Quantitative Analysis of Collective Adaptive Systems

Jane Hillston
LFCS, University of Edinburgh

15th February 2016

Outline

1 Introduction
- Collective Adaptive Systems
- Quantitative Analysis

2 Modelling CAS
- Challenges for modelling CAS

3 CARMA
- The CARMA Modelling Language
- Smart Taxi System Example

4 Conclusions

Outline

1 Introduction
- Collective Adaptive Systems
- Quantitative Analysis

2 Modelling CAS
- Challenges for modelling CAS

3 CARMA
- The CARMA Modelling Language
- Smart Taxi System Example

4 Conclusions

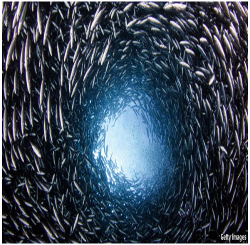
Collective Systems

We are surrounded by examples of collective systems:

Collective Systems

We are surrounded by examples of collective systems:

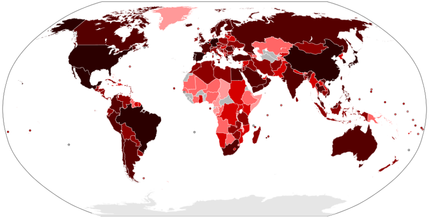
in the natural world ....

Collective Systems

We are surrounded by examples of collective systems:

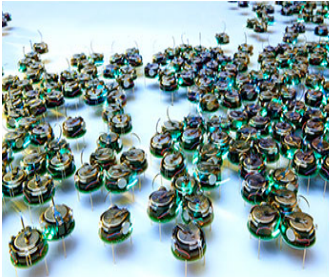
.... and in the man-made world



Collective Systems

We are surrounded by examples of collective systems:

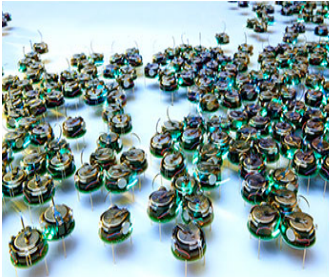
.... and in the man-made world



Collective Systems

We are surrounded by examples of collective systems:

.... and in the man-made world



Most of these systems are also adaptive to their environment

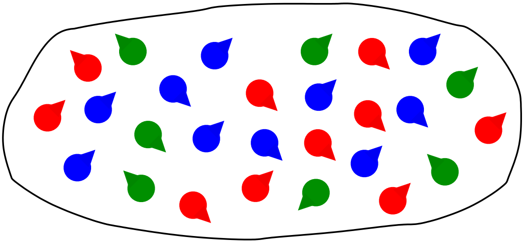
Collective Systems

We are surrounded by examples of collective systems:

an informatic 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.

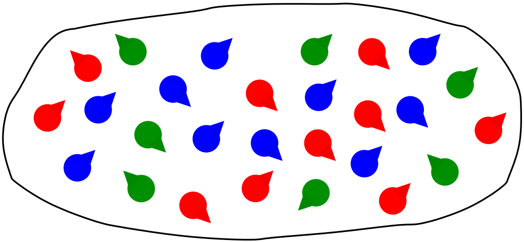


Each entity may have its own properties, objectives and actions.

At the system level these combine to create the collective behaviour.

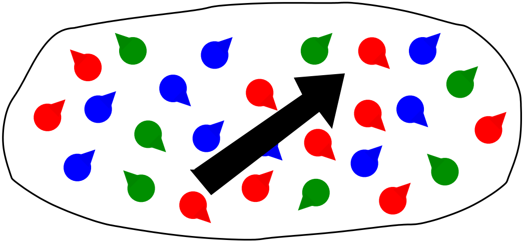
Collective Adaptive Systems

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



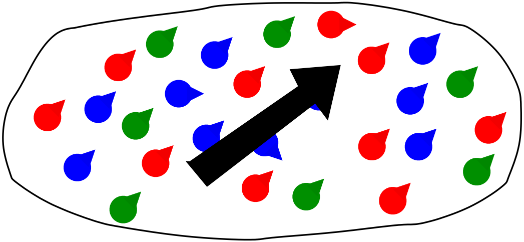
Collective Adaptive Systems

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



Collective Adaptive Systems

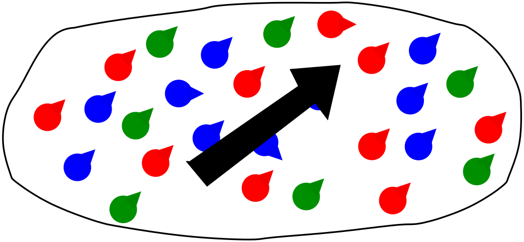
The behaviour of the system is thus dependent on the behaviour of the individual entities.



And the behaviour of the individuals will be influenced by the state of the overall system.

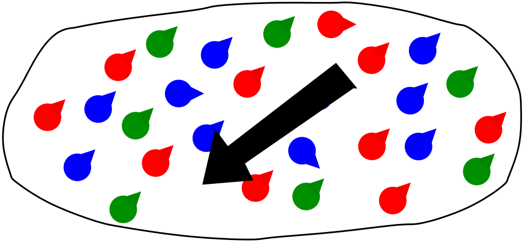
Collective Adaptive Systems

Such systems are often embedded in our environment and need to operate without centralised control or direction.



Collective Adaptive Systems

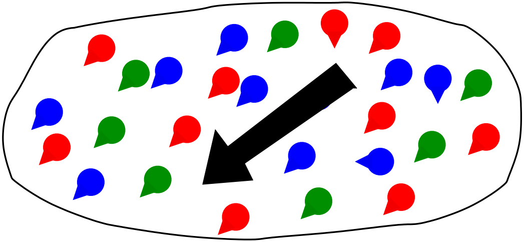
Such systems are often embedded in our environment and need to operate without centralised control or direction.



Moreover when conditions within the system change it may not be feasible to have human intervention to adjust behaviour appropriately.

Collective Adaptive Systems

Such systems are often embedded in our environment and need to operate without centralised control or direction.



Moreover when conditions within the system change it may not be feasible to have human intervention to adjust behaviour appropriately.

Thus systems must be able to autonomously adapt.

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.

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 are now becoming the reality, and many form collective adaptive systems, in which large numbers of computing elements collaborate to meet the human need.

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 are now becoming the reality, and many form collective adaptive systems, in which large numbers of computing elements collaborate to meet the human need.

For instance, may examples of such systems can be found in components of Smart Cities, such as smart urban transport and smart grid electricity generation and storage.

Quantitative Modelling

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

Quantitative Modelling

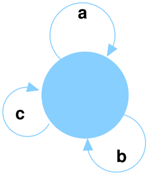
Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

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

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

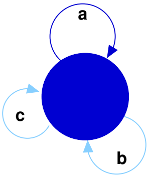
Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.



Quantitative Modelling

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

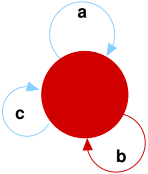
Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.



Quantitative Modelling

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

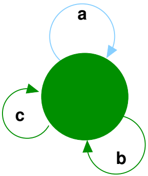
Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.



Quantitative Modelling

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

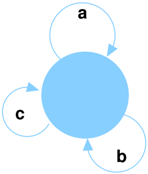
Markovian-based discrete event models have been applied to computer systems since the mid-1960s and communication systems since the early 20th century.



Quantitative Modelling

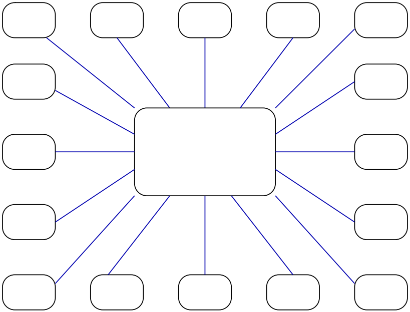
Performance modelling aims to construct models of the dynamic behaviour of systems in order to support the fair and efficient sharing of resources.

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



Various formalisms have been designed for capturing such behaviour.

Performance Modelling: Motivation



Capacity planning

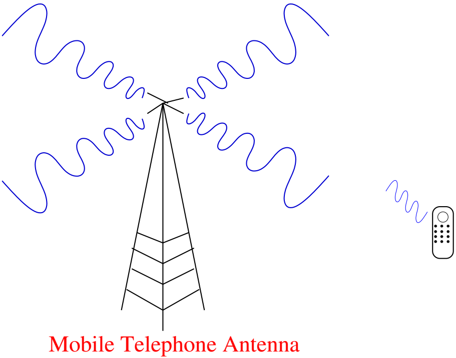
- How many clients can the existing server support and maintain reasonable response times?

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

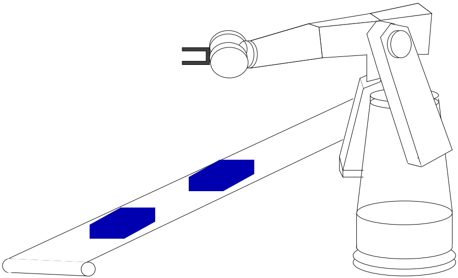
- How many frequencies do you need to keep blocking probabilities low?

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 speed of conveyor belt will minimize robot idle time and maximize throughput whilst avoiding lost widgets?

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.

Performance Modelling

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

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

Performance Modelling

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

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

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

Performance Modelling

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

For the last three decades there has been substantial interest in applying formal modelling techniques enhanced with information about timing and probability.

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

Primary examples include:

- Stochastic Petri Nets and
- Stochastic Process Algebras.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

SPA
MODEL

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

SPA MODEL $\xrightarrow{\text{SOS rules}}$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

$$\text{SPA MODEL} \xrightarrow{\text{SOS rules}} \text{LABELLED TRANSITION SYSTEM} \xrightarrow[\text{diagram}]{\text{state transition}}$$

Stochastic Process Algebra

- Models are constructed from components which engage in activities.
- Activities have a name and a rate.
- The rate defines an exponential distribution which means that the duration of an activity is a random variable.
- A small set of language constructs determine how the model will evolve.
- The language is used to generate a CTMC for performance modelling.

SPA MODEL $\xrightarrow{\text{SOS rules}}$ LABELLED TRANSITION SYSTEM $\xrightarrow[\text{diagram}]{\text{state transition}}$ CTMC **Q**

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

Integrated analysis

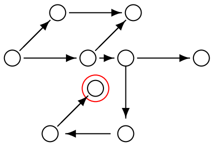
Qualitative verification can now be complemented by quantitative verification.

Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Reachability analysis

How long will it take for the system to arrive in a particular state?

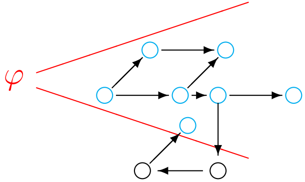


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

Does a given property φ hold within the system with a given probability?

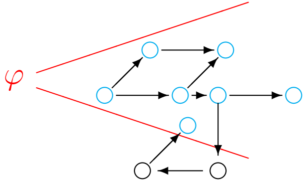


Integrated analysis

Qualitative verification can now be complemented by quantitative verification.

Model checking

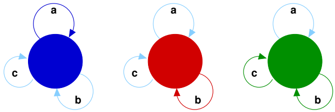
For a given starting state
how long is it until
a given property φ holds?



Outline

1. Introduction
   - Collective Adaptive Systems
   - Quantitative Analysis
2. Modelling CAS
   - Challenges for modelling CAS
3. CARMA
   - The CARMA Modelling Language
   - Smart Taxi System Example
4. Conclusions

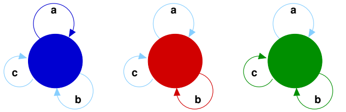
Solving discrete state models

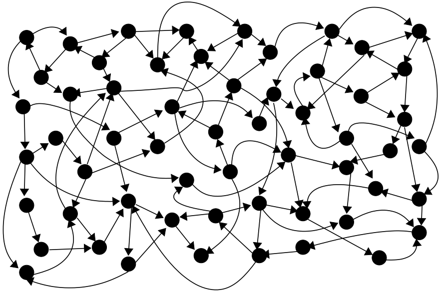


Under the SOS semantics a SPA model is mapped to a CTMC with global states determined by the local states of all the participating components.

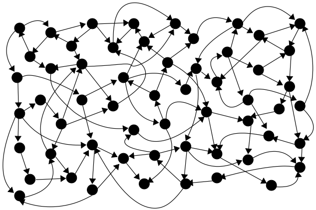
Solving discrete state models

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.



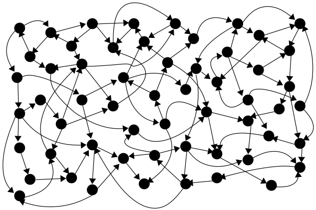


Solving discrete state models



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.

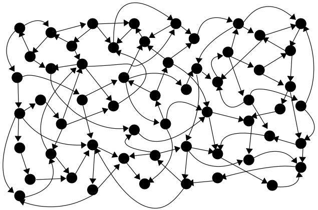
Solving discrete state models



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.

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

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.

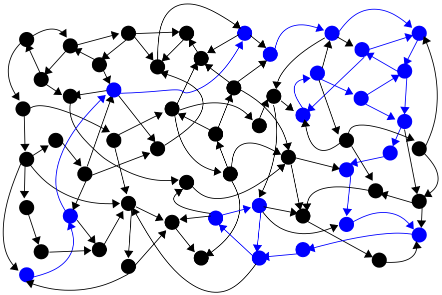
$$Q = \begin{pmatrix} q_{1,1} & q_{1,2} & \cdots & q_{1,N} \\ q_{2,1} & q_{2,2} & \cdots & q_{2,N} \\ \vdots & \vdots & & \vdots \\ q_{N,1} & q_{N,2} & \cdots & q_{N,N} \end{pmatrix}$$

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

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

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.

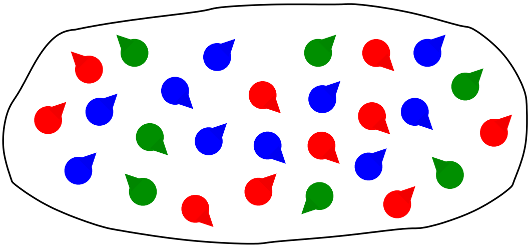


State space explosion

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.

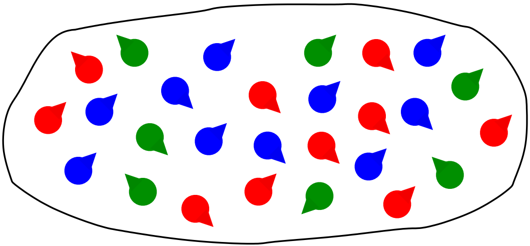
Modelling collective behaviour

A key feature of collective systems is the existence of populations of entities who share certain characteristics.



Modelling collective behaviour

A key feature of collective systems is the existence of populations of entities who share certain characteristics.



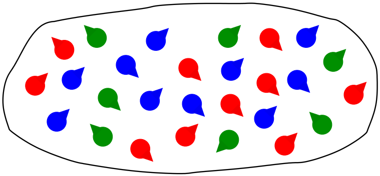
High-level modelling formalisms allow this repetition to be captured at the high-level rather than explicitly.

The Fluid Approximation Alternative

We can shift attention to the populations rather than the individual entities, and then consider the average behaviour within a population.

The Fluid Approximation Alternative

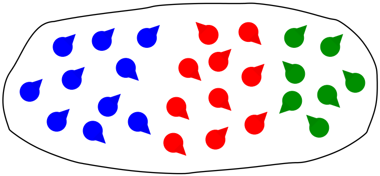
We can shift attention to the populations rather than the individual entities, and then consider the average behaviour within a population.



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

The Fluid Approximation Alternative

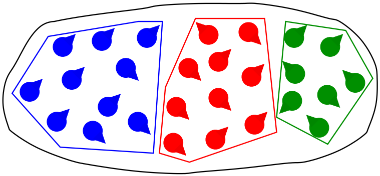
We can shift attention to the populations rather than the individual entities, and then consider the average behaviour within a population.



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

The Fluid Approximation Alternative

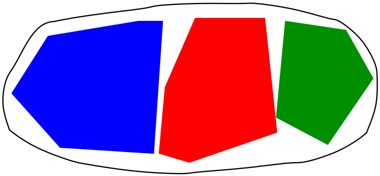
We can shift attention to the populations rather than the individual entities, and then consider the average behaviour within a population.



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

The Fluid Approximation Alternative

We can shift attention to the populations rather than the individual entities, 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 now disregard the identity of components.

Even better reductions can be achieved when we no longer regard the components as individuals.

Population models

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

Population models

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

To characterise the behaviour of a population we calculate the proportion of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

Population models

A shift in perspective allows us to model the interactions between individual components but then only consider the system as a whole as an interaction of populations.

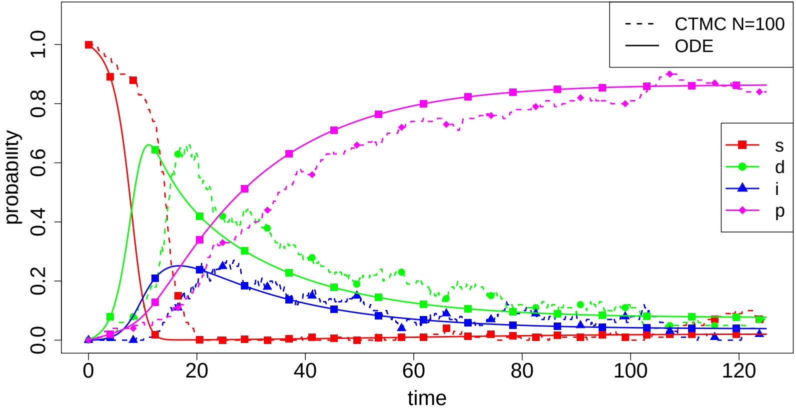
To characterise the behaviour of a population we calculate the proportion of individuals within the population that are exhibiting certain behaviours rather than tracking individuals directly.

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.

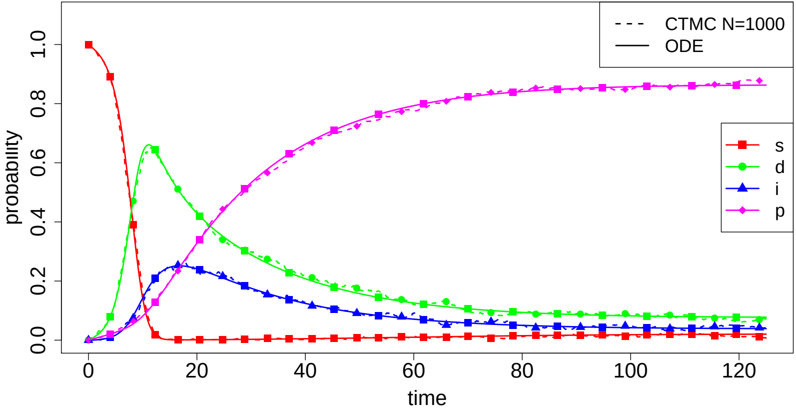
Illustrative trajectories

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 $N = 1000$



Example Applications

Large scale software systems

Issues of scalability are important for user satisfaction and resource efficiency but such issues are difficult to investigate using discrete state models.

Spread of viruses and malware

Improved modelling of networks under attack could lead to improved detection and better security in computer systems.

Example Applications

Biochemical signalling pathways

Understanding these pathways has the potential to improve the quality of life through enhanced drug treatment and better drug design.

Crowd dynamics

Technology enhancement is creating new possibilities for directing crowd movements in buildings and urban spaces, for example for emergency egress, which are not yet well-understood.

Challenges for modelling CAS

The work so far demonstrates provides 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

Richer forms of interaction

If we consider real collective adaptive systems, especially those with emergent behaviour, they embody rich forms of interaction, often based on asynchronous communication.

Richer forms of interaction

If we consider real collective adaptive systems, especially those with emergent behaviour, they embody rich forms of interaction, often based on asynchronous communication.

For example, pheromone trails left by social insects.

Richer forms of interaction

If we consider real collective adaptive systems, especially those with emergent behaviour, they embody rich forms of interaction, often based on asynchronous communication.

For example, pheromone trails left by social insects.

Languages like SCEL offer these richer communication patterns, with components which include a knowledge store which can be manipulated by other components and attribute-based communication.

R.De Nicola, G.Ferrari, M.Loreti, R.Pugliese. A Language-Based Approach to Autonomic Computing. FMCO 2011.

Richer forms of interaction

If we consider real collective adaptive systems, especially those with emergent behaviour, they embody rich forms of interaction, often based on asynchronous communication.

For example, pheromone trails left by social insects.

Languages like SCEL offer these richer communication patterns, with components which include a knowledge store which can be manipulated by other components and attribute-based communication.

R.De Nicola, G.Ferrari, M.Loreti, R.Pugliese. A Language-Based Approach to Autonomic Computing. FMCO 2011.

Developing scalable analysis techniques, such a fluid approximation, for such languages remains an open problem.

Modelling space

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

Modelling space

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

We can impose the effects of space by encoding it into the behaviour of the actions of components and distinguishing the same component in different location as distinct types, but this is modelling space implicitly.

Modelling space

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

We can impose the effects of space by encoding it into the behaviour of the actions of components and distinguishing the same component in different location as distinct types, but this is modelling space implicitly.

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

Modelling space

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

We can impose the effects of space by encoding it into the behaviour of the actions of components and distinguishing the same component in different location as distinct types, but this is modelling space implicitly.

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

Again this is difficult for scalable analysis which is often based on an implicit 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

1. Introduction
   - Collective Adaptive Systems
   - Quantitative Analysis
2. Modelling CAS
   - Challenges for modelling CAS
3. CARMA
   - The CARMA Modelling Language
   - Smart Taxi System Example
4. Conclusions

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

This includes the language, CARMA (Collective Adaptive Resource-sharing Markovian Agents), which handles:

1. The behaviours of agents and their interactions;

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

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

This includes the language, CARMA (Collective Adaptive Resource-sharing Markovian Agents), which handles:

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

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

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

This includes the language, CARMA (Collective Adaptive Resource-sharing Markovian Agents), 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...

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

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

This includes the language, CARMA (Collective Adaptive Resource-sharing Markovian Agents), 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;

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

A new language for CAS

The QUANTICOL project seeks to develop a coherent, integrated set of linguistic primitives, methods and tools to build systems that can operate in open-ended, unpredictable environments.

This includes the language, CARMA (Collective Adaptive Resource-sharing Markovian Agents), 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.

Interaction patterns in CAS

Typically, CAS exhibit two kinds of interaction pattern:

Interaction patterns in CAS

Typically, CAS exhibit two kinds of interaction pattern:

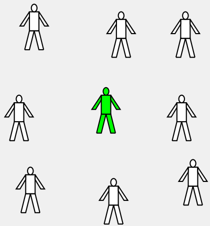
1. **Spreading**: one agent spreads relevant information to a given group of other agents

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

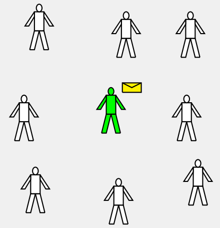


Interaction patterns in CAS

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

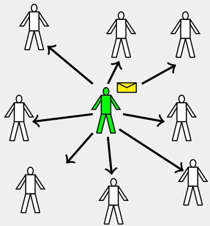


Interaction patterns in CAS

Typically, CAS exhibit two kinds of interaction pattern:

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

Spreading: 1-to-many

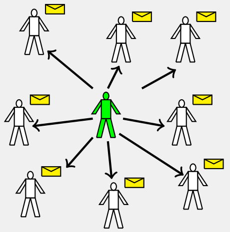


Interaction patterns in CAS

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

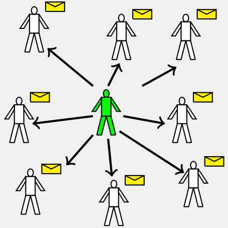


Interaction patterns in CAS

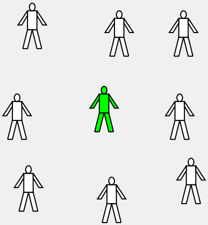
Typically, CAS exhibit two kinds of interaction pattern:

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

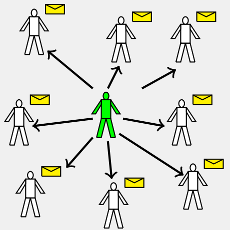


Interaction patterns in CAS

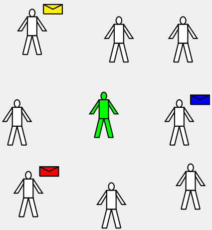
Typically, CAS exhibit two kinds of interaction pattern:

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

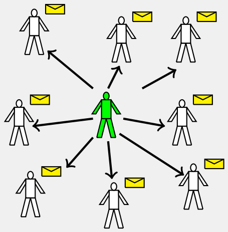


Interaction patterns in CAS

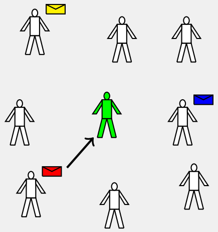
Typically, CAS exhibit two kinds of interaction pattern:

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

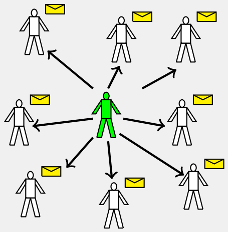


Interaction patterns in CAS

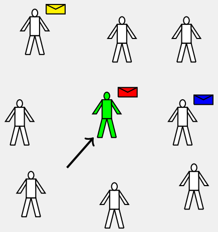
Typically, CAS exhibit two kinds of interaction pattern:

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

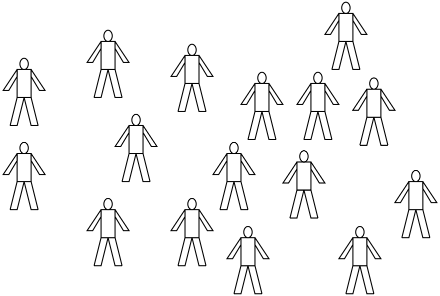


Collecting: 1-to-1



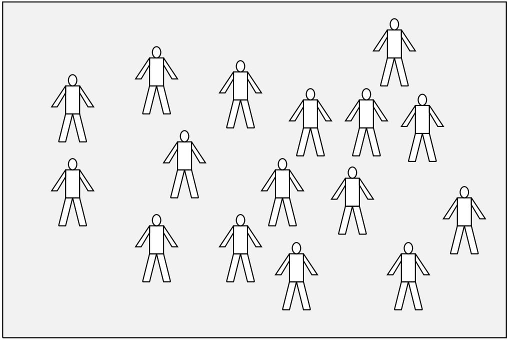
CAS: CARMA perspective

Collective



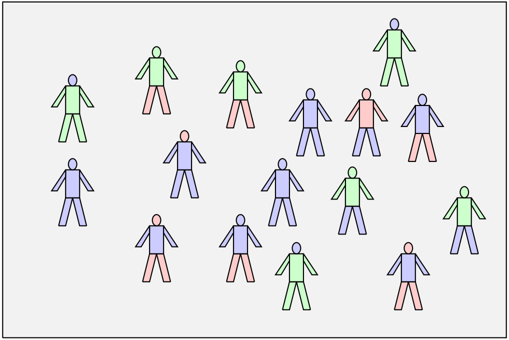
CAS: CARMA perspective

Collective **Environment**



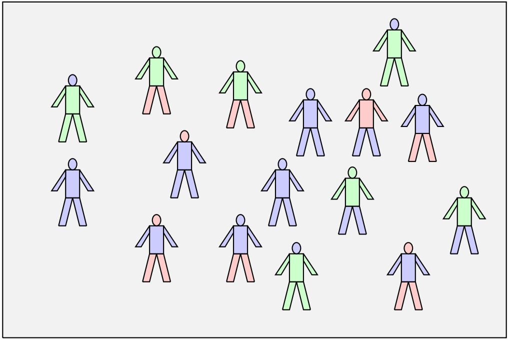
CAS: CARMA perspective

Collective **Environment** **Attributes**



CAS: CARMA perspective

Collective **Environment** **Attributes**



Processes are referenced via their attributes!

Collective Adaptive Resource-sharing Markovian Agents

A CARMA system consists of

Collective Adaptive Resource-sharing Markovian Agents

A CARMA system consists of

- a collective (N)...

Collective Adaptive Resource-sharing Markovian Agents

A CARMA system consists of

- a collective (N). . .
- . . . operating in an environment ($\mathcal{E}$).

Collective Adaptive Resource-sharing Markovian Agents

A CARMA system consists of

- a collective (N)...
- ...operating in an environment ($\mathcal{E}$).

Collective...

- is composed by a set of components, i.e. the Markovian agents that concur and cooperate to achieve a set of given tasks
- models the behavioural part of a system

Collective Adaptive Resource-sharing Markovian Agents

A CARMA system consists of

- a collective (N)...
- ...operating in an environment ($\mathcal{E}$).

Collective...

- is composed by a set of components, i.e. the Markovian agents that concur and cooperate to achieve a set of given tasks
- models the behavioural part of a system

Environment...

- models the rules intrinsic to the context where agents operate;
- mediates and regulates agent interactions.

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

Interaction primitives

Processes interact via attribute based communications. . .

Interaction primitives

Processes interact via attribute based communications. . .

- **Broadcast output**: a message is sent to all the components satisfying a predicate π;

Interaction primitives

Processes interact via attribute based communications. . .

- **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

Processes interact via attribute based communications. . .

- **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;

Interaction primitives

Processes interact via attribute based communications...

- **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

Processes interact via attribute based communications...

- **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

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

Interaction primitives
Syntax

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

- α is an action type;

Interaction primitives
Syntax

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

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

Interaction primitives
Syntax

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

- 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:

- σ denotes a function mapping each γ to a probability distribution over possible stores.

Updating the store

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

- σ denotes a function mapping each γ to a probability distribution over possible stores.

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

Updating the store

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

- σ denotes a function mapping each γ to a probability distribution over possible stores.

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

Remark:

- Processes running in the same component can implicitly interact via the local store;
- Updates are instantaneous.

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:

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

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:

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

Pattern matching can be encoded in CARMA.

Examples of interactions...

Broadcast synchronisation:

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

Examples of interactions...

Broadcast synchronisation:

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

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:

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

$$\Downarrow$$

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

Examples of interactions...

Broadcast synchronisation:

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

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:

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

$$\Downarrow$$

$$
\begin{array}{l}
(P, \sigma_1(\{\mathit{role} = \text{“}\mathit{master}\text{''}\})) \parallel \\
\quad (\mathrm{stop}^{\star}[\mathrm{role} = \text{“}\mathit{master}\text{''}](x)\sigma_2.Q_1, \{\mathrm{bl} = 45\%\}) \parallel \\
\qquad (\mathrm{stop}^{\star}[\mathrm{role} = \text{“}\mathit{super}\text{''}](x)\sigma_3.Q_2, \{\mathrm{bl} = 2\%\}) \parallel \\
\qquad\qquad (\mathrm{stop}^{\star}[\top](x)\sigma_4.Q_3, \{\mathrm{bl} = 25\%\})
\end{array}
$$

Examples of interactions...

Unicast synchronisation:

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

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:

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

Examples of interactions...

Unicast synchronisation:

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

$$\Downarrow$$

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

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;
- ...

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;

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):

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;

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

Both kinds of component use the local store to publish the relevant data that will be used to represent the state of the agent.

Taxis and Users: stores

Both kinds of component use the local store to publish 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.

Taxis and Users: stores

Both kinds of component use the local store to publish 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

$$
\begin{array}{l}
\text{process } \mathit{User} = \\
\qquad \mathit{Wait} : \text{call}^{\star}[\top]\langle \text{my.loc.}x, \text{my.loc.}y\rangle.\mathit{Wait} \\
\qquad\qquad + \\
\qquad\qquad \text{take}[\text{loc.}x == \text{my.loc.}x \;\wedge\; \text{loc.}y == \text{my.loc.}y] \\
\qquad\qquad\qquad\qquad \langle \text{my.dest.}x, \text{my.dest.}y\rangle.\mathbf{kill} \\
\text{endprocess}
\end{array}
$$

Taxi processes

Taxis

$$
\begin{array}{l}
\text{process } \mathit{Taxi} = \\
\qquad F : \text{call}^{\star}[(\text{my.loc.x} \neq \text{posx}) \;\wedge\; \text{my.loc.y} \neq \text{posy}](\mathit{posx}, \mathit{posy}) \\
\qquad\qquad\qquad\qquad \{\text{dest} := [x := \mathit{posx}, y := \mathit{posy}]\}.G \\
\qquad\quad + \\
\qquad\quad \text{take}[\top](\mathit{posx}, \mathit{posy}) \\
\qquad\qquad\qquad \{\text{dest} := [x := \mathit{posx}, y := \mathit{posy}], \text{occupancy} := 1\}.G \\
\qquad G : \text{move}^{\star}[\bot]\langle\circ\rangle \\
\qquad\qquad \{\text{loc} := \text{dest}, \text{dest} := [x := 3, y := 3], \text{occupancy} := 0\}.F \\
\text{endprocess}
\end{array}
$$

Modelling arrivals

The Arrivals process has a single attribute loc.

Arrivals process for users

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

Modelling arrivals

The Arrivals process has a single attribute loc.

Arrivals process for users

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

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

Modelling arrivals

The Arrivals process has a single attribute loc.

Arrivals process for users

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

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

The precise role of this process will be clear when the environment is described.

The Environment

The environment consists of a global store and an evolution rule, and provides the context in which the components operate.

The Environment

The environment consists of a global store and an evolution rule, and provides the context in which the components operate.

In this model, the global store just contains constants related to the rates of the key actions in the system, take and call.

The Environment

The environment consists of a global store and an evolution rule, and provides the context in which the components operate.

In this model, the global store just contains constants related to the rates of the key actions in the system, take and call.

The evolution rule consist of three functions:

μ_p: determines the probability of an action, capturing how the current state of the system influences the communication within the system;

The Environment

The environment consists of a global store and an evolution rule, and provides the context in which the components operate.

In this model, the global store just contains constants related to the rates of the key actions in the system, take and call.

The evolution rule consist of three functions:

- μ_p: determines the probability of an action, capturing how the current state of the system influences the communication within the system;
- μ_r: defines the rates of actions in the system; again this may depend on the current state of the system, capturing adaptivity;

The Environment

The environment consists of a global store and an evolution rule, and provides the context in which the components operate.

In this model, the global store just contains constants related to the rates of the key actions in the system, take and call.

The evolution rule consist of three functions:

- μ_p: determines the probability of an action, capturing how the current state of the system influences the communication within the system;
- μ_r: defines the rates of actions in the system; again this may depend on the current state of the system, capturing adaptivity;
- μ_u: allows the global store and/or the collective to be updated after an action, again capturing adaptivity.

Evolution rule: μ_p

Defining the probabilities of actions

$$
\begin{array}{l}
\mathrm{prob}\{ \\
\quad \top, \mathrm{take} : \mathit{Takeprob}(\mathit{real}(\#\{\mathit{Taxi}[F] \mid \\
\qquad\qquad (\mathrm{my.loc.x} == \mathrm{sender.loc.x}) \wedge \\
\qquad\qquad (\mathrm{my.loc.y} == \mathrm{sender.loc.y})\})); \\
\quad \top, \mathrm{call}^\star : \mathrm{global.p_{lost}} \\
\quad \mathrm{default}\ \ 1 \\
\}
\end{array}
$$

- Each taxi receives a user request (take) with a probability that depends on the number of taxis in the patch.
- call* can be missed with a probability p_{lost} defined in the global store.
- All the other interactions occur with probability 1.

Evolution rule: μ_r

Defining the rates of actions

$$
\begin{array}{l}
\mathsf{rate}\{ \\
\quad \top, \mathsf{take} : \mathsf{global.r_t} \\
\quad \top, \mathsf{call}^\star : \mathsf{global.r_c} \\
\quad \top, \mathsf{move}^\star : \mathit{Mtime}(\mathsf{now}, \mathsf{sender.loc}, \mathsf{sender.dest}, 6) \\
\quad \top, \mathsf{arrival}^\star : \mathit{Atime}(\mathsf{now}, \mathsf{sender.loc}, 1) \\
\quad \mathsf{default}\ \ 0 \\
\}
\end{array}
$$

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

$$
\begin{array}{l}
\text{update}\{ \\
\qquad \top, \text{arrival}^\star : \textbf{new } \text{User}(\text{sender.loc}, \text{DestLoc}(\text{now}, \text{sender.loc}), \text{Wait}) \\
\}
\end{array}
$$

Measures

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

Measures

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

The number of waiting users at a location

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

Measures

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

The number of waiting users at a location

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

The number of taxis relocating

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

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.

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).

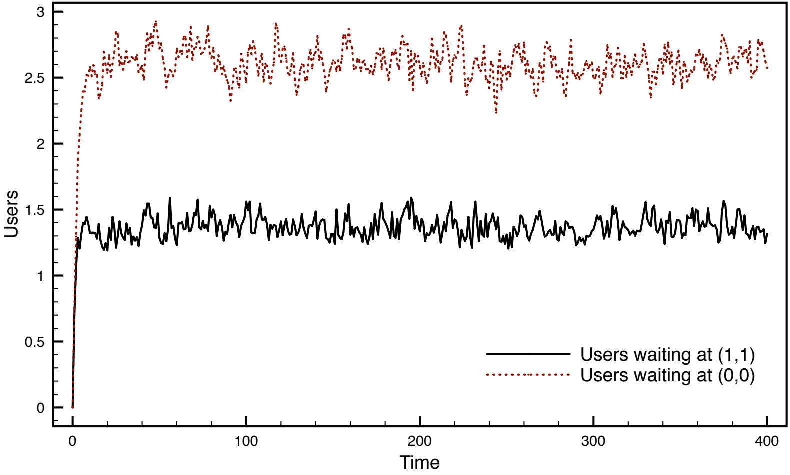
This can be analysed by

- 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.

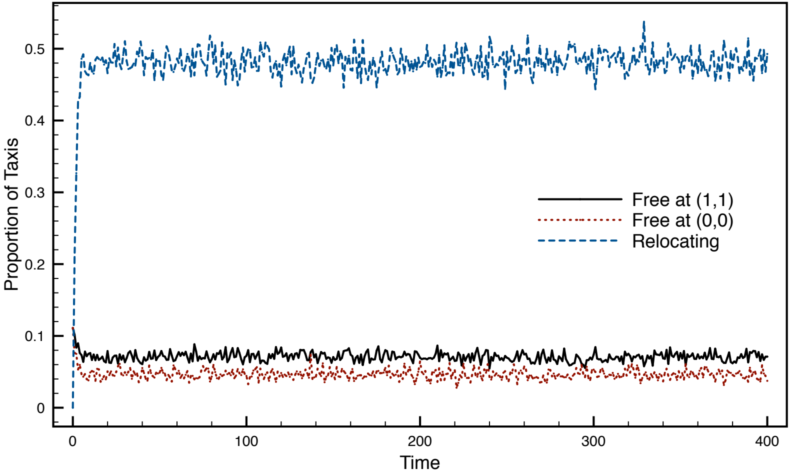
Scenario 1 results

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

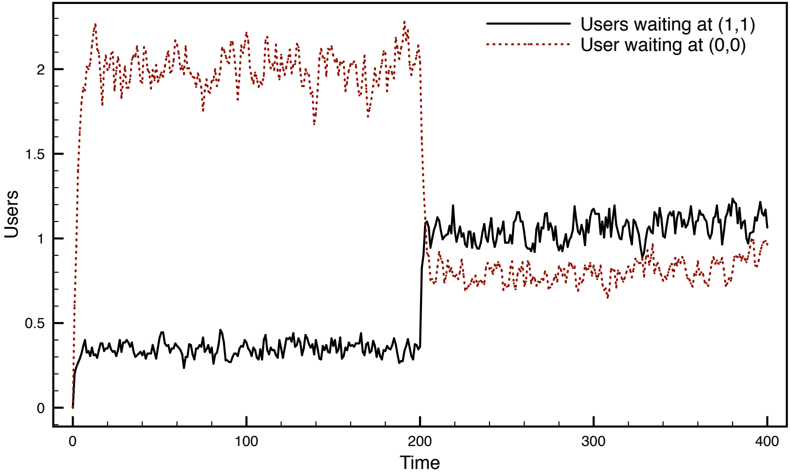


Comments: Scenario 1

- 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.

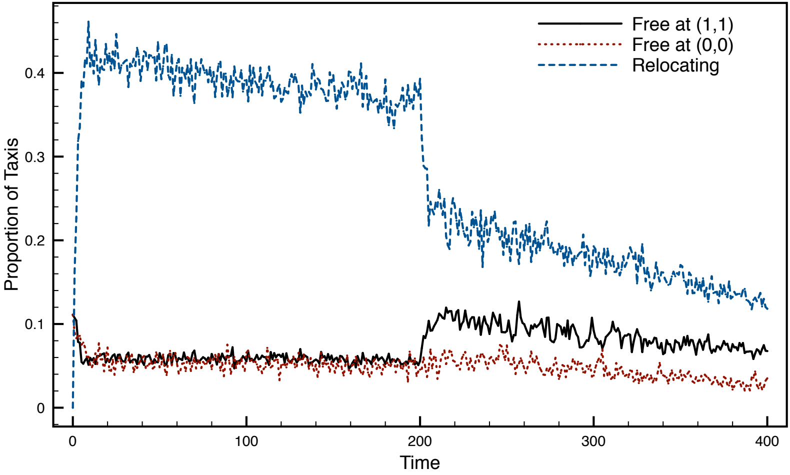
Scenario 2 results

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

1 Introduction
- Collective Adaptive Systems
- Quantitative Analysis

2 Modelling CAS
- Challenges for modelling CAS

3 CARMA
- The CARMA Modelling Language
- Smart Taxi System Example

4 Conclusions

Concluding remarks

- Collective Systems are an interesting and challenging class of systems to design and construct.

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.

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.

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.

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

Thanks to my collaborators and colleagues on the QUANTICOL project, especially Michele Loreti, Daokun Jiang and Christopher Williams.

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

quanticol

www.quanticol.eu