Theory of Computation

Introduction

What is computation?

- Computation is an effective method (an Algorithm!), which given a problem with possibly a finite number of inputs, can produce an output which can be recognized as a solution to the problem.
- 'Effective method' is here used in the rather special sense of a method each step of which is precisely predetermined and which is certain to produce the answer in a finite number of steps.

What is computation?

- Points to note:
- The “size” of both Input and Output should be finite
- The number of steps should also be finite.

- An arbitrarily large natural number is a finite number.

What is a Computer?

- A Computer is a physically realizable machine which can perform computation.
- **A Computer is a Formal System.** This is a direct implication of the requirement that each step in an “Effective Method” is precisely predetermined.

What is Theory of Computation?

- It is the study about the ultimate capability of Computers, that is what problems can be computed, what cannot and what can only be solved partially by any computer which works under the limitation of being “physically realizable” and employing only “Effective Methods”.
- This requires working with abstract models of computers / computation and as we will show, this has more to do with the nature of problems rather than computers.

Formal systems

1. A finite set of symbols, known as the alphabet, which concatenate to give formulas, so that a formula is just a finite string of symbols taken from the alphabet.
2. A grammar consisting of rules to form formulas from simpler formulas. A formula is said to be well-formed if it can be formed using the rules of the formal grammar. It is often required that there be a decision procedure for deciding whether a formula is well-formed.
3. A set of axioms, or axiom schemata, consisting of well-formed formulas.
4. A set of inference rules. A well-formed formula that can be inferred from the axioms is known as a theorem of the formal system.

Derivation in a Formal System

- The derivation rules determine in which cases a formal expression *A* can be deduced from other formal expressions *B*1,…,*Bn*. If *n*=0, then *A* is called an axiom. Derivations are either sequences or tree diagrams made up of formal expressions according to the derivation rules. If there are only axioms at the vertices of the derivation tree, then the formal expression at the end of the derivation is said to be deducible (or derivable) in the formal system.

Effective Formal Systems

- Very interesting formal systems are those for which the language and the concept of derivation satisfy the requirement of effectiveness. This means that there must be an effective procedure for determining whether an arbitrary sequence of symbols is an expression of the formal system or not. The concept of a derivation must satisfy the same requirement. The concept of a deducible expression in effective formal systems is, generally speaking, not effective.

Mathematical Logic

- A calculus in mathematical logic is an effective formal system given by the rules of formation of expressions and of constructing derivations in that calculus. The expressions of a formal system are regarded as purely-formal combinations of symbols;
- The two basic calculi are Zeroth order Logic (**Propositional Calculus**) and First order Logic (**Predicate Calculus**)

Important Properties of Logical Systems

- **Consistency**: no theorem of the system contradicts another.
- **Validity**: the system's rules of proof never allow a false inference from true premises.
- **Completeness**: if a formula is true, it can be proven, i.e. is a *theorem* of the system.
- **Soundness**: if any formula is a theorem of the system, it is true. This is the converse of completeness. (Note that in a distinct philosophical use of the term, an argument is sound when it is both valid and its premises are true.)
- **Expressivity**: what concepts can be expressed in the system.

Semantics – the Idea of Truth

- A formula in a Formal System is just a finite string of symbols taken from the alphabet without any meaning associated with them.
- Semantics is derived by associating a Domain of discourse so that the truth values of Logical Propositions/Predicates, domains of non-logical variables if any and non-logical functions can be defined over this domain. Such an association is called an **interpretation** or a **model**.

Zeroth Order Logic

- **Propositional calculus** is a branch of logic. It is also sometimes called **zeroth-order logic**. It deals with propositions (which can be true or false) and relations between propositions, including the construction of arguments based on them. Compound propositions are formed by connecting propositions by logical connectives. Propositions that contain no logical connectives are called atomic propositions.
- The formal language for propositional logic consists of formulas built up from propositional symbols and logical connectives. The only non-logical symbols in a formal language for propositional logic are the propositional symbols, which are often denoted by capital letters. To make the formal language precise, a specific set of propositional symbols must be fixed.

Zeroth Order Logic

- Any combinational circuit can be expressed in this logic.
- Unlike first-order logic, propositional logic does not deal with non-logical objects, predicates about them, or quantifiers. However, all the machinery of propositional logic is included in first-order logic and higher-order logics. In this sense, propositional logic is the foundation of first-order logic and higher-order logic.

Interpretations for propositional logic

- The standard kind of interpretation in this setting is a function that maps each propositional symbol to one of the truth values true and false. This function is known as a *truth assignment* or *valuation* function. In many presentations, it is literally a truth value that is assigned,
- For a language with n distinct propositional variables there are 2^n distinct possible interpretations.

Interpretations for propositional logic

- Given any truth assignment for a set of propositional symbols, there is a unique extension to an interpretation for all the propositional formulas built up from those variables. This extended interpretation is defined inductively, using the truth-table definitions of the logical connectives discussed above.

First Order Logic

- **First-order logic**—also known as **predicate logic**, **quantificational logic**, and **first-order predicate calculus**—is a collection of formal systems used in mathematics, philosophy, linguistics, and computer science. First-order logic uses quantified variables over non-logical objects, and allows the use of sentences that contain variables, so that rather than propositions such as "Socrates is a man", one can have expressions in the form "there exists x such that x is Socrates and x is a man", where "there exists" is a quantifier, while *x* is a variable. This distinguishes it from propositional logic, which does not use quantifiers or relations; in this sense, propositional logic is the foundation of first-order logic.

First Order Logic

- A theory about a topic is usually a first-order logic together with a specified domain of discourse (over which the quantified variables range), finitely many functions from that domain to itself, finitely many predicates defined on that domain, and a set of axioms believed to hold about them. Sometimes, "theory" is understood in a more formal sense, which is just a set of sentences in first-order logic

First Order Logic

- First-order logic is the standard for the formalization of mathematics into axioms, and is studied in the foundations of mathematics. Peano arithmetic and Zermelo–Fraenkel set theory are axiomatizations of number theory and set theory, respectively, into first-order logic. No first-order theory, however, has the strength to uniquely describe a structure with an infinite domain, such as the natural numbers or the real line. Axiom systems that do fully describe these two structures (that is, categorical axiom systems) can be obtained in stronger logics such as second-order logic.

First Order Logic

- So the boundary of what we can compute is somewhat hazily discernible in the infinite domain of all theories that can be captured by First order Logic.
- Arithmatic? Set Theory? Up to arbitarily large nos.
- Decision Problems? Some of them
- Functions on Natural nos.? Some of them
- We will now discuss First order logic and Arithmatic in greater detail

Language of First Order Logic

- The basic components of FOL are called **terms**. Essentially, a term is an object that denotes some object other than true or false.
- The simplest kind of term is a constant.
- A value such as 8 is a constant; the denotation of this term is the number 8—a value that is contained in the sets $\mathbb{N}$ and Z.
- We often use constants in maths; we introduce them by writing things like Let S be the set {1; 2; 3}. In this case, we have introduced a constant and made its denotation clear; we have given it an interpretation.
- We can have constants that stand for any kind of object; for example, we could have a constant that stood for (denoted) the individual ‘MichaelWooldridge’.

Language of First Order Logic

- The second simplest kind of **term** is a variable.
- A variable can stand for anything in a set of Objects.
- That is, a variable of type $\mathbb{N}$ could stand for any of the natural numbers.
- Lets just formalise this before going any further.
- **Definition:** A constant of type T is a name that denotes some particular object in the set T.
- **Definition:** A variable of type T is a name that can denote any value in the set T.

Language of First Order Logic

- We can now introduce a more complex class of terms — functions
- In FOL, we have a set of function symbols; each symbol corresponds to a particular function. (It denotes some function.)
- Each function symbol is associated with a natural number called its arity. This is just the number of arguments it takes.
- Each function symbol has a return-type associated with it

Language of First Order Logic

- Formally . . .
- **Definition:** Let f be an arbitrary function symbol of type T, with arity n $\in \mathbb{N}$, taking arguments of type T1; : : : ; Tn respectively.
- Also, let 1; : : : ; n be terms of type T1; : : : ; Tn respectively. Then f (1; : : : ; n) is a functional term.

Language of First Order Logic

- Consider the function plus:
- plus(2; 3) is an acceptable functional term;
- plus(0; 1) is acceptable;
- plus(plus(1; 2); 4) is acceptable;
- plus(plus(plus(0; 1); 2); 4) is acceptable;
- but
- plus(-1; 0) isn’t;

Language of First Order Logic

- The language of FOL contains a stock of predicate symbols. These symbols stand for relationships between objects.
- **Definition:** Let P be a predicate symbol of arity n $\in \mathbb{N}$, which takes arguments of
- types T1; : : : ; Tn. Then if 1; : : : ; n are terms
- of type T1; : : : ; Tn respectively, then P(1; : : : ; n) is a predicate, which will either be true or false under some interpretation.

Language of First Order Logic

- So a predicate just expresses a relationship between some values.
- What happens if a predicate contains variables: can we tell if it is true or false?
- Not usually; we need to know an interpretation for the variables.
- A predicate that contains no variables is a proposition

Language of First Order Logic

- Predicates of arity 1 are called properties.
- EXAMPLE. The following are properties:
  - Man(x)
  - Mortal(x)
  - Malfunctioning(x):
- Predicate that have arity 0 (i.e., take no arguments) are called primitive propositions.
- Predicates may be combined using Logical Operators $\neg$, $\wedge$, $\vee$.

Language of First Order Logic

- We now come to the central part of first order logic: quantification.
- Consider trying to represent the following:
  - all men have a mother;
  - every natural number has a prime factor.
- We can't represent these using the apparatus we've got so far; we need quantifiers.

Language of First Order Logic

- In Z, we shall use two quantifers:
- ∀— the universal quantifier; is read ‘for all. . . ’
- ∃ — the existential quantifier; is read ‘there exists. . . ’

Language of First Order Logic

- The simplest form of quantified formula in Z is as follows:
- quantifier signature predicate
- where
- – quantifier is one of ∀, ∃ ;
- – signature is of the form variable : type
- – and predicate is a predicate.

Language of First Order Logic

- EXAMPLES.
  - $\forall$ x : Man Mortal(x)
- ‘For all x of type Man, x is mortal.’
- (i.e. all men are mortal)
- If the logic is used in a type free manner,
  - $\forall$ x Mortal(x)

Language of First Order Logic

- Note that universal quantification is similar to conjunction:
- ∀ n : {2; 4; 6} Even(n)
- is the same as
- Even(2) ^ Even(4) ^ Even(6):
- In the same way, existential quantification is the same as disjunction:
- ∃ n : {7; 8; 9} Prime(n)
- is the same as
- Prime(7) ˅ Prime(8) ˅ Prime(9):

Language of First Order Logic

- The universal and existential quantifiers are in fact duals of each other:
- $\forall x : T \;\; P(x) \;\; \Leftrightarrow \neg \exists x : T \;\; \neg P(x)$
- Saying that everything has some property is the same as saying that there is nothing that does not have the property.

Language of First Order Logic

- $\exists x : T \;\; P(x) \Leftrightarrow \neg \;\; \forall \, x : T \neg P(x)$
- Saying that there is something that has the property is the same as saying that its not the case that everything doesn't have the property.

Interpretation of First Order Logic

- An interpretation of a first-order language assigns a denotation to each non-logical symbol in that language. It also determines a domain of discourse that specifies the range of the quantifiers. The result is that each term is assigned an object that it represents, each predicate is assigned a property of objects, and each sentence is assigned a truth value. In this way, an interpretation provides semantic meaning to the terms, the predicates, and formulas of the language.

Arithmetic – Peano's Axioms

- An Axiomatic Approach to Mathematics:
- In order to approach Arithmetic in a formal way, we have to be very careful when proving our various propositions and theorems to only use results we know to be true.
- However, many of the statements that we take to be true had to be proven at some point. Those proofs, of course, relied on other true statements. If we continue to "trace back" our mathematics proofs, we begin to notice that mathematics must have some initial set of true statements that cannot be proven. These statements, known as axioms, are the starting point for any mathematical theory.
- In this section, we will axiomatically define the natural numbers N.

Arithmetic – Peano's Axioms

- As we move through the various axioms, we will see that each one is crucial in defining N in such a way that they are equal to{0,1,2,3, . . .}, which are the natural numbers that we know and love. After establishing this, we will establish the basic arithmetic operations on N by defining addition and multiplication.
- **Axiomatizaing the Natural Numbers** In this section, we will develop the Peano Axioms and use them to provide a completely formal definition of the natural numbers N. In what follows, itis best to train yourself to assume nothing and use only statements that are known to be true via axioms or statements that follow from these axioms.

Arithmetic – Peano's Axioms

- We will formalize the notion of equality and then present the Peano axioms. The Notion of Equality. When approaching mathematics axiomatically, it is important to not assume anything at all, including something as rudimentary as how equality behaves. After all, "=" is merely a symbol until we declare it to have some important properties. Below, we will define the natural numbers N axiomatically. Before we get deep into this, let's establish the properties that "=" should have. First, every natural number should be equal to itself; this is known as the reflexivity axiom.
- Axiom 1. For every $x \in N$, $x=x$

Arithmetic – Peano's Axioms

- Next, if one natural number equals a second one, then that second one should equal the first one. This is called the symmetry axiom.
- Axiom 2. $\forall x, y \in N$, if $x=y$, then $y=x$
- The next property allows us to say that if one natural number is equal to a second, and that second natural number is equal to a third, then the first and third are equal to each other. This is known as the transitivity axiom.
- Axiom 3. $\forall x, y, z \in N$, if $x=y$ and $y=z$, then $x=z$.

Arithmetic – Peano's Axioms

- The above three properties of reflexivity, symmetry, and transitivity come up numerous times in essentially every mathematical field. Equality is an example of what is called a relation, and relations that enjoy the above three properties are known as equivalence relations. There remains one last axiom related to equality. In each of the above axioms, whenever we used a symbol (like x, y, or z), we always had an assumption that these elements were in the natural numbers. If we don't make this assumption, then we may run into problems.

Arithmetic – Peano's Axioms

- To help get around this, the fourth axiom, called the closure of equality axiom, says that if you have a natural number that is equal to something, then that "something" also has to be a natural number.
- Axiom 4. $\forall$ x and y, if $x \in N$ and $x=y$, then $y \in N$.
- In other words, the only way for something to be equal to a natural number is for it to be a natural number itself. In different versions of the Peano axioms, the above four axioms are excluded, as they these properties of equality are frequently assumed to be true as part of that logic system. For completeness, though, we include them in these notes. Furthermore, their inclusion here highlights the importance of questioning even the most basic mathematical assumptions.

Arithmetic – Peano's Axioms

- Axioms : Now, we are ready to present the main Peano axioms. It is important to keep in mind that when Peano and others constructed these axioms, their goal was to provide the fewest axioms that would generate the natural numbers that everyone was familiar with. The insight is that this could be done by asserting the existence of at least one natural number, and then defining a function, called successor function, that can be used to construct the remaining natural numbers. An obvious element to axiomatically include in the natural numbers is zero

Arithmetic – Peano's Axioms

- Axiom 5: 0 is a natural number.
- That is, $0 \in \mathbb{N}$. In alternate versions of the Peano axioms, Axiom 5 actually replaces 0 with1. This creates an almost identical set of natural numbers, which correspond to "positive whole numbers" (as we known them now). Whether a mathematician includes 0 in the natural numbers or not depends on the context. We use the convention of including 0 as a natural number. At this point, we are only guaranteed the existence of a single natural number, 0.

Arithmetic – Peano's Axioms

- The next axiom uses the successor function to generate other natural numbers. As its name implies, the successor function is a function S that has as its domain N. The next axiom simply states that the co-domain of S is also N.
- Axiom 6: If $x \in N$, then $S(x) \in N$.
- That is, if x is a natural number, then so is its successor.
- As the above axiom implies, we will commonly refer to S(x) as the successor of x. Intuitively, we should think of S(x) as x+1. Of course, we cannot formally define it this way yet since we do not know what + means! At this point, we are still quite far away from having the natural numbers as we know them. For example, if we have N={0} and define S(0) = 0, then all of the above axioms are satisfied. We, of course, want to avoid this.

Arithmetic – Peano's Axioms

- A way to ensure this is to insist that 0 is not the successor of any natural number (including itself, of course).
- Axiom 7: For every natural number $x \in N$, $S(x) = 0$ is false.
- Rephrasing this using our knowledge of functions, we can say that the pre-image of 0 under S in the natural numbers is the empty set. At this point, we are slightly closer to having N look more like the natural numbers we know. In particular, we know that S(0) is not equal to 0, and thus it must equal some other natural number. We can denote this natural number by 1. Thus, we can at this point, we know that N contains at least two natural numbers, 0 and 1.

Arithmetic – Peano's Axioms

- If we were to stop here, though, we could not be guaranteed that all the rest of the natural numbers (as we know them) exist. For example, we could define N={0,1} where S(0) = 1 andS(1) = 1. Again, using our knowledge of functions, we recognize that the fact that S(0) = 1 and S(1) = 1 means that the S is not an injective function. We would like this successor function to be injective, so we have the following axiom.
- Axiom 8. $\forall x, y \in N$, if S(x) =S(y), then x=y.

Arithmetic – Peano's Axioms

- The above axiom has some very important ramifications. First, it excludes the possibility of defining N to be just {0,1}. To see why, notice that we already have that S(0) = 1 and, by injectivity, we cannot have that S(1) = 1. Axiom 6 excludes the possibility that S(1) = 0. Thus,S(1) must be some other natural number, which we denote as 2. Thus, we can define 2 =S(1).. A similar argument gives that S(2) cannot be 0,1, or 2. Thus, it must be some other natural number, which we call 3. Continuing in this pattern, we see that N must contain all the natural numbers that we know! At this point, we have established that N must include 0, its successor 1 =S(0), its successor's successor 2 =S(1), and so on. Thus, we formally have that N must include 0, S(0), S(S(0)), S(S(S(0))), ...

Arithmetic – Peano's Axioms

- Of course, to avoid so many nested applications of S, we use the numerals 1,2,3 to denote S(0), S(S(0)),and S(S(S(0))), respectively. We are, however, not done. These first eight axioms have pushed our formal definition of N to include all of the "usual" natural numbers that we know and love. That is, we know now that$\{0,1,2, \ldots\} \subset N$. However, what disallows our axiomatic N from containing more? So far, nothing does. To see this, let's consider this version of N that satisfies all the above axioms, but is not the usual natural numbers we know: $N=\{0,1,2,3, \ldots,\} \cup \{a, b\}$.That is, this version of N contains all the natural numbers and also includes two other symbols a and b. We also need to describe the successor function 3

Arithmetic – Peano's Axioms

- On the portion{0,1,2,3, . . . ,}, we define S in the way described above, where S(0) = 1, S(1) = 2, S(2) = 3, and so on. On the portion {a, b}, we can define S(a) =b and S(b) = a. This version of N with this successor function satisfies all the axioms, but is "larger" than we want our natural numbers to be. The next (and final) axiom will exclude versions of N that are "too large" from occurring. Before we begin, we need a definition that is inspired by induction. A set V is called inductive if the following two conditions are satisfied: $0 \in V$ If $x \in V$, then $S(x) \in V$

Arithmetic – Peano's Axioms

- Of course, the name comes from the fact that the first condition is similar to our "base case" from induction, and the second condition is analogous to the induction step. The last axiom, many times called the Axiom of Induction says that if V is an inductive set, then V contains the set of natural numbers.
- Axiom 9. If V is an inductive set, then $N \subset V$.
- As stated above, the first 8 axioms ensure that$\{0,1,2,3, \ldots\} \subset N$. Furthermore, notice that the set$\{0,1,2,3, \ldots\}$ is an inductive set! Thus, by Axiom 9,it must be true that $N \subset \{0,1,2,3, \ldots\}$. Thus, we finally have the set equality that we were after: $N = \{0,1,2,3, \ldots\}$

Axiomatic Arithmetic

- As we know, the importance of N does not stem from its set-theoretic properties .Rather, N is of crucial importance because of the arithmetic that we can perform on it. Specifically, addition + and multiplication touch every single aspect of our academic (and non-academic) lives. Given that so much care was taken to axiomatically define the set of natural numbers, we should be sure to carefully define what is meant by addition and multiplication. The Peano axioms and the successor function allow us to do precisely that.

Axiomatic Arithmetic

- Addition We will now, using only the information provided in the Peano Axioms, define the operation + of addition. In what follows, we let a, b ∈ N. Axiom 5 guarantees that 0 ∈ N, so we begin by defining what it means to add 0. Thus, we define a+ 0 = a. To define the sum of any two natural numbers, we use the following recursive definition: a+S(b) =S(a+b).Therefore, if we want to compute 1 + 1, we note that 1 =S(0) and get 1 + 1 =1 +S(0) =S(1 + 0) =S(1) = 2. We can proceed further to compute 1 + 2. To do so, we note that 2 =S(1) and therefore that1 + 2 = 1 +S(1) =S(1 + 1) =S(2) = 3.Notice that this latter computation relied on the previous computation of 1+1 =2 and the fact that we defined 3 to beS(2). Similarly, the first computation of1 + 1 relied on using the given fact that 1 + 0 = 1. Iterating this process enough times will generate the usual addition that we use.

Axiomatic Arithmetic

- Multiplication When we first learned multiplication in early grade school, it was taught by asking us to perform addition in an iterative manner. Similarly, axiomatic multiplication is built upon the previously defined axiomatic addition. As with addition, axiomatic multiplication is given recursively: $a \cdot 0 = 0$ and $a \cdot S(b) = a + (a \cdot b)$
- Thus, we can easily show that $a \cdot 1 = a$ by noting that $1 = S(0)$ and therefore $a \cdot 1 = a \cdot S(0) = a + (a \cdot 0) = a + 0 = a$. We can use this to multiply $3 \cdot 2$. Of course, we know that $2 = S(1)$ and therefore$3 \cdot 2 = 3 \cdot S(1) = 3 + (3 \cdot 1) = 3 + 3 = 6$.Notice that in the last two steps of the computation, we use that $3 \cdot 1 = 3$ (which we proved above) and that $3 + 3 = 6$. This latter statement, of course, can be shown using enough iterations of addition from the previous section

What can be done with FOL

- So apparently we can do basic arithmetic with arbitrarily large numbers.
- The question is can we compute any function over Natural Numbers? Can we prove all truths about arithmetic of natural numbers? The answer is no.
- For this we will revisit Semantics and introduce the notion of Decidability.

Semantics – the Idea of Truth

- An interpretation of a first-order language assigns a denotation to each non-logical symbol in that language. It also determines a domain of discourse that specifies the range of the quantifiers. The result is that each term is assigned an object that it represents, each predicate is assigned a property of objects, and each sentence is assigned a truth value. In this way, an interpretation provides semantic meaning to the terms, the predicates, and formulas of the language. The question is whether there is an effective method to do this

Decidability

- In propositional logic, some formulae are tautologies—they have the property of being true under all interpretations.
- There is a procedure which can be used to tell whether any formula is a tautology—this procedure is the truth-table method.
- A formula of FOL that is true under all interpretations is said to be valid.

Decidability

- Now we can’t use truth tables to tell us whether a formula of FOL is valid.
- Is there any other procedure that we can use, that will be guaranteed to tell us, in a finite amount of time, whether a FOL formula is, or is not, valid?
- The answer is no.
- FOL is for this reason said to be undecidable.

Provability and Logical Validity

- Each logical system comes with both a syntactic component, which among other things determines the notion of provability, and a semantic component, which determines the notion of logical validity. The logically valid formulas of a system are sometimes called the **theorems** of the system, especially in the context of first-order logic where Gödel's completeness theorem establishes the equivalence of semantic and syntactic consequence.

Completeness

- A set of axioms is (*syntactically*, or *negation*-) complete if, for any statement in the axioms' language, that statement or its negation is provable from the axioms
- *semantic* completeness, which means that the set of axioms proves all the semantic tautologies of the given language. In his completeness theorem, Gödel proved that first order logic is *semantically* complete.

GöDel's Incompleteness Theorem

- The first incompleteness theorem states that no consistent system of axioms whose theorems can be listed by an effective procedure (i.e., an algorithm) is capable of proving all truths about the arithmetic of natural numbers. For any such consistent formal system, there will always be statements about natural numbers that are true, but that are unprovable within the system. The second incompleteness theorem, an extension of the first, shows that the system cannot demonstrate its own consistency.

- The theorems are widely, but not universally, interpreted as showing that Hilbert's program to find a complete and consistent set of axioms for all mathematics is impossible.

Unsolvability /Undecidabilty

- The impossibility of solving a given problem exactly by prescribed means. The non-existence of an algorithm or the impossibility of proving or disproving a statement within a formal system. Both aspects will be considered below. The non-existence of an algorithm for settling a given problem is often referred to as the unsolvability of the problem. Sometimes the two words "undecidable" and "unsolvable" are used as synonyms.

Unsolvability /Undecidabilty

- The notion of an algorithm has to be formalized in order to show that some problem is undecidable. The undecidability of a problem means that an algorithm is impossible in principle — not only that no algorithm is presently known.
- Three such formalizations will be considered in the course, namely General Recursive functions, λ-Calculus and Turing Machines

Unsolvability /Undecidabilty

- The most common among such formalizations is a Turing machine. It is to be emphasized, however, that all suggested formalizations have turned out to be equivalent and, moreover, the existence of undecidable problems is independent of the formalization used. An argument showing this will now be briefly outlined.

Unsolvability /Undecidabilty

- Consider any such formalization. For any algorithm *A* and any input word *x* of *A*, there are two possibilities: either *A* halts with *x*, that is, a terminating computation results when *A* is applied to *x*, or *A* does not halt with *x*. In the latter case one says that *A* loops with *x*. The halting problem consists of deciding, for an arbitrary pair (*A*,*x*), whether *A* halts or loops with *x*. This is a well known undecidable problem.

Semidecidability

- A property of a theory or logical system weaker than decidability is **semidecidability**. A theory is semidecidable if there is an effective method which, given an arbitrary formula, will always tell correctly when the formula is in the theory, but may give either a negative answer or no answer at all when the formula is not in the theory.

Semidecidability

- A logical system is semidecidable if there is an effective method for generating theorems (and only theorems) such that every theorem will eventually be generated. This is different from decidability because in a semidecidable system there may be no effective procedure for checking that a formula is *not* a theorem.

Semidecidability

- For example, the set of logical validities *V* of first-order logic is semi-decidable, but not decidable. In this case, it is because there is no effective method for determining for an arbitrary formula *A* whether *A* is not in *V*. Similarly, the set of logical consequences of any recursively enumerable set of first-order axioms is semidecidable.

General Recursive Functions

- In 1933, Kurt Gödel, with Jacques Herbrand, created a formal definition of a class called general recursive functions. The class of general recursive functions is the smallest class of functions (possibly with more than one argument) which includes all constant functions, projections, the successor function, and which is closed under function composition, recursion, and minimization.

λ-calculus

- In 1936, Alonzo Church created a method for defining functions called the λ-calculus. Within λ-calculus, he defined an encoding of the natural numbers called the Church numerals. A function on the natural numbers is called λ-computable if the corresponding function on the Church numerals can be represented by a term of the λ-calculus.
- Language LISP is based on λ-calculus

Turing Machine

- Also in 1936, before learning of Church's work,Alan Turing created a theoretical model for machines, now called Turing machines, that could carry out calculations from inputs by manipulating symbols on a tape. Given a suitable encoding of the natural numbers as sequences of symbols, a function on the natural numbers is called Turing computable if some Turing machine computes the corresponding function on encoded natural numbers

Church's Thesis

- Church and Turing proved that these three formally defined classes of computable functions coincide: a function is λ-computable if and only if it is Turing computable, and if and only if it is *general recursive*. This has led mathematicians and computer scientists to believe that the concept of computability is accurately characterized by these three equivalent processes. Other formal attempts to characterize computability (like combinators) have subsequently strengthened this belief

Resources Required for Solvable Problems

- The main resources we talk about are
  - **Time** and
  - **Space**
- A problem is regarded as inherently difficult if its solution requires significant resources, whatever the algorithm used.
- **Computational complexity theory** focuses on classifying computational problems according to their resource usage, and relating these classes to each other. A computational problem is a task solved by a computer

Complexity Classes

- A **complexity class** is a set of problems of related complexity. Simpler complexity classes are defined by the following factors:
- The type of computational problem: The most commonly used problems are decision problems. However, complexity classes can be defined based on function problems, counting problems, optimization problems, etc.
- The model of computation: The most common model of computation is the deterministic Turing machine, but many complexity classes are based on non-deterministic Turing machines, Boolean circuits, quantum Turing machines, monotone circuits, etc.
- The resource (or resources) that is being bounded and the bound.

Complexity Classes

- But bounding the computation time above by some concrete function $f(n)$ often yields complexity classes that depend on the chosen machine model. For instance, the language $\{xx \mid x \text{ is any binary string}\}$ can be solved in linear time on a multi-tape Turing machine, but necessarily requires quadratic time in the model of single-tape Turing machines

Complexity Classes by Space Requirement

Complexity class	Model of computation	Resource constraint
	Deterministic space	
DSPACE($f(n)$)	Deterministic Turing machine	Space O($f(n)$)
L	Deterministic Turing machine	Space O(log n)
PSPACE	Deterministic Turing machine	Space O(poly(n))
EXPSPACE	Deterministic Turing machine	Space O($2^{\text{poly}(n)}$)

What will not be discussed

- The universe is equivalent to a Turing machine; thus, computing non-recursive functions is physically impossible. This has been termed the strong Church–Turing thesis, or Church–Turing–Deutsch principle, and is a foundation of digital physics.
- The universe is not equivalent to a Turing machine (i.e., the laws of physics are not Turing-computable), but incomputable physical events are not "harnessable" for the construction of a hypercomputer. For example, a universe in which physics involves random real numbers, as opposed to computable reals, would fall into this category.

What will not be discussed

- The universe is a hypercomputer, and it is possible to build physical devices to harness this property and calculate non-recursive functions. For example, it is an open question whether all quantum mechanical events are Turing-computable, although it is known that rigorous models such as quantum Turing machines are equivalent to deterministic Turing machines. (They are not necessarily efficiently equivalent; see above.) John Lucas and Roger Penrose have suggested that the human mind might be the result of some kind of quantum-mechanically enhanced, "non-algorithmic" computation