Decision Problems

Introduction to Formal Language Theory — day 5

Wiebke Petersen, Kata Balogh

Heinrich-Heine-Universität

NASSLLI 2014

Decision problem

A decision problem is a problem of the form "Given $(x_1, \ldots, x_n)$, can we decide whether y holds?"

- A tuple $(x_1, \ldots, x_n)$ is called an instance of the problem.
- A tuple $(x_1, \ldots, x_n)$ for which y holds is called a positive instance of the problem.

Languages and problems

- Problems have the form: “Can we decide for every x whether it has property P?”
- Languages as problems: “Can we decide for every word whether it belongs to L?”
- Problems as languages: “The language of all x which have property P.”

- Problems have the form: “Can we decide for every x whether it has property P?”
- Languages as problems: “Can we decide for every word whether it belongs to L?”
- Problems as languages: “The language of all x which have property P.”

examples:

- Can we decide for any pair (M, w) consisting of a Turing machine M and a word w whether M halts on w?
- Can we decide for any pair (G_1, G_2) of two context-free grammars whether $L(G_1) = L(G_2)$?
- Can we decide for any context-free grammar G whether $L(G) = \emptyset$?

problem instances versus problems

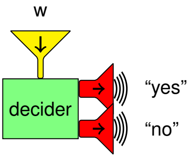
- Single instances are not problems! Whether '$S \rightarrow a$' generates a word is simple to answer, but not the general problem ranging over all possible instances.
- Problems can be represented by sets with positive instances as elements.

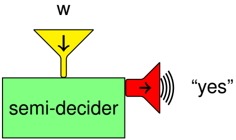
A language $L \subseteq \Sigma^*$ is decidable if its *characteristic function* $\chi_L : \Sigma^* \to \{0, 1\}$ is computable:

$$\chi_L(w) = \begin{cases} 1, & w \in L \\ 0, & w \notin L \end{cases}$$

A language $L \subseteq \Sigma^*$ is semi-decidable if $\chi'_L : \Sigma^* \to \{0, 1\}$ is computable:

$$\chi_L(w) = \begin{cases} 1, & w \in L \\ \textit{undefined}, & w \notin L \end{cases}$$





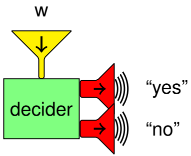
decidabiltiy

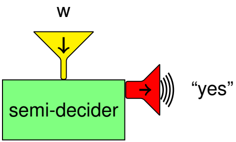
A language $L \subseteq \Sigma^*$ is decidable if its *characteristic function* $\chi_L : \Sigma^* \to \{0, 1\}$ is computable:

$$\chi_L(w) = \begin{cases} 1, & w \in L \\ 0, & w \notin L \end{cases}$$

A language $L \subseteq \Sigma^*$ is semi-decidable if $\chi'_L : \Sigma^* \to \{0, 1\}$ is computable:

$$\chi_L(w) = \begin{cases} 1, & w \in L \\ \textit{undefined}, & w \notin L \end{cases}$$





- L is decidable if and only if L and $\overline{L}$ are semi-decidable.
- A language L is recursively enumerable (RE) if and only if L is semi-decidable.

Given: grammars $G = (N, \Sigma, S, R)$, $G' = (N', \Sigma', S', R')$, and a word $w \in \Sigma$:

word problem: Is w derivable from G, i.e. $w \in L(G)$?

emptiness problem: Does G generate a nonempty language, i.e. $L(G) \neq \emptyset$?

equivalence problem: Do G and G' generate the same language, i.e. $L(G) = L(G')$?

	Type3	Type2	Type1	Type0
word problem	**D**	**D**	**D**	**U**
emptiness problem	**D**	**D**	**U**	**U**
equivalence problem	**D**	**U**	**U**	**U**

D: decidable; U: undecidable

	Type3	Type2	Type1	Type0
word problem	**D**	**D**	**D**	**U**
emptiness problem	**D**	**D**	**U**	**U**
equivalence problem	**D**	**U**	**U**	**U**

D: decidable; U: undecidable

- word problem for Type1: use the property that the derivation string does not shrink in any derivation step.
- emptyness problem for Type2: bottom up argument over the non-terminals from which terminal strings can be derived.
- equivalence problem for Type3: check via minimal automaton.

An universal Turing machine U is a TM that simulates arbitrary other TMs. It takes as input

- the description of a Turing machine M and
- an input string w

and accepts w if and only if M accepts w.

An universal Turing machine U is a TM that simulates arbitrary other TMs. It takes as input

- the description of a Turing machine M and
- an input string w

and accepts w if and only if M accepts w.

Construction idea: Use a 2-tape Turing machine

- 1st tape: encoding of M
- 2nd tape: w

The universal machine reads the code of M on tape 1 to see what to do with the word on tape 2 (tape 1 is not changed).

Gödel numbering

A Gödel numbering is a function $G : M \to \mathbb{N}$ with

- G is injective
- $G(M)$ is decidable
- $G : M \to \mathbb{N}$ and $G^{-1} : G(M) \to M$ are computable

Gödel numbering

A Gödel numbering is a function $G : M \to \mathbb{N}$ with

- G is injective
- $G(M)$ is decidable
- $G : M \to \mathbb{N}$ and $G^{-1} : G(M) \to M$ are computable

Gödel numbering of TMs (using binary code)

- Given $M = (Q, \Sigma, \Gamma, \delta, q_1, \Box, F)$, we assume that
  - $Q = \{q_1, q_2, \ldots\}$
  - $\Gamma = \{X_1, X_2, \ldots\}$
  - $\Box = X_1$
  - $F = \{q_2\}$
  - $D_1 = R$, $D_2 = L$
- Code each transition $\delta(q_i, X_j) = (q_k, X_l, D_m)$ as $0^i 1 0^j 1 0^k 1 0^l 1 0^m$
- Note that this code never has two successive 1's.
- Code M by concatenating all transition codes C_i with '11'-strings as separators: $G(M) = 11C_1 11C_2 11C_3 \ldots 11C_n$.
- $M \mapsto G(M)$ is a Gödel numbering of Turing machines.

Note: $\{G(M) | M \text{ is a TM}\}$ and $\{M | M \text{ is a TM}\}$ are countable sets.

$H = \{G(M)\#w | M(w) \text{ halts}\}$

- Given a Turing machine *M* and an input word *w*.
- Does *M* halt if it runs on input *w*?

$H = \{G(M)\#w | M(w) \text{ halts}\}$

- Given a Turing machine M and an input word w.
- Does M halt if it runs on input w?

The halting problem is undecidable.

Proof by a diagonal argument:

- Assume that the halting problem is decidable.

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	...
G_1	**0**	1	1	0	1	0	0	1	1	...
G_2	0	**1**	0	1	1	1	0	1	0	...
G_3	1	0	**1**	0	1	0	1	0	1	...
G_4	0	1	1	**1**	0	1	0	1	1	...
G_5	0	1	0	1	**0**	1	1	0	1	...
G_6	1	1	0	1	0	**1**	1	0	0	...
G_7	0	1	0	1	0	1	**0**	1	0	...
G_8	1	1	1	0	1	0	1	**0**	1	...
G_9	1	1	0	1	0	1	1	1	**1**	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

$H = \{G(M)\#w | M(w) \text{ halts}\}$

- Given a Turing machine M and an input word w.
- Does M halt if it runs on input w?

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	...
G_1	0	1	1	0	1	0	0	1	1	...
G_2	0	1	0	1	1	1	0	1	0	...
G_3	1	0	1	0	1	0	1	0	1	...
G_4	0	1	1	1	0	1	0	1	1	...
G_5	0	1	0	1	0	1	1	0	1	...
G_6	1	1	0	1	0	1	1	0	0	...
G_7	0	1	0	1	0	1	0	1	0	...
G_8	1	1	1	0	1	0	1	0	1	...
G_9	1	1	0	1	0	1	1	1	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

The halting problem is undecidable.
Proof by a diagonal argument:

- Assume that the halting problem is decidable.

$\Rightarrow$ there is a TM H which computes for every TM M and every word w, whether M halts on w.
Let w_i be the i-th word and G_i the TM with the i-th Gödel number.

$H = \{G(M)\#w|M(w) \text{ halts}\}$

- Given a Turing machine M and an input word w.
- Does M halt if it runs on input w?

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	...
G_1	0	1	1	0	1	0	0	1	1	...
G_2	0	1	0	1	1	1	0	1	0	...
G_3	1	0	1	0	1	0	1	0	1	...
G_4	0	1	1	1	0	1	0	1	1	...
G_5	0	1	0	1	0	1	1	0	1	...
G_6	1	1	0	1	0	1	1	0	0	...
G_7	0	1	0	1	0	1	0	1	0	...
G_8	1	1	1	0	1	0	1	0	1	...
G_9	1	1	0	1	0	1	1	1	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

The halting problem is undecidable.
Proof by a diagonal argument:

- Assume that the halting problem is decidable.

$\Rightarrow$ there is a TM H which computes for every TM M and every word w, whether M halts on w.
Let w_i be the i-th word and G_i the TM with the i-th Gödel number.

- From H construct a second TM H' which takes a word w_i as input and acts as follows:
  - Whenever H outputs 1 for (G_i, w_i), H' goes into an endless loop.
  - Whenever H outputs 0 for (G_i, w_i), H' halts.

$H = \{G(M)\#w|M(w) \text{ halts}\}$

- Given a Turing machine M and an input word w.
- Does M halt if it runs on input w?

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	...
G_1	0	1	1	0	1	0	0	1	1	...
G_2	0	1	0	1	1	1	0	1	0	...
G_3	1	0	1	0	1	0	1	0	1	...
G_4	0	1	1	1	0	1	0	1	1	...
G_5	0	1	0	1	0	1	1	0	1	...
G_6	1	1	0	1	0	1	1	0	0	...
G_7	0	1	0	1	0	1	0	1	0	...
G_8	1	1	1	0	1	0	1	0	1	...
G_9	1	1	0	1	0	1	1	1	1	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

The halting problem is undecidable.
Proof by a diagonal argument:

- Assume that the halting problem is decidable.

$\Rightarrow$ there is a TM H which computes for every TM M and every word w, whether M halts on w.
Let w_i be the i-th word and G_i the TM with the i-th Gödel number.

- From H construct a second TM H' which takes a word w_i as input and acts as follows:
  - Whenever H outputs 1 for (G_i, w_i), H' goes into an endless loop.
  - Whenever H outputs 0 for (G_i, w_i), H' halts.

$\Rightarrow$ H' is a TM of which the Gödel number is not in the matrix.

Halting problem

$H = \{G(M)\#w|M(w) \text{ halts}\}$

- Given a Turing machine M and an input word w.
- Does M halt if it runs on input w?

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	...
G_1	0	1	1	0	1	0	0	1	1	...
G_2	0	1	0	1	1	1	0	1	0	...
G_3	1	0	1	0	1	0	1	0	1	...
G_4	0	1	1	1	0	1	0	1	1	...
G_5	0	1	0	1	0	1	1	0	1	...
G_6	1	1	0	1	0	1	1	0	0	...
G_7	0	1	0	1	0	1	0	1	0	...
G_8	1	1	1	0	1	0	1	0	1	...
G_9	1	1	0	1	0	1	1	1	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

The halting problem is undecidable.
Proof by a diagonal argument:

- Assume that the halting problem is decidable.

$\Rightarrow$ there is a TM H which computes for every TM M and every word w, whether M halts on w.
Let w_i be the i-th word and G_i the TM with the i-th Gödel number.

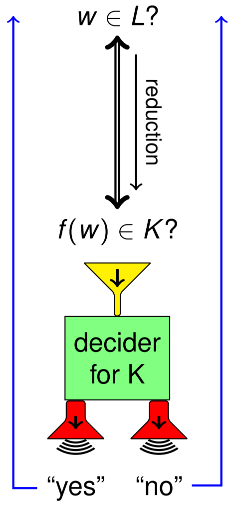
- From H construct a second TM H' which takes a word w_i as input and acts as follows:
  - Whenever H outputs 1 for (G_i, w_i), H' goes into an endless loop.
  - Whenever H outputs 0 for (G_i, w_i), H' halts.

$\Rightarrow$ H' is a TM of which the Gödel number is not in the matrix.

$\Rightarrow$ the assumption is wrong; the halting problem is undecidable.

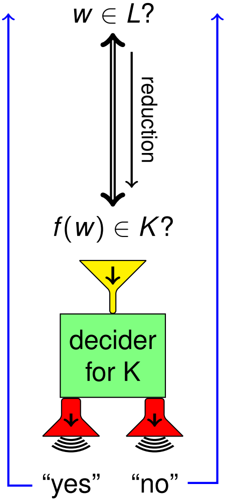
Given two languages $L \subseteq \Sigma^*$ and $K \subseteq \Gamma^*$. L is reducible to K (in symbols $L \leq K$) if there exists a total function $f : \Sigma^* \to \Gamma^*$, such that

- f is computable and
- $w \in L \Leftrightarrow f(x) \in K$ for all $w \in \Sigma^*$.



Given two languages $L \subseteq \Sigma^*$ and $K \subseteq \Gamma^*$. L is reducible to K (in symbols $L \leq K$) if there exists a total function $f : \Sigma^* \to \Gamma^*$, such that

- f is computable and
- $w \in L \Leftrightarrow f(x) \in K$ for all $w \in \Sigma^*$.



Lemma

- If $L \leq K$ and K is decidable, then L is decidable.
- If $L \leq K$ and K is semi-decidable, then L is semi-decidable.
- If $L \leq K$ and L is undecidable, then K is undecidable.

$H_0 = \{G(M) | M(\epsilon) \text{ halts}\}$

- Given a Turing machine M.
- Does M halt if it runs on input ϵ?

The halting problem on the empty tape is undecidable.

$H_0 = \{G(M) | M(\epsilon) \text{ halts}\}$

- Given a Turing machine M.
- Does M halt if it runs on input ϵ?

The halting problem on the empty tape is undecidable.

Proof by reduction $H \leq H_0$:

- Let $G(M)\#w$ be an instance of H.

$H_0 = \{G(M) | M(\epsilon) \text{ halts}\}$

- Given a Turing machine M.
- Does M halt if it runs on input ϵ?

The halting problem on the empty tape is undecidable.

Proof by reduction $H \leq H_0$:

- Let $G(M)\#w$ be an instance of H.
- Define a Turing machine M_w which starts with the empty tape, writes w onto the tape, and simulates M on w.

$H_0 = \{G(M) | M(\epsilon) \text{ halts}\}$

- Given a Turing machine M.
- Does M halt if it runs on input ϵ?

The halting problem on the empty tape is undecidable.

Proof by reduction $H \leq H_0$:

- Let $G(M)\#w$ be an instance of H.
- Define a Turing machine M_w which starts with the empty tape, writes w onto the tape, and simulates M on w.
- $f : G(M)\#w \mapsto G(M_w)$ is a computable function and
- $G(M)\#w \in H \Leftrightarrow G(M_w) \in H_0$

$H_0 = \{G(M) | M(\epsilon) \text{ halts}\}$

- Given a Turing machine M.
- Does M halt if it runs on input ϵ?

The halting problem on the empty tape is undecidable.

Proof by reduction $H \leq H_0$:

- Let $G(M)\#w$ be an instance of H.
- Define a Turing machine M_w which starts with the empty tape, writes w onto the tape, and simulates M on w.
- $f : G(M)\#w \mapsto G(M_w)$ is a computable function and
- $G(M)\#w \in H \Leftrightarrow G(M_w) \in H_0$

$\Rightarrow$ H_0 is undecidable.

If M is a Turing machine let f_M be the function computed by M. A functional property of M, i.e. a property of f_M is non-trivial if there is at least one Turing machine which has the property and one which has it not.

If M is a Turing machine let f_M be the function computed by M. A functional property of M, i.e. a property of f_M is non-trivial if there is at least one Turing machine which has the property and one which has it not.

Theorem of Rice

Let P be a non-trivial property of Turing machines.

- Given a Turing machine M.
- Does M has property P?

Any non-trivial property of a Turing machine is undecidable.

If M is a Turing machine let f_M be the function computed by M. A functional property of M, i.e. a property of f_M is non-trivial if there is at least one Turing machine which has the property and one which has it not.

Theorem of Rice

Let P be a non-trivial property of Turing machines.

- Given a Turing machine M.
- Does M has property P?

Any non-trivial property of a Turing machine is undecidable.

examples of non-trivial properties

- The computed function is constant.
- The Turing machine computes the successor function.
- The Turing machine computes a total function.

Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).

Proof of Rice's theorem

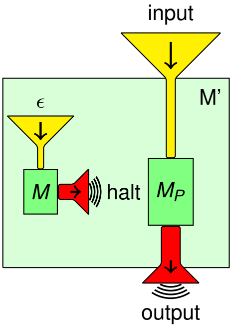
Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.

Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.
- Construct a new TM M'. For any input w
  - M' first computes $M(\epsilon)$ and if it halts
  - M' computes $M_P(w)$

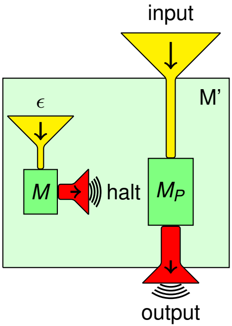


Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.
- Construct a new TM M'. For any input w
  - M' first computes $M(\epsilon)$ and if it halts
  - M' computes $M_P(w)$

If $G(M) \notin H_0$: $M(\epsilon)$ does not halt and M' computes $M_\perp$, thus $G(M') \notin P$



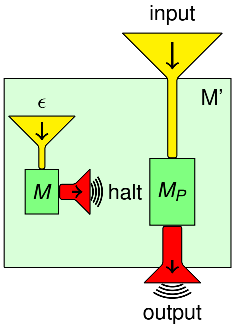
Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.
- Construct a new TM M'. For any input w
  - M' first computes $M(\epsilon)$ and if it halts
  - M' computes $M_P(w)$

If $G(M) \notin H_0$: $M(\epsilon)$ does not halt and M' computes $M_\perp$, thus $G(M') \notin P$

If $G(M) \in H_0$: $M(\epsilon)$ does halt and M computes M_P, thus $G(M') \in P$.



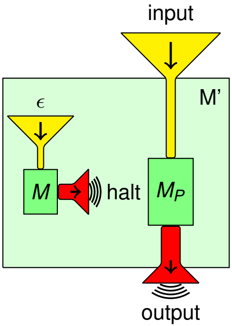
Proof of Rice's theorem

Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.
- Construct a new TM M'. For any input w
  - M' first computes $M(\epsilon)$ and if it halts
  - M' computes $M_P(w)$

  If $G(M) \notin H_0$: $M(\epsilon)$ does not halt and M' computes $M_\perp$, thus $G(M') \notin P$

  If $G(M) \in H_0$: $M(\epsilon)$ does halt and M computes M_P, thus $G(M') \in P$.
- As $f : G(M) \mapsto G(M')$ is computable and $G(M) \in H_0 \Leftrightarrow G(M') \in P$, we proved $H_0 \leq P$.

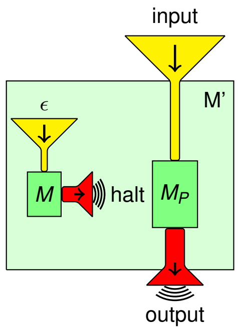


Given a non-trivial functional property. Proof by reduction $H_0 \leq P$:

- Construct a TM $M_\perp$ which never halts.
- Assume $M_\perp$ does not have property P (argument for $G(M_\perp) \in P$ is analogous).
- As P is non-trivial there is a TM M_P with $G(M_P) \in P$.
- Construct a new TM M'. For any input w
  - M' first computes $M(\epsilon)$ and if it halts
  - M' computes $M_P(w)$

  If $G(M) \notin H_0$: $M(\epsilon)$ does not halt and M' computes $M_\perp$, thus $G(M') \notin P$

  If $G(M) \in H_0$: $M(\epsilon)$ does halt and M computes M_P, thus $G(M') \in P$.
- As $f : G(M) \mapsto G(M')$ is computable and $G(M) \in H_0 \Leftrightarrow G(M') \in P$, we proved $H_0 \leq P$.
- As H_0 is undecidable, P is undecidable as well.



Given: A finite set of word pairs $(x_1, y_1), \dots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \dots, i_n \in \{1, 2, \dots, k\}$ such that $x_{i_1} x_{i_2} \dots x_{i_n} = y_{i_1} y_{i_2} \dots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0

0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0

0	1	0	0	0

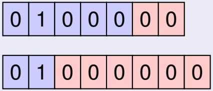
Given: A finite set of word pairs $(x_1, y_1), \dots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \dots, i_n \in \{1, 2, \dots, k\}$ such that $x_{i_1} x_{i_2} \dots x_{i_n} = y_{i_1} y_{i_2} \dots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223



Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1

0	1	0	0	0	0	0	0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1

example without solution

index	x_i	y_i
1	0	01
2	100	001

no solution

0	1	0	0	
0	1	0	0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \dots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \dots, i_n \in \{1, 2, \dots, k\}$ such that $x_{i_1} x_{i_2} \dots x_{i_n} = y_{i_1} y_{i_2} \dots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1

0	1	0	0	0	0	0	0	1

example without solution

index	x_i	y_i
1	0	01
2	100	001

no solution

0	1	0	0	1	0	0

0	1	0	0	1	0	0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1

0	1	0	0	0	0	0	0	1

example without solution

index	x_i	y_i
1	0	01
2	100	001

no solution

0	1	0	0	1	0	0	1	0	0

0	1	0	0	1	0	0	1	0	0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1

0	1	0	0	0	0	0	0	1

example without solution

index	x_i	y_i
1	0	01
2	100	001

no solution

0	1	0	0	1	0	0	1	0	0	1	0	0

0	1	0	0	1	0	0	1	0	0	1	0	0	1

Post's Correspondence Problem (PCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$?

example with solution

index	x_i	y_i
1	01000	01
2	0	000
3	01	1

solution: 1223

0	1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1

example without solution

index	x_i	y_i
1	0	01
2	100	001

no solution

0	1	0	0	1	0	0	1	0	0	1	0	0	
0	1	0	0	1	0	0	1	0	0	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1

0	1	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0

0	1	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1

0	1	1	0	0	1	1	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0

0	1	1	0	0	1	1	0	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1

0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1

0	0	1	1	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1

0	0	1	1	0	1	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0																											

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1																

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1																									

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1	1	0	1													

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1 0 0 1

0 1 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 0 1

0 0 1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1	1	0														

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	0	1

PCP: complex example

index	x_i	y_i
1	001	0
2	01	011
3	01	101
4	10	001

shortes solution: 66 indices long

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	0	1
0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0
1	1	0	0	1	1	0	0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0
1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0
1	0	0	1	1	0	0	0	1	0	0	1	0	1														

0	1	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	0	0	0	1
0	0	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	0	1	0
1	1	0	0	1	1	0	0	0	1	0	1	0	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0
1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0	1	0	0	1	0
1	0	0	1	1	0	0	0	1	0	0	1	0	1														

Given: A finite set of word pairs $(x_1, y_1), \dots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \dots, i_n \in \{1, 2, \dots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \dots x_{i_n} = y_{i_1} y_{i_2} \dots y_{i_n}$

Modified Post's Correspondence Problem (MPCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

MPCP

$\Sigma = \{0, 1\}$

index	x_i	y_i
1	100	10
2	10	01
3	11	111

$\xrightarrow{f}$

PCP

$\Sigma = \{0, 1\} \cup \{\#, \$\}\}$

index	x_i	y_i
1	#1#0#0#	#1#0
2	1#0#	#0#1
3	1#1#	#1#1#1)
4	&	#&

1	0	0
1	0	

#	1	#	0	#	0	#
#	1	#	0			

$p \in MPCP \Leftrightarrow f(p) \in PCP$

$MPCP \leq PCP$

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

MPCP

$\Sigma = \{0, 1\}$

index	x_i	y_i
1	100	10
2	10	01
3	11	111

$\xrightarrow{f}$

PCP

$\Sigma = \{0, 1\} \cup \{\#, \$\}\}$

index	x_i	y_i
1	#1#0#0#	#1#0
2	1#0#	#0#1
3	1#1#	#1#1#1)
4	&	#&

1	0	0	1	0
1	0	0	1	

#	1	#	0	#	0	#	1	#	0	#
#	1	#	0	#	0	#	1			

$p \in MPCP \Leftrightarrow f(p) \in PCP$

$MPCP \leq PCP$

Modified Post's Correspondence Problem (MPCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

MPCP

$\Sigma = \{0, 1\}$

index	x_i	y_i
1	100	10
2	10	01
3	11	111

$\xrightarrow{f}$

PCP

$\Sigma = \{0, 1\} \cup \{\#, \$\}\}$

index	x_i	y_i
1	#1#0#0#	#1#0
2	1#0#	#0#1
3	1#1#	#1#1#1)
4	&	#&

1	0	0	1	0	1	1
1	0	0	1	1	1	1

#	1	#	0	#	0	#	1	#	0	#	1	#	1	#
#	1	#	0	#	0	#	1	#	1	#	1	#	1	

$p \in MPCP \Leftrightarrow f(p) \in PCP$

$MPCP \leq PCP$

Modified Post's Correspondence Problem (MPCP)

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

MPCP

$\Sigma = \{0, 1\}$

index	x_i	y_i
1	100	10
2	10	01
3	11	111

$\xrightarrow{f}$

PCP

$\Sigma = \{0, 1\} \cup \{\#, \$\}\}$

index	x_i	y_i
1	#1#0#0#	#1#0
2	1#0#	#0#1
3	1#1#	#1#1#1)
4	&	#&

1	0	0	1	0	1	1
1	0	0	1	1	1	1

#	1	#	0	#	0	#	1	#	0	#	1	#	1	#	&
#	1	#	0	#	0	#	1	#	1	#	1	#	1	#	&

$p \in MPCP \Leftrightarrow f(p) \in PCP$

$MPCP \leq PCP$

Given: A finite set of word pairs $(x_1, y_1), \ldots (x_k, y_k)$, with $x_i, y_i \in \Sigma^+$.

Question: Is there a sequence of indices $i_1, i_2, \ldots, i_n \in \{1, 2, \ldots, k\}$ with $i_1 = 1$ such that $x_{i_1} x_{i_2} \ldots x_{i_n} = y_{i_1} y_{i_2} \ldots y_{i_n}$

MPCP

$\Sigma = \{0, 1\}$

index	x_i	y_i
1	100	10
2	10	01
3	11	111

$\xrightarrow{f}$

PCP

$\Sigma = \{0, 1\} \cup \{\#, \$\}\}$

index	x_i	y_i
1	#1#0#0#	#1#0
2	1#0#	#0#1
3	1#1#	#1#1#1)
4	&	#&

1	0	0	1	0	1	1
1	0	0	1	1	1	1

#	1	#	0	#	0	#	1	#	0	#	1	#	1	#	&
#	1	#	0	#	0	#	1	#	1	#	1	#	1	#	&

$p \in MPCP \Leftrightarrow f(p) \in PCP$

$MPCP \leq PCP$

The MPCP is undecidable, proof by $H \leq MPCP$

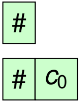
- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.

The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

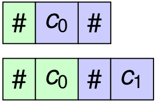
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \to MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

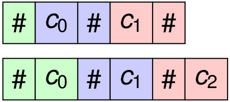
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \dots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

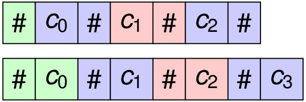
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
$\vdots$	$\vdots$	$\vdots$
n	c_f #	#
$\vdots$	$\vdots$	$\vdots$



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

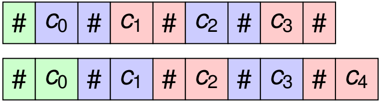
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

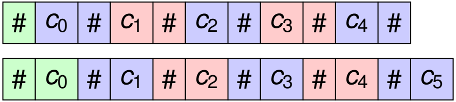
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

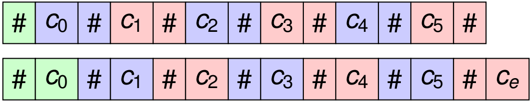
index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \to MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮



The MPCP is undecidable, proof by $H \leq MPCP$

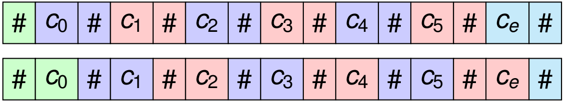
- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

index	x_i	y_i
1	#	# c_0
2	c_i#	#c_{i+1}
⋮	⋮	⋮
n	c_f #	#
⋮	⋮	⋮

#	c_0	#	c_1	#	c_2	#	c_3	#	c_4	#	c_5	#	c_e	#
#	c_0	#	c_1	#	c_2	#	c_3	#	c_4	#	c_5	#	c_e	#

- To prove $H \leq MPCP$ we need a computable reduction function $f : H \rightarrow MPCP$ such that $G(M) \in H \Leftrightarrow f(M) \in MPCP$.
- A machine-word pair (M, w) is an instance of H, i.e. $G(M)\#w \in H$, iff there is a sequence of configurations $c_0, c_1, c_2 \ldots c_f$ with $c_0 = q_0 w$, $c_i \Rightarrow c_{i+1}$, and c_f has a final state.
- The idea is to code this into a *MPCP* problem:

index	x_i	y_i
1	#	$\# \, c_0$
2	$c_i\#$	$\#c_{i+1}$
⋮	⋮	⋮
n	$c_f \, \#$	#
⋮	⋮	⋮



Be careful, this only shows the main idea. We are oversimplifying here as neither the set of $c_i \Rightarrow c_{i+1}$ nor the set of c_f's needs to be finite.
For a formal proof see Hopcroft & Ullman 1979.

Proposition

PCP restricted to words over the alphabet $\{0, 1\}$ is undecidable.

- Given a PCP instance p over an alphabet $\{a_1, \ldots a_k\}$ construct a PCP instance p' over $\{0, 1\}$ by replacing every a_i by 01^i.

PCP restricted to $\{0, 1\}$

Proposition

PCP restricted to words over the alphabet $\{0, 1\}$ is undecidable.

- Given a PCP instance p over an alphabet $\{a_1, \ldots a_k\}$ construct a PCP instance p' over $\{0, 1\}$ by replacing every a_i by 01^i.
- $p \in PCP \Leftrightarrow p' \in PCP$

PCP restricted to $\{0, 1\}$

Proposition

PCP restricted to words over the alphabet $\{0, 1\}$ is undecidable.

- Given a PCP instance p over an alphabet $\{a_1, \dots a_k\}$ construct a PCP instance p' over $\{0, 1\}$ by replacing every a_i by 01^i.
- $p \in PCP \Leftrightarrow p' \in PCP$

$\Rightarrow$ $PCP \leq PCP_{\{0,1\}}$

Proposition

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- *Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)*
- *Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)*
- *Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)*
- *Is $L(G_1) \subseteq L(G_2)$? ($GP_{\subseteq}$)*
- *Is $L(G_1) = L(G_2)$? ($GP_{=}$)*

Proposition

Given a context-free grammars G, the following problems are undecidable:

- *Is G ambiguous?*
- *Is $\overline{L(G)}$ infinite?*
- *Is $L(G_1) \cap L(G_2)$ context-free?*
- *Is $L(G)$ regular?*

Encode PCPs as grammars

Given a PCP instance $\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ over $\{0, 1\}$, construct two grammars

$$G_1: \begin{array}{lcl} S & \rightarrow & A\$B \\ A & \rightarrow & i_1 A\, x_1 \,|\, \ldots \,|\, i_k A\, x_k \\ A & \rightarrow & i_1 x_1 \,|\, \ldots \,|\, i_k x_k \\ B & \rightarrow & y_1^R B\, i_1 \,|\, \ldots \,|\, y_k^R B\, i_k \\ B & \rightarrow & y_1^R i_1 \,|\, \ldots \,|\, y_k^R i_k \end{array}$$

$$G_2: \begin{array}{lcl} S & \rightarrow & i_1 S\, i_1 \,|\, \ldots \,|\, i_k S\, i_k \,|\, T \\ T & \rightarrow & 0T0 | 1T1 | \$ \end{array}$$

Encode PCPs as grammars

Given a PCP instance $\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ over $\{0, 1\}$, construct two grammars

$$G_1: \begin{array}{lcl} S & \to & A\$B \\ A & \to & i_1 A x_1 \mid \ldots \mid i_k A x_k \\ A & \to & i_1 x_1 \mid \ldots \mid i_k x_k \\ B & \to & y_1^R B i_1 \mid \ldots \mid y_k^R B i_k \\ B & \to & y_1^R i_1 \mid \ldots \mid y_k^R i_k \end{array}$$

$$G_2: \begin{array}{lcl} S & \to & i_1 S i_1 \mid \ldots \mid i_k S i_k \mid T \\ T & \to & 0T0 \mid 1T1 \mid \$ \end{array}$$

Grammar G_1 generates words of the form

$$i_{n_1}\, i_{n_2} \cdots i_{n_k}\, x_{n_k} \cdots x_{n_2}\, x_{n_1}\ \$\ y_{m_1}^R\, y_{m_2}^R \cdots y_{m_j}^R\, i_{m_j} \cdots i_{m_2}\, i_{m_1}$$

Grammar G_2 generates words of the form

$$i_{n_1}\, i_{n_2} \cdots i_{n_k}\ 1\,1\,0 \cdots 1\ \$\ 1 \cdots 0\,1\,1\ i_{n_k} \cdots i_{n_2}\, i_{n_1}$$

Encode PCPs as grammars

Given a PCP instance $\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ over $\{0, 1\}$, construct two grammars

$$G_1: \begin{array}{lcl} S & \to & A\$B \\ A & \to & i_1 A x_1 | \ldots | i_k A x_k \\ A & \to & i_1 x_1 | \ldots | i_k x_k \\ B & \to & y_1^R B i_1 | \ldots | y_k^R B i_k \\ B & \to & y_1^R i_1 | \ldots | y_k^R i_k \end{array}$$

$$G_2: \begin{array}{lcl} S & \to & i_1 S i_1 | \ldots | i_k S i_k | T \\ T & \to & 0T0 | 1T1 | \$ \end{array}$$

Grammar G_1 generates words of the form

$$i_{n_1}\ i_{n_2} \cdots i_{n_k}\ x_{n_k} \cdots x_{n_2}\ x_{n_1}\ \$\ y_{m_1}^R\ y_{m_2}^R \cdots y_{m_j}^R\ i_{m_j} \cdots i_{m_2}\ i_{m_1}$$

Grammar G_2 generates words of the form

$$i_{n_1}\ i_{n_2} \cdots i_{n_k}\ 1\,1\,0 \cdots 1\,\$\,1 \cdots 0\,1\,1\ i_{n_k} \cdots i_{n_2}\ i_{n_1}$$

$L(G_1) \cap L(G_2)$ consists of words of the form:

$i_{n_1} \ldots i_{n_k} v\$v^R i_{n_k} \ldots i_{n_1}$ with $v = x_{n_1} \ldots x_{n_k}$ and $v^R = y_{n_k}^R \ldots y_{n_1}^R$

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \dots i_{n_k} v\$v^R i_{n_k} \dots i_{n_1}$ with $v = x_{n_1} \dots x_{n_k}$ and $v^R = y^R_{n_k} \dots y^R_{n_1}$

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \dots i_{n_k} v\$v^R i_{n_k} \dots i_{n_1}$ with $v = x_{n_1} \dots x_{n_k}$ and $v^R = y^R_{n_k} \dots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \ldots i_{n_k} v\$v^R i_{n_k} \ldots i_{n_1}$ with $v = x_{n_1} \ldots x_{n_k}$ and $v^R = y^R_{n_k} \ldots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

$\Rightarrow$ $PCP \leq GP_{\cap,\emptyset}$, the problem whether $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \dots i_{n_k} v\$v^R i_{n_k} \dots i_{n_1}$ with $v = x_{n_1} \dots x_{n_k}$ and $v^R = y^R_{n_k} \dots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

$\Rightarrow$ *$PCP \leq GP_{\cap,\emptyset}$*, the problem whether $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.

- If a PCP instance has one solution it has infinitely many solutions.

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \ldots i_{n_k} v\$v^R i_{n_k} \ldots i_{n_1}$ with $v = x_{n_1} \ldots x_{n_k}$ and $v^R = y^R_{n_k} \ldots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

$\Rightarrow$ $PCP \leq GP_{\cap,\emptyset}$, the problem whether $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.

- If a PCP instance has one solution it has infinitely many solutions.

$\Rightarrow$ $PCP \leq GP_{\cap,\infty}$ the problem whether $L(G_1) \cap L(G_2)$ is infinite is undecidable.

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \dots i_{n_k} v\$v^R i_{n_k} \dots i_{n_1}$ with $v = x_{n_1} \dots x_{n_k}$ and $v^R = y^R_{n_k} \dots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

⇒ $PCP \leq GP_{\cap,\emptyset}$, the problem whether $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.

- If a PCP instance has one solution it has infinitely many solutions.

⇒ $PCP \leq GP_{\cap,\infty}$ the problem whether $L(G_1) \cap L(G_2)$ is infinite is undecidable.

- If $L(G_1) \cap L(G_2) \neq \emptyset$ then $L(G_1) \cap L(G_2)$ is not context-free (Pumping-Lemma).

Undecidable grammar problems (proofs)

to prove:

Given two context-free grammars G_1, G_2, the following problems are undecidable:

- Is $L(G_1) \cap L(G_2) = \emptyset$? ($GP_{\cap,\emptyset}$)
- Is $L(G_1) \cap L(G_2)$ infinite? ($GP_{\cap,\infty}$)
- Is $L(G_1) \cap L(G_2)$ context-free? ($GP_{\cap,CF}$)

- Recall, $L(G_1) \cap L(G_2)$ consists of words of the form: $i_{n_1} \dots i_{n_k} v\$v^R i_{n_k} \dots i_{n_1}$ with $v = x_{n_1} \dots x_{n_k}$ and $v^R = y^R_{n_k} \dots y^R_{n_1}$
- Hence, the PCP instance $\{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$ has a solution if and only if $L(G_1) \cap L(G_2) \neq \emptyset$.

$\Rightarrow$ $PCP \leq GP_{\cap,\emptyset}$, the problem whether $L(G_1) \cap L(G_2) = \emptyset$ is undecidable.

- If a PCP instance has one solution it has infinitely many solutions.

$\Rightarrow$ $PCP \leq GP_{\cap,\infty}$ the problem whether $L(G_1) \cap L(G_2)$ is infinite is undecidable.

- If $L(G_1) \cap L(G_2) \neq \emptyset$ then $L(G_1) \cap L(G_2)$ is not context-free (Pumping-Lemma).

$\Rightarrow$ $PCP \leq GP_{\cap,CF}$, the problem whether $L(G_1) \cap L(G_2)$ is context-free is undecidable.

Undecidable grammar problems (proofs)

Proposition

Deterministic context-free grammars are closed under complement.
There is a computable function f such that for each context-free grammar G, $f(G)$ is a context-free grammar with $\overline{L(G)} = L(f(G))$

For a proof see Hopcroft & Ullman 1979.

Undecidable grammar problems (proofs)

Proposition

Deterministic context-free grammars are closed under complement.
There is a computable function f such that for each context-free grammar G, $f(G)$ is a context-free grammar with $\overline{L(G)} = L(f(G))$

For a proof see Hopcroft & Ullman 1979.

to prove:

Given two context-free grammars G, G', the following problems are undecidable:

- Is $L(G) \subseteq L(G')$? ($GP_{\subseteq}$)
- Is $L(G) = L(G')$? ($GP_{=}$)

Proposition

Deterministic context-free grammars are closed under complement.
There is a computable function f such that for each context-free grammar G, $f(G)$ is a context-free grammar with $\overline{L(G)} = L(f(G))$

For a proof see Hopcroft & Ullman 1979.

to prove:

Given two context-free grammars G, G', the following problems are undecidable:

- Is $L(G) \subseteq L(G')$? ($GP_{\subseteq}$)
- Is $L(G) = L(G')$? ($GP_{=}$)

- Note that the grammars G_1 and G_2 are deterministic.
- $L(G_1) \cap L(G_2) = \emptyset$ if and only if $L(G_1) \subseteq \overline{L(G_2)}$

$\Rightarrow$ $GP_{\cap,\emptyset} \leq GP_{\subseteq}$, the problem whether $L(G) \subseteq L(G')$ is undecidable.

Proposition

Deterministic context-free grammars are closed under complement.
There is a computable function f such that for each context-free grammar G, $f(G)$ is a context-free grammar with $\overline{L(G)} = L(f(G))$

For a proof see Hopcroft & Ullman 1979.

to prove:

Given two context-free grammars G, G', the following problems are undecidable:

- Is $L(G) \subseteq L(G')$? ($GP_{\subseteq}$)
- Is $L(G) = L(G')$? ($GP_{=}$)

- Note that the grammars G_1 and G_2 are deterministic.
- $L(G_1) \cap L(G_2) = \emptyset$ if and only if $L(G_1) \subseteq \overline{L(G_2)}$

$\Rightarrow$ $GP_{\cap,\emptyset} \leq GP_{\subseteq}$, the problem whether $L(G) \subseteq L(G')$ is undecidable.

- $L(G) \subseteq L(G')$ if and only if $L(G) \cup L(G') = L(G')$.

$\Rightarrow$ the problem whether $L(G) = L(G')$ is undecidable.

Undecidable grammar problems (proofs)

Given a context-free grammar G, the following problems are undecidable:

- Is G ambiguous? (GP_{amb})
- Is $\overline{L(G)}$ context-free? ($GP_{\overline{CF}}$)
- Is $L(G)$ regular? (GP_{reg})

Undecidable grammar problems (proofs)

Given a context-free grammar G, the following problems are undecidable:

- Is G ambiguous? (GP_{amb})
- Is $\overline{L(G)}$ context-free? ($GP_{\overline{CF}}$)
- Is $L(G)$ regular? (GP_{reg})

- Let G_1 and G_2 be as before. Let G_3 be the grammar which generates $L(G_1) \cup L(G_2)$.
  - The instance of the PCP problem has a solution iff there exists a word $w \in L(G_3)$ which has two derivation trees (one from G_1 and one from G_2).
  - $\Rightarrow$ $PCP \leq GP_{amb}$, the problem whether a context-free grammar is ambiguous is undecidable.

Undecidable grammar problems (proofs)

Given a context-free grammar G, the following problems are undecidable:

- Is G ambiguous? (GP_{amb})
- Is $\overline{L(G)}$ context-free? ($GP_{\overline{CF}}$)
- Is $L(G)$ regular? (GP_{reg})

- Let G_1 and G_2 be as before. Let G_3 be the grammar which generates $L(G_1) \cup L(G_2)$.
  - The instance of the PCP problem has a solution iff there exists a word $w \in L(G_3)$ which has two derivation trees (one from G_1 and one from G_2).
  - $\Rightarrow$ $PCP \leq GP_{amb}$, the problem whether a context-free grammar is ambiguous is undecidable.
- Remember, G_1 and G_2 are deterministic and $f(G_1)$, $f(G_2)$ generate the complement languages. Let G_4 be the grammar which generates $L(G_4) = L(f(G_1)) \cup L(f(G_2)) = \overline{L(G_1)} \cup \overline{L(G_2)} = \overline{L(G_1) \cap L(G_2)}$
  - The instance of the PCP problem has a solution iff $L(G_1) \cap L(G_2) = \overline{L(G_4)}$ is not context-free.
  - $\Rightarrow$ $GP_{\cap,CF} \leq GP_{\overline{CF}}$ The problem whether the complement of a context-free language is context-free is undecidable.

Undecidable grammar problems (proofs)

Given a context-free grammar G, the following problems are undecidable:

- Is G ambiguous? (GP_{amb})
- Is $\overline{L(G)}$ context-free? ($GP_{\overline{CF}}$)
- Is $L(G)$ regular? (GP_{reg})

- Let G_1 and G_2 be as before. Let G_3 be the grammar which generates $L(G_1) \cup L(G_2)$.
  - The instance of the PCP problem has a solution iff there exists a word $w \in L(G_3)$ which has two derivation trees (one from G_1 and one from G_2).

  $\Rightarrow$ $PCP \leq GP_{amb}$, the problem whether a context-free grammar is ambiguous is undecidable.
- Remember, G_1 and G_2 are deterministic and $f(G_1)$, $f(G_2)$ generate the complement languages. Let G_4 be the grammar which generates $L(G_4) = L(f(G_1)) \cup L(f(G_2)) = \overline{L(G_1)} \cup \overline{L(G_2)} = \overline{L(G_1) \cap L(G_2)}$
  - The instance of the PCP problem has a solution iff $L(G_1) \cap L(G_2) = \overline{L(G_4)}$ is not context-free.

  $\Rightarrow$ $GP_{\cap,CF} \leq GP_{\overline{CF}}$ The problem whether the complement of a context-free language is context-free is undecidable.
- $L(G_1) \cap L(G_2) = \emptyset$ iff $L(G_4) = \Sigma^*$. Remember: For regular languages it is easy to check whether $L = \Sigma^*$.

  $\Rightarrow$ $GP_{\cap,\emptyset} \leq GP_{reg}$ The problem whether a context-free grammar generates a regular language is undecidable.