Multi-agent systems

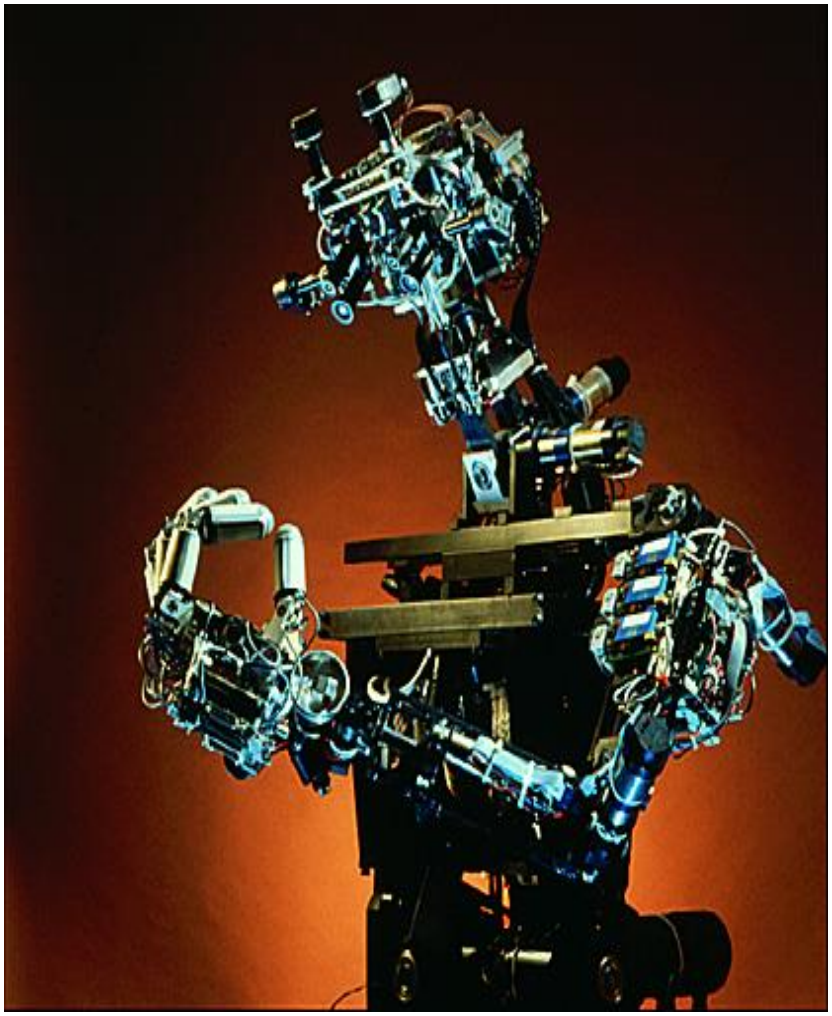
Andrej Lúčny

KAI FMFI UK

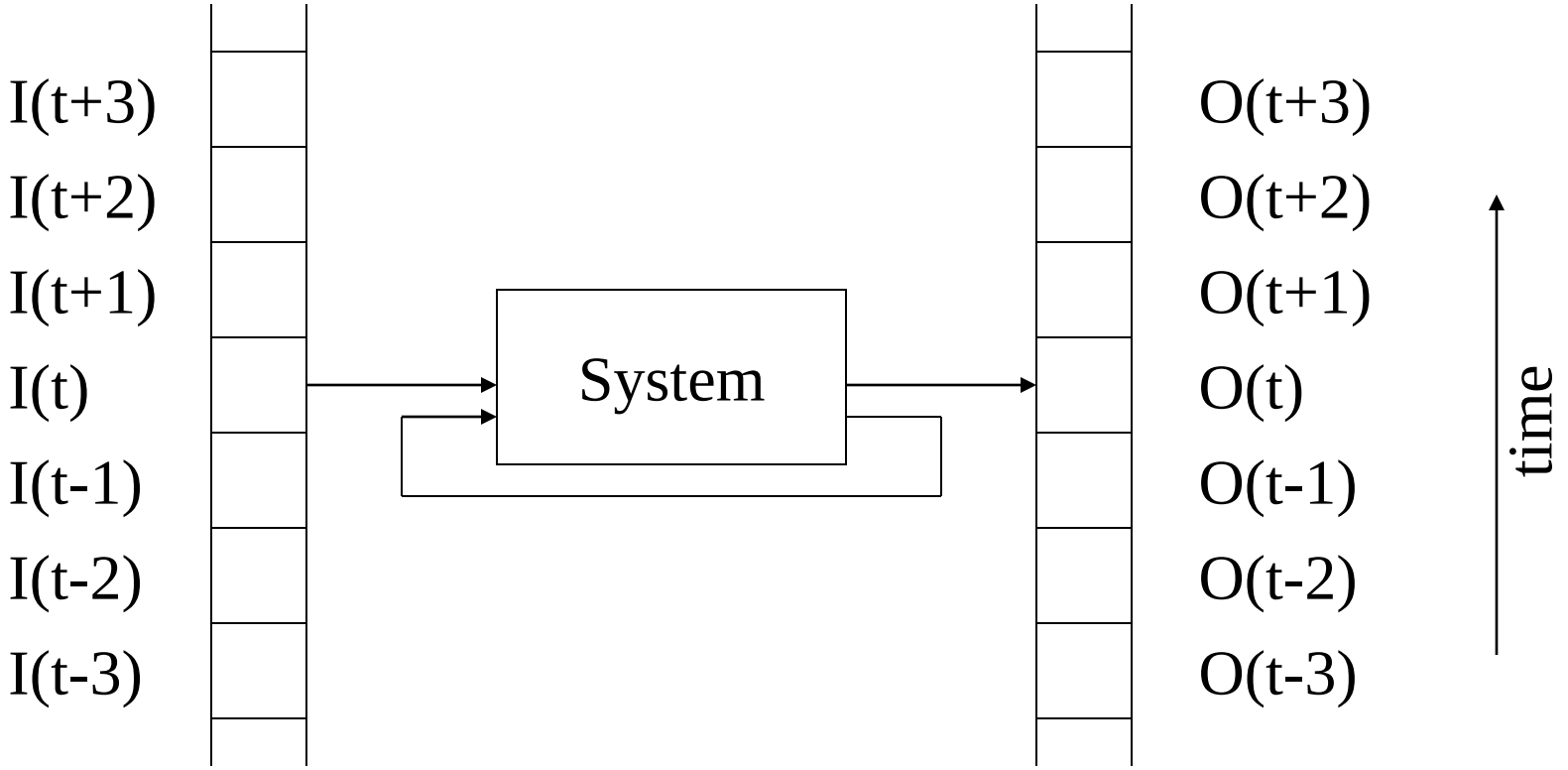
lucny@fmph.uniba.sk

http://www.agentspace.org/mas

Multi-agent approach to control



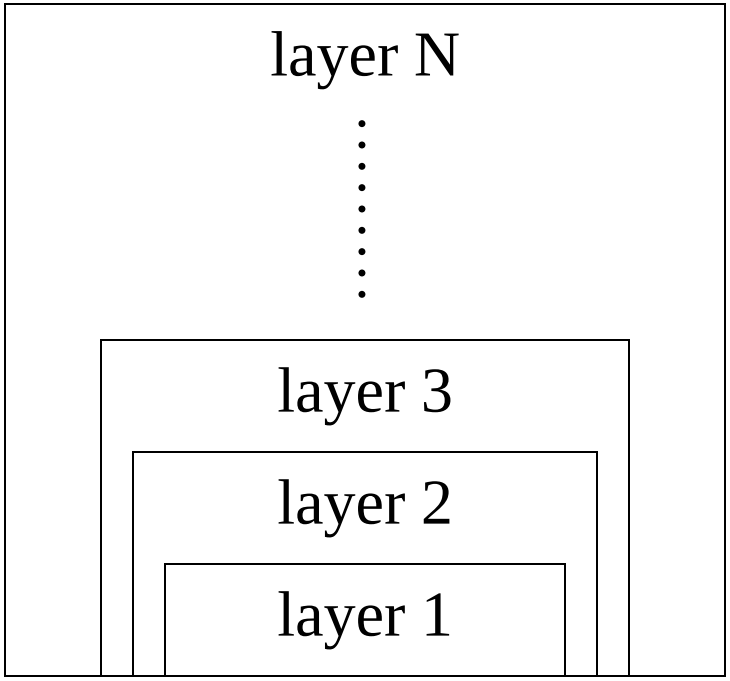
Control


I(t+3)
I(t+2)
I(t+1)
I(t)
I(t-1)
I(t-2)
I(t-3)
System
O(t+3)
O(t+2)
O(t+1)
O(t)
O(t-1)
O(t-2)
O(t-3)
time

Subsumption architecture

- architecture of autonomous system, typically mobile robot
- it mimics simplified biological evolution in such a way that newer versions of control subsume whole older versions
- individual steps of evolution are performed by incremental job of developer
- it comes from principles of Cambrian or “new” artificial intelligence

Biological motivation



Cambrian intelligence ("Embodied intelligence")

Fundamental postulates:

- Situatedness
- Embodiment
- Emergence
- Interaction
- Hierarchy (Incrementality)

Situatedness

- Control is designed for particular set of situations in which can appear
- Typically simpler situations are subsets of more complicated. Thus we can prepare solution for the simpler ones and then refine it to the more complicated ones.
- Specific situations are dedicated to be solved by specific sets of modules – once one trick works, another one in the other time.
- We do not aim to build universal representation of all the situations. „the world itself is its own best model“.

Embodiment

- Control is designed for concreate body
- We do not split design from implementation
- We do design with prototype in hand
- Thus we can test and volume each phase of development
- We work with real input and we can not only handle but even employ their real character

Emergence

- System behavior is a result of mutual interaction among its modules
- Relation between behavior of modules and behavior of system is not doomed to be clear
- When we implement some system faculties by proper specification of modules behavior, it can happen that we have implemented a faculty which has not been concerned by us. Even it can be difficult to predict that such behavior appears. It is much easier to explain why it has appeared a posteriori

Interaction

- System can raise its intelligence from dynamics of its environment. It is often possible to replace a complicated control structure by a simple one which properly employs environment dynamics
- Unlike traditional systems which contains cognition as a module, here system exhibit more intelligent environment in dynamical environment than in static one
- Intelligence is a global feature of system and we are not able to find an inner module which would correspond to it.
- System reactivity support it robustness to outer influences and transient states in its environment

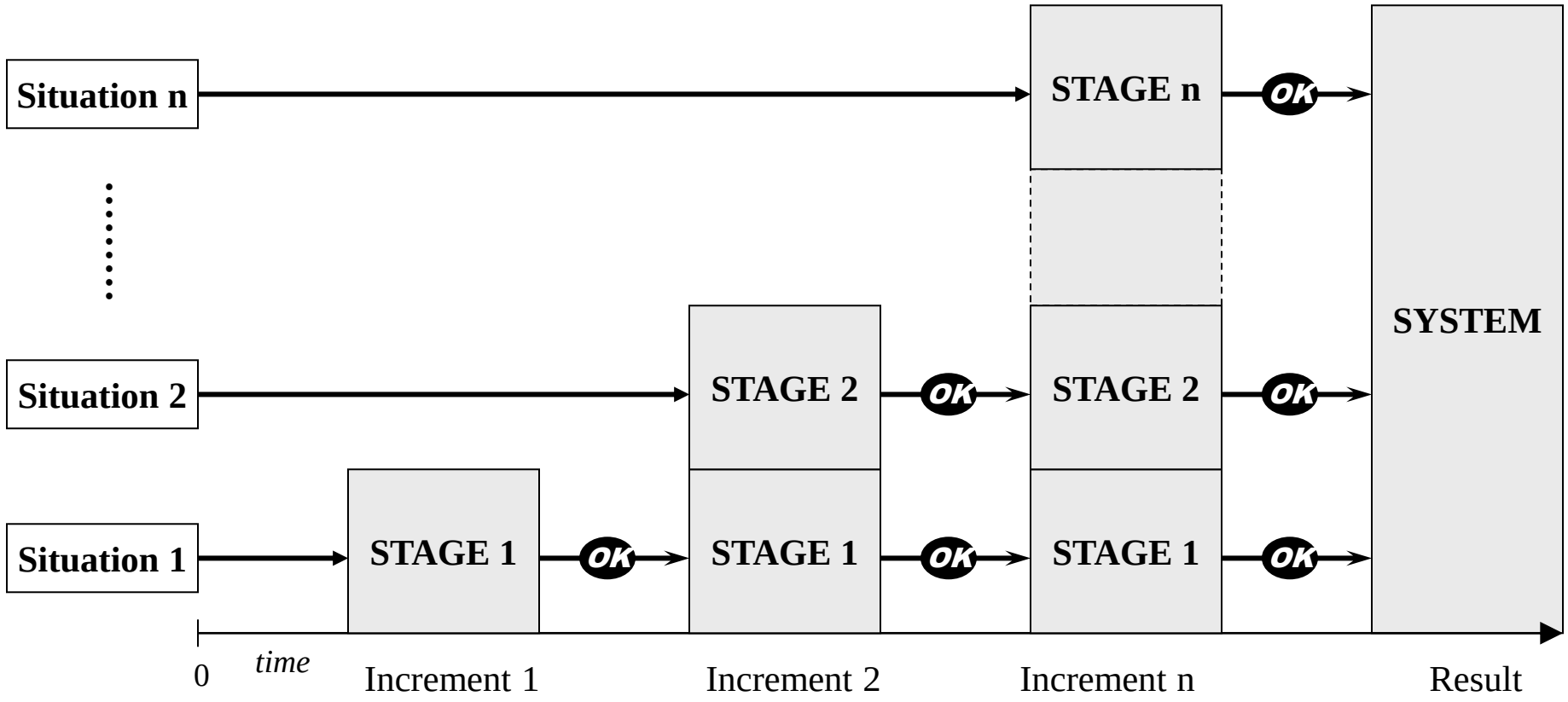
Hierarchy (Incrementality)

- We develop system in an incremental way, layer by layer. Each layer implements a particular activity or in other words it solves a particular situation
- We use bottom-up development. We start with hierarchically lower (resp. evolutionary older) layers, i.e. with more primitive activities or more simple situations.
- Gradually we add novel layers. After adding each layer we test and volume until the corresponding situation is satisfactory solved

Hierarchy (Incrementality)

- Thanks to incremental development we cannot fail in integration of individual components into system. On the other hand there is a danger that we fail to improve system to upper level and we will not be able to add the next activity keeping the already implemented activities in operation
- Hierarchy resides in ability of the newer layers to employ the older ones. The newer layers can monitor the older ones or even they can influence their operation

Incrementally, bottom-up



Subsumption

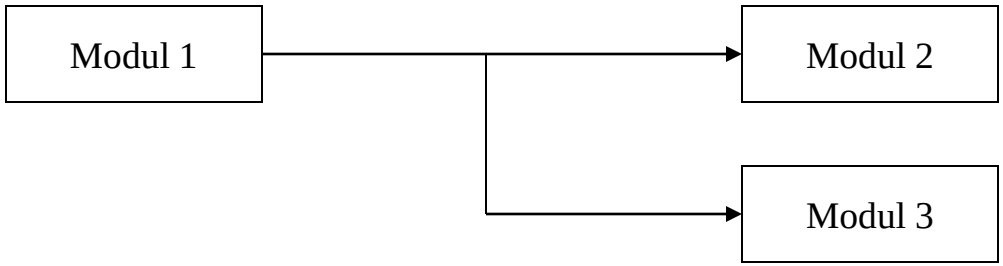
- How a newer layer can influence the older one ?
- Problem: developing the older layer we have not considered any interface which could be called later from layers implemented in future…

Subsumption

- Solution: system must be modular and the modularity should implicitly allow such influences (and should have interface for that)

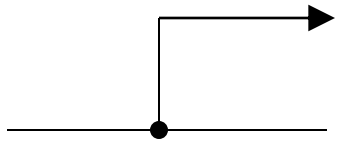
Subsumption architecture

- System is compounded from autonomous modules (Augmented Finite State Automata)
- Those modules are interconnected by communication lines which can transfer messages

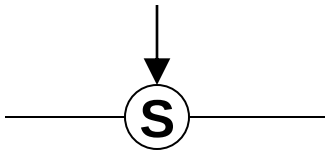


Subsumption architecture

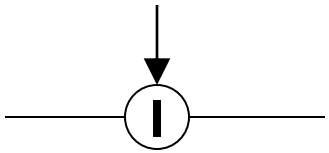
- The newer layer can manipulate the older one via
  - monitoring



  - suppression

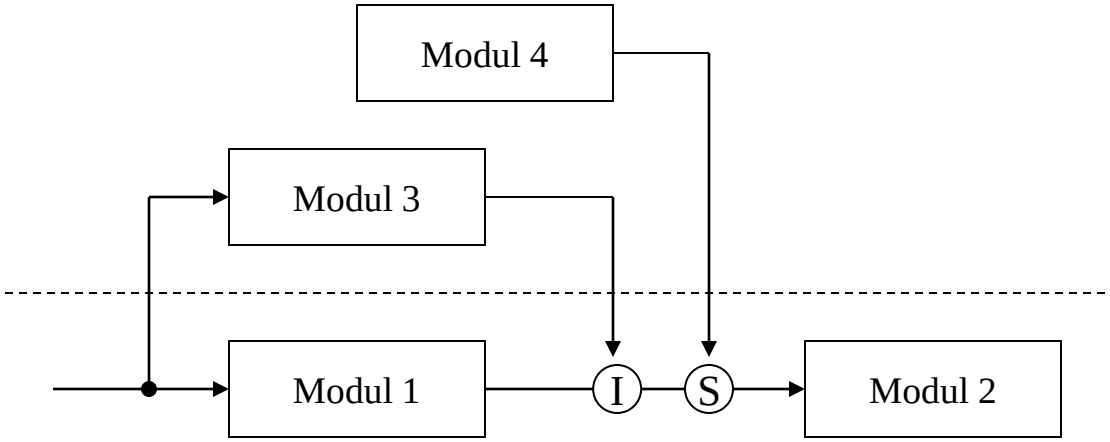


  - inhibition



Subsumption architecture

- When we add a new layer, we can employ these mechanisms:

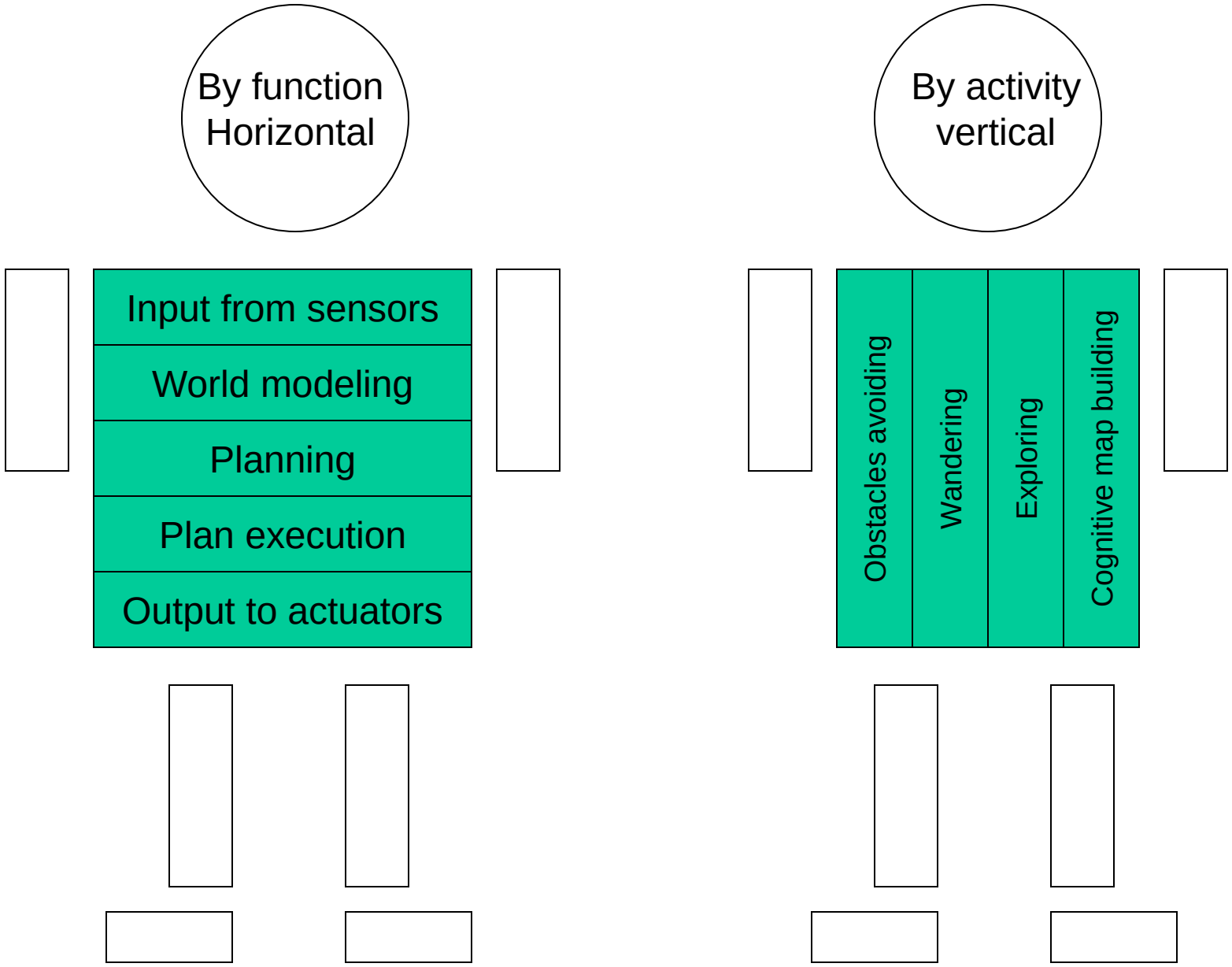


Decomposition

- by function = layers contain codes providing similar function (e.g. vision)

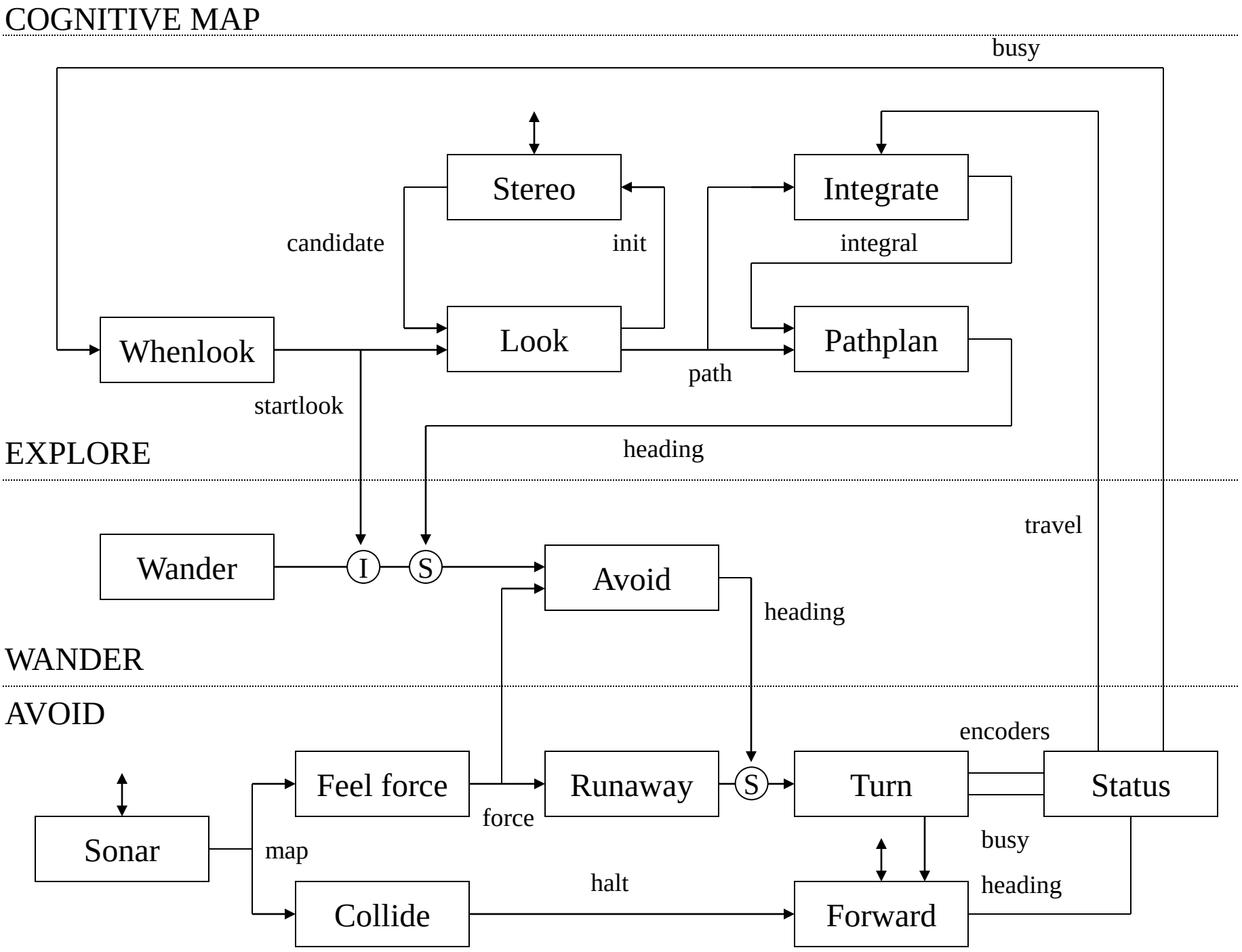
- by activity = layers contain codes providing similar activities (e.g. obstacles avoiding)

Subsumption architecture is based on decomposition by activity


By function
Horizontal
Input from sensors
World modeling
Planning
Plan execution
Output to actuators
By activity
vertical
Obstacles avoiding
Wandering
Exploring
Cognitive map building

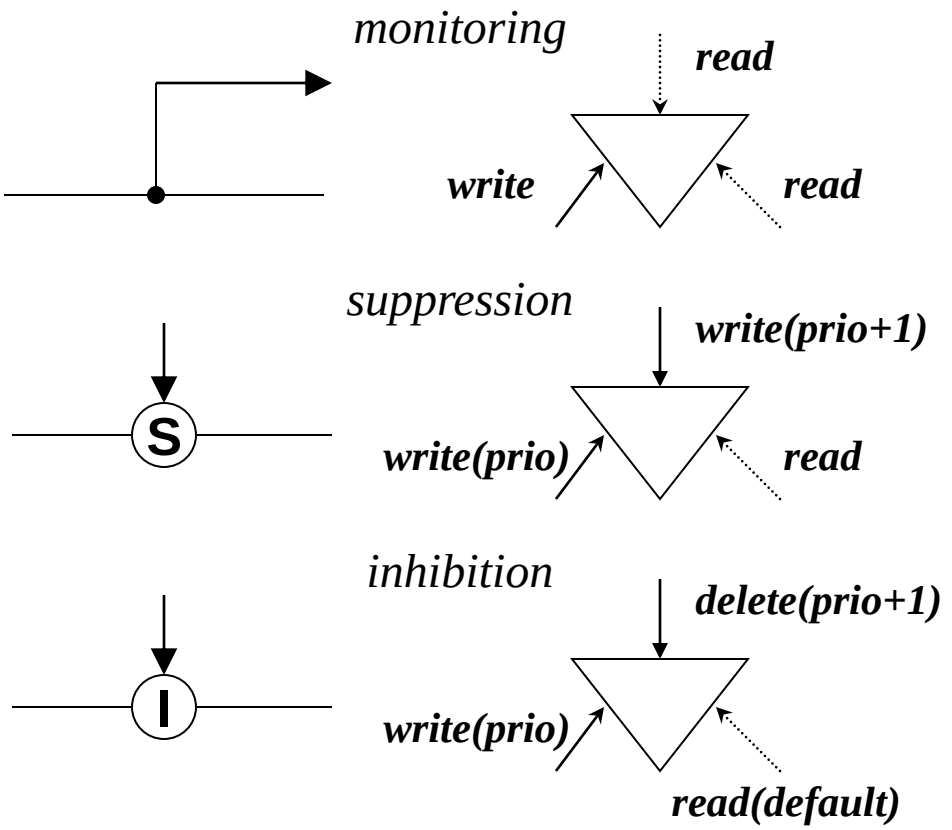
Example:

- *For example, navigation of two wheeled robot in bureau can be developed by subsumption in the following way: We start with robot which just goes forward. 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, but 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. However we perform such a turn only when no obstacles are detected and we implement it just by apparent detection of obstacles. Further we add another layer which provides an active search for suitable absolute directions for movement 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. 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 older level*


COGNITIVE MAP
busy
Stereo
Integrate
candidate
init
integral
Whenlook
Look
Pathplan
path
startlook
heading
EXPLORE
travel
Wander
I
S
Avoid
heading
WANDER
AVOID
encoders
Feel force
Runaway
S
Turn
Status
Sonar
force
map
busy
halt
heading
Collide
Forward

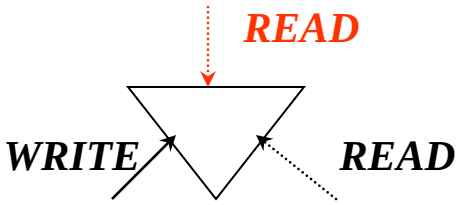
Implementation by Agent - Space

- Control designed by Subsumption architecture can be directly expressed as MAS (Agent – Space architecture)



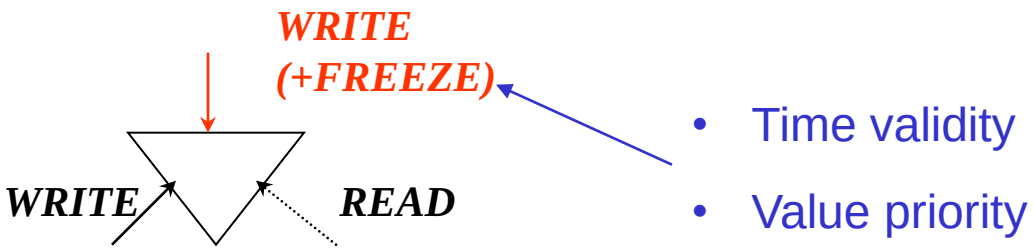
Monitoring

- The newer layer reads data which implements data flow in the older layer



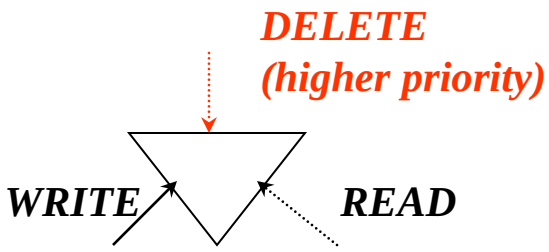
Suppression

- The newer layer overwrites data which implements data flow in the older layer



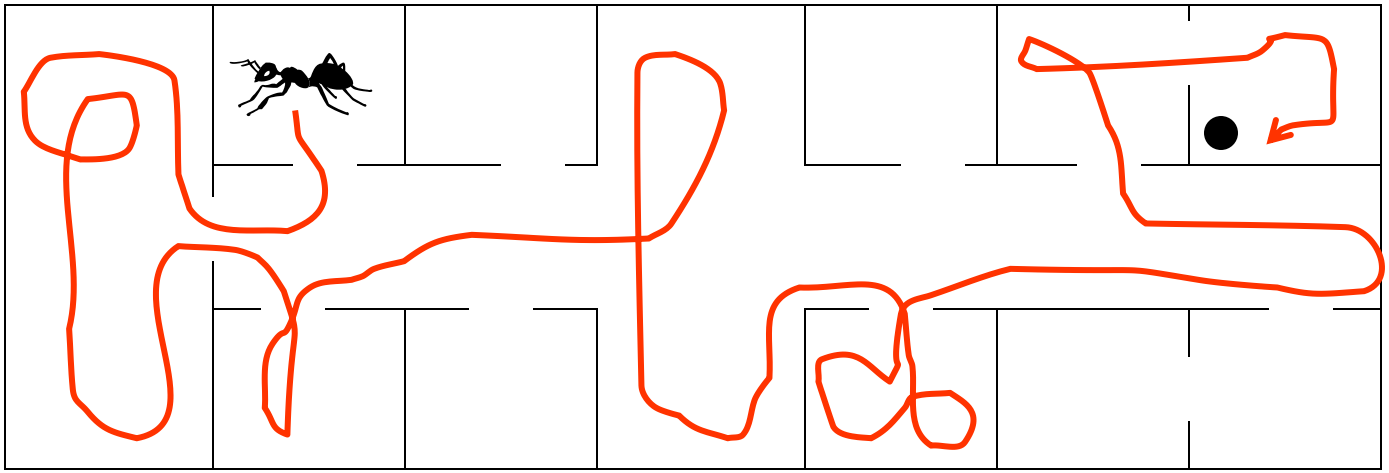
Inhibition

- The newer layer deletes data which implements data flow in the older layer

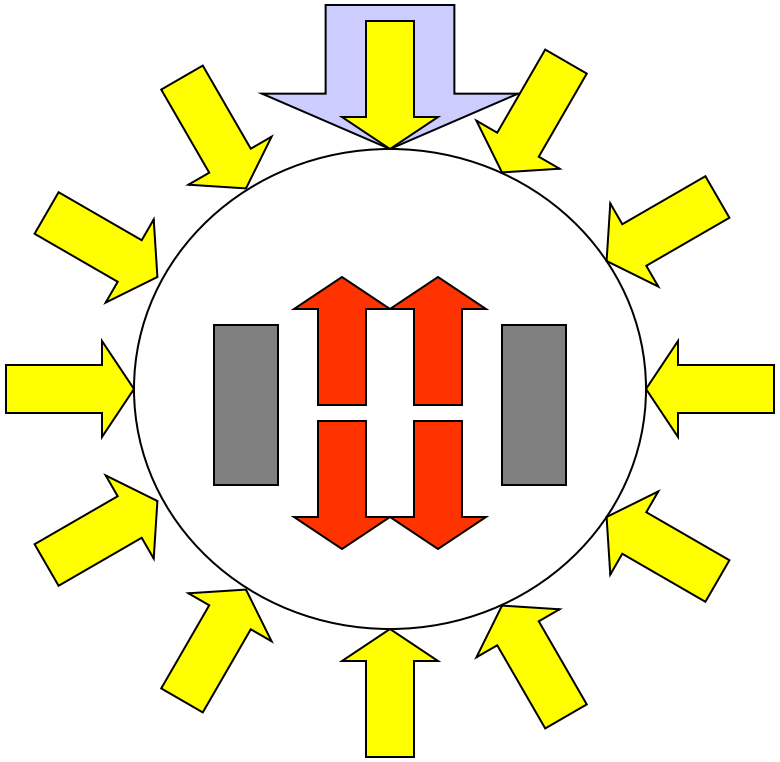


Control system of mobile robot

Task: find a ball in bureau environment



Sensors and actuators



Sonars measure distance to the closest obstacle

Wheels for movement forward, backward and rotation at the spot

Ball detector

System design

via subsumption architecture

STOP – Stop at the ball

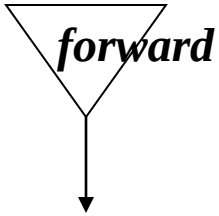
EXPLORE – Explore rooms

WANDER - Wander

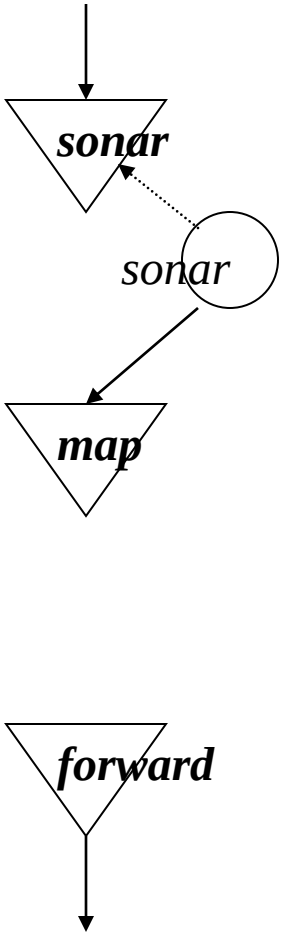
AVOID – Avoid obstacles

AVOID

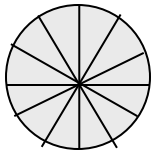
Usually *forward* tells: go ahead



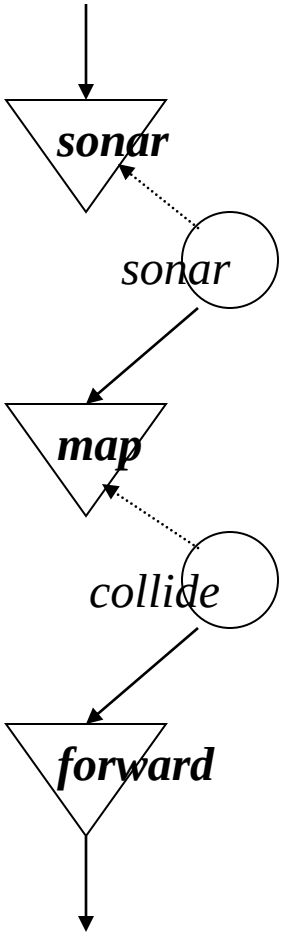
AVOID



Sonar measures field of distances to the closest obstacle in various directions



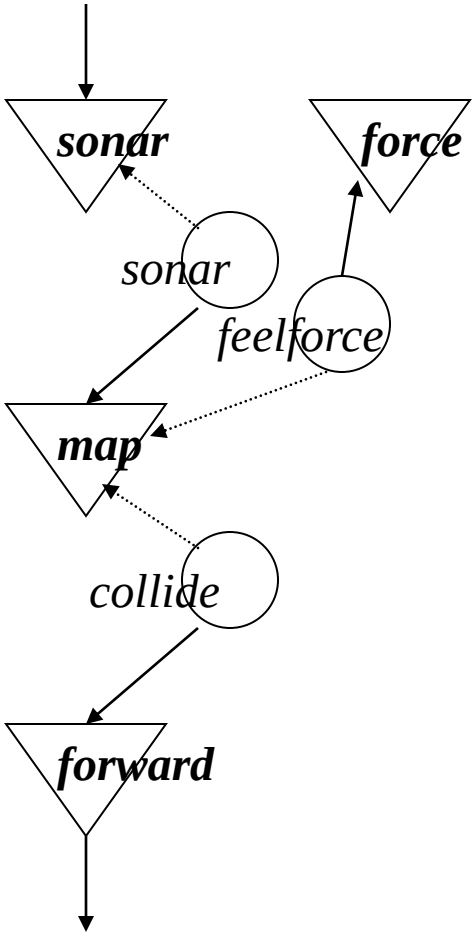
AVOID



When there is danger of collision, *collide* stops *forward*.

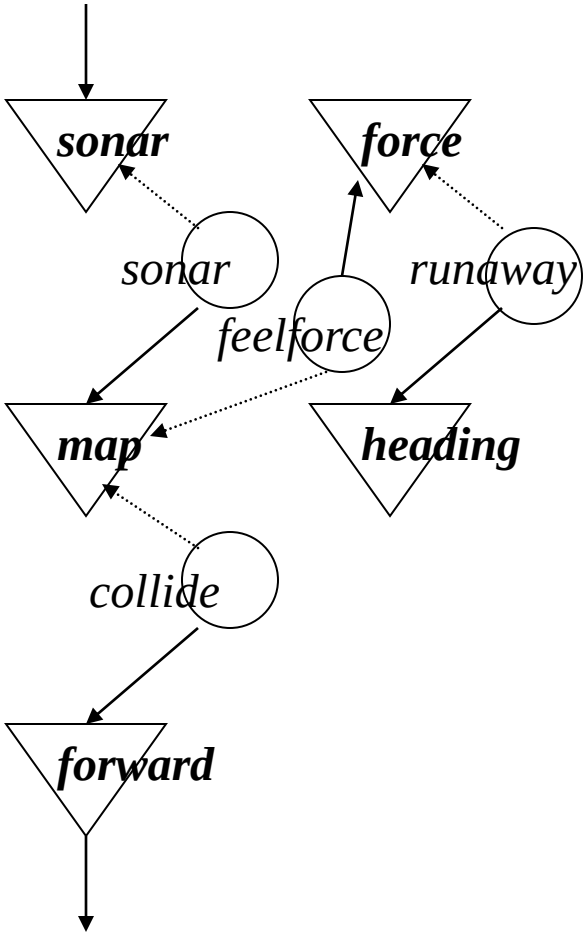
Thus the robot goes straight forward until it meets obstacle and stops.

AVOID



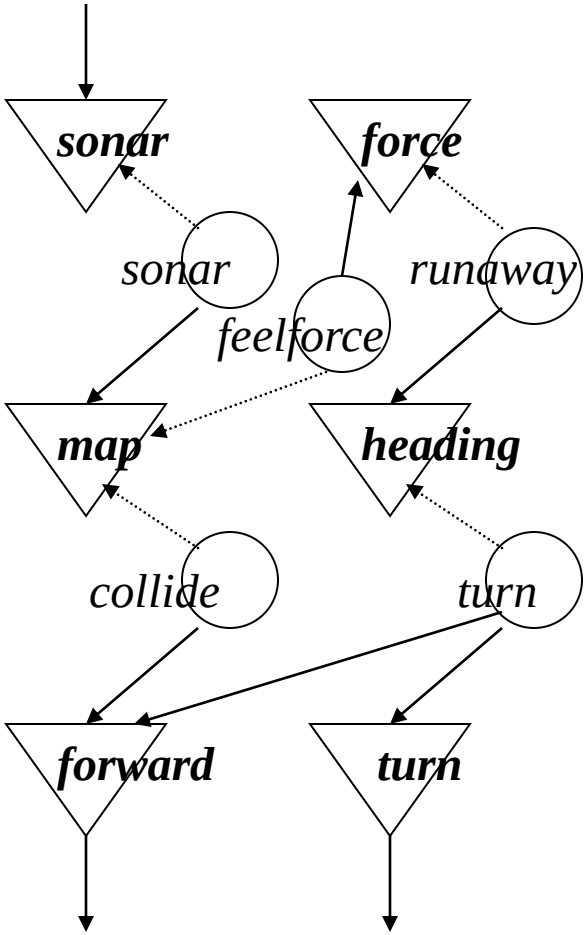
Feelforce evaluates direction in which there is the most probable danger of collision

AVOID



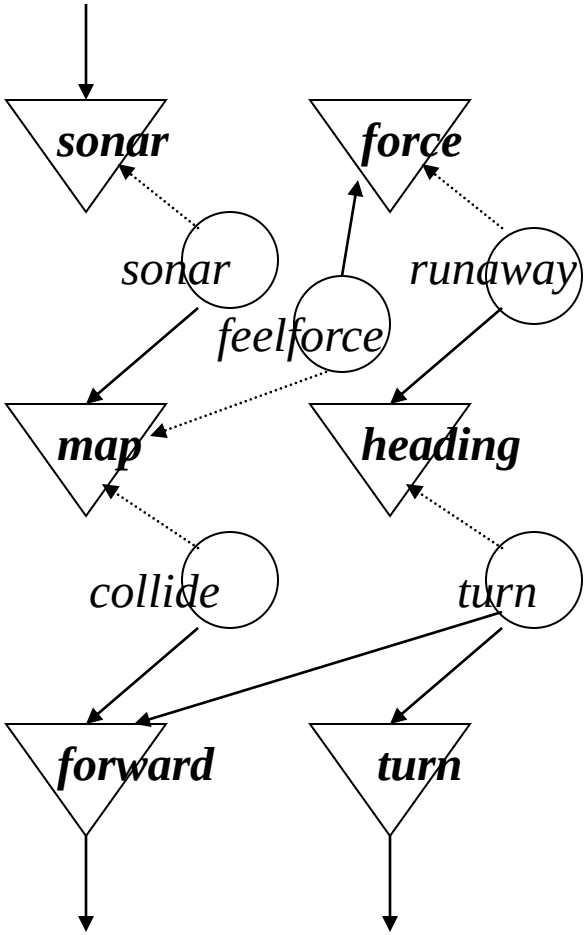
Runaway orders to move in the opposite direction

AVOID



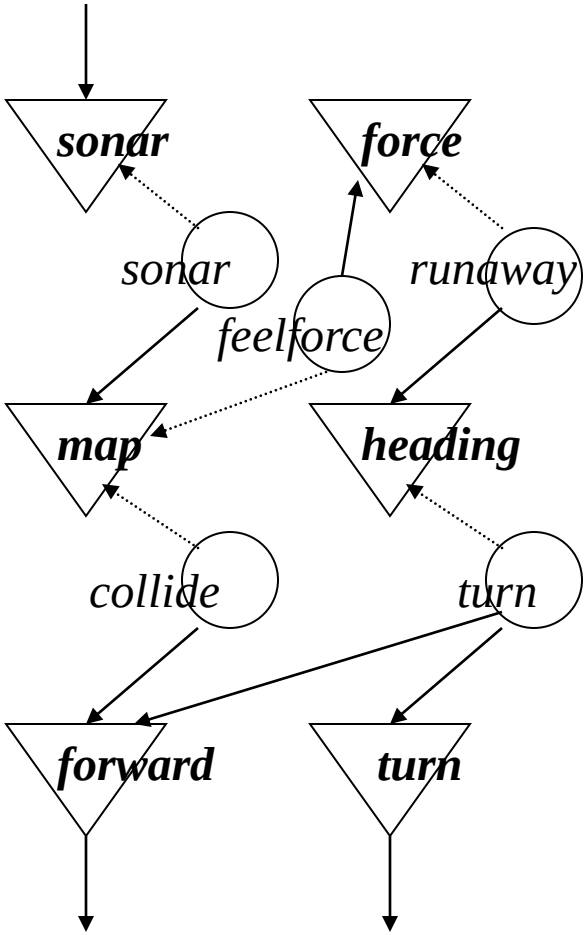
Finally *turn* transform the direction to wheel rotation

AVOID



Let us concern that rotation of robot provides feed back to sonar measurement what has impact to heading of robot. More the robot rotates from colliding direction, less rotation is ordered in *heading*.

AVOID

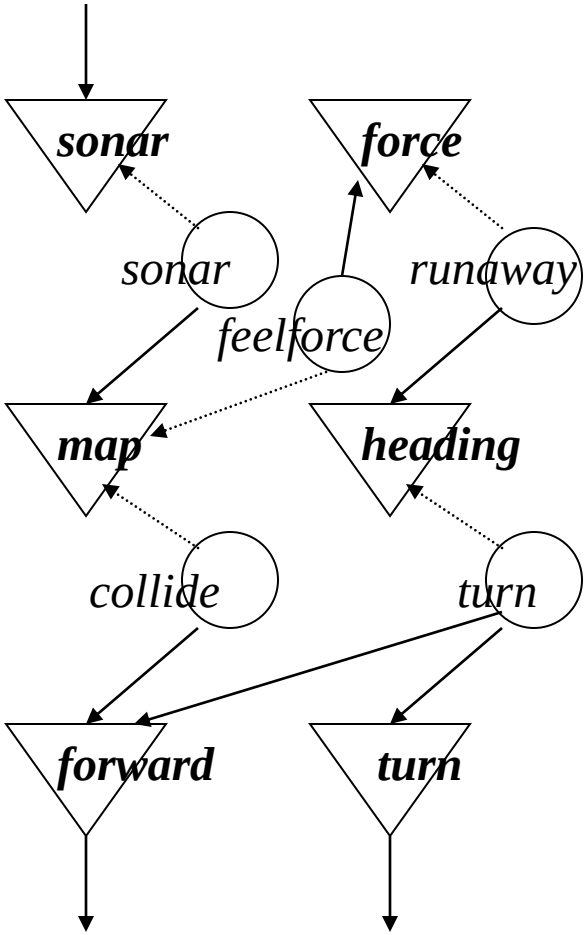


Thus the robot goes ahead until it meets obstacle. The it starts to rotate until no collision is threatened. Than it goes straight again.

In this way the robot easily gets into a cyclic trajectory, but it acceptable for this stage of development.

AVOID is finished.

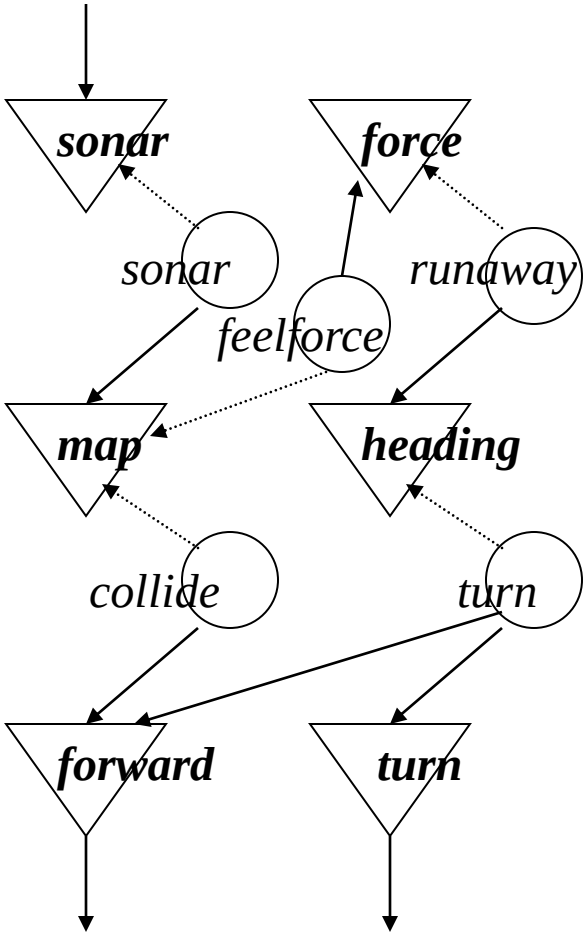
AVOID



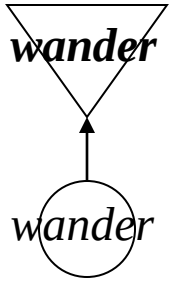
We can observe now that by implementation of static obstacles avoiding we have implemented also avoiding to slowly moving objects colliding with the robot.

It is a very trivial case of emergence

AVOID



WAN DER



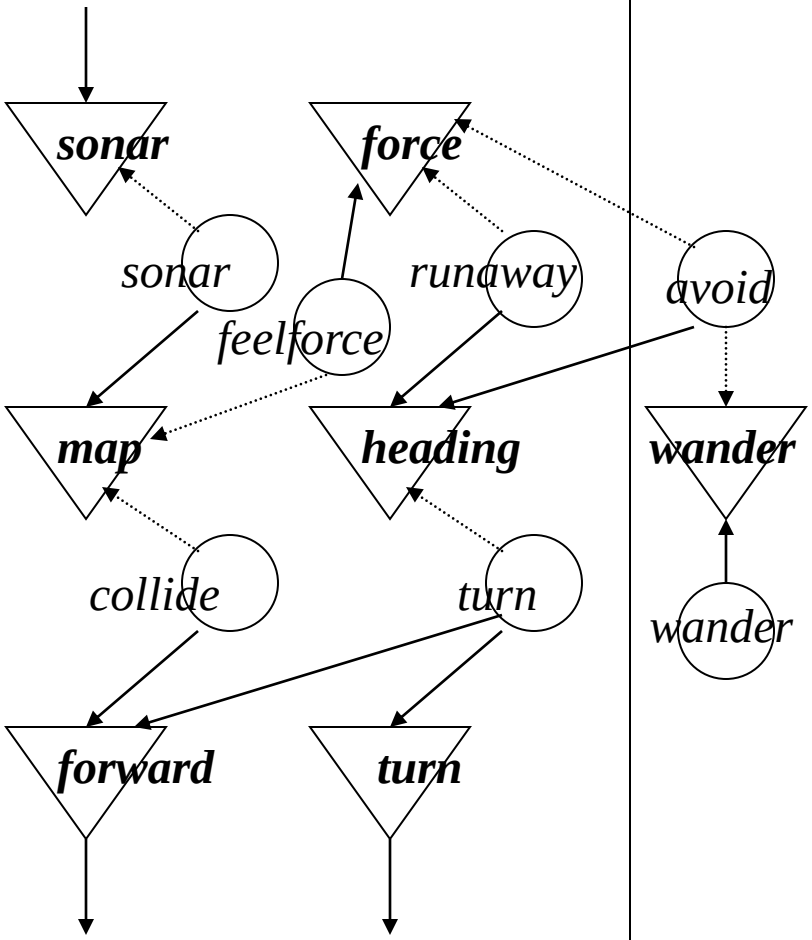
Now we would like to solve the cyclic trajectory of robot

We simply sometimes turn robot by random rotation angle.

So from time to time *wander* generates such proposal

AVOID

WAN
DER

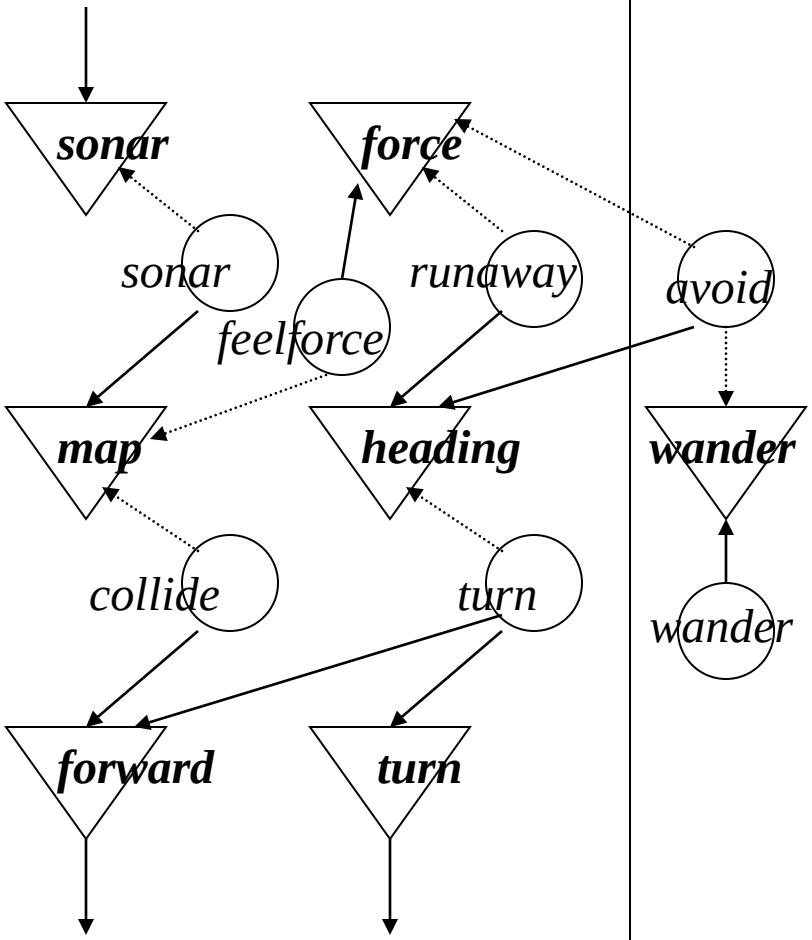


And agent *avoid* checks if there is no acute obstacle avoiding and if no, it writes the proposed angle into *heading* as if the angle would the right angle for obstacle avoiding.

Thus mechanism of obstacle avoiding serves also for random rotation of robot

AVOID

WAN
DER



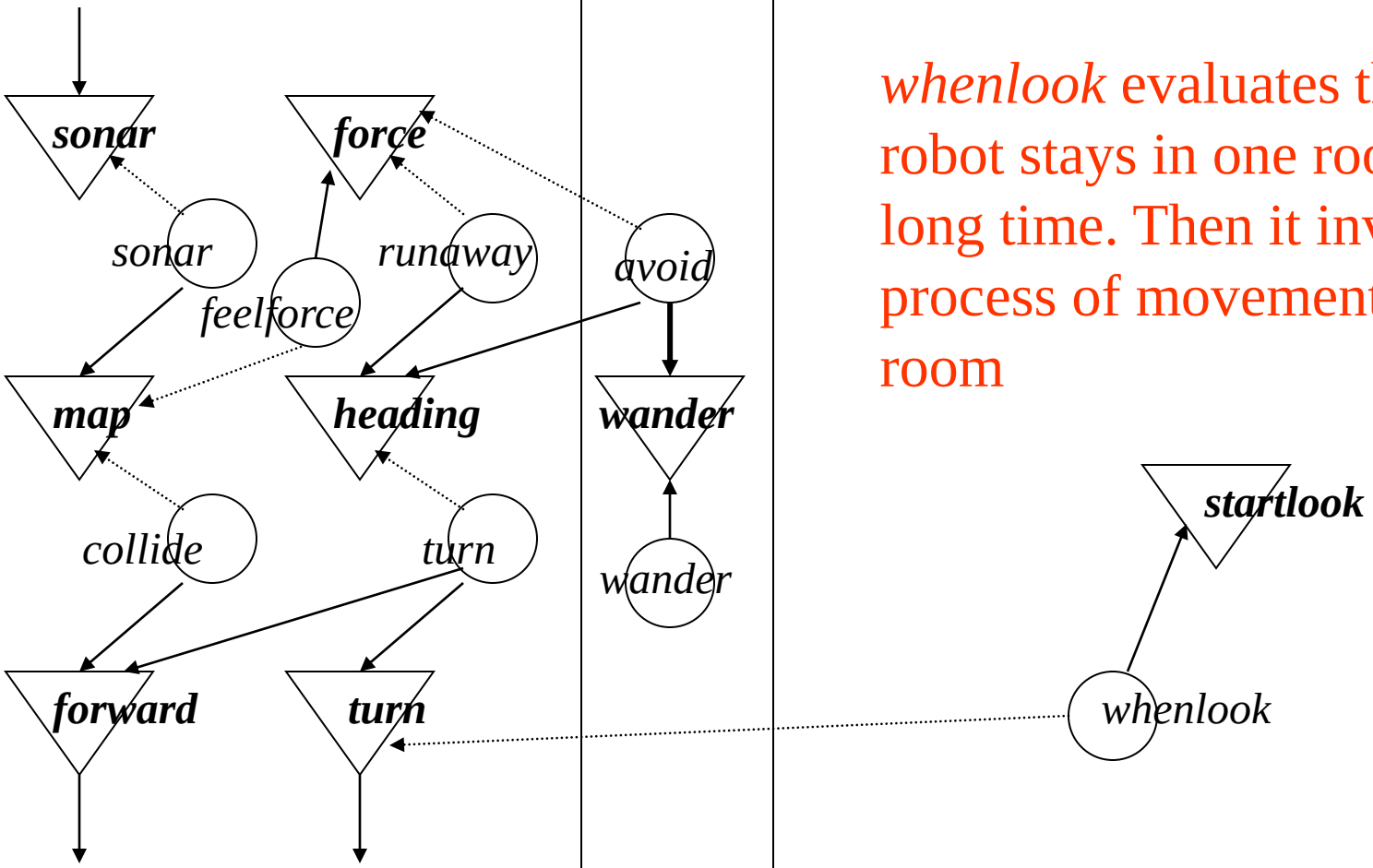
In this way robot not only avoid obstacles but also it trajectory is not predictable and it wander through room.

But so far it keeps in one room, rarely going out through door.

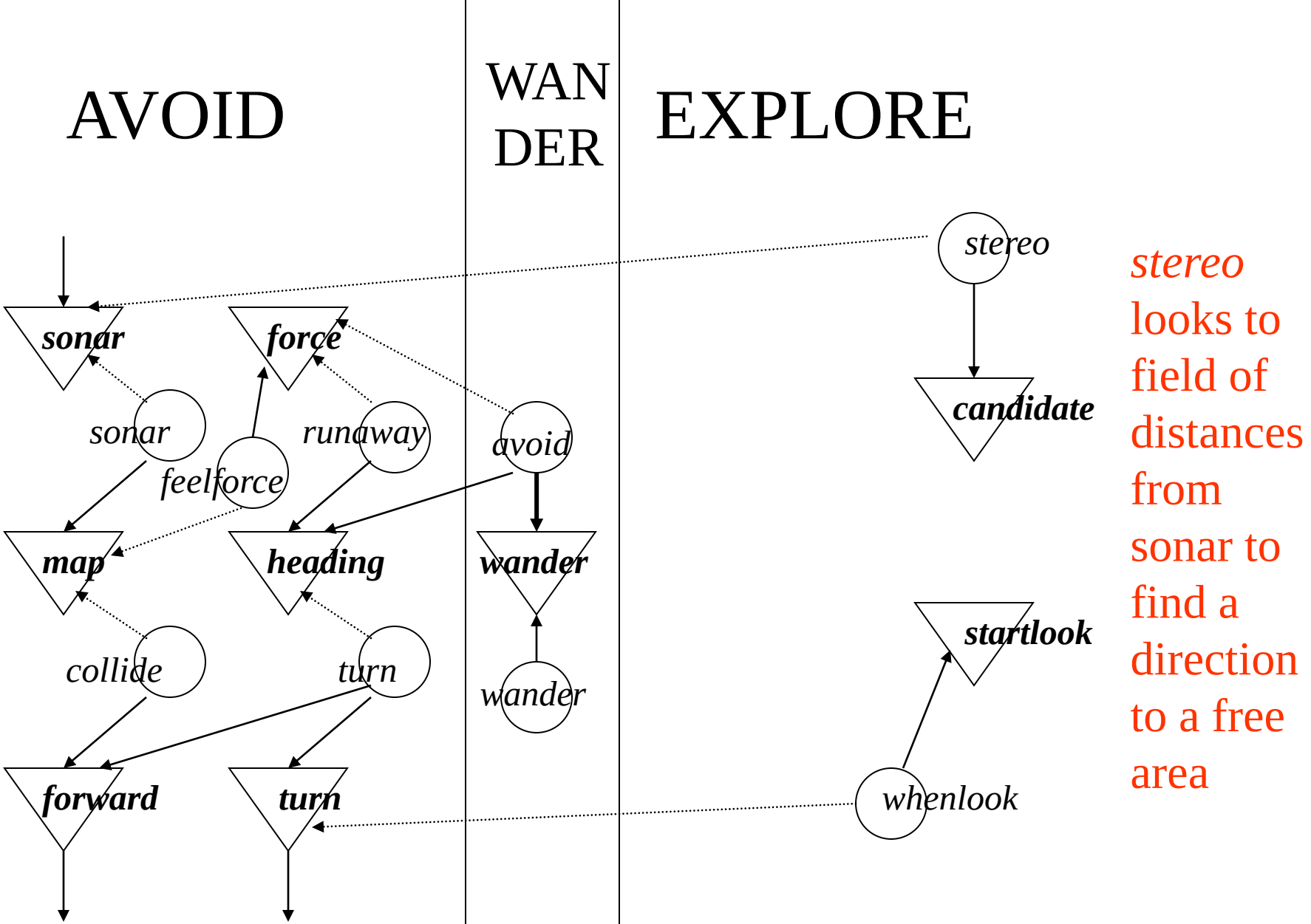
AVOID

WAN DER

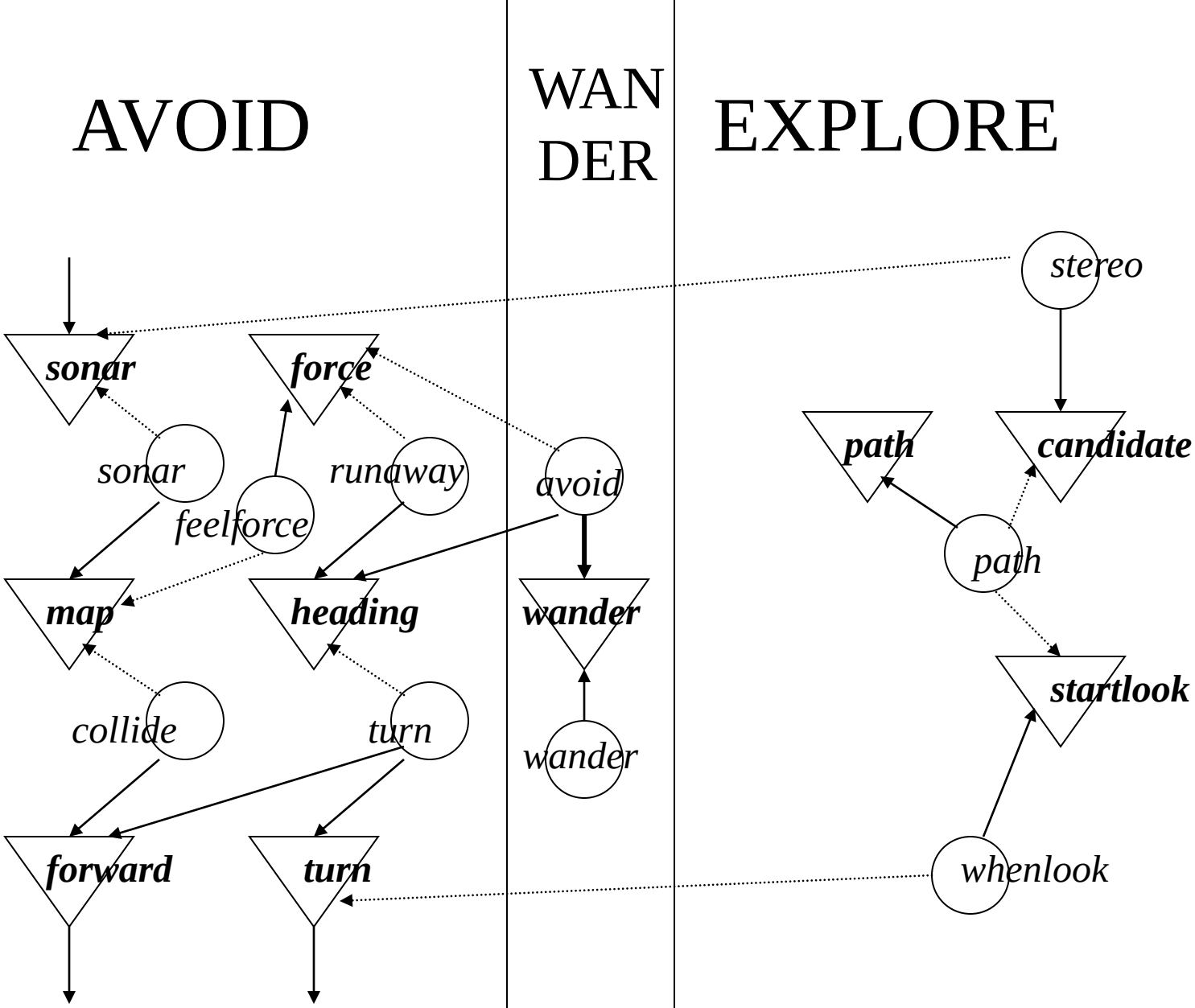
EXPLORE



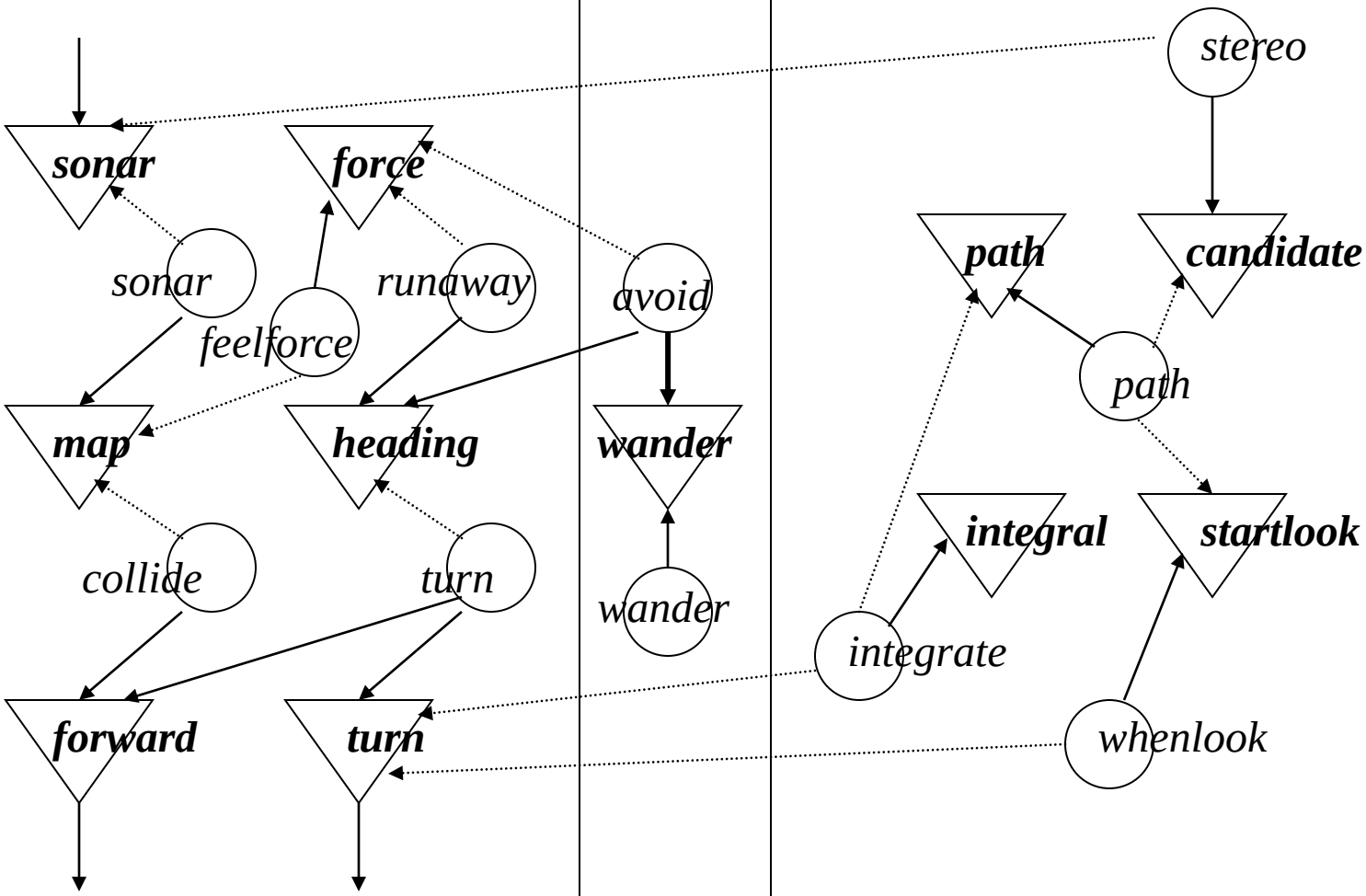
whenlook evaluates that the robot stays in one room for a long time. Then it invokes process of movement to other room



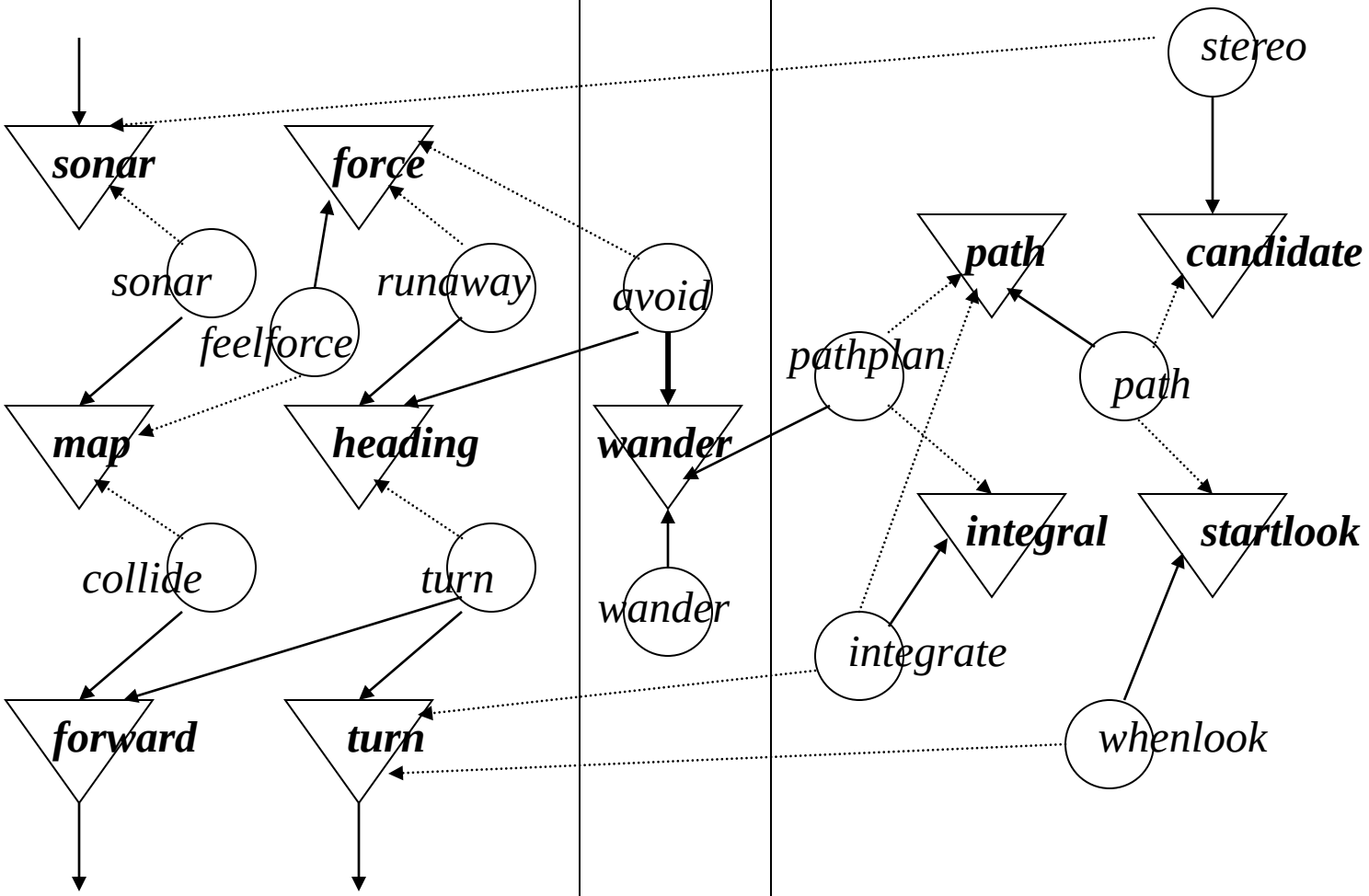
stereo looks to field of distances from sonar to find a direction to a free area



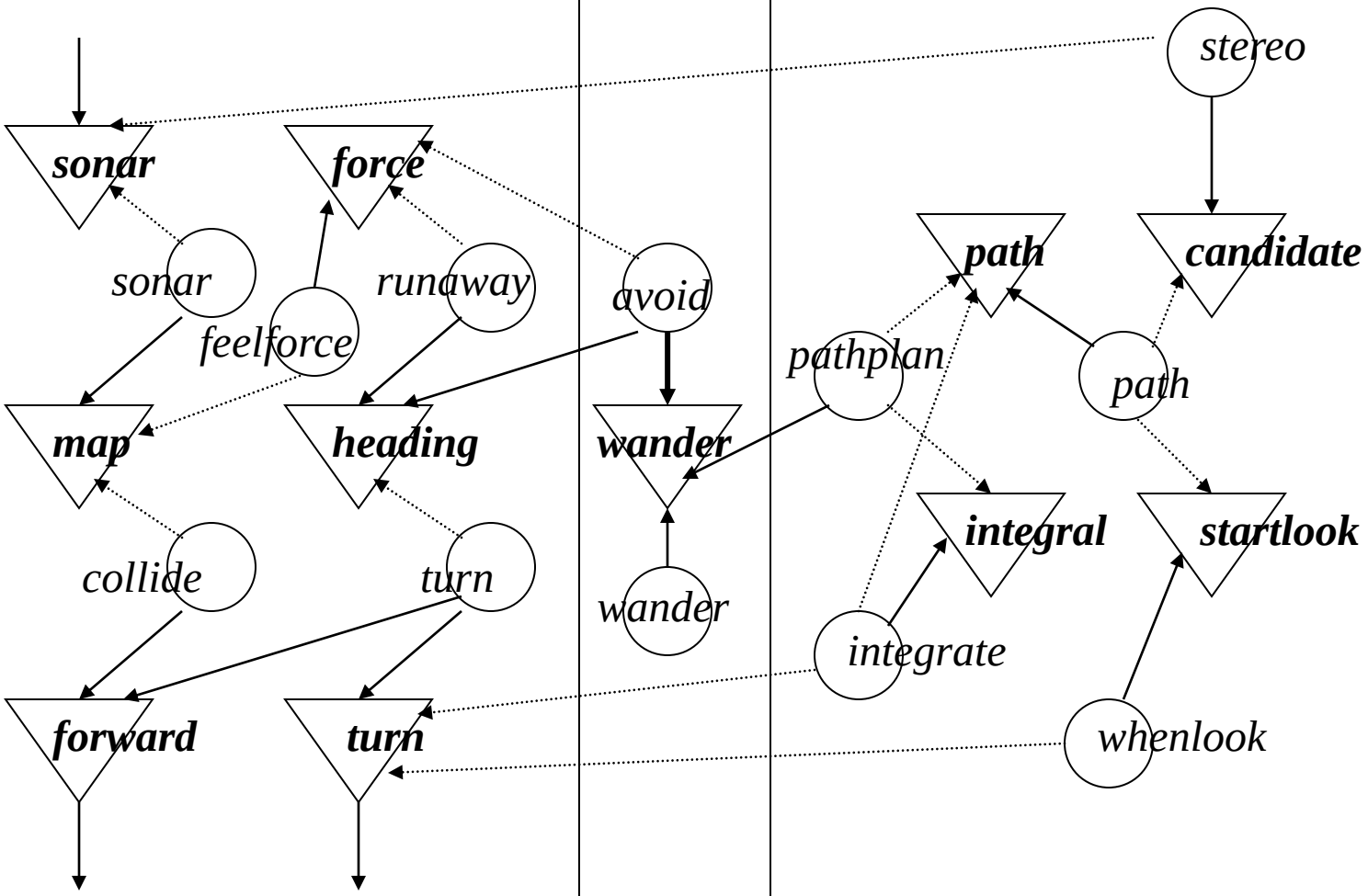
Path remembers the direction to the area



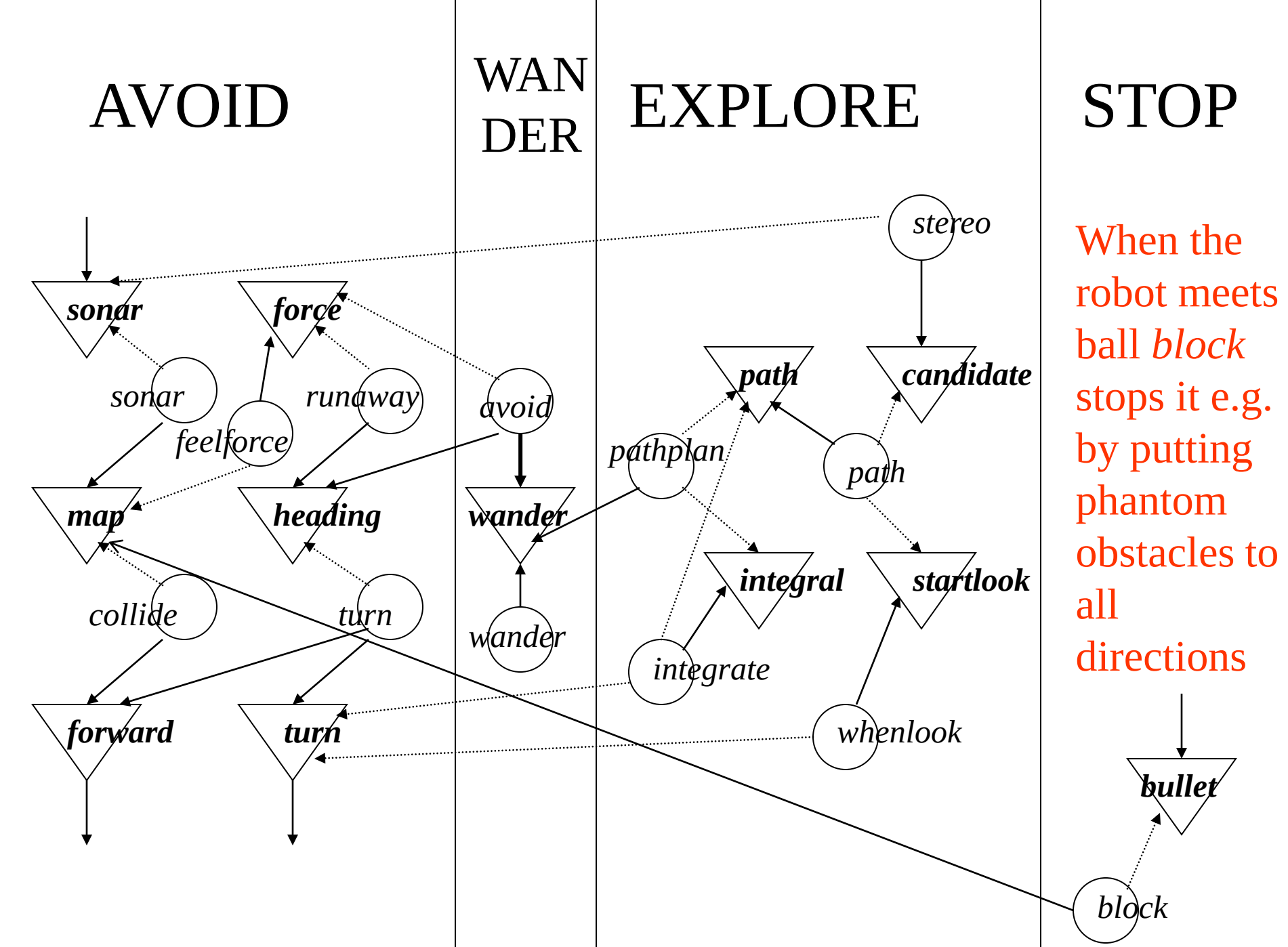
However it is direction relative to robot this *integral* helps to calculate its absolute value which will not change when robot rotates



Pathplan aim to turn robot to the absolute rotation for going to the free area



This control is rather enough for probable visit of each room


AVOID
WAN DER
EXPLORE
STOP
stereo
sonar
force
path
candidate
sonar
runaway
avoid
feelforce
pathplan
path
map
heading
wander
integral
startlook
collide
turn
wander
integrate
forward
turn
whenlook
bullet
block
When the robot meets ball block stops it e.g. by putting phantom obstacles to all directions