

# Mathematical analysis of behavior systems.

Luc Steels

Artificial Intelligence Laboratory  
Vrije Universiteit Brussel  
Pleinlaan 2, B-1050 Brussels, Belgium  
E-mail: `steels@arti.vub.ac.be`

## Abstract

*A behavior system consists of the components and control programs necessary to establish a particular behavior in a robotic agent. The paper proposes a mathematical approach for the analysis of behavior systems. The approach rests on viewing a behavior system as a dynamical system whose equilibrium state is attained when the behavior it is responsible for is achieved.*

## Keywords

BEHAVIOR SYSTEMS – MATHEMATICAL ANALYSIS

## 1 Introduction

An autonomous agent is a physical system that has its own resources to operate independently in a dynamically changing real world environment. The classical AI approach towards autonomous agents is based on a logical foundation, as exemplified by Shakey [Nilsson 1984]. More recently, an alternative approach, known as the bottom-up or behavior-oriented approach, has been explored by several researchers [Steels and Brooks 1994]. This paper makes a small contribution towards a more formal foundation for behavior-oriented AI. It defines the notion of a behavior system and proposes a mathematical technique known as equilibrium analysis for specifying and investigating behavior systems. We believe that the technique is sufficiently simple and straightforward to be used on a routine base in the design and analysis of behaviors.

The first part of the paper defines the notion of behavior systems. The second part introduces a mathematical framework for studying the behavior of autonomous agents. The third part introduces the technique of equilibrium analysis and goes through some examples. A discussion of some limitations and further potential extensions concludes the paper.

## 2 Defining behavior systems

A knowledge-oriented approach starts from the viewpoint that it is possible (and necessary) to identify individual actions of the agent. These actions are the primitive operators in the search space built up by a planner [Nilsson 1984]. A behavior-oriented approach starts from behaviors as the fundamental unit of analysis [Brooks 1991]:

*Definition:* A behavior is a regularity in the interaction dynamics between the agent and the environment [Smithers 1992].

For example, we may observe that an agent maintains a certain distance to the wall. As long as this regularity holds, we, as observers, may say that there is an obstacle avoidance behavior.

To realise a behavior, there must be some sort of mechanism available in the agent. For example, phototaxis could be based on a direct coupling between differences in intensity of light between a left and right photosensor and differences in left and right motor speed. The mechanism must be physically implemented using a set of components (sensors, body parts, actuators) and a control program relating changes in sensory values and internal state variables to changes in actuator parameters and internal state variables. The observed behavior is due to the interaction between the operation of the mechanism and a particular environment in which the agent finds itself.

*Definition:* A behavior system is a collection of components responsible for realising a particular behavior [Steels 1993]<sup>1</sup>.

The realisation of a behavior may be indirect. For example, sometimes the interaction of two existing behavior systems in a particular environment may give rise to a new behavior, i.e. a new regularity, which can only be described using other descriptive categories than those used to identify the regularities of the component behavior systems. In this case, we talk about an emergent behavior [Steels 1990].

Several languages have been proposed recently for defining the control programs of behavior systems. One example is the Behavior Language [Brooks 1990] which is based on augmented finite state machines. Another example is REX based on circuit diagrams [Kaelbling 1992]. Also several architectures have been proposed for regulating the way that different behavior systems work together. For example, in the subsumption architecture [Brooks 1991] one behavior system may override the impact of another behavior system by inhibiting the inflow of sensory information or by inhibiting the outflow of actuator commands.

In our own work, we follow a dynamics approach, which is based on the following assumptions [Steels 1993]:

[1] Control programs are viewed as dynamical systems which establish a continuous relation between the time varying data coming out of a sensor and a stream of values going to the actuators. Implementation on digital processors is achieved by discretising time, similar to cellular automata.

[2] All behavior systems are active all the time. There is no subsumption relation or no action selection mechanism that gives one behavior system precedence over another one. Instead, each behavior system determines an influence on the actuator parameters (or on changes in the internal variables), and the influences are summed.

We have been using a language PDL [Steels 1992] to make the writing of control programs based on these assumptions straightforward. A C-based variant of PDL is operational on all of the robot platforms used in our laboratory and will be used in the rest of the paper. Here is an example of the control program for a forward movement behavior:

```

AddValue(Translation,
(DefaultValue(Translation) - value(Translation))/MovementChange);
AddValue(Rotation, -value(Rotation)/MovementChange);

```

The PDL code of a behavior system, like the example shown above, consists of a set of calls to the procedure **AddValue** (**q**, **v**) which adds the value **v** to a quantity **q**. All these calls are executed at time instant **t** (using the current values of the quantities which are retrieved using the form **value(q)**) and the changes are effectively enacted at time **t+1**. Translation, Rotation, TouchSensor1, etc. are examples of quantities. Some quantities (such as PhotoLeft) are directly

---

<sup>1</sup>In earlier work (e.g. Brooks 1989) the term behavior was sometimes used instead of behavior system. This term confuses however the observed behavior with the physical structures (control programs, etc.) responsible for realising the behavior, and so we insist on the term behavior system.

connected to sensory outputs. Others (such as `DefaultTranslation`) are internal variables. And still others (such as `LeftMotorSpeed`) are directly coupled to actuators. Values of quantities range between -255 and 255. Integer arithmetic is assumed for all calculations. After each cycle of executing the `AddValue` procedures, the sensory quantities are updated using the latest readings from the sensors and the new values of the action parameter quantities are sent to the actuators. On our robots, a speed between 20 and 40 cycles per second is typically reached so that a very reactive response is achieved and decision making takes place in quasi real-time. A more elaborate example illustrating the dynamics approach is discussed in [Steels 1994b].

### 3 Behavior systems as vectors

The problem of designing or analysing behavior systems should now be apparent: How can we guarantee that a particular behavior system achieves its desired objective, particularly if there are other behavior systems which are active at the same time? The first step is to construct a mathematical model of a behavior system. Quantities obviously correspond to mathematical variables. Because of hardware restrictions on the robot, they have as domain the integers between -255 and 255. Each procedure-call `AddValue (q, v)` can be modeled in terms of a one-dimensional vector which influences at time  $t+1$  the quantity  $q$  by the amount  $v$ . We denote this vector as  $v_i : q$ . `AddValue(q, v)` is then modeled by the equation  $v_i : q = v$ .  $v$  is possibly a complex expression. The values of the variables used in this expression are the values of the quantities at time  $t$ .

Thus the PDL code:

```
AddValue(Translation, (DefaultTranslation - value(Translation))/MovementChange);
```

is modeled with the equation:

```
v : Translation = (DefaultTranslation - Translation)/Movementchange
```

As can be seen from this example, translation between PDL code and vector equations and vice-versa is straightforward. For other languages, such as the Behavior Language of Brooks [Brooks 1990], the translation would be more difficult, particularly if explicit timers are involved, but it might still be possible.

### 4 Equilibrium analysis

Equilibrium analysis is common in many areas in which dynamical systems are investigated, most notably economics. Equilibrium analysis decomposes a complex dynamical system into dynamical subsystems. A subsystem is chosen by selecting a set of interrelated variables which form a coherent whole. The interrelated variables are in (dynamic) equilibrium, in the sense that there is a lack of tendency to change as long as certain conditions outside the chosen subsystem remain constant. An equilibrium state does not mean that the values of the variables do not change, rather that the subsystem no longer forces the variables into changes into a different direction. Moreover an equilibrium state may never be reached in practice because other subsystems may constantly push variables outside the range of the equilibrium conditions. It is nevertheless important to know the equilibrium state in order to identify in which direction the dynamics of the subsystem evolves.

We now apply equilibrium analysis to the study of behavior systems. Obviously, a behavior system can be viewed as a dynamical subsystem because it groups an interrelated set of variables. A behavior system reaches an equilibrium state if it exerts no influence on the dynamics. Based on the formalisation introduced in the previous section, this means that the sum of all the

vectors which form part of the behavior system are equal to 0. It follows that the equilibrium conditions can be determined by solving the vector equations for

$$\sum v : q_j = 0 \quad (1)$$

$j$  ranges over all quantities  $q$ . *Correctness conditions* for behavior systems can be formulated in terms of equilibrium states. For example, a *stabiliser* is a behavior system which influences the dynamics to bring one or more quantities to specific values or to a range of values [Steels 1993]. For example, forward movement is achieved by bringing rotation to 0 and translation to the default forward translation speed. In this case, we can formulate the following correctness condition: *A stabiliser should be in an equilibrium state as long as its corresponding behavior is observed.* The next section contains two worked out examples.

## 5 Examples

The forward movement behavior system is a stabiliser and therefore equilibrium should be reached when there is forward movement at the default speed. The vector equations are:

$$v_1 : Translation = (DefaultTranslation - Translation) / MovementChange$$

$$v_1 : Rotation = -Rotation / MovementChange$$

It is easy to see that  $v_1 : Translation = 0$  iff  $Translation = DefaultTranslation$  and  $v_1 : Rotation = 0$  iff  $Rotation = 0$ . These conditions determine therefore the equilibrium state for the forward movement behavior system. We also see that the behavior system influences the dynamics towards the equilibrium state: If  $Translation > DefaultTranslation$ ,  $v : Translation$  is negative. If  $Translation < DefaultTranslation$ ,  $v : Translation$  is positive. Similarly if  $Rotation > 0$ ,  $v : Rotation$  is negative. If  $Rotation < 0$ ,  $v : Rotation$  is positive.

### Phase space diagrams

A phase space diagram shows the evolution of the dynamics for all the possible states of a system [Nicolis and Prigogine1985]. A phase space diagram can be constructed easily for each vector determined by a behavior system. This vector constitutes one dimension of the phase space. The other dimensions are given by the other quantities which play a role in the dynamics. When there is only one quantity, the phase space diagram is a graph. Two quantities can still be represented using a 3-dimensional representation. The x- and y-axis represent the two quantities concerned and the z-axis represents the vector quantity. For more dimensions, phase space diagrams can no longer be visualised, but often it is interesting to single out two crucial variables and plot the evolution of the vector quantity on the z-axis.

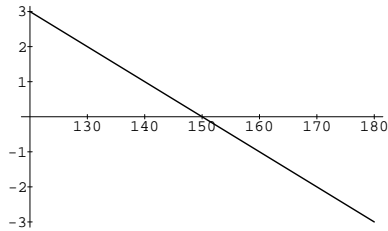
The phase space diagram in figure 1 shows the evolution of  $v_1 : Translation$  in relation to Translation. DefaultTranslation and MovementChange are parameters assumed to be constant and therefore do not need to be represented. The diagram shows the equilibrium state at the intercept with the x-axis and clearly visualises the evolution towards the equilibrium state. Positive values of  $v_1 : Translation$  mean increase in Translation. Negative values mean a decrease.

The phase space diagram in figure 2 shows the evolution of  $v:Rotation$  in relation to Rotation. This diagram is very similar to the previous one, as expected.

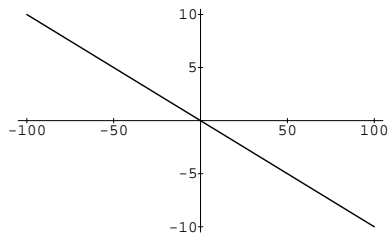
### Example: Motor Transduction

Translation and Rotation are internal quantities which are related to the two actuator parameters LeftMotorSpeed and RightMotorSpeed. We need therefore a transducer establishing this relation. This is implemented using the following PDL code:

```
AddValue(LeftMotorSpeed, value(LeftMotorSpeed) + value(Translation) -
```



**Figure 1.** Phase space diagram for  $v_1$  : *Translation* (y-axis) in function of Translation (x-axis). The equilibrium state is seen at the intercept with the x-axis. MovementChange is equal to 10 and DefaultTranslationspeed to 150. Only Translation values in the range 120-180 are plotted.



**Figure 2.** Phase space diagram for  $v_1$  : *Rotation* (y-axis) in function of Rotation (x-axis) for the range -100,100. The equilibrium state is seen at the intercept with the x-axis.

$(\text{value}(\text{LeftMotorSpeed}) - \text{value}(\text{RightMotorSpeed})) - \text{value}(\text{Rotation})$   
/2);

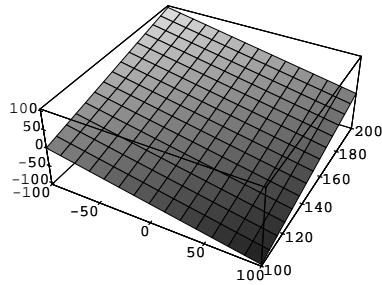
And similarly for the RightMotorspeed. The mathematical model can best be decomposed into two vector equations:

$v_1 : \text{LeftMotorSpeed} = -\text{LeftMotorSpeed} + \text{Translation}$     $v_2 : \text{LeftMotorSpeed} = -[(\text{LeftMotorSpeed} - \text{RightMotorSpeed}) - \text{Rotation}]/2$

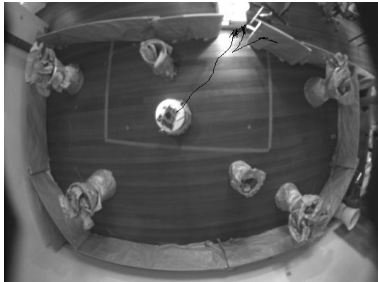
The transduction relation prescribes two conditions: 1.  $\text{LeftMotorSpeed} = \text{Translation}$ , 2.  $\text{Rotation} = \text{LeftMotorSpeed} - \text{RightMotorSpeed}$ . It is easy to see that  $v_1 = 0$  iff condition 1 is satisfied and  $v_2 = 0$  iff condition 2 is satisfied. The equilibrium conditions correspond therefore directly to the satisfaction of the transduction relation. If no equilibrium state holds, the behavior system will influence the LeftMotorSpeed quantity towards this state.

A phase space diagram for  $v:\text{LeftMotorSpeed}$  or  $v:\text{RightMotorSpeed}$  is more difficult to construct because there are four variables that play a role. But let us assume that  $\text{LeftMotorSpeed} = \text{RightMotorSpeed}$ , and then plot  $\text{Translation} - \text{DefaultTranslation}$  (y-axis) and  $\text{Rotation}$  (x-axis) against  $v:\text{LeftMotorSpeed}$  (z-axis). This gives the useful diagram in figure 3. The diagonal where  $v:\text{LeftMotorSpeed} = 0$  corresponds to states where the transduction relation is satisfied.

Equilibrium analysis can also be used for studying the relationship between different behavior systems. We expect for example that forward movement together with motorspeed transduction establishes a forward direction at the default speed. Again we focus on  $v:\text{LeftMotorSpeed}$ .  $v:\text{LeftMotorSpeed}$  will be equal to 0 if  $\text{LeftMotorSpeed} = \text{Translation}$ ,  $\text{Rotation} = 0$ , and  $\text{LeftMotorSpeed} = \text{RightMotorspeed}$ . Based on the analysis of  $v:\text{Translation}$  and  $v:\text{Rotation}$  in the forward movement behavior, Translation will be constant (i.e.  $v:\text{Translation}$  will no longer change) if  $\text{Translation} = \text{DefaultTranslation}$ . Rotation will be constant if  $\text{Rotation} = 0$ .



**Figure 3.** Phase space diagram for  $v:LeftMotorSpeed$  (z-axis) in function of Translation - DefaultTranslation (y-axis) and Rotation (x-axis). The 0-diagonal on the z-axis corresponds to the conditions when the transduction relation is satisfied.



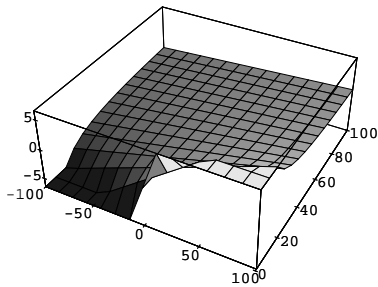
**Figure 4.** Phototaxis towards the light source mounted on top of the charging station. A characteristic emergent zig-zag behavior is observed.

The equilibrium state of forward movement and motorspeed transduction is therefore reached if  $LeftMotorSpeed = RightMotorSpeed = DefaultTranslation$ . This is exactly the condition that we expect of forward movement behavior: The agent should move forward (which is achieved when both motors are equal) at the default speed given by the value of DefaultTranslation.

**Example: Phototaxis.**

The next example is a disturber which implements phototaxis. Phototaxis is achieved by a combination of forward movement and pointing towards the light source. The latter behavior can be achieved by turning left or right depending on the difference in intensity of the left and right photosensors. Turning left (or right) means to influence rotation. Combination of forward movement behavior and heading behavior gives rise to a zig-zag phototaxis towards a light source as illustrated in figure 4.

Figure 5 shows the phase space diagram for phototaxis as influenced by energy availability. The light is attached to a charging station and the agent should be going towards the charging station when the energy availability (translated to the quantity EnergyDrain) is low. The vector equation for this behavior is:  $v_3 : Rotation = (PhotoFactor * (PhotoLeft - PhotoRight)) / EnergyDrain$



**Figure 5.** Phase space diagram for  $v_3$  : *Rotation* (z-axis) as a function of  $v_2$  : *Rotation* (x-axis) and *EnergyDrain* (y-axis). This diagram visualises that the impact will increase as *EnergyDrain* decreases.

## 6 Conclusions

Equilibrium analysis investigates the conditions under which a subsystem of a complex dynamical system reaches an equilibrium state, i.e. a state in which the change in direction imposed on quantities no longer changes, assuming parameters and variables external to the subsystem are constant. We applied equilibrium analysis to the design of behavior-oriented autonomous agents by proposing a mathematical modelisation of the control programs of behavior systems in terms of vectors. A behavior system reaches equilibrium when the sum of all its vectors is equal to 0. A phase space diagram visualises the evolution towards equilibrium. Future work intends to relate the phase space diagrams and the equilibrium conditions with a study of trajectories recorded while the agent interacts in real time with the environment.

## 7 Acknowledgement

Our work on autonomous agents is a group effort which includes engineers, computer scientists and biologists. The author is particularly indebted to David McFarland, Danny Vereertbrughen, Filip Vertommen and Peter Stuer for making various indirect contributions the present paper. Research has been funded by the Belgian Government IUAP Centre of Excellence grant to the VUB AI lab.

## References

- [1] Brooks, R. (1991) *Intelligence without reason*. Proceedings of IJCAI-91. Morgan Kaufmann, San Mateo Ca. p. 569-595.
- [2] Brooks, R. (1990) *The Behavior Language; User's Guide*. MIT AI Lab. Memo 1127.
- [3] Kaelbling, L. (1992) *An Adaptable Mobile Robot*. In: Varela, F.J. and P. Bourgine (eds.) (1992) *Toward a Practice of Autonomous Systems*. Proceedings of the First European Conference on Artificial Life. MIT Press/Bradford Books, Cambridge Ma. p. 41-47.
- [4] Nicolis, G. and I. Prigogine (1985) *Exploring Complexity*. Piper, Munchen.

- [5] Nilsson, N. (ed.) (1984) Shakey the Robot. SRI AI center. Technical Note 323.
- [6] Steels, L. (1992b) The PDL reference manual. VUB AI Lab memo. 92-5.
- [7] Steels, L. (1993) Building Agents with Autonomous Behavior Systems. In: Steels, L. and R. Brooks (eds.) (1993) The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents. Lawrence Erlbaum Associates, New Haven.
- [8] Steels, L. and R. Brooks (eds.) (1994) The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents. Lawrence Erlbaum Associates, New Haven.
- [9] Steels, L. (1994) The artificial life roots of artificial intelligence. Artificial Life Journal, Vol 1,1. MIT Press, Cambridge.
- [10] Steels, L. (1994b) A case study in the behavior-oriented design of autonomous agents. Submitted to SAB-94.