

Introduction to Robotics for cognitive science

Dr. Andrej Lúčný

KAI FMFI UK

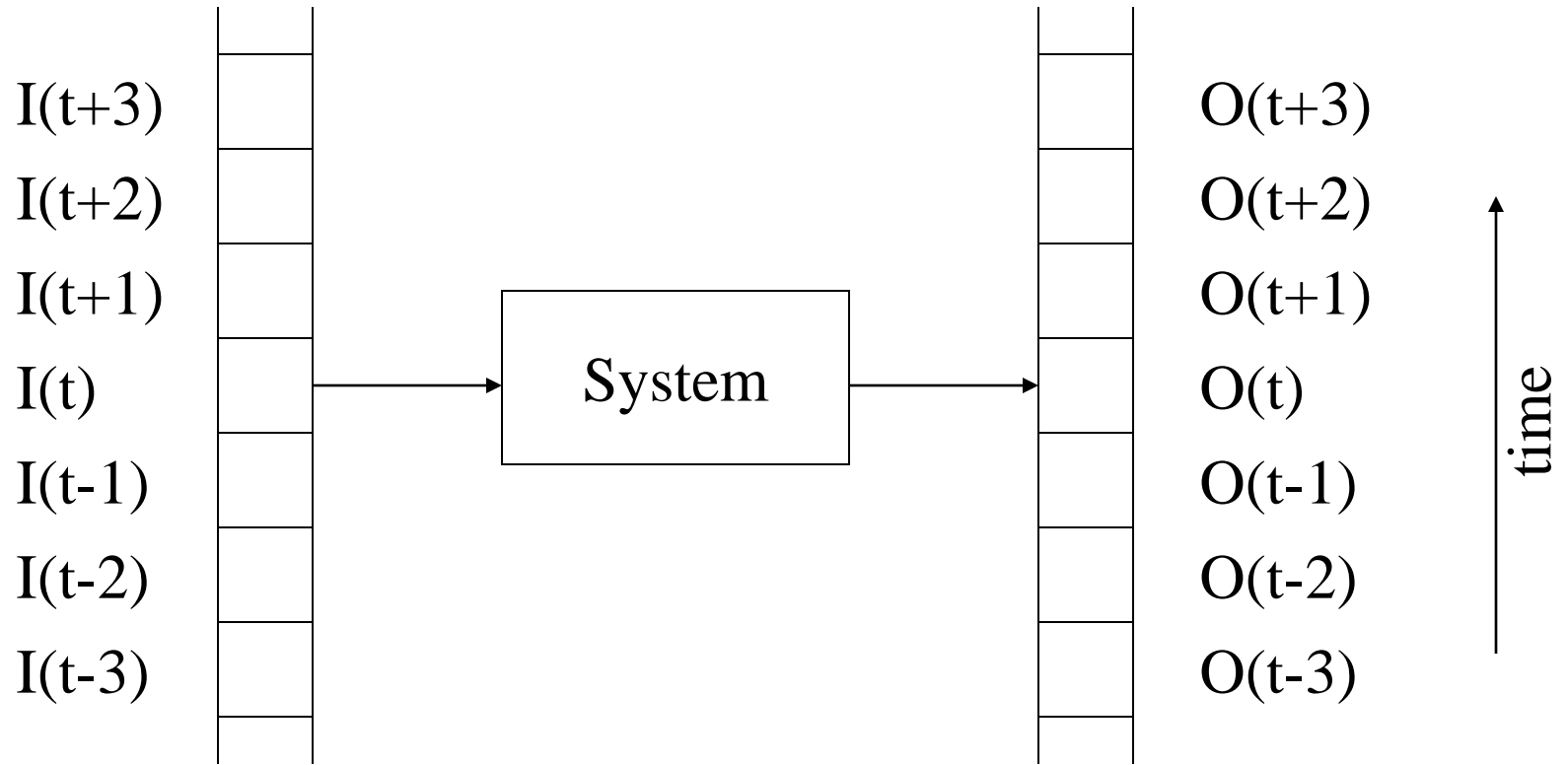
lucny@fmph.uniba.sk

Web page of the subject

www.agentspace.org/kv

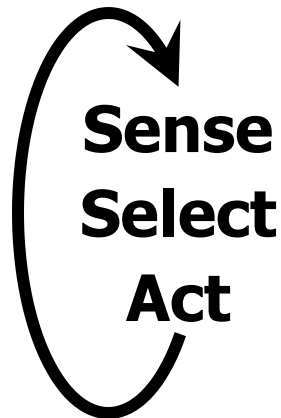


Control



Monolytic solution

- Single loop: read data from sensors, calculate actions, perform actions



Problems

- Code becomes complicated when more and more features are implemented
- Slow processes slow down fast processes.
As a result, the system is not working in real-time

Real-time

- Operation in real-time means that if an event should be performed in time t , it is sure it will be performed in time $t+L$, not later
- L is latency, and it is expressed in ms

Real time

```
import time

t0 = time.time()

time.sleep(3)

t1 = time.time()

print(t1-t0)
```

Modular architectures

We need to decompose the control somehow

Technical means: threads, IPC, middleware,...

Threads

- We can start more program counters in parallel:

```
import threading
```

```
def process1():  
    time.sleep(2.0)
```

```
def process2():  
    time.sleep(3.0)
```

```
threading.Thread(target=process1).start()  
threading.Thread(target=process2).start()
```

Modular architectures

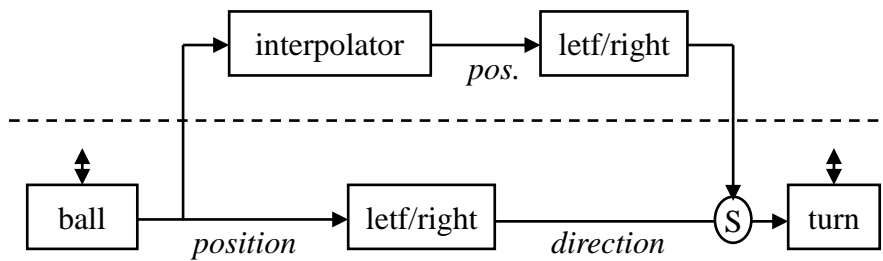
Historical architectures:

- Brooks' subsumption architecture
- Minsky's "society of mind"
- Payton & Rosenblat: Fine grained architecture
- RTOS architectures (pyramidal client-server)
- Arkin's Behavioral architecture
- ...

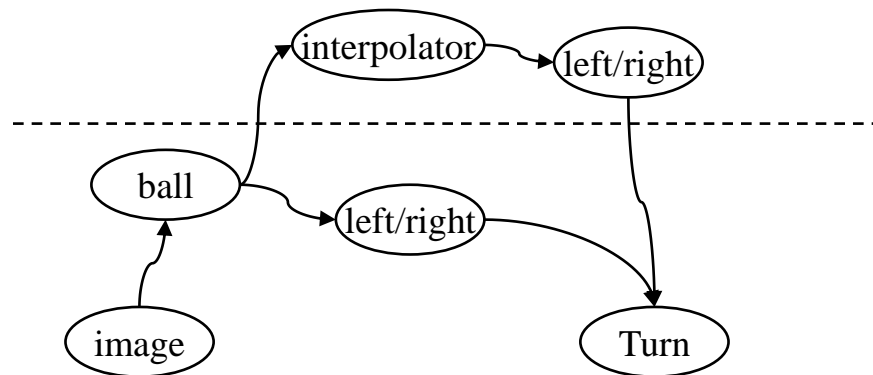
Modular architectures

Implementations:

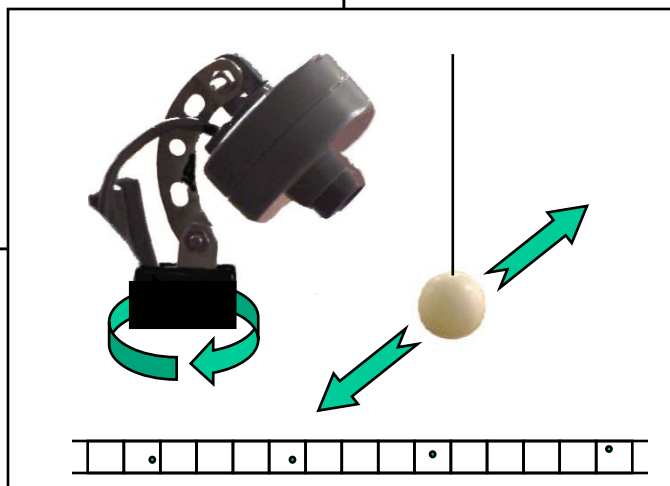
- Robot Operating System (ROS)
- Yet Another Robot Platform (Yarp)
- Agent-Space architecture
(our implementation, much more simple)



subsumption
architecture
[Brooks]

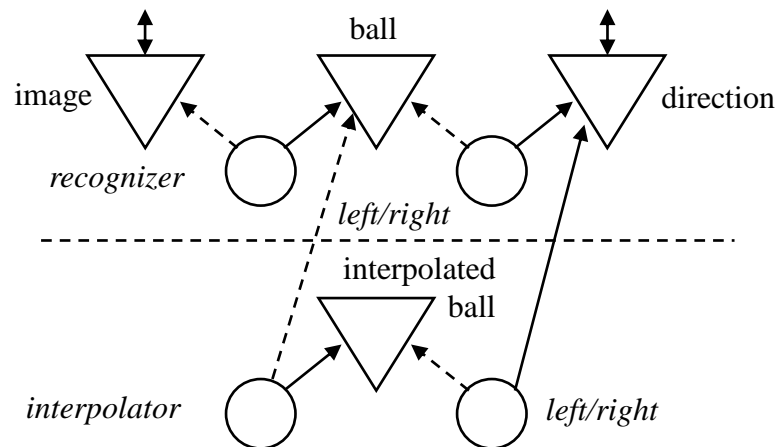
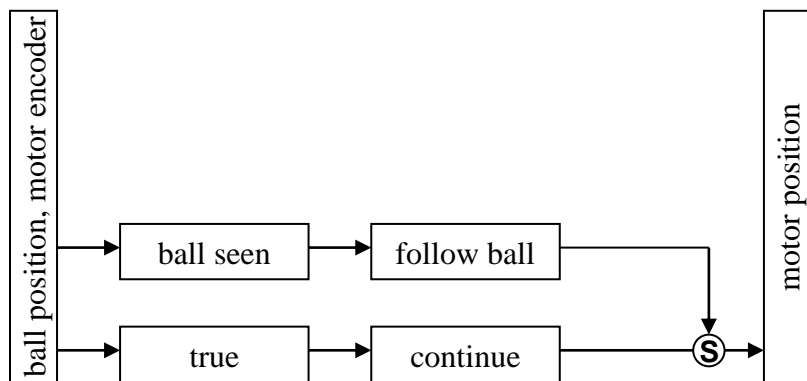


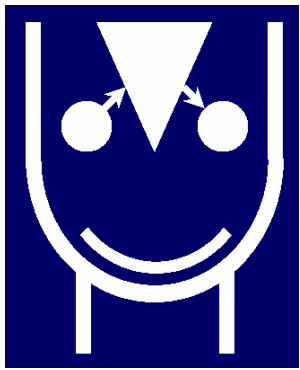
Society of mind
[Minsky]



behavioral
architecture
[Arkin]

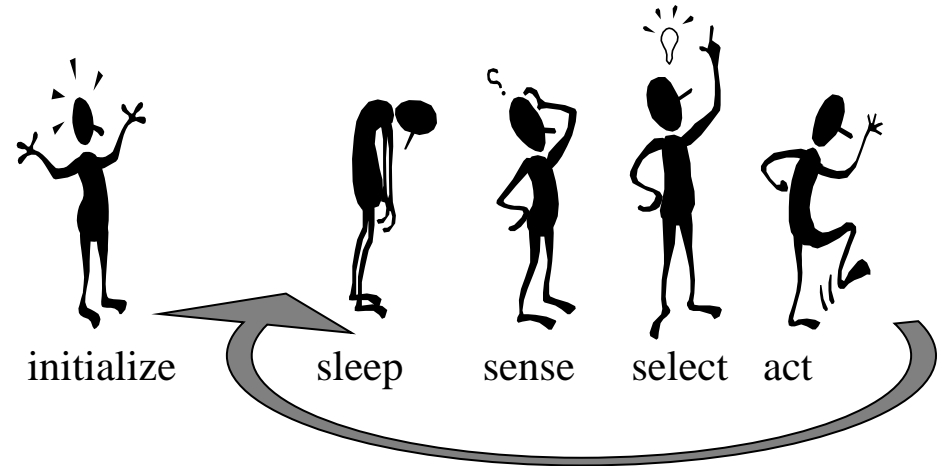
agent-space
architecture
[Lucny - Kelemen]



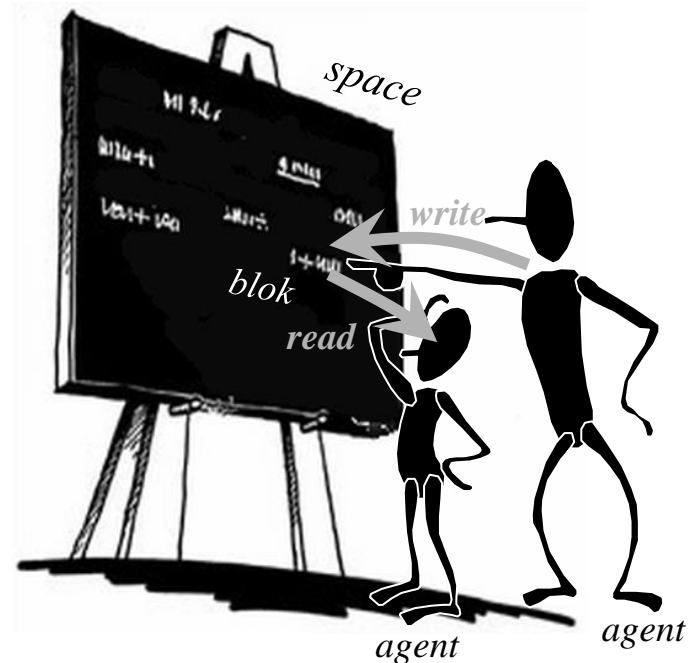


Architecture Agent-Space

- System consists of agents

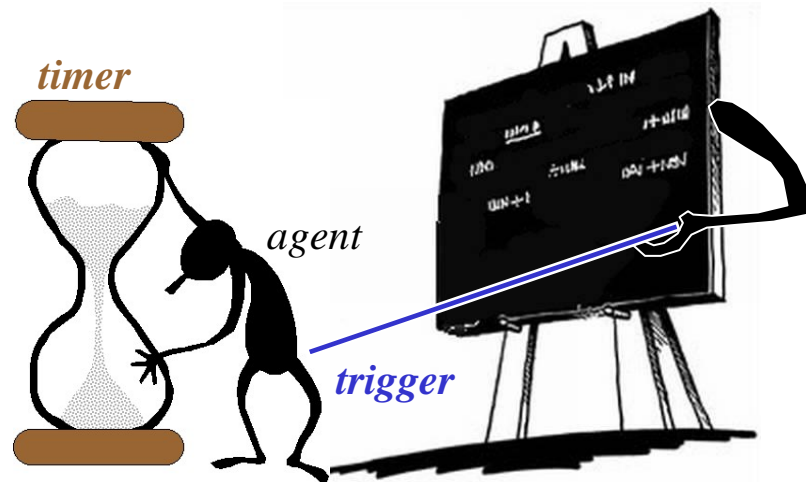


- Agents communicate through Space



Implementation in Python

- Each agent is an object with its own thread
- It calls *read* and *write* methods of singleton object *Space*
- Agent is regularly wakened up by timer or trigger (by the *write* operation performed by another agent)



Implementation in Python

Blackboard (extended dictionary)

```
from agentspace import space
```

```
space['a'] = 2
```

```
print(space['a']) # 2
```

```
print(space['b']) # None
```

```
print(space(default=-1)['b']) # -1
```

Implementation in Python

validity and priority

```
space(validity=0.5)['b'] = 1
print(space['b']) # 1
time.sleep(1)
print(space(default=-1)['b']) # -1
```

```
space['c'] = 1 # default priority is 1.0
space(validity=0.5,priority=2.0)['c'] = 2
space(validity=2.0,priority=1.5)['c'] = 3
print(space(default=0)['c']) # 2
time.sleep(1)
print(space(default=0)['c']) # 0
```

Implementation in Python

```
from agentspace import Agent
```

agents (objects that
have their own thread)

```
class MyAgent(Agent):
```

```
    def __init__(self, args):  
        process(args)  
        super().__init__()
```

invoked by timer

```
    def init(self):  
        self.attach_timer(1.0)
```

```
    def senseSelectAct(self):  
        print('hallo')
```

```
MyAgent()
```

Implementation in Python

```
from agentspace import Agent          agents (objects that
                                       have their own thread)

class MyAgent(Agent):

    def __init__(self, args):
        process(args)
        super().__init__()
                                       invoked by trigger

    def init(self):
        space.attach_trigger('a', self)

    def senseSelectAct(self):
        print('hallo')

MyAgent()
```