Multiagentové systémy

Andrej Lúčny

KAI FMFI UK

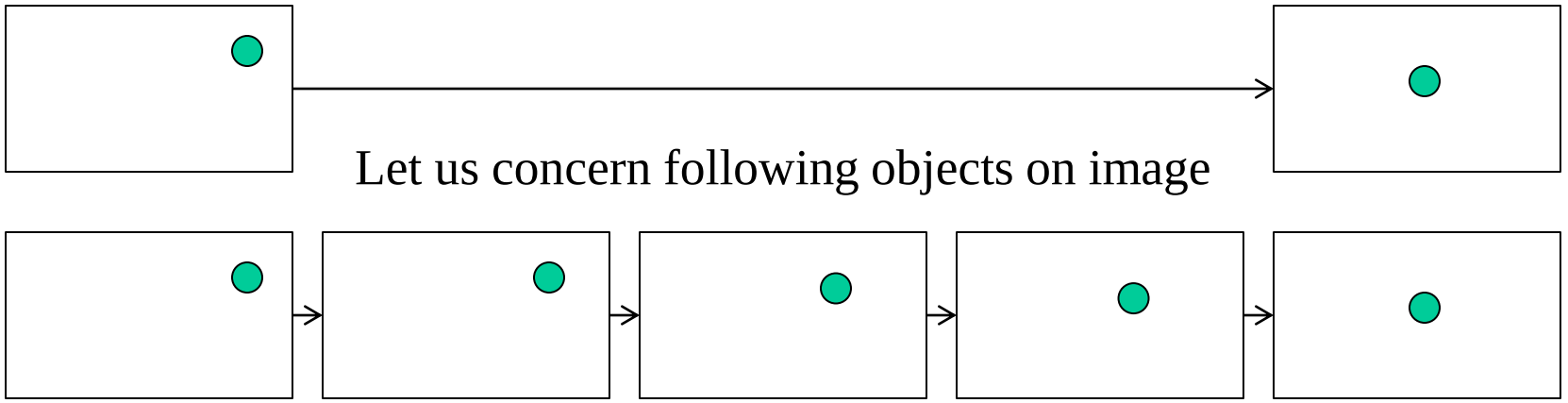
lucny@fmph.uniba.sk

http://www.agentspace.org/mas

Real-time operation

- If something is to happen, it must happen within a certain time (latency)
- System speed must not be determined by the slowest modules: which is fast, it is fast, what is slow, it is slow
- (Brain is fast, neurons slow)

Body as a computational device



- Instead of single calculation of non-linear function (what is difficult), we iterativelly calculate its linear approximation (what is easy)
- Just for nice math: we in fact employ Taylor's theorem

(Taylor's theorem)

$$f(x) = p_0 + p_1(x - x_0) + p_2(x - x_0)^2 + \cdots + p_n(x - x_0)^n + \ldots$$

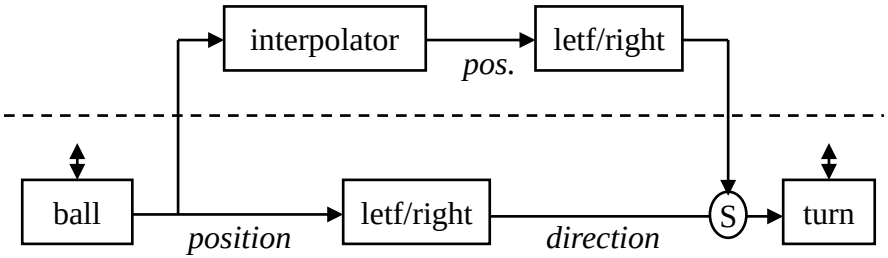
$$f(x_0) = p_0 \qquad f^{(1)}(x) = p_1 + 2p_2(x - x_0) + \cdots + np_n(x - x_0)^{n-1} + \ldots$$

$$\ldots$$

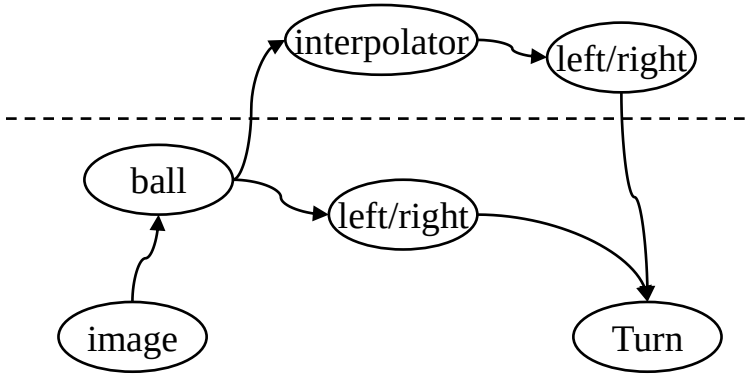
$$p_n = \frac{f^{(n)}(x_0)}{n!} \qquad f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

If $x - x_0$ is small: $f(x) \doteq f(x_0) + f'(x_0)(x - x_0)$

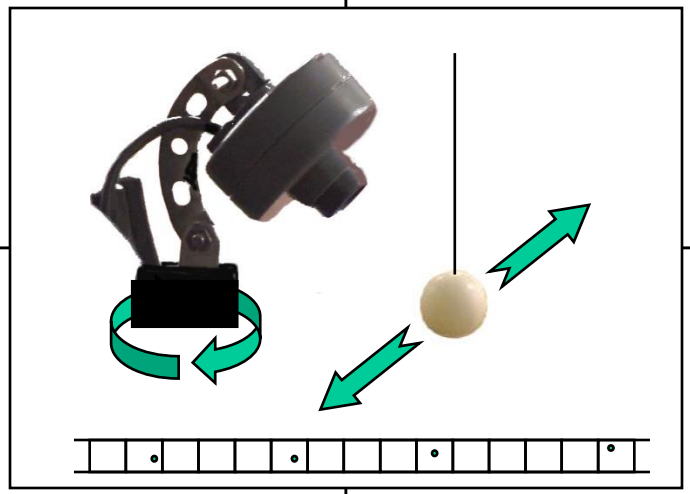
$$f_{t+1} := f_t + \bigtriangledown f_t \Delta x$$



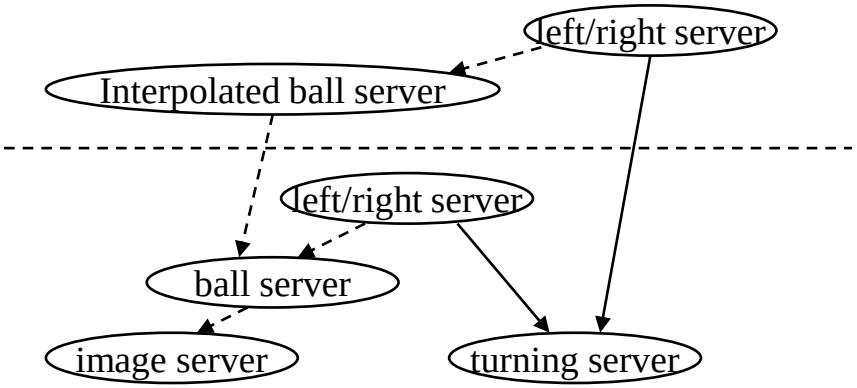
subsumption architecture



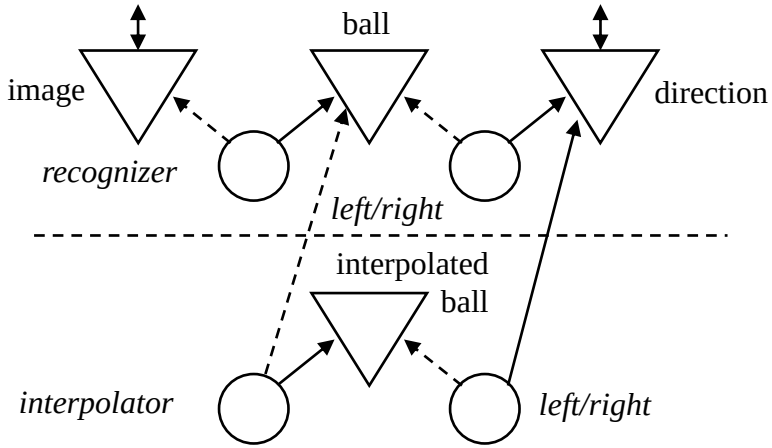
Society of mind



pyramidal client-server architecture



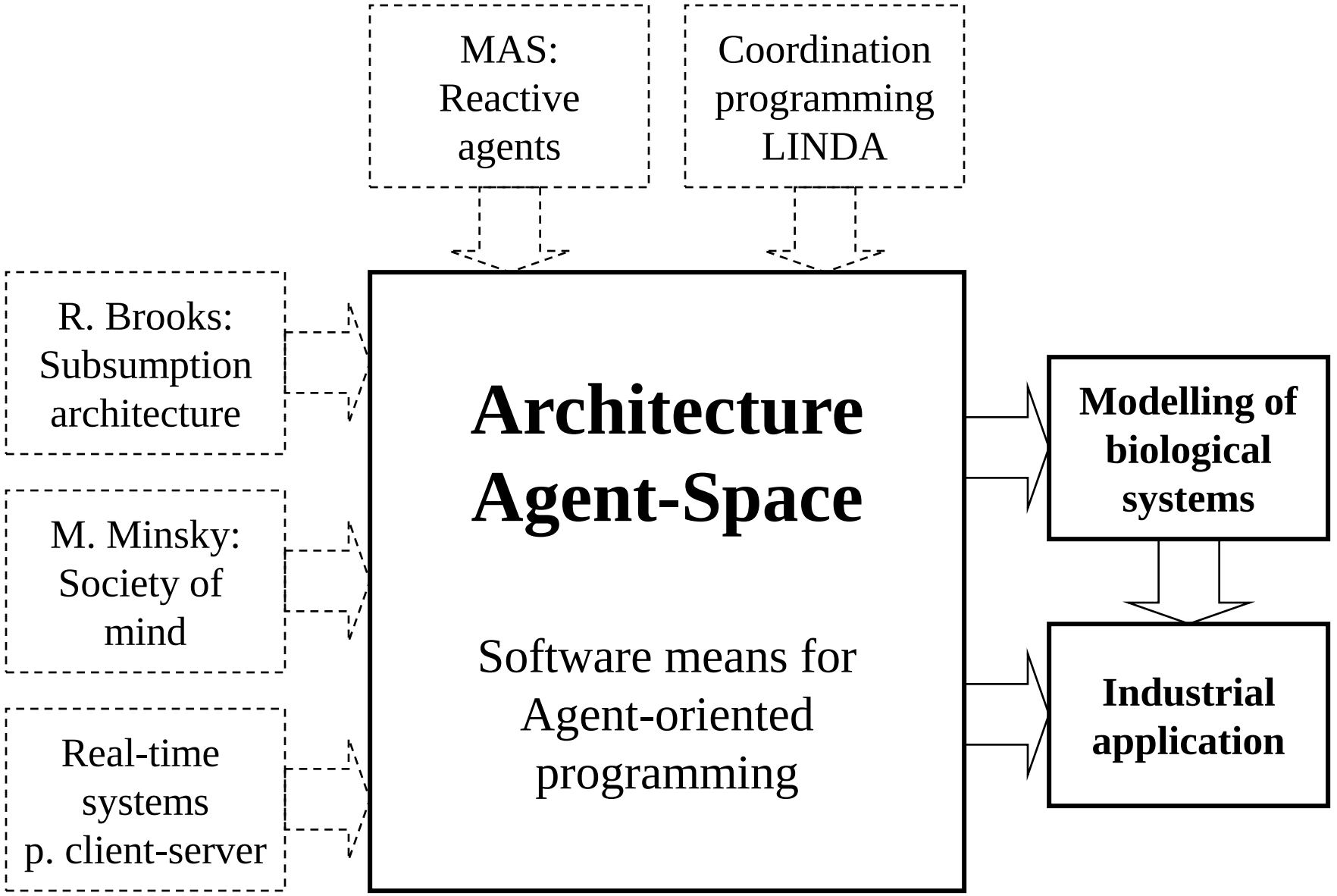
architecture agent-space



Agent-Space

- Multi-agent architecture designed at FMFI UK 1997-2004
- It express ideas of Brooks and Minsky ba language of MAS (with indirect communication)

Jozef Kelemen

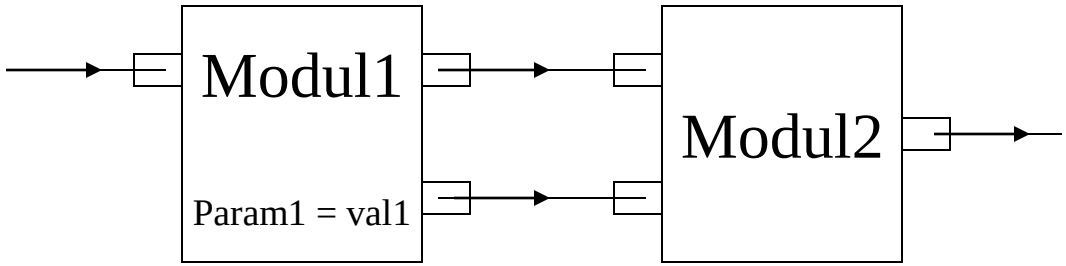

MAS:
Reactive
agents
Coordination
programming
LINDA
R. Brooks:
Subsumption
architecture
M. Minsky:
Society of
mind
Real-time
systems
p. client-server
Architecture
Agent-Space
Software means for
Agent-oriented
programming
Modelling of
biological
systems
Industrial
application

Motivation: development of biomimetic control systems

- Agent-Space is a kind of MAS implementation on VM
- It serves for control, e.g. of mobile robots
- It mimics modularity od distributed hardware in software by multi-agent modularity
- In software of course we can organize modules even in better way than we could organize distributed hardware modules

Traditional control

- modules have fixed number of inputs, outputs and parameters
- one output is linked to several inputs
- transformation of inputs to outputs is performed by a scheduler and it is often uninterruptible



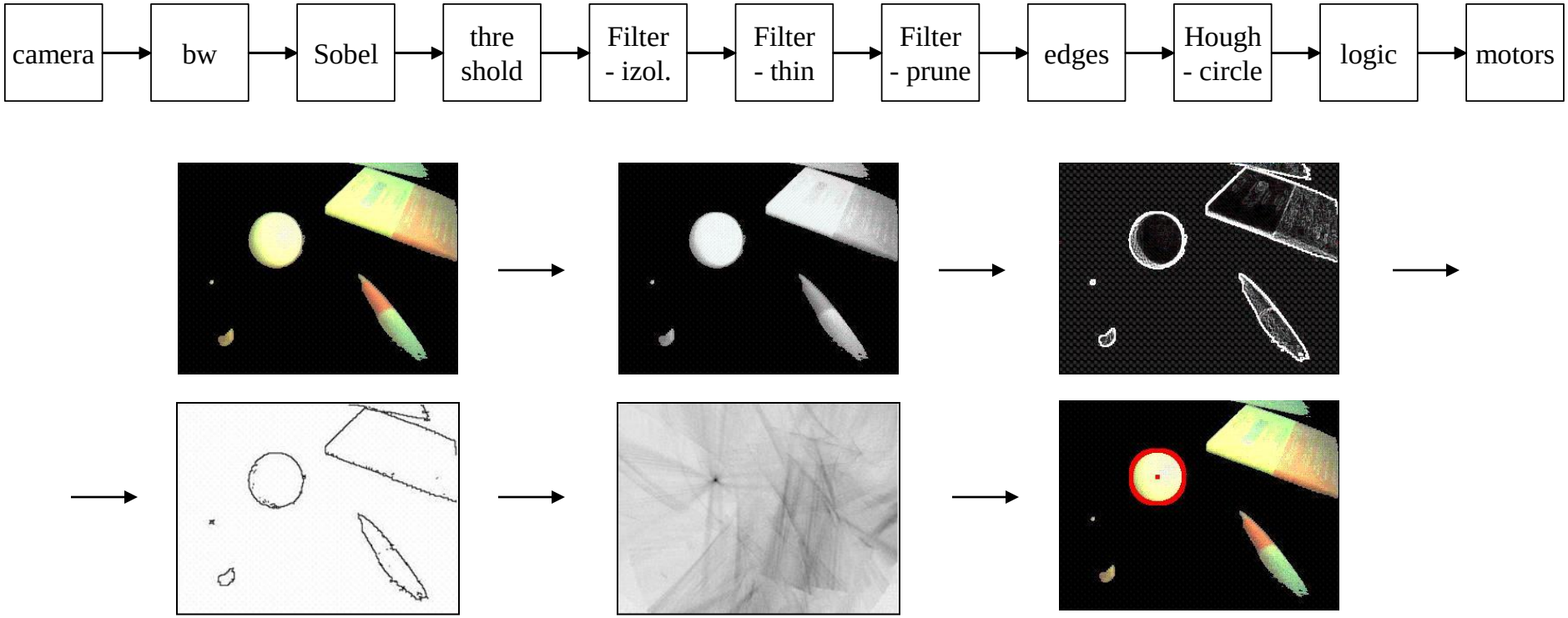
An example of alternative approach

- Let us concern a simple example:

  A mobile robot following a ping-pong ball

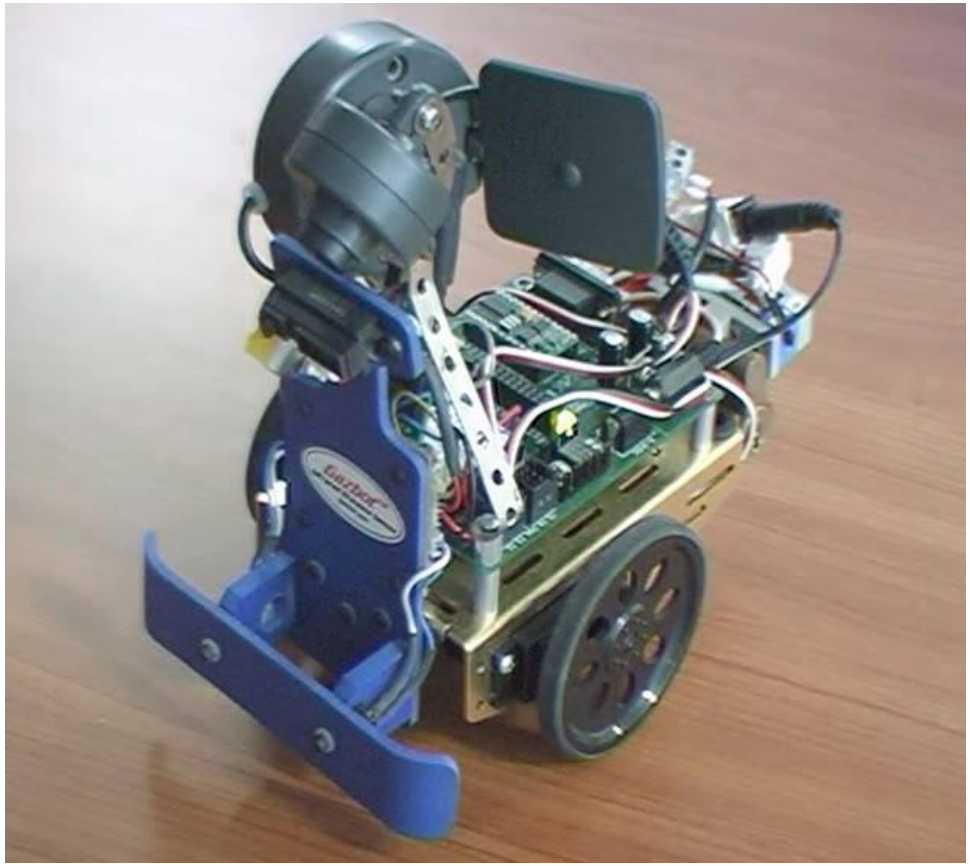
Motivation example

- Initial solution can be based on a single pipeline



Motivation example

- Gradually we concern more a more wide operational conditions
- and observe where the traditional approach falls into troubles



Motivation example

What condition can be considered:

- Different illumination of scene
- More balls in the scene
- Following of occluded ball
- Active search for ball which is not shoot to image

Troubles of the traditional approach

- Combination of fast and slow modules
- Different frequencies of individual inputs to module
- Dynamic change of outputs from module
- Limited time validity of some data
- Non priority based coordination

Alternative architecture

- Connections among modules will be replaced by named data on a blackboard (space), so called blocks, providing indirect communication among the agents
  - Agents can read, write and delete the blocks
  - Agents can define time validity and priority of its write operation

Alternative architecture

- Modules will be replaced by agents
  - We replace global scheduler of modules by own control of agents
  - Module calculation will be put to loop blocked by proper timers and triggers and it will run in own thread
  - Inputs and output to the module will turn to read and write operation over space

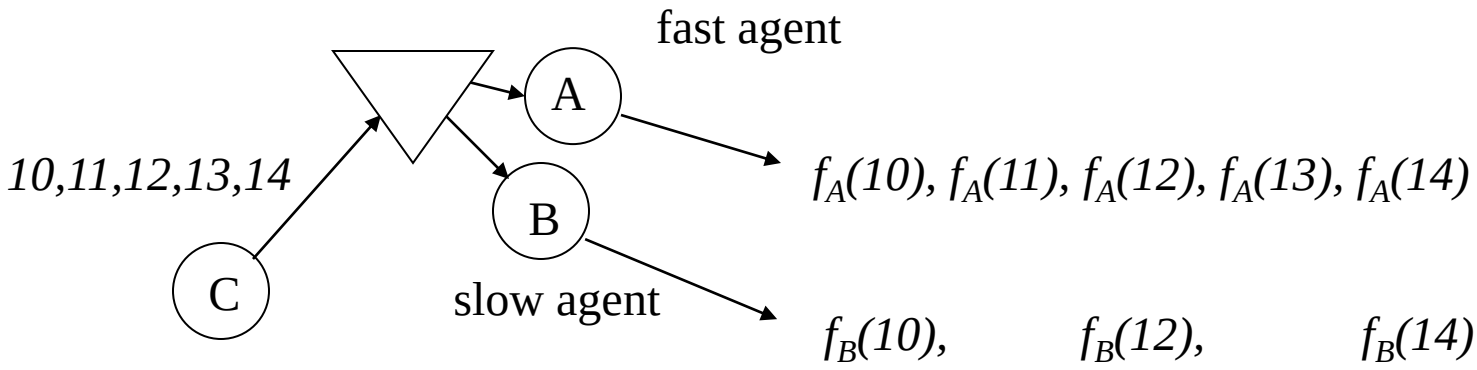
Alternative architecture

We have to pay attention to some implementation details, namely:

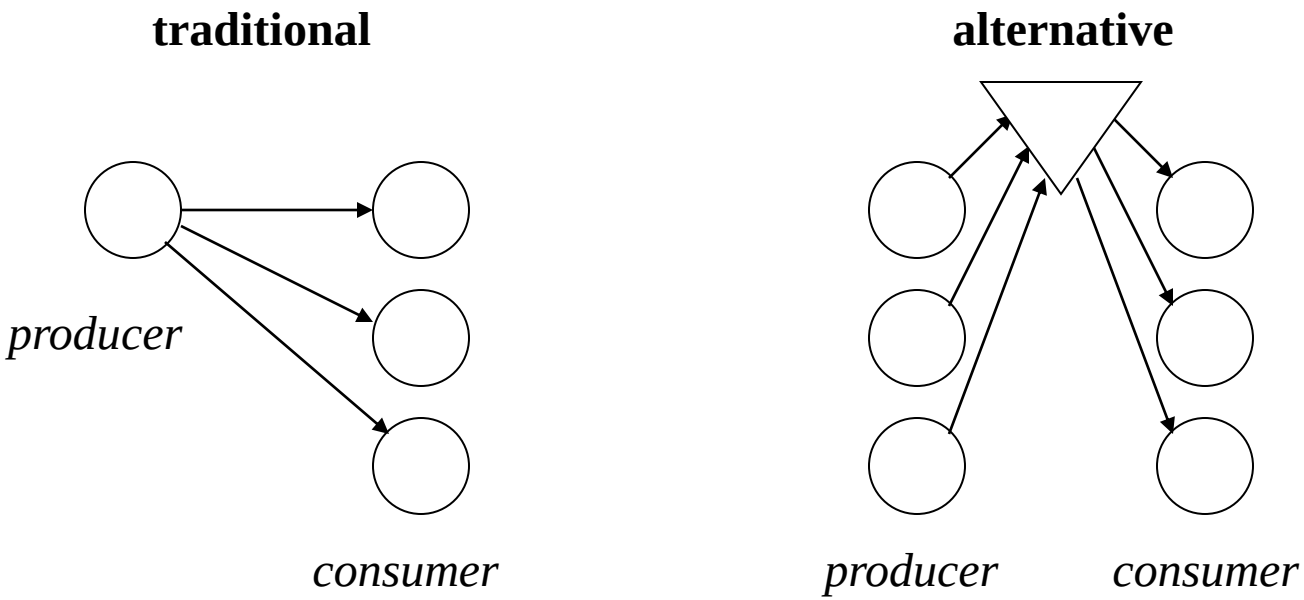
- It is not wished to create blocks by calling a create operation over space; it is much better to let the block to be create by the first write operation
- Blocks can be empty
- It is better to write nothing and let the original value disappear due to limited time validity than to write ‘bad value’
- Mass read and write operation are important, based on mask (wildcard or regex)

Implicit sampling

- Since write operation overwrites data stored in a block regardless their consumers have undertaken them or not, any data flow is inherently (potentially) sampled.



Data flow many:many



Code example

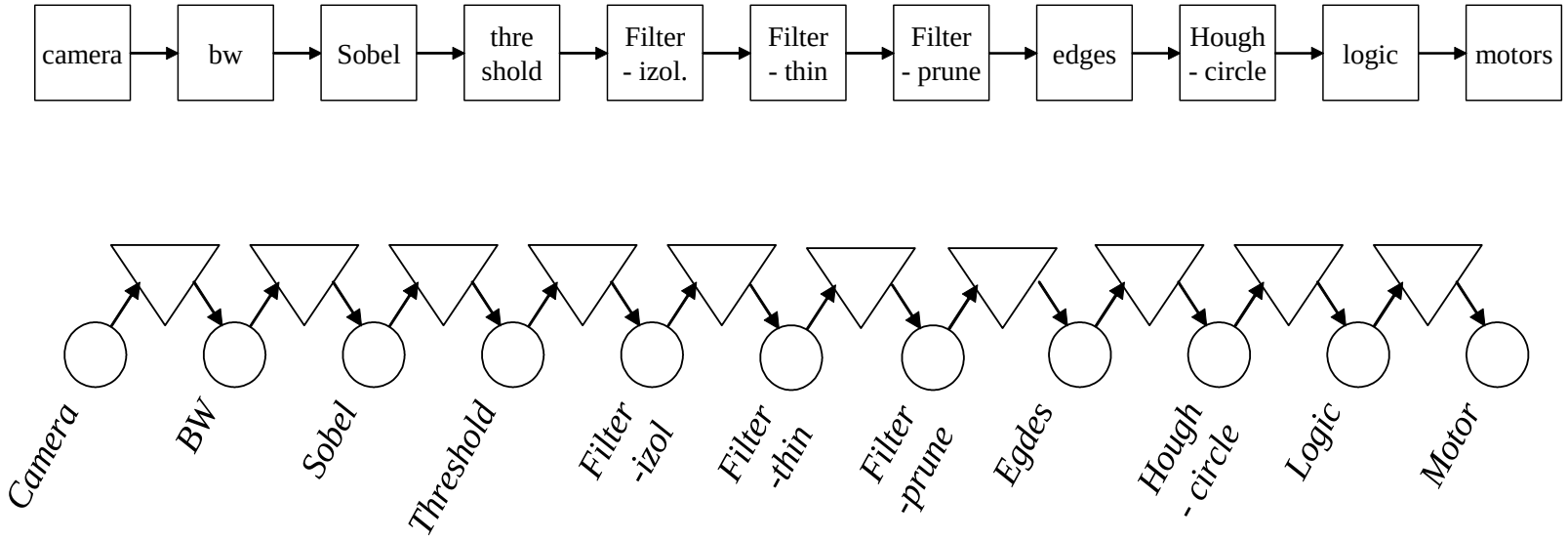
```
package com.microstepmis.agentspace.demo;
import com.microstepmis.agentspace.*;

public class Agent1 extends Agent {

  int i = 0;

  public void init(String[] args) {
    attachTimer(1000);
  }

  public void senseSelectAct() {
    System.out.println("write: "+i);
    write("a",i++);
  }

}
```

```
public class Agent2 extends Agent {

  int i;

  public void init(String args[]) {
    attachTrigger("a");
  }

  public void senseSelectAct() {
    i =  (Integer) read("a",-1);
    System.out.println("read "+i);
  }

}
```

```
public class Starter {
  public static void main(String[] args) {
    new SchdProcess("space","com.microstepmis.agentspace.SpaceFactory",new String[]{"DATA"});
    new SchdProcess("agent1","com.microstepmis.agentspace.demo.Agent1",new String[]{});
    new SchdProcess("agent2","com.microstepmis.agentspace.demo.Agent2", new String[]{});
  }
}
```

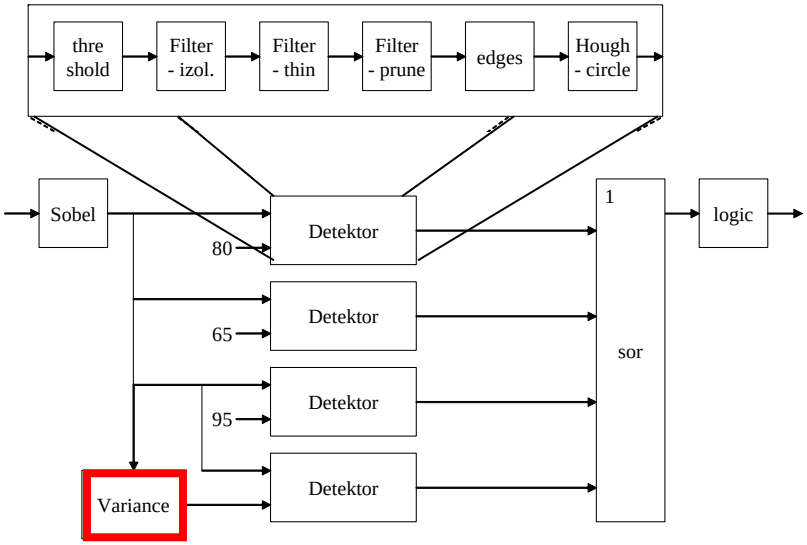
Comparison

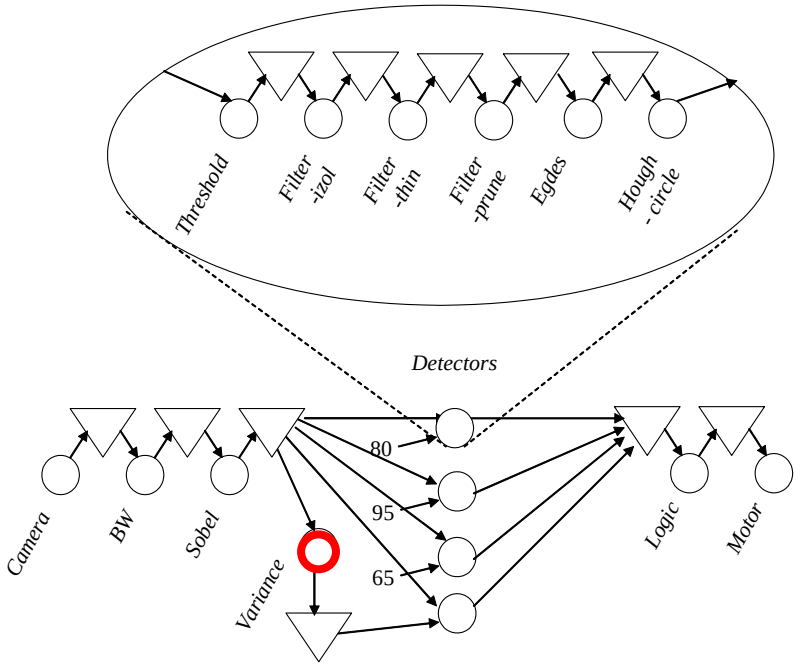
Different distribution of timer frequencies



Comparison

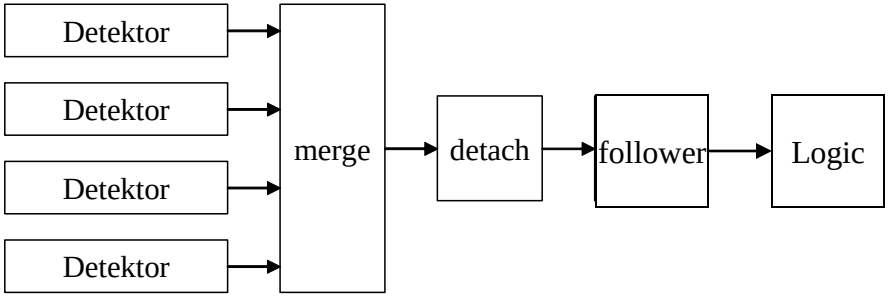
Combination of fast and slow modules

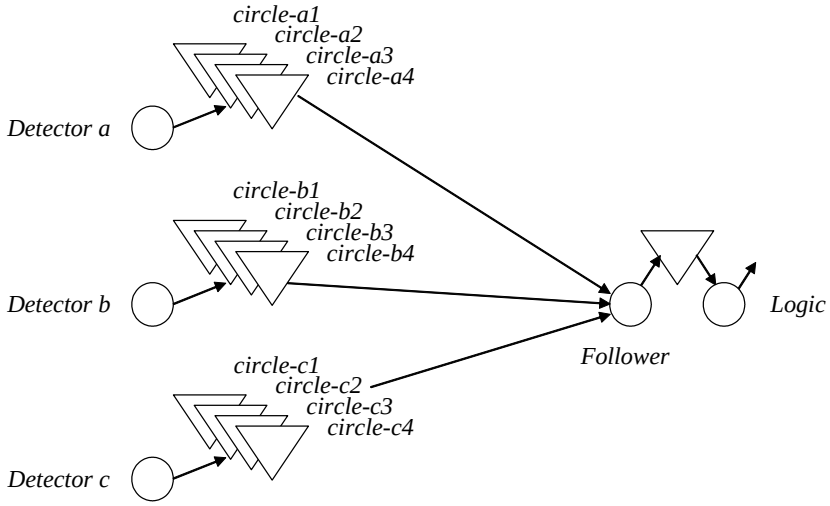




Comparison

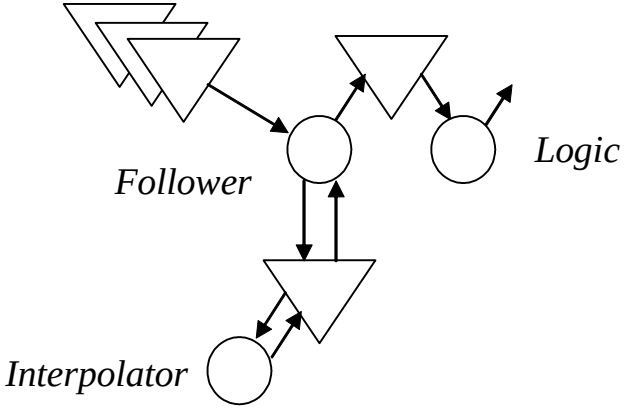
Dynamic change of inputs and outputs to modules

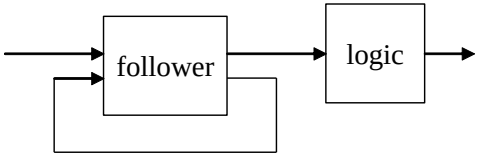


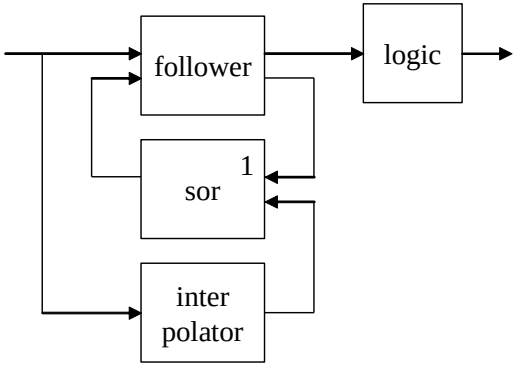


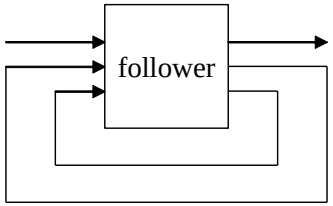
Comparison

Pure reactivity
and limited time validity



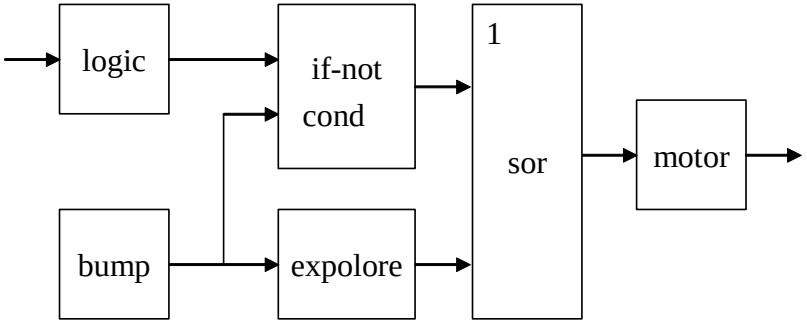


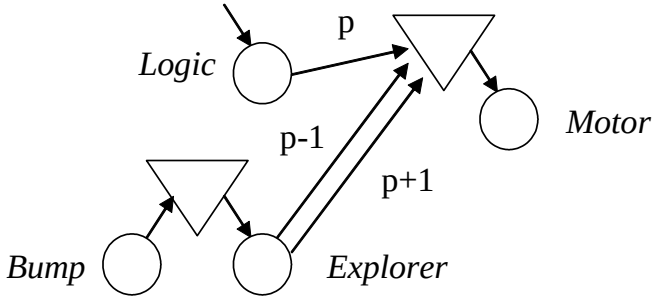




Comparison

Non-trivial coordination among layers based changing priority

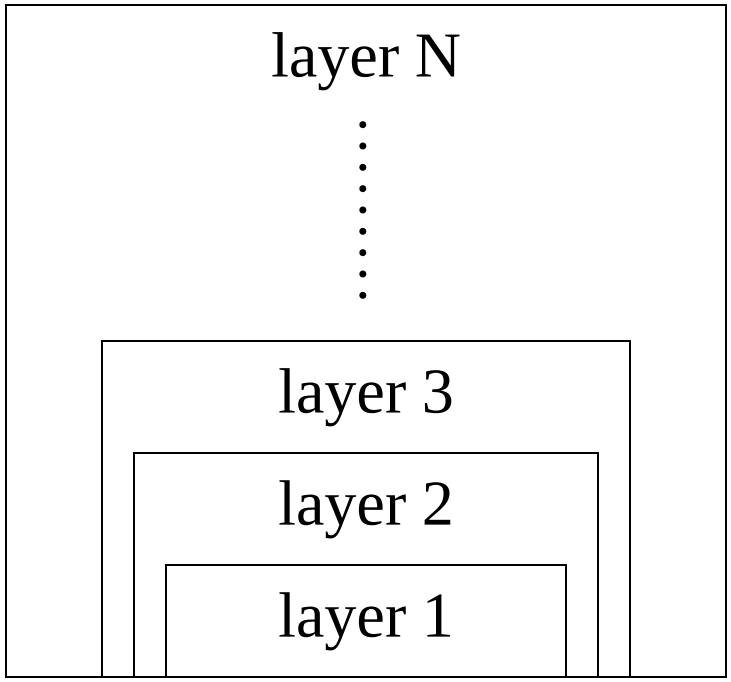




Advantages

- The selected problems of traditional architecture can be solved easily
- Yet another advantages ?
- Some ideas for improvement of control programming ?

Subsumption



Agent-Space is able to implement Brooks' subsumption

Operation in real time

- Agent-Space supports real-time operation
- Real-time is important for cognitive processes

https://youtu.be/NTrJfW939S0
https://youtu.be/DwjnkU4Hdf0
https://youtu.be/GzHnRXik41c
https://youtu.be/KWK7PpfglNw

Combining strong and weak agents

- MAS modularity enables incorporation of GOFAI methods into system in a comfortable way
- E.g. top hierarchical layers will be implemented with strong agents; as a result we employ GOFAI only where it is more suitable than stimulus-response modules

MAS and programming languages

- Potential to extend language by communication means among agents
- E.g. agent-space would extend assignment:

  var = val; var = val for 1s at priority 0.5;
- E.g. code can be invoked by assignment

  obj.met(arg); obj.attr = val; delay(1s);

So far, no adaptation

- So far, we have dealt with systems which can do someting intelligent, but their capabilities do not improve through time
- That is a feature of the so called Cambrian inteligence
- How could we provide adaptation?
  (to be continued ...)