = 45\%\}) \parallel$$
$$(\text{stop}^{\star}[\text{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\text{bl} = 2\%\}) \parallel$$
$$(\text{stop}^{\star}[\top](x)\sigma_4.Q_3, \{\text{bl} = 25\%\})$$

Examples of interactions. . .

Unicast synchronisation:

$$(\mathsf{stop}[\mathsf{bl} < 5\%]\langle\bullet\rangle\sigma_1.P, \{role = \text{``}master\text{''}\}) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\mathsf{bl} = 4\%\}) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel$$
$$(\mathsf{stop}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 2\%\})$$

Examples of interactions...

Unicast synchronisation:

$$
\begin{aligned}
&(\mathsf{stop}[\mathsf{bl} < 5\%]\langle\bullet\rangle\sigma_1.P, \{role = \text{“}master\text{”}\}) \parallel \\
&\quad (\mathsf{stop}[\mathsf{role} = \text{“}master\text{”}](x)\sigma_2.Q_1, \{\mathsf{bl} = 4\%\}) \parallel \\
&\qquad (\mathsf{stop}[\mathsf{role} = \text{“}super\text{”}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel \\
&\qquad\qquad (\mathsf{stop}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 2\%\})
\end{aligned}
$$

Examples of interactions...

Unicast synchronisation:

$$
\begin{aligned}
&(\mathsf{stop}[\mathsf{bl} < 5\%]\langle\bullet\rangle\sigma_1.P, \{role = \text{``}master\text{''}\}) \parallel \\
&\quad (\mathsf{stop}[\mathsf{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\mathsf{bl} = 4\%\}) \parallel \\
&\qquad (\mathsf{stop}[\mathsf{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel \\
&\qquad\qquad (\mathsf{stop}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 2\%\})
\end{aligned}
$$

Examples of interactions...

Unicast synchronisation:

$$(\mathsf{stop}[\mathsf{bl} < 5\%]\langle\bullet\rangle\sigma_1.P, \{role = \text{``}master\text{''}\}) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\mathsf{bl} = 4\%\}) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel$$
$$(\mathsf{stop}[\top](x)\sigma_4.Q_3, \{\mathsf{bl} = 2\%\})$$

$$\Downarrow$$

$$(P, \sigma_1(\{role = \text{``}master\text{''}\})) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}master\text{''}](x)\sigma_2.Q_1, \{\mathsf{bl} = 4\%\}) \parallel$$
$$(\mathsf{stop}[\mathsf{role} = \text{``}super\text{''}](x)\sigma_3.Q_2, \{\mathsf{bl} = 2\%\}) \parallel$$
$$(Q_3, \sigma_4(\{\mathsf{bl} = 2\%\}))$$

Modelling the environment

Interactions between components can be affected by the environment:

- a wall can inhibit wireless interactions;
- two components are too distant to interact;
- . . .

The environment. . .

- is used to model the intrinsic rules that govern the physical context;
- consists of a pair (γ, ρ):
  - a global store γ, that models the overall state of the system;
  - an evolution rule ρ that regulates component interactions (receiving probabilities, action rates,. . .).

Example: Smart Taxi System

System description:

- We consider a set of taxis operating in a city, providing service to users;
- Both taxis and users are modelled as components.
- The city is subdivided into a number of patches arranged in a grid over the geography of the city.
- The users arrive randomly in different patches, at a rate that depends on the specific time of day.
- After arrival, a user makes a call for a taxi and then waits in that patch until they successfully engage a taxi and move to another randomly chosen patch.
- Unengaged taxis move about the city, influenced by the calls made by users.

J.Hillston and M.Loreti. Specification and analysis of open-ended systems with CARMA. In LNCS 9068, 2015.

Taxis and Users: stores

Components use the local store to capture the relevant data that will be used to represent the state of the agent.

Taxis

- *loc*: identifies current taxi location;
- *occupancy*: ranging in $\{0, 1\}$ describes if a taxi is free ($occupancy = 0$) or engaged ($occupancy = 1$);
- *dest*: if occupied, this attribute indicates the destination of the taxi journey.

Users

- *loc*: identifies user location;
- *dest*: indicates user destination.

User processes

Users

process $User$ =

$Wait : \text{call}^{\star}[\top]\langle \text{my.loc.}x, \text{my.loc.}y\rangle.Wait$

$+$

$\text{take}[\text{loc.}x == \text{my.loc.}x \;\wedge\; \text{loc.}y == \text{my.loc.}y]$

$\langle \text{my.dest.}x, \text{my.dest.}y\rangle.\mathbf{kill}$

endprocess

Taxi processes

Taxis

process $Taxi$ =

F : $\text{call}^{\star}[(\text{my.loc.x} \neq \text{posx}) \wedge \text{my.loc.y} \neq \text{posy}](posx, posy)$
$\{\text{dest} := [x := posx, y := posy]\}.G$

$+$

$\text{take}[\top](posx, posy)$
$\{\text{dest} := [x := posx, y := posy], \text{occupancy} := 1\}.G$

G : $\text{move}^{\star}[\bot]\langle\circ\rangle$
$\{\text{loc} := \text{dest}, \text{dest} := [x := 3, y := 3], \text{occupancy} := 0\}.F$

endprocess

Modelling arrivals

The Arrivals process has a single attribute loc.

Arrivals process for users

$$\begin{aligned}&\text{process } \mathit{Arrivals} = \\ &\qquad A : \text{arrival}^{\star}[\bot]\langle\circ\rangle.A \\ &\text{endprocess}\end{aligned}$$

This process is executed in a separated component where attribute loc indicates the location where the user arrives.

The Environment: the evolution rule ρ

ρ is a function, dependent on current time, the global store and the current state of the collective, returns a tuple of functions $\varepsilon = \langle \mu_p, \mu_w, \mu_r, \mu_u \rangle$ known as the evaluation context

- $\mu_p(\gamma_s, \gamma_r, \alpha)$: the probability that a component with store γ_r can receive a broadcast message α from a component with store γ_s;
- $\mu_w(\gamma_s, \gamma_r, \alpha)$: the weight to be used to compute the probability that a component with store γ_r can receive a unicast message α from a component with store γ_s;
- $\mu_r(\gamma_s, \alpha)$ computes the execution rate of action α executed at a component with store γ_s;
- $\mu_u(\gamma_s, \alpha)$ determines the updates on the environment (global store and collective) induced by the execution of action α at a component with store γ_s.

Evolution rule: μ_p

Defining the probabilities of broadcast actions

$$\mathsf{prob}\{\quad \top, \mathsf{call}^\star : \mathsf{global.p_{lost}} \quad \mathsf{default}\ \ 1 \quad \}$$

- $\mathsf{call}^\star$ can be missed with a probability p_{lost} defined in the global store.
- All the other interactions occur with probability 1.

Evolution rule: μ_w

Defining the weights of unicast actions

$$
\begin{array}{l}
\text{prob}\{ \\
\quad \top, \text{take} : \mathit{Takeprob}(\mathit{real}(\#\{\mathit{Taxi}[F] \mid \\
\qquad\qquad (\text{my.loc.x} == \text{sender.loc.x}) \wedge \\
\qquad\qquad (\text{my.loc.y} == \text{sender.loc.y})\})); \\
\}
\end{array}
$$

- Each taxi receives a user request (take) with a weight that depends on the number of taxis in the patch.

Evolution rule: μ_r

Defining the rates of actions

$$
\begin{array}{l}
\text{rate}\{ \\
\quad \top, \text{take} : \text{global}.\text{r}_\text{t} \\
\quad \top, \text{call}^\star : \text{global}.\text{r}_\text{c} \\
\quad \top, \text{move}^\star : \mathit{Mtime}(\text{now}, \text{sender.loc}, \text{sender.dest}, 6) \\
\quad \top, \text{arrival}^\star : \mathit{Atime}(\text{now}, \text{sender.loc}, 1) \\
\quad \text{default} \;\; 0 \\
\}
\end{array}
$$

While take and call have constant rates, the rates of the actions move and arrival are functions that depend on time, reflecting shifting traffic patterns within the city over the course of a day.

Evolution rule: μ_u

In the taxi example, the arrival of a new user is achieved via the update rule:

Update rule

$$\text{update}\{ \\ \quad \top, \text{arrival}^\star : \textbf{new}\ \text{User}(\text{sender.loc}, \text{DestLoc}(\text{now}, \text{sender.loc}), \text{Wait}) \\ \}$$

Measures

To extract data from a system, a CARMA specifications also contains a set of measures.

The number of waiting users at a location

$$\textbf{measure}\ \text{WaitingUser}_{00}[i := 0] = \#\{\text{User}[\text{Wait}] \mid \text{my.loc.x} == 0 \ \wedge \ \text{my.loc.y} == 0\};$$

The number of taxis relocating

$$\textbf{measure}\ \text{Taxi\_Relocating}[i := 1] = \#\{\text{Taxi}[G] \mid \text{my.occupancy} == 0\};$$

Two Scenarios

We consider a grid of 3×3 patches, i.e., a set of locations (i, j) where $0 \leq i, j \leq 2$, and two different scenarios:

- **Scenario 1:** Users arrive in all the patches at the same rate;
- **Scenario 2:** At the beginning users arrive with a higher probability to the patches at the border of the grid; subsequently, users arrive with higher probability in the centre of the grid.

These are investigated by placing the same collective in different environments.

Smart Taxi System Collective

collective {
  new : Arrival(0 : 2, 0 : 2);
  new Taxi(0 : 2, 0 : 2, 3, 3, 0, F);
}

Quantitative Analysis

The semantics of CARMA gives rise to a Continuous Time Markov Chain (CTMC).

This can be analysed by

- by numerical analysis of the CTMC for small systems;
- by stochastic simulation of the CTMC;
- by fluid approximation of the CTMC under certain restrictions (particularly on the environment).

Quantitative Analysis

The semantics of CARMA gives rise to a Continuous Time Markov Chain (CTMC).

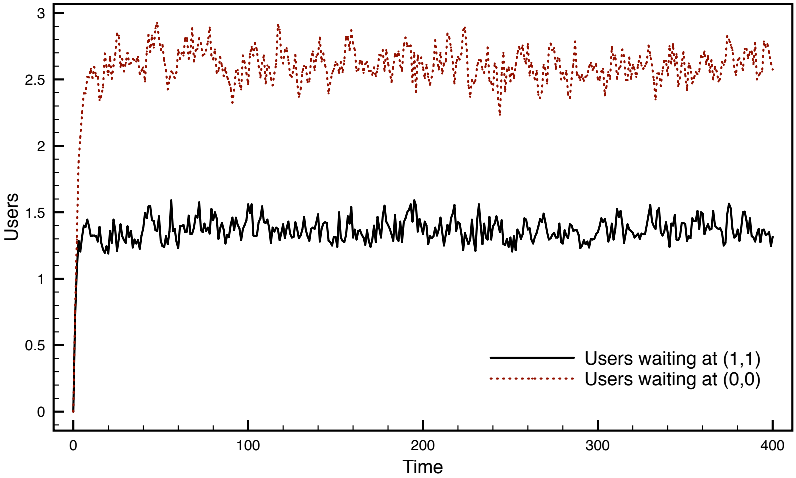
This can be analysed by

- by numerical analysis of the CTMC for small systems;
- by stochastic simulation of the CTMC;
- by fluid approximation of the CTMC under certain restrictions (particularly on the environment).

Here we show the results of stochastic simulation.

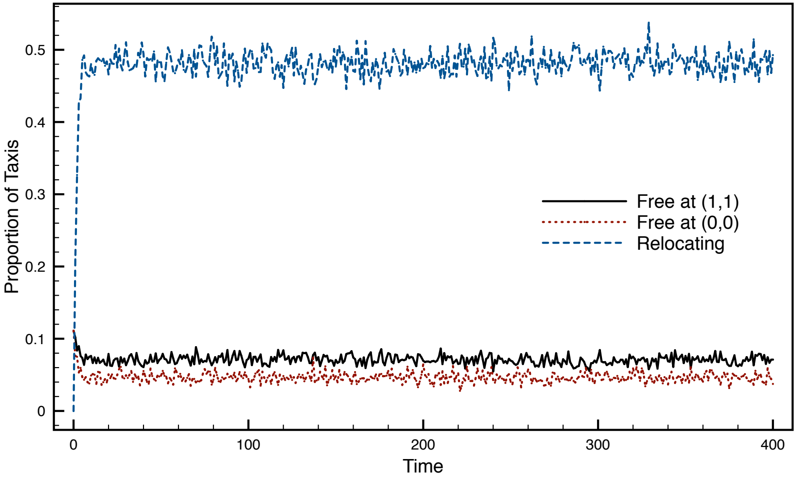
Scenario 1 results

Average number of users waiting at $(1,1)$ and $(0,0)$



Scenario 1 results

Proportion of free taxis at $(1,1)$ and $(0,0)$ and in transit

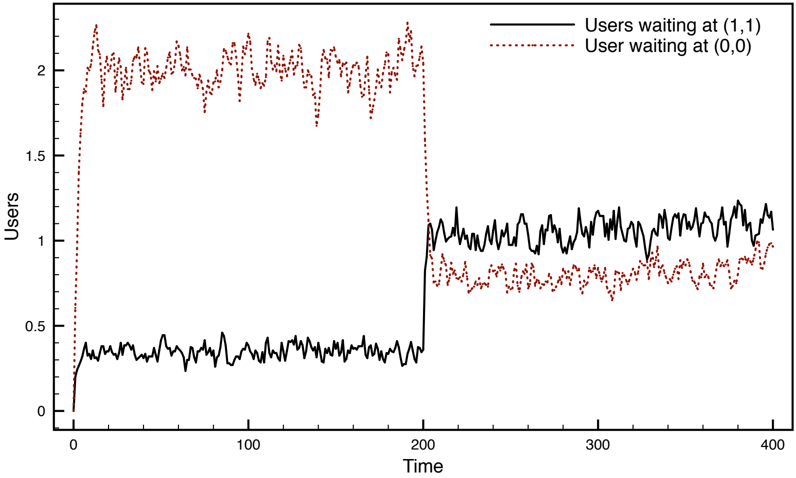


Comments: Scenario 1

- In Scenario 1 after an initial startup period, around 2.5 users are waiting for a taxi in the peripheral location while only 1.5 users are waiting for a taxi in location $(1, 1)$.
- In this scenario a larger fraction of users are delivered to location $(1, 1)$ so soon a larger fraction of taxis are available to collect users at the centre.
- A large fraction of taxis (around 50%) are continually moving between the different patches.

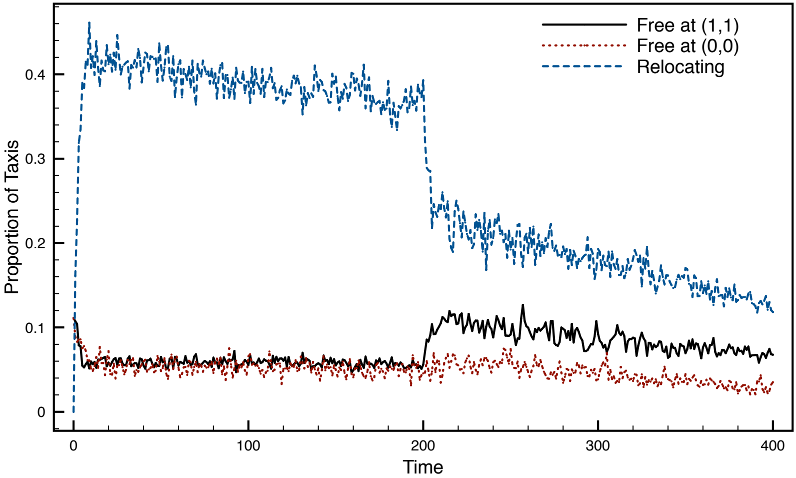
Scenario 2 results

Average number of users waiting at $(1,1)$ and $(0,0)$



Scenario 2 results

Proportion of free taxis at $(1, 1)$ and $(0, 0)$ and in transit



Comments: Scenario 2

- In Scenario 2 the location of new arrivals depends on the current time:
  - $[0, 200)$: 3/4 of users arrive on the border and only 1/4 in the centre;
  - $[200, 400)$: 1/4 of users arrive on the border and 3/4 in the centre.
- Results in the first phase are similar to Scenario 1.
- After time 200, the number of users waiting for a taxi in the border decreases below 1 whilst the average waiting for a taxi in the centre increases to just over 1 and the fraction of taxis continually moving is reduced to 20%.

Outline

Concluding remarks

- Collective Systems are an interesting and challenging class of systems to design and construct.
- Their role within infrastructure, such as within smart cities, make it essential that quantitive aspects of behaviour is taken into consideration, as well as functional correctness.
- The complexity of these systems poses challenges both for model construction and model analysis.
- CARMA aims to address many of these challenges, supporting rich forms of interaction, using attributes to capture explicit locations and the environment to allow adaptivity.
- Fluid approximation based analysis offers hope for scalable quantitative analysis techniques, but further work is needed to make this applicable to a wider class of CAS.

Thanks

Thanks to my collaborators and colleagues on the QUANTICOL project, especially Michele Loreti.

This work has been funded by the CEC through the ongoing FET-Proactive QUANTICOL project

quanticol

www.quanticol.eu