Chapter 6

Computation

6.1 Introduction

In the last two chapters we saw that both the logical and the cognitive models of scientific discovery include a condition to prefer simple or minimal explanations. None of the models further suggest why it is rational to prefer simplicity. I argued how the ACT-R model of cognition implicitly prefers simplicity as a consequence of a mechanism that prefers high probability in section 5.6 (page 67).

In this chapter I investigate the relation between probability and simplicity in the computational description, explanation and prediction of empirical data. I discuss the use of Kolmogorov complexity and Bayes' theorem in Solomonoff's inductive method to explicate a general concept of simplicity that is used for a distribution of probabilities of possible hypotheses. This makes it possible to understand how the search for simple, *i.e.*, short, descriptions of empirical data leads to the discovery of patterns in the data, and hence more probable predictions. I show how the simplicity bias of Langley's BACON.2 and Thagard's PI is subsumed by Rissanen's Minimum Description Length principle, which is a computable approximation of Solomonoff's uncomputable inductive method. A more lengthy discussion, including several other approaches to simplicity, can be found in (van den Bosch 1994).

In this chapter I pursue an answer to two particular questions: 1) How can simplicity most generally be defined? 2) Why should we prefer a simpler theory to a more complex one? I discuss simplicity definitions that stem from research in cognitive science and machine learning. In those approaches simplicity plays an important role in the process of scientific discovery, as implemented in Langley and Simon's computer model BACON, in inference to the best explanation, as implemented in Thagard's computer model PI, and in the probability of predictions, as explicated by Solomonoff.

Langley and Simon claim that the BACON programs search for simple consistent laws without making explicit what is meant by 'simple' laws and why we should pursue simplicity (Langley et al 1987). Thagard proposed an explicit definition of simplicity and employs it in his model PI, without providing a satisfying reason for it (Thagard 1988). However, Solomonoff proposed an explication of induction which makes use of a concept that can be used to understand simplicity and to provide a satisfying justification for its preference.

According to Solomonoff we should trust the theory yielding implications that can be generated by the shortest computer program that can generate a description of our known observational data. It is argued that a shorter computer program provides more probable predictions because it uses more patterns from that data. It is proved that this simplicity measure is reasonably independent of the computer language that is used. However, this measure has one drawback, it is uncomputable. Yet it is claimed that computable approaches to induction in machine learning constitute approximations of Solomonoff's method (Li and Vitányi 1994).

In this chapter I demonstrate how Solomonoff's approach can elegantly be used to make a universal prior probability distribution for Bayes' theorem. First it is shown that Rissanen's Minimum Description Length principle (MDL) can be derived from Solomonoff's approach. And from thereon I show that simplicity in Langley's BACON.2, and simplicity in Thagard's PI are nicely subsumed by MDL. I conclude this chapter by answering the three specific questions of this thesis, according to the study of computational description.

6.2 Turing machines

In 1964 an article by Solomonoff was published that contained a proposal for a general theory of induction. The objective was the extrapolation of a long sequence of symbols by finding the probability that a sequence is followed by one of a number of given symbols. It was Solomonoff's conviction that all forms of induction could be expressed in this form.

He argued that any method of extrapolation can only work if the sequence is very long and that all the information for an extrapolation is in the sequence. Solomonoff proposed a general solution that involved Bayes' theorem. This theorem requires that a prior probability of a hypothesis is known to determine the posterior probability making use of the known data. Solomonoff's solution is to provide for a universal distribution of prior probabilities, making use of a formal definition of computation.

It is widely believed that the notion of computation is fundamentally captured by the operation of a Turing machine, an idealized conceptual machine introduced by the mathematician Alan Turing. A Turing machine is thought of as consisting of a long tape and a finite automaton which controls a 'head' that can read, delete and print symbols on the tape. To compute the value of a function $y = f(x)$, write a program for $f(x)$ on the tape, together with a symbolic representation of x and start the Turing machine. The program is completed when the Turing-machine halts and the value of y is left on the tape as output. Turing proved that there is a universal Turing-machine that can compute every function that can be computed by any Turing machine. The famous Church-Turing thesis claims that every function that can be computed, can be computed by a Turing machine.

What Solomonoff did was to correlate all possible sequences of symbols with programs for a universal Turing machine that has a program as input and the sequence as output. He assigned a high prior probability to a sequence that can be computed with short and/or numerous programs. Sequences that need long programs and can only be compared by few programs, receive a low prior probability. For Solomonoff the validity of giving sequences calculated by a shorter program a higher prior probability

is suggested by a conceptual interpretation of Ockham's razor. But it is justified because a shorter program utilizes more patterns in the sequence to make the program shorter. So, if we trust the data as being representative of things to come, then the shorter program provides the more probable predictions.

If we, *e.g.*, have a sequence that has x as a prefix and x = 1234123412341234123, then we could write a program that describes x as dαααα123, where d is a definition of 1234 as α. If we want to predict the following letter we can entertain the hypothesis H_1: dαααααα, or still shorter H_1: d5α. Another option is H_2: d4α1231. The first hypothesis predicts a 4 and the second a 1. Both hypotheses are compatible with the known data x. Now Solomonoff argues that the prediction of H_1 is more probable because it requires a shorter program to generate a continuation of x than H_2 (Solomonoff 1964, p.10).

That a sequence with many programs gets a high prior probability is suggested by the idea that if an occurrence has many possible causes, then it is more likely. The principle of indifference is integrated by attributing sequences that are generated by programs of the same length the same prior probability.

Unfortunately this approach has as an important problem. It is not determinable whether a given program is the shortest program that computes a sequence. If that were determinable then there would exist a Turing machine that could determine for every possible program whether it would generate a given sequence. However, most of the possible Turing machine programs will never halt. Due to this halting problem we cannot know for every program whether the program computes a given sequence. But before I go into that problem I want to make Solomonoff's theory more specific by first discussing Kolmogorov complexity and its application in probability theory.

6.3 Kolmogorov complexity

The Kolmogorov complexity of a sequence or string is actually a measure of randomness or, when inverted, the regularity of the patterns in that string. We can use a Turing machine to measure that regularity with the length of the shortest program for that Turing machine that has the sequence as output. We can call such a program a description of the sequence. This description is relative to the Turing machine that has the description as input.

So when we have a sequence x and a description program p and a Turing machine T we can define the descriptional complexity of x, relative to T as follows (*cf.* Li and P.M.B. Vitányi 1993, pp.352):

Definition 1 *Descriptional complexity*. The descriptional complexity C_T of x, relative to Turing machine T is defined by:

$$C_T(x) = \min\{ l(p): p \in \{0,1\}^*, T(p) = x \}$$

or $C_T(x) = \infty$ if no such p exists.

We consider T to be a Turing machine that takes as input program a binary string of zeros and ones, so the program is an element of the set {0,1}*, which is the set of all

finite binary strings. We use binary strings because everything that can be decoded, like *e.g.*, scientific data, can be coded by a string of zeros and ones. The length of the program, l(p), is the number of zeros and ones. So the definition takes as the complexity of a string x the length of the program p that consists of the least number of bits and that will generate x when given to T. If no such program exists then the complexity is considered to be infinite.

When x is a finite string then there is always a program that will describe it. Just take a program that will merely print the number literally. This program will be larger than the string. However, if x is infinite and no finite program exists, then x is uncomputable by definition.

This complexity measure is relative to but surprisingly largely independent of the Turing machine in question, as long as it is a universal Turing machine. There exists a universal Turing machine that computes the same or a lower complexity than the complexity computed by any other Turing machine plus some constant dependent on that other Turing machine. For instance, when I have a string and two programs in different computer languages that compute that string, the difference in length between those programs cannot be more than a constant, independent of the string. This claim is called the invariance theorem (*cf.* Li and Vitányi 1993, pp.353).

In the literature Kolmogorov complexity K(x) is defined as a variant of descriptional complexity C(x), which makes use of a slightly different kind of Turing machines. In the definition of descriptional complexity a Turing machine was used with one infinite tape that can move in two directions and that starts with an input program on it and halts with a string on the tape as output. For the definition of Kolmogorov complexity a prefix machine is used. This kind of Turing machine uses three tapes, an input tape and an output tape which both move in only one direction, and a working tape that moves in two directions. The prefix Turing machine reads program p from the input tape, and writes string x on the output tape.

Kolmogorov complexity will render a similar measure of complexity as descriptional complexity, where C(x) and K(x) differ by at most 2 log K(x). This difference is important, because of its use in Bayes' formula. (Without it the sum of the probabilities of all possible hypotheses will not converge to one.) The invariance theorem for K(x) is similar to that of C(x). Now how can this measure be useful in extrapolating a sequence? First we will take a brief look at how Bayes' formula requires a prior probability.

6.4 Bayesian inference

We will start with a hypothesis space that consists of a countable set of hypotheses which are mutually exclusive, *i.e.*, only one can be right, and exhaustive, *i.e.*, at least one is right. Each hypothesis must have an associated prior probability $P(H_n)$ such that the sum of the probabilities of all hypotheses is one. If we want the probability of a hypothesis H_n given some known data D then we can compute that probability with Bayes' formula:

$$P(H_n \mid D) = P(D \mid H_n)\, P(H_n) / P(D)$$

where $P(D) = \sum_n P(D \mid H_n)P(H_n)$. This formula determines the *a posteriori* probability $P(H_n \mid D)$ of a hypothesis given the data, *i.e.*, the probability of H_n modified from the prior probability $P(H_n)$ after seeing the data D. The conditional probability $P(D \mid H_n)$ of seeing D when H_n is true is forced by H_n, *i.e.*, $P(D \mid H_n) = 1$ if H_n can generate D, and $P(D \mid H_n) = 0$ if H_n is inconsistent with D. So when we consider only hypotheses that are consistent with the data the prior probability becomes crucial. Because for all H_n where $P(D \mid H_n) = 1$ the posterior probability of H_n will become:

$$P(H_n \mid D) = P(H_n) / P(D)$$

Now let us see what happens when we apply Bayes' formula to an example of Solomonoff's inductive inference. In this example we only consider a discrete sample space, *i.e.*, the set of all finite binary sequences $\{0,1\}^*$.

What we want to do is, given a finite prefix of a sequence, assign probabilities to possible continuation of that sequence. What we do is, given the known data, make a probability distribution of all hypotheses that are consistent with the data. So if we have a sequence x of bits, we want to know what is the probability that x is continued by y. So in Bayes' formula:

$$P(xy \mid x) = P(x \mid xy)P(xy) / P(x)$$

We can take $P(x \mid xy) = 1$ no matter what we take for y, so we can say that:

$$P(xy \mid x) = P(xy) / P(x)$$

Hence if we want to determine the probability that sequence x is continued by y we only need the prior probability distribution for P(xy). Solomonoff's approach is ingenious because he first identifies x with the computer programs that can generate a continuation of x by a string y. In this way the a priori most probable continuation y can be determined in two ways: y is the next element that is predicted, *i.e.*, generated, by the smallest Turing machine program that can generate x; or the string y is predicted that is generated by most of the programs that can generate x.

So we can define the prior probability of a hypothesis in two different ways. We can give the shortest program the highest prior probability and define the probability of xy as:

$$P_{K(xy)} := 2^{-K(xy)}$$

i.e., the length of the shortest program that computes xy as the negative power of two (Li and Vitányi 1990, pp.216). Or we can define $P_{U(xy)}$ as the sum of $2^{-l(p)}$ for every program p (so not only the shortest) that generates xy on a reference universal prefix machine (Li and Vitányi 1993, pp.356). The latter is known as the Solomonoff-Levin distribution. Both have the quality that the sum of prior probabilities is equal to or less than one, *i.e.*,

$$\sum_x P_{K(x)} \leq 1 \text{ and } \sum_x P_{U(x)} < 1$$

However, it can be shown that if there are many 'long' programs that generate x and predict the same y, then a smaller program must exist that does the same. And it is proved that both prior probability measures coincide up to an independent fixed multiplicative constant (Li and Vitányi 1993, pp.357).

So we can take the Kolmogorov complexity of a sequence as the widest possible notion of shortness of description of that sequence. And if we interpret shortness of description, defined by Kolmogorov complexity, as a measure for parsimony, then the Solomonoff-Levin distribution presents a formal representation of the conceptual variant of Ockham's razor, since the predictions of a simple, *i.e.*, short, description of a phenomenon are more probable than the predictions of a more complex, *i.e.*, longer, description.

6.5 Description length

While both the Kolmogorov and Solomonoff-Levin measure are not computable, there are computable approximations of them. It is demonstrated that several independently developed inductive methods actually yield computable approximations of Solomonoff's method. I will first demonstrate this for Rissanen's minimum description principle (MDL), *cf.* Li and Vitányi (1993).

Rissanen made an effort to develop an inductive method that could be used in practice. Inspired by the ideas of Solomonoff he eventually proposed the minimum description length principle. This principle states that the best theory given some data is the one which minimizes the sum of the length of the binary encoded theory plus the length of the data, encoded with the help of the theory. The space of possible theories does not have to consist of all possible Turing machine programs, but can just as well be restricted to polynomials, finite automata, Boolean formulas, or any other practical class of computable functions.

To derive Rissanen's principle I first need to introduce a definition of the complexity of a string given some extra information, which is known as *conditional* Kolmogorov complexity:

Definition 2 *Conditional Kolmogorov complexity.* The conditional Kolmogorov complexity K_T of x, relative to some universal prefix Turing machine T(p, y) with program p and additional information y is defined by:

$$K_T(x \mid y) = \min\{ l(p) : p \in \{0,1\}^*, T(p, y) = x\}$$

Or $K_T(x \mid y) = \infty$ if such p does not exist.

This definition subsumes the definition of (unconditional) Kolmogorov complexity when we take y to be empty. Now, Rissanen's principle can elegantly be derived from Solomonoff's method. We start with Bayes' theorem:

$$P(H \mid D) = P(D \mid H)\, P(H) / P(D)$$

The hypothesis H can be any computable description of some given data D. Our goal is to find an H that will maximize P(H | D). Now first we take the negative logarithm of all probabilities in Bayes equation. The negative logarithm is taken because probabilities are smaller or equal to one and we want to ensure positive quantities. This results in:

$$-\log P(H \mid D) = -\log P(D \mid H) - \log P(H) + \log P(D)$$

When we consider P(D) to be a constant then maximizing P(H | D) is equivalent to *minimizing* its negative logarithm. Therefore we should minimize:

$$-\log P(D \mid H) - \log P(H)$$

This will result in the Minimum Description Length principle if we consider that the probability of H is approximated by the probability for the shortest program for H, *i.e.*,

$$P(H) = 2^{-K(H)}$$

Therefore the negative logarithm of the probability of H is exactly matched by the length of the shortest program for H, *i.e.*, the Kolmogorov complexity K(H). The same goes for P(D |H) and hence we should minimize:

$$K(D \mid H) + K(H)$$

This amounts to MDL principle, *i.e.*, minimizing the description or program length of the data, given the hypothesis, plus the description length of the hypothesis (Li and Vitányi 1990, pp.218). To make this principle practical all that remains is formulating a space of computable hypotheses that together have a prior probability smaller or equal to one, and searching this space effectively. It has been shown in several applications that this approach is an effective way of learning (Li & Vitányi 1993, p.371).

6.6 Cognitive models

Let us look at the simplicity bias of BACON.2 (BACON.2 is not representative for the other BACON programs. I discuss the simplicity bias of the other BACON's in (van den Bosch, 1994). BACON.2 will always construct the simplest consistent law in its range of search. The method it uses is called the differencing method. With this method BACON.2 is able to find polynomial and exponential (polynomial) laws that summarize given numeral data (Langley et al. 1987). One could define the simplicity bias of BACON.2 as follows:

Definition 3 *Simplicity bias in BACON.2* The simplicity of a polynomial decreases with the increase of the polynomial's highest power. A variable power is gathered to be a simpler polynomial than a polynomial with a high constant degree.

Langley et al. give no epistemical reason for preferring simplicity. However, after discussing simplicity in Thagard's PI I will argue that the simplicity bias of BACON.2, as defined above, is justified.

Thagard's account of the simplicity of a hypothesis does not depend on the simplicity of the hypothesis itself, but on the number of auxiliary hypotheses that the hypothesis needs to explain a given number of facts (Thagard, 1988). In PI, Thagard's cognitive model of scientific problem solving, discovery and evaluation are closely related. Simplicity plays an important role in both.

PI defines two hypotheses to be co-hypotheses if they were formed together for an explanation by abduction. From the number of co-hypotheses and the number of facts explained by a hypothesis its simplicity is calculated according to the following definition:

Definition 4 *Simplicity in PI.* The simplicity of hypothesis H, with the number of c co-hypotheses and with the number of f facts explained by H, is calculated in PI as

$$\text{simplicity(H)} = (f - c)/f \text{ , or zero if } f \leq c.$$

To determine the best explanation PI considers both consilience (*i.e.*, explanatory success, or in PI; number of facts explained) and simplicity. This is no difficult decision if in one of the dimensions the value of one of the explanations is superior to that of the others while the values in the other dimension are more or less equal. If both explanations explain the same number of facts but one is simpler than the other, or if they are both equally simple, but one explains more facts than the other, then there is no difficult choice. But when *e.g.*, the first theory explains most facts while the second is the simplest, that conflict seems to make the choice more difficult. In that case PI computes a value for both hypotheses according to the following definition:

Definition 5 *Explanatory value in PI*. The explanatory value of hypothesis H for IBE is calculated in PI as

$$\text{value(H)} = \text{simplicity(H)} \times \text{consilience(H)}.$$

In this way PI can pick out explanations that do not explain as much as their competitors but have a higher simplicity or explain more important facts. It also renders *ad hoc* hypotheses useless because if we add an extra hypothesis for every explanation then the simplicity of that theory will decrease at the same rate as its consilience increases.

One feature of IBE in PI is that its valuation formula admits of a much simpler definition which easily follows from the definitions of simplicity and the value of a hypothesis as given above.

Theorem 1 For IBE the explanatory value of a hypothesis H, with the number of c co-hypotheses and f facts explained, can be calculated in PI as

$$\text{value(H)} = f - c, \text{ or zero if } f \leq c.$$

Thagard does not satisfactorily argue why we should prefer this kind of simple hypotheses. In its defense he only demonstrates that several famous scientists used it. But he did not show that simplicity promotes the goals of inference to the best explanation, like truth, explanation and prediction.

6.7 Computable approximation

I will now compare Rissanen's minimum description length principle (MDL) with BACON.2, and with inference to the best explanation (IBE) as implemented by Thagard in PI. For BACON.2 Rissanen's principle suggests an improvement because in the case of noisy data, BACON.2 would probably come up with a polynomial as long as that data, while it could construct a much simpler one when it employs and encodes deviations from the polynomial as well.

An important difference between Rissanen's principle and BACON is that the former requires to search the whole problem space, while the latter searches it heuristically. But BACON's search is guided by the same patterns that eventually will be described by a law. However, a heuristic search, like BACON's, can be aided by Rissanen's principle. Actually BACON does search for a minimal description, but it does not try to minimize it, *i.e.*, if BACON finds a description, it halts, and will not search for a shorter one.

BACON.2 determines the shortest polynomial that can describe a given sequence. No Turing machine can be constructed that needs a shorter description for a more complex polynomial. It can be demonstrated that a polynomial formula with an exponential term has a shorter description than a polynomial formula without an exponential term that describes the same sequence. BACON.2's method always finds the simplest polynomial that exactly fits the data. So I will make the following claim:

Claim 1 The polynomial constructed by BACON.2 with the differencing method, based on a given sequence x is the polynomial with the shortest description that exactly describes x, if x can be described at all with a polynomial.

The validity of this claim can be derived from the differencing method. Every preference of BACON.2 between two polynomials that are compatible with the data is in agreement with the minimum description length principle. However, MDL can seriously improve BACON.2 by including a valuation of a description of the sequence, given a possible polynomial. A shorter description of the sequence may result when deviations from a possible polynomial are encoded as well.

In Thagard's explication of inference to the best explanation in PI, the simplicity of a hypothesis is determined by the number of additional assumptions or co-hypotheses that the hypothesis needs for its explanations. Rissanen's MDL accounts for the importance of auxiliary hypotheses as well. MDL requires that we minimize the sum of the description of an explaining hypotheses K(H) and the description of the data with the aid of the hypothesis K(D | H).

If an hypothesis can explain something right away the description of the data is minimal, while if the hypothesis requires additional assumptions, then the description of the data will be longer. So, Thagard's simplicity satisfies at least one of the requirements of MDL. Hence I want to make the following claim and argue for its plausibility.

Claim 1 In a case of equal consilience, the explanation that will be selected by IBE in PI will provide a shorter description of the facts, given the explanation, or at least no longer description with respect to the available alternatives.

This claim follows easily from the theorem stating that PI's IBE values hypotheses by subtracting the number of co-hypotheses c from the number of f explained facts, *i.e.*, f minus c. Two hypotheses that are of equal consilience explain the same number of facts, in which case the hypothesis with the least number of co-hypotheses is preferred. Hence, assuming that every such extra assumption is of about equal length, the simpler hypothesis will provide a shorter description of the facts given the hypothesis. However, if both have the same number of co-hypotheses, then PI can not make a choice, because both will provide a description of reasonably similar length.

6.8 Best hypothesis

One question may now come to mind: will the Solomonoff approach yield a unique preference when several simple hypotheses are compatible with the data? It seems possible that more than one theory or program, consistent with given data, can be of the same length. So in that case we cannot make a decision based on a simplicity consideration, because all alternatives are of equal simplicity.

To answer that criticism we first have to distinguish between the next symbol y that is predicted given a sequence x and the different programs p that can generate a prediction. Our goal can be a correct prediction y, given x, or a correct explanation of x. In the case that we want a correct prediction, if two programs are of the same length it may turn out that both predict something else. However, Solomonoff's method supplies two ways to solve this dilemma.

The first is the universal Solomonoff-Levin distribution with which probabilities can be assigned to different continuations of a sequence. A given prediction y not only receives a higher probability if it is predicted by a short program, but also if numerous programs make the same prediction. So if there is more than one shortest program, the prediction of the program that predicts the same as numerous other longer programs is preferred.

The second way out of the dilemma is in the situation when the given amount of data x is very long. It can be proved that in the limit all reasonable short programs will converge to the same prediction, so you can pick any of them. This feature of the Solomonoff approach is nice for practical and computable approximations. Because you can make reasonably good predictions with a given short program that may not be the shortest one.

However, if you value the best *explanation* of a given amount of observations, then you will not be satisfied by a grab bag of possible hypotheses that may not even

make the same predictions. Scientists that want to understand the world usually look for *one* best explanation. In this situation a case could be made for the simplest hypothesis as the best explanation. But with such an aim the Solomonoff approach seems troublesome. Because you can never know whether a given short program that computes x is also the shortest program possible. Because the only effective way to do so is to test all possible Turing machines one by one to see if they generate x. But any of those possible Turing machines may never halt and there is no way to find out whether it ever will. You may put a limit to the time you allow the machine to run before you test the following one. But a shorter program can always be just one second further away.

The philosopher Ernst Mach once made the claim that the best thing that science could do is to make predictions about phenomena, without explaining the success of such predictions by the (ontological) assumptions of the possible hypotheses. However, the best explanation, and hence possibly the simplest program, can be seen as the ultimate goal of science. And a nice property of the present kind of simplicity is that we can measure our progress. We may not have an effective, *i.e.*, computable, method to establish whether a hypothesis is *the* simplest but given a large amount of data we *can* establish the relative simplicity of any two hypotheses that yield the data.

6.9 Conclusion

I will try to state my conclusion in one sentence, but nevertheless it is probably not the shortest description of that conclusion: in scientific discovery it is rational to prefer those hypotheses, that, given discovered alternative hypotheses, amount to the shortest computational description of known data, because they provide more probable predictions. This approach to learning and discovery generalizes the rational pretension of the logical and cognitive models of discovery that prefer minimal or simple explanations. So to answer the specific questions of this thesis, we have:

Question 1 What is the structure of a scientific theory? In the computational approach a theory consists of a universal Turing machine, together with a program for that machine. The data that is explained by the theory is the result of a description of that data that can be generated by a particular program that can make predictions. So, a string P describing data and predictions is the output of a computation of a computer TM and program H, *i.e.* TM(H) = P. Both the logical and the cognitive models can be subsumed within this general scheme.

Question 2 What is the process of scientific reasoning? The process of reasoning in machine learning is summarized in Table 6.1. A string P describes given data of a phenomenon p, given a way of representation. The task of explanation is to find a short program H* that can generate that string, given a computer T. This task is uncomputable, in the sense that there is no algorithm that can guarantee to find that program. Yet it can be approximated heuristically. The shortest program has the highest a priori probability. Given a Turing machine TM, a program H and earlier data P a prediction P* and posterior probability can be computed.

Problem	Start	Background	Process	Goal	Goal properties
Explanation	String P	TM	Approximation	H*	TM(H*) = P Prior probability
Prediction	Program H	TM, String P	Computation	P*	TM(H, P) = P* Posterior probability

Table 6.1: Short overview of inferences discussed in this chapter

Question 3 What is the route between theory and experiment? The theoretical route between theory and experiment can in this approach also be summarized in six steps. (For comparison, I added the logical version of this process between brackets)

1. Observe phenomenon p: $p_m,\ldots, p_n$
2. Describe p: $(P_m \rightarrow P_n)$
   $P_{m, n}$: = string
3. Explain p: $(H^* \models P_m \rightarrow P_n)$
   $TM(H^*) = P_{m, n}$
4. Predict p: $(H \models P_i^* \rightarrow P_j^*)$
   $TM(H, P_i^*) = P_j^*$
5. Intervene p: do p_i^*
6. Observe p: see p_j^*?

In step 1. a phenomenon is observed. This phenomenon is described by a string that represents the observation, given a manner of representation. An hypothesis H* is searched in step 3. in the form of a short program for a Turing machine, such that the program can generate the string and possible continuations of that string. Based on the given manner of representation, the program for H* and the string representing the observed data, the Turing machine can make a prediction by generating a continuation of the string, in step 4. Based on prediction an intervention and observation close the circle. The observation of new data does not necessitate a new hypothesis as long as the description of the new data plus the old hypothesis is still the shortest available description. New data do change the a posteriori probability of predictions.

In the next part of this thesis I will analyze a scientific practice to find out how that practice compares with the epistemological theories addressed in this part.

* * * *