The three levels at which any machine carrying out an information-processing task must be understood.

1. **Computational theory**
   What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?
The three levels at which any machine carrying out an information-processing task must be understood.

1. **Computational theory**
   What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?

2. **Representation and algorithm**
   How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?
The three levels at which any machine carrying out an information-processing task must be understood.

1. **Computational theory**
   What is the goal of the computation, why is it appropriate, and what is the logic of the strategy by which it can be carried out?

2. **Representation and algorithm**
   How can this computational theory be implemented? In particular, what is the representation for the input and output, and what is the algorithm for the transformation?

3. **Hardware implementation**
   How can the representation and algorithm be realized physically?
Important of Computational Theory (Marr 1982, page 27)

Although algorithms and mechanisms are empirically more accessible, it is the top level, the level of computational theory, which is critically important from an information-processing point of view. The reason for this is that the nature of the computations that underlie perception depends more upon the computational problems that have to be solved than upon the particular hardware in which their solutions are implemented. To phrase the matter another way, an algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied.
Although algorithms and mechanisms are empirically more accessible, it is the top level, the level of computational theory, which is critically important from an information-processing point of view. The reason for this is that the nature of the computations that underlie perception depends more upon the computational problems that have to be solved than upon the particular hardware in which their solutions are implemented. To phrase the matter another way, an algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied.

In a similar vein, trying to understand perception by studying only neurons is like trying to understand bird flight by studying only feathers: It just cannot be done. In order to understand bird flight, we have to understand aerodynamics; only then do the structure of feathers and the different shapes of birds’ wings make sense.
1. **Computational theory**
   Specify an input/output relation
   — e.g., given a string $S$, return its reverse $S^{rev}$. 
1. **Computational theory**
   Specify an input/output relation
   — e.g., given a string $S$, return its reverse $S^{rev}$.

2. **Representation and algorithm**
   Represent a Turing machine as four lists MR, ML, WL, HL.
   Find a Turing machine $[mr, ml, wl, hl]$ meeting the I/O spec
   (w.r.t. a suitable representation).
Abstractions over Turing machines (as hardware)

1. **Computational theory**
   Specify an input/output relation
   — e.g., given a string $S$, return its reverse $S^{rev}$.

2. **Representation and algorithm**
   Represent a Turing machine as four lists MR, ML, WL, HL.
   Find a Turing machine $[mr, ml, wl, hl]$ meeting the I/O spec
   (w.r.t. a suitable representation).

3. **Implementation**
   \( nTm(mr, ml, wl, hl, \text{Input, Output}) \)
   — e.g., so that Output is Input in reverse
goal(Wish,Sitn) :- holds(Wish,Sitn).

holds(initCond1, init).

... holds(initCondi, init).

pos(A,S) :- ... % preconditions

holds(Cond,do(A,S)) :- poss(A,S), addL(A,Cond).

holds(Cond,do(A,S)) :- poss(A,S), holds(A,Cond), not delL(A,Cond). % inertia

?- goal(peace-on-earth,Sitn).

holds(peace-on-earth,do(A,S)) [Sitn=do(A,S)]
Finding Sitn (from init vs Wish)

goal(Wish,Sitn) :- sitn(Sitn), holds(Wish,Sitn).
sitn(init). % start from init
sitn(do(A,S)) :- sitn(S), poss(A,S).

Contra goal-directed Prolog search (from Wish)

i :- p, q.

i :- r.

p.

r.

-----------

| ?- i.
Finding Sitn (from init vs Wish)

goal(Wish,Sitn) :- sitn(Sitn), holds(Wish,Sitn).

sitn(init).  % start from init
sitn(do(A,S)) :- sitn(S), poss(A,S).

Contra goal-directed Prolog search (from Wish)

i :- p,q.                      [i]

i :- r.

p.

r.

| ?- i.                         StartNode = [i]
Finding Sitn (from init vs Wish)

goal(Wish,Sitn) :- sitn(Sitn), holds(Wish,Sitn).
sitn(init).  % start from init
sitn(do(A,S)) :- sitn(S), poss(A,S).

Contra goal-directed Prolog search (from Wish)

\[
\begin{align*}
  i & :\ p, q. \\
  i & :\ = r. \\
  p. \\
  r. \\
\end{align*}
\]

\[
\text{StartNode} = [i]
\]
Finding $\text{Sitn}$ (from init vs Wish)

goal(Wish, Sitn) :- sitn(Sitn), holds(Wish, Sitn).
sitn(init). % start from init
sitn(do(A, S)) :- sitn(S), poss(A, S).

Contra goal-directed Prolog search (from Wish)

\[
\begin{align*}
\text{i} & :- \text{p}, \text{q}. & \text{[i]} \\
\text{i} & :- \text{r}. & \quad \text{[p,q]} & \quad \text{[r]} \\
\text{p.} & & \quad \text{[q]} \\
\text{r.} & \\
\text{__________} \\
\text{| ?- i.} & \quad \text{StartNode} = \text{[i]}
\end{align*}
\]
Finding Sitn (from init vs Wish)

goal(Wish,Sitn) :- sitn(Sitn), holds(Wish,Sitn).
sitn(init). % start from init
sitn(do(A,S)) :- sitn(S), poss(A,S).

Contra goal-directed Prolog search (from Wish)

\[
\begin{align*}
  i & :- p, q. & \text{[i]} \\
  i & :- r. & \text{[p,q]} & \text{[r]} \\
  p. & \text{[q]} & \text{[]} \\
  r. \\
\end{align*}
\]

------------------
\[
| ?- i. \quad \text{StartNode} = [i]
\]
Finding Sitn (from init vs Wish)

\[
\text{goal}(\text{Wish}, \text{Sitn}) :- \text{sitn}(\text{Sitn}), \text{holds}(\text{Wish}, \text{Sitn}).
\]

\[
\text{sitn}(\text{init}). \quad \% \text{start from init}
\]

\[
\text{sitn}(\text{do}(A, S)) :- \text{sitn}(S), \text{poss}(A, S).
\]

Contra goal-directed Prolog search (from Wish)

\[
\begin{align*}
& \text{i} :- \text{p}, \text{q}. & \text{[i]} \\
& \text{i} :- \text{r}. & \text{[p,q]} \quad \text{[r]} \\
& \text{p}. & \text{[q]} \quad \text{[]} \\
& \text{r}. \quad \\
\end{align*}
\]

\[
\begin{align*}
\text{StartNode} &= \text{[i]} \\
\text{GoalNode} &= \text{[]}
\end{align*}
\]
Plans from situations

\[ \text{init} \rightsquigarrow \text{nothing to do} \]
\[ \text{do}(A,S) \rightsquigarrow \text{do A after doing what’s in } S \]
Plans from situations

\[ \text{init} \sim \text{nothing to do} \]
\[ \text{do}(A,S) \sim \text{do } A \text{ after doing what’s in } S \]

\[ \text{e.g., } \text{do}(a, \text{do}(b, (\text{do}(c, \text{init})))) \approx \text{do}(c; b; a, \text{init}) \]

string
Plans from situations

\[ \text{init} \rightsquigarrow \text{nothing to do} \]
\[ \text{do}(A,S) \rightsquigarrow \text{do } A \text{ after doing what's in } S \]

\[ \text{e.g., do}(a, \text{do}(b, (\text{do}(c, \text{init})))) \approx \text{do}(c; b; a, \text{init}) \]

string

Time - given by string of actions from init

Assume: actions are deterministic and fully known

i.e., all that happens between S and do(A,S) is A
Plans from situations

\[ \text{init} \sim \rightarrow \text{nothing to do} \]
\[ \text{do}(A,S) \sim \rightarrow \text{do } A \text{ after doing what's in } S \]

\[
\text{e.g., } \text{do}(a,\text{do}(b,\text{do}(c,\text{init})))) \approx \text{do}(c;b;a,\text{init})
\]

agement

Time - given by string of actions from init

Assume: actions are deterministic and fully known
i.e., all that happens between S and do(A,S) is A

Determinize action by incorporating effects into the action:

\[ \text{WRITE} \sim \rightarrow \text{WRITE}(\text{symbol-written},\text{next-state}) \]

Represent time without committing to full knowledge of actions
(many agents + partial knowledge)

Study strings of actions and derived/primitive relations

\[ \text{peace-on-earth} :- \text{peace(nIre)},\text{peace(mideast)},... \]
Logic of goal: a declarative turn (Marr’s Comp Lev)

Recall from Marr 1982

*an algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied.*
Logic of goal: a declarative turn (Marr’s Comp Lev)

Recall from Marr 1982

an algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied.

John R. Anderson’s *Rational analysis* in 6 steps (Wikipedia)

1. Goals: Specify precisely the goals of the cognitive system.
2. Environment: Develop a formal model of the environment to which the system is adapted.
3. Computational Limitations: Make the minimal assumptions about computational limitations.
4. Optimization: Derive the optimal behavioral function given 1-3 above.
5. Data: Examine the empirical literature to see whether the predictions of the behavioral function are confirmed.
6. Iteration: Repeat, iteratively refining the theory