Process-Oriented Subsumption Architectures in Swarm Robotic Systems

Jeremy Posso
Department of Computer Science, University of York

Adam Sampson
IAMG, University of Abertay Dundee

Jon Simpson
School of Computing, University of Kent

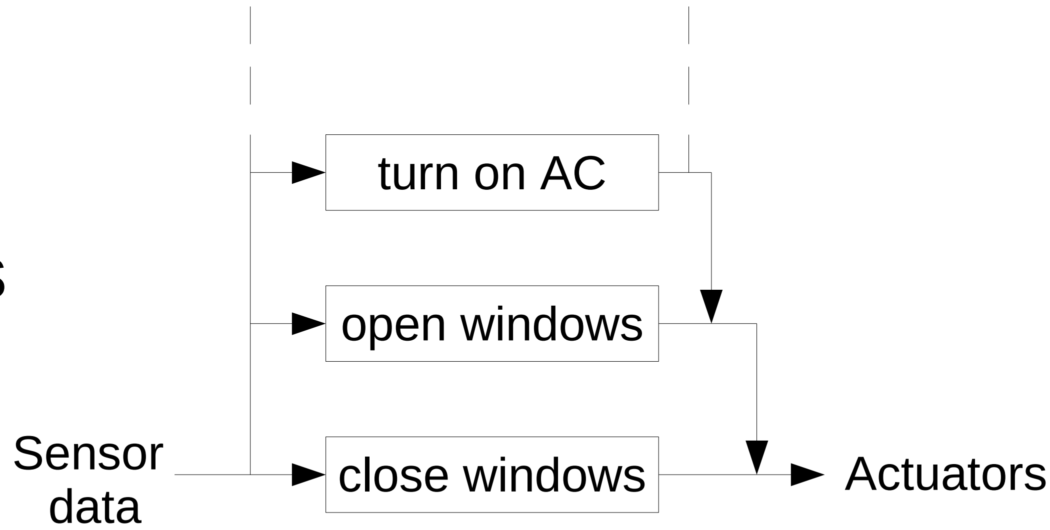
Jon Timmis
Department of Electronics, University of York

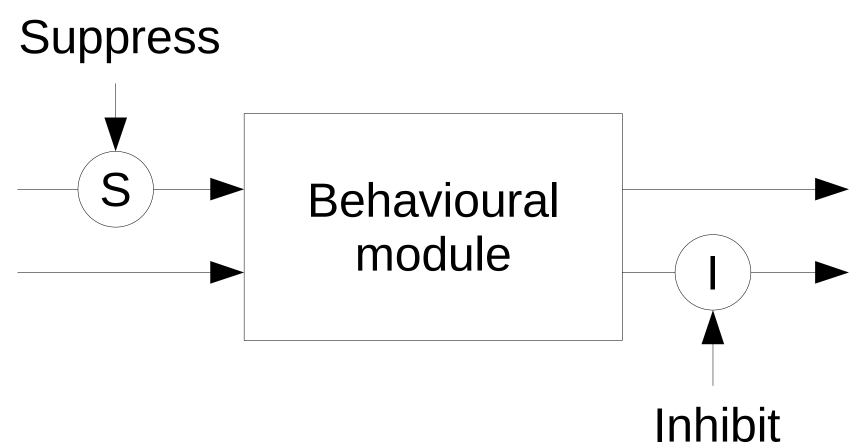
Introduction

- Jeremy Posso's MSc project at York in 2009
  - Supervised by Jon Timmis
  - Worked with Jon Simpson on architecture, and Adam Sampson on Player/Stage bindings
- Robotic control is an inherently **concurrent** problem: sensors, actuators...
- **Process-oriented programming** should be a convenient way to implement control systems
  - Transterpreter, Plumbing...

Subsumption architecture

- Layered behaviours
  - Map sensor inputs to actuator outputs
- Suppressors
- Inhibitors
- No planning – purely reactive
- Modular √ Compositional **?**





Past work

- **CPA 2006**: Simpson, Jacobsen and Jadud, "Mobile Robot Control: the Subsumption Architecture and occam-π"
  - Implemented subsumptive control components in a process-oriented system
- **CPA 2009**: Simpson and Ritson, "Toward Process Architectures for Behavioural Robotics"
  - Compared subsumption with other approaches; identified scalability problems

Swarm robotics

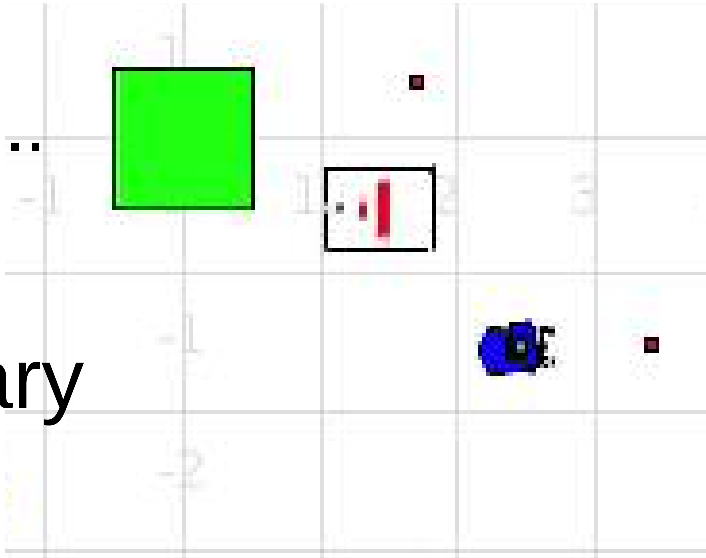
- Several robots collaborate to perform a task
  - May involve **engineering emergence**
- **Local** intelligence, not remote control
- Robustness
- Flexibility
- How can we use a process-oriented subsumptive control system in a swarm context?

The task

- Foraging: common swarm problem
- Many identical robots collect pieces of **rubbish** from a field, and deposit them in a **bin**
- Robots must coordinate to avoid collisions, while covering as much ground as possible
- Robots have limited battery life – must recharge at charging stations when low

The robot

- Pioneer platform, modelled within Stage
  - Realistic, noisy... so nondeterministic
- Gripper for collecting rubbish
- Camera for spotting rubbish, other robots, chargers and the bin
  - Rubbish is red, robots are blue...
- Sonar for avoiding walls, etc.
- All driven through the Player library
  - … which Jeremy significantly improved our bindings to

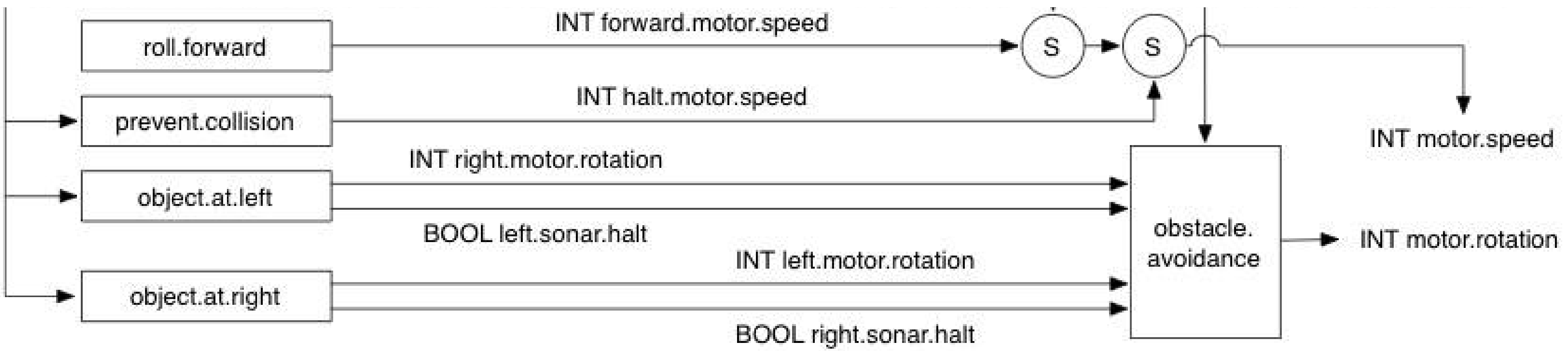


Design

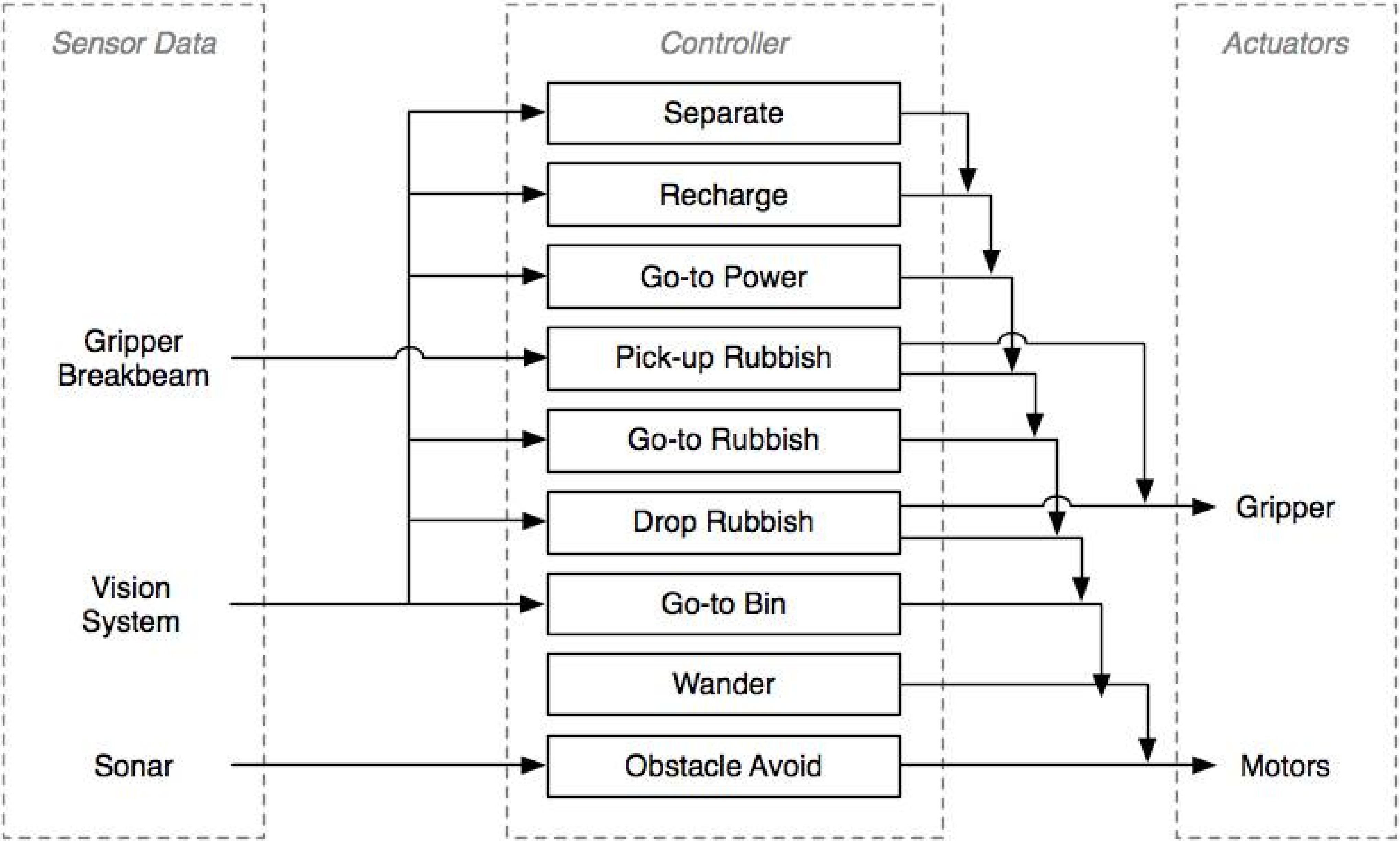
- First we identify the high-level behaviours
    - Arrange in priority order, most important last
- **Explore**
- **Avoid collisions**
- **Acquire rubbish**
- **Deposit rubbish**
- **Recharge**
- **Collaborate**

Design

- Then we can break those down into simpler behaviours, and map those to processes
- e.g. **Avoid collisions**
  - Move forwards
  - … unless you're about to run into something
  - If sonar senses something to one side, turn away from it

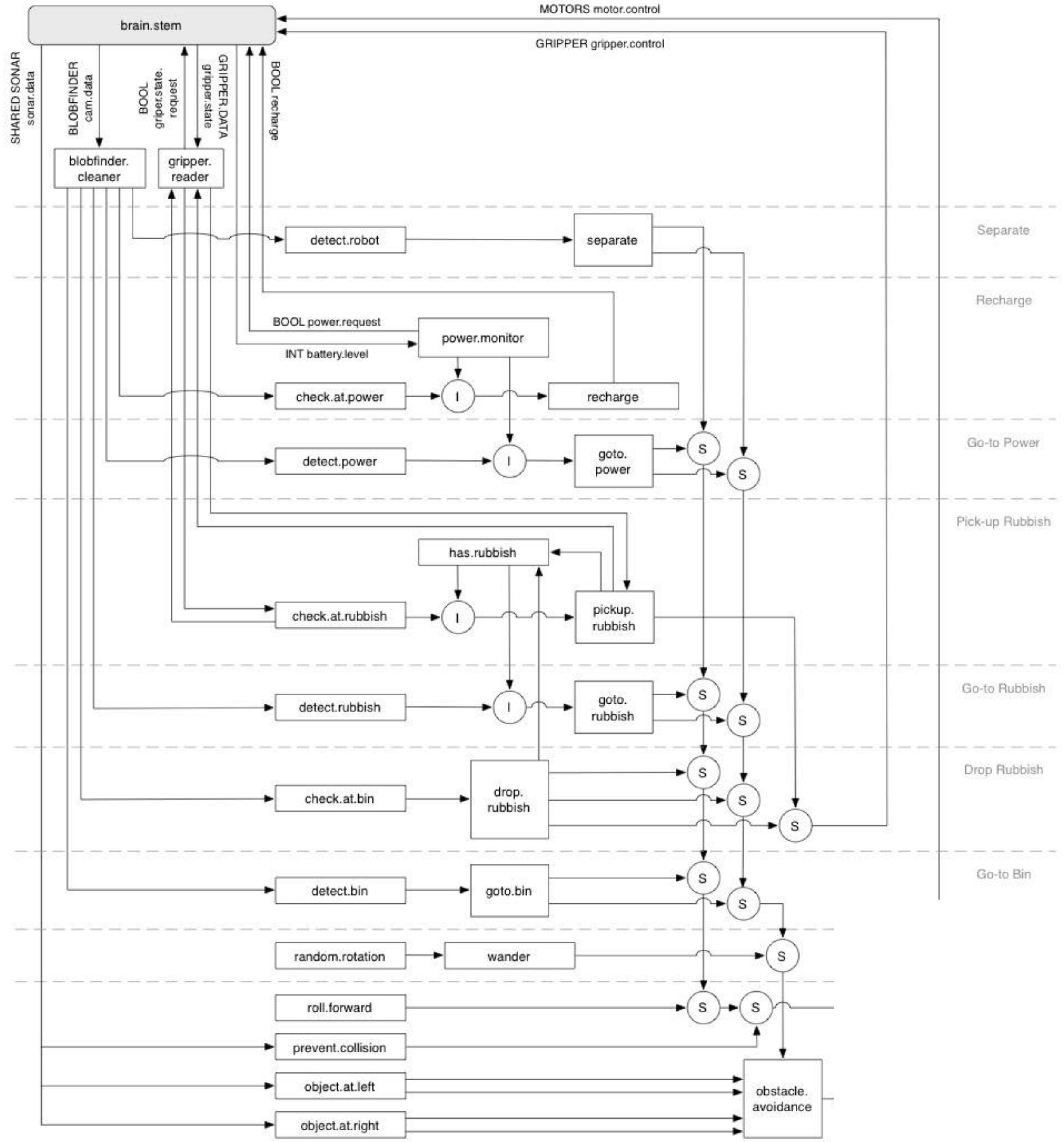


The control system



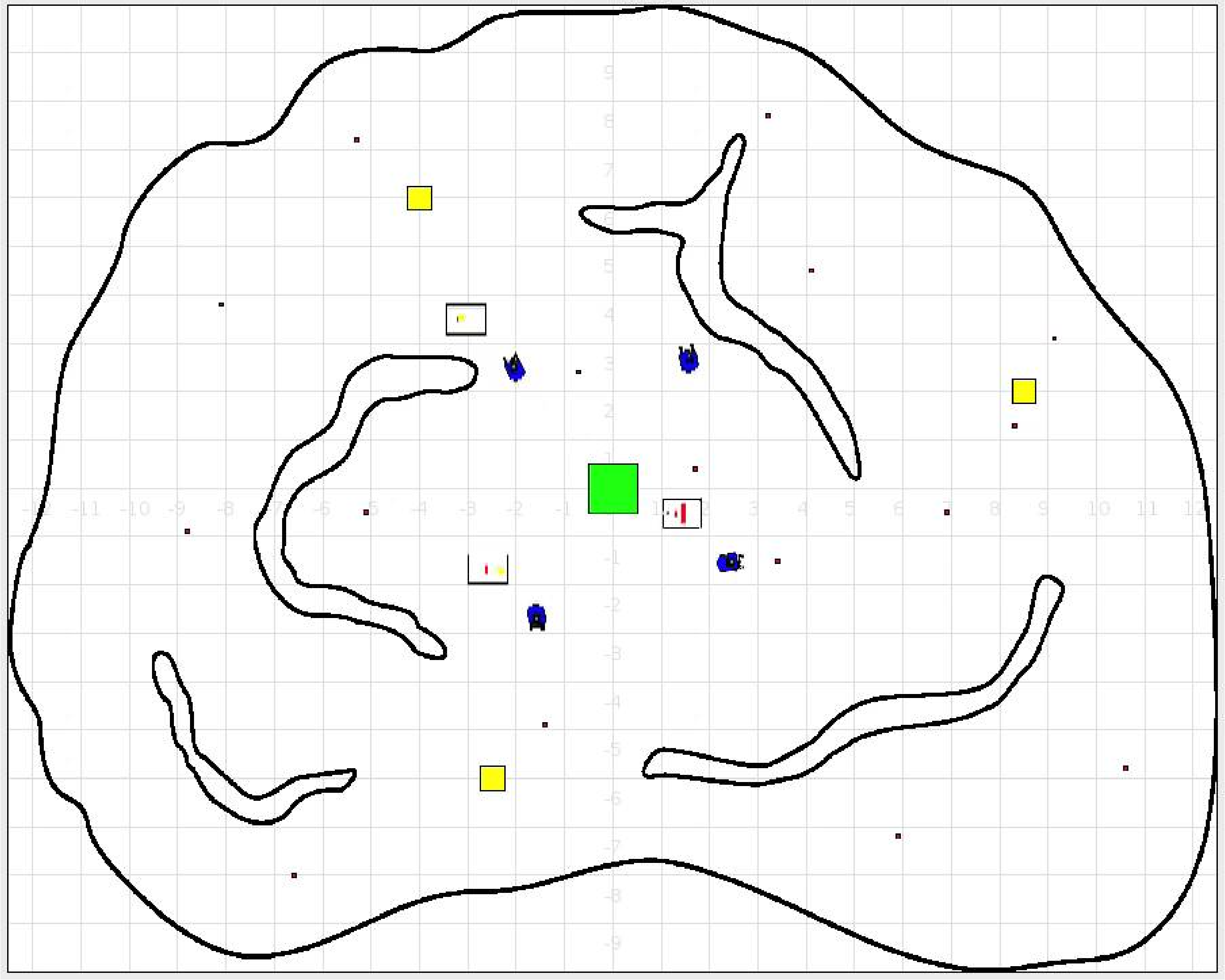
The process network

(see paper for more detail)



Trials

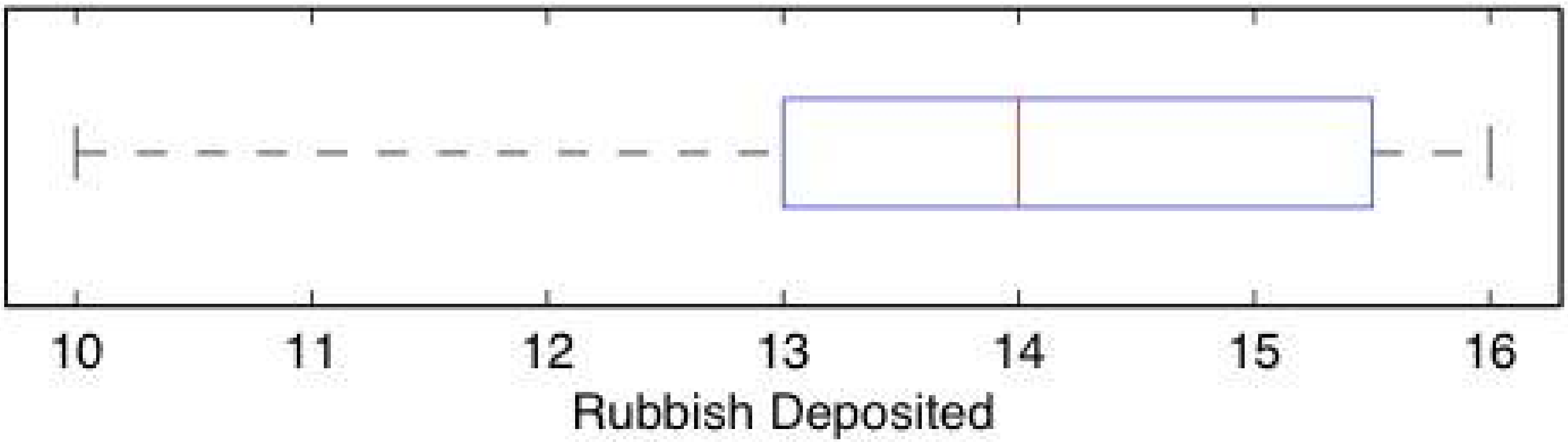
- Four robots, sixteen pieces of rubbish
- **Success** when all rubbish in bin within twenty minutes



Video

Results

- Ran **20** trials...
- … of which only **5** were completely successful



- It works sometimes – why not always?

Diagnosis

- Some robots wander around but don't pick up or put down rubbish...
- Some behaviours aren't working
- Part of the control system has **deadlocked**
- … but no way to detect this until it's used
- This appears to be a common problem with complex subsumptive controllers

Desiderata

- **Synchronous** channels aren't a good fit here – we want **overwriting-buffered** channels
  - Can we identify new design patterns for safe programming with **asynchronous** communications?
- We don't have good tools for **debugging** or **performance analysis** (e.g. tracking latency)
  - The Transterpreter can give you the data...
  - … we just need to display/explore it

Conclusion

- We've built a complex **subsumptive** control system using **process-oriented** techniques
- Design and implementation straightforward
- It works... sometimes!
  - We need better tools to tune and debug it
- Previous attempts at subsumption in occam-π built **much simpler** systems, and didn't run into these scalability problems

Future work

- Build subsumptive swarm systems that **span multiple robots**
  - e.g. allow one robot to suppress behaviours in another robot
- Investigate **other approaches** for complex problems like this
  - e.g. Colony architecture

Any questions?