

Post's Correspondence Problem

Bas van Gijzel

May 17, 2010

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks



Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small motivation

- ▶ **Post's Correspondence Problem (PCP)** is an instance of a **recursively unsolvable decision problem**.
 - Why would we need another one?
- ▶ PCP is remarkably simple to explain, and thus an easy instance of an **undecidable problem**.
- ▶ Other problems can then be **reduced** to PCP to prove undecidability.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small motivation

- ▶ **Post's Correspondence Problem** (PCP) is an instance of a **recursively unsolvable decision problem**.
 - Why would we need another one?
- ▶ PCP is remarkably simple to explain, and thus an easy instance of an **undecidable problem**.
- ▶ Other problems can then be **reduced** to PCP to prove undecidability.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small motivation

- ▶ **Post's Correspondence Problem** (PCP) is an instance of a **recursively unsolvable decision problem**.
 - Why would we need another one?
- ▶ PCP is remarkably simple to explain, and thus an easy instance of an **undecidable problem**.
- ▶ Other problems can then be **reduced** to PCP to prove undecidability.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Emil Leon Post



Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks



Universiteit Utrecht

Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1 x_2 \dots x_m$ is equal to the corresponding $y_1 y_2 \dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1x_2\dots x_m$ is equal to the corresponding $y_1y_2\dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem

- ▶ Let's consider strings containing a's and b's.
 - For example "aaabba".
 - We don't consider empty strings.
- ▶ We have multiple pairs $(x_1, y_1), \dots, (x_n, y_n)$.
 - These pairs x_1 etc. are strings on a's and b's.
 - Again not ϵ (empty string).

The correspondence decision problem is then the problem of determining whether there is a solution where the sequence/concatenation $x_1x_2\dots x_m$ is equal to the corresponding $y_1y_2\dots y_m$. This can contain repeated pairs, miss certain pairs, differ in order.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(1)

- ▶ The problem can be formulated as a problem about matching domino tiles.
- ▶ We have multiple domino tiles with on the top the x_i strings and on the bottom the y_i strings.
- ▶ A solution to the decision problem is then a solution of matching a row of dominoes. (possibly containing doubles)

Can we match the dominoes d_1, d_2, d_3 below? We can't turn dominoes but can use dominoes multiple times.

bb	ab	b
b	ba	bb

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Introduction to the problem: intuitively(1)

- ▶ The problem can be formulated as a problem about matching domino tiles.
- ▶ We have multiple domino tiles with on the top the x_i strings and on the bottom the y_i strings.
- ▶ A solution to the decision problem is then a solution of matching a row of dominoes. (possibly containing doubles)

Can we match the dominoes d_1, d_2, d_3 below? We can't turn dominoes but can use dominoes multiple times.

bb	ab	b
b	ba	bb

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(1)

- ▶ The problem can be formulated as a problem about matching domino tiles.
- ▶ We have multiple domino tiles with on the top the x_i strings and on the bottom the y_i strings.
- ▶ A solution to the decision problem is then a solution of matching a row of dominoes. (possibly containing doubles)

Can we match the dominoes d_1, d_2, d_3 below? We can't turn dominoes but can use dominoes multiple times.

bb	ab	b
b	ba	bb

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Introduction to the problem: intuitively(1)

- ▶ The problem can be formulated as a problem about matching domino tiles.
- ▶ We have multiple domino tiles with on the top the x_i strings and on the bottom the y_i strings.
- ▶ A solution to the decision problem is then a solution of matching a row of dominoes. (possibly containing doubles)

Can we match the dominoes d_1, d_2, d_3 below? We can't turn dominoes but can use dominoes multiple times.

bb	ab	b
b	ba	bb

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(1)

- ▶ The problem can be formulated as a problem about matching domino tiles.
- ▶ We have multiple domino tiles with on the top the x_i strings and on the bottom the y_i strings.
- ▶ A solution to the decision problem is then a solution of matching a row of dominoes. (possibly containing doubles)

Can we match the dominoes d_1, d_2, d_3 below? We can't turn dominoes but can use dominoes multiple times.

bb	ab	b
b	ba	bb

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(2)

A solution to the problem would be the sequence of dominoes $d_1 d_2 d_2 d_3$.

$$\begin{array}{c|c|c|c} \text{bb} & \text{ab} & \text{ab} & \text{b} \\ \text{b} & \text{ba} & \text{ba} & \text{bb} \end{array}$$

This sequence of dominoes can be seen as the equality between the upper and lower part of the dominoes, namely

$$x_1 x_2 x_2 x_3 = y_1 y_2 y_2 y_3.$$

So we have $bbababb = bbababb$, which is a solution.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(2)

A solution to the problem would be the sequence of dominoes $d_1 d_2 d_2 d_3$.

$$\begin{array}{c|c|c|c} bb & ab & ab & b \\ \hline b & ba & ba & bb \end{array}$$

This sequence of dominoes can be seen as the equality between the upper and lower part of the dominoes, namely

$$x_1 x_2 x_2 x_3 = y_1 y_2 y_2 y_3.$$

So we have $bbababb = bbababb$, which is a solution.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Introduction to the problem: intuitively(2)

A solution to the problem would be the sequence of dominoes $d_1 d_2 d_2 d_3$.

$$\begin{array}{c|c|c|c} bb & ab & ab & b \\ \hline b & ba & ba & bb \end{array}$$

This sequence of dominoes can be seen as the equality between the upper and lower part of the dominoes, namely

$$x_1 x_2 x_2 x_3 = y_1 y_2 y_2 y_3.$$

So we have $bbababb = bbababb$, which is a solution.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(1)

Referring back to the slides of Clemens.

Suppose $A \subseteq E$ where E is a set of **concrete objects**.

*A **decision method** for A in E is a method by which, given an element $a \in E$, we can decide in a finite number of steps whether or not $a \in A$.*

*A **decision problem** for A in E : Find a decision method for A in E , or show that no such method can exist.*

*The **decision problem** for A in E is **solvable** (the set A in E is (effectively) **calculable**) if there exists a decision method for A in E .*

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(1)

Referring back to the slides of Clemens.

Suppose $A \subseteq E$ where E is a set of **concrete objects**.

A **decision method** for A in E is a method by which, given an element $a \in E$, we can decide in a finite number of steps whether or not $a \in A$.

A **decision problem** for A in E : Find a decision method for A in E , or show that no such method can exist.

The **decision problem** for A in E is **solvable** (the set A in E is (effectively) **calculable**) if there exists a decision method for A in E .

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



A small rerun on decision problems(1)

Referring back to the slides of Clemens.

Suppose $A \subseteq E$ where E is a set of **concrete objects**.

A **decision method** for A in E is a method by which, given an element $a \in E$, we can decide in a finite number of steps whether or not $a \in A$.

A **decision problem** for A in E : Find a decision method for A in E , or show that no such method can exist.

The **decision problem** for A in E is **solvable** (the set A in E is (effectively) **calculable**) if there exists a decision method for A in E .

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



A small rerun on decision problems(1)

Referring back to the slides of Clemens.

Suppose $A \subseteq E$ where E is a set of **concrete objects**.

A **decision method** for A in E is a method by which, given an element $a \in E$, we can decide in a finite number of steps whether or not $a \in A$.

A **decision problem** for A in E : Find a decision method for A in E , or show that no such method can exist.

The **decision problem** for A in E is **solvable** (the set A in E is **(effectively) calculable**) if there exists a decision method for A in E .

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



A small rerun on decision problems(2)

- ▶ First we consider a simple version of PCP which *is* calculable.
- ▶ Then we consider the full undecidable problem.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(3): The simplified problem.

- ▶ Assume we are given a sequence of dominoes.
- ▶ Decide whether the top and bottom describe the same string.

Consider all possible finite sequences of dominoes $d_1 d_2 \dots d_n$ and call this E . The subset of E for which it holds that $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m$ is the set of solutions, called A . To solve a decision problem we would need a decision procedure.

Here a decision procedure would just test equality between pair components of the domino sequence, resulting in a yes/no answer. This obviously takes finite steps, because the sequences are finite.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(3): The simplified problem.

- ▶ Assume we are given a sequence of dominoes.
- ▶ Decide whether the top and bottom describe the same string.

Consider all possible finite sequences of dominoes $d_1 d_2 \dots d_n$ and call this E . The subset of E for which it holds that $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m$ is the set of solutions, called A . To solve a decision problem we would need a decision procedure.

Here a decision procedure would just test equality between pair components of the domino sequence, resulting in a yes/no answer. This obviously takes finite steps, because the sequences are finite.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(3): The simplified problem.

- ▶ Assume we are given a sequence of dominoes.
- ▶ Decide whether the top and bottom describe the same string.

Consider all possible finite sequences of dominoes $d_1 d_2 \dots d_n$ and call this E . The subset of E for which it holds that $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m$ is the set of solutions, called A . To solve a **decision problem** we would need a **decision procedure**.

Here a decision procedure would just test equality between pair components of the domino sequence, resulting in a yes/no answer. This obviously takes finite steps, because the sequences are finite.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(3): The simplified problem.

- ▶ Assume we are given a sequence of dominoes.
- ▶ Decide whether the top and bottom describe the same string.

Consider all possible finite sequences of dominoes $d_1 d_2 \dots d_n$ and call this E . The subset of E for which it holds that $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m$ is the set of solutions, called A . To solve a **decision problem** we would need a **decision procedure**.

Here a decision procedure would just test equality between pair components of the domino sequence, resulting in a yes/no answer. This obviously takes finite steps, because the sequences are finite.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(4): The simplified problem.

So considering we have a decision method for the simplified problem we can see it *is* calculable.

Now to consider the full problem.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



A small rerun on decision problems(4): The simplified problem.

So considering we have a decision method for the simplified problem we can see it *is* calculable.
Now to consider the full problem.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



A small rerun on decision problems(5)

Full PCP is also an instance of a decision problem.

- ▶ Consider a finite list of tuples containing strings on a and b (dominoes): $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.
 - The decision problem: Is there a sequences of indices $i_{1 \leq k \leq K}$ with $K \geq 1$ and $1 \leq i_k \leq N$ for which it holds that $x_1 x_2 \dots x_K = y_1 y_2 \dots y_K$?

To solve this **decision problem** we would need a **decision procedure**. There does not exist (an algorithmic) decision procedure for this!

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

A small rerun on decision problems(5)

Full PCP is also an instance of a decision problem.

- ▶ Consider a finite list of tuples containing strings on a and b (dominoes): $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.
 - The decision problem: Is there a sequences of indices $i_{1 \leq k \leq K}$ with $K \geq 1$ and $1 \leq i_k \leq N$ for which it holds that $x_1 x_2 \dots x_K = y_1 y_2 \dots y_K$?

To solve this decision problem we would need a decision procedure. There does not exist (an algorithmic) decision procedure for this!

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation
Introduction

**Proof of
recursive
unsolvability of
PCP**

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline of the proof

- ▶ PCP:
 - Introduce **Post Canonical Systems** (PCS).
 - Introduce **class of normal systems** on PCS.
 - State **decision problem** for this class.
- ▶ Introduce a problem that was already proved undecidable (by Post).
- ▶ **Reduction** of this decision problem to PCP.
- ▶ Prove that the reduction is **recursively** (computable).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline of the proof

- ▶ PCP:
 - Introduce **Post Canonical Systems** (PCS).
 - Introduce **class of normal systems** on PCS.
 - State **decision problem** for this class.
- ▶ Introduce a problem that was already proved undecidable (by Post).
- ▶ **Reduction** of this decision problem to PCP.
- ▶ Prove that the reduction is **recursively** (computable).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline of the proof

- ▶ PCP:
 - Introduce **Post Canonical Systems** (PCS).
 - Introduce **class of normal systems** on PCS.
 - State **decision problem** for this class.
- ▶ Introduce a problem that was already proved undecidable (by Post).
- ▶ **Reduction** of this decision problem to PCP.
- ▶ Prove that the reduction is **recursively** (computable).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(1)

- ▶ Post describes a Post Canonical System (PCS) as a logical system.
- ▶ It is now mostly seen as one of the first string rewriting systems.
- ▶ A PCS contains:
 - a finite **alphabet** (for example $\{a, b\}$).
 - a finite number of axioms/**primitive assertions**.
 - a finite number of **productions** (production rules).
- ▶ **Enunciations** (formulas of the system) can then be obtained by applying a finite number of production rules.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(1)

- ▶ Post describes a Post Canonical System (PCS) as a logical system.
- ▶ It is now mostly seen as one of the first string rewriting systems.
- ▶ A PCS contains:
 - a finite **alphabet** (for example $\{a, b\}$).
 - a finite number of axioms/**primitive assertions**.
 - a finite number of **productions** (production rules).
- ▶ **Enunciations** (formulas of the system) can then be obtained by applying a finite number of production rules.

Post's
Correspondence
Problem

Motivation

Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(2)

- ▶ a finite number of axioms/**primitive assertions**.

Primitive assertions are a specified finite set of enunciations. So simply strings on the defined alphabet.

- ▶ a finite number of productions (production rules).

In a canonical system, productions can be of the following form:

$$\begin{aligned} &g_{11}P_{i_1'}g_{12}P_{i_2'}\dots g_{1m_1}P_{i_{m_1}'}g_{1(m_1+1)} \\ &g_{21}P_{i_1''}g_{22}P_{i_2''}\dots g_{2m_2}P_{i_{m_2}''}g_{2(m_2+1)} \\ &\dots \\ &g_{k1}P_{i_1^{(k)}}g_{k2}P_{i_2^{(k)}}\dots g_{km_k}P_{i_{m_k}^{(k)}}g_{k(m_k+1)} \\ &\Rightarrow \\ &g_1P_{i_1}g_2P_{i_2}\dots g_mP_{i_m}g_{m+1} \end{aligned}$$

g_{ki} are fixed sequences on the alphabet (words).

P_{ki} are variables standing for words.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(2)

- ▶ a finite number of axioms/**primitive assertions**.

Primitive assertions are a specified finite set of enunciations. So simply strings on the defined alphabet.

- ▶ a finite number of productions (production rules).

In a canonical system, productions can be of the following form:

$$\begin{aligned} &g_{11}P_{i_1'}g_{12}P_{i_2'}\dots g_{1m_1}P_{i_{m_1}'}g_{1(m_1+1)} \\ &g_{21}P_{i_1''}g_{22}P_{i_2''}\dots g_{2m_2}P_{i_{m_2}''}g_{2(m_2+1)} \\ &\dots \\ &g_{k1}P_{i_1^{(k)}}g_{k2}P_{i_2^{(k)}}\dots g_{km_k}P_{i_{m_k}^{(k)}}g_{k(m_k+1)} \\ &\Rightarrow \\ &g_1P_{i_1}g_2P_{i_2}\dots g_mP_{i_m}g_{m+1} \end{aligned}$$

g_{ki} are fixed sequences on the alphabet (words).

P_{ki} are variables standing for words.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(2)

- ▶ a finite number of axioms/**primitive assertions**.

Primitive assertions are a specified finite set of enunciations. So simply strings on the defined alphabet.

- ▶ a finite number of productions (production rules).

In a canonical system, productions can be of the following form:

$$\begin{aligned} &g_{11}P_{i_1}'g_{12}P_{i_2}'2\dots g_{1m_1}P_{i_{m_1}}'g_{1(m_1+1)} \\ &g_{21}P_{i_1}''g_{22}P_{i_2}''2\dots g_{2m_2}P_{i_{m_2}}''g_{2(m_2+1)} \\ &\dots \\ &g_{k1}P_{i_1}^{(k)}g_{k2}P_{i_2}^{(k)}\dots g_{km_k}P_{i_{m_k}}^{(k)}g_{k(m_k+1)} \\ &\quad \Rightarrow \\ &g_1P_{i_1}g_2P_{i_2}\dots g_mP_{i_m}g_{m+1} \end{aligned}$$

g_{ki} are fixed sequences on the alphabet (words).

P_{ki} are variables standing for words.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(2)

- ▶ a finite number of axioms/**primitive assertions**.

Primitive assertions are a specified finite set of enunciations. So simply strings on the defined alphabet.

- ▶ a finite number of productions (production rules).

In a canonical system, productions can be of the following form:

$$\begin{aligned} &g_{11}P_{i_1}'g_{12}P_{i_2}'2\dots g_{1m_1}P_{i_{m_1}}'g_{1(m_1+1)} \\ &g_{21}P_{i_1}''g_{22}P_{i_2}''2\dots g_{2m_2}P_{i_{m_2}}''g_{2(m_2+1)} \\ &\dots \\ &g_{k_1}P_{i_1}^{(k)}g_{k_2}P_{i_2}^{(k)}\dots g_{km_k}P_{i_{m_k}}^{(k)}g_{k(m_k+1)} \\ &\quad \Rightarrow \\ &g_1P_{i_1}g_2P_{i_2}\dots g_mP_{i_m}g_{m+1} \end{aligned}$$

g_{ki} are fixed sequences on the alphabet (words).

P_{ki} are variables standing for words.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(3): an example

Alphabet: $\{a, b\}$

Primitive assertion(s): a

Production rules:

(1): $P_1, a \Rightarrow aP_1b$ (note two antecedents)

(2): $P_2aP_3 \Rightarrow P_2P_3$

Derivation of enunciation ab :

$a \Rightarrow_1 aab \Rightarrow_2 ab$

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(4)

- ▶ Determining whether an enunciation is part of a PCS is **undecidable**.
- ▶ A simpler (normal) form of a PCS exists, also called a Post Normal System (PNS).
 - A PNS has restrictions on the assertions and production rules.
 - Using the Normal-form theorem proved by Post, you can reduce any PCS to a PNS.
 - Thus the decision problem for PNS is **still** recursively unsolvable.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(5): normal form

The normal form of a PCS has a number of restrictions/simplifications.

- ▶ Only one primitive assertion, non-null.
- ▶ One antecedent in production rules.
- ▶ Production rules have additional restrictions on form.

Form of production rules:

$a_i P$ produces Pa'_i

a_i stands for a fixed word.

P is a variable standing for a word, possibly null.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Post Canonical Systems(5): normal form

The normal form of a PCS has a number of restrictions/simplifications.

- ▶ Only **one** primitive assertion, non-null.
- ▶ **One antecedent** in production rules.
- ▶ Production rules have additional **restrictions on form**.

Form of production rules:

$$a_i P \text{ produces } Pa_i'$$

a_i stands for a fixed word.

P is a variable standing for a word, possibly null.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Post Canonical Systems(5): normal form

The normal form of a PCS has a number of restrictions/simplifications.

- ▶ Only **one** primitive assertion, non-null.
- ▶ **One antecedent** in production rules.
- ▶ Production rules have additional **restrictions on form**.

Form of production rules:

$$a_i P \text{ produces } Pa_i'$$

a_i stands for a fixed word.

P is a variable standing for a word, possibly null.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (1)

- ▶ Determining whether an enunciation is part of a PNS is again **undecidable**.
- ▶ This can also be formulated as the transformation from the axiom A to the enunciation B . Thus:
 - The transformation is a finite number of applications of production rules to A . Giving the equation:
 - $A = a_i P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B$
 - Special case $n = 0$ gives $A = B$.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

From PNS to PCP (1)

- ▶ Determining whether an enunciation is part of a PNS is again **undecidable**.
- ▶ This can also be formulated as the transformation from the axiom A to the enunciation B . Thus:
 - The transformation is a finite number of applications of production rules to A . Giving the equation:
 - $A = a_i P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B$
 - Special case $n = 0$ gives $A = B$.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

From PNS to PCP (1)

- ▶ Determining whether an enunciation is part of a PNS is again **undecidable**.
- ▶ This can also be formulated as the transformation from the axiom A to the enunciation B . Thus:
 - The transformation is a finite number of applications of production rules to A . Giving the equation:
 - $A = a_i P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B$
 - Special case $n = 0$ gives $A = B$.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

From PNS to PCP (1)

- ▶ Determining whether an enunciation is part of a PNS is again **undecidable**.
- ▶ This can also be formulated as the transformation from the axiom A to the enunciation B . Thus:
 - The transformation is a finite number of applications of production rules to A . Giving the equation:
 - $A = a_i P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B$
 - Special case $n = 0$ gives $A = B$.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.
5. Arrive at the equation from the PCP problem:

$$x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$$

6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.
5. Arrive at the equation from the PCP problem:
 $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$
6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).

4. Remove the A and B in the equations.

5. Arrive at the equation from the PCP problem:

$$x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$$

6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.

5. Arrive at the equation from the PCP problem:

$$x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$$

6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.
5. Arrive at the equation from the PCP problem:

$$x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$$

6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.
5. Arrive at the equation from the PCP problem:
 $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$
6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



From PNS to PCP (2)

A very rough sketch of the further proof:

1. Take the previous equations:

$$A = a_1 P_1, P_1 a'_{i_1} = a'_{i_2} P_2, \dots, P_{n-1} a'_{i_{n-1}} = a_{i_n} P_n, P_n a'_{i_n} = B.$$

2. Rewrite this to a more suitable form.
3. Remove additional constraints regarding the length (derive these from the equations themselves).
4. Remove the A and B in the equations.
5. Arrive at the equation from the PCP problem:
 $x_1 x_2 \dots x_m = y_1 y_2 \dots y_m.$
6. Prove the transformation steps recursive.

We now have "proven" that the decision problem for PNS is recursively **uncomputable**.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(1)

A very rough short sketch of the proof done by Michael Sipser.

1. Consider arbitrary Turing Machine (TM), M , and arbitrary input, w .
2. Let the language
$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}.$$
(This is the halting problem!)

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

PCP as a Turing machine(1)

A very rough short sketch of the proof done by Michael Sipser.

1. Consider arbitrary Turing Machine (TM), M , and arbitrary input, w .
2. Let the language
 $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$.
(This is the halting problem!)

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(1)

A very rough short sketch of the proof done by Michael Sipser.

1. Consider arbitrary Turing Machine (TM), M , and arbitrary input, w .
2. Let the language
$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}.$$
(This is the halting problem!)

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(1)

A very rough short sketch of the proof done by Michael Sipser.

1. Consider arbitrary Turing Machine (TM), M , and arbitrary input, w .
2. Let the language
$$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}.$$
(This is the halting problem!)

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

PCP as a Turing machine(2)

- 3 He now constructs an instance of PCP called P , where a match is an accepting computation history for M on w .
- i This done by taking the initial state (q_0) and the input string on the bottom of the first domino. This will be called MPCP for Modified PCP.
 - ii Introduce separators to distinguish states (add these to the alphabet of P).
 - iii Adding dominoes corresponds to taking computation steps.
 - iv The last state on the domino will contain an accepting state.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(2)

- 3 He now constructs an instance of PCP called P , where a match is an accepting computation history for M on w .
- i This done by taking the initial state (q_0) and the input string on the bottom of the first domino. This will be called MPCP for Modified PCP.
 - ii Introduce separators to distinguish states (add these to the alphabet of P).
 - iii Adding dominoes corresponds to taking computation steps.
 - iv The last state on the domino will contain an accepting state.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(2)

- 3 He now constructs an instance of PCP called P , where a match is an accepting computation history for M on w .
- i This done by taking the initial state (q_0) and the input string on the bottom of the first domino. This will be called MPCP for Modified PCP.
 - ii Introduce separators to distinguish states (add these to the alphabet of P).
 - iii Adding dominoes corresponds to taking computation steps.
 - iv The last state on the domino will contain an accepting state.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(2)

- 3 He now constructs an instance of PCP called P , where a match is an accepting computation history for M on w .
- i This done by taking the initial state (q_0) and the input string on the bottom of the first domino. This will be called MPCP for Modified PCP.
 - ii Introduce separators to distinguish states (add these to the alphabet of P).
 - iii Adding dominoes corresponds to taking computation steps.
 - iv The last state on the domino will contain an accepting state.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(2)

- 3 He now constructs an instance of PCP called P , where a match is an accepting computation history for M on w .
 - i This done by taking the initial state (q_0) and the input string on the bottom of the first domino. This will be called MPCP for Modified PCP.
 - ii Introduce separators to distinguish states (add these to the alphabet of P).
 - iii Adding dominoes corresponds to taking computation steps.
 - iv The last state on the domino will contain an accepting state.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



PCP as a Turing machine(3)

- 4 Constructing a solution to P now corresponds to finding a solution to the halting problem (roughly).
- 5 The halting problem is recursively uncomputable, thus computing P must also be recursively uncomputable. Because P is an instance of PCP, PCP must also be recursively uncomputable in general.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

PCP as a Turing machine(3)

- 4 Constructing a solution to P now corresponds to finding a solution to the halting problem (roughly).
- 5 The halting problem is recursively uncomputable, thus computing P must also be recursively uncomputable. Because P is an instance of PCP, PCP must also be recursively uncomputable in general.

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Outline

Post's Correspondence Problem

Motivation

Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)

A modern proof (Sipser)

Concluding remarks

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Summary

- ▶ PCP is a relatively easy to describe **decision problem**.
- ▶ We have seen two proofs that generally, PCP is recursively unsolvable.
- ▶ We can use this fact to reduce another problem we suspect to be undecidable to PCP. Thus proving that problem undecidable. Don't forget to prove this step to be recursive/computable!

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Variants of PCP

There are various useful and interesting variants that can be made on PCP.

- ▶ Conditions on the alphabet. (An alphabet of size 1 is decidable.)
- ▶ Conditions on the number of transformations. (Size ≤ 2 is decidable and ≥ 7 is undecidable.)
- ▶ Bounded PCP: Can we find a match between domino tiles using at most k (possibly repeated) tiles?
- ▶ And many more useful variations...

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Variants of PCP

There are various useful and interesting variants that can be made on PCP.

- ▶ Conditions on the alphabet. (An alphabet of size 1 is decidable.)
- ▶ Conditions on the number of transformations. (Size ≤ 2 is decidable and ≥ 7 is undecidable.)
- ▶ Bounded PCP: Can we find a match between domino tiles using at most k (possibly repeated) tiles?
- ▶ And many more useful variations...

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Variants of PCP

There are various useful and interesting variants that can be made on PCP.

- ▶ Conditions on the alphabet. (An alphabet of size 1 is decidable.)
- ▶ Conditions on the number of transformations. (Size ≤ 2 is decidable and ≥ 7 is undecidable.)
- ▶ Bounded PCP: Can we find a match between domino tiles using at most k (possibly repeated) tiles?
- ▶ And many more useful variations...

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Variants of PCP

There are various useful and interesting variants that can be made on PCP.

- ▶ Conditions on the alphabet. (An alphabet of size 1 is decidable.)
- ▶ Conditions on the number of transformations. (Size ≤ 2 is decidable and ≥ 7 is undecidable.)
- ▶ Bounded PCP: Can we find a match between domino tiles using at most k (possibly repeated) tiles?
- ▶ And many more useful variations...

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Bounded PCP and NP-completeness

- ▶ Bounded PCP (BPCP) is an interesting case of PCP.
- ▶ Solving BPCP can be solved brute force, by trying all cases until size k .
- ▶ BPCP is in the class of non-deterministic polynomial time solvable problems (NP).
- ▶ Any problem in NP can be reduced to BPCP in polynomial time, and thus BPCP is NP-complete.

So putting these constraints on PCP gives us an NP-complete problem!

And therefore very relevant to resource bounded models of computation (Dragos).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Bounded PCP and NP-completeness

- ▶ Bounded PCP (BPCP) is an interesting case of PCP.
- ▶ Solving BPCP can be solved brute force, by trying all cases until size k .
- ▶ BPCP is in the class of non-deterministic polynomial time solvable problems (NP).
- ▶ Any problem in NP can be reduced to BPCP in polynomial time, and thus BPCP is NP-complete.

So putting these constraints on PCP gives us an NP-complete problem!

And therefore very relevant to resource bounded models of computation (Dragos).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Bounded PCP and NP-completeness

- ▶ Bounded PCP (BPCP) is an interesting case of PCP.
- ▶ Solving BPCP can be solved brute force, by trying all cases until size k .
- ▶ BPCP is in the class of non-deterministic polynomial time solvable problems (NP).
- ▶ Any problem in NP can be reduced to BPCP in polynomial time, and thus BPCP is NP-complete.

So putting these constraints on PCP gives us an NP-complete problem!

And therefore very relevant to resource bounded models of computation (Dragos).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Bounded PCP and NP-completeness

- ▶ Bounded PCP (BPCP) is an interesting case of PCP.
- ▶ Solving BPCP can be solved brute force, by trying all cases until size k .
- ▶ BPCP is in the class of non-deterministic polynomial time solvable problems (NP).
- ▶ Any problem in NP can be reduced to BPCP in polynomial time, and thus BPCP is NP-complete.

So putting these constraints on PCP gives us an NP-complete problem!

And therefore very relevant to resource bounded models of computation (Dragos).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Bounded PCP and NP-completeness

- ▶ Bounded PCP (BPCP) is an interesting case of PCP.
- ▶ Solving BPCP can be solved brute force, by trying all cases until size k .
- ▶ BPCP is in the class of non-deterministic polynomial time solvable problems (NP).
- ▶ Any problem in NP can be reduced to BPCP in polynomial time, and thus BPCP is NP-complete.

So putting these constraints on PCP gives us an NP-complete problem!

And therefore very relevant to resource bounded models of computation (Dragos).

Post's
Correspondence
Problem

Motivation
Introduction

Proof of
recursive
unsolvability of
PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding
remarks



Universiteit Utrecht

Questions



Post's Correspondence Problem

Motivation
Introduction

Proof of recursive unsolvability of PCP

The classic proof (Post)
A modern proof (Sipser)

Concluding remarks

