The Complexity of Theorem-Proving Procedures (by Stephen A. Cook)

Emmanouil Lardas

November 23, 2023

Outline of the Presentation

Introduction and Basic Concepts

- We are interested in recoginition problems. Specifically, the difficulty of recognizing sets of strings.

Introduction and Basic Concepts

- We are interested in recoginition problems. Specifically, the difficulty of recognizing sets of strings.
- For this purpose, a concept of "difficulty" that is based on a certain kind of reduction (called P-reduction, P for polynomial) is introduced.

Introduction and Basic Concepts

- We are interested in recoginition problems. Specifically, the difficulty of recognizing sets of strings.
- For this purpose, a concept of "difficulty" that is based on a certain kind of reduction (called P-reduction, P for polynomial) is introduced.
- What does it mean for a set of strings S to be reducible to a set of strings T?

Introduction and Basic Concepts

- We are interested in recoginition problems. Specifically, the difficulty of recognizing sets of strings.
- For this purpose, a concept of "difficulty" that is based on a certain kind of reduction (called P-reduction, P for polynomial) is introduced.
- What does it mean for a set of strings S to be reducible to a set of strings T?
- It means that if we had an oracle that could instantly respond to any query about whether or not a given string is in T, then we would be able to recognize S in polynomial time, deterministically.

Introduction and Basic Concepts

- We are interested in recoginition problems. Specifically, the difficulty of recognizing sets of strings.
- For this purpose, a concept of “difficulty” that is based on a certain kind of reduction (called P-reduction, P for polynomial) is introduced.
- What does it mean for a set of strings S to be reducible to a set of strings T?
- It means that if we had an oracle that could instantly respond to any query about whether or not a given string is in T, then we would be able to recognize S in polynomial time, deterministically.
- It is assumed that all strings contain characters from a fixed, finite alphabet Σ, which is unspecified, but large enough to contain every necessary character.

Notation

- We will be talking about formulas in propositional calculus, which means we will need infinite propositional sumbols (atoms). They will be represented as strings by a member of Σ, followed by the binary representation of a number.

Notation

- We will be talking about formulas in propositional calculus, which means we will need infinite propositional sumbols (atoms). They will be represented as strings by a member of Σ, followed by the binary representation of a number.
- We will also be using the symbols $\neg, \wedge, \vee$, with their usual meanings.

Notation

- We will be talking about formulas in propositional calculus, which means we will need infinite propositional sumbols (atoms). They will be represented as strings by a member of Σ, followed by the binary representation of a number.
- We will also be using the symbols $\neg, \wedge, \vee$, with their usual meanings.
- We also define the set $\{\text{tautologies}\}$ of all strings that represent tautologies.

Basic Definitions

Definition

A query machine is a multitape Turing machine with a distinguished tape called the query tape and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape and each time M assumes the query state there is a string u on the query tape and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle", which knows T, placing M in the yes state or no state.

Basic Definitions

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that, for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps, where $|w|$ is the length of w, and ends in an accepting state iff $w \in S$.

Basic Definitions

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that, for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps, where $|w|$ is the length of w, and ends in an accepting state iff $w \in S$.

- This relation is transitive.

Basic Definitions

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that, for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps, where $|w|$ is the length of w, and ends in an accepting state iff $w \in S$.

- This relation is transitive.
- The relation of two sets of strings being P-reducible to each other is an equivalence relation.

Basic Definitions

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that, for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps, where $|w|$ is the length of w, and ends in an accepting state iff $w \in S$.

- This relation is transitive.
- The relation of two sets of strings being P-reducible to each other is an equivalence relation.
- The equivalence class of a set of strings S is denoted by $\deg(S)$ (the polynomial degree of difficulty of S).

Basic Definitions

Definition

A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that, for each input string w, the T-computation of M with input w halts within $Q(|w|)$ steps, where $|w|$ is the length of w, and ends in an accepting state iff $w \in S$.

- This relation is transitive.
- The relation of two sets of strings being P-reducible to each other is an equivalence relation.
- The equivalence class of a set of strings S is denoted by $\deg(S)$ (the polynomial degree of difficulty of S).
- $\mathcal{L}_* = \deg(\emptyset)$ is the class of sets of strings for which membership can be decided in polynomial time.

Basic Definitions

Some interesting problems (i.e. sets of strings):

Basic Definitions

Some interesting problems (i.e. sets of strings):

- {subgraph pairs}

Basic Definitions

Some interesting problems (i.e. sets of strings):

- {subgraph pairs}
- {isomorphic graphpairs}

Basic Definitions

Some interesting problems (i.e. sets of strings):

- {subgraph pairs}
- {isomorphic graphpairs}
- {primes}

Basic Definitions

Some interesting problems (i.e. sets of strings):

- {subgraph pairs}
- {isomorphic graphpairs}
- {primes}
- {DNF tautologies}

Basic Definitions

Some interesting problems (i.e. sets of strings):

- {subgraph pairs}
- {isomorphic graphpairs}
- {primes}
- {DNF tautologies}
- D_3

Main Results

Theorem

If a set S of strings is accepted by some nondeterministic Turing machine within polynomial time, then S is P-reducible to $\{DNF\ tautologies\}$.

Main Results

Theorem

If a set S of strings is accepted by some nondeterministic Turing machine within polynomial time, then S is P-reducible to $\{$DNF tautologies$\}$.

Corollary

Each of the previous sets is P-reducible to $\{$DNF tautologies$\}$.

Main Results

Proof of the Theorem:

Main Results

Proof of the Theorem:

What we have: Nondeterministic Turing machine M, which accepts S in time $Q(n)$, and input w.

Main Results

Proof of the Theorem:

What we have: Nondeterministic Turing machine M, which accepts S in time $Q(n)$, and input w.
What we want: A formula in DNF such that the input is in S iff the formula is not a tautology.

Main Results

Proof of the Theorem:

What we have: Nondeterministic Turing machine M, which accepts S in time $Q(n)$, and input w.
What we want: A formula in DNF such that the input is in S iff the formula is not a tautology.
Method: We will define a formula $A(w)$ in CNF, which is satisfiable iff M accepts w. Then $\neg A(w)$ can be put in DNF using De Morgan's laws and $w \in S$ iff $A(w)$ is not a tautology.

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)
- Time: $T = Q(n)$, where $n = |w|$

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)
- Time: $T = Q(n)$, where $n = |w|$
- Proposition symbols ($s, t \in \{1, \ldots, T\}$):

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)
- Time: $T = Q(n)$, where $n = |w|$
- Proposition symbols ($s, t \in \{1, \ldots, T\}$):
  - $P^i_{s,t}$ ($i \in \{1, \ldots, l\}$), for the symbols in the tape squares

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)
- Time: $T = Q(n)$, where $n = |w|$
- Proposition symbols ($s, t \in \{1, \ldots, T\}$):
  - $P^i_{s,t}$ ($i \in \{1, \ldots, l\}$), for the symbols in the tape squares
  - Q^i_t ($i \in \{1, \ldots, r\}$), for the states

Main Results

Notation (some small changes have been made to the notation of the paper, in an attempt to make it more consistent):

- Tape alphabet: $\{\sigma_1, \ldots, \sigma_l\}$ (σ_1 is the blank symbol)
- States: $\{q_1, \ldots, q_r\}$ (q_1 and q_r are the starting state and accepting state, respectively)
- Time: $T = Q(n)$, where $n = |w|$
- Proposition symbols ($s, t \in \{1, \ldots, T\}$):
  - $P^i_{s,t}$ ($i \in \{1, \ldots, l\}$), for the symbols in the tape squares
  - Q^i_t ($i \in \{1, \ldots, r\}$), for the states
  - $S_{s,t}$, for the Turing machine head position

Main Results

The formula:

Main Results

The formula:

- $A(w) = B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$, where:

Main Results

The formula:

- $A(w) = B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$, where:
  - $B = \bigwedge_{t=1}^{T} B_t$

Main Results

The formula:

- $A(w) = B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$, where:
  - $B = \bigwedge_{t=1}^{T} B_t$
    - $B_t = \left(\bigvee_{s=1}^{T} S_{s,t}\right) \wedge \left(\bigvee_{\substack{s_1, s_2 \in \{1,\ldots,T\} \\ s_1 \neq s_2}} (\neg S_{s_1,t} \vee \neg S_{s_2,t})\right)$

Main Results

The formula:

- $A(w) = B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$, where:
  - $B = \bigwedge_{t=1}^{T} B_t$
    - $B_t = \left(\bigvee_{s=1}^{T} S_{s,t} \right) \wedge \left(\bigvee_{\substack{s_1, s_2 \in \{1, \ldots, T\} \\ s_1 \neq s_2}} (\neg S_{s_1,t} \vee \neg S_{s_2,t}) \right)$
  - $C = \bigwedge_{s,t=1}^{T} C_{s,t}$

Main Results

The formula:

- $A(w) = B \wedge C \wedge D \wedge E \wedge F \wedge G \wedge H \wedge I$, where:
  - $B = \bigwedge_{t=1}^{T} B_t$
    - $B_t = \left(\bigvee_{s=1}^{T} S_{s,t}\right) \wedge \left(\bigvee_{\substack{s_1, s_2 \in \{1,\dots,T\} \\ s_1 \neq s_2}} (\neg S_{s_1,t} \vee \neg S_{s_2,t})\right)$
  - $C = \bigwedge_{s,t=1}^{T} C_{s,t}$
    - $C_{s,t} = \left(\bigvee_{i=1}^{I} P^{i}_{s,t}\right) \wedge \left(\bigvee_{\substack{i_1, i_2 \in \{1,\dots,I\} \\ i_1 \neq i_2}} (\neg P^{i_1}_{s,t} \vee \neg P^{i_2}_{s,t})\right)$

Main Results

- $D = \bigwedge_{t=1}^{T} D_t$

Main Results

- $D = \bigwedge_{t=1}^{T} D_t$
    - $D_t = \left(\bigvee_{i=1}^{r} Q_t^i \right) \wedge \left(\bigvee_{\substack{i_1, i_2 \in \{1, \dots, r\} \\ i_1 \neq i_2}} (\neg Q_t^{i_1} \vee \neg Q_t^{i_2}) \right)$

Main Results

- $D = \bigwedge_{t=1}^{T} D_t$
  - $D_t = \left(\bigvee_{i=1}^{r} Q_t^i \right) \wedge \left(\bigvee_{\substack{i_1, i_2 \in \{1, \ldots, r\} \\ i_1 \neq i_2}} (\neg Q_t^{i_1} \vee \neg Q_t^{i_2}) \right)$
- $E = Q_1^1 \wedge S_1^1 \wedge \bigwedge_{s=1}^{n} P_{s,1}^{i_s} \wedge \bigwedge_{s=n+1}^{T} P_{s,1}^1$ (όπου $w = \sigma_{i_1} \cdots \sigma_{i_n}$)

Main Results

- $D = \bigwedge_{t=1}^{T} D_t$
  - $D_t = \left(\bigvee_{i=1}^{r} Q_t^i \right) \wedge \left(\bigvee_{\substack{i_1, i_2 \in \{1, \ldots, r\} \\ i_1 \neq i_2}} (\neg Q_t^{i_1} \vee \neg Q_t^{i_2}) \right)$
- $E = Q_1^1 \wedge S_1^1 \wedge \bigwedge_{s=1}^{n} P_{s,1}^{i_s} \wedge \bigwedge_{s=n+1}^{T} P_{s,1}^{1}$ (όπου $w = \sigma_{i_1} \cdots \sigma_{i_n}$)
- $F = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} F_{i,j}^{t}$

Main Results

- $D = \bigwedge_{t=1}^{T} D_t$
  - $D_t = \left(\bigvee_{i=1}^{r} Q_t^i \right) \wedge \left(\bigvee_{\substack{i_1, i_2 \in \{1, \ldots, r\} \\ i_1 \neq i_2}} (\neg Q_t^{i_1} \vee \neg Q_t^{i_2}) \right)$
- $E = Q_1^1 \wedge S_1^1 \wedge \bigwedge_{s=1}^{n} P_{s,1}^{i_s} \wedge \bigwedge_{s=n+1}^{T} P_{s,1}^1$ (όπου $w = \sigma_{i_1} \cdots \sigma_{i_n}$)
- $F = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} F_{i,j}^t$
  - $F_{i,j}^t = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee P_{s,t+1}^k \right)$ (where σ_k is the symbol given by M's transition function at (q_i, σ_j))

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$
  - $G_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^{j} \vee Q_{t+1}^{k} \right)$ (where q_k is the state given by M's transition function at (q_i, σ_j))

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$
  - $G_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee Q_{t+1}^k \right)$ (where q_k is the state given by M's transition function at (q_i, σ_j))
- $H = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} H_{i,j}^{t}$

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$
  - $G_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee Q_{t+1}^k \right)$ (where q_k is the state given by M's transition function at (q_i, σ_j))
- $H = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} H_{i,j}^{t}$
  - $H_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee S_{k,t}^k \right)$ (where k is the tape cell to which M's head must move, according to M's transition function at (q_i, σ_j))

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$
  - $G_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^{j} \vee Q_{t+1}^{k} \right)$ (where q_k is the state given by M's transition function at (q_i, σ_j))
- $H = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} H_{i,j}^{t}$
  - $H_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^{j} \vee S_{k,t}^{k} \right)$ (where k is the tape cell to which M's head must move, according to M's transition function at (q_i, σ_j))
- $I = \bigvee_{t=1}^{T} Q_t^r$

Main Results

- $G = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} G_{i,j}^{t}$
  - $G_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee Q_{t+1}^k\right)$ (where q_k is the state given by M's transition function at (q_i, σ_j))
- $H = \bigwedge_{t=1}^{T} \bigwedge_{i=1}^{r} \bigwedge_{j=1}^{l} H_{i,j}^{t}$
  - $H_{i,j}^{t} = \bigwedge_{s=1}^{T} \left(\neg Q_t^i \vee \neg S_{s,t} \vee \neg P_{s,t}^j \vee S_{k,t}^k\right)$ (where k is the tape cell to which M's head must move, according to M's transition function at (q_i, σ_j))
- $I = \bigvee_{t=1}^{T} Q_t^r$

Note: There appears to be a slight omission, regarding the nondeterministic nature of M (it has a transition relation, not function). However, this should not affect the correctness of the results.

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

- By the corollary to the first Theorem, each of the sets is P-reducible to {DNF tautologies}.

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

- By the corollary to the first Theorem, each of the sets is P-reducible to {DNF tautologies}.
- Obviously, {DNF tautologies} is P-reducible to {tautologies}.

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

- By the corollary to the first Theorem, each of the sets is P-reducible to {DNF tautologies}.
- Obviously, {DNF tautologies} is P-reducible to {tautologies}.
- It remains to show the following:

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

- By the corollary to the first Theorem, each of the sets is P-reducible to {DNF tautologies}.
- Obviously, {DNF tautologies} is P-reducible to {tautologies}.
- It remains to show the following:
  - {DNF tautologies} is P-reducible to D_3.

Main Results

Theorem

The following sets are P-reducible to each other in pairs (and hence they have the same polynomial degree of difficulty): $\{tautologies\}, \{DNF\ tautologies\}, D_3, \{subgraph\ pairs\}$.

Steps of the proof:

- By the corollary to the first Theorem, each of the sets is P-reducible to {DNF tautologies}.
- Obviously, {DNF tautologies} is P-reducible to {tautologies}.
- It remains to show the following:
  - {DNF tautologies} is P-reducible to D_3.
  - D_3 is P-reducible to {subgraph pairs}.

Discussion

- The above results at the time seemed to suggest that the sets we were examining are difficult to recognize.

Discussion

- The above results at the time seemed to suggest that the sets we were examining are difficult to recognize.
- In fact, they seemed to suggest that searching for a polynomial algorithm may be fruitless.

Discussion

- The above results at the time seemed to suggest that the sets we were examining are difficult to recognize.
- In fact, they seemed to suggest that searching for a polynomial algorithm may be fruitless.
- Of course, this concept of difficulty is what we now know as NP-hardness.

Discussion

- The above results at the time seemed to suggest that the sets we were examining are difficult to recognize.
- In fact, they seemed to suggest that searching for a polynomial algorithm may be fruitless.
- Of course, this concept of difficulty is what we now know as NP-hardness.
- It was also noted that it had not been possible up to then to add {isomorphic graphpairs} and {primes} to the list of the above Theorem.

Extensions to the Predicate Calculus

- We can extend our notation, by including symbols for the universal and existential quantifiers.

Extensions to the Predicate Calculus

- We can extend our notation, by including symbols for the universal and existential quantifiers.
- We can also accommodate infinite predicate and function symbols, as we did with infinite variables.

Extensions to the Predicate Calculus

- We can extend our notation, by including symbols for the universal and existential quantifiers.
- We can also accommodate infinite predicate and function symbols, as we did with infinite variables.
- Our alphabet is still finite.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.
- However, we want to consider processes which operate on formulas of the predicate calculus and terminate iff their input is unsatisfiable.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.
- However, we want to consider processes which operate on formulas of the predicate calculus and terminate iff their input is unsatisfiable.
- We can't have a recursive function as an upper bound for the termination times of such a process.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.
- However, we want to consider processes which operate on formulas of the predicate calculus and terminate iff their input is unsatisfiable.
- We can't have a recursive function as an upper bound for the termination times of such a process.
- The Herbrand Theorem states briefly that a formula A is unsatisfiable iff some conjunction of substitution instances of the functional form $fn(A)$ of A is truth functionally inconsistent.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.
- However, we want to consider processes which operate on formulas of the predicate calculus and terminate iff their input is unsatisfiable.
- We can't have a recursive function as an upper bound for the termination times of such a process.
- The Herbrand Theorem states briefly that a formula A is unsatisfiable iff some conjunction of substitution instances of the functional form $fn(A)$ of A is truth functionally inconsistent.
- We can make a natural ordering of these substitution instances and simply check ever-increasing in size conjunctions of such substitution instances.

Extensions to the Predicate Calculus

- Satisfiability in the predicate calculus is undecidable.
- However, we want to consider processes which operate on formulas of the predicate calculus and terminate iff their input is unsatisfiable.
- We can't have a recursive function as an upper bound for the termination times of such a process.
- The Herbrand Theorem states briefly that a formula A is unsatisfiable iff some conjunction of substitution instances of the functional form $fn(A)$ of A is truth functionally inconsistent.
- We can make a natural ordering of these substitution instances and simply check ever-increasing in size conjunctions of such substitution instances.
- If we ever get one that is truth functionally inconsistent, we terminate.

Extensions to the Predicate Calculus

We can order the substitution instances $A_1, A_2, \ldots$. Then, we have the following definition:

Extensions to the Predicate Calculus

We can order the substitution instances $A_1, A_2, \ldots$. Then, we have the following definition:

Definition

If A is unsatisfiable, then $\phi(A)$ is the least k such that $A_1 \wedge A_2 \wedge \ldots \wedge A_k$ is truth-functionally inconsistent. If A is satisfiable, then $\phi(A)$ is undefined.

Efficiency of Theorem Proving Procedures

- If we call a process that operates as described previously Q, then there is a recursive $T(k)$ such that for all k and all formulas A, if the length of A is at most k and $\phi(A) \leq k$, then Q will terminate within $T(k)$ steps.

Efficiency of Theorem Proving Procedures

- If we call a process that operates as described previously Q, then there is a recursive $T(k)$ such that for all k and all formulas A, if the length of A is at most k and $\phi(A) \leq k$, then Q will terminate within $T(k)$ steps.
- As a result, $T(k)$ is a proposed as a measure of the efficiency of Q.

Efficiency of Theorem Proving Procedures

Definition

Given a machine M_Q and recursive function $T_Q(k)$, we will say M_Q is of type Q and runs within time $T_Q(k)$ provided that, when M_Q starts with a predicate formula A written on its tape, M_Q halts if and only if A is unsatisfiable and for all k, if $\phi(A) \leq k$ and $|A| \leq \log_2 k$, then M_Q halts within $T_Q(k)$ steps. In this case we will also say that $T_Q(k)$ is of type Q. Here $|A|$ is the length of A.

Efficiency of Theorem Proving Procedures

Theorem

A) *For any* $T_Q(k)$ *of type* Q*,*

$$\frac{T_Q(k)}{\frac{\sqrt{k}}{\log^2 k}}$$

is unbounded.

B) *There is a* $T_Q(k)$ *of type* Q*, such that*

$$T_Q(k) \leq k2^{k \log^2 k}.$$

Efficiency of Theorem Proving Procedures

Theorem

If a set S of strings is accepted by a nondeterministic machine within time $T(n) = 2^n$ and if $T_Q(k)$ is an honest (i.e. real-time countable) function of type Q, then there is a constant K, such that S can be recognized by a deterministic machine within time $T_Q(K8^n)$.

References

Stephen A. Cook. The Complexity of Theorem-Proving Procedures. Paper presented at the meeting of the STOC, 1971.

Thank you!

Thank you for your time!