CS313H
Logic, Sets, and Functions: Honors
Fall 2012

Prof: Peter Stone
TA: Jacob Schrum
Proctor: Sudheesh Katkam

Department of Computer Science
The University of Texas at Austin

Good Morning, Colleagues

Good Morning, Colleagues

Are there any questions?

Logistics

- Your quest quest is over!

Logistics

- Your quest quest is over! (no more modules)
- Homework due at start of discussion

Logistics

- Your quest quest is over! (no more modules)
- Homework due at start of discussion
- Warning: Quiz on program correctness in discussion

Logistics

- Your quest quest is over! (no more modules)
- Homework due at start of discussion
- Warning: Quiz on program correctness in discussion
  - If you do your homework well, you shouldn't have trouble with the quiz.

Logistics

- Your quest quest is over! (no more modules)
- Homework due at start of discussion
- Warning: Quiz on program correctness in discussion
  - If you do your homework well, you shouldn't have trouble with the quiz.
- Thursday: wrap up and test review

Questions / Important Points

- Diagonalization argument?

Questions / Important Points

- Diagonalization argument?
- How did T(P) work? IGN?

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.
- Prove: There is no program "EQUAL" such that EQUAL(P,Q) outputs "yes" if $\forall I P(I) = Q(I)$ else "no"

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.
- Prove: There is no program "EQUAL" such that EQUAL(P,Q) outputs "yes" if $\forall I P(I) = Q(I)$ else "no"
- i.e. EQUAL tells us if P and Q have same behavior on all inputs

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.
- Prove: There is no program "EQUAL" such that EQUAL(P,Q) outputs "yes" if $\forall I P(I) = Q(I)$
  else "no"
- i.e. EQUAL tells us if P and Q have same behavior on all inputs
- Let HI = Print "Hello"; halt;

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.
- Prove: There is no program "EQUAL" such that EQUAL(P,Q) outputs "yes" if $\forall I P(I) = Q(I)$ else "no"
- i.e. EQUAL tells us if P and Q have same behavior on all inputs

- Let HI = Print "Hello"; halt;
- P $\in$ HELLO iff EQUAL(P,HI) = yes

Proving Undecidability

- HELLO = {P | P Prints "Hello" and halts}
- Recall: HELLO is undecidable.
- Prove: There is no program "EQUAL" such that EQUAL(P,Q) outputs "yes" if $\forall I P(I) = Q(I)$ else "no"
- i.e. EQUAL tells us if P and Q have same behavior on all inputs
- Let HI = Print "Hello"; halt;
- P $\in$ HELLO iff EQUAL(P,HI) = yes
- So EQUAL would give us a decision procedure for HELLO

Enumerating K

- K is not decidable

Enumerating K

- K is not decidable
- But we can write a program to enumerate its elements!

Enumerating K

- K is not decidable
- But we can write a program to enumerate its elements!
- How?

Enumerating K

- K is not decidable
- But we can write a program to enumerate its elements!
- How?
- $\overline{K}$ = {P | P(P) does not halt}

Enumerating K

- K is not decidable
- But we can write a program to enumerate its elements!
- How?
- $\overline{K}$ = {P | P(P) does not halt}
- Theorem: $\overline{K}$ can't be enumerated by a program

Enumerating K

- K is not decidable
- But we can write a program to enumerate its elements!
- How?
- $\overline{K}$ = {P | P(P) does not halt}
- Theorem: $\overline{K}$ can't be enumerated by a program
- Why not?

Vocabulary

- Decidable set also called **Recursive**

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.
  - K recursive?

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.
  - K recursive? no
  - K recursively enumerable?

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.
  - K recursive? no
  - K recursively enumerable? yes
  - $\overline{K}$ recursively enumerable?

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.
  - K recursive? no
  - K recursively enumerable? yes
  - $\overline{K}$ recursively enumerable? no

Vocabulary

- Decidable set also called **Recursive**
  - (Nothing to do with recursion)
- A set whose elements can be enumerated by a program is called **Recursively enumerable**.
  - K recursive? no
  - K recursively enumerable? yes
  - $\overline{K}$ recursively enumerable? no
- Whole topic: "Computability Theory"

Philosophy: Church-Turing Thesis

- Any calculation method that can be grasped and performed by the human mind can be programmed on a conventional digital computer.

Philosophy: Church-Turing Thesis

- Any calculation method that can be grasped and performed by the human mind can be programmed on a conventional digital computer.

- If true, problems which are undecidable to a computer are similarly undecidable to the human mind.

Philosophy: Church-Turing Thesis

- Any calculation method that can be grasped and performed by the human mind can be programmed on a conventional digital computer.
- If true, problems which are undecidable to a computer are similarly undecidable to the human mind.
- A matter of belief...

Undecidable Problems

- Given an initial configuration in the game of life, will it go on forever?
- Given 2 context-free grammars, are they equivalent?
- Given a multi-variate polynomial ove rthe integers, does it have a root?
- Generalization of the Collatz conjecture

Undecidable Problems

- Given an initial configuration in the game of life, will it go on forever?
- Given 2 context-free grammars, are they equivalent?
- Given a multi-variate polynomial ove rthe integers, does it have a root?
- Generalization of the Collatz conjecture
  - n even: $\rightarrow$ n/2

Undecidable Problems

- Given an initial configuration in the game of life, will it go on forever?
- Given 2 context-free grammars, are they equivalent?
- Given a multi-variate polynomial ove rthe integers, does it have a root?
- Generalization of the Collatz conjecture
  - n even: $\rightarrow$ n/2
  - n odd: $\rightarrow$ 3n+1