Next: TCS 6 and 7.

Reactive Robot Programming:

Early Artificial Intelligence focused on intellectual activities, i.e., the sorts of things that, if a human did them, would be considered intelligent. Examples: playing chess and other board games, proving mathematical theorems, doing logic and solving puzzles, and so on.

Many of these activities are somewhat abstract. They take place on a board, or by reading and writing symbols. In other words you can do them with a keyboard and a display screen (or hardcopy printout).
That is, they didn't have to worry about the complexity of the physical world, which seemed irrelevant to intellectual activities (after all, other animals behave in the physical world at least as competently as we do).

Robots took a similar approach: an intellectual approach to controlling their behavior based on:
- memory
- knowledge
- planning (analyzing goals and how to achieve them)
- representations of the environment (e.g., internal maps).

A knowledge-based approach.

Problems:
In effect, robots were controlling behavior by thinking about it (reasoning about it).
A famous robot called "Shakey," which would find its way around a room, took hours to plan each move.

This turned out to be very slow (because thinking is slow).
We do not normally think about walking and many other ordinary activities; we just do them.

Another problem: memory and representations can both be wrong. Either the original data was wrong, or something has changed.
This leads to problems of knowledge correction and "non-monotonic reasoning" (referring to the fact that something that was true, isn't true anymore).
This takes more complicated processing ("thinking").

Paradox: Insects and other simple animals seem get along in the physical world very competently, and yet they have very tiny brains.
Animals seem to be able to behave very intelligently without intellectual reasoning processes. I.e., without "thinking" in the usual sense.

How do they do it? Part of the answer is that they don't try to represent their environment in detail in their tiny brains, and they don't do a lot of detailed planning, but in most cases they are reacting to their physical environments in fairly simple ways.

We can be fooled into thinking that their behavior is more complicated than it actually is, because a simple system in a complex environment, can appear to be behaving in a complex way.
Herb Simon (Nobel laureate), noted this, and used ants as an example.

This is an important new direction in AI and robotics for the past 15 years: looking at low-level reactive behavior as a basis for robot control. Basic sensory-motor skills may be a foundation for more abstract intellectual activities. This has been much more successful.

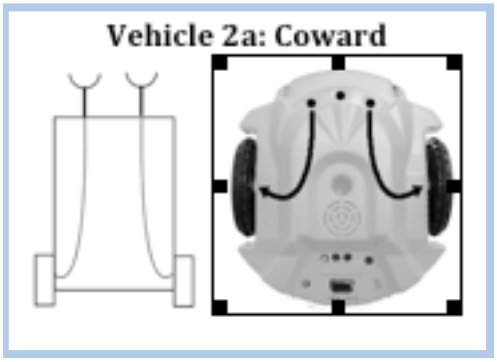
Braitenberg Vehicles:
These were thought experiments, designed to show how simple neural networks could lead to interesting behavior. More generally, how simple reactive control could lead to interesting behavior.

Reactive control means that the agent is primarily reacting to its environment, with little or no memory or planning.

By reacting to the environment (as opposed to planning based on internal representations of the environment) the agent's behavior is usually more robust.

B. enunciated the Law of Uphill Analysis and Downhill Design: It's often easier to design an agent that exhibits some behavior of interest, than to analyze an existing system exhibiting that behavior, and figuring out how it does it.

Examples: "Coward."



Normalization:
Sensors and other I/O devices often give information in an inconvenient form. For example, the light sensors on the scribbler return ints in the range 0 to 5000,

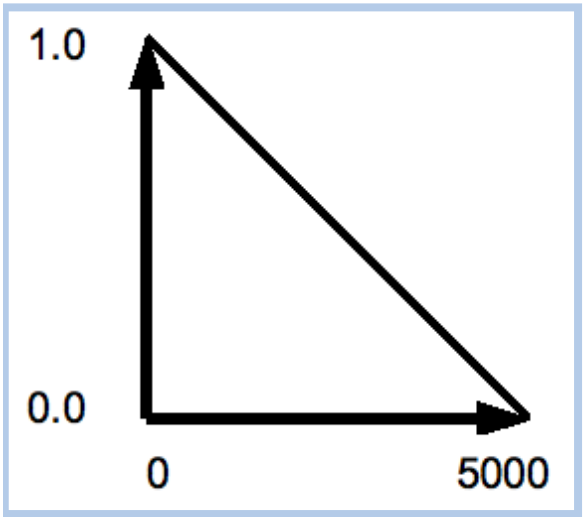
where 5000 is total darkness, and 0 is total brightness. That's the way they are made. Some other light sensors might work differently.

(1) This is unintuitive.
(2) I want to translate the light values into motor speeds, which are numbers in the range 0.0 to 1.0.

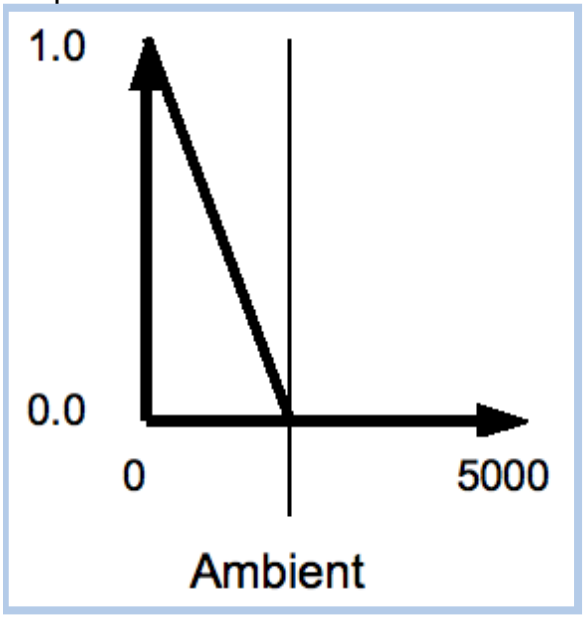
One solution is to normalize the outputs from the sensors so that the values are more meaningful and easier to use. For example, produce 0.0 for darkness and 1.0 for max. brightness.

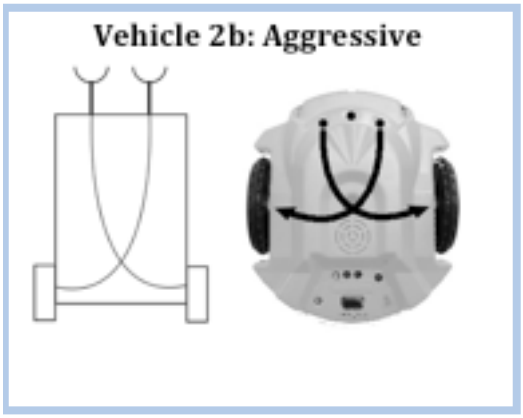
This also has the advantage that it confines the peculiarities of the sensor to the normalization function. Which means that if you change the sensors, or are combining info. from several sensors, it's easier and more reliable to do. This is an example of modularization. Keep the sensor details in the normalization function.

We could normalize the brightness sensor this way:



A more sophisticated normalizer is adaptive. It adapts to the ambient illumination, and only pays attention to brightness above the ambient level. It's better to respond to contrasts in the environment than to absolute values.





You can also have excitatory and inhibitory connections.