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?
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.
Abstractions over Turing machines (as hardware)

1. **Computational theory**
   Specify an input/output relation
   — e.g., given a string $S$, return its reverse $S^{rev}$
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 $MR$, $ML$, $WL$, $HL$
   and find a Turing machine $[mr, ml, wl, hl]$ satisfying 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 $MR, ML, WL, HL$
   and find a Turing machine $[mr, ml, wl, hl]$ satisfying the
   I/O spec (w.r.t. a suitable representation)

3. **Implementation**
   Run the Turing machine $[mr, ml, wl, hl]$
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 $MR$, $ML$, $WL$, $HL$
   and find a Turing machine $[mr, ml, wl, hl]$ satisfying the
   I/O spec (w.r.t. a suitable representation)

3. **Implementation**
   Run the Turing machine $[mr, ml, wl, hl]$

There are many programming languages that we can use
in place of Turing machines — choices can be arbitrary
(differences inessential)