Artificial Life
Summer 2015

Subsumption Architecture

Master Computer Science [MA-INF 4201]
Mon 8:30 – 10:00, LBH, Lecture Hall III.03a

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn

Subsumption Architecture, 1985, R.Brooks

Rodney A. Brooks (1985)

„ *A Robust Layered Control System for a Mobile Robot* “

A.I Memo 864, 1985
Massachusetts Institute of Technology
Artificial Intelligence Laboratory

http://www.uv.edu.mx/mia/ingreso/documents/Brooks.pdf

Behavior Based Robotics

R. Brooks defines a number of requirements of a control system for an intelligent autonomous mobile robot.
They each put constraints on possible control systems that one might build and employ.

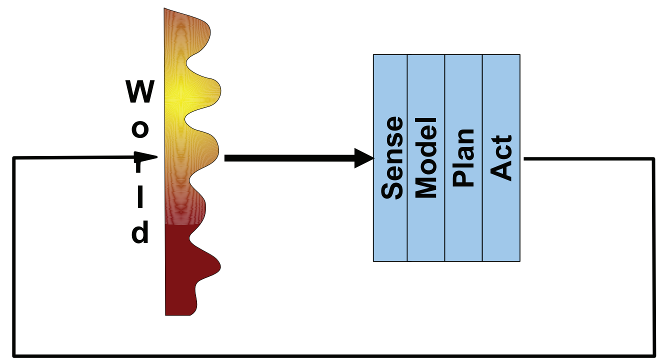
A robotics wish list for Santa Claus:

- Multiple Goals
- Multiple Sensors
- Robustness
- Additivity

See publication of R.Brooks (1985) for details.

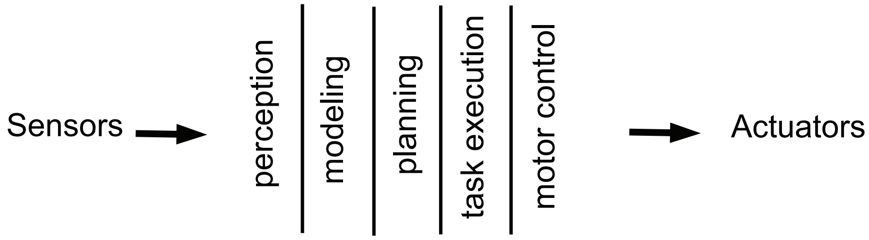
SMPA Architecture - reminder

The sensory input (**Sense**) obtained from the outer world is processed by the SMPA architecture onto commands controlling the robot/or the environment (**Act**).



SMPA Architecture - once more

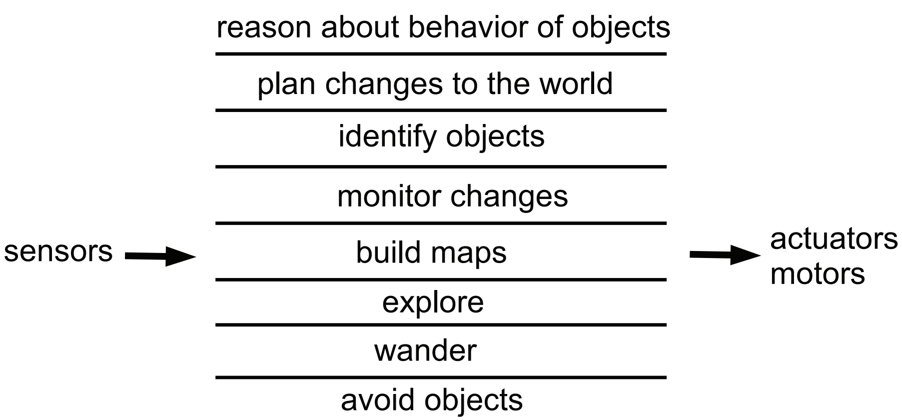
A traditional decomposition of a mobile robot control system into functional modules *(Brooks 1985)*



The classical SMPA architecture in a more detailed version:

Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*



Nine Dogmatic Principles

Brooks bases the design of the robot on **nine dogmatic principles**.

- Complex (and useful) behavior
- Things should be simple
- Map making is of crucial importance
- Three dimensional environment
- Relational maps for the robot
- No artificial environment for the robot, (no exact world model)
- Visual data for the robot (not just sonar ranger data)
- Self calibrating robot system, self calibration at all time
- Robots must be self sustaining

Levels of Competence, Levels of Behavior

The approach of Rodney Brooks decomposes the robot control problem into a number of ***levels of competence***.

A ***level of competence*** is an informal specification of a desired class of behaviors for the robot over all environments it will encounter.

A higher level of competence implies a more specified desired class of behaviors.

Thus the different levels of behavior are used to structure the robot controller.

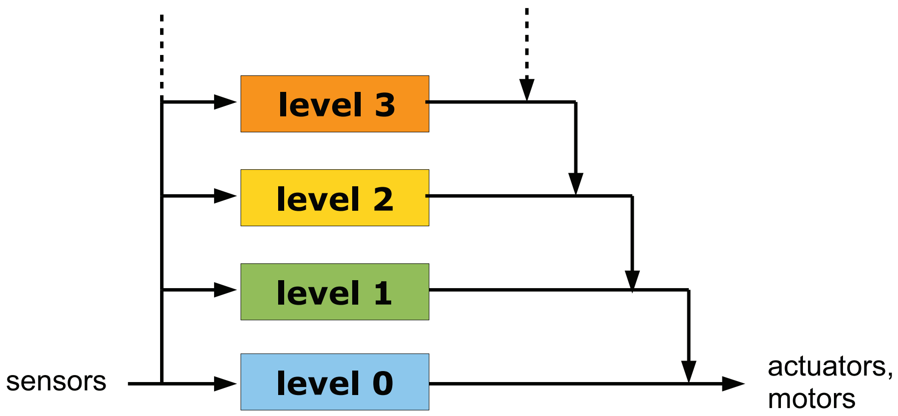
Levels of Competence, Levels of Behavior

The following 8 levels of competence were proposed (Brooks):

0. Avoid contact with objects
1. Wander aimlessly around (without hitting things)
2. Explore the world
3. Build a map, and plan routes within this map
4. Notice changes in this (static) environment
5. Reason about the world
6. Formulate and execute plans changing the world
7. Reason about the behavior of other objects in the world

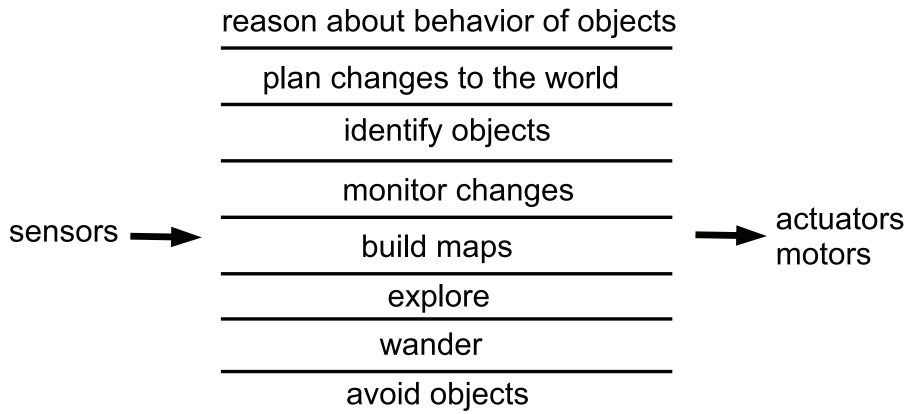
Layered Control

Control is layered with higher level layers subsuming the lower levels.



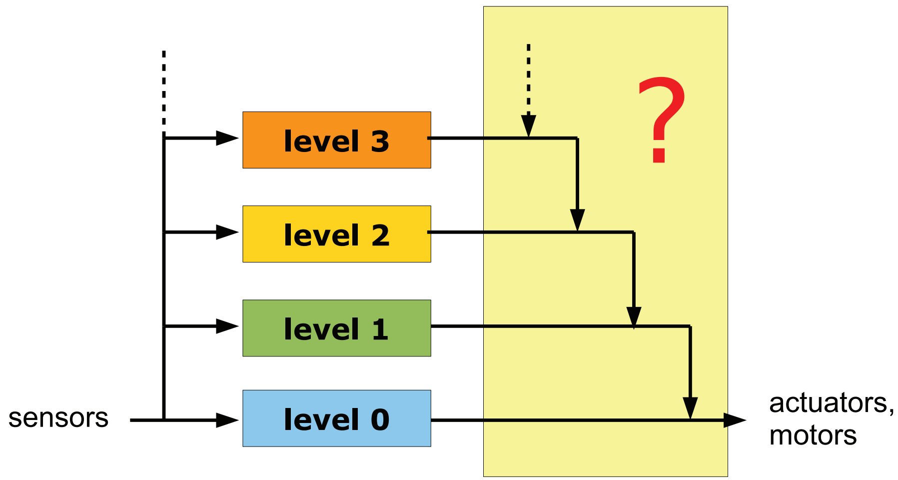
Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*



Layered Control

Control is layered with higher level layers subsuming the lower levels.



Subsumption Control

If necessary, or appropriate for the task, the higher level layers subsum the effect of the lower layers.

Thus, the higher level behaviors can dominate the lover level behaviors.

And as an additional positive effect:

If, in any case some part of this layered control structure is not producing any commands (busy, or damaged, or ...) the lower levels would still be operational and would thus implement a working controller.

Implementing the Layers

The layers are build up following a modular approach. They consist of simple structured control units (modules).

The modules are as simple as a finite state machine.

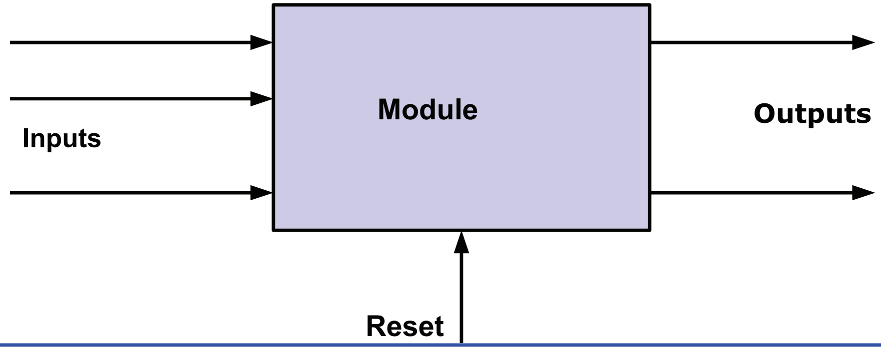
(In fact, R.Brooks has explicitly designed them to be finite state machines, for easily implementing them in hardware.)

A module has input and output lines connecting the modules among each other.

The interconnected modules combine to a network.

Structure of a Module

Each module is implemented as a **finite state machine** augmented with some instance variables

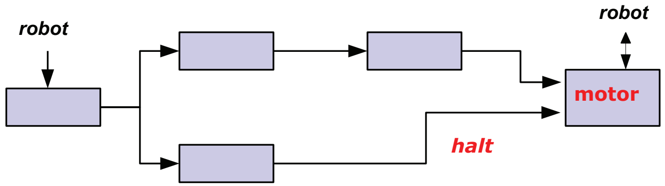


Modules are Connected

The output of one module can be connected to the input of one of the other modules.

Each module is responsible for a special subtask:
e.g. ***motor*** control

The signals between the modules correspond to messages between the subtasks: e.g. ***halt***

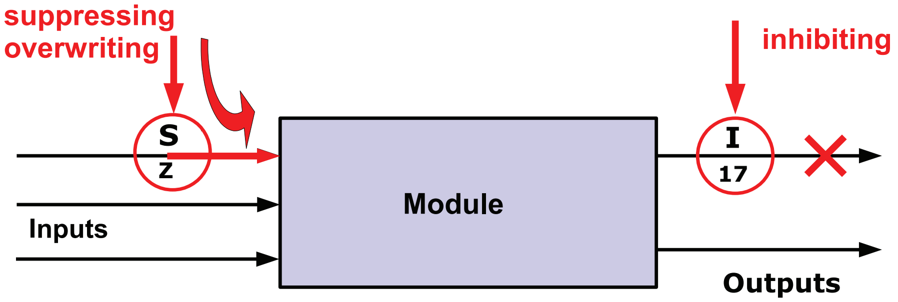


Suppress and Inhibit

Beside the connection of input and output lines between the modules, the output of a module can alter the signal on another connection by 2 methods:

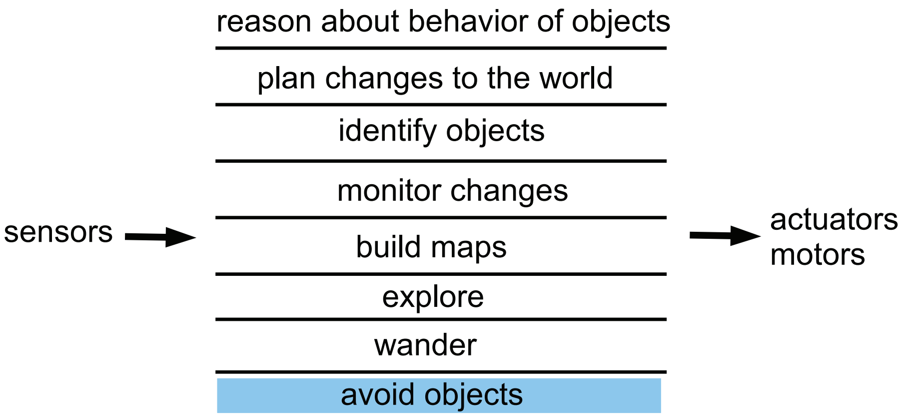
Suppressing: overwriting signal lines for **z** time steps

Inhibiting: canceling signals for **z** time steps.



Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*



Level 0: *Avoid Objects*

The designated behavior for level 0 is to:
Avoid the contact between the robot and any objects.

Therefore, two mechanisms have been implemented:

Stop the robot if a collision is detected: **halt**

If something approaches the robot - move away: **runaway**

Level 0: *Avoid Objects*

sonar:
the sonar module takes a vector of sonar readings, filters them for invalid readings, and effectively produces a robot centered map of obstacles in polar coordinates.

Level 0: *Avoid Objects*

motor:
the motor module communicates with the actual robot.
The communication is going in both ways, so the motor controller can be implemented as closed loop control.
It accepts a command specifying angle and turning velocity, magnitude of forward movement and velocity.

Level 0: *Avoid Objects*

The designated behavior for level 0 is to:
Avoid the contact between the robot and any objects.

Therefore, two mechanisms have been implemented:

Stop the robot if a collision is detected: **halt**

If something approaches the robot - move away: **runaway**

Level 0: *Avoid Objects*

collide?:
the collide module monitors the sonar map and if it detects objects dead ahead it sends a signal on the halt line to the motor module. The collide module does not know or care whether the robot is moving Halt messages sent while the robot is stationary are essentially lost.

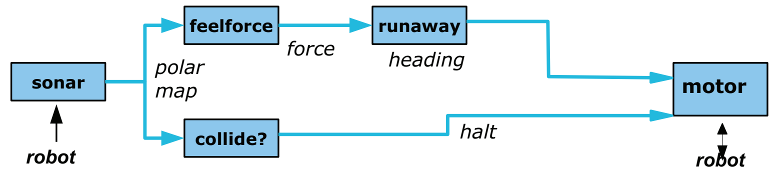
Level 0: *Avoid Objects*

The designated behavior for level 0 is to:
Avoid the contact between the robot and any objects.

Therefore, two mechanisms have been implemented:

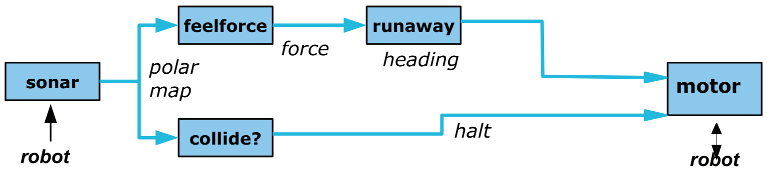
Stop the robot if a collision is detected: **halt**

If something approaches the robot - move away: **runaway**



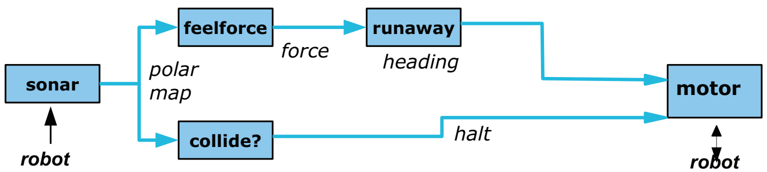
Level 0: *Avoid Objects*

sonar:
the sonar module takes a vector of sonar readings, filters them for invalid readings, and effectively produces a robot centered map of obstacles in polar coordinates.



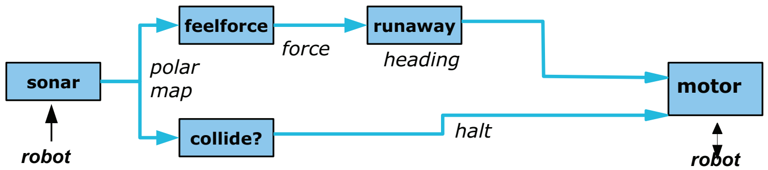
Level 0: *Avoid Objects*

feelforce:
the feelforce module sums the results of considering each detected object as a repulsive force, generating a simple resultant force.



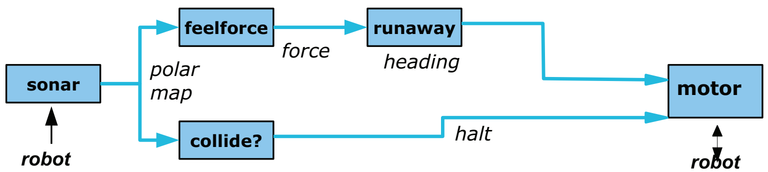
Level 0: *Avoid Objects*

runaway:
the runaway module monitors the force produced by the sonar detected obstacles and sends command (e.g. heading) to the motor module if it ever becomes significant.



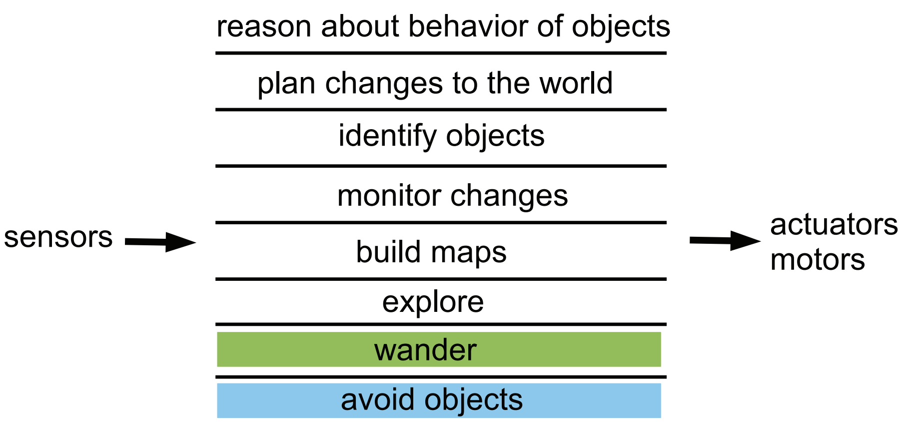
Level 0: *Avoid Objects*

motor:
the motor module communicates with the actual robot.
The communication is going in both ways, so the motor controller can be implemented as closed loop control.
It accepts a command specifying angle and turning velocity, magnitude of forward movement and velocity.



Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*



Level 1: *Wander*

The designated behavior for **level 1** is to:

Wander around

The first level layer of control, when combined with level 0, imbues the robot with the ability to wander around aimlessly without hitting obstacles.

This control level relies in a large degree on the level 0 aversion to hitting obstacles.

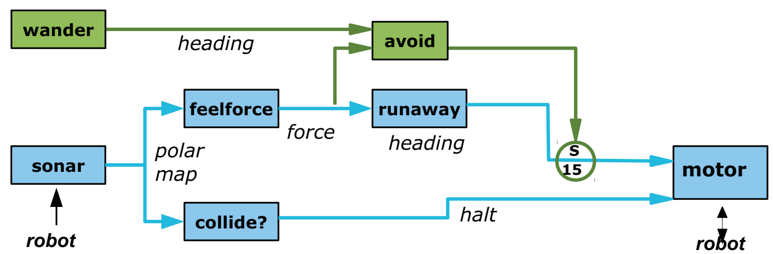
Level 1: *Wander*

The designated behavior for **level 1** is to:

Wander around

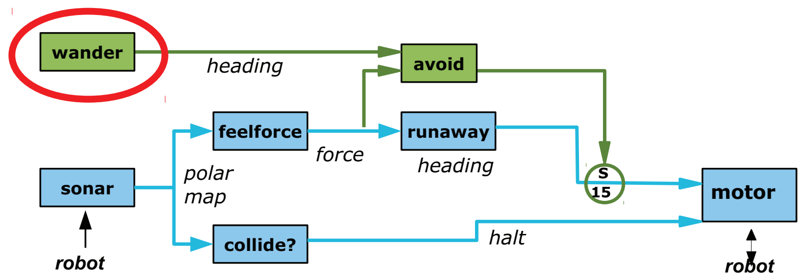
The first layer level of control will add new modules to the existing layer 0.

Level 1 modules: **wander** and **avoid**



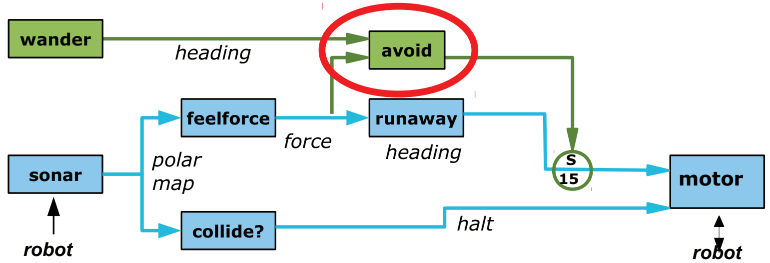
Level 1: *Wander*

wander:
the wander module generates a new heading for the robot in a fixed time interval (e.g. every 10 seconds or so).



Level 1: *Wander*

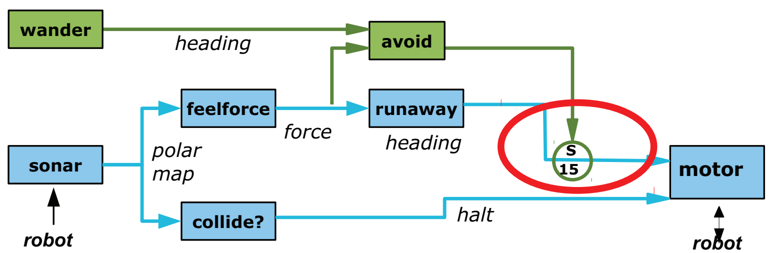
avoid:
the avoid module takes the result of the force computation from level 0, and combines it with the desired heading to produce a modified heading which usually points in roughly the correct direction, but perturbed to avoid any obvious obstacles.



Level 1: *Wander*

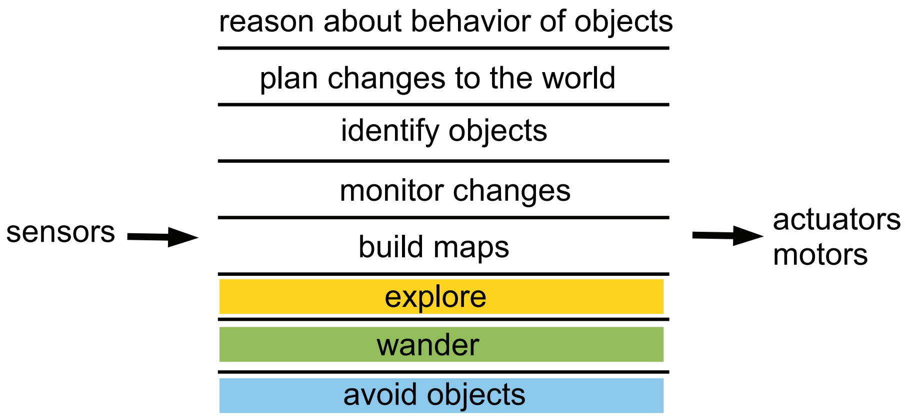
avoid:
the output form the avoid module suppresses the output from the runaway module of level 0, before entering the motor module.

The signal from level 0 is suppressed (overwritten, replaced) by the level 1 commands for 15 time steps



Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*



Level 2: *Explore*

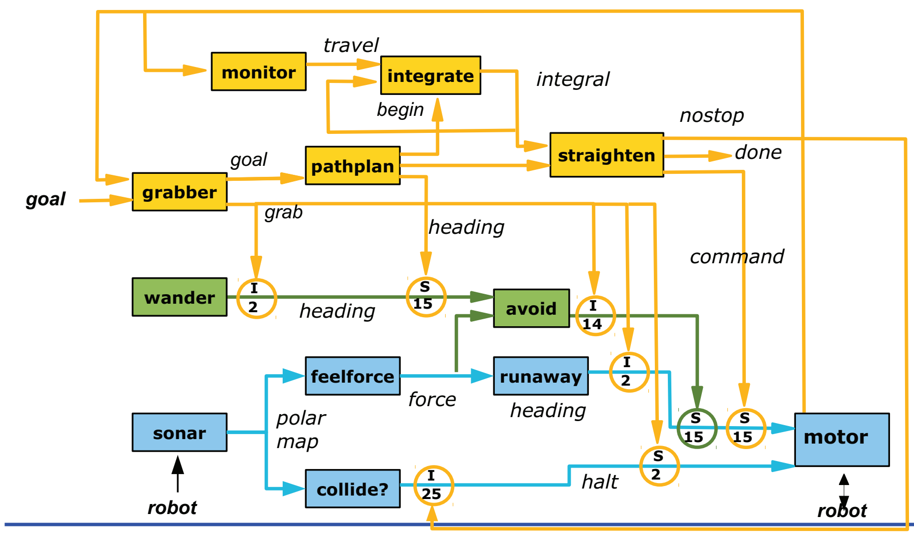
The designated behavior for **level 2** is to:

Explore the environment

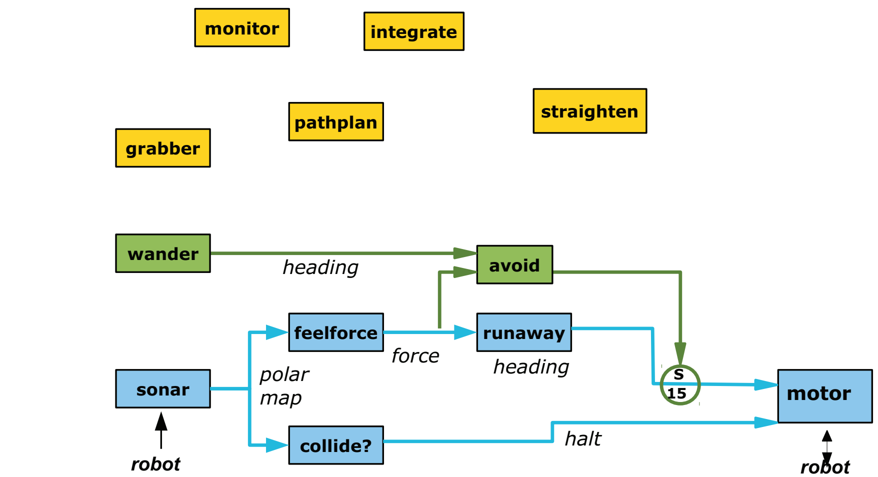
Level 2 is meant (by R. Brooks) to add an exploratory mode of behavior to the robot, using visual observations to select interesting places to visit.
(At the time of the publication in 1985, R.Brooks could only implement the non-vision aspects of this level)

A total of 5 new modules, 10 new signals, 17 connections, 4 places of signal inhibition and 3 places of signal supression are added in level 2 to the existing level 0 and level 1.

Level 2: *Explore*

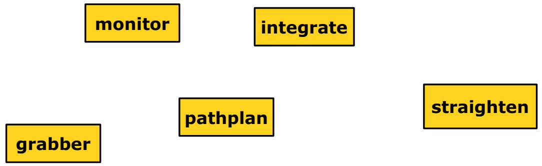


Level 2: *Explore*



Level 2: *Explore*

Level 2 implies the following 5 new modules:



grabber module
monitor module
integrate module
pathplan module
straighten module

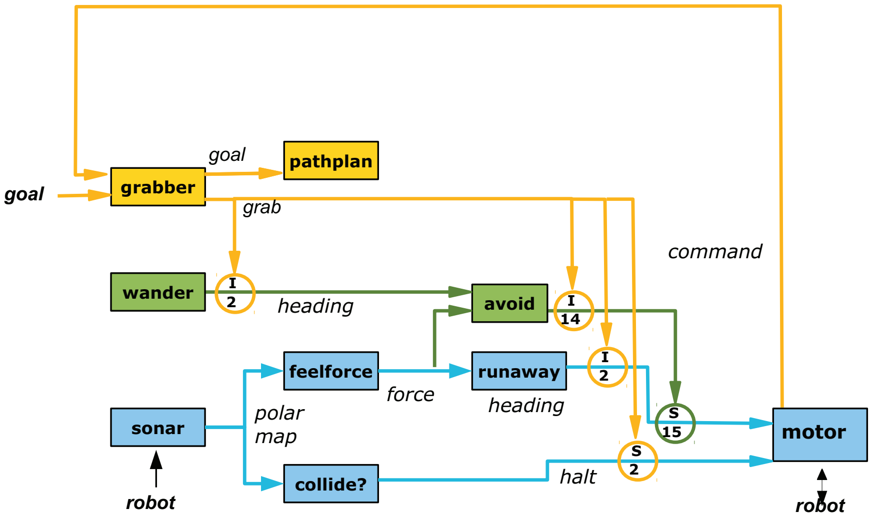
Level 2: *Explore*

grabber:
the grabber (video grabber) module ensures that level 2 has control of the motors by sending a halt signal to the motor module, then temporarily inhibiting a number of communication paths in the lower levels so that no new actions can be initiated, and waiting for the motor module to indicate that it is no longer controlling the robot motion.

At this point the (visual) sensor will be giving stable readings sufficient to plan a detailed motion, so a goal can be sent to the pathplan module.

Please notice: while stopping the other levels, the robot will be unable to avoid approaching objects for a brief 2 seconds.

Level 2: *Explore*



Level 2: *Explore*

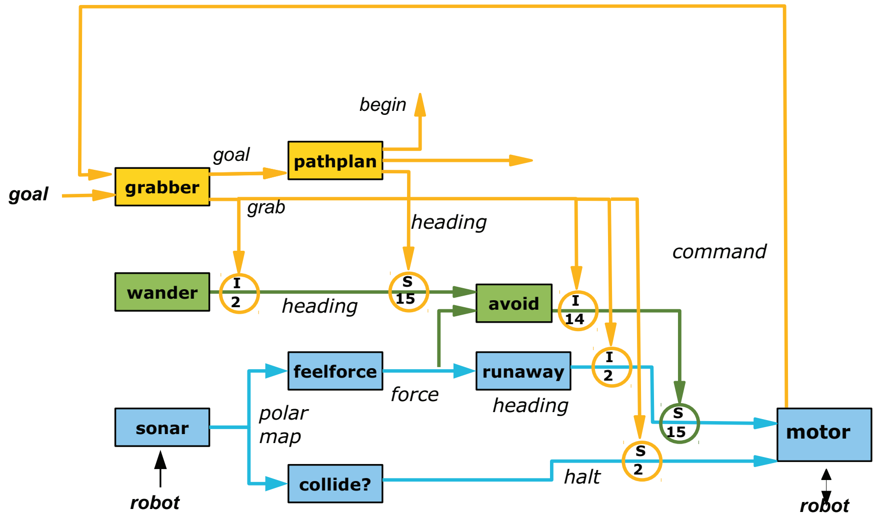
pathplan:
the pathplan module takes a goal specification and attempts to reach that goal.

To do this, it sends headings to the avoid module, which may perturb them to avoid local obstacles, and monitors its integral input which is an integration of actual motions.

The messages to the avoid module suppress random wanderings of the robot, so long as the higher level planner remains active.

When the position of the robot is close to the desired position it outputs the goal to the straighten module.

Level 2: *Explore*

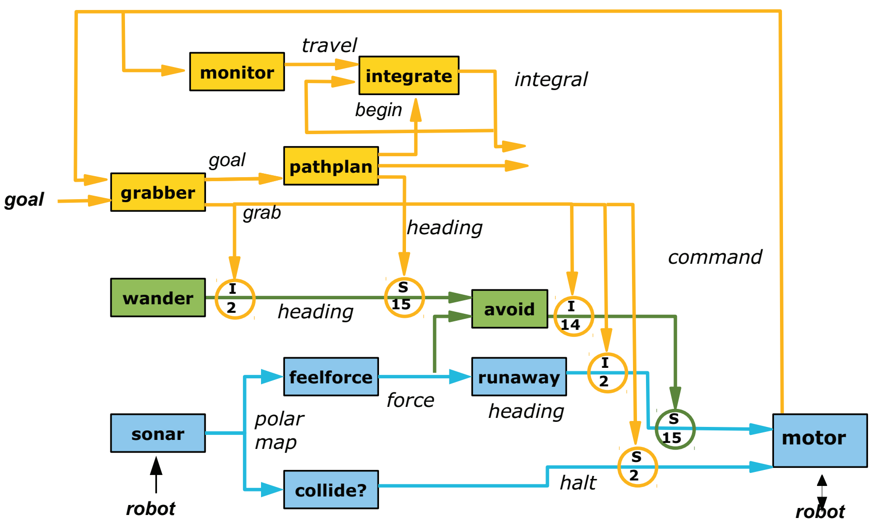


Level 2: *Explore*

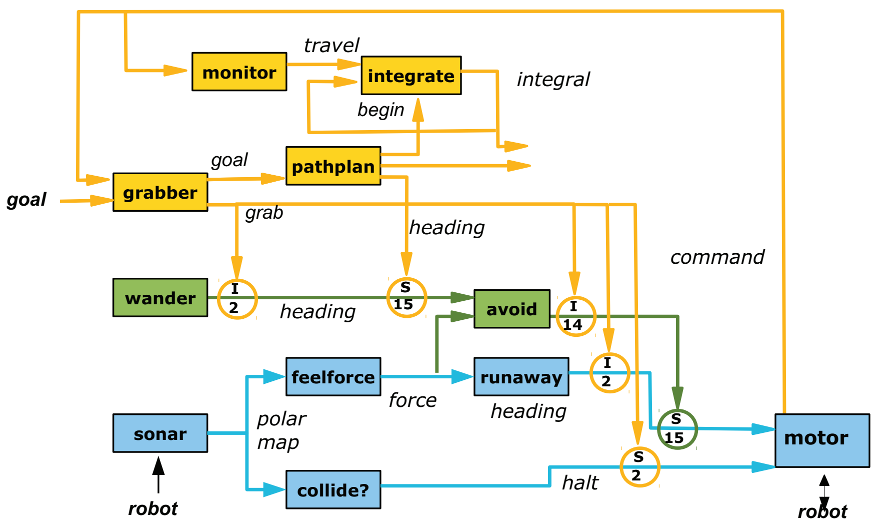
monitor:
the monitor module continually monitors the status of the motor module. As soon as the module becomes inactive the monitor module queries the robot via a direct connection to get a reading from its shaft encoders on how far it has traveled

Thus it is able to track each motion, whether it terminated as intended or if there was an early hat due to looming obstacles.

Level 2: *Explore*



Level 2: *Explore*



Level 2: *Explore*

integrate:
the integrate module accumulates reports of motions from the monitor module and always sends its most recent result out on its integral line.

It gets restarted by application of a signal to its reset input.

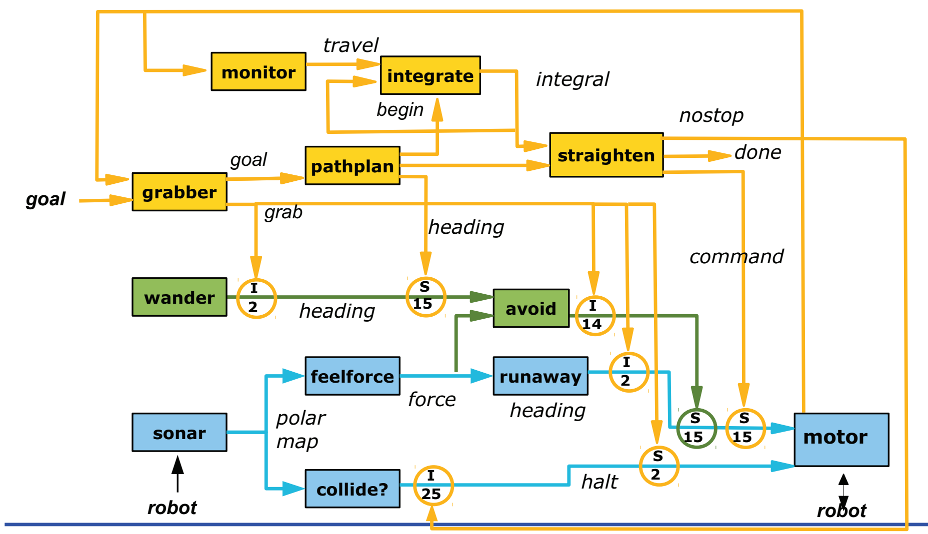
Level 2: *Explore*

straighten:
the straighten module is responsible for modifying the final orientation of the robot.

Any command with just an angular heading will not get through the avoid module as it filters out small motions, since the pressure of forces from remote obstacles would otherwise make it „buzz“ with large turns and small forward motions.

The straighten module sends its commands directly to the motor module, and monitors the integral line to make sure it is successful. For good measures it also inhibits the collision reflex, since there is no danger of a collision from forward motion (straighten only turns the robot).

Level 2: *Explore*



Task Achieving Behaviors

A decomposition of a mobile robot control system based on task achieving behaviors *(Brooks 1985)*

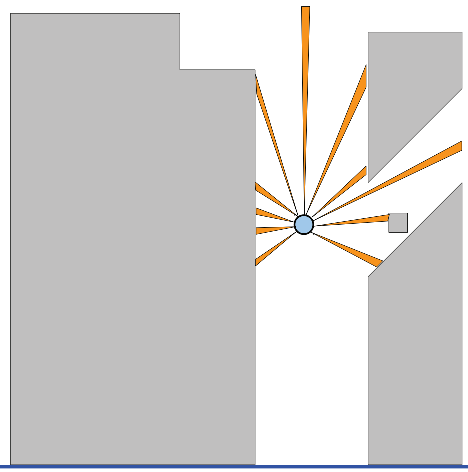
sensors →

- reason about behavior of objects
- plan changes to the world
- identify objects
- monitor changes
- build maps
- explore
- wander
- avoid objects

→ actuators motors

Subsumption Architecture

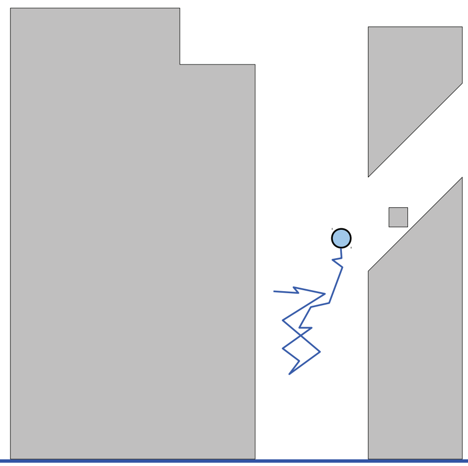
R.Brooks' test world for the subsumption controlled robot.



Robot is equipped with sonar based distance sensors.

Subsumption Architecture

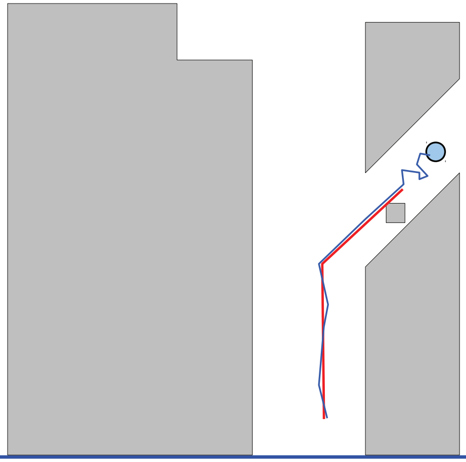
R.Brooks' test world for the subsumption controlled robot.



Under levels 0 and 1 control the robot wanders around aimlessly.
It does not hit obstacles

Subsumption Architecture

R.Brooks' test world for the subsumption controlled robot.

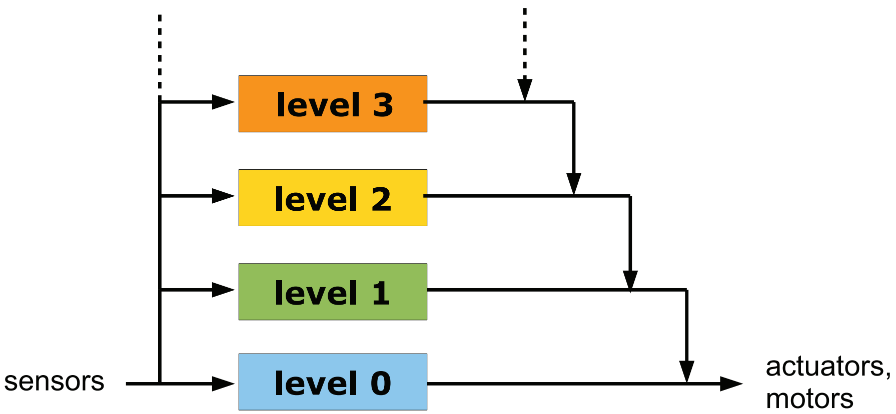


With level 2 control the robot tries to achieve command goals. The nominal goals are the two straight (red) lines.

After reaching the second (final) goal the robot reverts to aimless level 1 behavior.

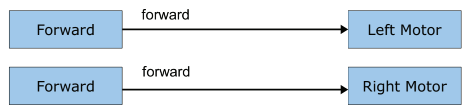
Layered Control

Control is layered with higher level layers subsuming the lower levels.



Subsumption Architecture, Example

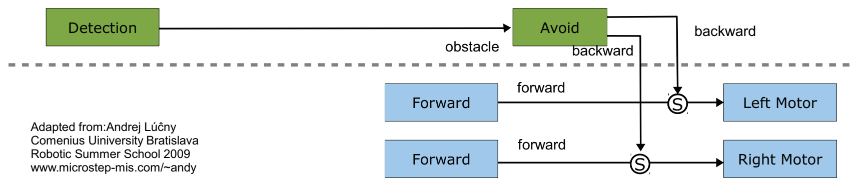
Start with a robot which can just go forward.



Adapted from:Andrej Lúčny
Comenius University Bratislava
Robotic Summer School 2009
www.microstep-mis.com/~andy

Subsumption Architecture, Example

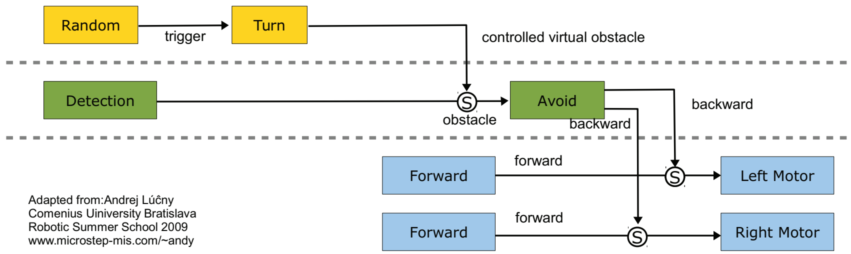
Add a layer which recognizes obstacles and while they are detected, the layer generates a message for one wheel to go backward to prevent the robot from colliding.



Adapted from:Andrej Lúčny
Comenius University Bratislava
Robotic Summer School 2009
www.microstep-mis.com/~andy

Subsumption Architecture, Example

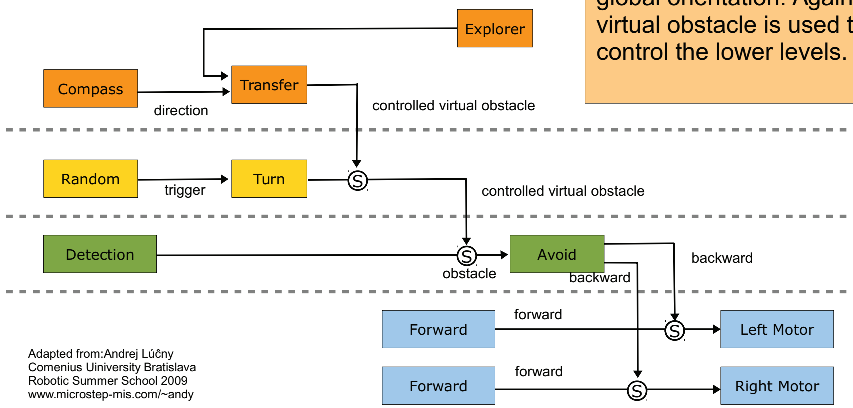
Add a layer which sometimes causes the robot to perform random turns. Making the lower level believe that there is an obstacle it is caused to do an avoidance movement.



Adapted from:Andrej Lúčny
Comenius University Bratislava
Robotic Summer School 2009
www.microstep-mis.com/~andy

Subsumption Architecture, Example

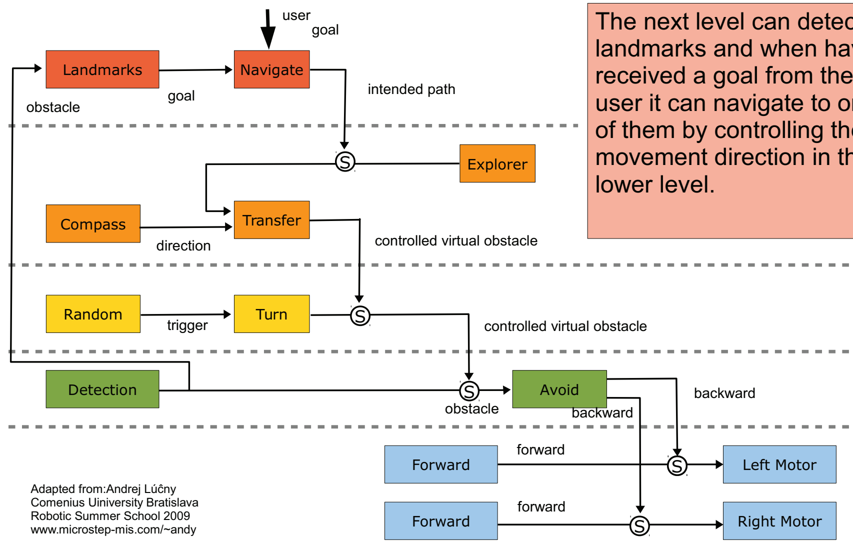
The next layer implements an exploratory behavior by generating sequences of headings adjusted with a global orientation. Again a virtual obstacle is used to control the lower levels.



Adapted from:Andrej Lúčny
Comenius University Bratislava
Robotic Summer School 2009
www.microstep-mis.com/~andy

Subsumption Architecture, Example

The next level can detect landmarks and when having received a goal from the user it can navigate to one of them by controlling the movement direction in the lower level.



Adapted from:Andrej Lúčny
Comenius University Bratislava
Robotic Summer School 2009
www.microstep-mis.com/~andy

Summary Subsumption Architecture

The subsumption architecture is:

- A hierarchical control architecture
- Organized in layers of competence
- Each layer is working autonomously (unless)
- Controlled by higher layers
- Inhibition and suppression control lower layers
- The layers consist of simple structured modules
- Rather robust against damage or module failure
- Implementation can be very complex

Control Architectures

Examples for control architectures:

- reactive control
- open-loop / closed loop control
- SMPA architectures
- Subsumption architecture
- Learning control (Neural Networks, Fuzzy, ...)
- Blackboard architectures
- ACT-R
- Hierarchical control
- ... ... and many more interesting approaches ...

Some important dates:

Next Monday, 13.7.2015
Q & A
so, please prepare questions!

Written examination, is scheduled for:
Thursday, **30. July 2015** from **10:00** to 11:40,
LBH Building, Lecture Hall: III.03a

Re-Sit examination, will be:
Tuesday, **8. Sept 2015** from **10:00** to 11:40,
LBH Building

Some important details for the exam

Written exam:

Thursday 30. July 2015, Room III.03, LBH, 10:00 – 11:40

It would be nice, if you arrive earlier

Last possibility to retreat from the exam without getting negative credits is one week before exam date; which is:

Wednesday 22nd, July.

Some important details for the exam

For the exam:

- Please take your students id card (Studentenausweis) with you.
- Documents with a photo, to check your an identity with you (identity card, or passport, or drivers license, …).
- Bring a pen, ball pen, felt-tip pen with you.
- You will not need a calculator for the exam.
- Paper will be provided.

Some important details for the exam

Thursday 30. July 2014, Room III.03, LBH, 10:00 – 11:40

- only pen, ball pen, felt-tip pen are allowed, NO pencil.
- only blue or black, NO red or green colors
- all answers need an explanation.
- please indicate clearly what you consider to be the solution.
- when you use formulas, all variables must be explained explicitly.
- short sentences and keywords are preferable to long text passages.
- no extra tools or utilities or electrical devices are allowed.

Questions that could be part of the exam

Compare Wolframs class III and class IV behavior.
Explain differences and similarities between these two classes of behavior.

Name and explain in detail the two ways how the higher levels in the Subsumption Architecture can influence the lower levels.

Describe the overall structure of the so called Glidergun (Gospers Glidergun) in Conway's Game of Life.

Explain how a hypercube and a binary genome of an Evolutionary Algorithm are related to each other.
Draw a sketch, visualizing this for a binary genome that has more than two bits.

Artificial Life
Summer 2015

Subsumption Architecture

Thank you for your attention

Dr. Nils Goerke, Autonomous Intelligent Systems,
Department of Computer Science, University of Bonn