Multifunctionality: a fundamental property of behavior mechanisms based on dynamical systems

Herbert Jaeger
GMD, Sankt Augustin, Germany
herbert.jaeger@gmd.de

Abstract

The mechanisms supporting robot behaviors are increasingly often designed as dynamical systems. Such mechanisms are inherently multifunctional. This means that they can exhibit different qualitative behavior in different circumstances. Multifunctionality makes system design difficult. On the other hand, it may be beneficial for adaptiveness, since it allows qualitative changes in a robot's behaviors without changing the supporting mechanisms.

1. Introduction

The "behavior based" approach to designing mobile robots (Brooks, 1991) has been very fertile. However, a formal theoretical foundation, comparable to the role of logics in traditional AI, has been lacking in the beginning. A promising road toward such a theoretical foundation is to view agents as dynamical systems (Beer, 1997)(Steinhage and Schöner, 1997). This perspective is also being developed in cognitive science (van Gelder and Port, 1995). In these approaches, agents (animals, humans, or robots) are described with formalisms which lend themselves to an analysis in terms of dynamical systems theory. Such formalisms include differential equations (e.g., (Jaeger and Christaller, 1998)(Large et al., 1997)), recurrent neural networks (e.g. (Beer, 1995)), and stochastic automata (e.g. (Baasye et al., 1995)). The most common kind of formal analyis concerns the description of attractor structures emerging from learning or evolution (e.g. (Robertson et al., 1993)(Smithers, 1995)).

In the present article I wish to point out a basic property of dynamical-systems based robots, namely, the inherent multifunctionality of input-driven dynamical systems. I shall demonstrate the ubiquity of this property, and discuss the difficulties as well as the chances that it might afford.

First, a clarification of terminology. By the term *functionality*, I refer to a combination of a *mechanism* with a particular *purpose*. In the context of this article, I deal with computational mechanisms. Examples are all kinds of functions and algorithms, dynamical systems, neural networks, etc., when they are physically implemented on a robot. There, they (usually) serve a particular purpose, e.g. extracting features from a sensor signal, or producing a base oscillation for a leg controller, etc. *Multifunctionality* refers to the fact that a particular mechanism can serve many purposes.

This fact is well known. It has been referred to under various names, e.g. "relative systems" (Shimizu, 1993), "functional indeterminacy", and "multifunctionality" (Pasemann, 1994).

The multifunctionality described by those authors results from bifurcations. In this article, I wish to point out that multifunctionality even arises without bifurcations, due to nonlinearities and exchanges in relative timescales of system inputs.

In order to illustrate this fact, I shall describe a "pushing boxes" behavior of a robot, which was programmed as a dynamical system according to the "dual dynamics" design scheme (section 2). The behavior is ruled by simple differential equations. One of them is inspected in section 3, where I describe how the differential equation serves a particular purpose in the robot. In section 4 I will show that the same equations may support different functionalities when fed with different input. In the final discussion section I argue that this kind of multifunctionality may turn out to be beneficial for adaptiveness, although it presents difficulties for design.

2. Background: dual dynamics and the Black Knight

I start this section with a brief introduction to the "dual dynamics" (DD) scheme. A more detailed account can be found in (Jaeger and Christaller, 1998).

DD is a formal scheme for designing behavior control systems for mobile robots. The scheme prescribes how to specify a collection of behaviors as a comprehensive dynamical system. It consists of many coupled subsystems, each of which is responsible for controlling a particular behavior. These behavior subsystems are specified through ordinary differential equations.

Like in many behavior-oriented control architectures, behaviors are ordered in levels in the DD scheme, too. At the bottom level of a DD behavior hierarchy, one finds *elementary* behaviors. These are sensomotoric coordinations with direct access to external sensor data

and actuators. Typical examples are `move_forward` and `turn_left`. Higher levels are constituted by increasingly comprehensive behaviors. They also have access to sensoric information but cannot directly activate actuators. Typical examples are `work` and `replenish_energy`.

Elementary behaviors are different from higher-level behaviors in that they are made from two subsystems, which serve different purposes. This has given the approach its name, "dual dynamics".

The first of these subsystems is called the *target dynamics*. It calculates target trajectories for all actuators which are relevant for the particular behavior. Its output consists of as many variables as there are degrees of freedom to be controlled.

A requirement for the target dynamics is that this system must not undergo bifurcations. This is what makes elementary behaviors elementary. For instance, the target trajectories of `turn_left` in a simple 2-wheeled robot, which moves on a flat surface, are likely to remain qualitatively unchanged in different instances of the maneuver. Thus, `turn_left` would be a good candidate for an elementary behavior in such a simple robot. By contrast, in a walking machine which has to cope with different surfaces, it is likely that there will be qualitatively different maneuvers for turning left in different circumstances. Each of them would thus yield a separate elementary behavior.

The other subsystem of an elementary behavior is its *activation dynamics*. It regulates a single variable, the behavior's *activation*. The equation ruling this variable should be written in a way that the variable displays a dynamic range between 0 and 1. A value of 1 means that the behavior is fully active, whereas 0 means that it is completely inhibited.

The activation dynamics is allowed to undergo bifurcations. The control parameters which induce these bifurcations are the activation variables of higher-level behaviors. This is the core idea behind DD. The ways of how, exactly, these bifurcations are induced by changing control parameters, are highly constrained by the DD formalism. These constraints and the fact that bifurcations are confined to the one-dimensional activation dynamics subsystems, warrants the transparency of the DD design scheme.

I would like to emphasise that an elementary behavior is not "called to execute" from higher levels. The level of elementary behaviors is fully operative on its own and would continue to work even if the higher levels were cut off. The effect of higher levels is not to "select actions", but to change the overall characteristics of the elementary level, by inducing bifurcations in that level.

This concludes the introduction to DD. In the remainder of the section, I sketch the Black Knight (BK) robot and its `pushbox` behavior.

BK (now deceased) was a typical 2-df robot made from Lego bricks. It was built and maintained at the VUB AI Lab[1] by Peter Stuer and Dany Vereertbrugghen. Programming was done using the PDL programming language, which was also developed at the VUB AI Lab. PDL is well suited to implement control systems written in differential equations[2].

BK's basic set of tasks is to wander around in an arena limited by plywood walls; avoiding obstacles while wandering; knocking into light-emitting "pushboxes"; and recharging at a power station when the batteries run low. Fig. 2 shows a view of this setup.

Figure 1 The Black Knight approaching the charging station (brightly lit construction). A pushbox emitting a thin horizontal line of red light is seen at the right.

One of BK's behaviors is to "work" by reapeatedly knocking into one of several cylindrical pushboxes scattered in the arena. The pushboxes emit a modulated red light. BK can distinguish pushboxes from other light-sources due to this modulation. When a pushbox is knocked repeatedly, it gradually dims its light until it cannot longer be recognised by BK. A typical `pushbox` episode proceeds in three phases: (i) the robot gets into recognition distance of a light-emitting pushbox, and steers toward it, (ii) then drives head-on into it (a "knock"), retracts a bit, and repeats a few times until (iii) the pushbox is almost completely dimmed, thereby becoming an ordinary obstacle, from which the robot turns away.

3. A closer look at the pushing boxes behavior

In this section I explain the activation dynamics of the `pushbox` behavior, a simple differential equation which

[1] http://arti.vub.ac.be/~cyrano/robot_home.html

[2] The implementation was done for teaching purposes and is richly documented. It is obtainable at http://www.gmd.de/People/Herbert.Jaeger/Resources.html

successfully served its intended purpose. However, in the next section it will turn out that this simple equation might support quite different functionalities, too.

The activation dynamics has two kinds of inputs. One kind of input enables the pushbox behavior to detect the pushboxes by virtue of the modulated light they emit. More specifically, the readings of the narrow-angled and wide-angled photodiodes are preprocessed to yield two signals M_l and M_r, which roughly correspond to "modulated red light is seen at the left (right, respectively) before the robot". Furthermore, while M_l and M_r rise sharply with modulated light influx, they decay exponentially, providing a kind of "afterimage" or "short-term memory" effect.

M_l and M_r are used to alert the pushbox behavior by switching on its activation, when even only a little modulated light is seen. Technically, in the activation dynamics (1) the sum $M_l + M_r$ is fed into a steep sigmoid σ which essentially returns 1 when $M_l + M_r$ rises over a certain small threshold, and returns 0 otherwise.

$$\begin{aligned} \dot{A}_{pbx} &= 20 \cdot A_{work} \cdot (\sigma(M_l + M_r) - A_{pbx}) \\ &\quad + decayterm \end{aligned} \tag{1}$$

The other input into the activation dynamics is the activation A_{work} of the `work` behavior. Mathematically, it appears as a factor. If the `work` behavior is not active, then $A_{work} = 0$, and (1) is dominated by the decay term. Intuitively, that means that when BK is not in working mood, it will not be tempted by pushboxes. If, however, the activation of `work` is high (i.e., about 1), then the dynamics of (1) is dominated by the main term $(\sigma(M_l + M_r) - A_{pbx})$. This term makes A_{pbx} follow $\sigma(M_l + M_r)$. Since the latter term basically can attain the values 0 and 1 due to the sigmoid, the resulting overall dynamics of (1), when `work` is activated, is a soft "toggling" between 0 and 1. This toggling switch is triggered when $M_l + M_r$ rises/falls over/under the small threshold of 0.03, which is the switching point of the sigmoid σ.

Figure 2 depicts a typical pushbox episode, recorded when BK was in working "mood" (i.e., when A_{work} was about 1). The variable R_{pbx} is the signal issued to the right motor by the target dynamics of the `pushbox` behavior. It is given in the figure to indicate BK's motor action during that pushbox episode, which consisted in an initial right curve for the first approach until the first knock, followed by a back-and-forth maneuver for the second knock, and a final retraction from the pushbox after it became invisible. The activation A_{pbx} quickly went to 1 when M_r started to climb, and quickly fell to 0 after modulated light "afterimages" had died out. In sum, the dynamics specified in (1) behaved in the way it was supposed to behave, namely, as a "toggle switch" for A_{pbx}.

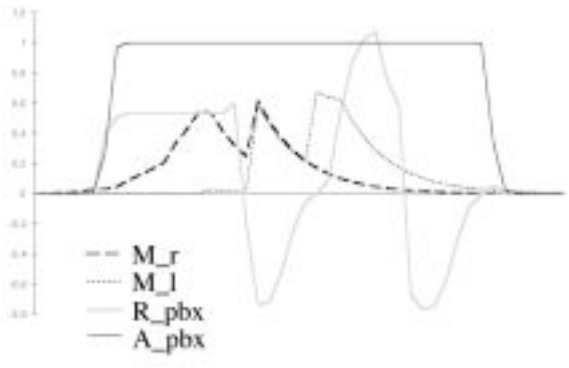

Figure 2 A trace of a pushbox episode comrpising two knocks. Duration is approximately 8 seconds. For variables see text.

4. Other functionalities supported by the mechanism

Now I shall re-consider equation (1) and list some functionalities which it might assume, other than the toggling functionality just described. In each case, I briefly state what characteristics the input into (1) must have, in order to make (1) assume that functionality.

Since the names of variables in (1) refer to the functionality within BK, it is better to rename them, in order to avoid confusion. For the sake of simplicity, I also join $M_l + M_r$ into a single variable. This gives the following condensed version of (1):

$$\dot{x} = 20 \cdot a \cdot (\sigma(b) - x) + decayterm \tag{2}$$

With a little experience in analysing differential equations, one will find it easy to detect many other functionalities (2) might acquire, besides the "toggling switch" characteristics. Here is a list of some of them:

Low-pass filter. Assume again that a remains fixed at 1, and that b is a signal of mixed frequencies with a small amplitude averaging at a value of 0.03, i.e. the threshold of σ.

In these circumstances, (2) works as a low-pass filter. Roughly, frequencies with a period greater than 1/20 will be suppressed, while lower frequencies will be followed by x.

Adaptable low-pass filter. Just like before, assume that input b is a signal of mixed frequencies. However, this time allow a to take various values between 0.1 and 10.

Then, (2) works as a low-pass filter like before, but with cutoff frequencies set by a in a period range from 1/2 to 1/200.

Signal amplifier. Again, let a be fixed at 1. Assume that b is a weak signal which slowly (relative to time increments of 1/20) varies in a range of 0.03 ± 0.001. In this context, the mechanism (2) will appear as an amplifier, with x following the input signal in a range of $0.5 \pm 0.001 \times S$, where S is the steepness of σ.

Leaky spike integration. Assume that $a = b$, and that b is some kind of "spiking" signal, i.e. it consists of short pulses, separated by refractory periods, which are long compared to spike duration, and where the signal is zero. Assume further that spike duration is shorter by at least an order of magnitude than 1/20 time units, and that the length of the refractory period is of the same order of magnitude as the time constant of the decay term in (2).
Under these conditions, (2) works as a leaky integrator of spikes. More concretely, the current value of x will measure the average number of spikes which occured in the past, with more recent spikes weighted stronger. The measurement is highly nonlinear, saturating at a value of 1.

Two factors are crucial for multifunctionality in these examples, namely, nonlinearities (represented by the sigmoid in (2)), and changes in relative timescales of a vs. b vs. the decay term vs. the overal system time scale fixed by the time constant 20.

Note that through all these examples, (2) does not bifurcate: the phase portraits of the system are characterised by a single point attractor through all variations of input, in all examples.

5. Discussion

The preceding sections illustrate that even simple dynamical systems can support different functionalities. These differences in functionalities need not result from bifurcations, but can arise from different characteristics of system input only. If bifurcations were included into the picture, multifunctionality would become even more pervasive.

Multifunctionality obviously makes a dynamical-systems oriented design more difficult, compared to design schemes which rely on functions in the mathematical sense, like e.g. typical C procedures or feedforward neural networks. Likewise, recurrent neural networks (which are dynamical systems) are notoriously more difficult to train than feedforward networks. A designer, who wishes to obtain a specific input-output trajectory from a differential equation, has to take into account the system's internal state evolution, which is connected to the desired IO-characteristics in no generally treatable way. Furthermore, the designer has to be careful about the exact dynamical properties and timescales of inputs, in order to fish a particular functionality from the reservoir of potential functionalites of a dynamical system.

Considering these problems, what reasons are there for using dynamical systems? One obvious advantage of dynamical systems is that they are intrinsically temporal formalisms (detailed discussion in (van Gelder, 1998)). This renders them natural for achieving functionalities which are time-sensitive, in particular, for motor control; and renders them charming for researchers who believe that cognition grows out of bodily activity. However, I will not further this aspect here, but instead point out another possible advantage of dynamical systems.

Multifunctionality of mechanisms affords a robot with a reservoir of novel ways of behaving. As I hope to have demonstrated, this reservoir is a rich one. A strategy for a robot to exploit multifunctionality might consist of the following three steps: (i) get into novel situations, e.g. by playing or exploration, (ii) discover that the behavior-generating mechanisms respond in novel ways due to multifunctionality, and (iii) reinforce and memorise novel responses if they are beneficial, and replay them when similar situations occur. I would call this a *discover-and-memorise* strategy for generating adaptive novel behaviors.

Obtaining new functionalities by a discover-and-memorise strategy should be compared to the more familiar way of obtaining novel functionalities, namely, by evolutionary strategies. I use here the term "evolutionary" in a broad sense, to denote any scheme which first randomly *modifies* an existing system and then *tests* the result by evaluating the resulting fitness. Evolutionary schemes for finding new functionalities do not necessarily require many generations of agents, but can work inside a single agent (e.g., (Steels, 1997)).

A big difference between discover-and-memorise vs. modify-and-test strategies is that in the former, an already existing (but dormant) reservoir of novelty is tapped, whereas in the latter, novel functionalities are added to the system. Conversely, this entails that discover-and-memorise leaves existing functionalities intact, whereas modify-and-test may be destructive. Thus, the former seems intrinsically safer than the latter. It might also be faster since new functionalities are harvested together with the situations in which they arise. By contrast, modify-and-test is apt to produce undetectable functionalities for situations which will never occur.

One price to be payed for the benefits of multifunctionality is that it may be difficult to keep subsystems within the operative range of a particular functionality. Keeping multifunctionality within bounds may be one of the reasons why homeostasis is so important in animals.

To my knowledge, a discover-and-memorise strategy capitalising on multifunctionality has not yet been realized on a robot. A prerequisite for investigations in that direction is to be able to program robots on a dynamical systems basis in a convenient fashion. The dual dynamics

design scheme is a step toward that goal.

Acknowledgments. I wish to express my deep gratitude to Luc Steels, Peter Stuer and Dany Vereertbrugghen at the VUB AI Lab. Thanks to Tim van Gelder for insightful discussions, and an anonymous reviewer for very helpful comments. The work described in this article was mostly carried out under a postdoc grant from GMD, Sankt Augustin.

References

Baasye, K., Dean, T., and Kaelbling, L. (1995). Learning dynamics: System identification for perceptually challenged agents. *Artificial Intelligence*, 72:139–171.

Beer, R. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72(1/2):173–216.

Beer, R. (1997). The dynamics of adaptive behavior: a research program. *Robotics and Autonomous Systems*, 20:257–289.

Brooks, R. A. (1991). Intelligence without reason. A.i. memo 1293, ftp'able at http://www.ai.mit.edu/, MIT AI Lab.

Jaeger, H. and Christaller, T. (1998). Dual dynamics: Designing behavior systems for autonomous robots. *To appear in Artificial Life and Robotics*.

Large, E., Christensen, H., and Bajcsy, R. (1997). Scaling the dynamic approach to autonomous path planning: planning horizon dynamics. In Pollack, M., editor, *Proceedings of IJCAI-97, Vol. 2*, pages 1360–1365. Morgan Kaufmann.

Pasemann, F. (1994). Neuromodules: A dynamical systems approach to brain modelling. In Herrman, H., Wolf, D., and Pöppel, E., editors, *Proceedings of the Workshop: Supercomputing in Brain Research – from Tomography to Neural Networks,*, pages 331–348. HLRZ, KFA Jülich, Germany, World Scientific.

Robertson, S., Cohen, A., and Mayer-Kress, G. (1993). Behavioral chaos: Behind the metaphor. In Smith, L. and Thelen, E., editors, *A Dynamic Systems Approach to Development: Applications*, pages 119–150. Bradford/MIT Press, Cambridge, Mass.

Shimizu, H. (1993). Biological autonomy: The self-creation of constraints. *Applied Mathematics and Computation*, 56:177–201.

Smithers, T. (1995). On quantitative performance measures of robot behavior. *Robotics and Autonomous Systems*, 15(1/2):107–134.

Steels, L. (1997). Synthesising the origins of language and meaning using co-evolution, self-organization, and level formation. In Hurford, J., editor, *Evolution of Human Language*. Edinburgh University Press, Edinburgh.

Steinhage, A. and Schöner, G. (1997). Self-calibration based on invariant view recognition: Dynamic approach to navigation. *Robotics & Autonomous Systems*, 20(2-4):133–156.

van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *To appear in Behavioural and Brain Sciences*.

van Gelder, T. and Port, R., editors (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*. Bradford/MIT Press.