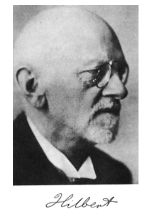
Background; Languages

Models of Computation
Lecture #1
Chapters 1 and 2

Background

- 1900, David Hilbert
- Wanted to know that every true result was provable and wanted an algorithm to reveal the proof



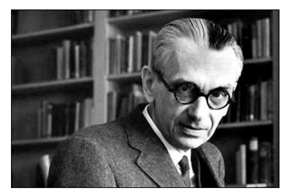
http://aleph0.clarku.edu/~djoyce/hilbert/

Background

- So the new question:
- What statements are provable? And how can we prove them?
- Many mathematicians joined this study…

Background

- 1931, Kurt Gödel
- There is no algorithm to provide proofs of all true statements in mathematics
  - Either there are true statements that can't be proved (or worse!)



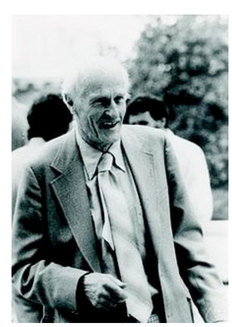
http://25.media.tumblr.com/tumblr_ls48r9i97i1r39l6yo1_400.jpg

Background

- Alonzo Church
  - Defined formal notion of "algorithm"
  - Answered Hilbert's question with a "no" on decidability

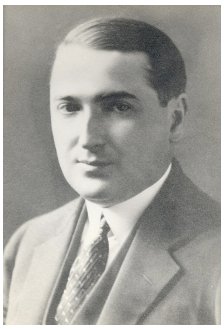
Background

- Stephen Kleene
  - Worked in the domain of computability and pioneered some concepts in mathematical logic



Background

- Emil Post
  - Developed Post's Machine model of computation independently of Turing



Background

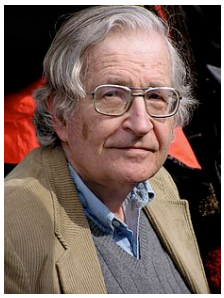
- Alan Turing
  - Most celebrated mathematician in the computer theory domain

Background

- Alan Turing
  - Most celebrated mathematician in the computer theory domain

Background

- Noam Chomsky
  - Mathematical models for describing languages
  - Only one of these pioneers who is still living
    - Professor at MIT
    - political lightning rod



Formal Language

- Letters, words, "sentences"
- Rules of grammar

Formal Languages

- Letters of the "alphabet"
  - Set
  - Σ = { a b c d # }, for example
  - Special symbol, the empty string: Λ

    Λ is not considered part of the alphabet set

Formal Languages

- Words
  Dictionary (for finite languages)
  - Set
  - Γ = { bad cad aa#dd c }, for example

    Note that we call groups of 0 or more letters *strings* or *words* in the language
  - Special symbol, the empty word: Λ

    Λ would appear in the dictionary if it was a valid word in the language under discussion; otherwise, it is not included

Formal Languages

- Languages can be infinite
  Dictionary won't work here
  - ex: English-sentences vs. English words
- Grammar (rules) instead
  - All words that are allowable from the alphabet, can define an infinite (or, of course, finite) LANGUAGE
  - Note that meaning is not implied; form only

Formal Language Oddities

- Language can be empty, containing NO words
  - Symbol for empty language: ϕ

Formal Language Oddities

- Words don't have to make sense
  - not useful, but syntactically valid, English snippet

    whirl monkey tan cannon kite

Formal Language Oddities

- Words don't have to make sense
  - not useful, but syntactically valid, Java snippet

```
while ( i > 0 )
{
    i = 1;
}
```

Defining a Language

For Validation – or – Construction

Σ = { a }

L_1 = { a aa aaa aaaa … }

Defining a Language

For Validation – or – Construction

Σ = { a }

L_1 = { a aa aaa aaaa … }
= { a^n for n = 1 2 3 4 … }

Defining a Language

For Validation – or – Construction

Σ = { a }

L_1 = { a aa aaa aaaa … }
= { a^n for n = 1 2 3 4 … }
= { any grouping or one or more a's }

Defining a Language

For Validation – or – Construction

Σ = { a }

L_1 = { a aa aaa aaaa … }
= { a^n for n = 1 2 3 4 … }
= { any grouping or one or more a's }

Notice that Λ is not in this language

Working with strings

Concatenation

Words placed side-by-side
aaa $\subset L_1$
aa $\subset L_1$
Define x $\leftarrow$ aaa; y $\leftarrow$ aa

Concatenating: xy gives aaaaa

(note: in this case, xy $\subset L_1$)

Working with strings

Question:

Are concatenated strings from the same language always elements of the language?

Working with strings

Question:

Are concatenated strings from the same language always elements of the language?

No. Proof by demonstration:

L_2 = { a aaa aaaaa … }
= { a^{odd} }

Defining a Language

Example
Σ = { x y 0 }
L_3 =
{ x y xx xy x0 yx yy y0 xxx xxy xx0 xyx xyy xy0 x0x x0y x00 … }

L_3 = { any finite string of alphabet letters that does not start with 0 }

Questions:
1. What is the smallest non-empty string *not* in L_3?
2. If r, s $\subset L_3$, then is rs $\subset L_3$?

Defining a Language

- Functions that operate on strings
  length(*string*)
  reverse(*string*)

Defining a Language

- Functions that operate on strings
  length(*string*)
  reverse(*string*)
- length(Λ) ?

Defining a Language

- Define a PALINDROME over a given alphabet

Σ = { a b }
PALINDROME =
{Λ a b aa bb aaa aba bab bbb … }

Defining a Language

- Define a PALINDROME over a given alphabet

Σ = { a b }
PALINDROME =
{Λ a b aa bb aaa aba bab bbb … }
{ Λ and all strings x such that
x = reverse(x) }

Defining a Language

- "Kleene closure" over an alphabet

Σ = { a }
Σ^* = { Λ a aa aaa aaaa … }

Defining a Language

- "Kleene closure" over an alphabet

Σ = { a }
Σ^* = { Λ a aa aaa aaaa … }

Question:
Σ = { 0 1 2 }
Σ^* = ?

Defining a Language

- Kleene closure over a set of words, S

S = { cat dog }
S^* = ?

Defining a Language

- Kleene closure over a set of words, S

S = { cat dog }
S^* =
{ Λ cat dog catdog catcat dogdog dogcat … }

Defining a Language

- Kleene closure over a set of words, S

S = { cat dog }
S^* =
{ Λ cat dog catdog catcat dogdog dogcat … }

Note that even though the words chosen made sense in English, the words in the closure aren't necessarily grammatically valid in English.

Defining a Language

Other examples
S = { aa b}
S^* = ?

Defining a Language

Other examples
S = { aa b}
S^* = ?

S^* = { Λ plus all strings of a's and b's where a's are in even clumps }

Note that this is *not* a proof. Why not?

Defining a Language

Other examples

S = { a ab}
S^* = ?

Defining a Language

Other examples

S = { a ab}
S^* = ?

S^* = { Λ plus all strings of a's and b's that begin with an a and have no double b's }

Again, intuitive but not a proof

Proving inclusion

Is a word in S^* ?

- Yes, if and only if it is factorable into elements of the base set S

Proving inclusion

Is a word in S^* ?

- Yes, if and only if it is factorable into elements of the base set S

Example: S = { a bb abba }
Is bbabbabb in S^*?

Proving inclusion

Is a word in S^* ?

- Yes, if and only if it is factorable into elements of the base set S

Example: S = { a bb abba }
Is bbabbabb in S^*?

Yes: (bb)(abba)(bb)

Proving inclusion

Is a word in S^* ?

- Yes, if and only if it is factorable into elements of the base set S

Example: S = { a bb abba }
Is bbabbabb in S^*?

Yes: (bb)(abba)(bb)
Or: (bb)(a)(bb)(a)(bb) -- not a unique factoring

Proof by Construction

S = { aa aaa }
Prove, for n > 1, a^n is found in S^*

Proof by Construction

S = { aa aaa }
Prove, for n > 1, a^n is found in S^*

Proof:

n = 2; an element of S, (aa)

n = 3; an element of S, (aaa)

n > 3; $a^n = (a^2)(a^{n-2}) = (aa)a^{n-2}$

Positive Closure

- S^+
- excludes the null string

Closure of a closure

$(S^*)^*$
$= S^{**}$

Theorem: $S^{**} = S^*$
To prove, show $S^{**} \subset S^*$ and $S^* \subset S^{**}$

Closure of a closure

Part 1. $S^* \subset S^{**}$

Each element of S^* would have to be in S^{**} since S^{**} is the set made of all possible combinations of S^* including each individual element of S^*

Closure of a closure

Part 2. $S^{**} \subset S^*$

Each element of S^{**} is composed of factors of S^*
Each element of S^* is composed of factors of S
Implying
Each element of S^{**} is composed of factors of S
Therefore
Every element in S^{**} is also in S^*

Homework for Friday

2.1 - 2.5, 2.14, 2.19