

Comparing Different Cognitive Paradigms with a Virtual Laboratory

Carlos Gershenson

Centrum Leo Apostel, Vrije Universiteit Brussel

Krijgskundestraat 33, Brussels, 1160, Belgium

<http://student.vub.ac.be/~cgershen> E- mail: cgershen@vub.ac.be

Abstract

A public virtual laboratory is presented, where animats are controlled by mechanisms from different cognitive paradigms. A brief description of the characteristics of the laboratory and the uses it has had is given. Mainly, it has been used to contrast philosophical ideas related with the notion of cognition, and to elucidate debates on “proper” paradigms in AI and cognitive science.

1. Introduction

Virtual laboratories have been used in very different areas with different purposes. They are especially useful when they simulate situations which are difficult to reproduce, control, or observe. Cognitive science studies phenomena that fall in this category.

A virtual laboratory has been developed for the comparison of different cognitive paradigms. Experiments carried out in this virtual laboratory were used as “opaque thought experiments” for discussing the notion of cognition (Gershenson, 2003). Detailed information about the virtual laboratory can be found in Gershenson (2002).

2. A Virtual Laboratory

Following the ideas presented in Gershenson, González, and Negrete (2000), a virtual laboratory was developed for testing the performance of animats controlled by mechanisms proposed from different perspectives in a simple virtual environment. Programmed in Java with the aid of Java3D libraries, this software is available to the public, source code and documentation included, at <http://student.vub.ac.be/~cgershen/cogs/keb/>. The software allows the user to create and repeat controlled experiments for comparing the different animats in different situations in a friendly and informative fashion.

In the virtual laboratory, the user can create different phenomena, such as rocks (grey cubes), food sources (green spheres), rain (blue semitransparent cylinders), lightnings

(black cylinders), and spots of different colours (circles): randomly or in specific positions. These also can be generated randomly during the simulation at a selected frequency. Lightnings turn into rain after ten time steps, and rain turns into food after fifty time steps.

All the animats have an energy level, which decreases when their hunger or thirst are high, and is increased when these are low. An animat dies if its energy is exhausted. Eating food decreases their hunger. They can decrease their thirst by drinking under rains. Hunger and thirst are increased if they attempt to drink or eat “incorrect” stimuli. They lose energy if they touch lightnings or rocks. Basically, an animat in order to survive needs to eat when hungry, drink when thirsty, and avoid lightnings and rocks. We can say that they are cognitive systems if they are successful, because they would *know* how to survive (Gershenson, 2003). The animats can leave a coloured trail in order to observe their trajectories.

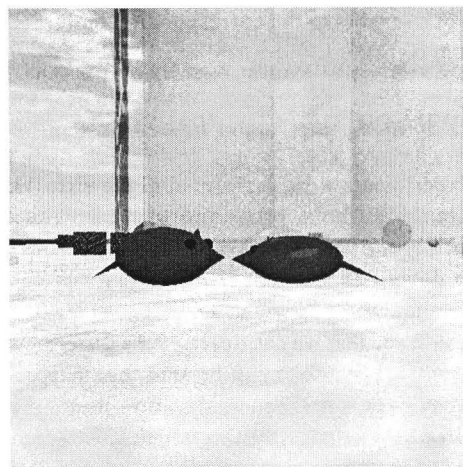


Figure 1. Screenshot of virtual environment.

There are many models that would solve the problem of surviving in such an environment, but it was decided to implement representative models of different paradigms in order to observe their differences and similitudes. These models are as follows: a **rule-based** system typical of traditional knowledge-based and expert systems; Maes' (1990) action selection mechanism, an already classical **behaviour-**

based system; an original architecture of recursive concept development (Gershenson, 2002) as an example of the novel **concept-based** approach; a simple feed-forward artificial **neural** network; and a **Braitenberg**-style architecture (Braitenberg, 1984). All the animats survive fairly well in the simple environment. A short description of each mechanism follows:

- The **rule-based** animats have a set of rules (*if... then*). These receive information from “cheater” sensors, *i.e.* with meaning given by the programmer (*e.g.* food perceived, hungry, obstacle close, etc.), and produce behaviours (*e.g.* explore, approach food, eat, avoid obstacle, etc.), with which a motor system has to deal.
- The **behaviour-based** animats also have “cheater” perceptual and motor systems, but the control is determined by a network of behaviours (*e.g.* approach food, eat, avoid obstacle, etc.) which inhibit and excite according to the types of connections they have.
- The **concept-based** animats only have basic sensors (*e.g.* redness, hardness, flavour, etc.), and they develop recursively concepts that they associate by reinforcement learning with predetermined behaviours.
- The **neural** animats have three sensor pairs and two motors. Each sensor perceives if a phenomenon (food, rain, or rock) is left or right of the animat. These signals, and signals from the internal medium (hunger and thirst) are inputs to a three-layered feed-forward neural network with fixed weights. The outputs go straight to the motors.
- The **Braitenberg** animats also have six sensors and two motors, but they are directly connected: food and rain sensors to the inverse motors (left to right, right to left), and rock sensors to the corresponding motors (left to left, right to right).

3. Comparison

Several experiments were performed in the virtual laboratory to compare the different properties of the architectures that control each type of animat (Gershenson, 2002). The reader is invited to download the virtual laboratory and compare the different architectures as well.

We could see that we can describe the different animats in the same terms, because it can be said that they perform the same behaviours, independently on how these were implemented. Also, distinguishing which architecture controls which type of animat is not possible for a naive observer.

We observed that there is no general “best” architecture for the simple task of surviving in the virtual environment. We can say that each animat is better in different situations. Yet it seems that this is more a consequence of the particular implementation than of the paradigm on which it stands, because the models can be adjusted and refined to any desired degree of detail. In order to judge which architecture is better, we need to refer to a particular context. Their performance

cannot be generally measured, but only relatively to specific tasks.

Some models were very easy to implement in software code, others not so much, but occasionally it is a different story if we want to implement an architecture in a real robot. Moreover, if a model works in a simulation and/or robot, it does not mean that animals function in the same way. Some models are very robust. Others would break up quite easily. Some models are quite good if we have just practical purposes, and this also depends on the experience of the engineer. Still, if we are interested in using them as explanatory models, the simplicity of their implementation might be secondary. Also, if we would like to increment the systems, for example to include more environmental stimuli and internal variables, some would need to be redesigned, others could be easily extended. Some models would have more ease in adapting to changes of their environment than others, but this does not mean that we cannot adjust different architectures in order to obtain the desired behaviour.

We can say that different models, architectures, and paradigms, study different aspects of cognition. Will we find ever a “best” model? It depends on our purposes and our context. Generally speaking, we can say that all approaches are useful, since they illustrate different aspects of cognition.

The observations in the virtual laboratory were useful for discussing and illustrating ideas related to the notion of cognition (Gershenson, 2003), and to clarify debates concerning the “proper” paradigm for studying cognition (Gershenson, 2002).

4. Conclusions

Virtual laboratories allow us to explore different avenues than physical laboratories. This virtual laboratory was useful for illustrating questions such as “what makes a system cognitive?”. It helped in showing that the observer plays a great deal in the cognition of the system, and that cognition is independent of the implementation of the system.

References

- Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.
- Gershenson, C. (2002). A Comparison of Different Cognitive Paradigms Using Simple Animats in a Virtual Laboratory, with Implications to the Notion of Cognition. *Unpublished MSc Thesis*. University of Sussex
- Gershenson, C. (2003). On the Notion of Cognition. Submitted to IJCAI'03.
- Gershenson, C., P. P. González, and J. Negrete (2000). Thinking Adaptive: Towards a Behaviours Virtual Laboratory. In Meyer, J. A. et. al. (Eds.), *SAB2000 Proceedings Supplement*. Paris, France. ISAB Press.
- Maes, P. (1990). Situated agents can have goals. *Journal of Robotics and Autonomous Systems*, 6 (1&2).

Towards domain-independent, task-oriented, conversational adequacy

Darsana P. Josyula¹, Michael L. Anderson², and Don Perlis^{1,2}

(1) Department of Computer Science

(2) Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
{darsana, mikeoda, perlis}@cs.umd.edu

Humans are often able to communicate effectively to solve a joint task, *even* when they do not speak much of a common language, for instance, when buying a train ticket in a foreign country. Although the conversation might not be fully fluent and error-free, the communication does not break down; instead, through clarifications, confirmations and inferences, the dialog can proceed towards solving the common task. Key to this is the ability to note an error (or miscommunication), assess it, and address it. We believe that these principles of human dialog repair can be used to enhance the conversational abilities of task oriented interactive systems, allowing them to interact more smoothly with human users (Perlis, Purang, & Andersen 1998). To this end, we are developing an agent that can act as an interface between the user and a task-oriented system, providing a bed for dialog correction and repair.

Our agent, ALFRED (Active Logic for Reason Enhanced Dialog) is built on a time-tracking, logical framework called active logic (Elgot-Drapkin *et al.* 1993; Elgot-Drapkin & Perlis 1990). Active logic works by combining inference rules with a constantly evolving measure of time (a “Now”) that can itself be referenced in those rules. As an example, from Now(*t*)—the time is now “*t*”—one infers Now(*t*+1). Special rules governing the persistence of formulas in the database, including rules for dis inheriting direct contradictions, along with a quotation mechanism which allows formulas to refer to one another, give active logic the expressive and inferential power to monitor its own reasoning, allowing it to watch for errors (such as mismatches between conveyed and intended meanings), note temporal conversational cues such as pauses that may signal a turn change, and to exert reasoned control over its past and upcoming inferential processes, including re-examination and alteration of beliefs and inferences.

Capabilities

Some of the capabilities that ALFRED currently has are as follows.

1. Understanding the use-mention distinction

Copyright © 2003, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

ALFRED is sensitive to whether a word is being used or mentioned, and interprets the relevant utterance accordingly. Following (Saka 1998), we have chosen to characterize the use-mention distinction in terms of the possible ostensions of words. That is, we consider *X* is being “used”, if the speaker intends to direct the thoughts of the audience to the extension of *X*; and *X* is being “mentioned”, if he intends to direct the thoughts of the audience to some item associated with *X* *other than its extension*. See (Anderson *et al.* 2002) for more details. The features that ALFRED uses to make this distinction are context, cues and meta-dialog.

2. Using meta-dialog

ALFRED is capable of engaging in meta-dialog with the human user when necessary, in order to identify miscommunication problems and perform dialog repairs.

3. Learning new words

ALFRED can learn new names that refer to already existing objects in the domain. It can also learn new ways to accept known commands. This feature can allow a given user to negotiate a more comfortable vocabulary for interaction, and helps to increase the vocabulary of the system as a whole. We are working on methods of structuring and manipulating conceptual relations to allow ALFRED to learn genuinely new objects and concepts.

4. Maintaining context

ALFRED maintains the context of the ongoing conversation by keeping track of user intentions (interpretations of past utterances), needs and expectations. A “need” is created when there is some requirement (like determining the meaning of a word) that has to be met before the system can interpret a user utterance. An “expectation” is created when the system expects a particular kind of response from the user.

5. Identifying miscommunication

ALFRED recognizes miscommunication problems by looking for contradictions in its interpretations of the user’s intentions (Traum *et al.* forthcoming).

6. Connecting to different domains

ALFRED functions as a kind of translator between human natural language and the specialized language of the task-oriented domain, and enhances the performance of the original system interface by incorporating a suite of dialog error detection and repair strategies. These can allow a dialog to continue even in the presence of various kinds of miscommunications.

The upshot of this is that ALFRED can act as a natural language interface between a user and *any* task-oriented system as long as the specialized language of the task-oriented system is specified explicitly to ALFRED. This specialized language can vary from menu driven commands in the simplest case to natural language-like commands in a more complicated scenario. However, since the same principles and reasoning strategies are being used in each case to enhance the dialog, orienting ALFRED to a different domain is as easy as changing the files specifying the specialized language for that domain.

Example

Consider a task-oriented system whose domain is defined by the following objects and commands.

- Domain objects
train: Metroliner, Northstar, Bullet; city: Baltimore, Richmond, Buffalo
- Domain commands
send: [send train city]

A stepwise example of command translation and intentional discernment follows:

1. User types "Send Metro to Baltimore"; the parser sends output to ALFRED
2. Since it is a request for an action, ALFRED first checks whether it is a command for the domain or for itself.
3. Since it is a domain command, ALFRED will get the structure of the command from the domain commands knowledge base. In this case, the command structure is *send* - [send train city].
4. ALFRED then tries to disambiguate the object references in the user utterance by finding the mapping from these object references to known domain objects of the types specified in the structure.
5. Since there is no direct mapping between "Metro" and any existing train in the domain, ALFRED now "needs" the meaning of the word "Metro". This "need" causes ALFRED to temporarily stop further interpretation of the utterance until this "need" gets satisfied.
6. This "need" then causes ALFRED to introspect (to examine its existing knowledge base or try to reason) about the meaning of the word "Metro". Since it does not get an answer after introspection, it asks the

user: "What does 'Metro' mean?", and "expects" the user to provide the answer. This "expectation" helps in keeping track of the context of the conversation, which is essential for understanding the dialog state, and has particular use in helping distinguish whether a word is being used or mentioned in an utterance.

7. The user says "Metroliner". The expectation, above, helps ALFRED to interpret this elliptical utterance as " 'Metro' means Metroliner". Hence ALFRED would learn this meaning so that at any later point in time, if the user says "Metro", ALFRED would be able to identify the correct referent by introspection.
8. Once ALFRED gets the meaning of "Metro", the "need" created above is satisfied, which in turn causes the further interpretation of that user utterance.

Conclusion

ALFRED can enhance the interactive capability of a task-oriented computer system by adding the ability to detect and recover from miscommunication problems, including ambiguous references, incompatible or contradictory user intentions (Traum *et al.* forthcoming), and the use of unknown words.

This technology can already be usefully applied to current application domains, such as home-control software, and we expect that the techniques employed can be refined and extended to handle more sophisticated domains.

References

- Anderson, M. L.; Okamoto, Y.; Josyula, D.; and Perlis, D. 2002. The use-mention distinction and its importance to HCI. In *Proceedings of the Sixth Workshop on the Semantics and Pragmatics of Dialog*.
- Elgot-Drapkin, J., and Perlis, D. 1990. Reasoning situated in time I: Basic concepts. *Journal of Experimental and Theoretical Artificial Intelligence* 2(1):75-98.
- Elgot-Drapkin, J.; Kraus, S.; Miller, M.; Nirkhe, M.; and Perlis, D. 1993. Active logics: A unified formal approach to episodic reasoning. Technical Report UMIACS TR # 99-65, CS-TR # 4072, Univ of Maryland, UMIACS and CSD.
- Perlis, D.; Purang, K.; and Andersen, C. 1998. Conversational adequacy: mistakes are the essence. *Int. J. Human-Computer Studies* 48:553-575.
- Saka, P. 1998. Quotation and the use-mention distinction. *Mind* 107:113-135.
- Traum, D.; Andersen, C.; Chong, Y.; Josyula, D.; O'Donovan-Anderson, M.; Okamoto, Y.; Purang, K.; and Perlis, D. forthcoming. Representations of dialogue state for domain and task independent meta-dialogue. *Electronic Transactions on Artificial Intelligence*.