Copyrighted material

These slides contain copyrighted material for exclusive use by the students currently enrolled in this course and for the duration of current semester only.

No other use or access, or use or access by other persons is allowed.

CSC 301

A classroom presentation on

Limits of Computability

by

Dr. Marek A. Suchenek ©

Computer Science

CSUDH

Short definition of proof

Short definition of proof

1) Proofs never lie

Short definition of proof

1) Proofs never lie

2) Proofs are finite

Short definition of proof

1) Proofs never lie

2) Proofs are finite

3) A diligent reader can recognize any proof whenever he sees one

Short definition of proof

1) Proofs never lie

2) Proofs are finite

3) A diligent reader can recognize any proof whenever he sees one (i.e., the question whether a finite sequence is a proof or not is effectively decidable)

Proof systems

Proof systems

1) Proof system is any fixed collection of proofs all of which satisfy the short definition of proof.

Proof systems

1) Proof system is any fixed collection of proofs all of which satisfy the short definition of proof.

1) There are many proof systems.

Proof systems

1) Proof system is any fixed collection of proofs all of which satisfy the short definition of proof.

2) There are many proof systems.

3) Examples:

Proof systems

1) Proof system is any fixed collection of proofs all of which satisfy the short definition of proof.

2) There are many proof systems.

3) Examples: the empty set

Proof systems

1) Proof system is any fixed collection of proofs all of which satisfy the short definition of proof.

2) There are many proof systems.

3) Examples: the empty set,
the set of proofs in predicate calculus

Ambitious goal

Ambitious goal

1) Find a proof system Pr such that:

Ambitious goal

1) Find a proof system Pr such that:
   every true sentence φ

Ambitious goal

1) Find a proof system Pr such that:
   every true sentence φ
   has its proof in Pr.

Ambitious goal

1) Find a proof system Pr such that:
   every true sentence φ
   has its proof in Pr.

2) Prove all true sentences using Pr

Ambitious goal

1) Find a proof system Pr such that:
   every true sentence φ
   has its proof in Pr.

2) Prove all true sentences using Pr

3) Given 1), use computers to accomplish 2)

Ambitious goal

1) Find a proof system Pr such that:
   every true sentence φ
   has its proof in Pr.

2) Prove all true sentences using Pr

3) Given 1), use computers to accomplish 2) (possible since proofs are decidable)

A fatal flaw

A fatal flaw

The Ambitious Goal is unattainable!

A fatal flaw

The Ambitious Goal is unattainable!

No matter how powerful computers we have.

A fatal flaw

The Ambitious Goal is unattainable!

No matter how powerful computers we have.

Even if there were no time limitations to accomplish the Ambitious Goal.

A fatal flaw

Example

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr.

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr. For instance, C can be operated by a very clever professor who is the best expert in proving things

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr. For instance, C can be operated by a very clever professor who is the best expert in proving things with and without computers.

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr.

A fatal flaw

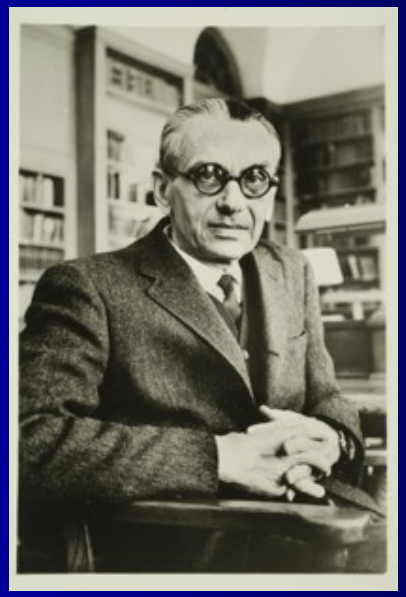
Example

Let C be some very powerful computer with software that can prove things using proof system Pr.

Computer C cannot prove this sentence.

A fatal flaw

Kurt Gödel



Computer C cannot prove this sentence.

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr.

Computer C cannot prove this sentence.

A fatal flaw

Example

Let C be some very powerful computer with software that can prove things using proof system Pr.

Computer C cannot prove this sentence.

But we can prove the above sentence!

A fatal flaw

Computer C cannot prove this sentence.

But we can prove the above sentence!

A fatal flaw

Computer C cannot prove this sentence.

But we can prove the above sentence!

If C can prove the above sentence

A fatal flaw

Computer C cannot prove this sentence.

But we can prove the above sentence!

If C can prove the above sentence then the above sentence is true

A fatal flaw

Computer C cannot prove this sentence.

But we can prove the above sentence!

If C can prove the above sentence then the above sentence is true because proofs never lie.

A fatal flaw

Computer C cannot prove this sentence.

But we can prove the above sentence!

If C can prove the above sentence then the above sentence is true because proofs never lie.

Therefore, C cannot prove it.

A fatal flaw

Conclusion:

A fatal flaw

Conclusion:

For every computer C

A fatal flaw

Conclusion:

For every computer C there is a true sentence φ

A fatal flaw

Conclusion:

For every computer C there is a true sentence φ that C cannot prove

A fatal flaw

Conclusion:

For every computer C there is a true sentence φ that C cannot prove even if all scientists of the world are helping it.

Computable functions

Computable functions

1) Every function computed by a Java program is computable.

Computable functions

1) Every function computed by a Java program is computable.

2) No other function is computable.

Computable functions

1) Every function computed by a Java program is computable.

2) No other function is computable.

3) Example

Computable functions

1) Every function computed by a Java program is computable.

2) No other function is computable.

3) Example: Function

$$f(x) = 2x$$

Computable functions

1) Every function computed by a Java program is computable.

2) No other function is computable.

3) Example: Function

$$f(x) = 2x$$

is computable.

Limitations

Limitations

No computer can always correctly decide

Limitations

No computer can always correctly decide whether given Java program

Limitations

No computer can always correctly decide whether given Java program correctly computes function

Limitations

No computer can always correctly decide whether given Java program correctly computes function

$$f(x) = 2x.$$

Limitations

Limitations

No computer can always correctly decide whether given Java program halts for every valid input to that program.

Limitations

Alan Turing

Limitations

Limitations

For every proof system Pr

Limitations

For every proof system Pr

there exists a total computable function F

Limitations

For every proof system Pr
there exists a total computable function F
such that no program that computes F(x)

Limitations

For every proof system Pr
there exists a total computable function F
such that no program that computes F(x)
can be proved

Limitations

For every proof system Pr
there exists a total computable function F
such that no program that computes F(x)
can be proved (in Pr)

Limitations

For every proof system Pr
there exists a total computable function F
such that no program that computes F(x)
can be proved

Limitations

For every proof system Pr
there exists a total computable function F
such that no program that computes F(x)
can be proved to halt on every input x.

Limitations

And there are many more examples ...

Limitations

As a matter of fact, almost every function is non-computable

Limitations

As a matter of fact, almost every function is non-computable and almost every problem is unsolvable.

Limitations

Here is a classic example that illustrates how seemingly simple programs can be difficult to figure out.

Limitations

Here is a "simple" recursive program whose performance appears very hard to evaluate:

```
public static int f(int n)
 {
     if (n <= 1) return n;
     if (n%2 == 0) return (f(n/2));
           else return (f(3*n + 1));
 }
```

Limitations

For instance, the execution trace for n = 15 is:

n: 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Limitations

It is not known whether the above program always halts or falls into an endless loop for some integer n.

Conclusions

Conclusions

So, if anyone tells you

Conclusions

So, if anyone tells you that there are no limits on computability

Conclusions

So, if anyone tells you that there are no limits on computability, he is flat wrong!

Conclusions

This explains why

Conclusions

This explains why when the government and its bureaucracy

Conclusions

This explains why when the government and its bureaucracy are interfering with free market and competition

Conclusions

This explains why when the government and its bureaucracy are interfering with free market and competition by means of regulation

Conclusions

This explains why when the government and its bureaucracy are interfering with free market and competition by means of regulation, “rationalization”

Conclusions

This explains why when the government and its bureaucracy are interfering with free market and competition by means of regulation, “rationalization”, and redistribution

Conclusions

This explains why when the government and its bureaucracy are interfering with free market and competition by means of regulation, “rationalization”, and redistribution, the economy must get worse.

Conclusions

They simply take upon the task that is computationally unattainable.

Conclusions

They simply take upon the task that is computationally unattainable.

And they worsen the economy in the process

Conclusions

They simply take upon the task that is computationally unattainable.

And they worsen the economy in the process, often blaming free-market capitalism for the problems that they have caused.

Conclusions

There is more Computer Science to it.

Conclusions

There is more Computer Science to it.

The government, its bureaucracy, and central planners act like a polynomially-bound deterministic algorithm.

Conclusions

There is more Computer Science to it.

The government, its bureaucracy, and central planners act like a polynomially-bound deterministic algorithm.

Note: “Polynomially-bound” includes all algorithms that can be executed in a practically feasible amount of time.

Conclusions

There is more Computer Science to it.

The government, its bureaucracy, and central planners act like a polynomially-bound deterministic algorithm.

Conclusions

There is more Computer Science to it.

The government, its bureaucracy, and central planners act like a polynomially-bound deterministic algorithm.

Free market acts like a polynomially-bound nondeterministic algorithm.

Conclusions

Free market acts like a polynomially-bound nondeterministic algorithm.

In this context, nondeterminism is a model of freedom.

Conclusions

Let

P be the class of problems that are solvable by polynomially-bound deterministic algorithms

Conclusions

Let

P be the class of problems that are solvable by polynomially-bound deterministic algorithms

NP be the class of problems that are solvable by polynomially-bound nondeterministic algorithms

Conclusions

Although every problem in P is also in NP,

Conclusions

Although every problem in P is also in NP,

the leading theoretical Computer Scientists believe that NP is larger than P.

Conclusions

In particular, a large collection of practical problems like:

Conclusions

In particular, a large collection of practical problems like:

optimal job scheduling,

Conclusions

In particular, a large collection of practical problems like:

optimal job scheduling, optimal delivery routing,

Conclusions

In particular, a large collection of practical problems like:

optimal job scheduling, optimal delivery routing, satisfiability of a propositional formula,

Conclusions

In particular, a large collection of practical problems like:

optimal job scheduling, optimal delivery routing, satisfiability of a propositional formula, and many more

Conclusions

In particular, a large collection of practical problems like:

optimal job scheduling, optimal delivery routing, satisfiability of a propositional formula, and many more

are in NP but are *not* believed to be in P.

Conclusions

All such problems are solvable in a reasonable time by a nondeterministic algorithm,

Conclusions

All such problems are solvable in a reasonable time by a nondeterministic algorithm, but all their known solutions by any deterministic algorithm are generally so slow that they are impractical.

Conclusions

All such problems are solvable in a reasonable time by a nondeterministic algorithm, but all their known solutions by any deterministic algorithm are generally so slow that they are impractical.

Conclusions

Let’s take it for granted, following the belief of leading theoretical Computer Scientists, that NP is larger than P.

Conclusions

This scientifically explains why the Internet has dramatically more computing power than any centralized mainframe computer, no matter how large.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

If you still have doubts, just look at gridlocks on SoCal freeways to see the limitations on what a government can handle.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

Conclusions

This scientifically explains why governmental central planning often fails where free market succeeds.

Note: Freedom rules out determinism and makes planning generally impossible.

Conclusions

Also, it rules out general practicality of utilitarianism which attempts to deterministically solve some NP-hard problems.

Conclusions

Also, it rules out general practicality of utilitarianism which attempts to deterministically solve some NP-hard problems.

Thus utilitarianism is too simplistic to compete with free market.

Conclusions

Also, it rules out general practicality of utilitarianism which attempts to deterministically solve some NP-hard problems.

In particular, socialism is too simplistic to compete with free market.

Conclusions

Also, it rules out general practicality of utilitarianism which attempts to deterministically solve some NP-hard problems.

In particular, socialism is too simplistic to compete with free market.

No surprise that it doesn't deliver!

Conclusions

The above results remain true even without the assumption that NP is larger than P.