

Class 8: Learning I

Selmer Bringsjord

June 10, 1999

Misc Issues:

- Want sample responses from Eliza?
 - recursion reducible to iteration, not magic: TMs
 - take a look at Fibonacci function $\times 3$
- Will go over midterm when “safe”
- Review where we’re going:
 - Learning I today
 - Learning II next
 - Then Communication
 - Then
 - * Robotics
 - * Mathematical Objection
 - * The Future
 - Then Final

The Lottery Paradox (recap)

Suppose you hold one ticket (t_k , for some $k \geq 1$) in a fair lottery consisting of 1 million tickets, and suppose it is known that one and only one ticket will win. Since the probability is only .000001 of t_k 's being drawn, it seems reasonable to believe that t_k will not win. By the same reasoning it seems reasonable to believe that t_1 will not win, that t_2 will not win, ..., that $t_{1000000}$ will not win. Therefore it is reasonable to believe

$$\neg \exists t_i (t_i \text{ will win}).$$

But we know that

$$\exists t_i (t_i \text{ will win}).$$

So we have an outright contradiction.

Lottery Paradox Solution?

Suppose that we are warranted in believing r and that we have equally good prima facie reasons for

$$p_1, \dots, p_n$$

where $\{p_1, \dots, p_n\} \cup \{r\}$ is inconsistent but no proper subset of $\{p_1, \dots, p_n\}$ is inconsistent with r . Then, for every p_i :

$$r \wedge \{p_1 \wedge \dots \wedge p_{i-1} \wedge p_{i+1} \wedge \dots \wedge p_n\} \vdash \neg p_i$$

In this case we have equally strong support for each p_i and each $\neg p_i$, so they collectively defeat one another.

- OSCAR is smart enough to figure this out and suspend judgement.
- How would this be handled by the belief network approach?

coerce and SEEK

- Can you build SEEK?
- The attempt marks a nice concrete baptism into learning
- **coerce** valuable function for doing so:

```
(defun make-fn2 (exp)
  (coerce '(lambda (n) ,exp) 'function))
```

```
? (funcall (make-fn2 '(+ n 1)) 7)
```

```
8
```

- Note: **funcall** applies its first argument's value to its other argument's values. The following two forms are completely equivalent:

```
(funcall #'first '(a b c))
```

```
(first '(a b c))
```

Intro to General Scheme of Learning Functions

- SEEK revisited . . .
- But of course SEEK is primitive — or is it?
- Let's play the 2-4-6 game. . .
- Learning in the limit game: The Guessing Game. . .
- from pairs $(x, f(x))$ return h that approximates f ?
- Can general logical descriptions be understood as functions? How so?

The Guessing Game¹

I have a set A of positive integers (Z^+ denotes the positive integers) in mind; A is $Z^+ - \{n\}$, for some $n \in Z^+$. (I.e., A includes all positive integers save for one.) I will not tell you what my A is, but I'll give unlimited clues. You are to guess what A is after each clue. I will never tell you whether you are right.

¹The following game is borrowed from the elegant text *Systems that Learn*, by Osherson, D.N., Stob, M., and Weinstein, S., published in 1986 by MIT Press, Cambridge, MA.

C1 $1 \in A$.

G1 What would you like to guess? (Perhaps you'll agree that $\{ 2, 3, 5, \dots \}$ would be a bad guess.)

C2 $3 \in A$.

G2 What would you like to guess? How about $\{ 1, 3, 4, 5, \dots \}$ — or does that seem arbitrary to you?

C3 $4 \in A$.

G3 Guess again.

C4 $2 \in A$.

G4 Surprised?

C5 $6 \in A$.

G5 Guess:

C6 $7 \in A$.

G6 Guess:

C7 $8 \in A$.

G7 Guess:

I interrupt the game for some questions:

Q1 How confident are you about your seventh guess? Give an example of an 8th clue that would lead you to repeat that guess, and one that would lead you to change.

Q2 A “guessing rule” is a list of instructions for converting clues received up to a given point into a guess about the set I have in mind. Were your guesses chosen according to some guessing rule? If so, which one?

Q3 What should count as winning the game? How about: *You win just in case one of your guesses is right*— is that any good? (No, it's not any good. Why not?)

Q4 How about the following criterion: *You win just in case you eventually make the right guess and subsequently never change your mind regardless of the new clues you receive.* We will call this **winning in the limit.**

Q5 Suppose that all the clues I give you are of the form: $n \in A$. Suppose furthermore that for every $i \in \mathbb{Z}^+$, we eventually give you a clue of this form if and only if $i \in A$. (So for every number i in our set, you are eventually told that the set contains i ; also you receive no false information about A .) Do not suppose anything about the order in which you will get all these clues: I will order them any way I please. (Recall how surprised you were by the fourth clue.) Now let's say that a guessing rule is "winning" just in case the following is true. If you use the rule to choose your guesses, then no matter which set I have in mind, you are guaranteed to win in the limit. Specify a winning guessing rule for our game.

The Big Myth About Learning

- Myth: Only neural networks learn.
- Actually:
 - NNs = CA_{Life} = k -tape TMs = TMs_{*Turing'sWorld*}
 - e.g., “Is The Connectionist-Logicist Clash a Red Herring?”
- Let your applications make the decision for you, taking account of what we know after comparing and contrasting the two approaches (see R&N, pp. 583-4)

Learning Decision Trees

- What is going on mathematically?
 - recall truth tables
 - every Boolean function f corresponds to a truth-table, e.g.,
 - * \wedge is $f \rightarrow \{0, 1\} \times \{0, 1\}$ defined by truth table
- but we want a more efficient way to represent Boolean functions:
- decision trees, e.g, for conditional it-self

Learning General Logical Descriptions

- Each hypothesis a **candidate definition**:

$$\forall x(Q(x) \Rightarrow C(x))$$

- Each hypothesis H_i has an **extension**
- We assume that one of the hypotheses is correct
- Let $D_i(X_i)$ be the description of the example X_i

- Learning General Logical Descriptions (con.)
 - See **CURRENT-BEST-LEARNING**
 - But what is a “false positive”? A “false negative”?
 - Let’s consider R&N’s example . . .

H_r is

$$\forall x WillWait(r) \Leftrightarrow Patrons(r, Some)$$
$$\vee Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, French)$$
$$\vee Patrons(r, full) \wedge \neg Hungry(r) \wedge Type(r, Thai) \wedge Fri/Sat(r)$$
$$\vee Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Burger)$$

X_{13} is

$$Patrons(X_{13}, Full) \wedge Wait(X_{13}, 0-10) \wedge \neg Hungry(X_{13}) \wedge \dots \wedge WillWait(X_{13})$$

- How is it that the algorithm will prove a contradiction here, as R&N assert on pp. 546-7?

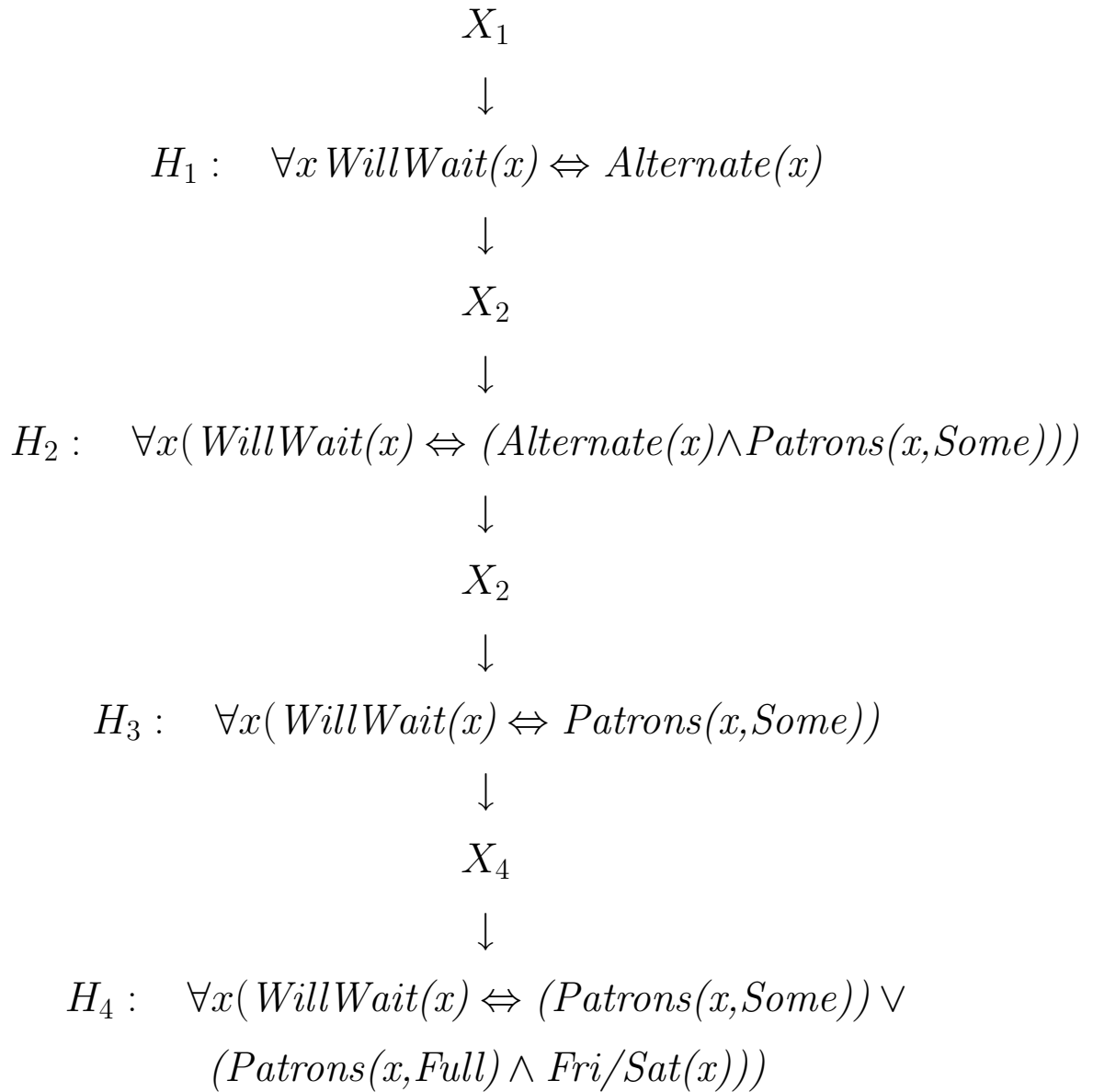
- What is the intuitive picture for generalization and specialization?
- How do they work, formally?
 - If H_1 generalizes H_2 , then

$$\forall x(C_2(x) \Rightarrow C_1(x))$$

so to generalize we can produce new defs implied by their predecessors.

- We can work the other direction to specialize

E.g.



Artificial Neural Nets

- Transfer function of a “neuron” computed in 3-step process:
 1. compute the weighted input

$$in_i = \sum_{j=1}^n w_{ij}x_j$$

2. convert net input to activation level via some activation function g :

$$g(in_i)$$

3. convert activation level to output signal a_i via (commonly)

$$a_i = \begin{cases} g(in_i) & \text{if } g(in_i) > T \\ 0 & \text{otherwise} \end{cases}$$

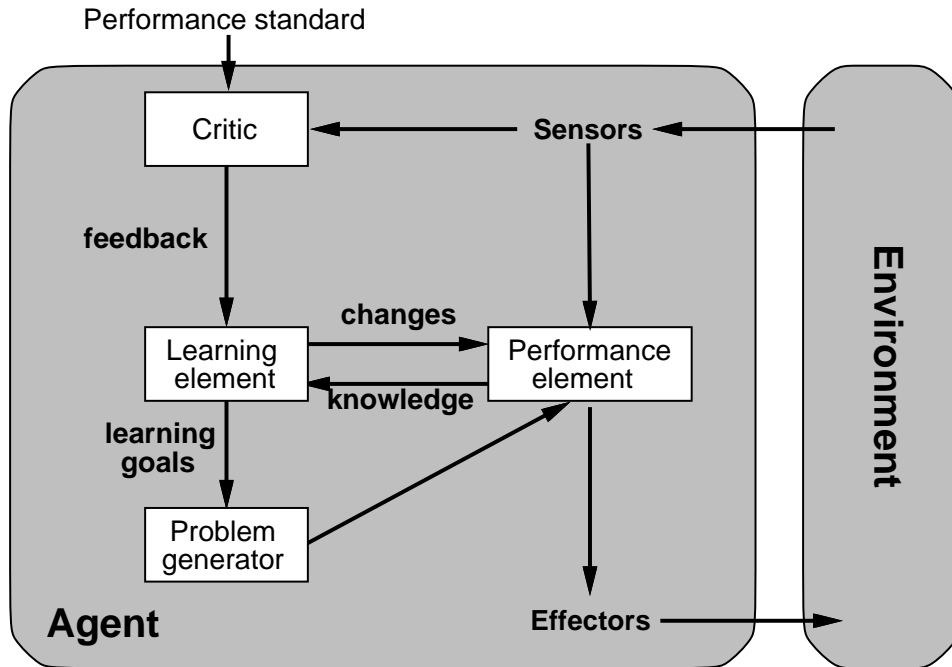


Figure 1:

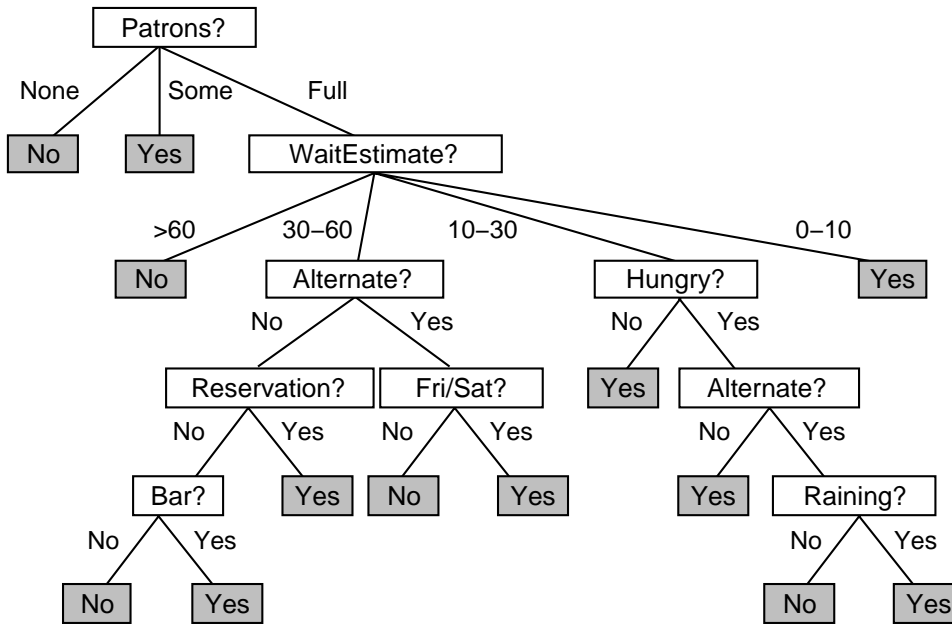


Figure 2:

Example	Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
<i>X</i> ₁	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₂	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>No</i>
<i>X</i> ₃	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₄	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>Yes</i>
<i>X</i> ₅	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>No</i>
<i>X</i> ₆	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₇	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>No</i>
<i>X</i> ₈	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>Yes</i>
<i>X</i> ₉	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>No</i>
<i>X</i> ₁₀	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>No</i>
<i>X</i> ₁₁	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>No</i>
<i>X</i> ₁₂	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>Yes</i>

Figure 3:

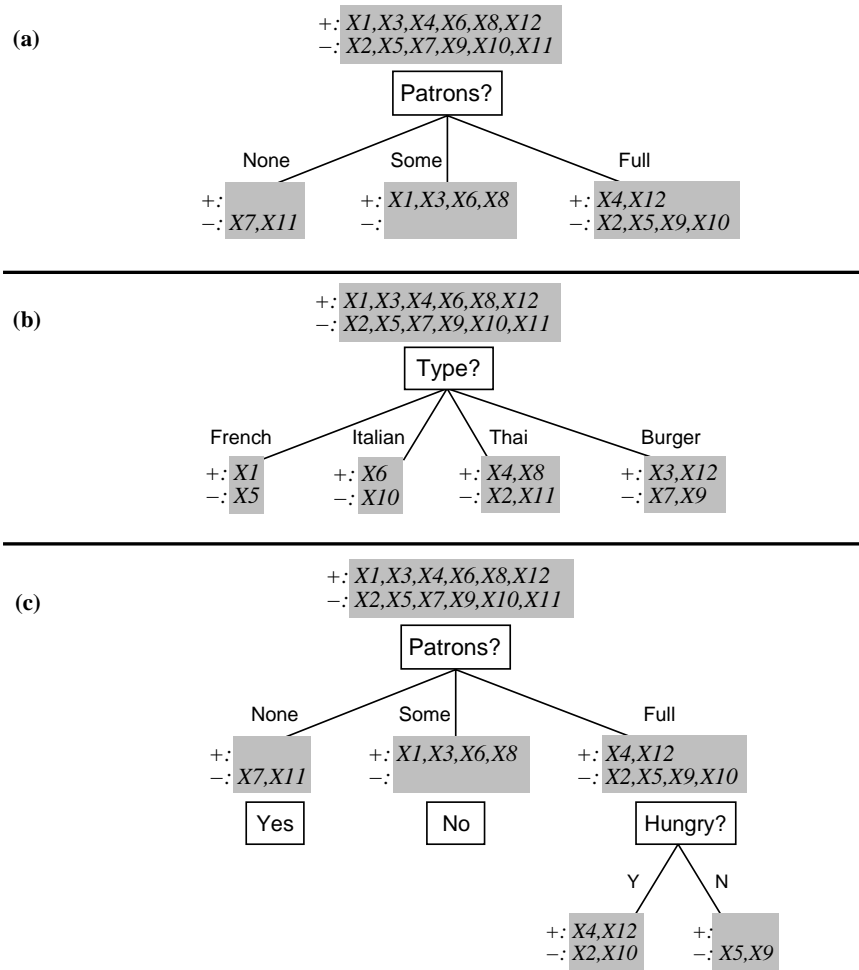


Figure 4:

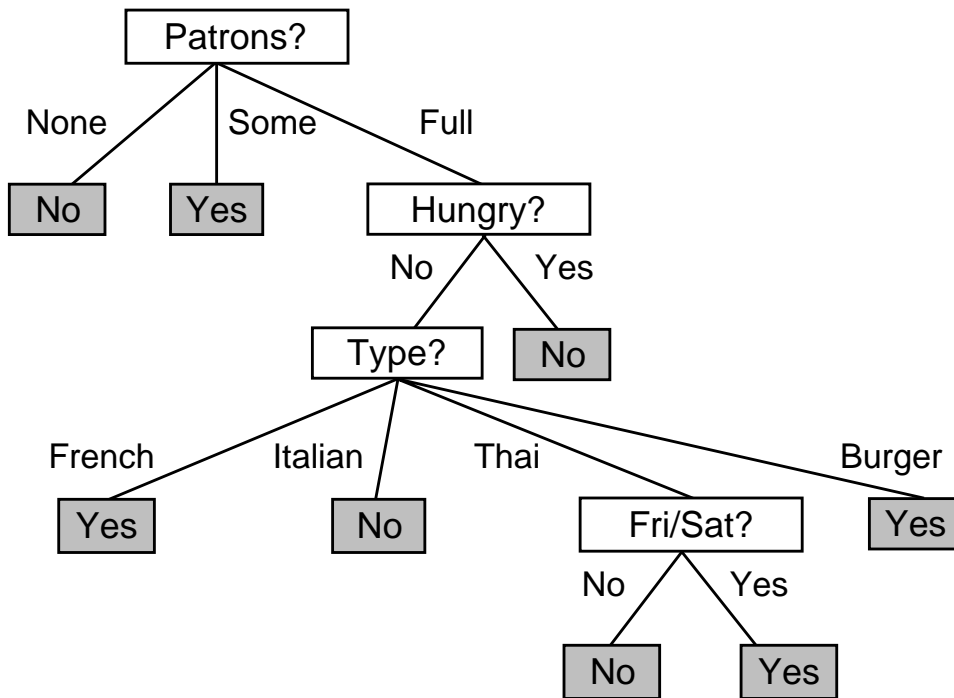


Figure 5:

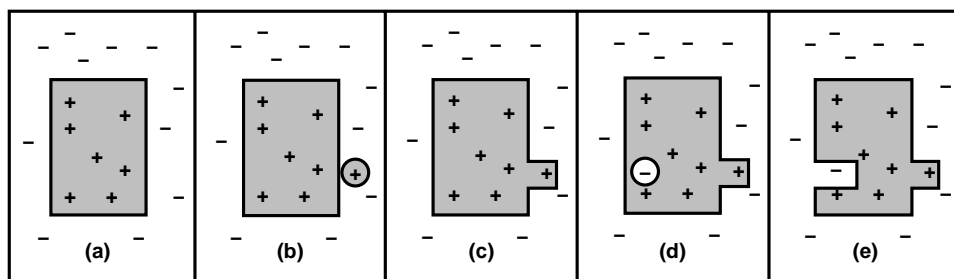


Figure 6:

```

function CURRENT-BEST-LEARNING(examples) returns a hypothesis
   $H \leftarrow$  any hypothesis consistent with the first example in examples
  for each remaining example in examples do
    if  $e$  is false positive for  $H$  then
       $H \leftarrow$  choose a specialization of  $H$  consistent with examples
    else if  $e$  is false negative for  $H$  then
       $H \leftarrow$  choose a generalization of  $H$  consistent with examples
    if no consistent specialization/generalization can be found then fail
  end
  return  $H$ 

```

Figure 7:

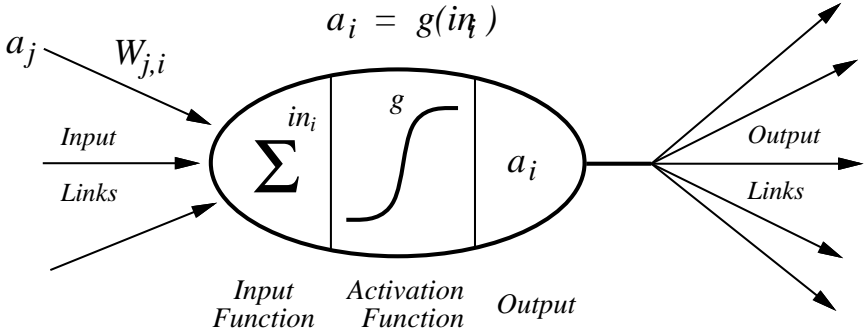


Figure 8: