CS310 Automata Theory – 2016-2017

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Lecture 30: Turing machines, computability
March 30, 2017

At the end of last class

Introduction to Turing machines

- Undecidability of the following languages:
  - $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$.
  - $\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$.
  - $E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$.
  - $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$.
  - $REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$.
- Note that undecidability of REG_{TM} and E_{TM} can be proved using Rice's theorem.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

We say that a property P is trivial if either $\mathcal{L}_P = \varnothing$ or $\mathcal{L}_P$ is the set of all the Turing recognizable languages.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

We say that a property P is trivial if either $\mathcal{L}_P = \varnothing$ or $\mathcal{L}_P$ is the set of all the Turing recognizable languages.

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

We say that a property P is trivial if either $\mathcal{L}_P = \emptyset$ or $\mathcal{L}_P$ is the set of all the Turing recognizable languages.

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Property P and $\mathcal{L}_P$

Rice's theorem: A systematic way of proving undecidability of languages.

Definition

A property P is simply a subset of Turing recognizable languages. We say that a language L satisfies a property P, if $L \in P$.

For any property P, let $\mathcal{L}_P = \{M \mid L(M) \in P\}$, i.e. the set of all Turing machine such that $L(M) \in P$.

We say that a property P is trivial if either $\mathcal{L}_P = \varnothing$ or $\mathcal{L}_P$ is the set of all the Turing recognizable languages.

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that
$\mathcal{L}_P = \{M \mid L(M) \in P\}$.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M \rangle \mid M$ has at least ten states$\}$.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M \rangle \mid M$ has at least ten states$\}$.

$\{\langle M \rangle \mid M$ never moves left on any input string $\}$.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M \rangle \mid M$ has at least ten states$\}$.

$\{\langle M \rangle \mid M$ never moves left on any input string $\}$.

$\{\langle M \rangle \mid M$ has no useless state $\}$.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M\rangle \mid M$ has at least ten states$\}$.

$\{\langle M\rangle \mid M$ never moves left on any input string $\}$.

$\{\langle M\rangle \mid M$ has no useless state $\}$.

To prove non-recognizability of a property of languages.

Rice's theorem cannot be used to prove non-recognizability of languages.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M\rangle \mid M$ has at least ten states$\}$.

$\{\langle M\rangle \mid M$ never moves left on any input string $\}$.

$\{\langle M\rangle \mid M$ has no useless state $\}$.

To prove non-recognizability of a property of languages.

Rice's theorem cannot be used to prove non-recognizability of languages.

It is only used to prove undecidability.

Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

When is the theorem NOT applicable?

Rice's theorem cannot be used to prove undecidability of languages which deal with machine behaviour.

$\{\langle M\rangle \mid M$ has at least ten states$\}$.

$\{\langle M\rangle \mid M$ never moves left on any input string $\}$.

$\{\langle M\rangle \mid M$ has no useless state $\}$.

To prove non-recognizability of a property of languages.

Rice's theorem cannot be used to prove non-recognizability of languages.

It is only used to prove undecidability.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M\rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M \rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

$\{\langle M \rangle \mid L(M)$ is recognized a TM with at least 10 states$\}$.

Applicable, but property is trivial.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M\rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

$\{\langle M\rangle \mid L(M)$ is recognized a TM with at least 10 states$\}$.

Applicable, but property is trivial.

$\{\langle M\rangle \mid L(M)$ is recognized a TM with atmost 10 states$\}$.

Applicable and property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M \rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

$\{\langle M \rangle \mid L(M)$ is recognized a TM with at least 10 states$\}$.

Applicable, but property is trivial.

$\{\langle M \rangle \mid L(M)$ is recognized a TM with atmost 10 states$\}$.

Applicable and property is not trivial, therefore undecidable.

$\{\langle M \rangle \mid M$ has at most 10 states$\}$.

Not applicable, but the language is decidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M\rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

$\{\langle M\rangle \mid L(M)$ is recognized a TM with at least 10 states$\}$.

Applicable, but property is trivial.

$\{\langle M\rangle \mid L(M)$ is recognized a TM with atmost 10 states$\}$.

Applicable and property is not trivial, therefore undecidable.

$\{\langle M\rangle \mid M$ has at most 10 states$\}$.

Not applicable, but the language is decidable.

$\{M \mid L(M)$ contains $\langle M\rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

$\{M \mid L(M)$ is finite $\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{\langle M \rangle \mid M$ runs for atmost 10 steps on $aab\}$.

Not applicable, but language decidable.

$\{\langle M \rangle \mid L(M)$ is recognized a TM with at least 10 states$\}$.

Applicable, but property is trivial.

$\{\langle M \rangle \mid L(M)$ is recognized a TM with atmost 10 states$\}$.

Applicable and property is not trivial, therefore undecidable.

$\{\langle M \rangle \mid M$ has at most 10 states$\}$.

Not applicable, but the language is decidable.

$\{M \mid L(M)$ contains $\langle M \rangle\}$.

Applicable, the property is not trivial, therefore undecidable.

$\{M \mid L(M)$ is finite $\}$.

Applicable, the property is not trivial, therefore undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{M \mid M \text{ has a useless state }\}.$$

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid M \text{ has a useless state } \}$.

Not applicable, but the language is in fact undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$\{M \mid M \text{ has a useless state }\}$.

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{(M, w) \mid M \text{ writes a symbol a on the tape on input } w\}.$$

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{(M, w) \mid M \text{ writes a symbol a on the tape on input } w\}.$$

Not applicable, but the language is in fact undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\{(M, w) \mid M \text{ writes a symbol a on the tape on input } w\}.$$

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\left\{ M \;\middle|\; \begin{array}{l} M \text{ tries to write on the left of the cell when it} \\ \text{is at the leftmost bit of the input} \end{array} \right\}.$$

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\left\{ M \,\middle|\, \begin{array}{l} M \text{ tries to write on the left of the cell when it} \\ \text{ is at the leftmost bit of the input} \end{array} \right\}.$$

Not applicable, but the language is in fact undecidable.

Applications of Rice's theorem

We now learn how to apply Rice's theorem

$$\left\{ M \;\middle|\; \begin{array}{l} M \text{ tries to write on the left of the cell when it} \\ \text{is at the leftmost bit of the input} \end{array} \right\}.$$

Not applicable, but the language is in fact undecidable.

Rice's theorem cannot be used to prove the undecidability of this language!

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$.

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Using this assumption prove that A_{TM} is decidable.

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Using this assumption prove that A_{TM} is decidable.

More specifically:

$$(M, w) \qquad \longrightarrow \quad N$$

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Using this assumption prove that A_{TM} is decidable.

More specifically:

$$(M, w) \longrightarrow N$$

$$\text{if } w \in L(M) \longrightarrow \langle N \rangle \in \mathcal{L}_P$$

Proof of Rice's theorem

Theorem

Let P be a property such that it is not trivial. Recall that $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof Idea:

Let P be a non-trivial property.

Assume that $\mathcal{L}_P$ is decidable.

Using this assumption prove that A_{TM} is decidable.

More specifically:

$$(M, w) \longrightarrow N$$

$$\begin{array}{lcl} \text{if } w \in L(M) & \longrightarrow & \langle N \rangle \in \mathcal{L}_P \\ \text{if } w \notin L(M) & \longrightarrow & \langle N \rangle \notin \mathcal{L}_P \end{array}$$

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \emptyset$.

we assume that $\emptyset$ does not have property P

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

we assume that $\varnothing$ does not have property P[1]

[1] We will remove this assumption later.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

we assume that $\varnothing$ does not have property P

on input x
{
    if M accepts w
    then if M_1 accepts x
        then accept
}

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

we assume that $\varnothing$ does not have property P

on input x
{
    if M accepts w
    then if M_1 accepts x
        then accept
}

Claim: $w \in L(M)$ if and only if $\langle N \rangle \in \mathcal{L}_P$

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \emptyset$.

we assume that $\emptyset$ does not have property P

```
on input x
{
    if M accepts w
    then if M1 accepts x
        then accept
}
```

Claim: $w \in L(M)$ if and only if $\langle N \rangle \in \mathcal{L}_P$

if $w \in L(M)$ then $L(N) = L(M_1)$.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \varnothing$.

we assume that $\varnothing$ does not have property P

```
on input x
{
    if M accepts w
    then if M_1 accepts x
        then accept
}
```

Claim: $w \in L(M)$ if and only if $\langle N \rangle \in \mathcal{L}_P$

if $w \in L(M)$ then $L(N) = L(M_1)$.

if $w \notin L(M)$ then $L(N) = \varnothing$.

Proof of Rice's theorem

Theorem

Let P be a non-trivial property of Turing recognizable languages. Let $\mathcal{L}_P = \{M \mid L(M) \in P\}$. Then $\mathcal{L}_P$ is undecidable.

Design of N

Let M_1 be the TM s.t. $L(M_1)$ has Property P.

Let $L(M_2)$ be the TM s.t. $L(M_2) = \emptyset$.

we assume that $\emptyset$ does not have property P

```
on input x
{
    if M accepts w
    then if M1 accepts x
        then accept
}
```

Claim: $w \in L(M)$ if and only if $\langle N \rangle \in \mathcal{L}_P$

if $w \in L(M)$ then $L(N) = L(M_1)$.

if $w \notin L(M)$ then $L(N) = \emptyset$.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Consider $\overline{P}$.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Consider $\overline{P}$.

Now $\varnothing$ does not have property $\overline{P}$.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Consider $\overline{P}$.

Now $\varnothing$ does not have property $\overline{P}$.

Use Rice's theorem on $\mathcal{L}_{\overline{P}}$ to prove undecidibility.

Getting rid of the assumption on P

We now show how to get around the assumption.

Suppose $\varnothing$ has property P.

Consider $\overline{P}$.

Now $\varnothing$ does not have property $\overline{P}$.

Use Rice's theorem on $\mathcal{L}_{\overline{P}}$ to prove undecidibility.

Conclude undecidibility of $\mathcal{L}_P$.

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$ *is undecidable.*

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*)

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M\rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$ *is undecidable.*

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

Input $(M, w) \quad \longrightarrow \quad N_{M,w}$

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M \rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

$$\text{Input } (M, w) \quad \longrightarrow \quad N_{M,w}$$

$$\text{if } w \in L(M) \quad \longrightarrow \quad \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w})$$

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M\rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Universality of CFLs

Lemma

$ALL_{CFL} = \{\langle M\rangle \mid M$ is a PDA and $L(M) = \Sigma^*\}$ is undecidable.

Proof Strategy

For a TM M and input w we create a PDA $N_{M,w}$ such that

$N_{M,w}$ accepts all string (i.e. accepts Σ^*) if M accepts w, and

$N_{M,w}$ rejects at least one string if M does not accept w.

Formally,

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Filling in the details

The following two details need to be addressed.

Q_1 How should we design $N_{M,w}$?

Filling in the details

The following two details need to be addressed.

Q_1 How should we design $N_{M,w}$?

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\text{Input } (M, w) \quad \longrightarrow \quad N_{M,w}$$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \end{array}$$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Assume that ALL_{CFL} is decidable.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*$, s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Design A as follows:

For an M, w pair, create $N_{M,w}$.

$$\begin{array}{lcl} \text{Input } (M, w) & \longrightarrow & N_{M,w} \\ \text{if } w \in L(M) & \longrightarrow & \exists x \in \Sigma^*, \text{ s.t. } x \notin L(N_{M,w}) \\ \text{if } w \notin L(M) & \longrightarrow & L(N_{M,w}) = \Sigma^* \end{array}$$

Feed $\langle N_{M,w} \rangle$ to C.

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*$, s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*$, s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

else accept.

Details for Q_2

Q_2 If such an $N_{M,w}$ is designed then why have we proved that ALL_{CFL} is undecidable?

Input (M, w)	$\longrightarrow$	$N_{M,w}$
if $w \in L(M)$	$\longrightarrow$	$\exists x \in \Sigma^*,$ s.t. $x \notin L(N_{M,w})$
if $w \notin L(M)$	$\longrightarrow$	$L(N_{M,w}) = \Sigma^*$

Assume that ALL_{CFL} is decidable.
C be the TM deciding it.

Design A as follows:

For an M, w pair, create $N_{M,w}$.

Feed $\langle N_{M,w} \rangle$ to C.

If C accepts then reject;

else accept.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

C_1 is a start configutation.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is a rejecting configuration.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is a rejecting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Details for Q_1: reduction via computation history

Q_1 How should we design $N_{M,w}$?

Main idea

Use computational history of M on w.

Accepting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is an accepting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Rejecting computation history is a sequece of configutations: $C_1, C_2, \ldots, C_\ell$ such that

- C_1 is a start configutation.
- C_ℓ is a rejecting configuration.
- for each $1 \leq i \leq \ell$, C_i yields C_{i+1}.

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x OR
- x is a computational history, but C_1 is not a start configuration

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x OR
- x is a computational history, but C_1 is not a start configuration OR
- x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

x does not have the pattern of a computational history of x OR

x is a computational history, but C_1 is not a start configuration OR

x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR

x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x OR
- x is a computational history, but C_1 is not a start configuration OR
- x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
- x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \leq i \leq \ell - 1$ and C_i does not yield C_{i+1}.

If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

- $N_{M,w}$ will reject $\tilde{x}$

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

x does not have the pattern of a computational history of x OR

x is a computational history, but C_1 is not a start configuration OR

x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR

x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \le i \le \ell - 1$ and C_i does not yield C_{i+1}.

If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

$N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x OR
- x is a computational history, but C_1 is not a start configuration OR
- x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
- x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \le i \le \ell - 1$ and C_i does not yield C_{i+1}.

If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.

$N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.

If M does not accept w, then

no matter what x is, $N_{M,w}$ will accept x

Details for Q_1: reduction via computation history

Interprete input x to $N_{M,w}$ as a computational history of M on w.

Design $N_{M,w}$ s.t. it accepts x if any of the following conditions holds:

- x does not have the pattern of a computational history of x OR
- x is a computational history, but C_1 is not a start configuration OR
- x is a computational history, C_1 is a start configuration, but C_ℓ is not an accepting configutation OR
- x is a computational history, C_1 is a start configuration, C_ℓ is an accepting configutation, but there exists an i s.t. $1 \le i \le \ell - 1$ and C_i does not yield C_{i+1}.

If M accepts w, let $\tilde{x}$ be a accepting computation history of M on w.
 $N_{M,w}$ will reject $\tilde{x}$, i.e. $\tilde{x} \notin L(N_{M,w})$.

If M does not accept w, then
 no matter what x is, $N_{M,w}$ will accept x, i.e. $L(N_{M,w}) = \Sigma^*$.