for the use of internal models to distinguish "your coffee cup" from "my coffee cup." This is where top-down, model-based perception occurs.

On a more practical note, Neisser's dichotomy suggests that the first decision in designing a behavior is to determine whether a behavior can be accomplished with an affordance or requires recognition. If it can be accomplished with an affordance, then there may be a simple and straightforward way to program it in a robot; otherwise, we will most certainly have to employ a more sophisticated (and slower) perceptual algorithm.

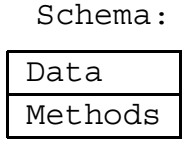
3.5 Schema Theory

Schema theory provides a helpful way of casting some of the insights from above into an object-oriented programming format.[6] Psychologists have used schema theory since the early 1900's. It was first brought to the serious attention of AI roboticists by Michael Arbib while at the University of Massachusetts, and later used extensively by Arkin and Murphy for mobile robotics, Lyons and Iberall for manipulation,[75] and Draper et al. for vision.[46]

SCHEMA

Schemas were conceived of by psychologists as a way of expressing the basic unit of activity. A *schema* consists both of the knowledge of how to act and/or perceive (knowledge, data structures, models) as well as the computational process by which it is uses to accomplish the activity (the algorithm). The idea of a schema maps nicely onto a class in object-oriented programming (OOP). A *schema class* in C++ or Java would contain both data (knowledge, models, releasers) and methods (algorithms for perceiving and acting), as shown below.

SCHEMA CLASS



A schema is a generic template for doing some activity, like riding a bicycle. It is a template because a person can ride different bicycles without starting the learning process all over. Since a schema is parameterized like a class, parameters (type of bicycle, height of the bicycle seat, position of the handlebars) can be supplied to the object at the time of instantiation (when an object is created from the class). As with object-oriented programming, the creation of a specific schema is called a *schema instantiation (SI)*.

SCHEMA INSTANTIATION (SI)

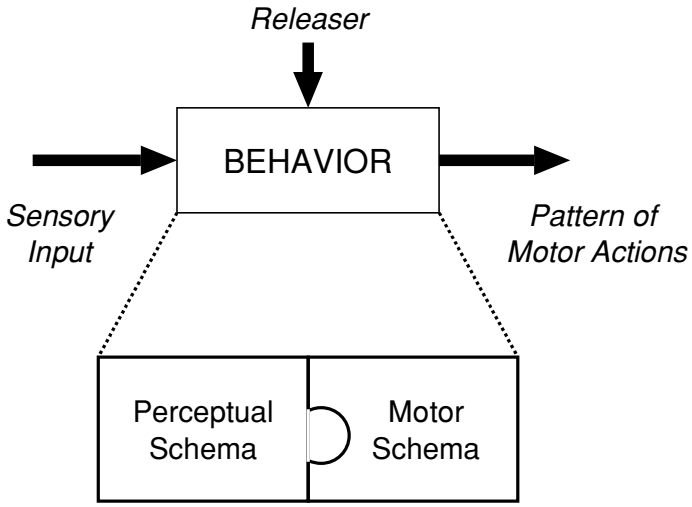


Figure 3.8 Behaviors decomposed into perceptual and motor schemas.

The schema instantiation is the object which is constructed with whatever parameters are needed to tailor it to the situation. For example, there could be a `move_to_food` schema where the agent always heads in a straight line to the food. Notice that the "always heads in a straight line" is a template of activity, and a reusable algorithm for motion control. However, it is just a method; until the `move_to_food` schema is instantiated, there is no specific goal to head for, e.g., the candy bar on the table. The same schema could be instantiated for moving to a sandwich.

3.5.1 Behaviors and schema theory

COMPOSITION OF A BEHAVIOR
MOTOR SCHEMA
PERCEPTUAL SCHEMA

In the Arbibian application of schema theory towards a computational theory of intelligence, a *behavior* is a schema which is composed of a motor schema and a perceptual schema. The *motor schema* represents the template for the physical activity, the *perceptual schema* embodies the sensing. The motor schema and perceptual schema are like pieces of a puzzle; both pieces must be together in place before there is a behavior. This idea is shown below in Fig. 3.8.

Essentially, the perceptual and motor schema concept fits in with ethology and cognitive psychology as follows:

- A behavior takes sensory inputs and produces motor actions as an output.

- A behavior can be represented as a schema, which is essentially an object-oriented programming construct.
- A behavior is activated by releasers.
- The transformation of sensory inputs into motor action outputs can be divided into two sub-processes: a perceptual schema and a motor schema.

In OOP terms, the motor schema and perceptual schema classes are derived from the schema class. A primitive behavior just has one motor and one perceptual schema.

Behavior::Schema

Data	
Methods	perceptual_schema() motor_schema()

Recall from IRMs, more sophisticated behaviors may be constructed by sequencing behaviors. In the case of a sequence of behaviors, the overall behavior could be represented in one of two ways. One way is to consider the behavior to be composed of several primitive behaviors, with the releasing logic to serve as the knowledge as to when to activate each primitive behaviors. This is probably the easiest way to express a "meta" behavior.

A meta-behavior composed of three behaviors can be thought of as:

Behavior::Schema

Data	releaser1 releaser2 releaser3 IRM_logic()
Methods	behavior1() behavior2() behavior3()

However, in more advanced applications, the agent may have a choice of either perceptual or motor schemas to tailor its behavior. For example, a person usually uses vision (the default perceptual schema) to navigate out of a room (motor schema). But if the power is off, the person can use touch (an alternate perceptual schema) to feel her way out of a dark room. In this case, the schema-specific knowledge is knowing which perceptual schema

to use for different environmental conditions. Schema theory is expressive enough to represent basic concepts like IRMs, plus it supports building new behaviors out of primitive components. This will be discussed in more detail in later chapters.

This alternative way of creating a behavior by choosing between alternative perceptual and motor schemas can be thought of as:

Behavior::Schema

Data	environmental_state
Methods	choose_PS(environmental_state) perceptual_schema_1() perceptual_schema_2() motor_schema()

RANA COMPUTATRIX

Arbib and colleagues did work constructing computer models of visually guided behaviors in frogs and toads. They used schema theory to represent the toad's behavior in computational terms, and called their model *rana computatrix* (rana is the classification for toads and frogs). The model explained Ingle's observations as to what occasionally happens when a toad sees two flies at once.[33] Toads and frogs can be characterized as responding visually to either small, moving objects and large, moving objects. Small, moving objects release the feeding behavior, where the toad orients itself towards the object (taxis) and then snaps at it. (If the object turns out not to be a fly, the toad can spit it out.) Large moving objects release the fleeing behavior, causing the toad to hop away. The feeding behavior can be modeled as a behavioral schema, or template, shown in Fig. 3.9.

When the toad sees a fly, an instance of the behavior is instantiated; the toad turns toward that object and snaps at it. Arbib's group went one level further on the computational theory.[7] They implemented the taxis behavior as a vector field: rana computatrix would literally feel an attractive force along the direction of the fly. This direction and intensity (magnitude) was represented as a vector. The direction indicated where rana had to turn and the magnitude indicated the strength of snapping. This is shown in Fig. 3.10.

What is particularly interesting is that the rana computatrix program predicts what Ingle saw in real toads and frogs when they are presented with two flies simultaneously. In this case, each fly releases a separate instance of the feeding behavior. Each behavior produces the vector that the toad needs to turn to in order to snap at that fly, without knowing that the other be-

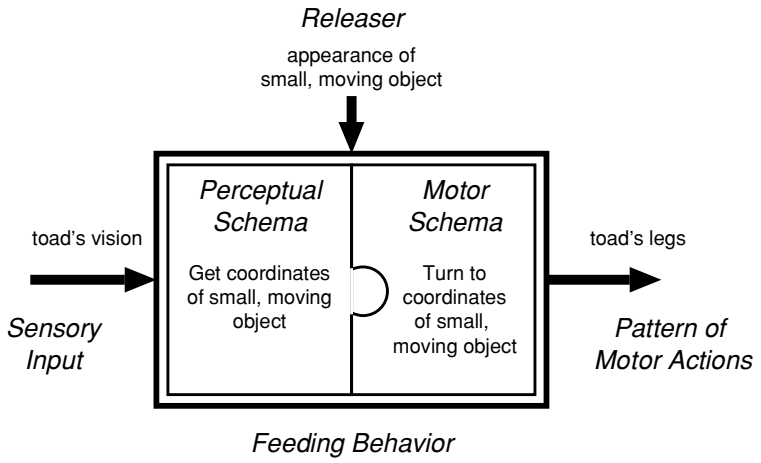


Figure 3.9 Toad's feeding behavior represented as a behavior with schema theory.

havior exists. According to the vector field implementation of the schema model, the toad now receives two vectors, instead of one. What to do? Well, rana computatrix summed the two vectors, resulting in a third vector in between the original two! The toad snaps at neither fly, but in the middle. The unexpected interaction of the two independent instances probably isn't that much of a disadvantage for a toad, because if there are two flies in such close proximity, eventually one of them will come back into range.

This example illustrates many important lessons for robotics. First, it validates the idea of a computational theory, where functionality in an animal and a computer can be equivalent. The concept of behaviors is Level 1 of the computational theory, schema theory (especially the perceptual and motor schemas) expresses Level 2, and Level 3 is the vector field implementation of the motor action. It shows the property of emergent behavior, where the agent appears to do something fairly complex, but is really just the result of interaction between simple modules. The example also shows how behaviors correspond to object-orienting programming principles.

Another desirable aspect of schema theory is that it supports reflex behaviors. Recall that in reflex behaviors, the strength of the response is proportional to the strength of the stimulus. In schema theory, the perceptual schema is permitted to pass both the percept and a gain to the motor schema. The motor schema can use the gain to compute a magnitude on the output action. This is an example of how a particular schema can be tailored for a behavior.

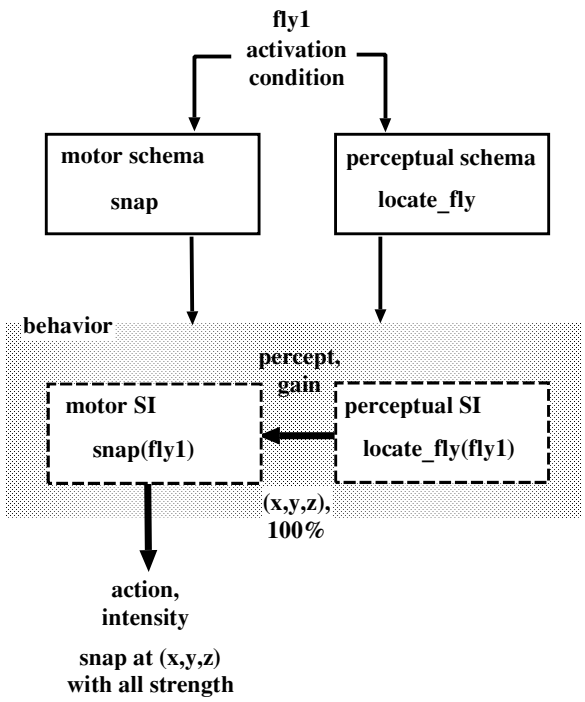


Figure 3.10 Schema theory of a frog snapping at a fly.

Schema theory does not specify how the output from concurrent behaviors is combined; that is a Level 3, or implementation, issue. Previous examples in this chapter have shown that in some circumstances the output is combined or summed, in others the behaviors would normally occur in a sequence and not overlap, and sometimes there would be a winner-take-all effect. The winner-take-all effect is a type of inhibition, where one behavior inhibits the instantiation of another behavior.

INHIBITION

Arbib and colleagues also modeled an instance of *inhibition* in frogs and toads.[7] Returning to the example of feeding and fleeing, one possible way to model this behavior is with two behaviors. The `feeding` behavior would consist of a motor schema for moving toward an object, with a perceptual schema for finding small, moving objects. The `fleeing` behavior would be similar only with a motor schema for moving away from the perception of large moving objects. Lesion studies with frogs showed something different. The `feeding` behavior actually consists of moving toward *any* moving object. So the perceptual schema is more general than anticipated. The frog would try to eat anything, including predators. The perceptual schema in the `fleeing` behavior detects large moving objects. It flees from them, but

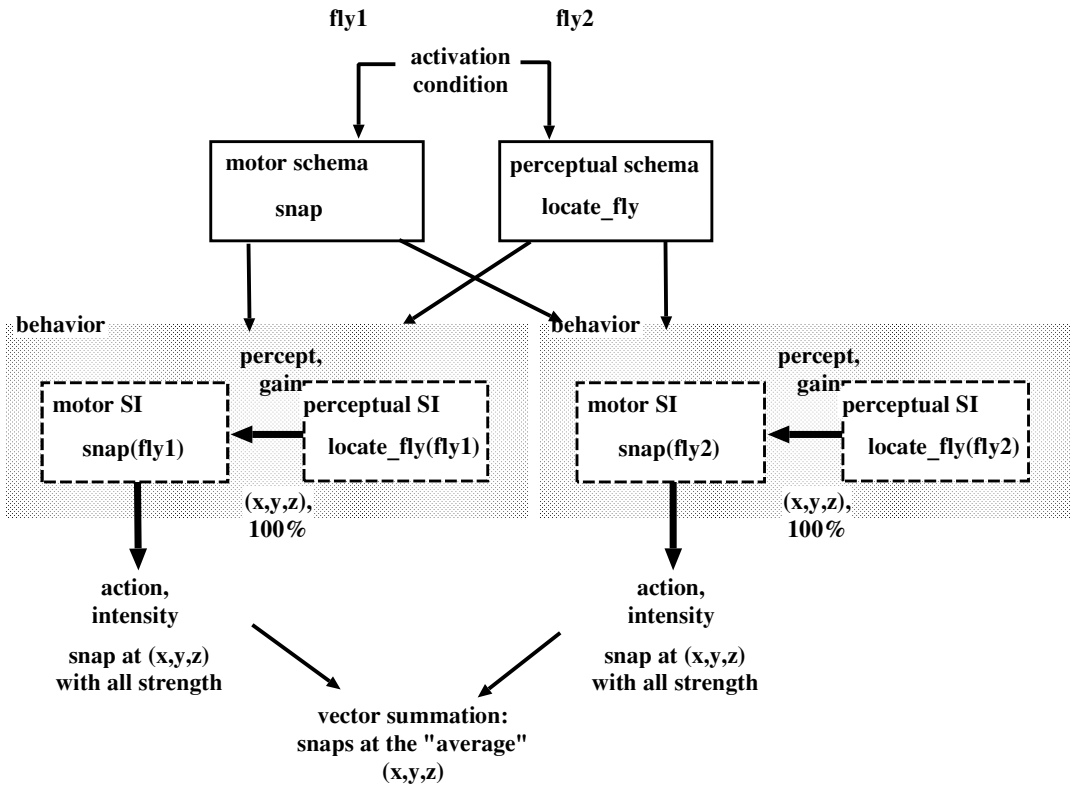


Figure 3.11 Schema theory of a frog snapping at a fly when presented with two flies equidistant.

it also inhibits the perceptual schema for feeding. As a result, the inhibition keeps the frog from trying to both flee from predators and eat them.

3.6 Principles and Issues in Transferring Insights to Robots

PRINCIPLES FOR PROGRAMMING

To summarize, some general principles of natural intelligence which may be useful in programming robots:

- Programs should decompose complex actions into independent behaviors, which tightly couple sensing and acting. Behaviors are inherently parallel and distributed.

- In order to simplify control and coordination of behaviors, an agent should rely on straightforward, boolean activation mechanisms (e.g. IRM).

- In order to simplify sensing, perception should filter sensing and consider only what is relevant to the behavior (action-oriented perception).
- Direct perception (affordances) reduces the computational complexity of sensing, and permits actions to occur without memory, inference, or interpretation.
- Behaviors are independent, but the output from one 1) may be combined with another to produce a resultant output, or 2) may serve to inhibit another (competing-cooperating).

Unfortunately, studying natural intelligence does not give a complete picture of how intelligence works. In particular there are several unresolved issues:

UNRESOLVED ISSUES

- *How to resolve conflicts between concurrent behaviors?* Robots will be required to perform concurrent tasks; for example, a rescue robot sent in to evacuate a building will have to navigate hallways while looking for rooms to examine for people, as well as look for signs of a spreading fire. Should the designer specify dominant behaviors? Combine? Let conflicting behaviors cancel and have alternative behavior triggered? Indeed, one of the biggest divisions in robot architectures is how they handle concurrent behaviors.
- *When are explicit knowledge representations and memory necessary?* Direct perception is wonderful in theory, but can a designer be sure that an affordance has not been missed?
- *How to set up and/or learn new sequences of behaviors?* Learning appears to be a fundamental component of generating complex behaviors in advanced animals. However, the ethological and cognitive literature is unsure of the mechanisms for learning.

It is also important to remember that natural intelligence does not map perfectly onto the needs and realities of programming robots. One major advantage that animal intelligence has over robotic intelligence is evolution. Animals evolved in a way that leads to survival of the species. But robots are expensive and only a small number are built at any given time. Therefore, individual robots must "survive," not species. This puts tremendous pressure on robot designers to get a design right the first time. The lack of evolutionary pressures over long periods of time makes robots extremely vulnerable to design errors introduced by a poor understanding of the robot's ecology.

Ch. 5 will provide a case study of a robot which was programmed to follow white lines in a path-following competition by using the affordance of white. It was distracted off course by the white shoes of a judge. Fortunately that design flaw was compensated for when the robot got back on course by reacting to a row of white dandelions in seed.

Robots introduce other challenges not so critical in animals. One of the most problematic attributes of the Reactive Paradigm, Ch. 4, is that roboticists have no real mechanism for completely predicting emergent behaviors. Since a psychologist can't predict with perfect certainty what a human will do under a stressful situation, it seems reasonable that a roboticist using principles of human intelligence wouldn't be able to predict what a robot would do either. However, robotics end-users (military, NASA, nuclear industry) have been reluctant to accept robots without a guarantee of what it will do in critical situations.

3.7 Summary

A behavior is the fundamental element of biological intelligence, and will serve as the fundamental component of intelligence in most robot systems.

BEHAVIOR

A *behavior* is defined as a mapping of sensory inputs to a pattern of motor actions which then are used to achieve a task. Innate Releasing Mechanisms are one model of how intelligence is organized. IRMs model intelligence at Level 2 of a computational theory, describing the process but not the implementation. In IRM, releasers activate a behavior. A releaser may be either an internal state (motivation) and/or an environmental stimulus. Unfortunately, IRMs do not make the interactions between concurrent, or potentially concurrent, behaviors easy to identify or diagram.

Perception in behaviors serves two roles, either as a releaser for a behavior or as the percept which guides the behavior. The same percept can be used both as a releaser and a guide; for example, a fish can respond to a lure and follow it. In addition to the way in which perception is used, there appear to be two pathways for processing perception. The direct perception pathway uses affordances: perceivable potentialities for action inherent in the environment. Affordances are particularly attractive to roboticists because they can be extracted without inference, memory, or intermediate representations. The recognition pathway makes use of memory and global representations to identify and label specific things in the world.

Important principles which can be extracted from natural intelligence are: