

Towards behaviour verification in self-organising (growing) artificial systems

Leo Motus, Merik Meriste and Jurgo Preden
Research Laboratory for Proactive Technologies
CENS, TUT DCC

Growth of a system may come out as change in

- number of components in the system, and/or
- topology of connections between components
- interactions between components , and/or
- behaviour of components, and/or
- behaviour of overall system

Growing systems possess cognitive power, autonomy of decisions, and often behave proactively

Growth is influenced by:

- intrinsic health of a system,
- properties of environment(s),
- goal-functions of a system

Growth and its control

Growth – increase in some physical or abstract quantity, such as:

- a system becoming more complex
- an organism becoming more mature
- an artificial system developing new functionality
- a tumour evolving from benign to pre-malignant

Control means checking system's current performance against pre-determined requirements and expectations, and actions to ensure adequate progress and satisfactory performance.

Control implicitly insists on (on-line) verification that the system functions in conformity with its prefixed specifications

Difficulty with growing systems -- prefixed specifications evolve in time, and due to changing of internal and external situations

Ability to control the growth assumes:

Verification:

- On-line monitoring the health of the system and its environment,
- (Preferably formal) assessment of the monitoring results w.r.t. expected health condition specifications, if necessary invoke healing and/or control procedures

Control proper:

- Detection of discrepancies between the actual performance of the systems and its current goals
- Eliminate/minimise the detected discrepancies with minimal cost by changing the integration rules and parameters, and/or the behaviour of components

Physical models of components in growing systems (1)

sensing health parameters in devices and the environment

Crossbow MicaZ

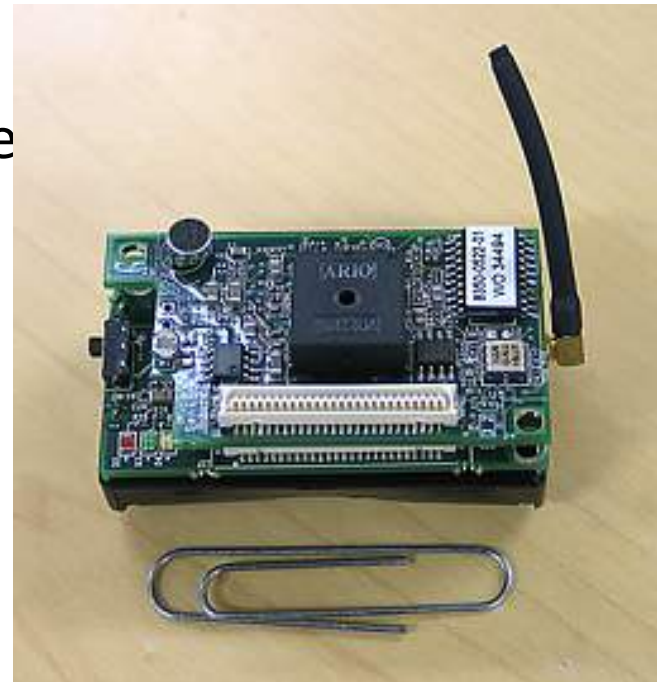
Radio: 2.4 GHz, IEEE802.15.4 compatible

Processor: 8bit Atmel Atmega 128L

ROM: 128 kB

Power 20 μ A/20mA

Sensors: magnetic field (2d),
acceleration (3d),
light, temperature,
microphone



Physical models of components in growing systems (2)

assessing the behaviour of software-intensive devices

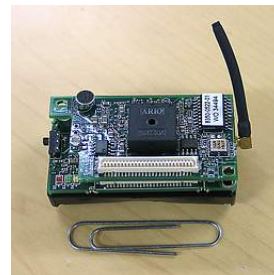
Gumstix Verdex XL6P

Radio: 2.4 GHz, IEEE 802.11

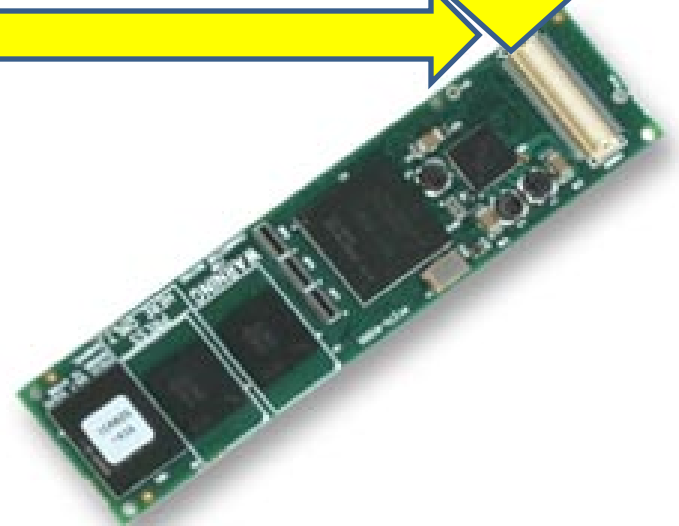
Processor: 600 MHz

ROM: 32MB

Power: 500mA (radio on)

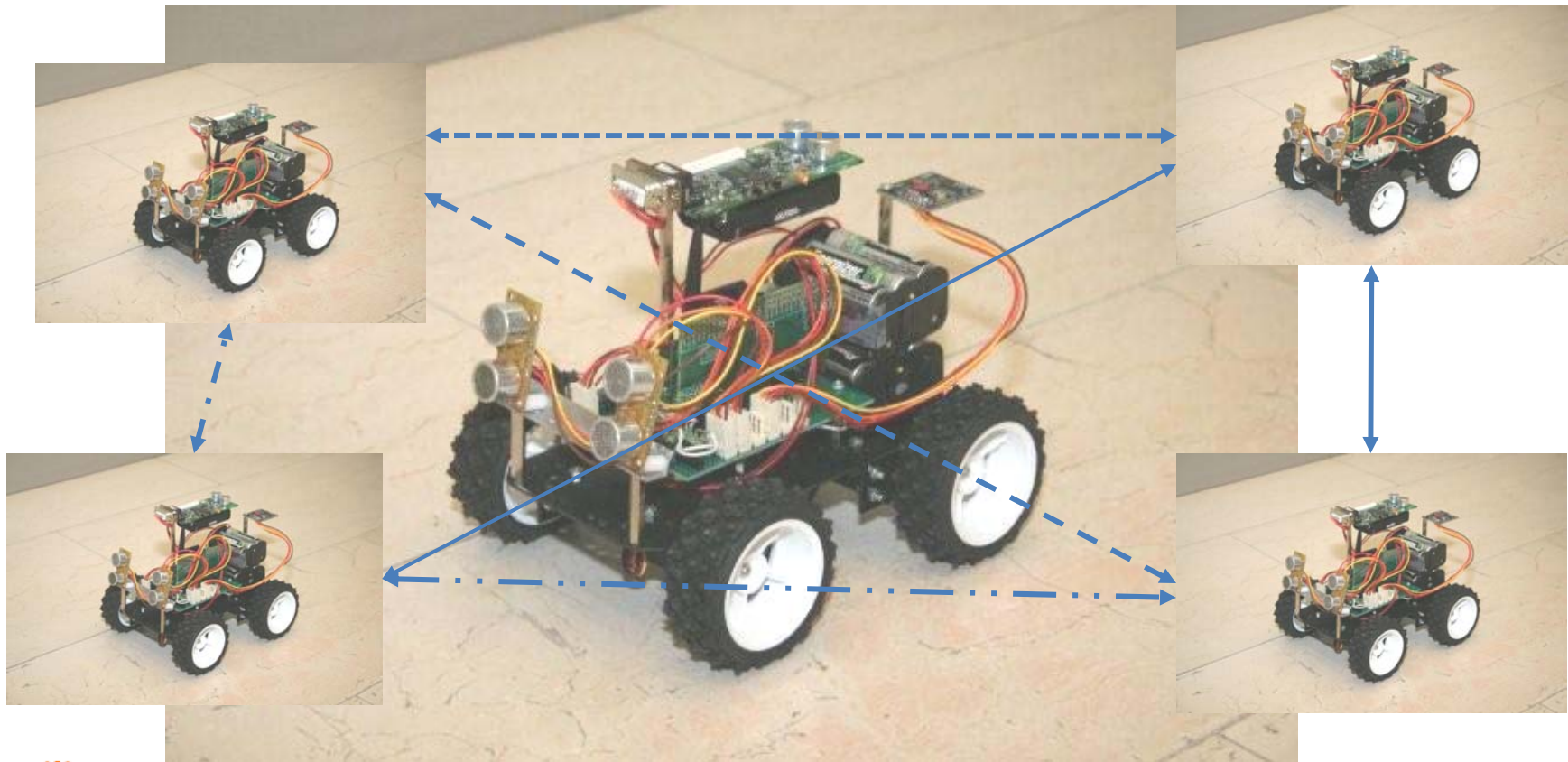


verdex XL6P

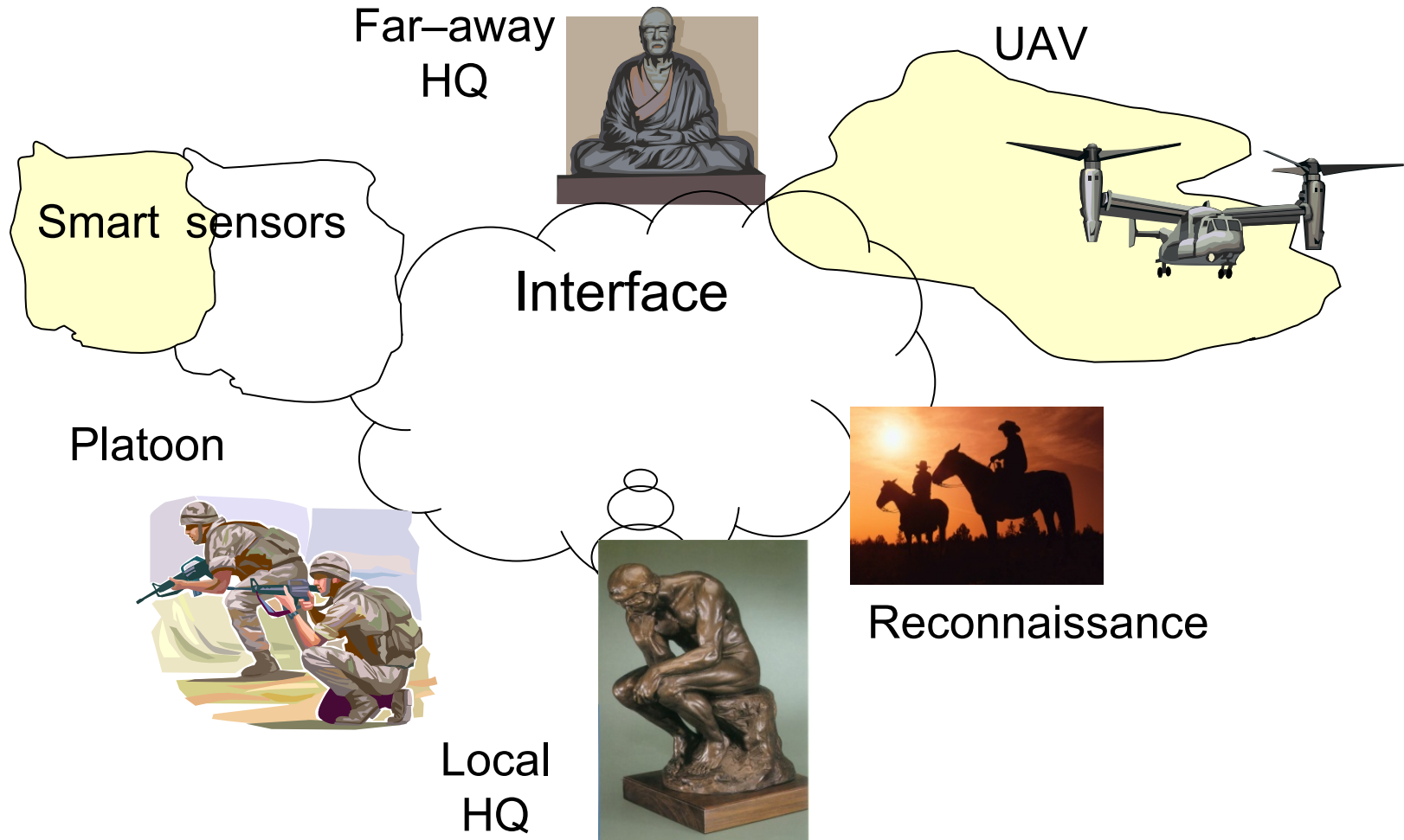


Physical models of components in growing systems (3)

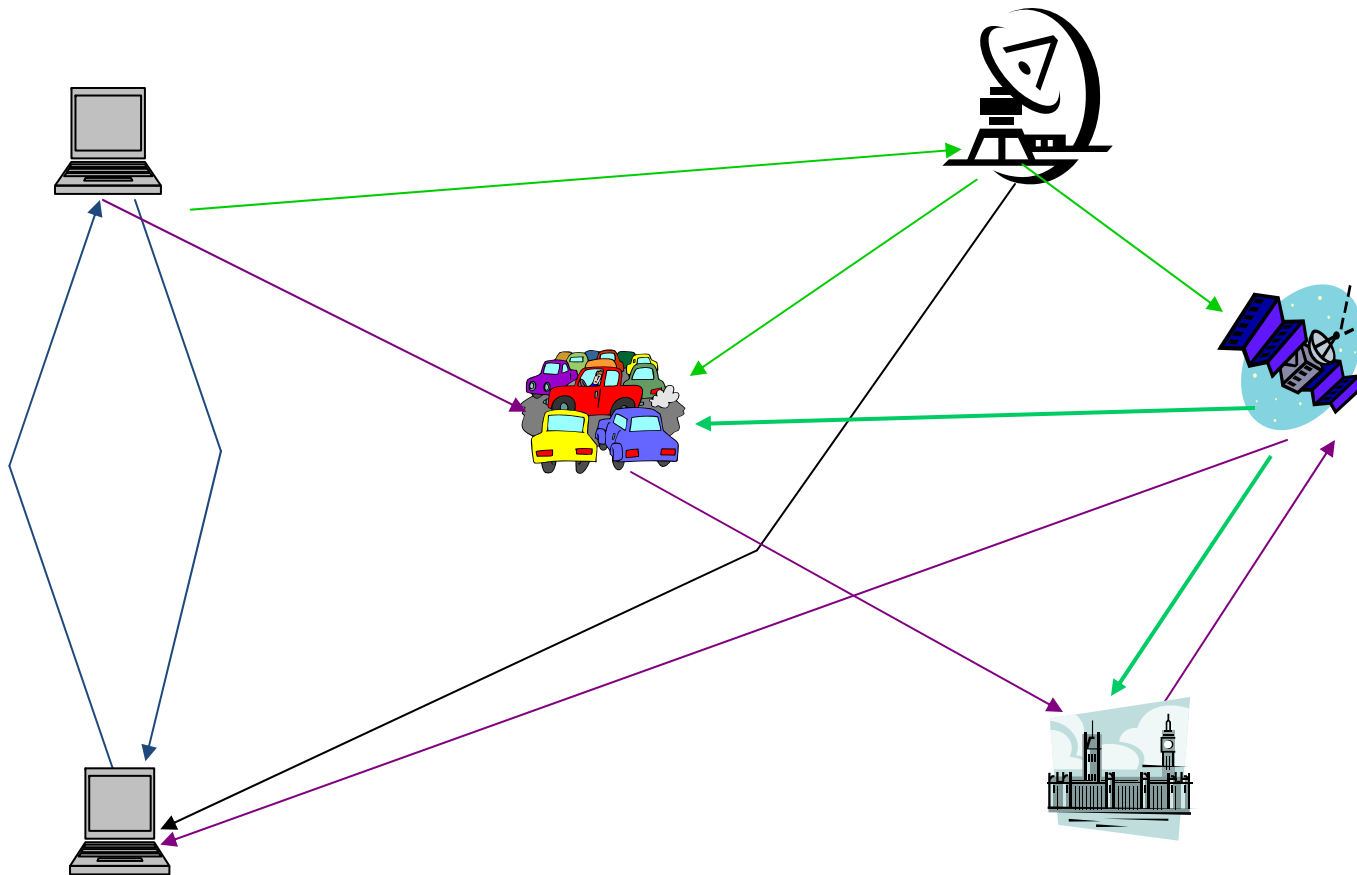
A proactive autonomous self-navigating device. Model of a growing system is an ad hoc network of such devices.



A generic example of a cooperative problem solving



Abstraction of the generic example (ad hoc network)



Simultaneous, interactive, time-aware computing in an ad hoc network

In terms of the Q-model (a prototype of a multi-stream interaction machine)

consists of processes and channels

process is a mapping

$$p: T(p) \times \text{dom } p \rightarrow \text{val } p$$

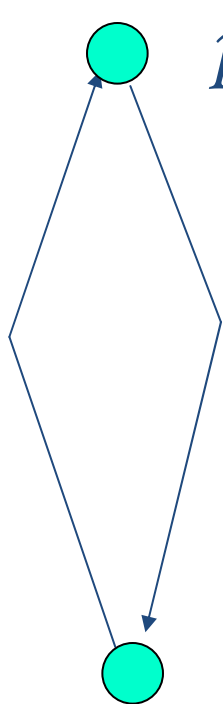
channel is a mapping

$$\sigma_{ij}: T(p_i) \times T(p_j) \times \text{val } p_i \rightarrow \text{proj}_{\text{val } p_i} \text{ dom } p_j$$

channel function is

$$K(\sigma_{ij}, t) \subset T(p_i), \quad t \in T(p_j)$$

Description of interactive, time-aware computation in an *ad hoc* network



$$p_i: \mathcal{T}(p_i) \times \text{dom } p_i \rightarrow \text{val } p_i$$

$$\sigma_{ij}: \text{val } p_i \times \mathcal{T}(p_i) \times \mathcal{T}(p_j) \rightarrow \text{proj}_{\text{val } p_i} \text{dom } p_j$$

$$\mathcal{K}(\sigma_{ij}, t) \subset \mathcal{T}(p_i), t \in \mathcal{T}(p_j)$$

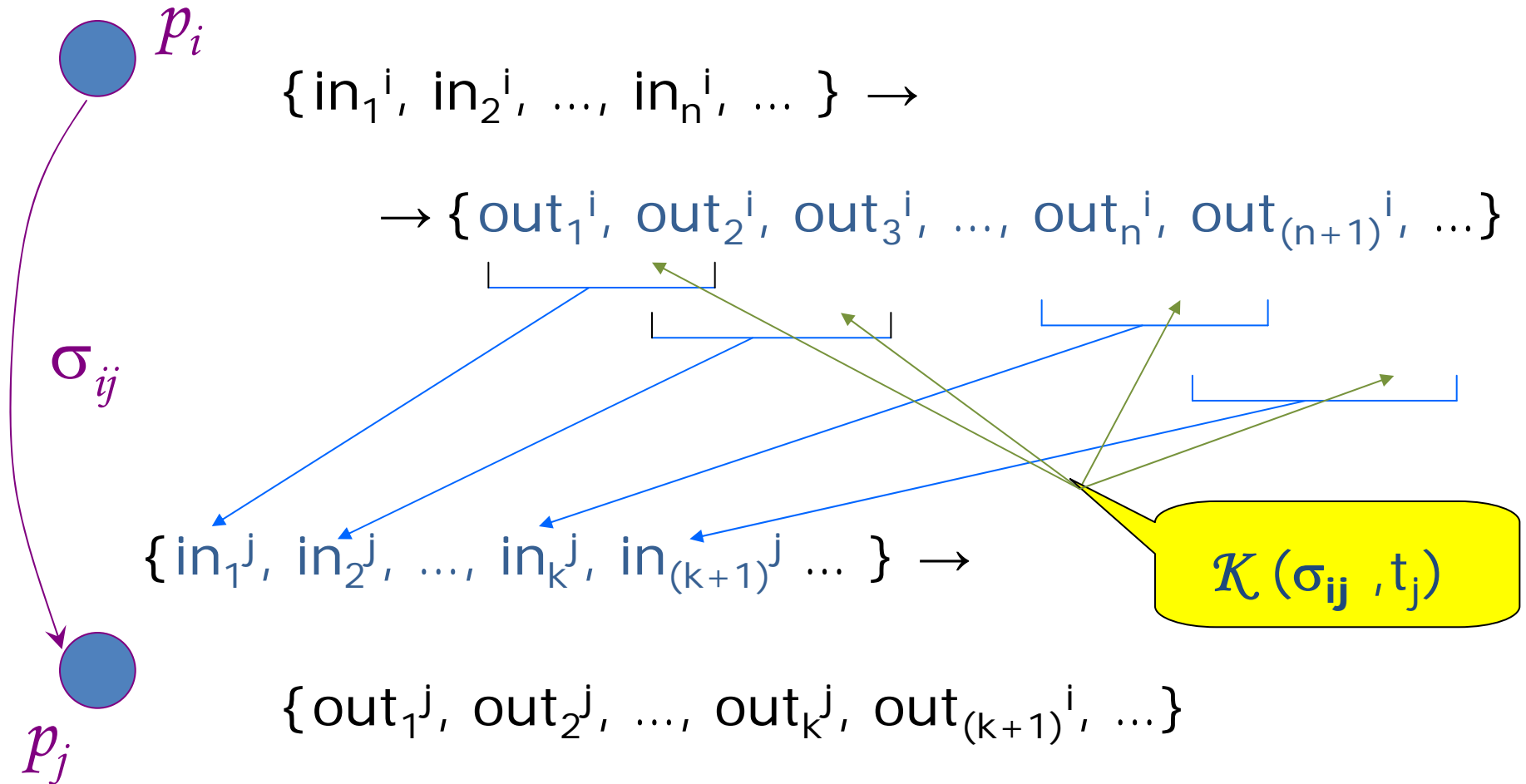
The definition of $\mathcal{K}(\sigma_{ij}, t)$ depends on whether

$$p_j: \mathcal{T}(p_j) \times \text{dom } p_j \rightarrow \text{val } p_j \text{ or}$$

$$\mathcal{T}(p_i) \rightarrow \mathcal{T}(p_j), \text{ or}$$

$$\mathcal{T}(p_i) = \{t_i^0, t_i^1, t_i^2, \dots, t_i^n, t_i^{(n+1)}, \dots\}$$

Interaction of nodes – a stream-based view



On-line verification of timing constraints in the prototype model

The case of synchronous channels:



Sample only

1. Communication via a synchronous channel will not violate the specified timing parameters of the consumer-process iff

$$\beta(p_i) < \gamma(\sigma_{ij}) + v t_{\min}(p_i)$$

2. A relaxed form of the previous proposition is

$$\beta(p_i) < \gamma(\sigma_{ij}) + v t_{\min}(p_i) + [t_{\min}(p_j) - \beta(p_j)]$$

3. A process consuming its own previous state, should never wait for this state

On-line detection of informational deadlock in growing systems

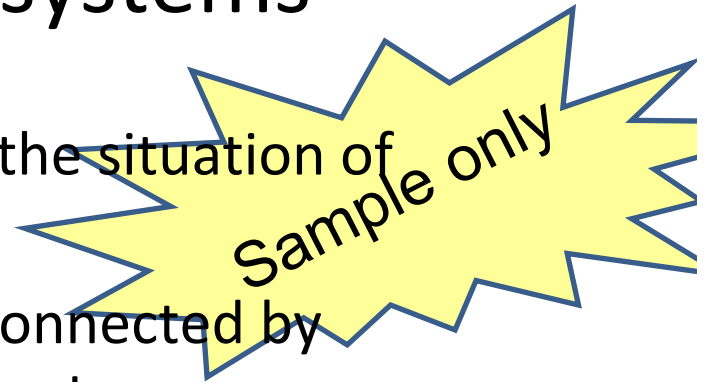
Informational deadlock in the Q-model is the situation of circular wait for messages

Synchronous cluster -- a set of processes connected by synchronous and synchronous null channels

Synchronous chain -- a sequence of common processes connected by synchronous channels

Synchronous loop -- a synchronous chain where $p_1 = p_n$ and the channel function for all channels in the loop has the form $\mathbf{K}(\sigma_{ij}, \mathbf{t}) = [\mu, 0]$

Synchronous loop is a sufficient condition for informational deadlock.



From time-aware computation to situation-aware computation (1)

- Time-awareness is sufficient for conventional real-time embedded systems
- The cases of networked embedded systems, and self-organising (growing) systems, need also situation-awareness
- Each monitored change in situations, and in the structure of system leads to verification of operational constraints and requirements against actual values; and may lead to correcting actions
- On-line monitoring and control of system's (and its environment's) behaviour is enabled by insertion of mediated interactions

From time-aware computation to situation-aware computation (2)

A situation is the aggregate of biological, socio-cultural, and environmental factors influencing on a group of agents and conditioning its behavioural pattern.

Roughly, time-aware process

$$p_i: T(p_i) \times \text{dom } p_i \rightarrow \text{val } p_i$$

is to be upgraded to a situation-aware process

$$p_i: T(p_i) \times S(p_i) \times \text{dom } p_i \rightarrow \text{val } p_i$$

Where $S(p_i)$ is a set of situational parameter values – e.g.

location, environmental health, current goal function, etc

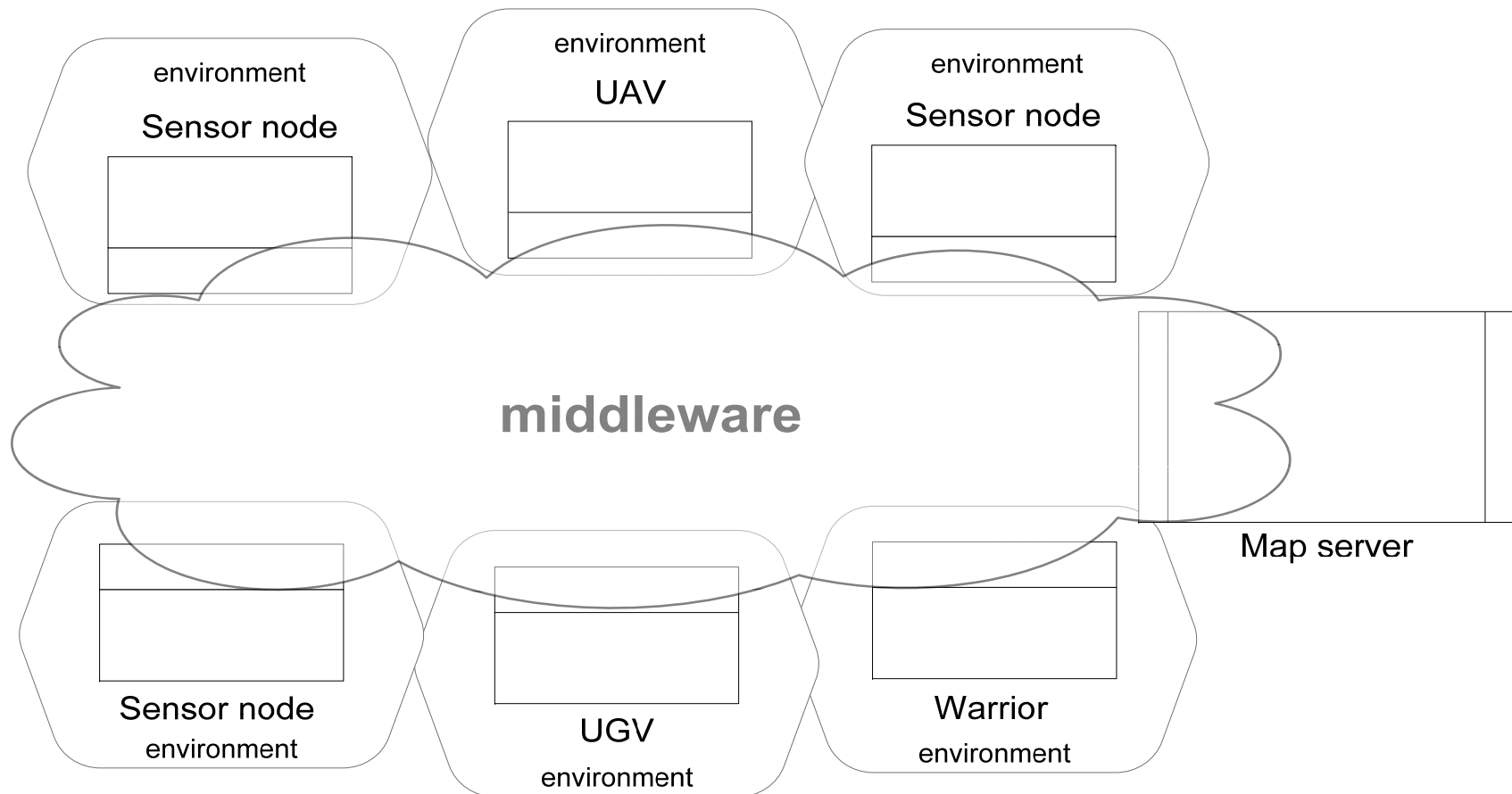
Correspondingly the definition of channel is to be upgraded.

A new problem: Team situation-awareness

- During the collaborative interpretation of situations, as grasped by individual components, the system wide interpretation often leads to modifications of individual decisions.
- Reaching team situation-awareness is the basis of efficient joint actions to invoke and control the growth process.
- Reaching on-line team situation-awareness needs specific supporting infrastructure – the middleware, that is to be implemented and operates according to the interactive, situation-aware model of computation

The use of middleware for creating team situational awareness

In the case of cooperative problem solving



Conclusions

1. Many disciplines are involved in understanding the essence of artificial self-organising systems and reasoning about them, e.g. :
 - modelling and theory, complex systems, biological systems
 - foundations of computation, software and computer engineering
 - communication and collaborative problem solving
2. We try to merge the concepts of multi-agent systems, team situation-awareness, and interaction-centred computation
3. For validation and verification of the results we apply physical toy models, and formal models of systems

Questions, please

SUPPORTING SLIDES

The QMS channels (1)

Time selectivity of a channel is realised by the consumer defined channel function:

$$K(\sigma_{ij}, t) \subset T(p_i), \quad t \in T(p_j).$$

A more practical presentation of the channel function is in backward relative time

$K(\sigma_{ij}, t) = [\mu, \nu]$, where ν is the latest and μ is the earliest state value accessible via the channel σ_{ij} .

To enable time-selective communication, all the state values must be time-labelled and each channel must have its own relative backward time.

The QMS channels (2)

Types of channels

Different types of channels are needed to connect processes with different time-sets and different communication requirements; in practice five types of channels are used:

1. **Synchronous channel**, if $T(p_i) = T(p_j)$
2. **Semi-synchronous channel**, if $T(p_i) \rightarrow T(p_j)$
3. **Asynchronous channel**, if $T(p_i)$ and $T(p_j)$ are independent
4. **Synchronous null channel**, to activate two processes at the same time
5. **Semi-synchronous null channel**, for sequential activation of two processes