Computability, Cantor's diagonalization, Russell's Paradox, Gödel's's Incompleteness, Turing Halting Problem.

Advanced Algorithms

By Me Dr. Mustafa Sakalli

March, 06, 2012.

Incompleteness.

- Lecture notes presented here are incomplete, due to the same argument known as Cantors diagonalization argument.

  – Dr.M. Sakalli

Uncountable SET-not enumerable

- A set is uncountable if its cardinal number is larger than that of the set of all cardinal numbers.
- Its characterization iff any holds..
  - No injective correspondence (f) from *X* to the set of natural numbers.
  - X is nonempty and there is surjective function from the set of natural numbers to *X*.
  - The cardinality of *X* is neither finite nor equal to the cardinality of natural numbers!!
  - The cardinality of X is strictly grater than cardinality of natural number.
- If cardinality of subset of a set Y is infinite, than set Y is uncountable.
- Cantor's diagonal argument.
- Uncountable sets
  - R, the cardinality of R (c or $2^{\aleph_0}$,]1 - beth-one) is called *cardinality of the continuum*.]2 beth-two cardinality of more uncountable numbers.
  - Cantor set that is an uncountable subset of R and has Hausdorff dimension number between 0 and 1. (Fact: Any subset of R of Hausdorff dimension greater than 0 must be uncountable).
  - Earlier Cantor poses question if]1 = $\aleph_1$ (aleph1), cardinality of ordinal numbers,
  - First in the list of Hilbert's 23 questions, 1900.

Schubfachprinzip drawer principle

- n pigeons > m nests. 10 to 9, at least one nest must be shared. At least one box must contain no fewer than ceiling(n/m), and at least one box must contain no more than floor(n/m).
- Collision probability with a uniform probability of 1/m is equal to
- $1-\{m(m-1)..(m-n+1)\}/m^n$
- If the cardinality of set A is greater than those of set B, then there is NO injection from A to B.

Numbers that cannot be counted cannot be computed, but infinite c programs can be counted.

- static void Main() 1/7..
- { Console.Write("0.");
  while (true) Console.Write("142857");
- }
- Pi
- …

Cantor's Diagonal Argument

Let T be a set of consisting all **infinite sequences**, and its permutations of S=f(i).

- Characteristic function of sequence is a function f(i)=1.
- If X has characteristic function *f*, then its complement is 1-*f*(i).

Sequence S is **countable** since possible to associate only one number n of N for every element of the sequence S.

And T is countable..

- s_1 = {1…..0 ….}, ➔ f_1(i) mapping *f*(i)
- s_2 = {……., ….}, ➔ f_2(i)
- s_3 =
- ..
- Thus new *s* is not in {s_1, s_2, s_3, ...}.
  This contradicts the assumption that the list has contained all the sequences.

Cantor's Diagonal Argument

- All the possible countable numbers must be included in the set, but always possible to obtain another s_0=cmplt{diag(T)}=cmplt($f_n(n)$)=cmplt($s_{n,n}$), which is $s_0 \notin$ T. Cmplt referes complement of.
- Therefore T is uncountable, since if S_0. included in, there will be another one which cannot be placed in one-to-one correspondence with the set of N.
- There is no bijection between N and T, but T may be subcountable.
- Although both sets are infinite in conclusion there are more infinite sequences of real numbers than that could be mapped with natural numbers.

Cantor's Diagonal Argument

- Some function will not yield bijection condition, f(t)=0.t, check if t=1000.. And its cmpt(t).. Two different sequences corresponds to one number. But possible to modify function and avoid this situation that takes us to.. I am not sure how to prove, but it says that R and T have the same cardinality which is called generalized cardinality of continuum which is also given as cardinality between |S| and |P(s)|. Cardinality between integers and reals is called continuum hypothesis.

Inconsistency of notions.

- P(S) is the Power of a set, defined as all the subsets of S. Cantor's theorem states that cardinality of P(S) is larger then those of S.
- Suppose f is a function from S to P(S),
- T={s $\in$ S: s$\notin$f(s)}, second statement is from Cantor's diagonalization. Either s is in T (by the definition, from the same definition f(s)$\notin$T, then |T|+|f(s)|➔|P(s)|) or if s is not in T, which is a violation of the definition, but also and |T|+|s|➔|P(s)|.
- Inconsistency of "set of all sets that do not contain themsleves in the set".
- If *S* were the set of all sets, then ***P***(*S*) would be bigger than {*S* and a subset of *S*}.
- Similar to Russell's Paradox, based on an unrestricted comprehension scheme, is contradictory.
- Russell's Paradox if let $D = \{x \mid x \text{ not in } x\}$.
  Then *D* in *D* iff *D* not in *D*, a contradiction.
- For example, the conventional proof of the unsolvability of the halting problem is essentially a diagonal argument of Cantors arg.
- Also, diagonalization was originally used to show the existence of arbitrarily hard complexity classes and played a key role in early attempts to prove P does not equal NP. In 2008, diagonalization was used to "slam the door" on Laplace's demon.1

Russell's paradox

- **The set M is the set of all sets that do not contain themselves as members. Does M contain itself?**

- If it does, it is not a member of M according to the definition. If it does not, then it has to be a member of M, again according to the definition of M. Therefore, both of the statements "M is a member of M " and "M is not a member of M " lead to contradictions.

  For example:
- **"A is an element of M if and only if A is not an element of A"**.
- $M = \{s \in S | s \notin f(s)\}$, or $M = \{x | x \notin x\}$.

- **1) List of all lists that do not contain themselves**.
  If the "List of all lists that do not contain themselves" contains itself, then it does not belong to itself and should be removed. However, if it does not list itself, then it should be added to itself.

  2) Barber paradox
  The paradox considers a town with a male barber who shaves all and only those men who do not shave themselves.

Function x → x < 2 is the set of all numbers less than 2.

Set membership is via application: e member-of S iff S(e) is true.

Since (Function x → x < 2)(1) is true, 1 is in this "set".

Now consider P = "the set of all sets that do not contain themselves as members"!:

P = Function x → Not(x)(x) (Note, it may make sense to have a set with itself as member: the set {{{{...}}}}, infinitely receding, has itself as a member; this only happens in so-called non-well-founded set theory).

Now, is P P? Namely is P a member of itself? This is written:

(function x → not(x x)) (function x → not(x x)) --if this were viewed as a **D** program, it would *loop forever*:

Compute Not((Function x → Not(x x))(Function x → Not(x x))))

Now, notice we have P is a member of itself if and only if it isn't, a contradiction!

The computational realization of the paradox is that the predicate cannot compute to true or false, so its not a sensible logical statement.

To avoid this problem, Russell developed his theory of types.

Avoiding Russell's paradox: type theory and axiomatic set theory

- Old classic set of self-referential paradoxical statements causing ambiguities that are neither true nor false could be avoided.
- In mathematical terms. the set of all sets that are members of the themselves, permitting self-referential sets, allowing the set of all sets that contain themselves as a member.
- But the set of all sets that do not contain themselves as a member? are they a member of themselves.

- To avoid paradox, Russell with Whitehead propose a **type theory**, in which sentences were arranged hierarchically.

- This definition avoids the paradox: the definition of R must now define R as a set of type k set containing all sets of type k − 1 and below that do not contain themselves as members. Since R is a type k set, it cannot contain itself, since it cannot contain any type k sets. This avoids the possibility of having to talk about "the set of all sets that are not members of themselves", because the two parts of the sentence are of different types - that is, at different levels.

Avoiding Russell's paradox: type theory and axiomatic set theory

- This definition avoids the paradox: the definition of R must now define R as a set of type k set containing all sets of type k − 1 and below that do not contain themselves as members. Since R is a type k set, it cannot contain itself, since it cannot contain any type k sets. This avoids the possibility of having to talk about "the set of all sets that are not members of themselves", because the two parts of the sentence are of different types - that is, at different levels.

- Another approach to avoid such types of paradoxes was an **axiomatic set theory**, proposed by Ernst Zermelo. This theory determines what operations were allowed and when.

Some Logics used in axiomatizations of mathematics

- A logic usually refers to a set of rules about constructing valid sentences. Here are a few logics. Propositional logic concerns sentences such as $(p \vee \neg q) \wedge (\neg p \wedge r)$ where p, q, r are boolean variables. Recall that the SAT problem consists of determining the satisfiability of such sentences. In first order logic, the symbols allowed are relation and function symbols as well as quantification symbols $\exists$ and $\forall$. For instance, the statement $\forall \mathbf{x S(x)} \neq \mathbf{x}$ is a first order sentence in which x is quantified universally, S() is a unary relation symbol and $\neq$ is a binary relation. Finally, second order logic allows sentences in which one is allowed quantification over structures, i.e., functions and relations. An example of a second order sentence is $\exists \boldsymbol{S} \forall \mathbf{x} \boldsymbol{S}\mathbf{(x)} \neq \mathbf{x}$, where S is a unary relation symbol.

Corollary: There are uncountably many subsets of N. D:\godel.html

- Hence one must accept that not possible to comprehend the list of comprehensible sequences.
- The same applies to "sequences which God can comprehend". Thus omniscience has some limits.

Godel showed that a formal system that is both complete and consistent could not be created since he proved that any formal system which is complete cannot be prevented from including self-referential statements. The way that happens is through something called *models* - a formal system is only valid if there is a *model* which fits its system of rules. If a system is capable of being represented by a model which is complete, then it is *also* capable of being represented by a model which encodes self-referential statements, and thus comes the self referential paradox.

Alan Turing and others were interested in with a mechanical perfect complete computing device. If such a system exists, then one can use a computer program with the rules of that system to enumerate *every* true statement - or can write a program which can take any statement as input, and tell whether or not that statement is true. REDUCTION.

Which was something not possible as proved by Godel. But Turing and others found an easier way to prove its impossibility: the halting problem.

The halting problem considers a simple question: Write a program which looks at any arbitrary computer programs, and determines whether or not any other prg will eventually stop when it is run for some input.

Computing system S(p,i), which computes a function with a pair of inputs that are a program p, and an input i for that program of p, then, we are writing a program P which will take a pair (S_j(q,i), j) as an input, and gives an answer of TRUE if S(q,i) eventually halts.

By feeding the program itself we drive it into a never ending status which means halting problem cannot return an answer which means it will never return a solution, therefore problem is unsolvable. Contradiction that can be also justified with Cantor's digitalization, that is also Godel's incompleteness. P() = S(i, i)+ 1, if S(i, i)={0, 1} ➔ P={1,0}.

Question a program that is going to generate P, cannot generate P, therefore S is incomplete.

- S is a collection of the objects that are members of the set S.
- $f_s(i)$ is a function which takes input $i \in N$, and returns "True 1 or False 0" (decision) for values that are/aren't members of the set, respectively.
  Define a function g which just asks, *what does f_s return if you give it '$s \in S$' as an input*?
- 1) If $f_s(s)$ returns true, then g(s) returns True; if $f_s(s)$ returns False, then g(s) also returns False (declining f's wrong decision, since $s \in S$).
- 2) Think reverse scenario. $s \notin S$, If $f_s(s)$ returns True, then g(s) must return False.
- 1- g(s) is then a confirmation function for *the set of all sets that do contain themselves as members*.
- 2- g(s) is also a confirmation function for *the set of all sets that donot contain themselves as members*.

- Is g a member of itself? There are two different valid functions fitting to the definition of g - one which contains itself, and the other which doesn't contain itself: a flip of g: g'.

- Is g a member of itself? There are two different valid functions fitting to the definition of g - one which contains itself, one which doesn't. but just a flip of g: g'.
- g‘(s) is the function for *the set of all sets that do not contain themselves as members*. if g'(s) returns true, s does not contain itself as member of S, that is s $\notin$ S.

- Suppose s=g’, flipping regions.

- Inconsistency of g'. (Russel’s paradox): if g'(g') returns true, then g' is a member of itself, which is wrong, so g'(g') should have been false. And if g'(g') returns false, then that means that g' is not a member of itself, which means that g'(g') should have returned true.

- We say that the elements of a set can be counted if they can be listed in a single sequence. N, neg or positive wouldn't matter.
- Cantor's D.
- Anything that can be computed according to a finite list of rules, can be computed by one of TM. Consistent

- Briefly, a Turing machine can be thought of as a black box, which performs a calculation of some kind on an *input* number. If the calculation reaches a conclusion, or *halts* then an output number is returned.
- One of the consequences of Turing's theory is that there is a Universal (GENERIC) Turing machine, in other words one which can simulate all possible Turing machines. This means that we can think of the Turing machines as countable and listed T1, T2,... by a Universal Machine through a sort of alphabetical listing. Turing used this to describe his own version of Gödel's Theorem: that there is no mechanical procedure for telling whether a Turing machine will halt on a given input: the Halting Problem.

- The set of functions is uncountable, the set of turing machines (programs) is countable, therefore there are more functions than programs. By the diagonal argument Turing proves that the existence of a TM that decides the halting problem is contradictory.
- Let's represent the result of using the *n*th Turing machine, T_n on the input *i* as $T_n(i)$. Suppose that there was a rule or procedure for deciding whether or not $T_n(i)$ halts for all values of *n* and *i*.
- But then by a similar diagonalising procedure, we can define a new Turing machine, say D, which will halt for all inputs and return the following output for input *i*:
- 0 if $T_i(i)$ does not halt.
- $T_i(i)+1$ if $T_i(i)$ does halt.

- But this machine D must be one of those machines, in other words it must be T*d* for some *d*. However, we just defined it to give a different answer from T_d with input *d*. Contradiction.

- The extra sophistication here over the original diagonalising argument lies in all the listing done is itself computable and any machine T_n may or may not halt in carrying out its computations. None of this enters into Cantor's original diagonal argument.

Computable sequences

- A sequence *f*(*i*) is *computable* if there is a program (Machine) for given input *i* computes *f*(*i*).
- **Gödel proved his Completeness Theorem, namely that a formula is provable from the axioms iff it is valid.**
- **Godel's First InCompleteness Theorem. Any adequate axiomatizable theory is incomplete. In particular the sentence "This sentence is not provable" is true but not provable in the theory.** Enumerate
- **Godel's Second Incompleteness Theorem.** In any consistent axiomatizable theory (axiomatizable means the axioms can be computably generated) which can encode sequences of numbers (and thus the syntactic notions of "formula", "sentence", "proof") the consistency of the system is not provable in the system.
- **The Liar Paradox.** "Truth" for English sentences is not definable in English. *Proof*. Suppose it is. Then so is its complement "False". Let *s* be the sentence "This sentence is false" .
  Since the phrase "This sentence" refers to *s*, we have
  s iff "This sentence is false" iff "*s* is false" iff not *s*.
  A contradiction to the statement. “Everything I say is a lie, I am lying”

What can be computable in principle

- Alonzo Church defined the Lambda calculus,
- Kurt Gödel defined Recursive functions,
- Stephen Kleene defined Formal systems,
- Markov defined, Markov algorithms,
- Emil Post and Alan Turing defined abstract machines now known as Post machines and Turing machines.
- Church Turing thesis. Anything computable with these computational methodologies is computable by a Turing machine.

- Earlier 1900, David Hilbert believed that all of mathematics could be precisely axiomatized. Once this LIST was completed, there would be an "effective procedure", i.e., an algorithm that would accept any precise mathematical statement as input, and, after a finite number of steps, it would reach to decision whether the statement was true or false. YES or NO statement.
- Hilbert was introducing what is called now a *decision procedure* for all of mathematics.

Satisfiable and Valid Structures

- "Hilbert considered the validity problem for first-order logic". First-order logic is a mathematical language in which most mathematical statements can be formulated. This is a special case of decision problem.
- Every statement in first-order logic has a precise meaning in every appropriate logical structure, i.e., it is true or false in each such structure. Those statements that are true in every appropriate structure are called *valid*. Those statements that are true in some structure are called *satisfiable*.

- Notice that a formula, Φ, is valid iff its negation, $\neg\Phi$, is not satisfiable.

- Hilbert calls First Order Logic as *entscheidungsproblem*.
- In a textbook, *Principles of Mathematical Logic* by Hilbert and Ackermann, the authors wrote, "The Entscheidungsproblem is solved when we know a procedure that allows for any given logical expression to decide by finitely many operations its validity or satisfiability. SAT or 3SAT.. Etc..

- Axioms and Inference rules.
- 1930, Gödel's presents a complete axiomatization of first-order logic, based on the *Principia Mathematica* by Whitehead and Russell
- He proves in his Completeness Theorem, that a formula is provable from the axioms iff it is valid.

- In particular, since the axioms are easily recognizable, and rules of inference very simple, there is a mechanical procedure that can LIST out all proofs.

- Each line in a proof is either an axiom, or a inference following from the previous lines by one of the simple rules.

- For any given string of characters, we can tell if it is a proof.

- Thus we can systematically list all strings of characters and check whether each one is a proof. If so, then we can add the proof's last line to our list of theorems.

- In this way, we can list out all theorems, i.e., exactly all the valid formulas of first-order logic, can be listed out by a simple mechanical procedure.

- More precisely, the set of valid formulas is the range of a computable function. In modern terminology we say that the set of valid formulas of first-order logic is *recursively enumerable (r.e.)*.
- "Yes, Φ is valid." However, if Φ were not valid then we might never find this fact out. The list of all the non-valid formulas, or the list of all satisfiable formulas.
- Gödel's Incompleteness Theorem: there is no complete and computable axiomatization of the first-order theory of the natural numbers. That is, there is no reasonable list of axioms from which we can prove exactly all true statements of number theory (Gödel 1931).
- Church and Turing independently prove that the *entscheidungsproblem* is unsolvable. Turing's unsolvability of halting problem.

Application of halting: Goldbach’ s conjecture

- Conjecture: (even n) $\wedge$ ($n \geq 2$), can be represented with two prime numbers p and q as n=p+q.
- That could be disproven via counterexample by simulating an n-state TM, such that quits (halts-if the result is TRUE) if finds a counterexample, an even n $\neq$ (p+q), {p, q}$\in$primes. Then the conjecture will be disproven.
- If the result is FALSE, then the conjecture is proven. That means halting algorithm for this problem must not halt.

- FSM, Busy Beaver. Next week.