Proving Undecidability via Reductions

Alice Gao

Lecture 23

Outline

Learning Goals

By the end of this lecture, you should be able to:

- Define reduction.
- Describe at a high level how we can use reduction to prove that a decision problem is undecidable.
- Prove that a decision problem is undecidable by using a reduction from the halting problem.

Outline

Proving that other problems are undecidable

We proved that the halting problem is undecidable.

How do we prove that another problem is undecidable?

- We could prove it from scratch, or
- We could prove that it is as difficult as the halting problem. Hence, it must be undecidable.

Proving undecidability via reductions

We will prove undecidability via reductions.

Reduce the halting problem to problem P_B.

- Given an algorithm for solving P_B,
  we could use it to solve the halting problem.
- If P_B is decidable, then the halting problem is decidable.
- If the halting problem is undecidable, then P_B is undecidable.

Proving undecidability via reductions

Theorem: Problem P_B is undecidable.

Proof by Contradiction.

Assume that there is an algorithm B, which solves problem P_B.

reduction

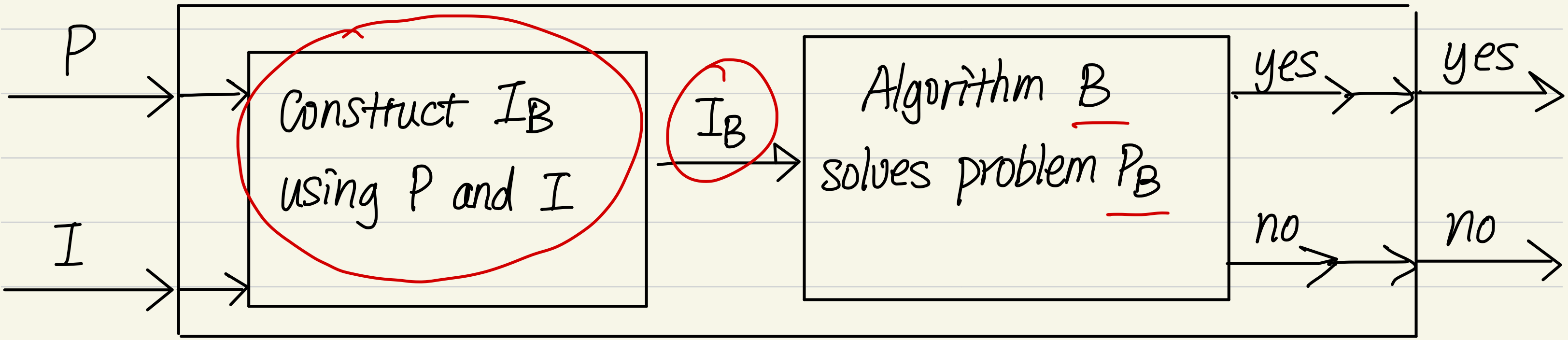
We will construct algorithm A, which uses algorithm B to solve the halting problem. (Describe algorithm A.)

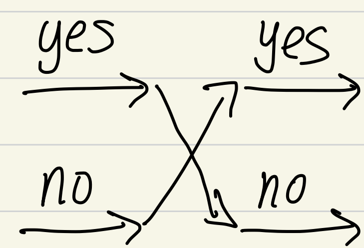
explanation

Since algorithm B solves problem P_B, algorithm A solves the halting problem, which contradicts with the fact that the halting problem is undecidable.

Therefore, problem P_B is undecidable. □

Algorithm A solves the halting problem





Outline

Example 1 of reduction proofs

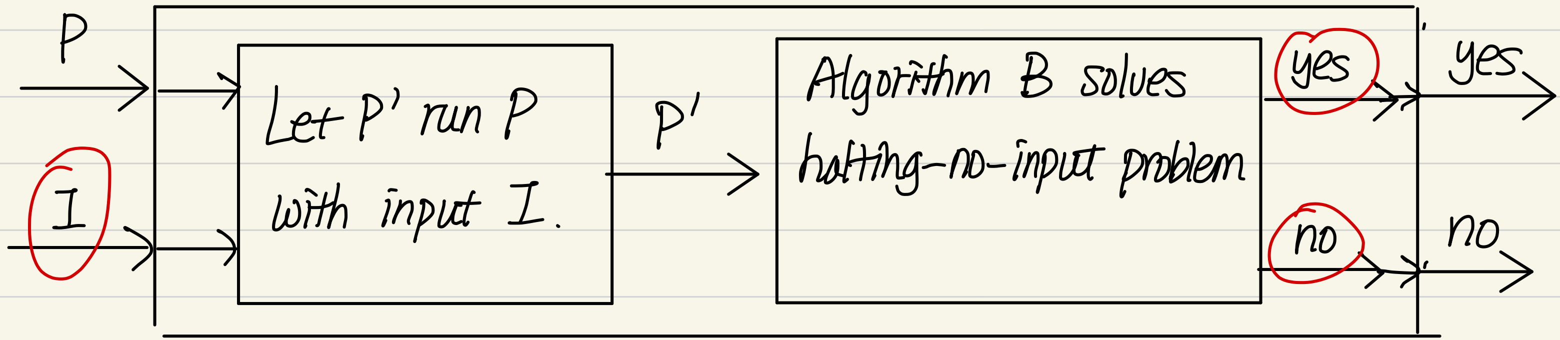
The halting-no-input problem:

Given a program P which takes no input, does P halt?

Theorem: The halting-no-input problem is undecidable.

```
P(void) {   -> no input
    ......
}
```

Algorithm A solves the halting problem



program P:

```
P( I) {
  ......
}
```

halting problem:
takes two inputs
① "P(I) { ...... }"
② I

program P':

```
P'() {
  return P(I);
}
```

algorithm B
takes one input
① "P'() { ...... }"

Proof by contradiction:

Assume that there is an algorithm B, which solves the halting-no-input problem.

We will construct an algorithm A to solve the halting problem. Algorithm A works as follows:

- A takes two inputs, a program P and an input I.
- Let program P' run P with input I and return P(I).
- Run algorithm B with P' as its input.
- Return B(P').

By our construction of algorithm A,
P halts on input I if and only if P' halts.

Since algorithm B solves the halting-no-input problem, algorithm A solves the halting problem, which contradicts the fact that the halting problem is undecidable.

Therefore, the halting-no-input problem is undecidable.

∎

Example 2 of reduction proofs

The both-halt problem:

> *Given two programs P_1 and P_2 which take no input, do both programs halt?*

Theorem: The both-halt problem is undecidable.

Example 3 of reduction proofs

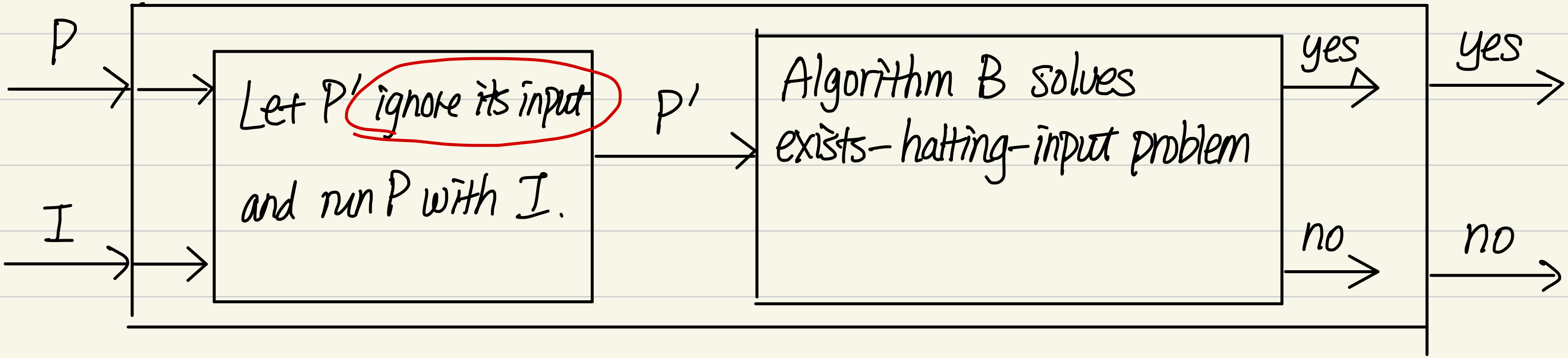
which takes one input I,

The exists-halting-input problem

> *Given a program P, does there exist an input I such that P halts with input I?*

Theorem The exists-halting-input problem is undecidable.

Algorithm A solves the halting problem



Program P

```
P(I) {
  ------
}
```

Program P'

```
P'(I') {
  return P(I);
}
```

algorithm B takes one input:

① "
```
P'(I') {
  return P(I);
}
```
"

Proof by contradiction:

Assume that there is an algorithm B, which solves the exists-halting-input problem.

We will construct an algorithm A, which solves the halting problem. Algorithm A works as follows:

- A takes two inputs, a program P and an input I.
- Let program P' ignore its input, run P with input I, and return P(I).
- Run algorithm B with P' as the input.
- Return B(P').

By our construction of algorithm A,
P halts on input I if and only if there exists an input I' such that P' halts on I'.

Since algorithm B solves the exists-halting-input problem, algorithm A solves the halting problem, which contradicts the fact that the halting problem is undecidable.

Therefore, the exists-halting-input problem is undecidable.

∎

Revisiting the learning goals

By the end of this lecture, you should be able to:

- Define reduction.
- Describe at a high level how we can use reduction to prove that a decision problem is undecidable.
- Prove that a decision problem is undecidable by using a reduction from the halting problem.