The $\emptyset$-Problem and the Σ^*-Problem

May 2, 2024

Goals

For all of the classes of problems studied this semester we ask the following two questions

Goals

For all of the classes of problems studied this semester we ask the following two questions

1. Is the **$\emptyset$-problems solvable?**
   (Given M determine if $L(M) = \emptyset$.)

Goals

For all of the classes of problems studied this semester we ask the following two questions

1. Is the **$\emptyset$-problems solvable?**
   (Given M determine if $L(M) = \emptyset$.)
2. Is the **Σ^*-problems solvable?**
   (Given M determine if $L(M) = \Sigma^*$.)

Goals

For all of the classes of problems studied this semester we ask the following two questions

1. Is the **$\emptyset$-problems solvable?**
   (Given M determine if $L(M) = \emptyset$.)
2. Is the **Σ^*-problems solvable?**
   (Given M determine if $L(M) = \Sigma^*$.)

We will also look at the complexity of these problems.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Given a regex α, can we tell if $L(\alpha) = \emptyset$?

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Given a regex α, can we tell if $L(\alpha) = \emptyset$?
Yes Convert to an NFA M and test $L(M) = \emptyset$. Complexity Linear.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Given a regex α, can we tell if $L(\alpha) = \emptyset$?
Yes Convert to an NFA M and test $L(M) = \emptyset$. Complexity Linear.

Caveat Might be easier.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Given a regex α, can we tell if $L(\alpha) = \emptyset$?
Yes Convert to an NFA M and test $L(M) = \emptyset$. Complexity Linear.

Caveat Might be easier.
The only way for $L(\alpha) = \emptyset$ is if $\emptyset$ is in it.

Regular Language: $\emptyset$

Given a DFA M, can we tell if $L(M) = \emptyset$?
Yes Det. if there is a path from **the** start state to **some** final state.
Complexity This is reachability prob. Linear in numb. of states.

Given a NFA M, can we tell if $L(M) = \emptyset$?
Yes Same exact algorithm, so still linear time.

Given a regex α, can we tell if $L(\alpha) = \emptyset$?
Yes Convert to an NFA M and test $L(M) = \emptyset$. Complexity Linear.

Caveat Might be easier.
The only way for $L(\alpha) = \emptyset$ is if $\emptyset$ is in it.
Alg Scan for $\emptyset$ and simplify expression. See if have $\emptyset$ in the end.

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?
Yes Convert to DFA and Solve. 2^n time. Better?

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?
Yes Convert to DFA and Solve. 2^n time. Better?
Vote Known-Poly, Known-NPC, Known-harder, Unknown!

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?
Yes Convert to DFA and Solve. 2^n time. Better?
Vote Known-Poly, Known-NPC, Known-harder, Unknown!
Complexity PSPACE-complete. So likely harder than NP.

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?
Yes Convert to DFA and Solve. 2^n time. Better?
Vote Known-Poly, Known-NPC, Known-harder, Unknown!
Complexity PSPACE-complete. So likely harder than NP.

Given a regex α, can we tell if $L(\alpha) = \Sigma^*$?

Regular Language: Σ^*

Given a DFA M, can we tell if $L(M) = \Sigma^*$?
Yes Complement and Solve $\emptyset$ problem.
Complexity Linear.

Given a NFA M, can we tell if $L(M) = \Sigma^*$?
Yes Convert to DFA and Solve. 2^n time. Better?
Vote Known-Poly, Known-NPC, Known-harder, Unknown!
Complexity PSPACE-complete. So likely harder than NP.

Given a regex α, can we tell if $L(\alpha) = \Sigma^*$?
Yes Convert to an NFA M and test $L(M) = \emptyset$. PSPACE-complete.

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC
3. Known: Decidable but likely harder than NP

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC
3. Known: Decidable but likely harder than NP
4. Known: Decidable and known to be harder than NP

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC
3. Known: Decidable but likely harder than NP
4. Known: Decidable and known to be harder than NP
5. Known: Undecidable

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC
3. Known: Decidable but likely harder than NP
4. Known: Decidable and known to be harder than NP
5. Known: Undecidable
6. Unknown to Science!

CFL: $\emptyset$

Given a CFG G in Chomsky Normal Form, can we tell if $L(G) = \emptyset$?
We know $L(G) = \Sigma^*$ is undecidable.

So is $L(G) = \emptyset$ decidable?

Vote

1. Known: P
2. Known: NPC
3. Known: Decidable but likely harder than NP
4. Known: Decidable and known to be harder than NP
5. Known: Undecidable
6. Unknown to Science!

Answer on Next Page.

Known: P

Known: P

1. Input G a CFG in Chomsky Normal Form.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \to \sigma$, **mark** A.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \rightarrow \sigma$, **mark** A.
4. For all rules of the form $A \rightarrow BC$ if B, C are marked, then **mark** A.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \rightarrow \sigma$, **mark** A.
4. For all rules of the form $A \rightarrow BC$ if B, C are marked, then **mark** A.
    4.1 If S is marked then output YES and HALT.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \rightarrow \sigma$, **mark** A.
4. For all rules of the form $A \rightarrow BC$ if B, C are marked, then **mark** A.
   4.1 If S is marked then output YES and HALT.
   4.2 If no new nonterms are marked then output NO and HALT.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \rightarrow \sigma$, **mark** A.
4. For all rules of the form $A \rightarrow BC$ if B, C are marked, then **mark** A.
    4.1 If S is marked then output YES and HALT.
    4.2 If no new nonterms are marked then output NO and HALT.
    4.3 If a new nonterm is marked but its not S then repeat Step 4.

Known: P

1. Input G a CFG in Chomsky Normal Form.
2. Our algorithm will **mark** every nonterm that generates some string. At the end we see if S is marked.
3. For all rules of the form $A \rightarrow \sigma$, **mark** A.
4. For all rules of the form $A \rightarrow BC$ if B, C are marked, then **mark** A.
   4.1 If S is marked then output YES and HALT.
   4.2 If no new nonterms are marked then output NO and HALT.
   4.3 If a new nonterm is marked but its not S then repeat Step 4.

Numb of iterations $\leq$ numb of nonterms, so alg is poly time.

CFL: Σ^*

As shown in a prior lecture, the problem of given a CFL does it equal Σ^* is undecidable.

CFL: Σ^*

As shown in a prior lecture, the problem of given a CFL does it equal Σ^* is undecidable.

The key step was an algorithm for the following:

CFL: Σ^*

As shown in a prior lecture, the problem of given a CFL does it equal Σ^* is undecidable.

The key step was an algorithm for the following:

Given (e, x) output a CFL for $\overline{\mathrm{ACC}_{e,x}}$.

P: $\emptyset$

We need a way to represent languages in P.

P: $\emptyset$

We need a way to represent languages in P.

We take TM along with a poly p.

P: $\emptyset$

We need a way to represent languages in P.

We take TM along with a poly p.

$L(M, p) = \{x : M(x) \downarrow = 1 \text{ within time } p(|x|)\ \}$

P: $\emptyset$

We need a way to represent languages in P.

We take TM along with a poly p.

$L(M, p) = \{x : M(x) \downarrow = 1 \text{ within time } p(|x|) \}$

Is the following decidable: Given M, p, is $L(M, p) = \emptyset$?

P: $\emptyset$

We need a way to represent languages in P.

We take TM along with a poly p.

$L(M, p) = \{x : M(x) \downarrow = 1 \text{ within time } p(|x|) \}$

Is the following decidable: Given M, p, is $L(M, p) = \emptyset$?

No.

P: $\emptyset$

We need a way to represent languages in P.

We take TM along with a poly p.

$L(M,p) = \{x : M(x) \downarrow = 1 \text{ within time } p(|x|)\ \}$

Is the following decidable: Given M, p, is $L(M,p) = \emptyset$?

No.

We give two proofs.

Proof 1: Use CFG-$\emptyset$

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M, p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M, p) \neq \emptyset$.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M,p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M,p) \neq \emptyset$.
4. Using ALG find if $L(M,p) = \emptyset$.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M,p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M,p) \neq \emptyset$.
4. Using ALG find if $L(M,p) = \emptyset$.
   If YES then output YES.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M, p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M, p) \neq \emptyset$.
4. Using ALG find if $L(M, p) = \emptyset$.
   If YES then output YES.
   If NO then output NO.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M,p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M,p) \neq \emptyset$.
4. Using ALG find if $L(M,p) = \emptyset$.
   If YES then output YES.
   If NO then output NO.

Interesting since we are NOT using HALT.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M,p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M,p) \neq \emptyset$.
4. Using ALG find if $L(M,p) = \emptyset$.
   If YES then output YES.
   If NO then output NO.

Interesting since we are NOT using HALT.
Contrast:

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M, p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M, p) \neq \emptyset$.
4. Using ALG find if $L(M, p) = \emptyset$.
   If YES then output YES.
   If NO then output NO.

Interesting since we are NOT using HALT.
Contrast:
Common to **not** use SAT to show a problem NPC.

Proof 1: Use CFG-$\emptyset$

1. Input CFG G. (We want to know if $L(G) = \Sigma^*$.)
2. Create Poly time TM M for $\overline{L(G)}$.
   (The Dynamic Programming Algorithm from CYK, but at the end reverse the answers.)
   Let p be its run time.
3. **Comment**
   $L(G) = \Sigma^* \rightarrow \overline{L(G)} = \emptyset \rightarrow L(M, p) = \emptyset$.
   $L(G) \neq \Sigma^* \rightarrow \overline{L(G)} \neq \emptyset \rightarrow L(M, p) \neq \emptyset$.
4. Using ALG find if $L(M, p) = \emptyset$.
   If YES then output YES.
   If NO then output NO.

Interesting since we are NOT using HALT.
Contrast:
Common to **not** use SAT to show a problem NPC.
Uncommon to **not** use HALT to show a problem UNDEC.

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.

Proof 2: Use HALT

Assume, BWOC, that $L(M,p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

Proof 2: Use HALT

Assume, BWOC, that $L(M,p)=\emptyset$ problem is DEC.
We also take $\Sigma=\{1\}$.
We show that HALT is DEC.

1. Input e,x. (We want to know if $M_e(x)\downarrow$.)

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $O(s \log s) \le s^2$ time.)

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $O(s \log s) \leq s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output YES. If not then output NO.

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $O(s \log s) \leq s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output YES. If not then output NO.
3. Test if $L(M, n^2) = \emptyset$.

Proof 2: Use HALT

Assume, BWOC, that $L(M, p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $O(s \log s) \leq s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output YES. If not then output NO.
3. Test if $L(M, n^2) = \emptyset$.
   3.1 If $L(M, n^2) = \emptyset$ then output NO ($M_e(x) \uparrow$).

Proof 2: Use HALT

Assume, BWOC, that $L(M,p) = \emptyset$ problem is DEC.
We also take $\Sigma = \{1\}$.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x)\downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1\cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $O(s\log s) \le s^2$ time.)
   2.3 If $M_{e,s}(x)\downarrow$ then output YES. If not then output NO.
3. Test if $L(M,n^2) = \emptyset$.
   3.1 If $L(M,n^2) = \emptyset$ then output NO ($M_e(x)\uparrow$).
   3.2 If $L(M,n^2) \neq \emptyset$ then output YES ($M_e(x)\downarrow$).

P: Σ^*

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

P: Σ^*

Is the following decidable: Given M, p, is $L(M,p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M,p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)

P: Σ^*

Is the following decidable: Given M, p, is $L(M,p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M,p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x)\downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $s \log s \leq s^2$ time.)

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $s \log s \le s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output NO. If not then output YES.

P: Σ^*

Is the following decidable: Given M, p, is $L(M,p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M,p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x)\downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $s \log s \le s^2$ time.)
   2.3 If $M_{e,s}(x)\downarrow$ then output NO. If not then output YES.
3. Test if $L(M, n^2) = \Sigma^*$.

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $s \log s \le s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output NO. If not then output YES.
3. Test if $L(M, n^2) = \Sigma^*$.
   3.1 If $L(M, n^2) = \Sigma^*$ then output NO ($M_e(x) \uparrow$).

P: Σ^*

Is the following decidable: Given M, p, is $L(M, p) = \Sigma^*$?
No.
Assume, BWOC, that $L(M, p) = \Sigma^*$ is DEC.
We show that HALT is DEC.

1. Input e, x. (We want to know if $M_e(x) \downarrow$.)
2. Create a machine M and poly p as follows:
   2.1 Input 1^s (This is $1 \cdots 1$.)
   2.2 Run $M_e(x)$ for s steps.
   (For technical reasons this take $s \log s \le s^2$ time.)
   2.3 If $M_{e,s}(x) \downarrow$ then output NO. If not then output YES.
3. Test if $L(M, n^2) = \Sigma^*$.
   3.1 If $L(M, n^2) = \Sigma^*$ then output NO ($M_e(x) \uparrow$).
   3.2 If $L(M, n^2) \neq \Sigma^*$ then output YES ($M_e(x) \downarrow$).

NP, DEC, Σ_1

NP, DEC, Σ_1

For NP, DEC, Σ_1 the $L(M) = \emptyset$ is undecidable.

NP, DEC, Σ_1

For NP, DEC, Σ_1 the $L(M) = \emptyset$ is undecidable.

For NP, DEC, Σ_1 the $L(M) = \Sigma^*$ is undecidable.

NP, DEC, Σ_1

For NP, DEC, Σ_1 the $L(M) = \emptyset$ is undecidable.

For NP, DEC, Σ_1 the $L(M) = \Sigma^*$ is undecidable.

Proofs are similar to that for P.

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

   Interesting Very few problems about DFAs are NP-complete.

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

   Interesting Very few problems about DFAs are NP-complete.

2. For CFG this problem is PSPACE-hard but **Open** as to whether or not its in PSPACE .

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

   Interesting Very few problems about DFAs are NP-complete.

2. For CFG this problem is PSPACE-hard but **Open** as to whether or not its in PSPACE .

   Interesting Very few problems about CFG's are open.

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

   Interesting Very few problems about DFAs are NP-complete.

2. For CFG this problem is PSPACE-hard but **Open** as to whether or not its in PSPACE .

   Interesting Very few problems about CFG's are open.

   For P, NP, DEC, Σ_1, Undecidable.

The Σ^n Problem

Given M does there exist n such that $\Sigma^n \subseteq L(M)$?

1. For DFA this problem is NP-complete.

   Interesting Very few problems about DFAs are NP-complete.

2. For CFG this problem is PSPACE-hard but **Open** as to whether or not its in PSPACE .

   Interesting Very few problems about CFG's are open.

   For P, NP, DEC, Σ_1, Undecidable.

   Boring Most problems about P, NP, DEC, Σ_1 are undecidable.