New Trends in CS

Embodiment from Engineer's Point of View

Andrej Lúčny

Department of Applied Informatics

FMFI UK Bratislava

lucny@fmph.uniba.sk

www.microstep-mis.com/~andy

Cognitivism

- Cognitivism is a kind of philosophy of mind which interpret mental functions as internal manipulation with symbols
- Cognitivists believed that mind is independent from biological hardware (wetware) and analogically can be created on different platforms
- Cognitivists looked for an universal algorithm implementing mind as a whole (example: General Problem Solver)

Postcognitivsm

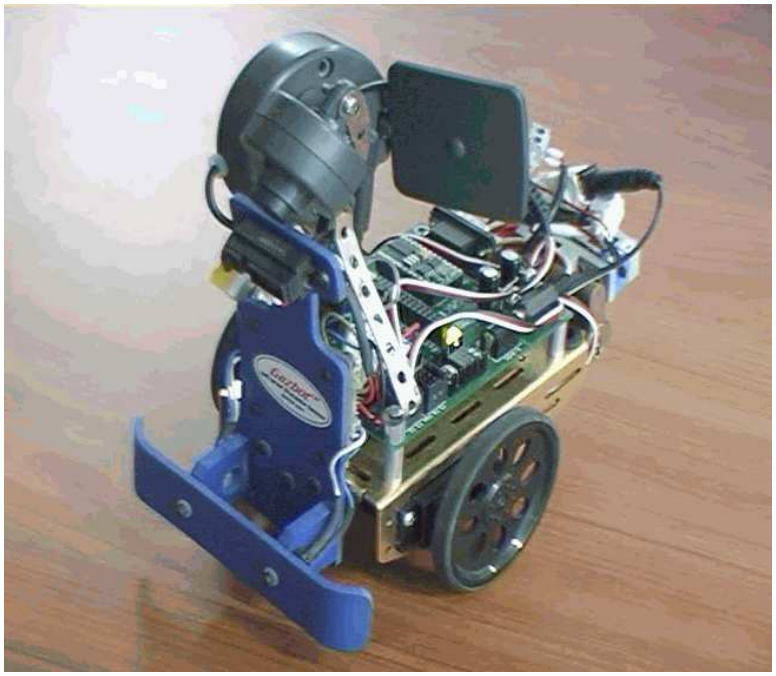
- Postcognitivism is going beyond cognitivism
- Postconitivists believe that wetware details are crucial – though can be simulated – even in form of symbols but not implementing a kind of logic
- Postcognitivist looks for particular solutions for particular processes in mind, they does not concern mind model as traditional algorithm but as modular computer system.

Embodiment

- Embodiment is a kind of postcognitivism which concentrate on significance of body for building of a mind model (a control structure) which manipulates with it.
- Embodiment is belief that mind is largely determined by body which it controls

The key idea of embodiment

- Imagine two different bodies, e.g. humanoid robot and two wheeled gear robot

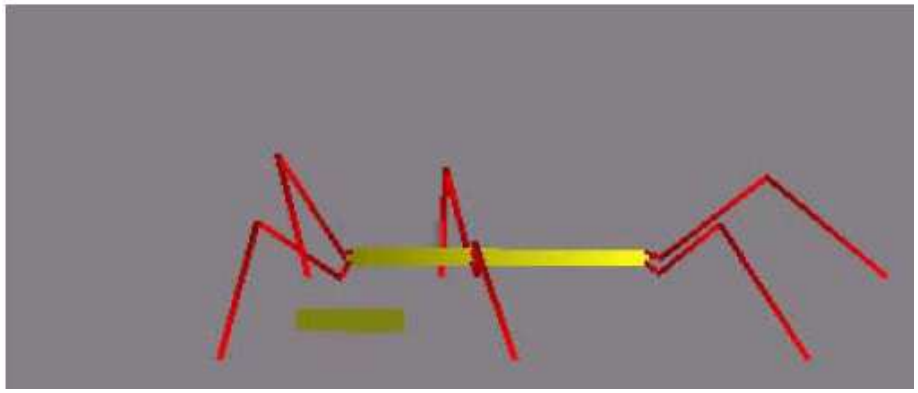


The key idea of embodiment

- cognitivists would expect the same algorithm fed by little bit different data expressed in the same mechanism of representation
- postcognitivists would expect totally different systems implementing different methods, different data structures, different representation mechanisms, …

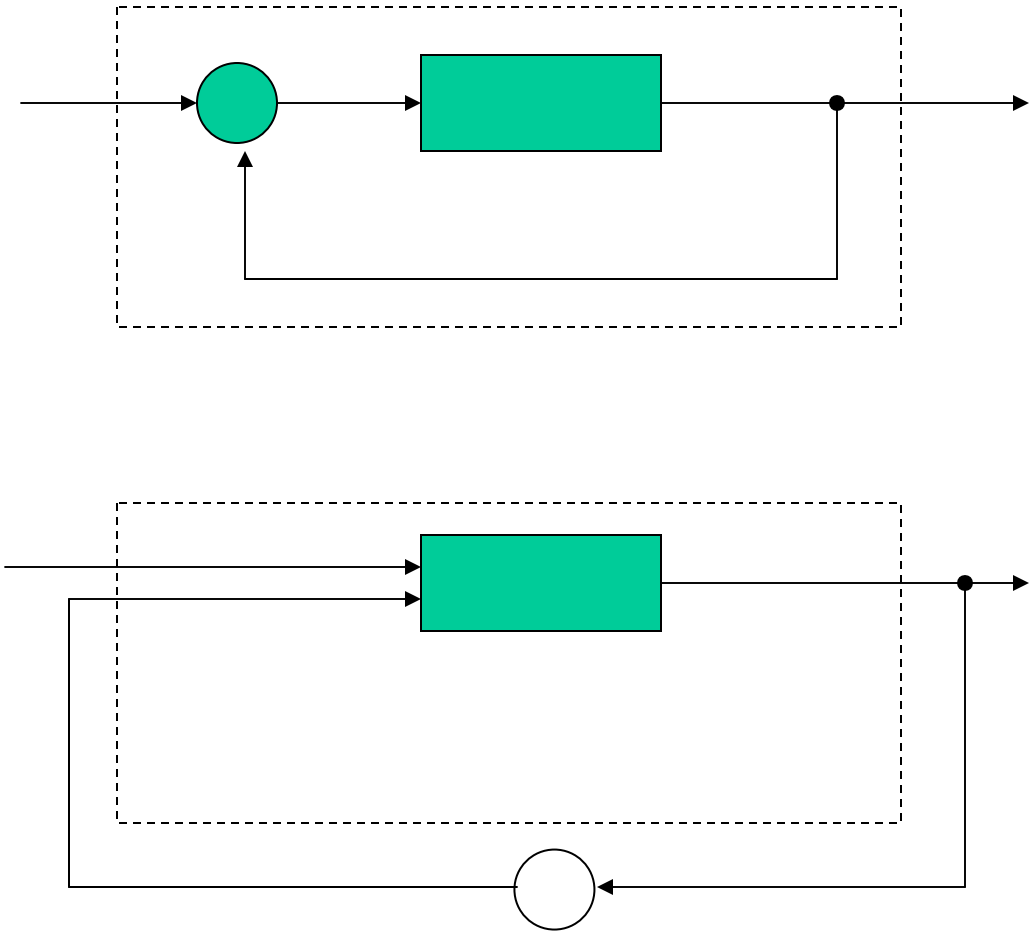
Why embodiment can be right?

- Because body can be used as a computational device
- It can replace difficult internal computation
- And different body provides different capabilities to be employed in this way



Why embodiment can be right?

System created according embodiment idea can have simpler internal structure



Engineer’s point of view

- How such kind of thinking can helps to engineer ?
- Imagine that the engineer is developing a control system for a mobile robot operating in dynamic environment
- His job is similar to emulation of mind – in fact of a part mind.

Engineer’s point of view

- Living creatures provide to engineers an inspiration – they minic what nature has already done (biomimetic approach)
- Question: When such inspiration is necessary ?
- Answer: When the intended behavior of the robot is really complex

Scalability problem

For many useful industrial applications it is sufficient to solve a particular problem

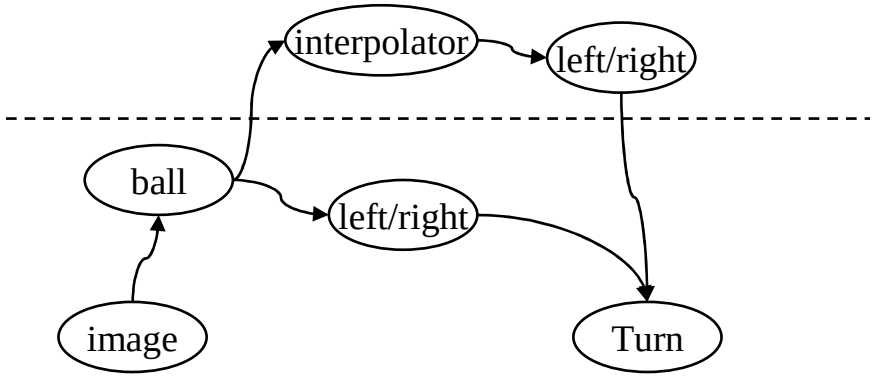
A simple pipeline is an appropriate architecture in this case

Scalability problem

However other tasks like control of mobile robots or simulated creatures require a more complex behavior and an advanced architecture

Minsky's approach to architecture

- system containing many parallel modules (agents, resources)
- Control = activation of a proper set of modules at a proper situation



Motivation

- How can we provide the proper activation?
- Let us look to the most complex systems we are able to observe – to living creatures

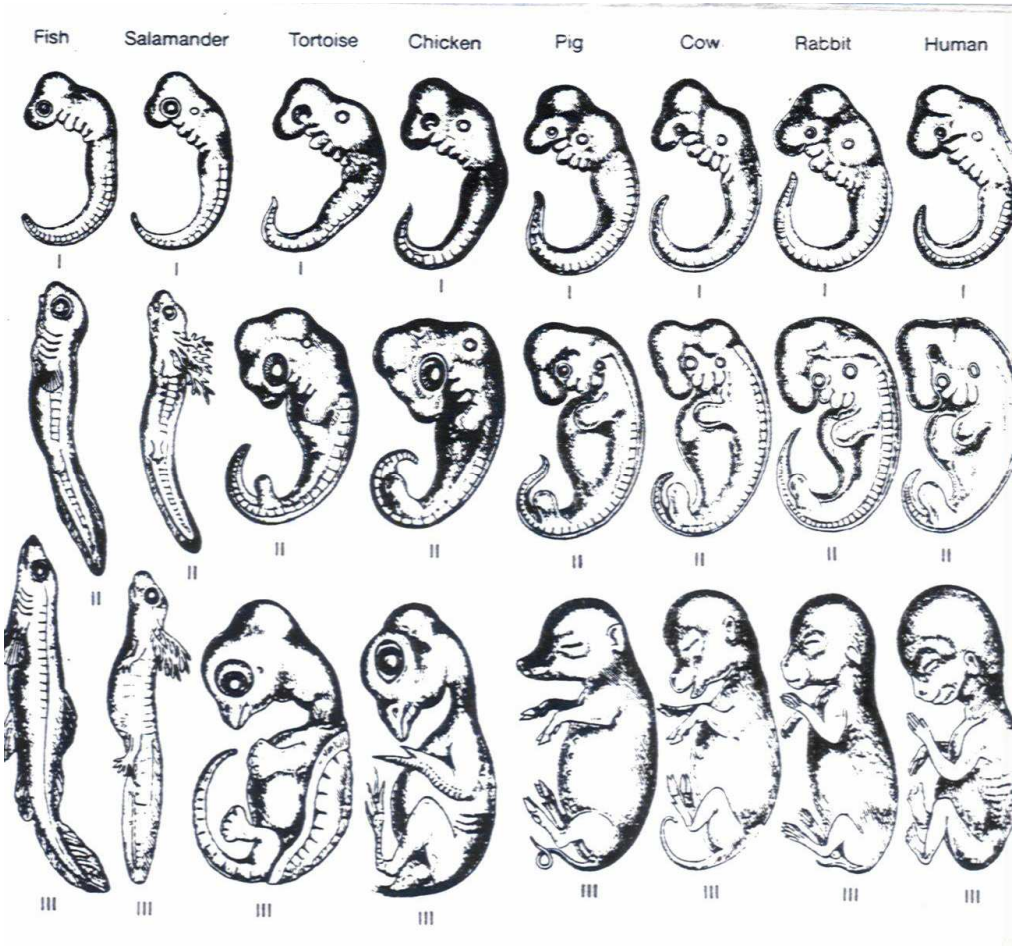
Fact 1

- Structure of living creatures has several typical features: parallelism and hierarchy
- Organization is based rather on regulation than activation

if one severs an eel's head from its spinal cord, the eel does not stop its sinuous swimming but its movements become perfectly regular and continuous. It means its brain inhibits and regulates its spinal cord than controlling it directly

Fact 2

- Structure of a living creature is a result of Darwinian evolution
- Hierarchy of structure copies steps of evolution



Adaptive systems versus incremental development

Engineer has two choices:

- To specify structure and let it to evolve population of such structures under simulated selection pressure
- To specify structure and perform hand-made evolution by its incremental development

Adaptive systems versus incremental development

- So far, adaptive system (reinforcement learning, neural networks, hidden Markov models, … have big troubles with scalability - i.e. which the primary problem which want to solve by them (!)

- Incremental development has been successfully used (1986 ALLEN, 1993 COG, …) - subsumption and its derivates

Subsumption

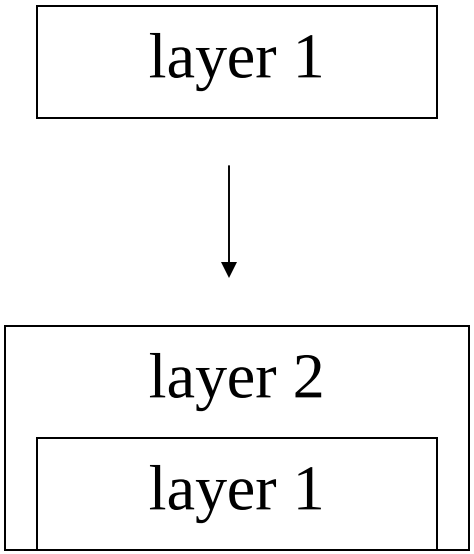
- It is a method for engineering of artificial systems with complex behavior
- It was proposed by R. Brooks in the mid-eighties

- It mimics simplified biological evolution

Subsumption

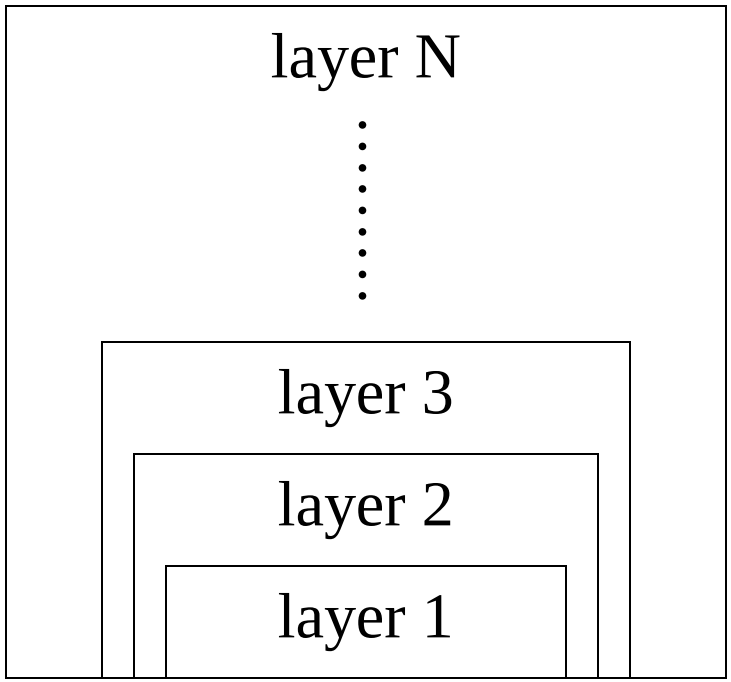
It is based on the evolutionary fact that any complex control has an origin in a simpler ancestor

The relation between the ancestor and its descendant is simplified here in such a way that the descendant contains exactly the same control mechanism as the ancestor, enriched just by an additional layer of control.



Subsumption

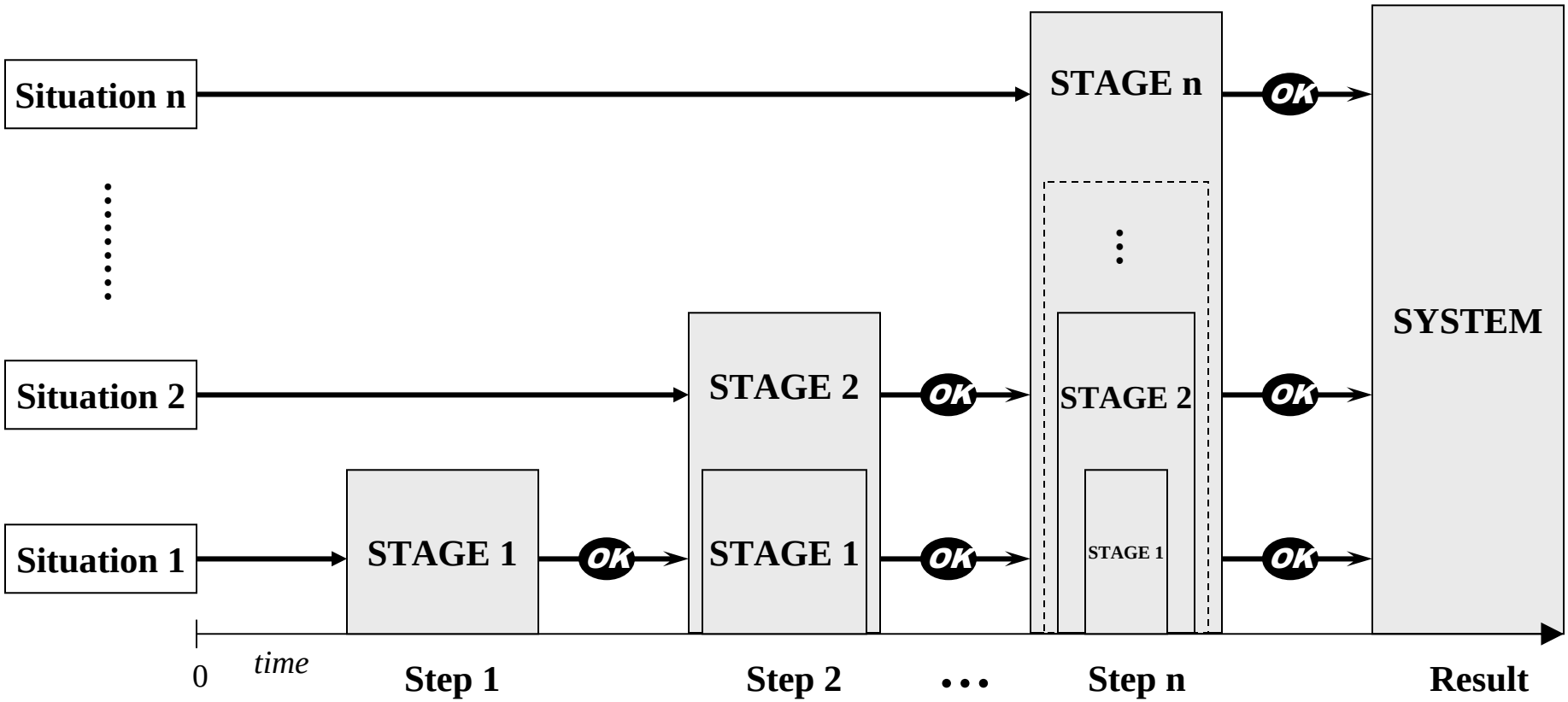
- In other words, the descendant mechanism subsumes the mechanism of its ancestor; therefore the principle is called *subsumption*.



Development by subsumption

- at the first we design suitable sensors and actuators which are expected to be sufficient.
- then we imagine a sequence of evolutionary steps which could result in the desired control starting from a simple base.
- we then incrementally develop each step as an additional layer to the previous simpler version.
- In doing so, each step brings a set of new features, but causes no harm to features which have been already implemented.

Development by subsumption



Situatedness

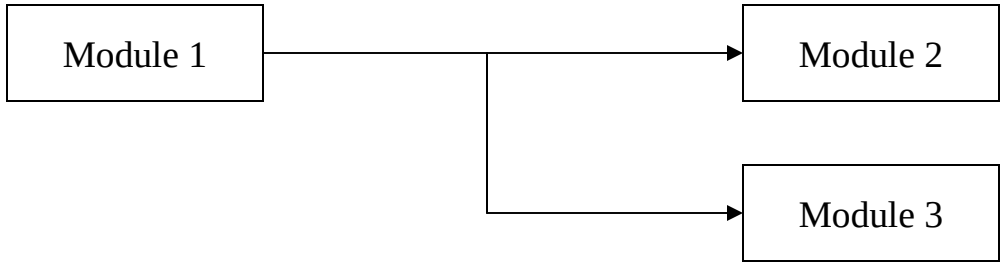
- It is recommended to design the evolutionary steps in such a way that each step corresponds to the desired control under simplified conditions.
- When the real situation is as simple as concerned for a particular step, it will be handled only by the corresponding layer and layers which are (evolutionary) older.
- Getting to more and more difficult situation, newer and newer levels are activated to influence the resulting control.

Appropriate modularity required

- However, how could the newer levels influence the older ones?
- The older levels have been designed for a particular purpose and have no interfaces for future development!
- The answer is: they have to have modular structure which enables it.

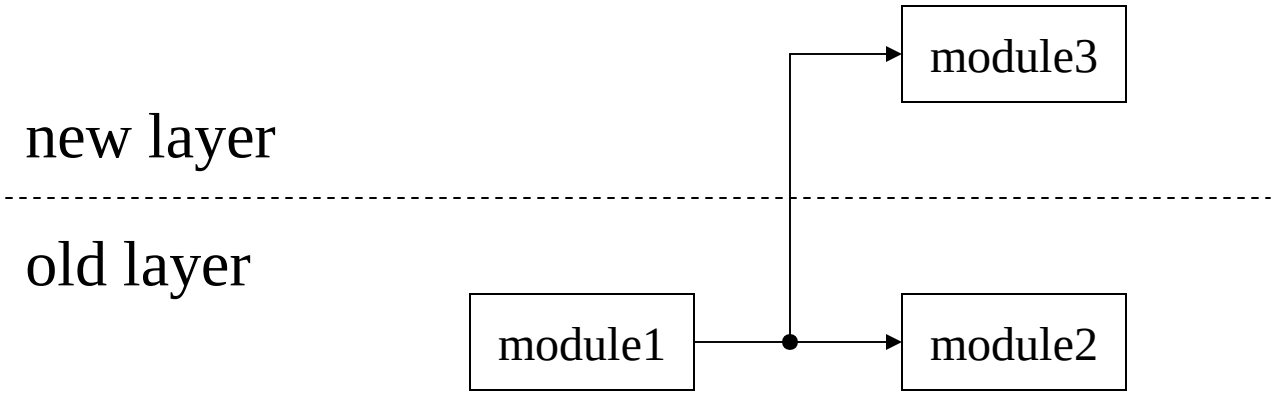
Subsumption architecture

- level consists of quite simple modules
- these modules communicate by messages sent through wires



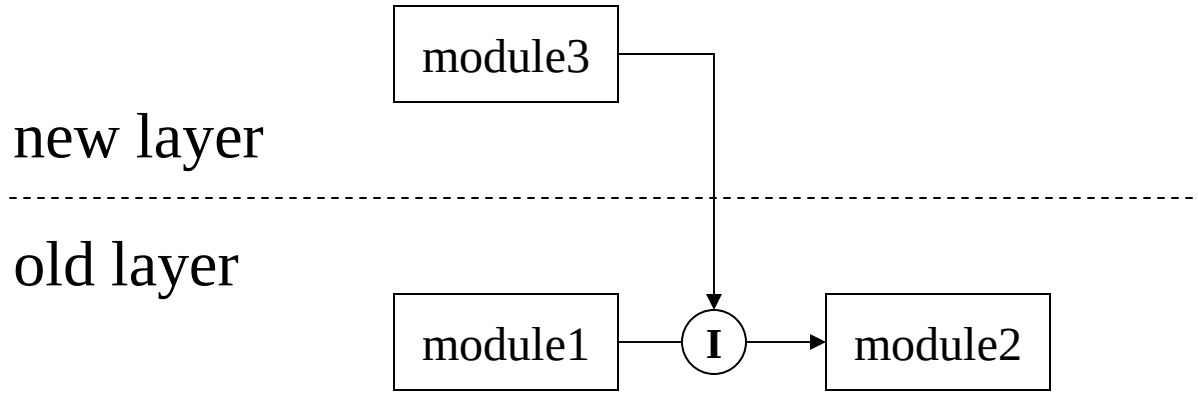
Monitoring

- the newer level can monitor messages communicated between modules in the older level by connecting to the same wire.



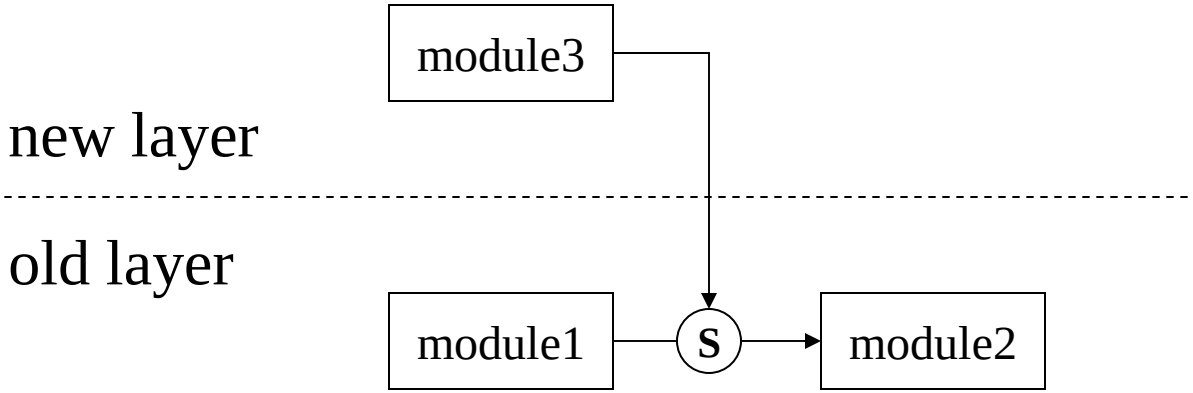
Inhibition

- it can also inhibit the communication by temporary interruption of the wire



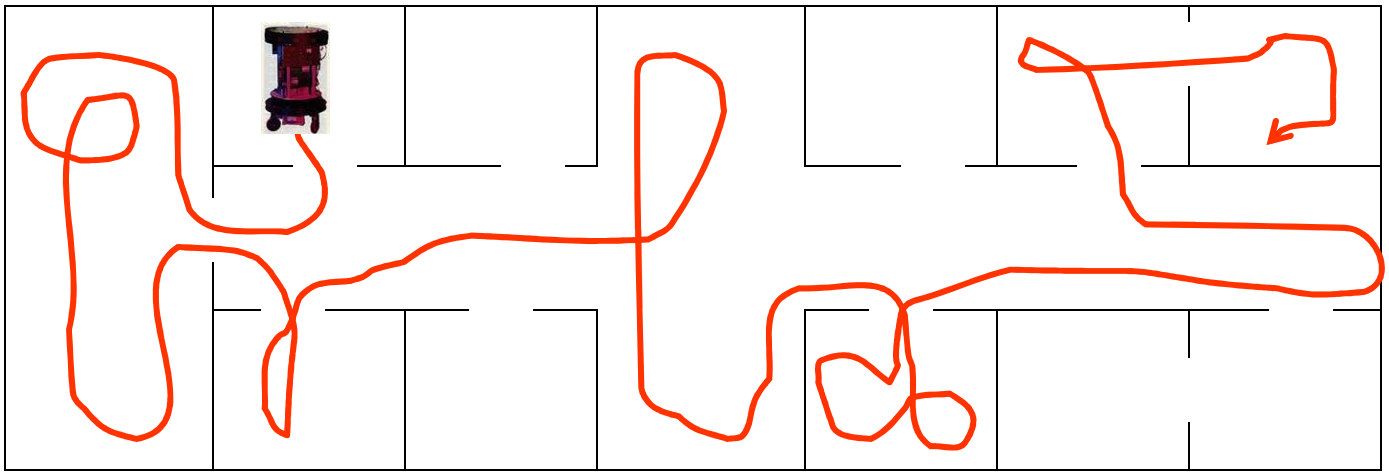
Suppression

- even it can replace communicated messages



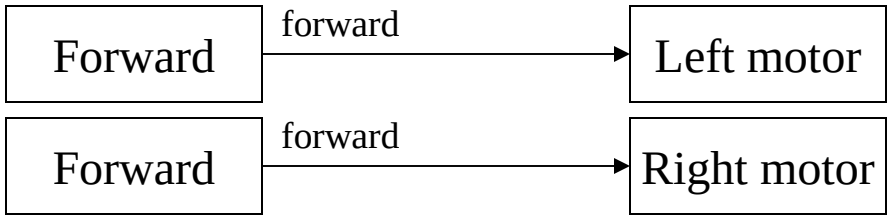
Example

- Two wheeled robot navigating in a bureau (ALLEN 1986)



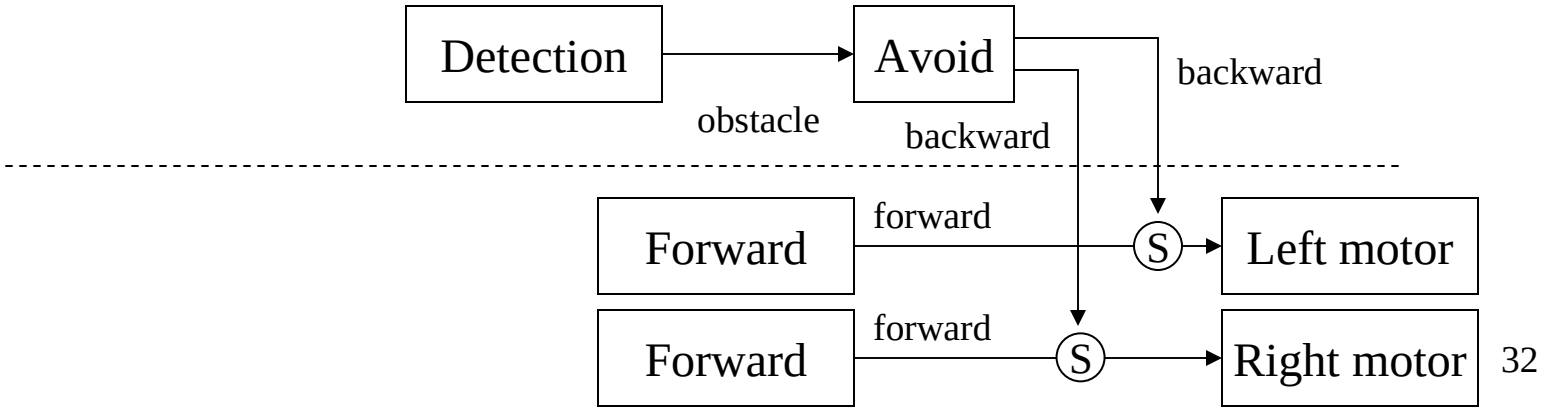
Example – step 1

- *we start with robot which just goes forward*



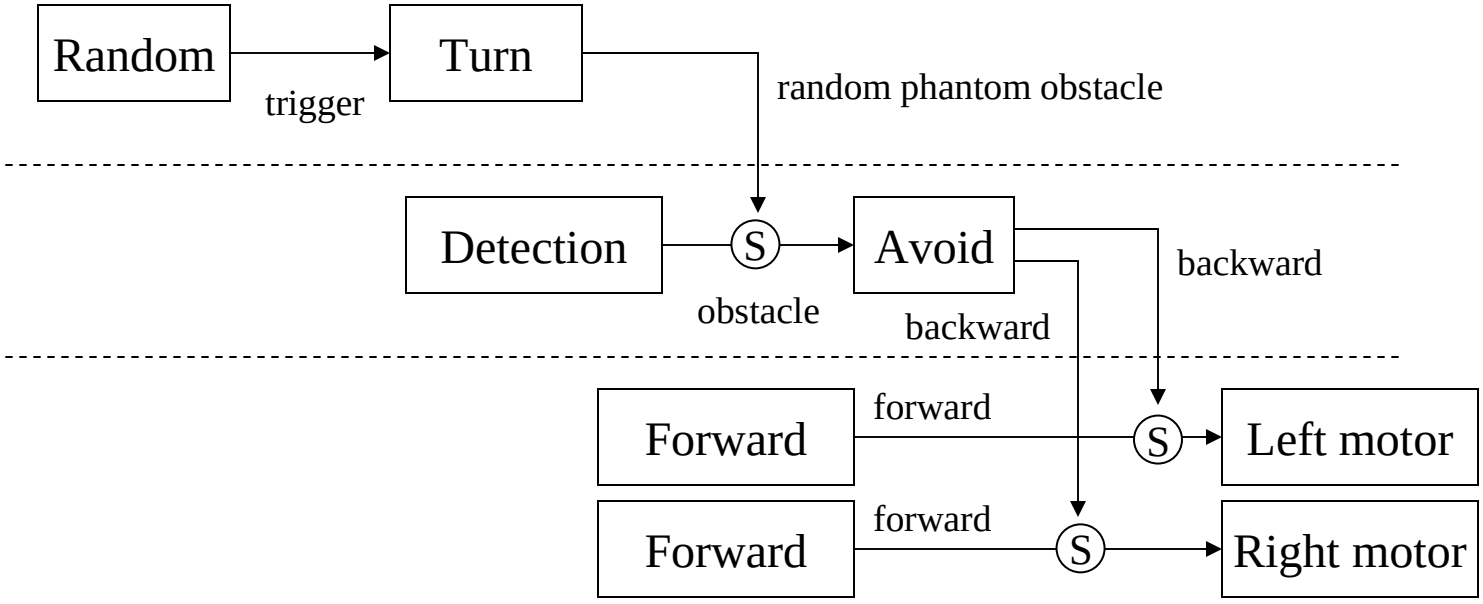
Example – step 2

- *Then we add a layer which recognizes obstacles and while they are detected, the layer replaces messages for one wheel to backward. As a result, the robot does not collide.*



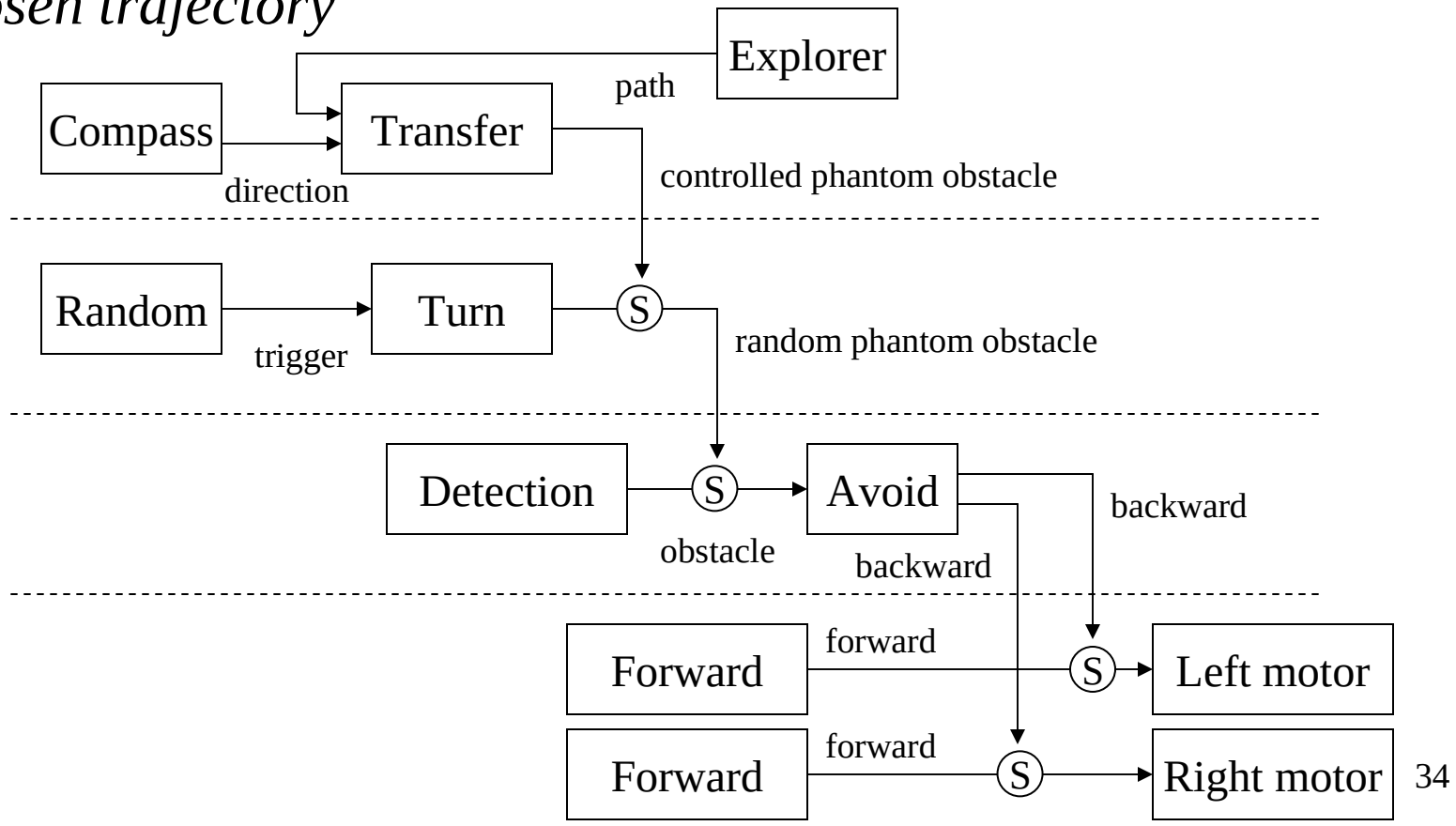
Example – step 3

- *However easily it can happen that it stays in the same region, moving in a cycle. Thus we add a layer which sometimes causes its random turn. We perform such a turn only when no obstacles are detected and we implement it just by apparent detection of obstacles*



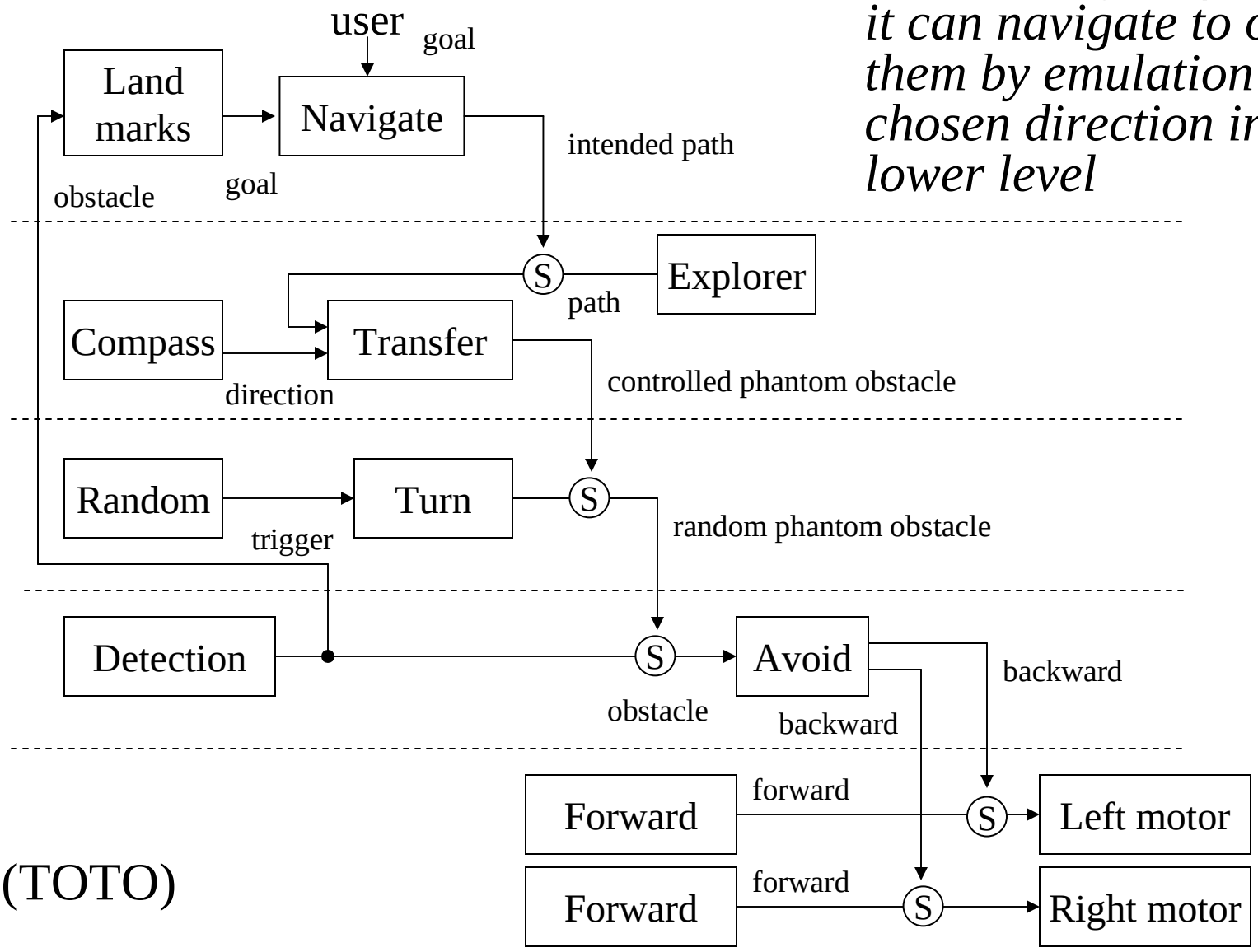
Example – step 4

- *another layer can a global movement in an absolute direction – from one part to another part of bureau. Once such direction is chosen, we implement its following by turns which are apparently random for the older layers, but in fact they keep the robot at the chosen trajectory*



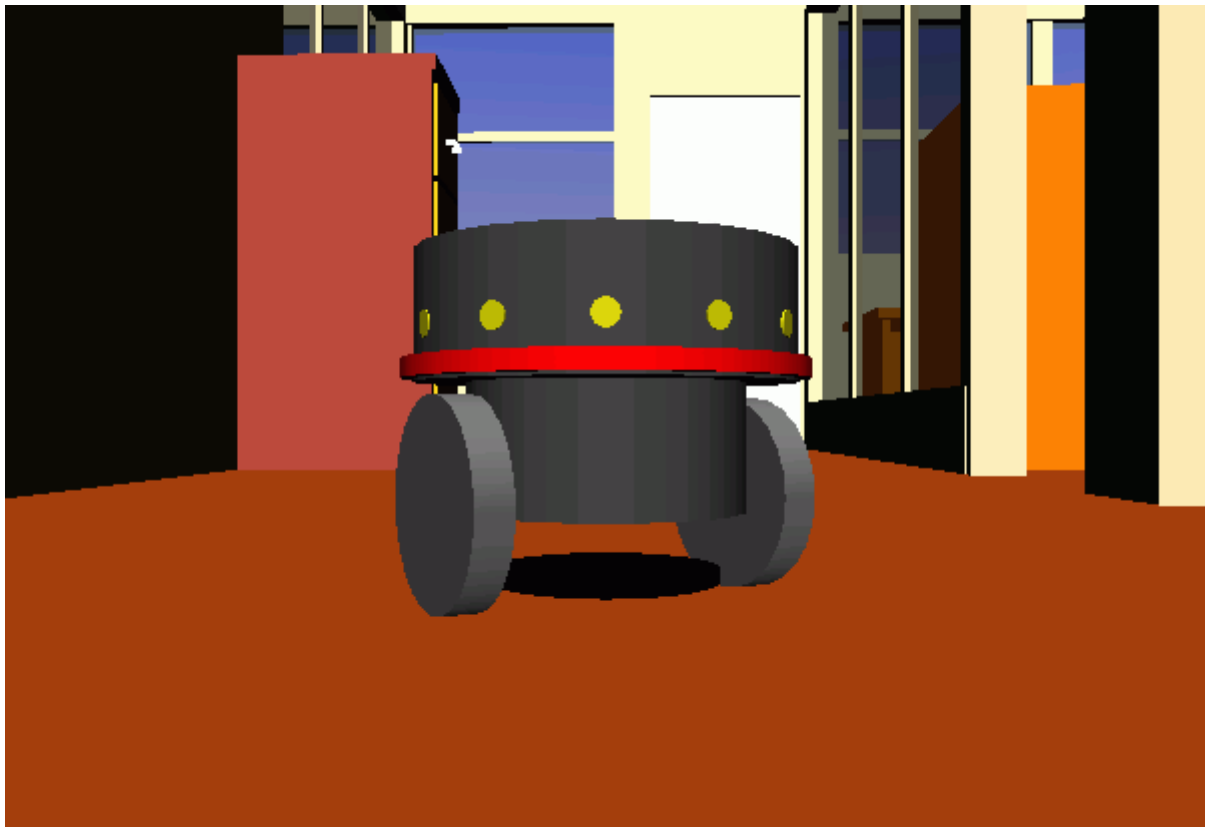
Example – step 5

- *Other level can detects landmarks and having received a goal from user it can navigate to one of them by emulation of the chosen direction in the lower level*



(TOTO)

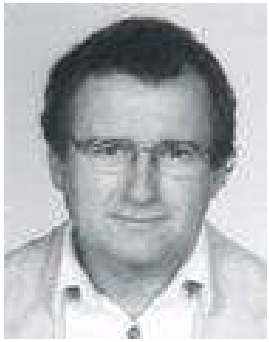
Example: reimplementation of ALLEN in VRML



Derivates of subsumption architecture

- *behavior-based architectures*: restriction of the influence to suppression of layer outputs (simplification)
- *fine-grained architecture*: accumulation of various actions generated by various levels is enabled (data fusion, more close to neural networks)
- many others

Derivate implemented in FMFI UK



- Long tradition of embodied approch for engineering due to Jozef Kelemen (1992 common work with Marvin Minsky)
- *agent-space architecture*: extension of the influence potential by modernizing the architecture which **overcomes the limitations of the hardware layout** typical for the original concept

  (Lucny 2004)

Another example

A mobile robot following a ping-pong ball

Conclusion

- Embodiment is an interesting inspiration for implementation of control structures of mobile robots
- Embodiment led to several architectures (e.g. subsumption architecture) and further will appear
- Its application is matter of art
- However, no better methods exist for engineering of systems with complex behavior (like foreseen vehicle)

Thank you !

Andrej Lúčny

Department of Applied Informatics

FMFI UK Bratislava

lucny@fmph.uniba.sk

www.microstep-mis.com/~andy