



# Komplexitätstheorie

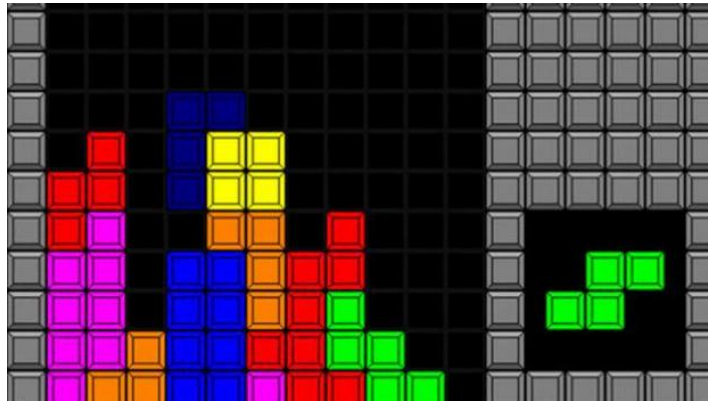
Prof. Dr. Björn Grohmann



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# Ordne nach Schwierigkeit



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Prime Number				
2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

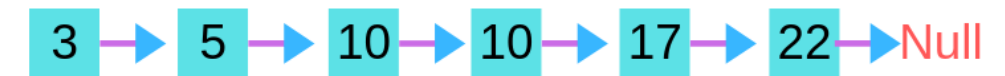


$$\begin{array}{r}
 588 \\
 \swarrow \quad \searrow \\
 2 \quad 294 \\
 \swarrow \quad \searrow \\
 2 \quad 147 \\
 \swarrow \quad \searrow \\
 3 \quad 49 \\
 \swarrow \quad \searrow \\
 7 \quad 7
 \end{array}$$

$$588 = 2^2 \cdot 3 \cdot 7^2$$



Original List



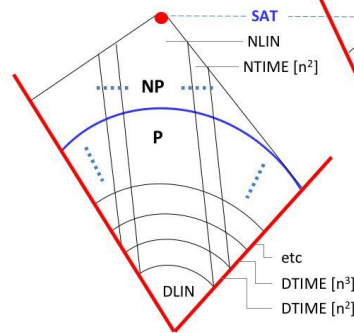
Sorted List

A complete language for EXPSpace:  
PIM, "Polynomial Ideal Membership"—the simplest natural completeness level that is known not to have polynomial-size circuits.

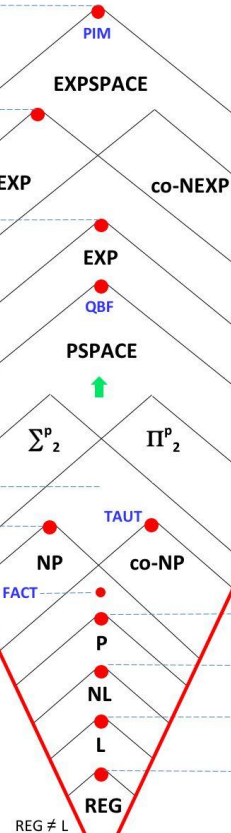
Succinct 3SAT

nxn Chess

For any fixed  $k$ , there is a problem in this intersection that can NOT be solved by circuits of size  $O(n^k)$

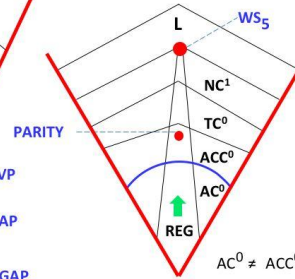


**A** Deterministic and Nondeterministic Time Hierarchies Within NP



**B** Complexity "Main Sequence"

$WS_5$ , the word problem for the symmetric group  $S_5$ , is a regular language that is complete for  $NC^1$  under  $AC^0$  many-one reductions.



**C** Low-Level Classes

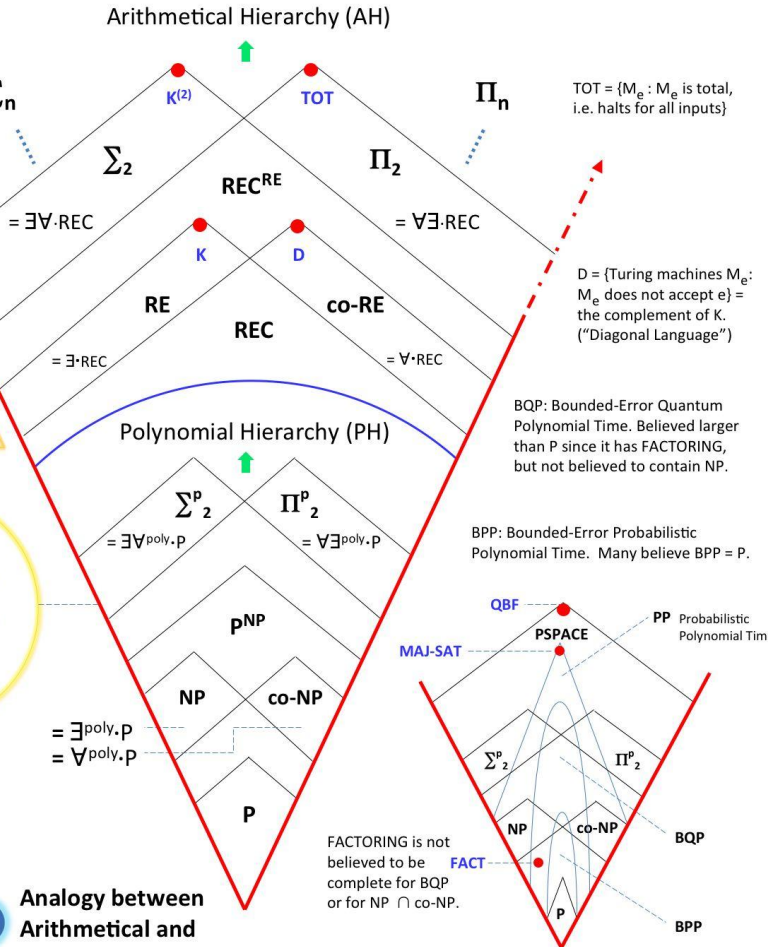
Unknown but Commonly Believed:

- $L \neq NL$  .....  $L \neq PH$
- $P \neq NP \cap co-NP$  .....  $P \neq PSPACE$
- $NP \neq \Sigma_2^P \cap \Pi_2^P$  .....  $NP \neq EXP$

Best Known Separations:

- $AC^0 \subset ACC^0 \subset PP$ , also  $TC^0 \subset PP$
- $NC^1 \subset PSPACE$ , ... ,  $NL \subset PSPACE$
- $P \subset EXP$ ,  $NP \subset NEXP$
- $PSPACE \subset EXPSpace$

The levels of AH and PH are analogous, except that we believe  $NP \cap co-NP \neq P$  and  $\Sigma_2^P \cap \Pi_2^P \neq P^{NP}$ , which stand in contrast to  $RE \cap co-RE = REC$  and  $\Sigma_2 \cap \Pi_2 = REC^{RE}$



**D** Analogy between Arithmetical and Polynomial Hierarchies

FACT is not believed to be complete for BQP or for  $NP \cap co-NP$ .

**E** Realm of Feasibility?

# Complexity Zoo

## Introduction

Welcome to the **Complexity Zoo**... There are now 545 classes and counting!

*Complexity classes by letter:* [Symbols](#) - [A](#) - [B](#) - [C](#) - [D](#) - [E](#) - [F](#) - [G](#) - [H](#) - [I](#) - [J](#) - [K](#) - [L](#) - [M](#) - [N](#) - [O](#) - [P](#) - [Q](#) - [R](#) - [S](#) - [T](#) - [U](#) - [V](#) - [W](#) - [X](#) - [Y](#) - [Z](#)

*Lists of related classes:* [Communication Complexity](#) - [Hierarchies](#) - [Nonuniform](#)

### Zookeeper

[Scott Aaronson](#)

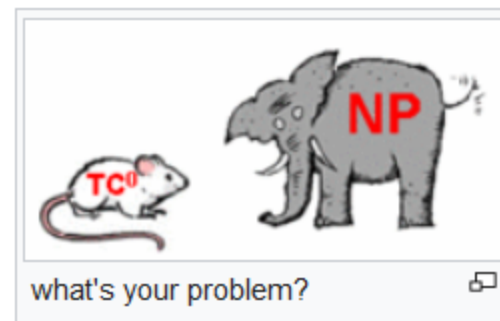
### Veterinarian

[Greg Kuperberg](#)

### Zoo Conservationist

[Oliver Habryka](#) on behalf of the [LessWrong](#) community

The Zoo first opened in 2002. It was made into a wiki in 2005, and hosted at the University of Waterloo from 2012 to 2020.



# Rule 110



current automaton contents



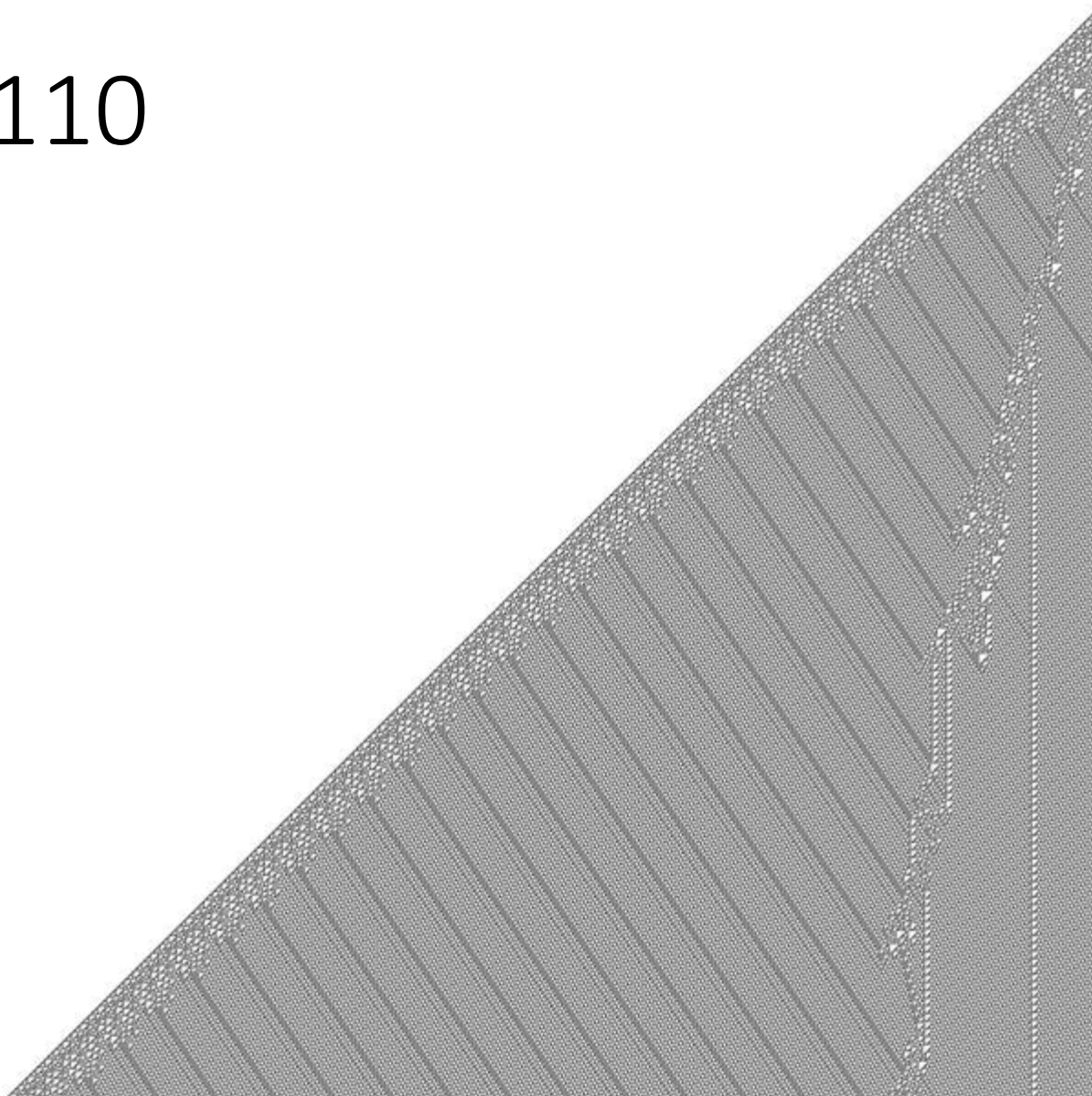
rule 110 (01101110)



the next generation of the automaton



# Rule 110





# WHILE Programme

$$\begin{array}{l} P \rightarrow X := X + C \\ | X := X - C \\ | P; P \\ | \text{WHILE } X \neq 0 \text{ DO } P \text{ END} \end{array}$$

# GOTO Programme



$M_1 : A_1; M_2 : A_2; \dots; M_k : A_k$

$x_i := x_j + n$

$x_i := x_j - n$

GOTO  $M_i$

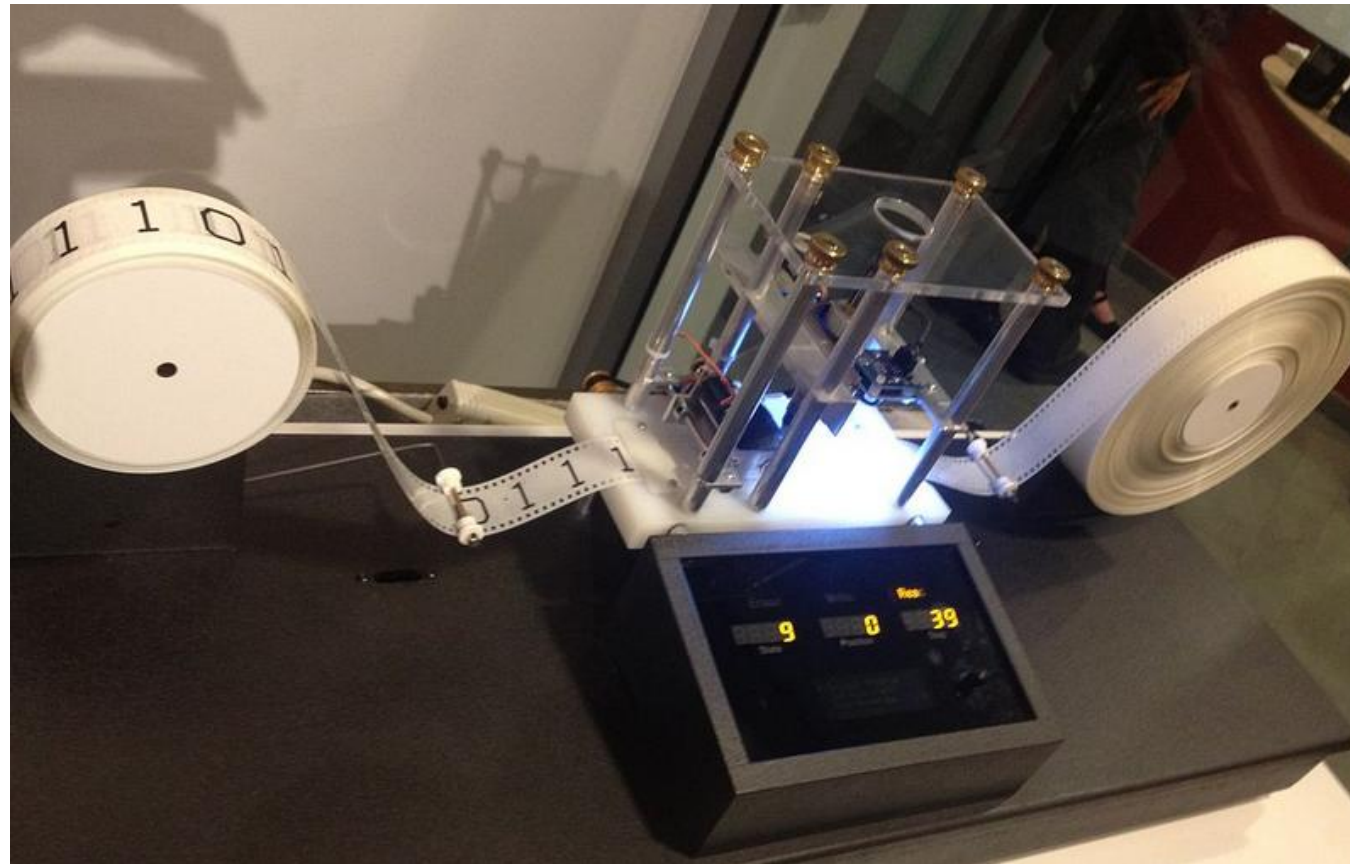
IF  $x_i = n$  GOTO  $M_j$

HALT

# Turing Maschine



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# Turing Maschine



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

DIED JUNE 8, 1954

PRINCETON UNIVERSITY THE GRADUATE SCHOOL

TURING, ALAN MATHISON Enrolled 9/29/36 Department MATHEMATICS

Date and place of birth June 23, 1912 (Paddington, London) Single ☒ Married

Bachelor and other degrees B.A. University of Cambridge, 1934; Ph.D. Princeton University, 1938

Previous graduate study 1934 (July) to August 1936 University of Cambridge

Teaching experience Jan. 1935 to June 1936 Supervisor of Undergraduates, University of Cambridge

Address: Princeton 183 G.C.; 182 G.C.

Parent or Guardian and address Mr. J. M. Turing, 8 Ennismore Ave., Guildford, England.

1936-37 Fellow from King's College

1937-38 Jane Eliza Procter Visiting Fellow in Mathematics

1938- Fellow at King's College, Cambridge

A. M. or M. F. A. [21-803] Degree granted

Address

PH. D.

French Satisfactory May 20, 1937 German satisfactory May 20, 1937

General Examination Passed May 26, 1937

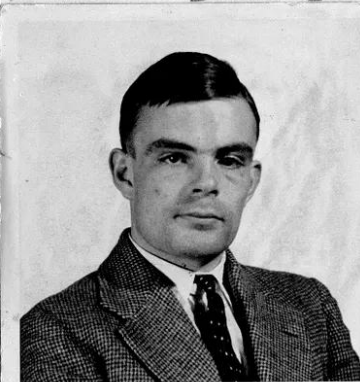
Dissertation Subject "Systems of Logic Based on Ordinals".

Dissertation accepted May 18, 1938 Published under

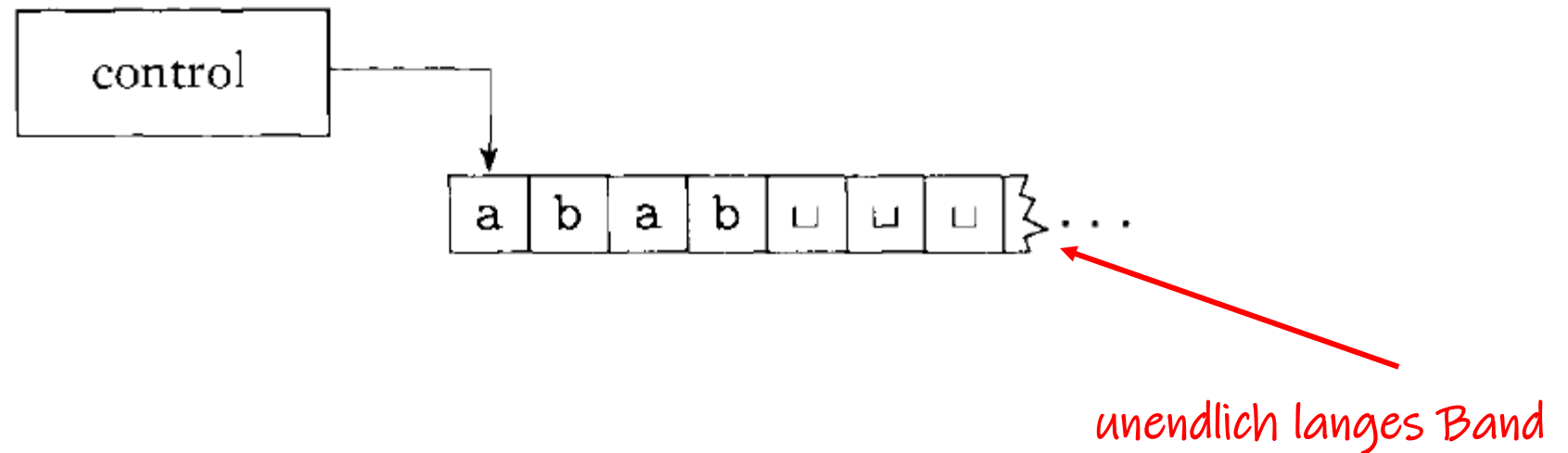
Published 1939. Printed by C.F. Hodgson and Son, Ltd., 2 Newton St., London, W.C.2, England. Copies sent University Library 1939.

Final Examination Passed May 27, 1938 Degree granted June 21, 1938

Diploma address



# Turing Maschine



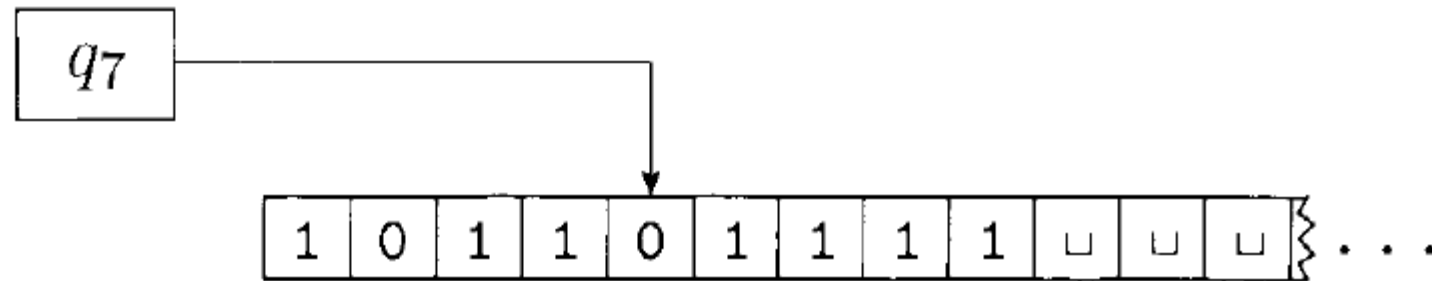


# Turing Maschine

A **Turing machine** is a 7-tuple,  $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ , where  $Q, \Sigma, \Gamma$  are all finite sets and

1.  $Q$  is the **set of states**,
2.  $\Sigma$  is the **input alphabet** not containing the *blank symbol*  $\sqcup$ ,
3.  $\Gamma$  is the **tape alphabet**, where  $\sqcup \in \Gamma$  and  $\Sigma \subseteq \Gamma$ ,
4.  $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$  is the **transition function**,
5.  $q_0 \in Q$  is the **start state**,
6.  $q_{\text{accept}} \in Q$  is the **accept state**, and
7.  $q_{\text{reject}} \in Q$  is the **reject state**, where  $q_{\text{reject}} \neq q_{\text{accept}}$ .

# Turing Maschine

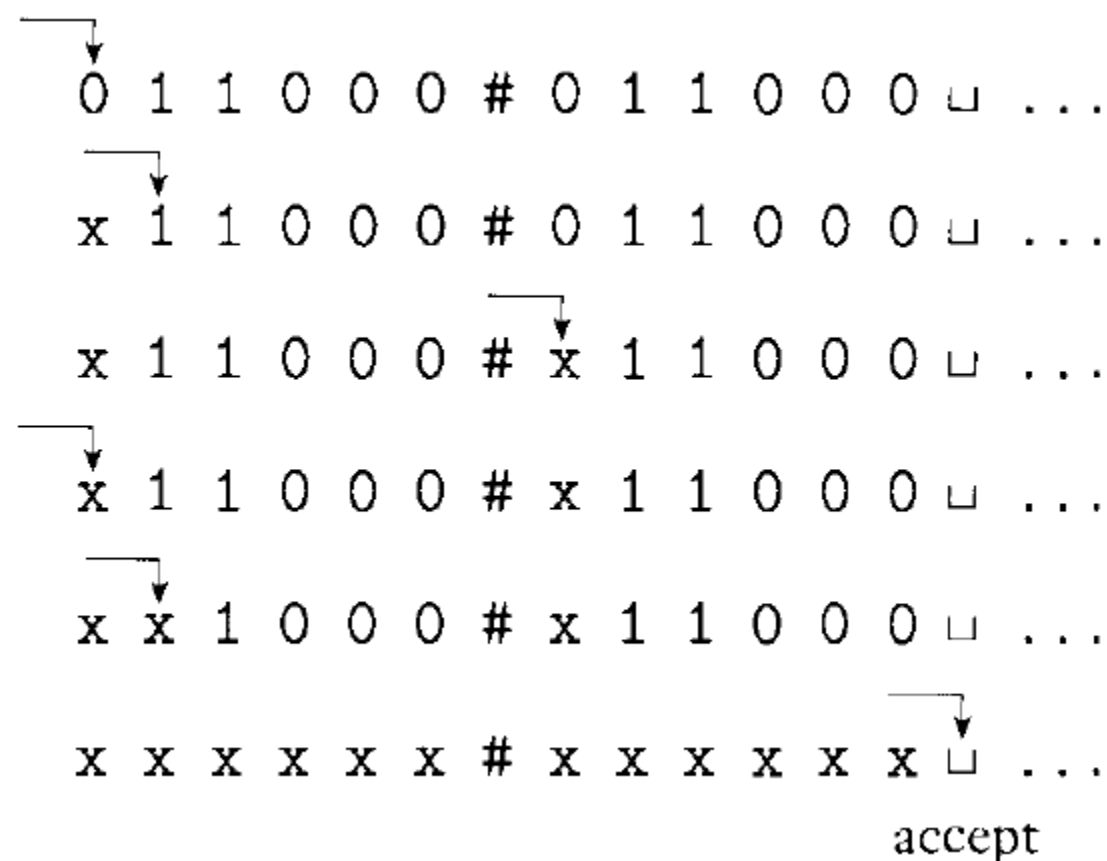


A Turing machine with **configuration**  $1011q_701111$



# Turing Maschine

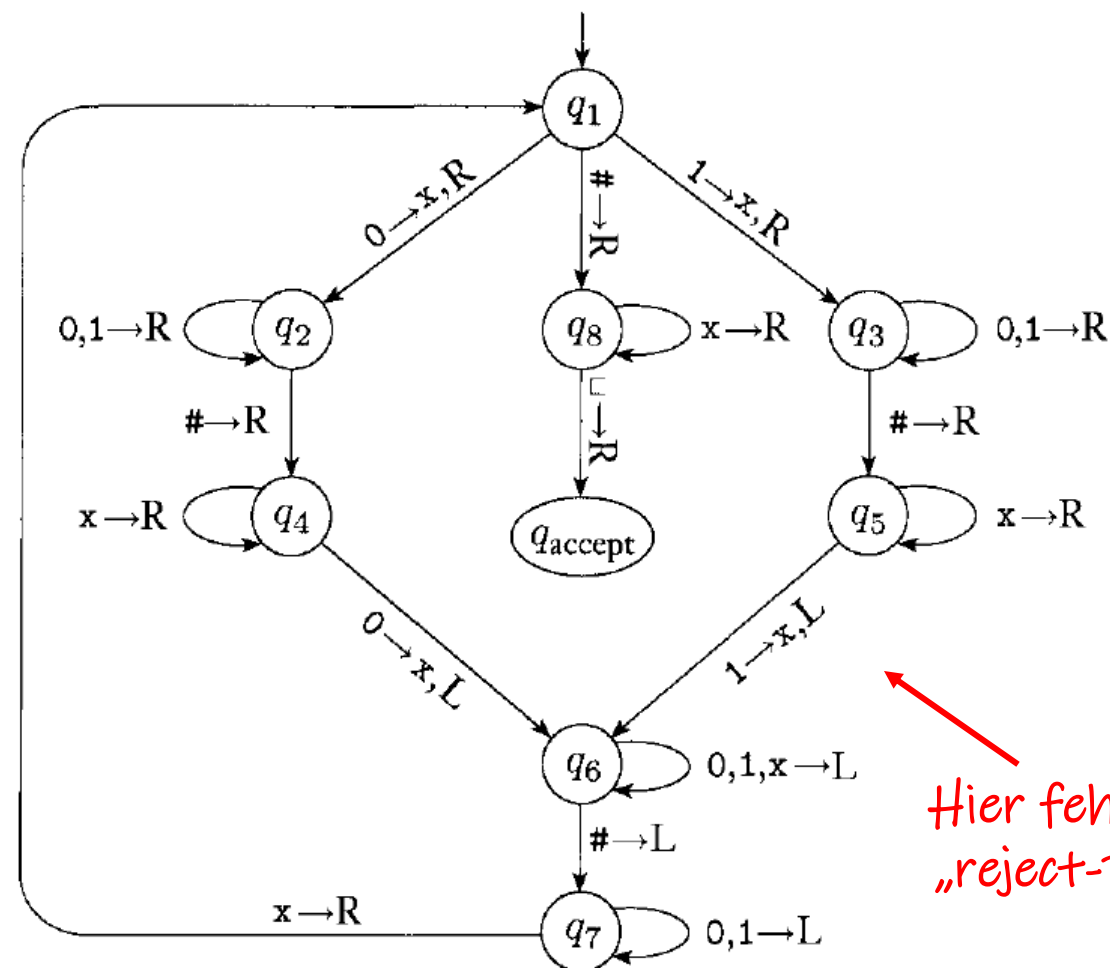
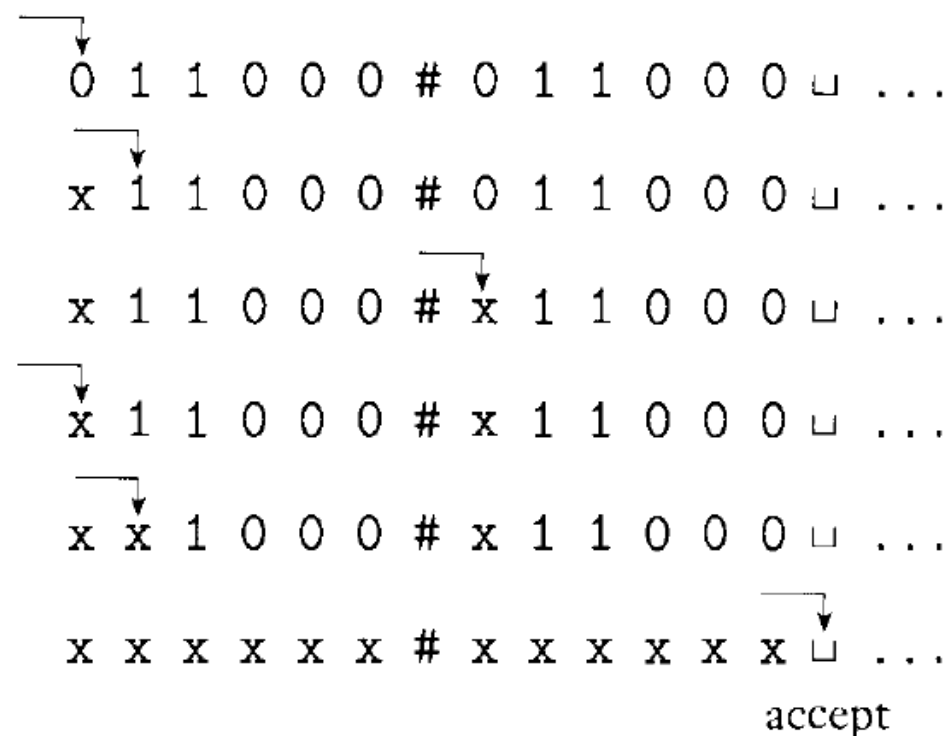
Wir suchen eine Turing Maschine, welche folgende Sprache akzeptiert:  $B = \{w\#w \mid w \in \{0,1\}^*\}$





# Turing Maschine

Wir suchen eine Turing Maschine, welche folgende Sprache akzeptiert:  $B = \{w\#w \mid w \in \{0,1\}^*\}$



Hier fehlen die  
„reject-Pfade“

# Church-Turing These



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

„Jede **effektiv** berechenbare  
Funktion kann durch eine  
**Turing-Maschine** berechnet werden“



# Turing Maschine

Let  $M$  be a deterministic Turing machine that halts on all inputs. The **running time** or **time complexity** of  $M$  is the function  $f: \mathcal{N} \rightarrow \mathcal{N}$ , where  $f(n)$  is the **maximum number of steps** that  $M$  uses on any input of length  $n$ . If  $f(n)$  is the running time of  $M$ , we say that  $M$  runs in time  $f(n)$  and that  $M$  is an  $f(n)$  time Turing machine. Customarily we use  $n$  to represent the length of the input.



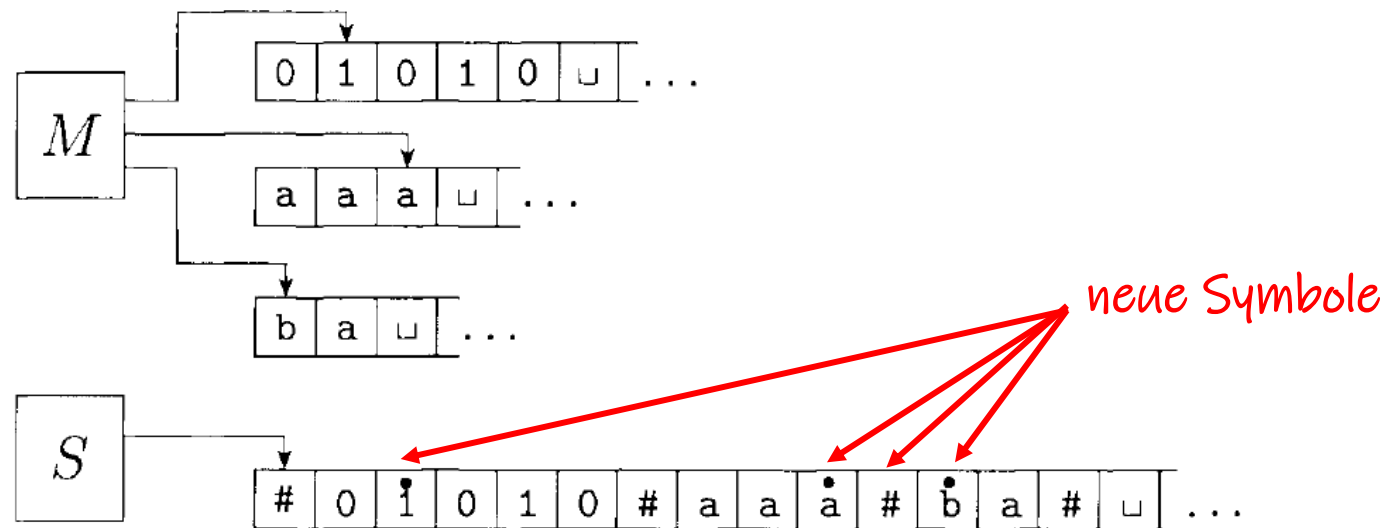
# Turing Maschine

„Multitape Turing Machine“

$$\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

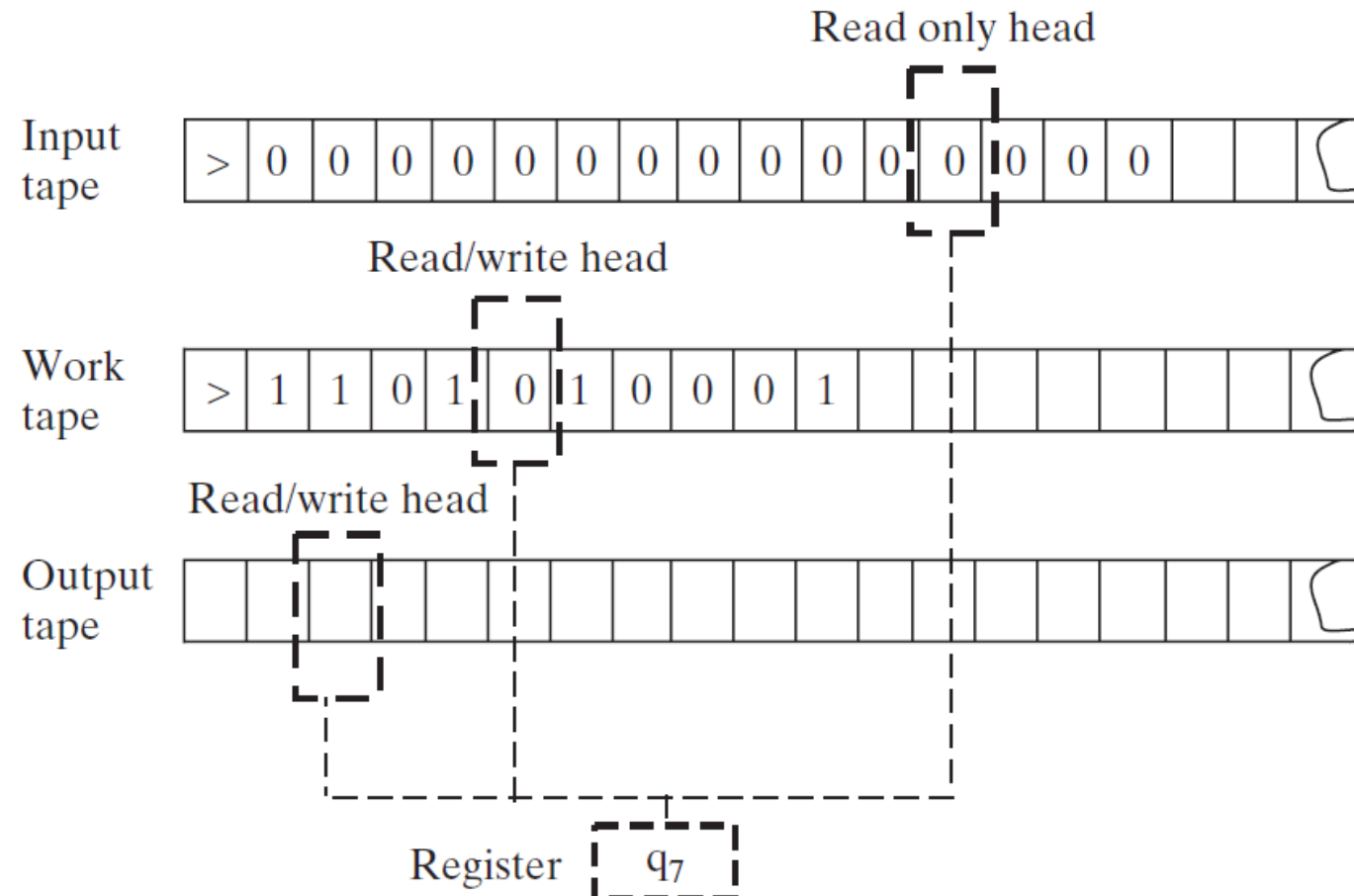
## THEOREM

Every multitape Turing machine has an equivalent single-tape Turing machine.



# Turing Maschine

## „Multitape Turing Machine“



# Turing Maschine

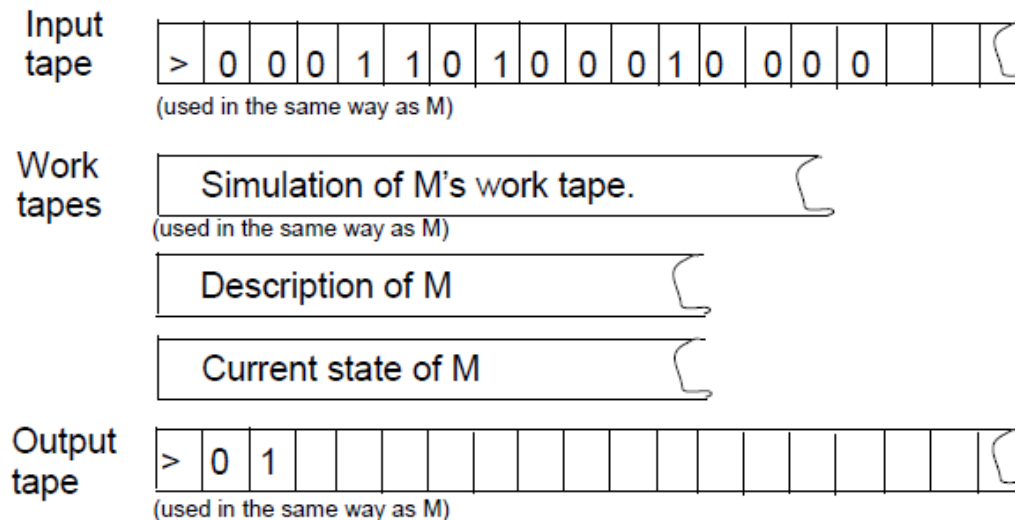


## „Universal Turing Machine“

There exists a TM  $\mathcal{U}$  such that for every  $x, \alpha \in \{0, 1\}^*$ ,  $\mathcal{U}(x, \alpha) = M_\alpha(x)$ , where  $M_\alpha$  denotes the TM represented by  $\alpha$ .

Furthermore, if  $M_\alpha$  halts on input  $x$  within  $T$  steps then  $\mathcal{U}(x, \alpha)$  halts within  $CT \log T$  steps, where  $C$  is a number independent of  $|x|$  and depending only on  $M_\alpha$ 's alphabet size, number of tapes, and number of states.

Gödelnummer



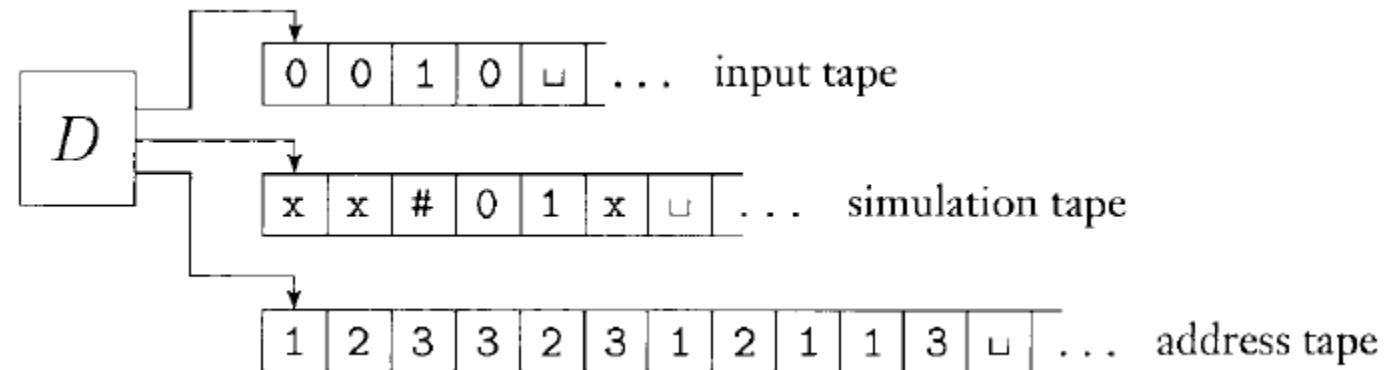
# Turing Maschine



„Nondeterministic Turing Machine“

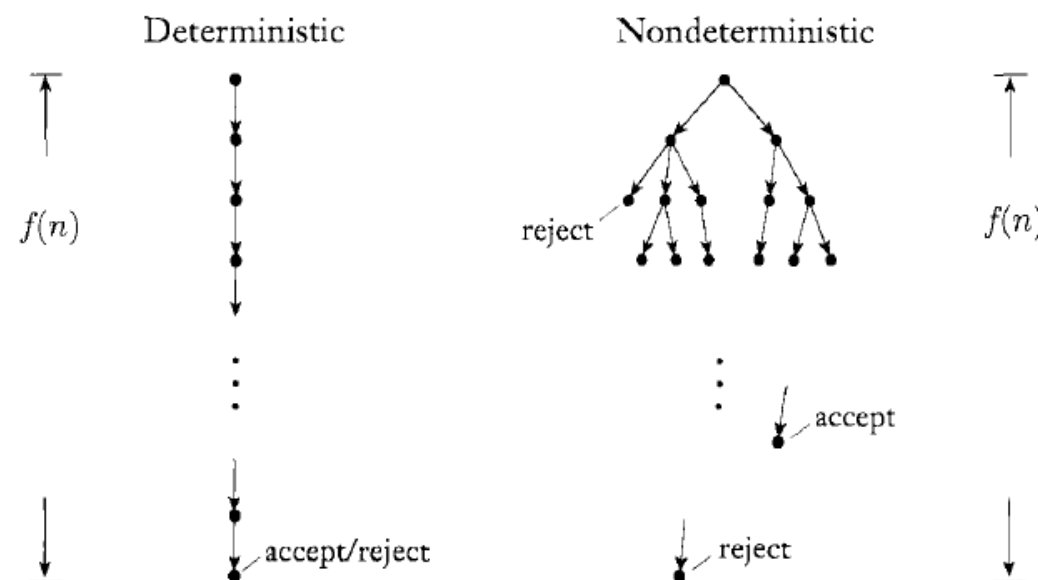
$$\delta: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

Every nondeterministic Turing machine has an equivalent deterministic Turing machine.



# Turing Maschine

Let  $N$  be a **nondeterministic Turing machine** that is a decider. The *running time* of  $N$  is the function  $f: \mathcal{N} \rightarrow \mathcal{N}$ , where  $f(n)$  is the **maximum number of steps** that  $N$  uses on any branch of its computation on any input of length  $n$ , as shown in the following figure.



# Turing Maschine



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

Let  $t(n)$  be a function, where  $t(n) \geq n$ . Then every  $t(n)$  time nondeterministic single-tape Turing machine has an equivalent  $2^{O(t(n))}$  time deterministic single-tape Turing machine.

# Entscheidbarkeit



Call a language **Turing-recognizable** if some Turing machine recognizes it.

„rekursiv aufzählbar“ oder auch „semi-entscheidbar“

Call a language **Turing-decidable** or simply *decidable* if some Turing machine decides it.

„rekursiv“ oder „entscheidbar“

# Entscheidbarkeit



A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

“On input  $w$ :

1. Run both  $M_1$  and  $M_2$  on input  $w$  in parallel.
2. If  $M_1$  accepts, *accept*; if  $M_2$  accepts, *reject*.”

# Das „Halte-Problem“



Wir betrachten als Beispiel folgende Sprache:

$$A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$$

...diese ist zumindest semi-entscheidbar:

“On input  $\langle M, w \rangle$ , where  $M$  is a TM and  $w$  is a string:

1. Simulate  $M$  on input  $w$ .
2. If  $M$  ever enters its accept state, *accept*; if  $M$  ever enters its reject state, *reject*.”

# Das „Halte-Problem“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## **THEOREM**

$A_{TM}$  is undecidable.

# Das „Halte-Problem“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$$

## **THEOREM**

$HALT_{TM}$  is undecidable.



Eingabe

	$i_1$	$i_2$	$i_3$	$\dots$	$d$
$T_1$	1	0	1		1
$T_2$	1	0	1	$\dots$	0
$T_3$	0	0	<u>1</u>		1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
D	<u>0</u>	<u>1</u>	<u>0</u>	$\dots$	?

Turingmaschine

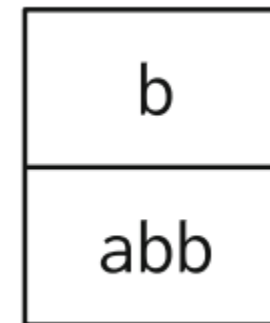
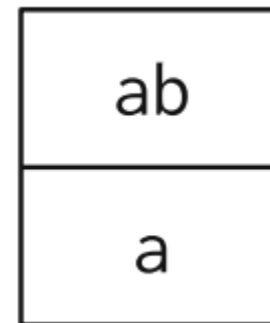
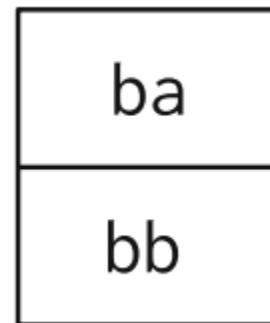
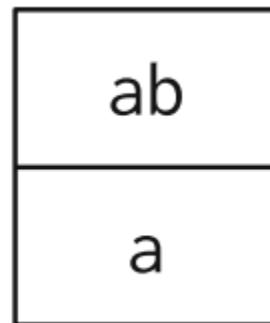
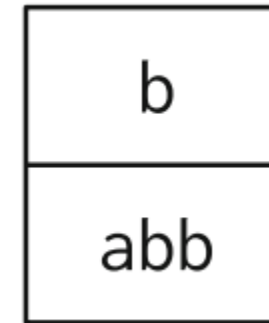
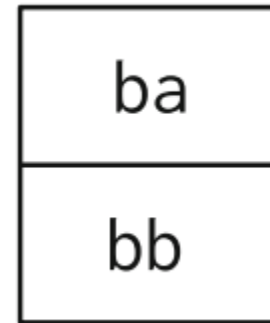
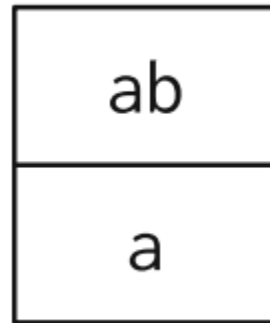
Halteproblem

Widerspruch

# weitere unentscheidbare Probleme



Frage: Kann  
ich **Kopien**  
dieser  
Dominos so  
aneinander  
legen, dass  
oben und  
unten das  
**gleiche Wort**  
steht?



# weitere unentscheidbare Probleme



## Postsche Korrespondenzproblem (PCP)

**gegeben:** Eine endliche Folge von Wortpaaren  $(x_1, y_1), \dots, (x_k, y_k)$ , wobei  $x_i, y_i \in \{0, 1\}^+$

**gefragt:** Gibt es eine endliche Folge von Indizes  $i_1, \dots, i_n \in \{1, \dots, k\}$ , mit  $x_{i_1} \dots x_{i_n} = y_{i_1} \dots y_{i_n}$

(ist zumindest semi-entscheidbar...)



Emil Leon Post, 1897-1957

# weitere unentscheidbare Probleme



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



*Hilbert*

Eine *D i o p h a n t i s c h e* Gleichung mit irgend welchen Unbekannten und mit ganzen rationalen Zahlencoefficienten sei vorgelegt: *man soll ein Verfahren angeben, nach welchem sich mittelst einer endlichen Anzahl von Operationen entscheiden. läßt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.*

# weitere unentscheidbare Probleme



## **Erfüllbarkeitsproblem:**

Gegeben: prädikatenlogischer Ausdruck  $A$  über einer Signatur  $\mathcal{S}$

Frage: Ist  $A$  erfüllbar ?

## **Gültigkeitsproblem:**

Gegeben: prädikatenlogischer Ausdruck  $A$  über einer Signatur  $\mathcal{S}$

Frage: Ist  $A$  allgemeingültig ?

## **Unerfüllbarkeitsproblem:**

Gegeben: prädikatenlogischer Ausdruck  $A$  über einer Signatur  $\mathcal{S}$

Frage: Ist  $A$  unerfüllbar ?

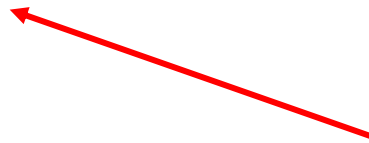
diese zwei sind  
immerhin  
semi-  
entscheidbar...

# weitere unentscheidbare Probleme



If  $M$  is a Turing machine, then we say that the *length* of the description  $\langle M \rangle$  of  $M$  is the number of symbols in the string describing  $M$ . Say that  $M$  is *minimal* if there is no Turing machine equivalent to  $M$  that has a shorter description. Let

$$MIN_{TM} = \{\langle M \rangle \mid M \text{ is a minimal TM}\}.$$



ist noch nicht einmal semi-entscheidbar...

# weitere unentscheidbare Probleme...



$$E_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

$$EQ_{\text{TM}} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

$$REGULAR_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$$

# Turing Maschine



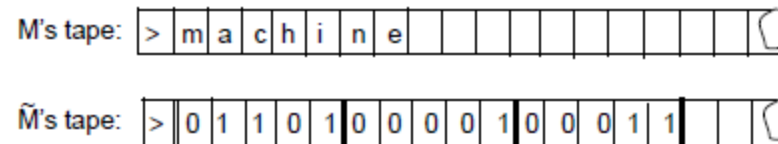
Es gibt auch eine  
alternative Definition bei  
der lediglich  $t(n) \geq n$   
gefordert wird...

A function  $t: \mathcal{N} \rightarrow \mathcal{N}$ , where  $t(n)$  is at least  $O(n \log n)$ , is called **time constructible** if the function that maps the string  $1^n$  to the binary representation of  $t(n)$  is computable in time  $O(t(n))$ .

# Turing Maschine



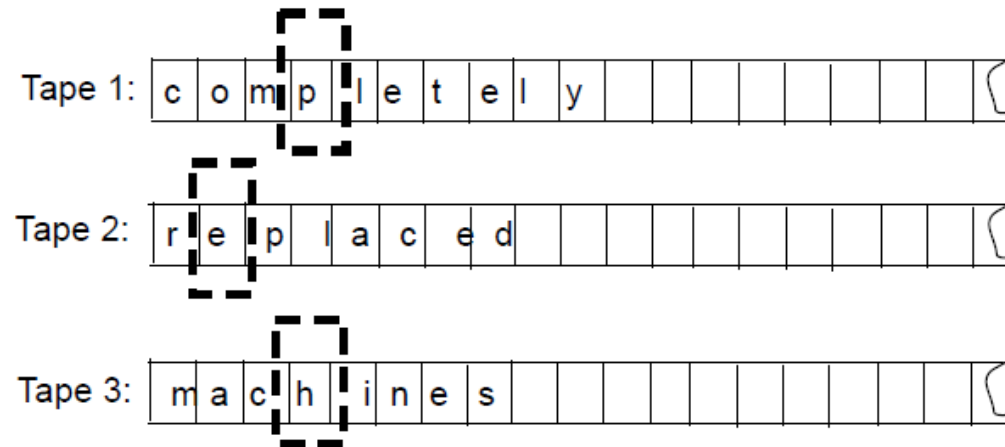
For every  $f : \{0,1\}^* \rightarrow \{0,1\}$  and time-constructible  $T : \mathbb{N} \rightarrow \mathbb{N}$ , if  $f$  is computable in time  $T(n)$  by a TM  $M$  using alphabet  $\Gamma$  then it is computable in time  $4 \log |\Gamma| T(n)$  by a TM  $\tilde{M}$  using the alphabet  $\{0,1,\square,\triangleright\}$ .



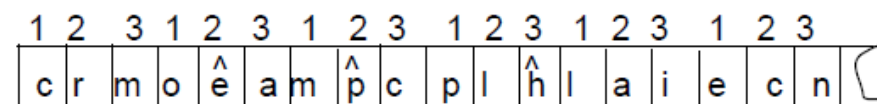
# Turing Maschine



For every  $f : \{0,1\}^* \rightarrow \{0,1\}$ , time-constructible  $T : \mathbb{N} \rightarrow \mathbb{N}$ , if  $f$  is computable in time  $T(n)$  by a TM  $M$  using  $k$  tapes (plus additional input and output tapes) then it is computable in time  $5kT(n)^2$  by a TM  $\tilde{M}$  using only a single work tape (plus additional input and output tapes).



Encoding this in one tape of  $\tilde{M}$ :

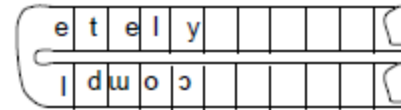
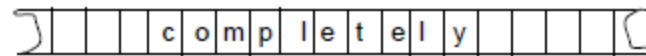


# Turing Maschine



Define a **bidirectional TM** to be a TM whose tapes are infinite in both directions. For every  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  and time constructible  $T : \mathbb{N} \rightarrow \mathbb{N}$ , if  $f$  is computable in time  $T(n)$  by a bidirectional TM  $M$  then it is computable in time  $4T(n)$  by a standard (unidirectional) TM  $\tilde{M}$ .

$M$ 's tape is infinite in both directions:



$\tilde{M}$  uses a larger alphabet to represent it on a standard tape:



# TIME



Let  $t: \mathcal{N} \longrightarrow \mathcal{R}^+$  be a function. Define the *time complexity class*, **TIME**( $t(n)$ ), to be the collection of all languages that are decidable by an  $O(t(n))$  time Turing machine.

# Die Klasse P



**P** is the class of languages that are **decidable in polynomial time** on a **deterministic** single-tape Turing machine. In other words,

$$P = \bigcup_k \text{TIME}(n^k).$$

# Die Klasse P



Sind die folgenden Mengen in P ?

$$\mathbb{N} = \{1, 2, 3, \dots\}$$

$$2\mathbb{N} = \{2, 4, 6, \dots\}$$

$$\mathbb{P} = \{n \in \mathbb{N} \mid n \text{ ist Primzahl}\}$$

$$\{p + q \mid p, q \in \mathbb{P}\}$$

$$\{p \cdot q \mid p, q \in \mathbb{P}\}$$

# Die Klasse NP



A *verifier* for a language  $A$  is an algorithm  $V$ , where

$$A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}.$$

We measure the time of a verifier only in terms of the length of  $w$ , so a *polynomial time verifier* runs in polynomial time in the length of  $w$ . A language  $A$  is *polynomially verifiable* if it has a polynomial time verifier.

# Die Klasse NP



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

**NP** is the class of languages that have polynomial time verifiers.

# Die Klasse NP



## THEOREM

A language is in NP iff it is decided by some **nondeterministic polynomial time** Turing machine.

# NTIME



$\text{NTIME}(t(n)) = \{L \mid L \text{ is a language decided by a } O(t(n)) \text{ time}$   
 $\text{nondeterministic Turing machine}\}.$

# Die Klasse NP



$$P = \bigcup_k \text{TIME}(n^k).$$

$$NP = \bigcup_k \text{NTIME}(n^k).$$

P = the class of languages for which membership can be *decided* quickly.

NP = the class of languages for which membership can be *verified* quickly.

# Die Klasse NP



Sind die folgenden Mengen in NP ?

$$\{p + q \mid p, q \in \mathbb{P}\}$$

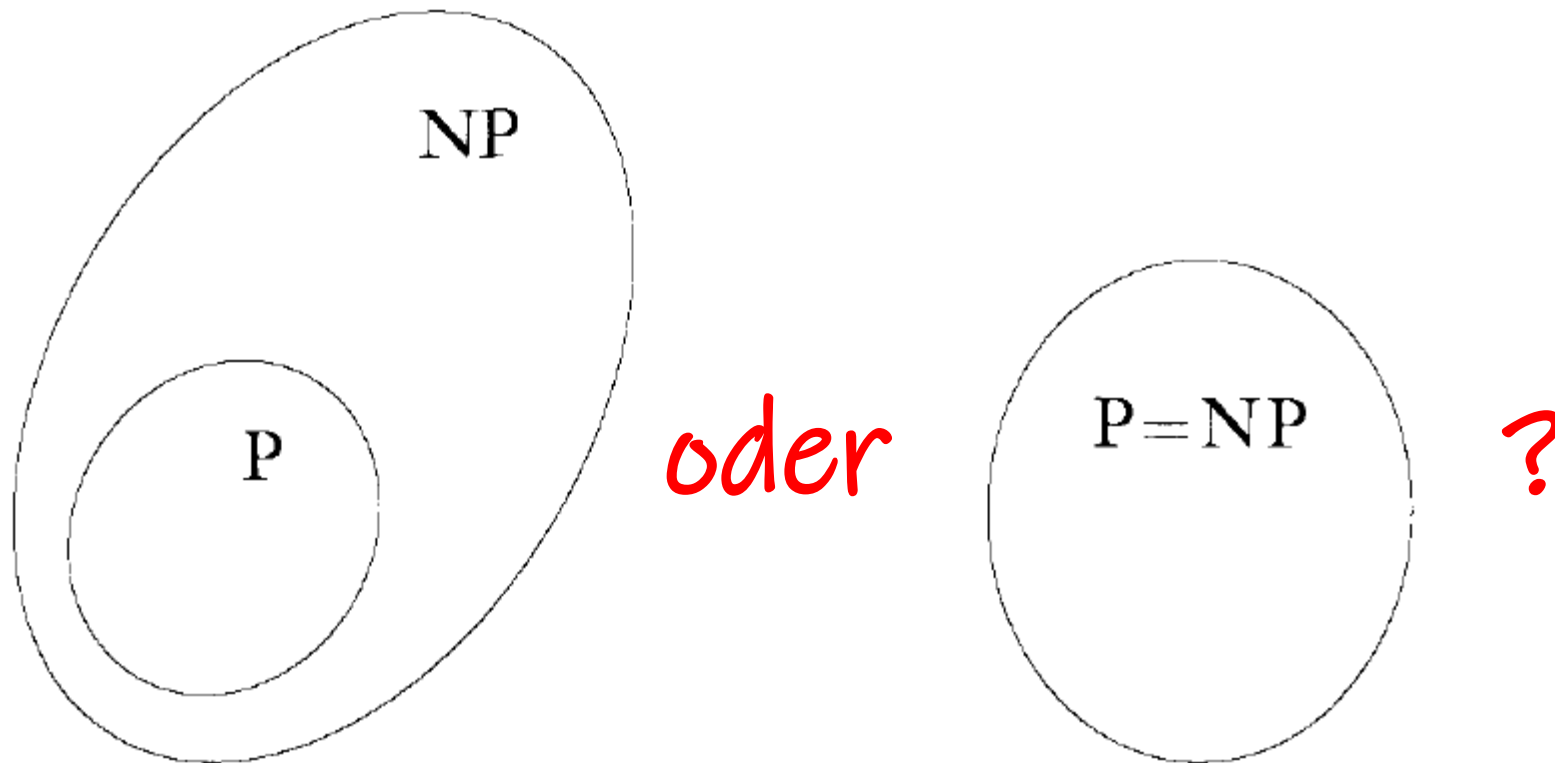
$$\{p \cdot q \mid p, q \in \mathbb{P}\}$$

$$\{n \in \mathbb{N} \mid n \neq p \cdot q, p, q \in \mathbb{P}\}$$

$$\text{SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable Boolean function}\}$$

$$\text{UNSAT} = \{\langle \phi \rangle : \phi \text{ is an unsatisfiable Boolean function}\}$$

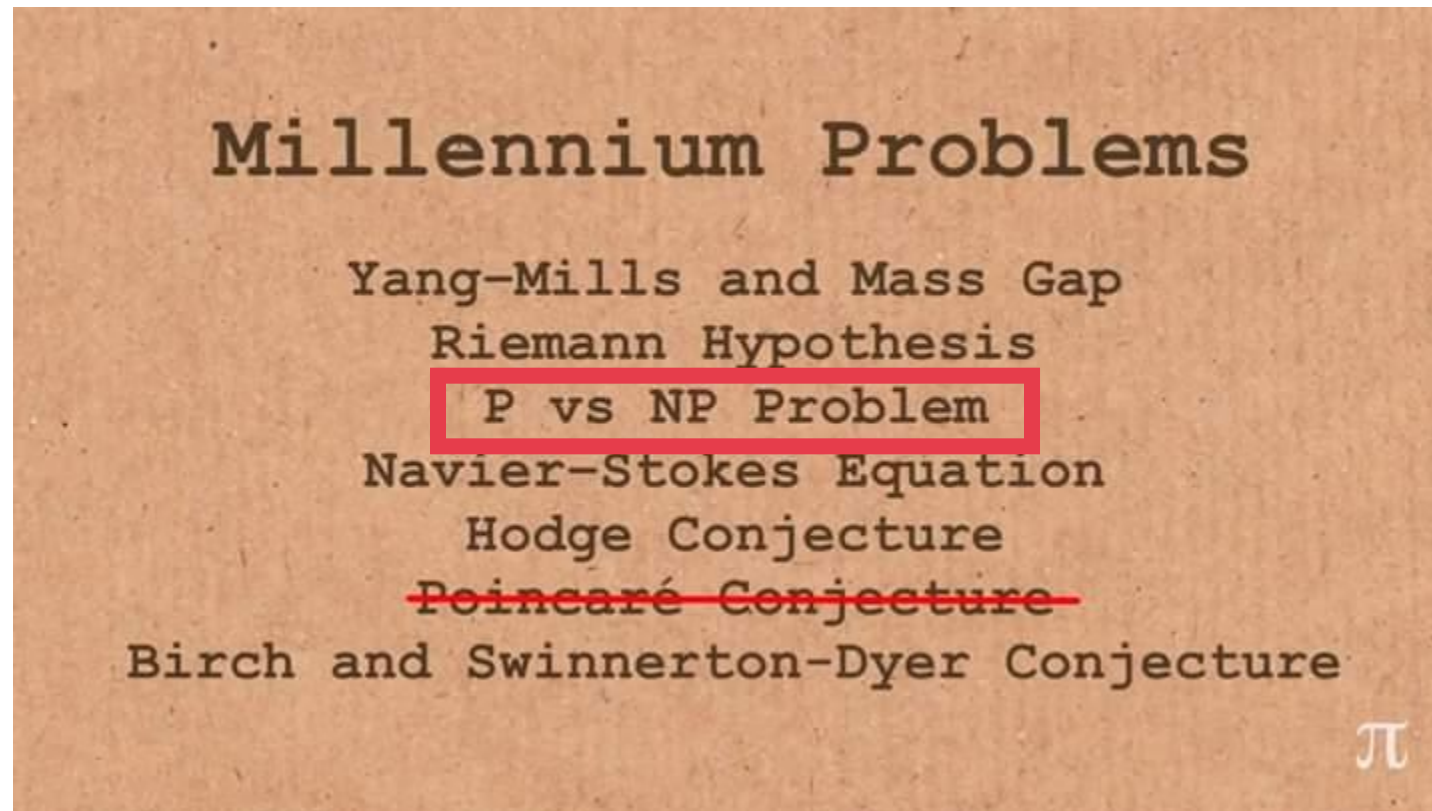
# P vs NP



# The Million Dollar Problem(s)



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# NP-Vollständigkeit



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

$$SAT = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula} \}$$

## THEOREM

**Cook-Levin theorem**  $SAT \in P$  iff  $P = NP$ .

# Reduktion



A function  $f: \Sigma^* \longrightarrow \Sigma^*$  is a **polynomial time computable function** if some polynomial time Turing machine  $M$  exists that halts with just  $f(w)$  on its tape, when started on any input  $w$ .

Language  $A$  is **polynomial time mapping reducible**, or simply **polynomial time reducible**, to language  $B$ , written  $A \leq_P B$ , if a polynomial time computable function  $f: \Sigma^* \longrightarrow \Sigma^*$  exists, where for every  $w$ ,

$$w \in A \iff f(w) \in B.$$

The function  $f$  is called the **polynomial time reduction** of  $A$  to  $B$ .

# NP-Vollständigkeit

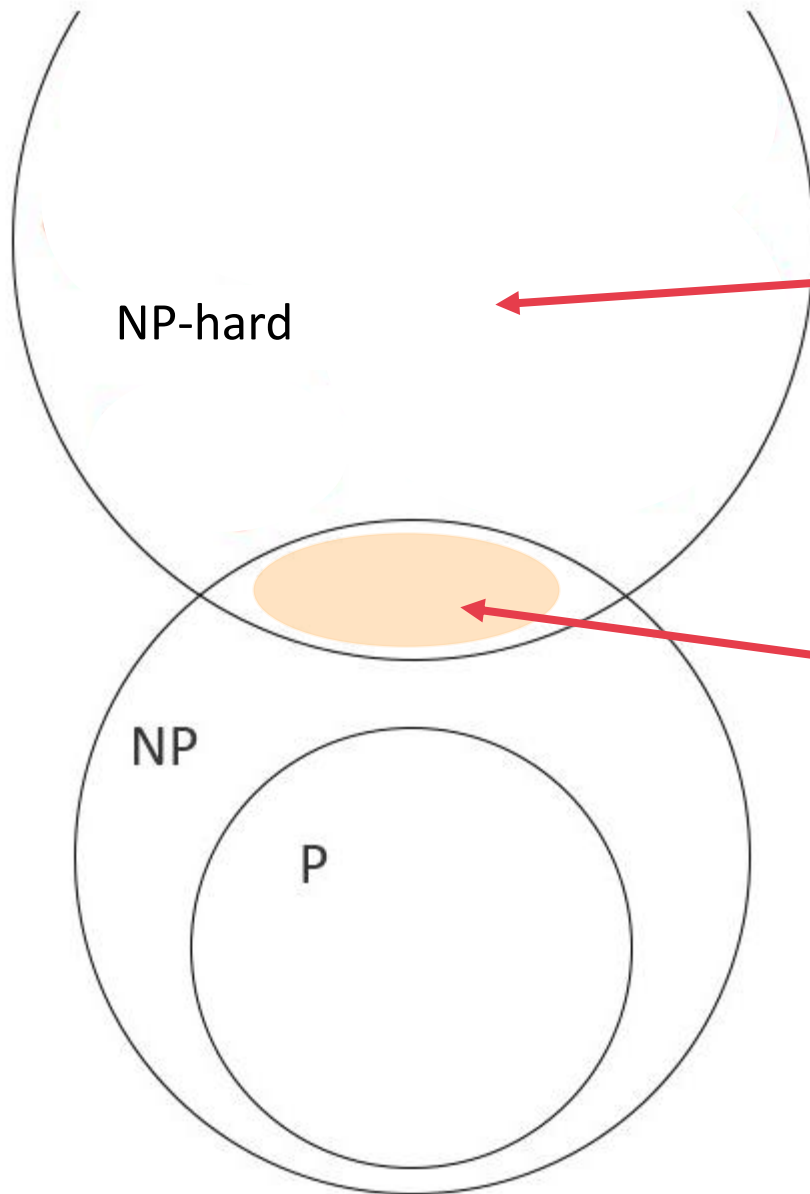


A language  $B$  is **NP-complete** if it satisfies two conditions:

1.  $B$  is in NP, and
2. every  $A$  in NP is polynomial time reducible to  $B$ .

Falls  $B$  (nur) die zweite Bedingung erfüllt, so heißt  $B$   
„NP-hard“





Wenn ich ein Problem aus  
dieser Menge lösen kann,  
dann kann ich auch **jedes  
andere Problem** aus NP lösen

Probleme in dieser  
Schnittmenge  
heißen **NP-  
vollständig**

# Cook-Levin Theorem



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

## **THEOREM**

*SAT* is NP-complete.

# Cook-Levin Theorem



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

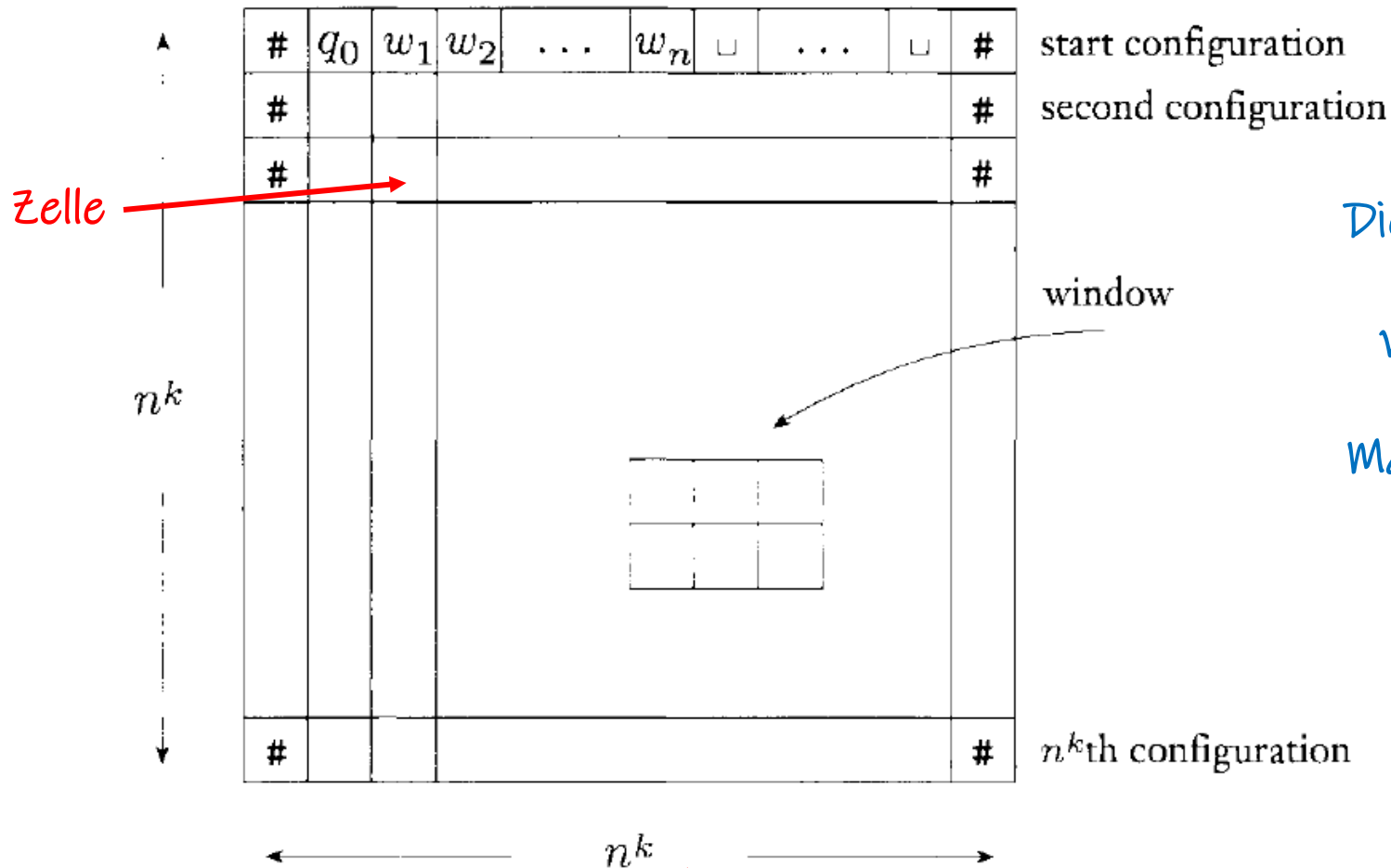
- 1) SAT ist in NP
- 2) Zeige für eine Sprache  $A$  in NP, dass sie polynomial reduzierbar auf SAT ist.

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Die Idee ist hier für jedes Wort  $w$  eine aussagenlogische Formel zu finden, welche **polynomiale Größe** hat und die Berechnung der (nichtdet.-) Turing-Maschine simuliert, d.h. die genau dann erfüllbar ist, falls die Maschine  $w$  akzeptiert.

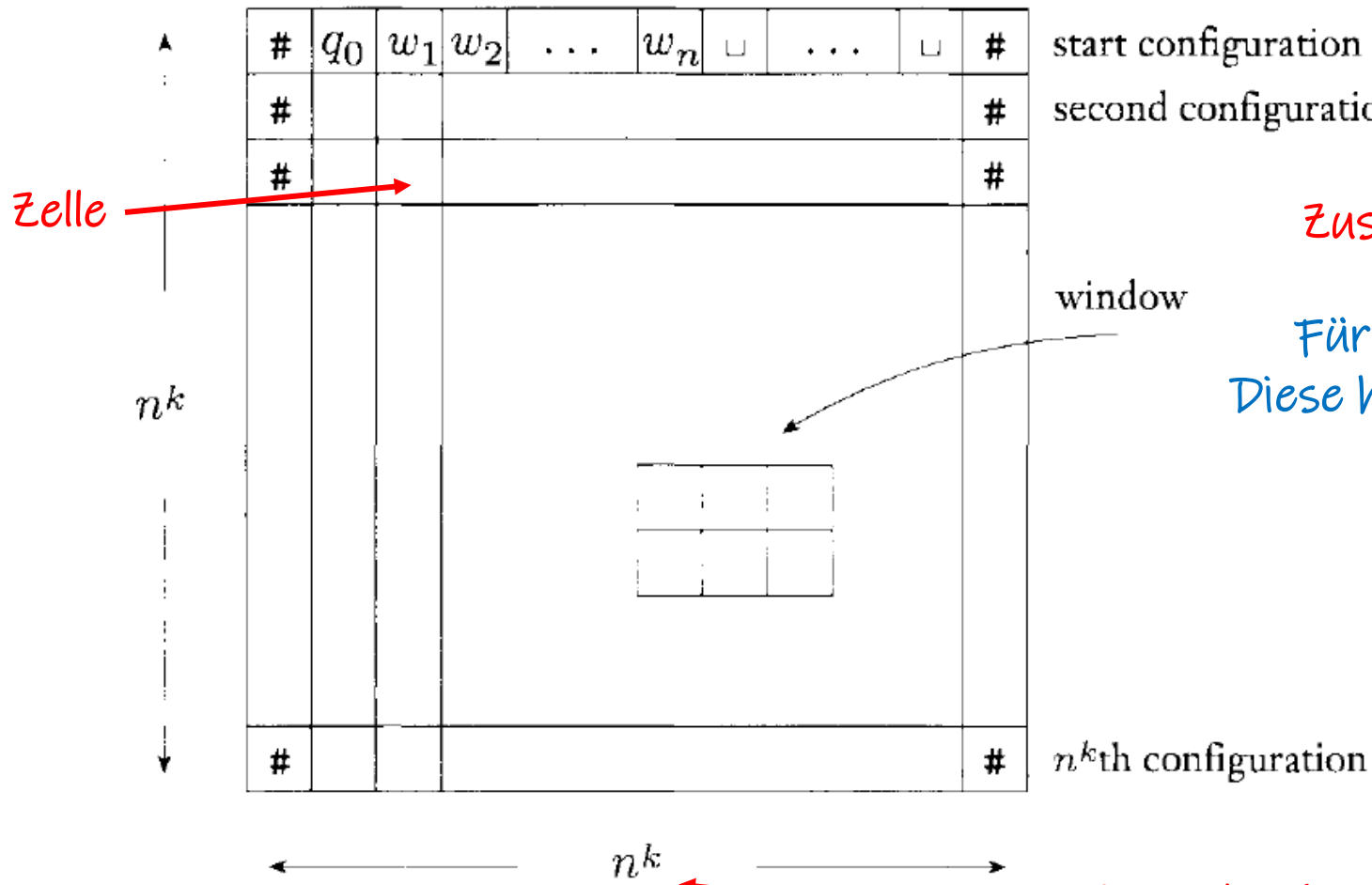
Laufzeit der (nichtdet.-) Turing-Maschine, welche  $A$  akzeptiert

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



$$C = Q \cup \Gamma \cup \{\#\}$$

Zustandsmenge

Bandalphabet

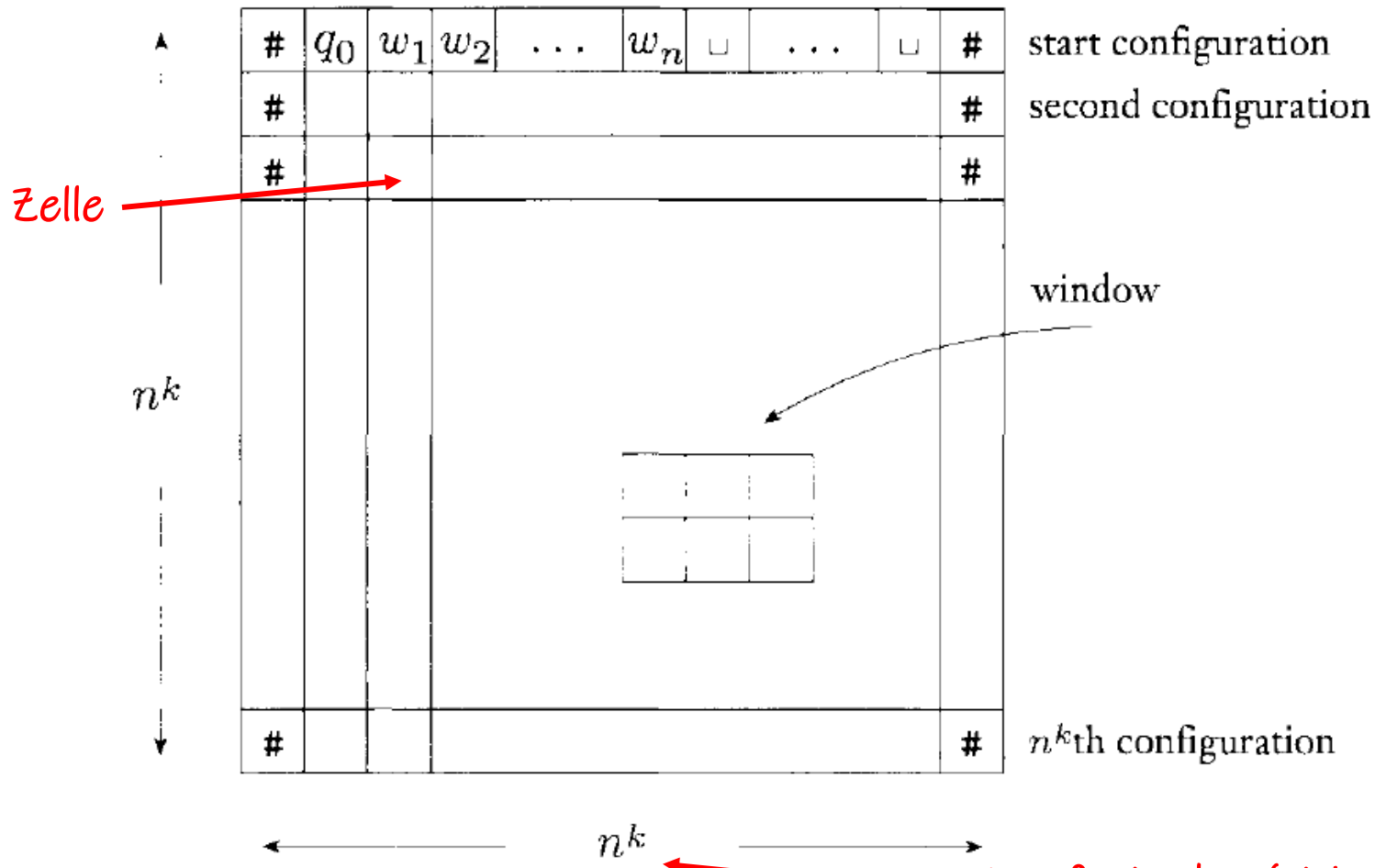
Für  $s$  aus  $C$  definieren wir Variablen  $x_{i,j,s}$ .  
Diese haben genau dann den Wert „1“, falls  $s$  in  
Zelle  $(i,j)$  vorkommt.

Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert

# Cook-Levin Theorem



akzeptierendes  
„Tableaux“



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

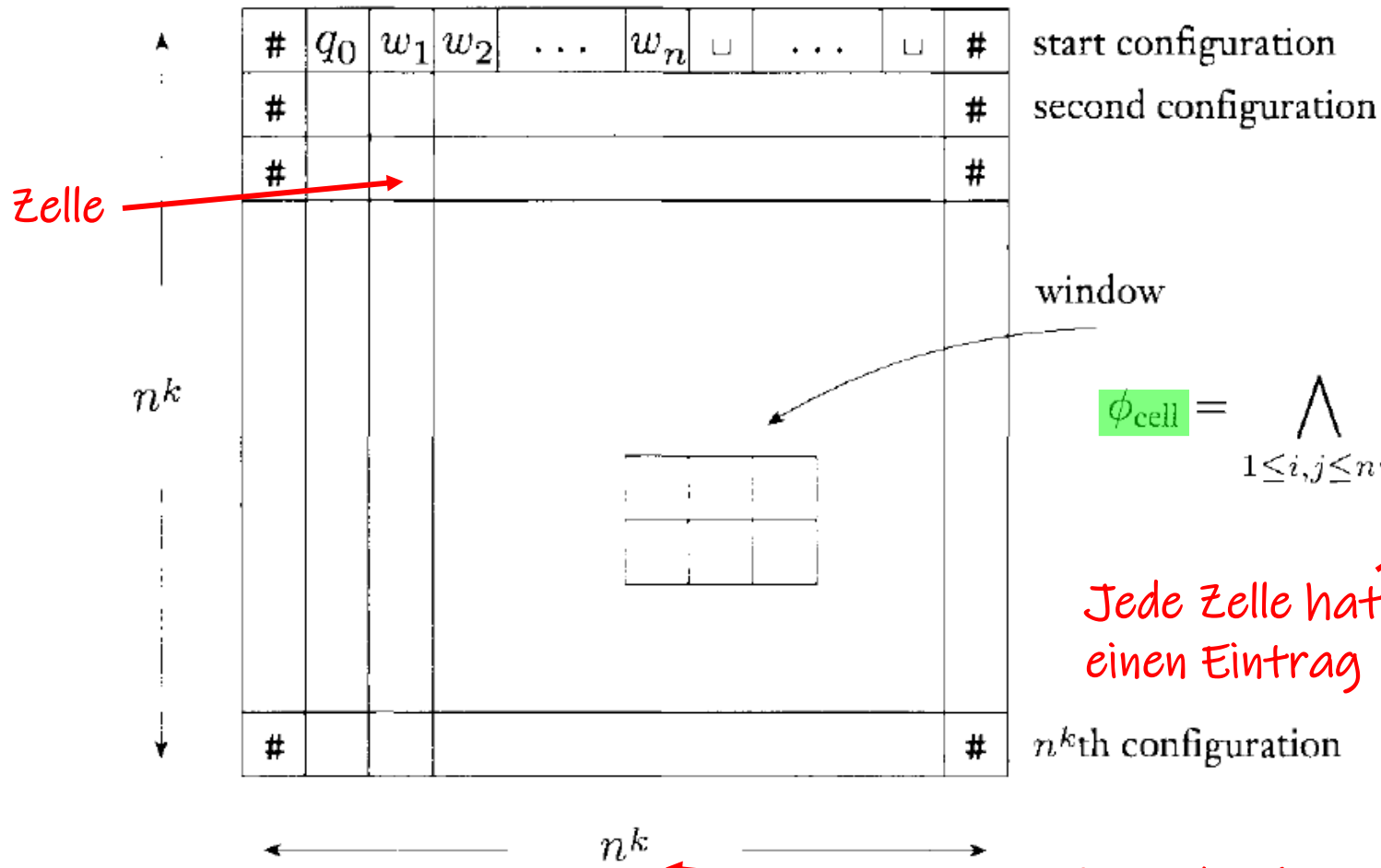
Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}},$$

wobei

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{s, t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

Jede Zelle hat mind.  
einen Eintrag

Jede Zelle hat  
höchstens einen Eintrag

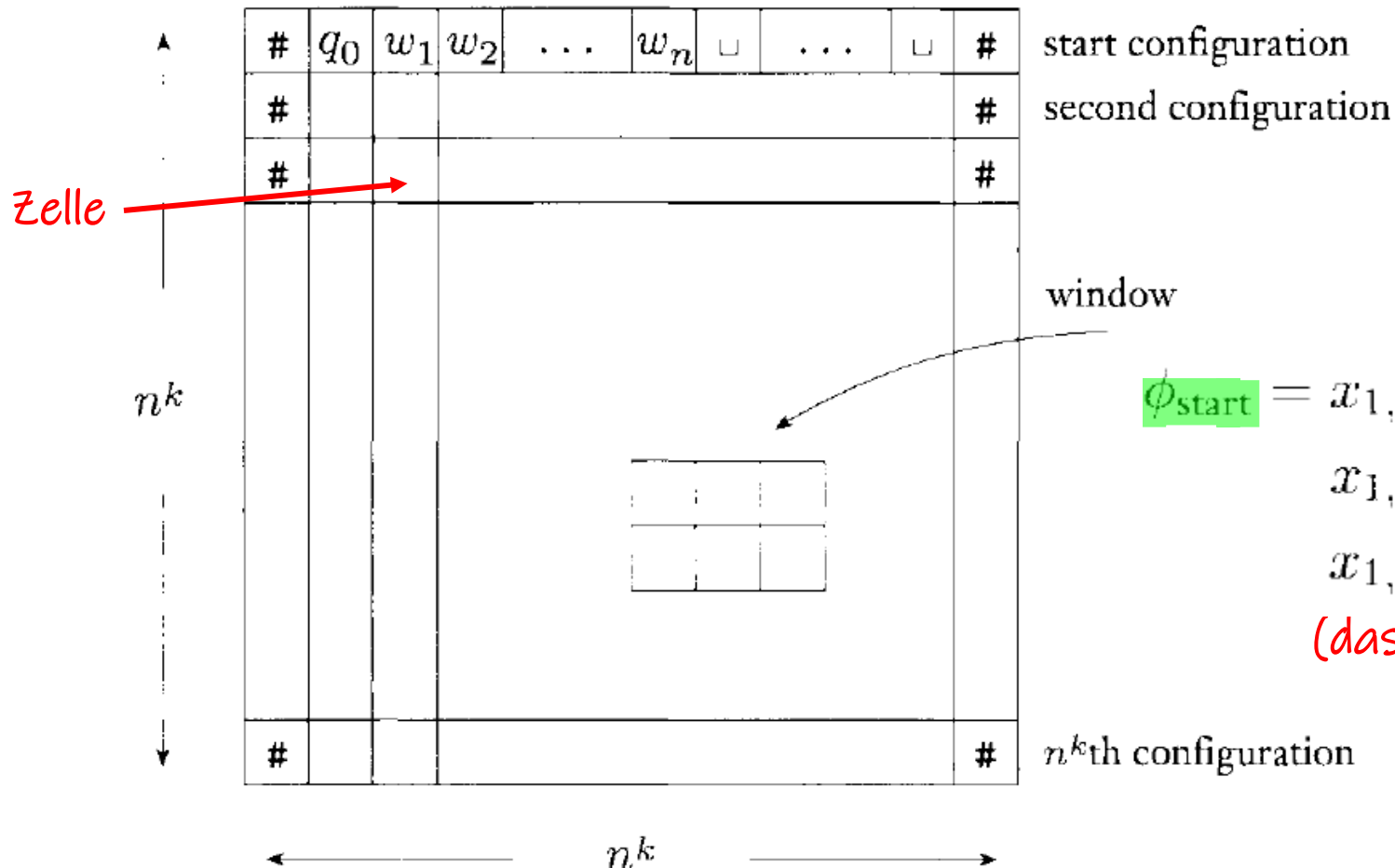
Laufzeit der (nichtdet.-) Turing-Maschine,  
welche A akzeptiert

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}},$$

wobei

$$\begin{aligned} \phi_{\text{start}} = & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \dots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \end{aligned}$$

(das ist gerade die Anfangskonfiguration)

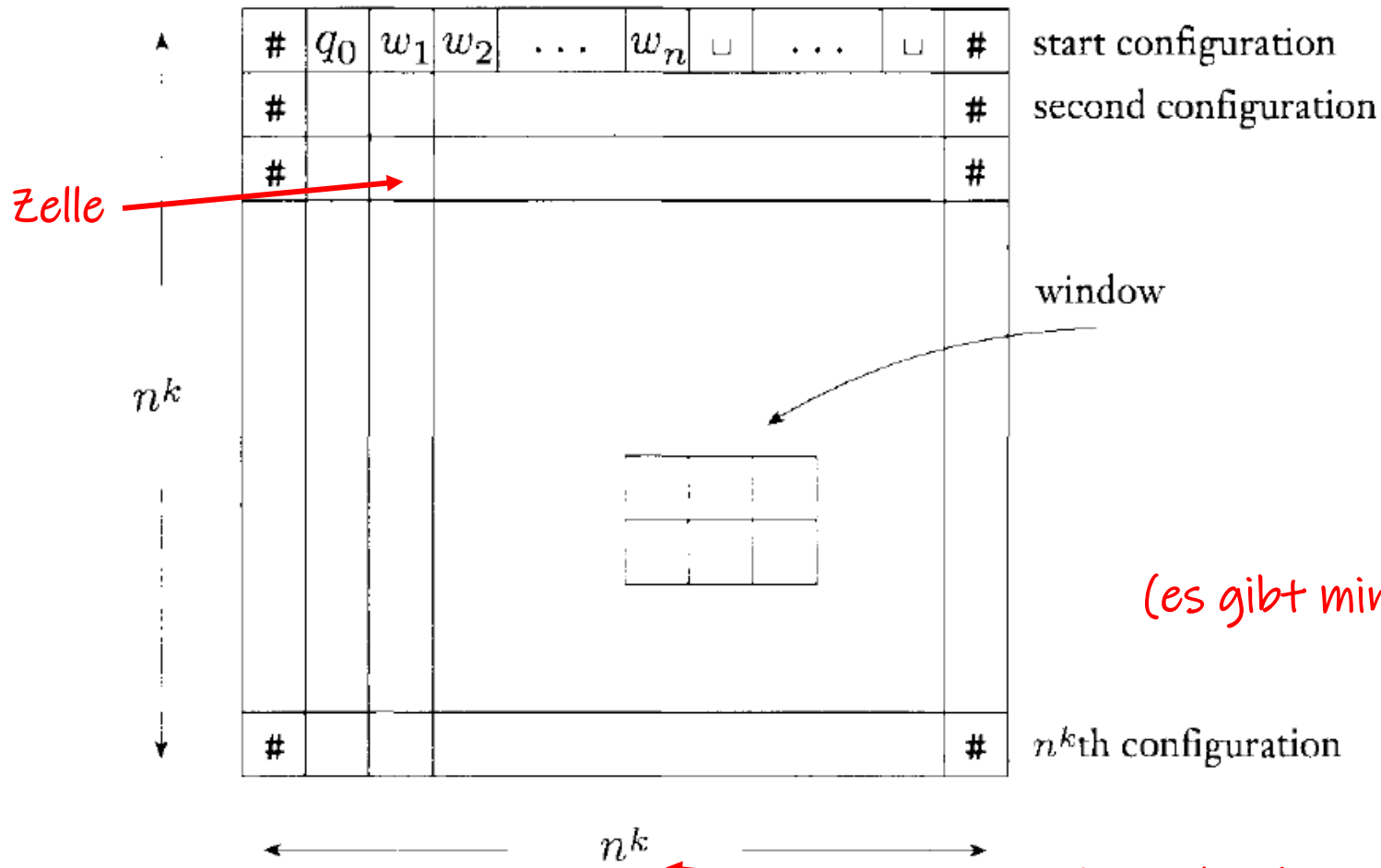
Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}},$$

wobei

$$\phi_{\text{accept}} = \bigvee_{1 \leq i, j \leq n^k} x_{i, j, q_{\text{accept}}}$$

(es gibt mind. eine akzeptierende Konfiguration)

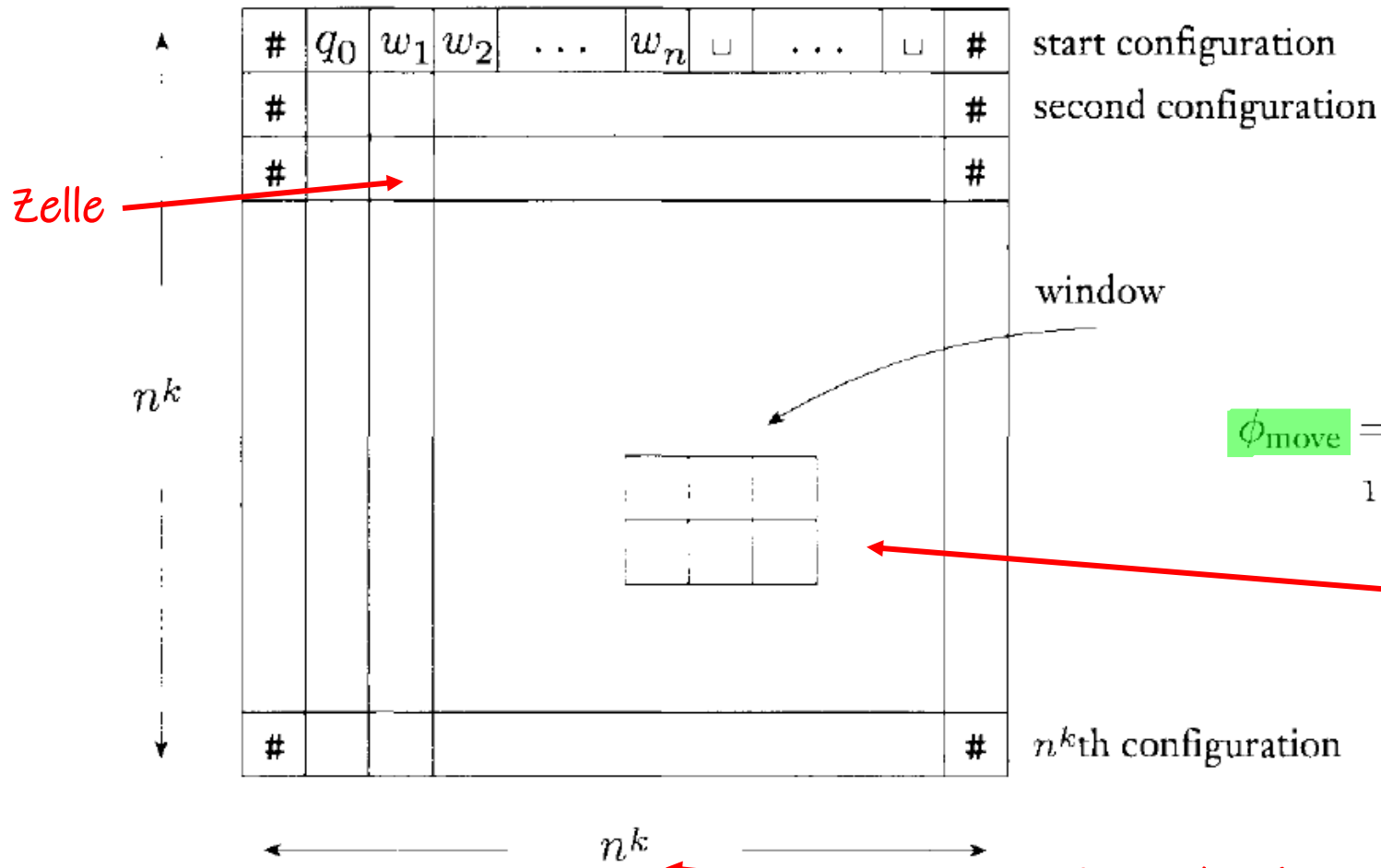
Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}},$$

wobei

$$\phi_{\text{move}} = \bigwedge_{1 \leq i \leq n^k, 1 \leq j < n^k} (\text{the } (i, j) \text{ window is legal})$$

Ein Fenster hat hier die  
Größe 2x3

Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert

# Cook-Levin Theorem



Ein Fenster ist **legal**, falls es nicht der Spezifikation der Übergangsfunktion der entsprechenden Maschine widerspricht:

Beispiel:  $\delta(q_1, a) = \{(q_1, b, R)\}$   
 $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$

Diese Fenster sind alle  
„legal“

a	b	a
a	a	a

a	$q_1$	b
$q_2$	a	c

a	$q_1$	b
a	a	$q_2$

a	a	$q_1$
a	a	b

Diese nicht:

a	$q_1$	b
$q_1$	a	a

#	b	a
#	b	a

a	b	a
a	b	$q_2$

b	b	b
c	b	b

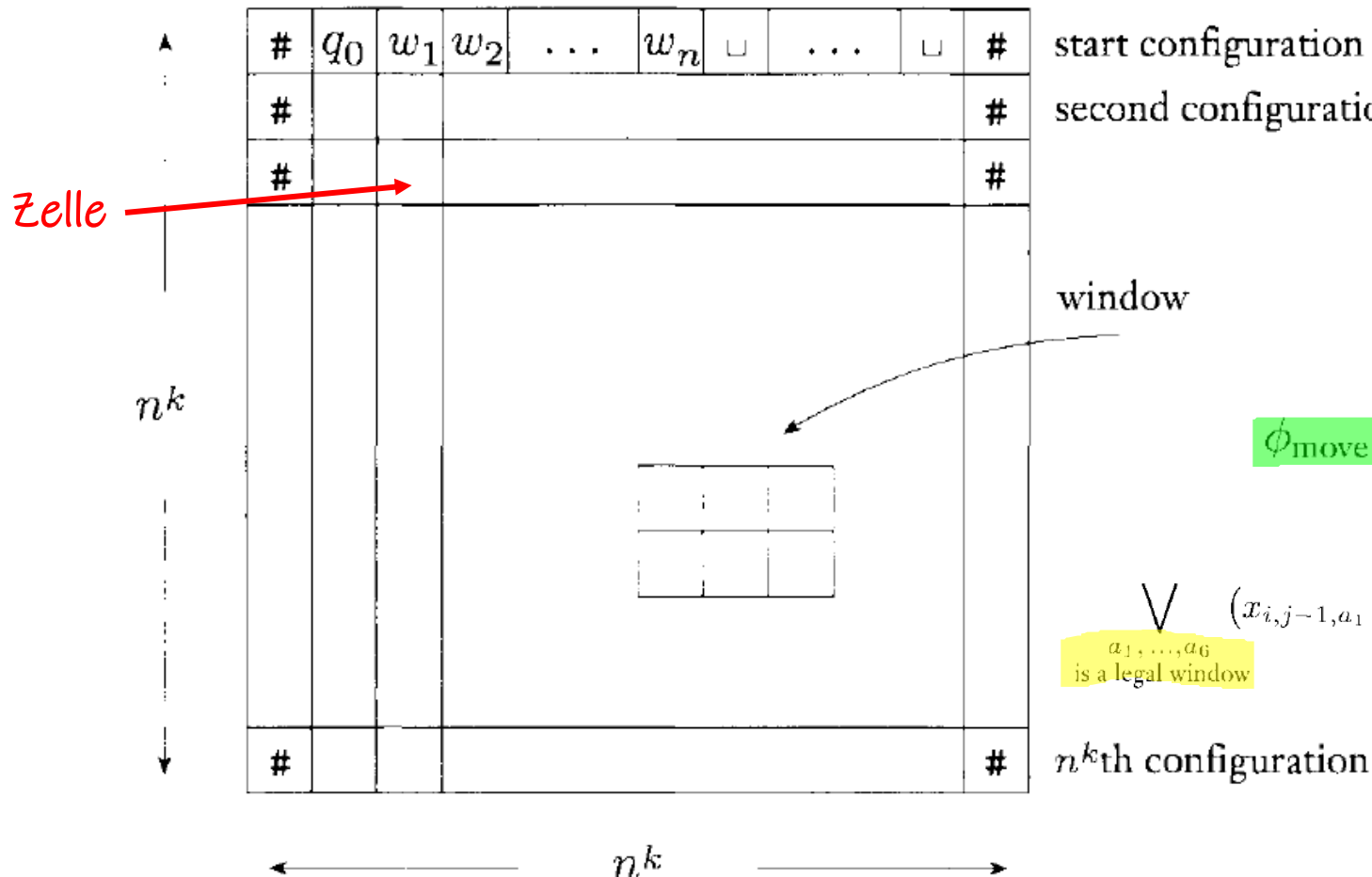
b	$q_1$	b
$q_2$	b	$q_2$

# Cook-Levin Theorem

akzeptierendes  
„Tableaux“



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



Unsere Formel sieht dann so aus:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}},$$

wobei

$$\phi_{\text{move}} = \bigwedge_{1 \leq i \leq n^k, 1 \leq j < n^k} (\text{the } (i, j) \text{ window is legal})$$

$$\bigvee_{\substack{a_1, \dots, a_6 \\ \text{is a legal window}}} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6})$$

Laufzeit der (nichtdet.-) Turing-Maschine,  
welche  $A$  akzeptiert



# Cook-Levin Theorem

## CLAIM

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Bleibt noch die Größe der Formel:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

$$O(n^{2k})$$

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i, j \leq n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{s, t \in C \\ s \neq t}} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right]$$

# Cook-Levin Theorem



## CLAIM

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Bleibt noch die Größe der Formel:

$$\begin{array}{l} O(n^{2k}) \xrightarrow{\quad} \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}} \\ O(n^k) \xrightarrow{\quad} \phi_{\text{start}} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ \quad x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \dots \wedge x_{1,n+2,w_n} \wedge \\ \quad x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#} \end{array}$$

# Cook-Levin Theorem



## CLAIM

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Bleibt noch die Größe der Formel:

$$\begin{array}{ccccc} & & \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}} & & \\ & \nearrow & & \nwarrow & \\ O(n^{2k}) & & & & O(n^{2k}) \\ & \nearrow & & \nwarrow & \\ & O(n^k) & \phi_{\text{move}} = \bigwedge_{1 \leq i \leq n^k, 1 \leq j \leq n^k} (\text{the } (i, j) \text{ window is legal}) & & \end{array}$$

# Cook-Levin Theorem



## CLAIM

If the top row of the table is the start configuration and every window in the table is legal, each row of the table is a configuration that legally follows the preceding one.

Bleibt noch die Größe der Formel:

$$\phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

$O(n^{2k})$   $O(n^k)$   $O(n^{2k})$   $O(n^{2k})$

$$\phi_{\text{accept}} = \bigvee_{1 \leq i, j \leq n^k} x_{i, j, q_{\text{accept}}}$$

# 3SAT



## COROLLARY

*3SAT* is NP-complete.

(es reicht hier einzusehen, dass zu jeder Formel  $F$  eine Formel  $F'$  in 3KNF existiert, welche **erfüllbarkeitsäquivalent** zu  $F$  ist: schreibe  $F$  in KNF und ersetze jede Klausel mit  $l > 3$  Literalen,  $(a_1 \vee a_2 \vee \dots \vee a_l)$ , durch die  $l-2$  Klauseln  $(a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee z_2) \wedge (\overline{z_2} \vee a_4 \vee z_3) \wedge \dots \wedge (\overline{z_{l-3}} \vee a_{l-1} \vee a_l)$ )

# Hamiltonian Circuit Problem



Das Hamiltonian Circuit Problem ist NP-vollständig, und zwar schon falls der Graph planar und 3-regulär ist.

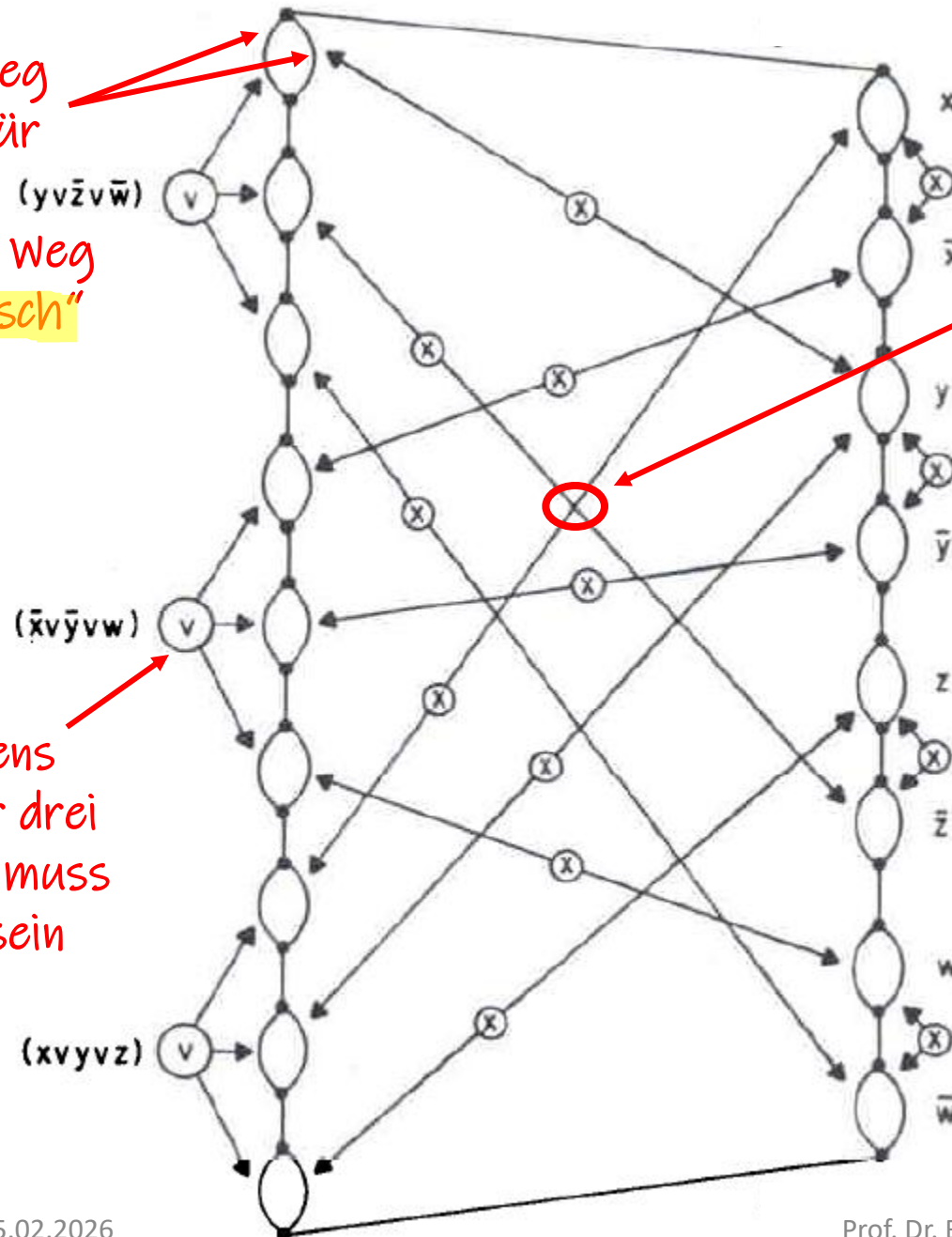
Graph, welcher in der Ebene dargestellt werden kann, so dass sich keine zwei Kanten schneiden

jeder Knoten besitzt genau 3 Nachbarn

für einen (ungerichteten) Graph soll entschieden werden, ob ein Rundweg existiert, welcher jeden Knoten genau einmal enthält



linker Weg  
steht für  
„wahr“,  $(yv\bar{z}v\bar{w})$   
rechter Weg  
für „falsch“



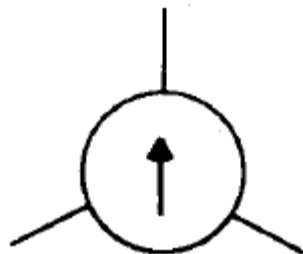
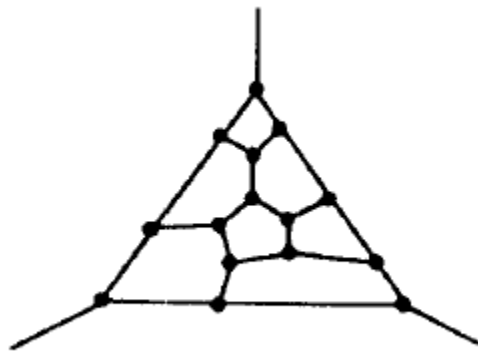
„Kreuzung“: Kanten des Graphen  
dürfen sich nicht kreuzen

Wir zeigen die Konstruktion an  
einem Beispiel: gesucht ist ein  
Graph (mit den genannten  
Eigenschaften) welcher genau dann  
einen (Hamilton-) Rundweg besitzt,  
falls die unten stehende Formel  
erfüllbar ist.

$$F = (x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee w) \wedge (y \vee \bar{z} \vee \bar{w})$$

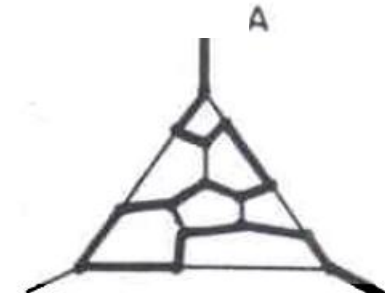
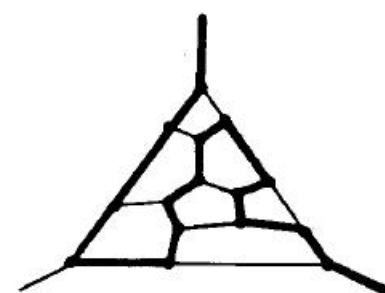
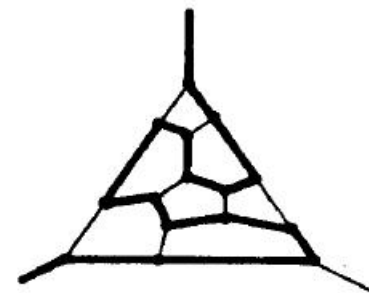
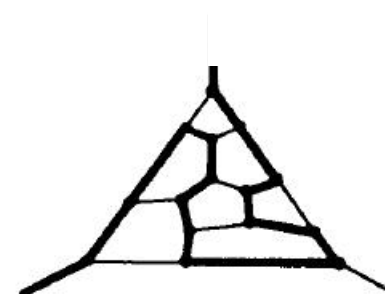
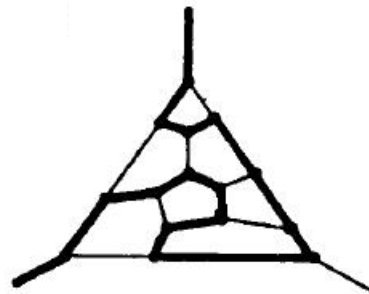
„xor“: genau einer der  
beiden Wege gehört  
zum Pfad

# Tutte Fragment

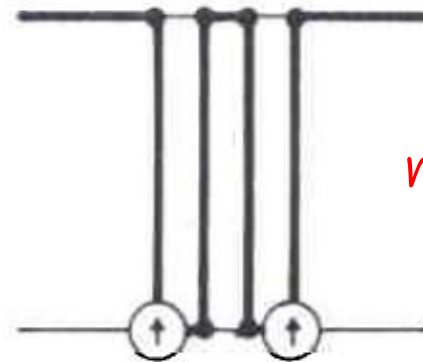
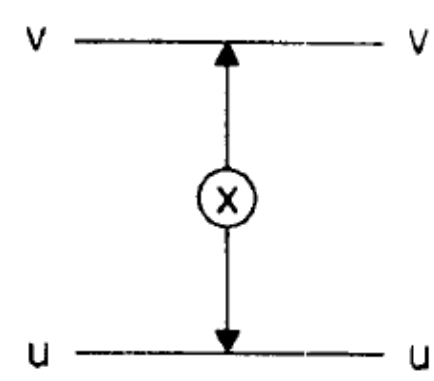
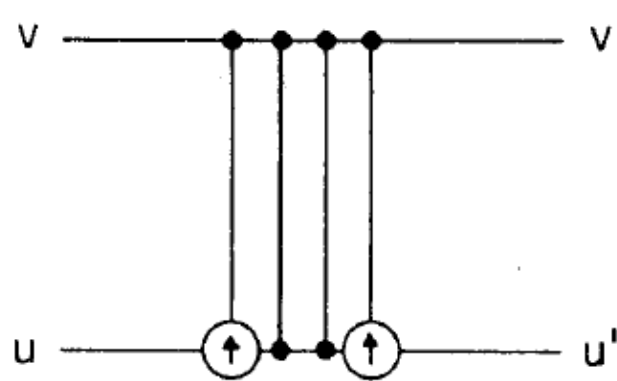


Symbol

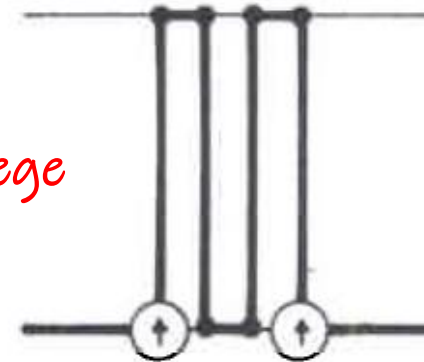
mögliche Wege



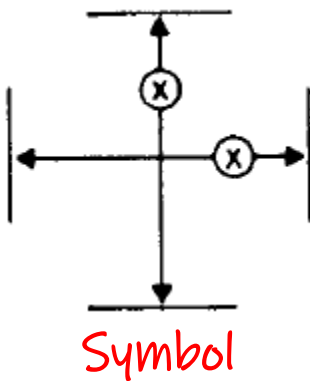
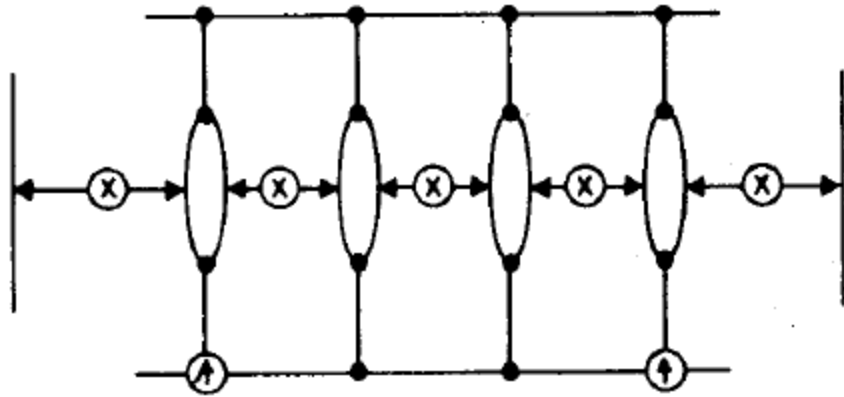
# „Exclusive-OR“



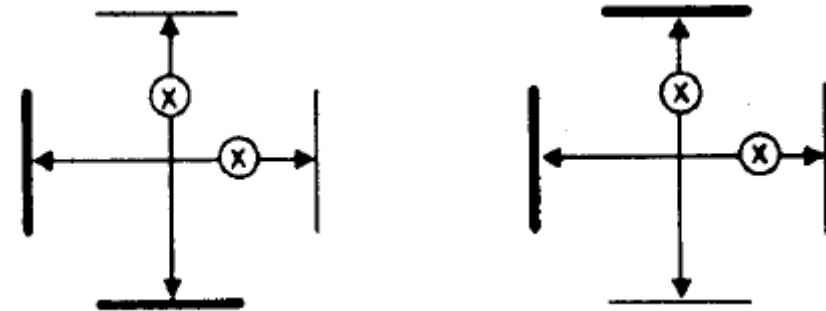
mögliche Wege



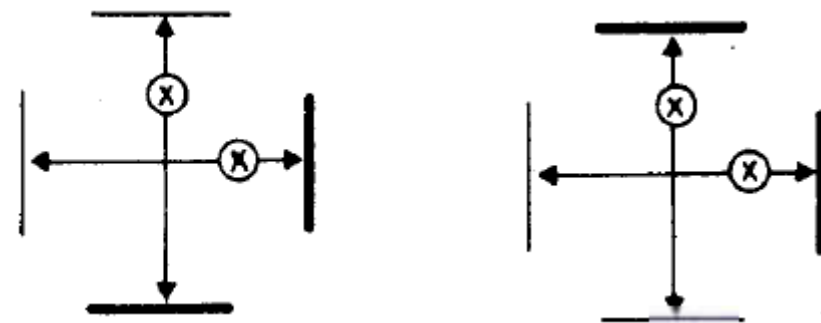
# „Kreuzung“



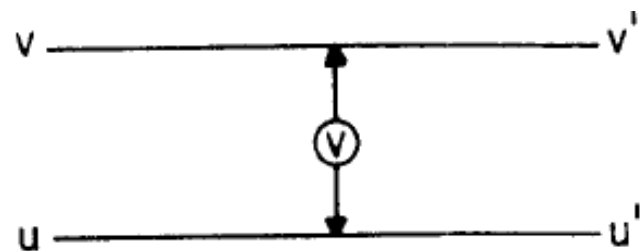
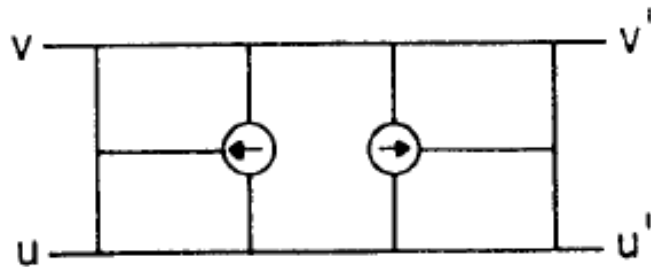
Symbol



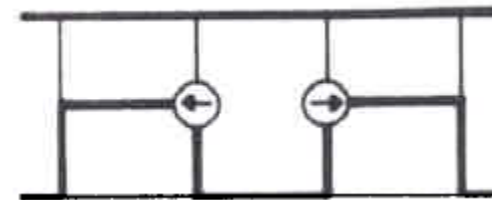
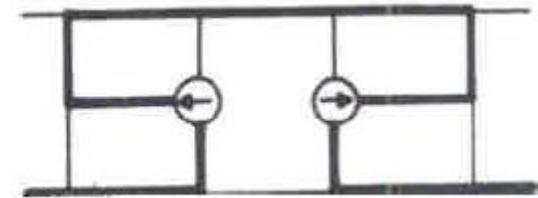
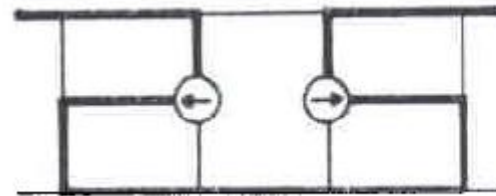
mögliche Wege



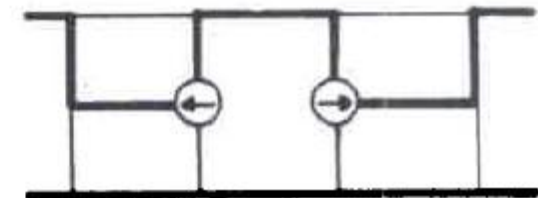
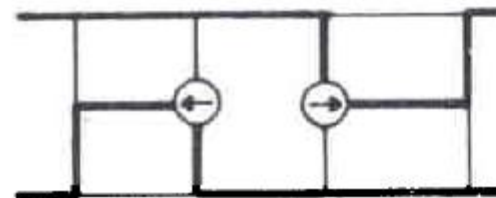
„OR“



Symbol

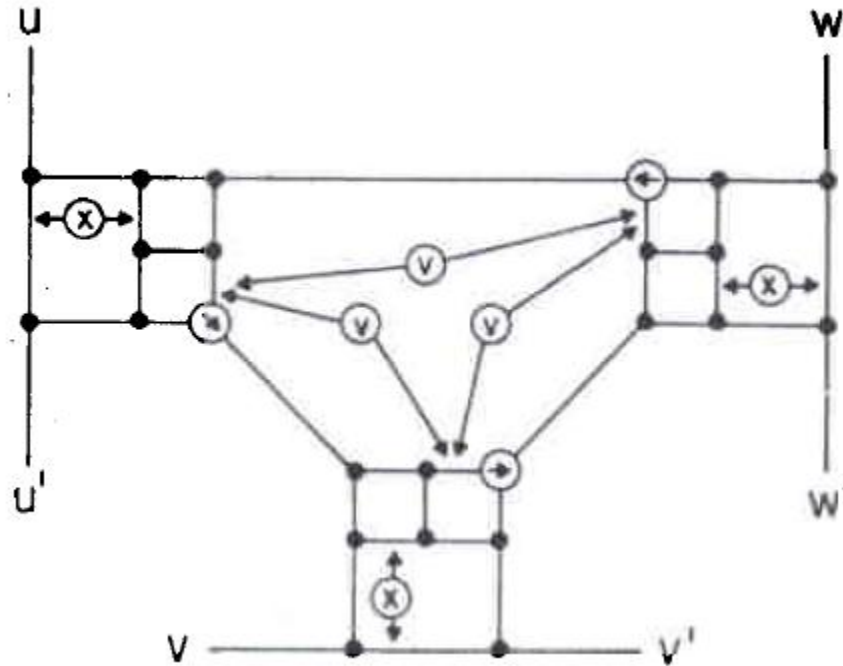


mögliche Wege

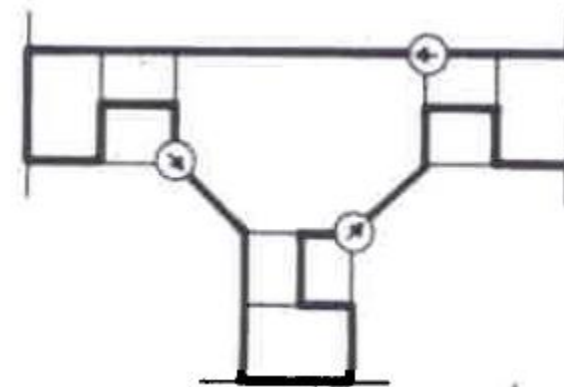
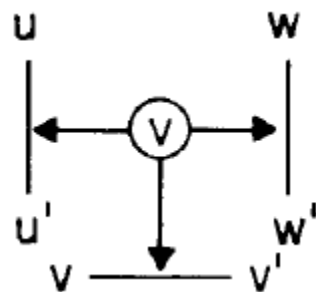




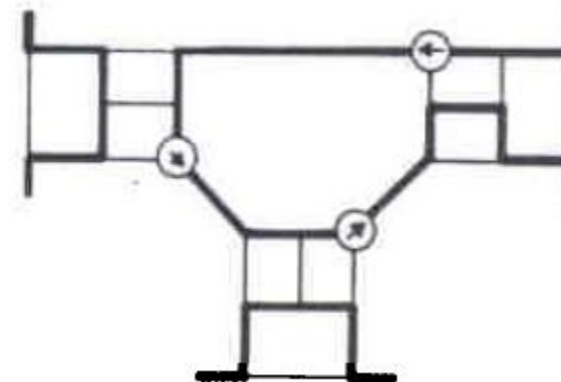
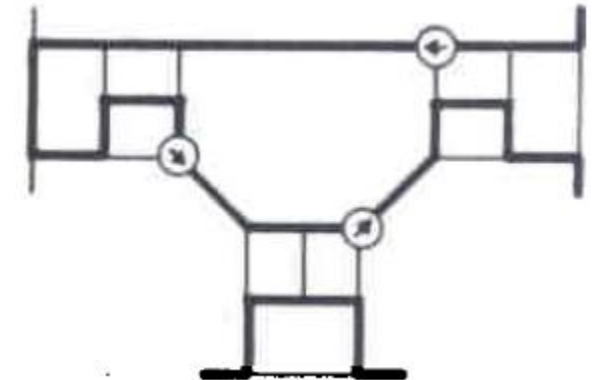
# „(tripple-)OR“



Symbol



mögliche Wege  
(ohne Symmetrie)



# MINESWEEPER



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# MINESWEEPER



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

	2	2	2	2	
	2	0	0	2	
	2	0	0	2	
	2	2	2	2	

# MINESWEEPER ist NP-vollständig

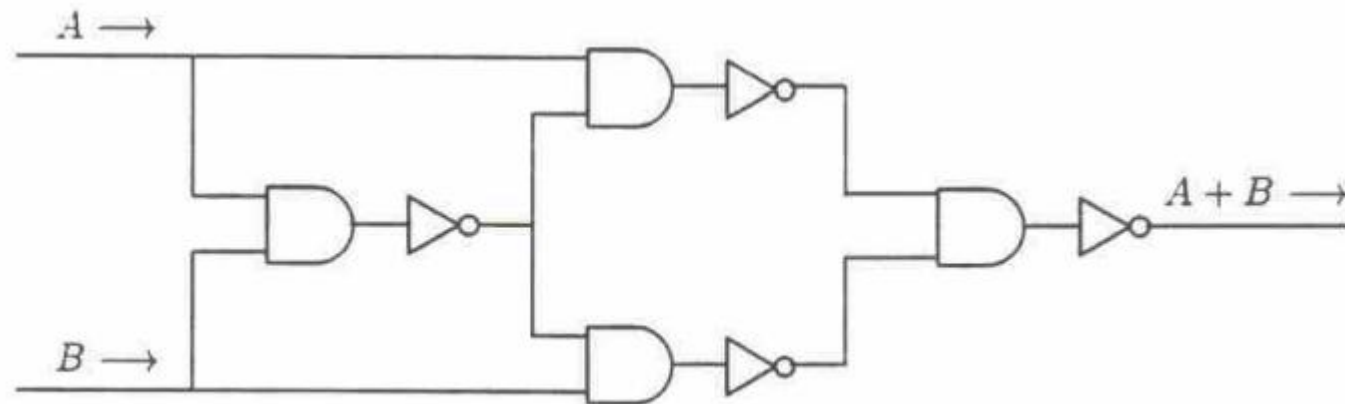


Figure 12. Making an XOR gate with AND and NOT gates.

# MINESWEEPER ist NP-vollständig



$X \rightarrow$

...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...
...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...
...	$x$	1	$x'$	$x$	1	$x'$	$x$	1	$x'$	$x$	1	$x'$	$x$	1	$x'$	$x$	...
...	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	...
...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...

Figure 6. A wire.

# MINESWEEPER ist NP-vollständig

$X \rightarrow$				1	2	2	1	
...	1	1	1	2	*	*	3	1
...	1	$x'$	$x$	1	$x'$	*	*	2
...	1	1	1	1	2	$x$	*	2
					1	1	2	1
					1	$x'$	1	
					1	$x$	1	$X$
					1	1	1	$\downarrow$
					$\vdots$	$\vdots$	$\vdots$	

1	1	1	X $\rightarrow$					
2	*	3	1	1	1	1	1	...
3	*	$x'$	$x$	1	$x'$	$x$	1	...
2	*	3	1	1	1	1	1	...
1	1	1						

				:	:	:				
				1	x	1				
				1	x'	1				
X →				1	1	1			X' →	
..	1	1	1	1	x	1	1	1	1	..
..	x'	x	1	x'	2	x'	1	x	x'	..
..	1	1	1	1	x	1	1	1	1	..
				1	1	1				
				1	x'	1				
				1	x	1				
				:	:	:				

# MINESWEEPER ist NP-vollständig

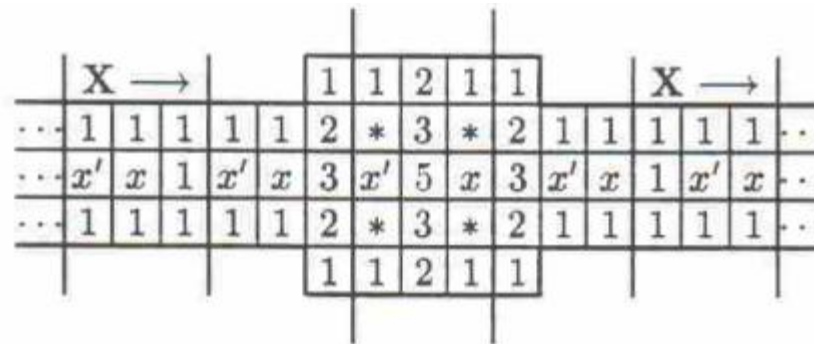


Figure 10. A phase-changer made from two NOT gates.

# MINESWEEPER ist NP-vollständig

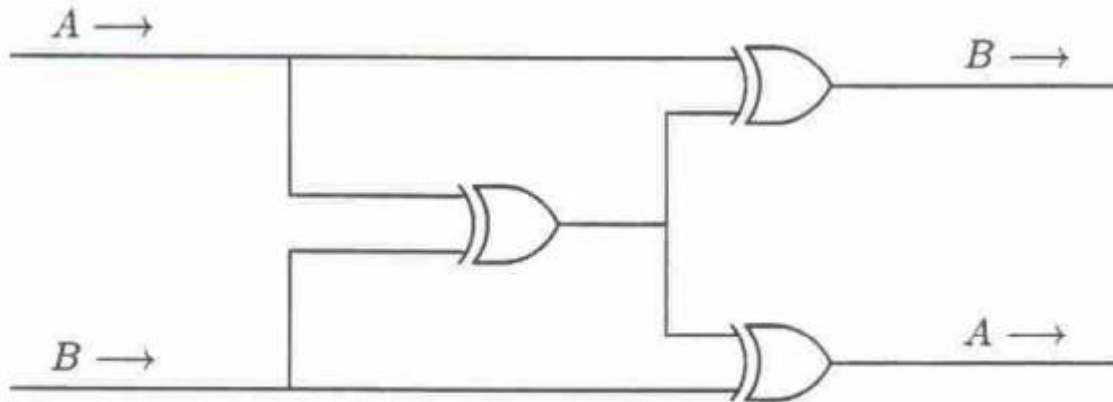


Figure 11. Crossing two wires with three XOR gates.

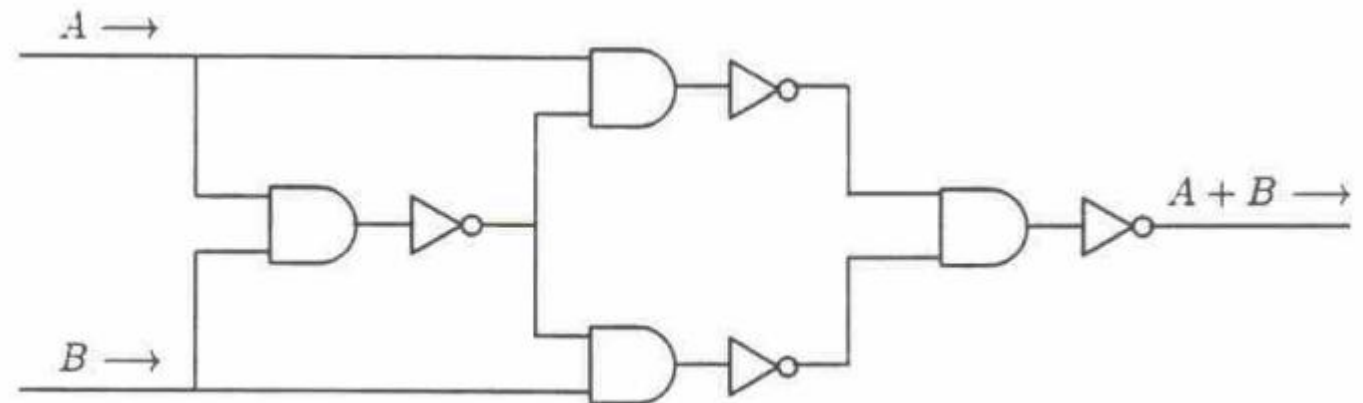


Figure 12. Making an XOR gate with AND and NOT gates.

# MINESWEEPER ist NP-vollständig



$X \rightarrow$					1	1	1	$X' \rightarrow$						
...	1	1	1	1	1	2	*	2	1	1	1	1	1	...
...	$x'$	$x$	1	$x'$	$x$	3	$x'$	3	$x$	$x'$	1	$x$	$x'$	...
...	1	1	1	1	1	2	*	2	1	1	1	1	1	...
					1	1	1							

Figure 9. A NOT gate.

# MINESWEEPER ist NP-vollständig

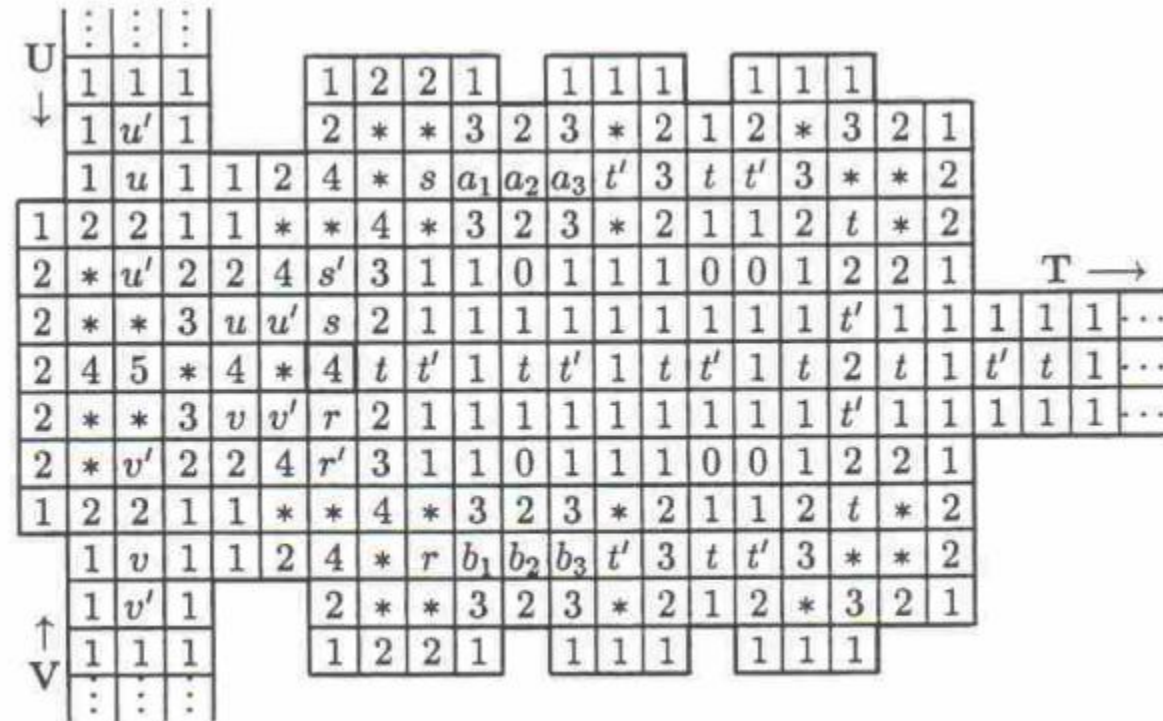


Figure 13. An AND gate.

# „Gaming is hard“, Part I



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

↙ d.h. bestimmte Punkte  
müssen erreicht werden



(a) Initial configuration.



(b) Configuration after a traversal from left to right.



(c) On the second traversal attempt, a boulder blocks the way.

**Metatheorem 1.** Any game exhibiting both *location traversal* (with or without a starting location or an exit location) and *single-use paths* is NP-hard.

Wir bauen einen Level, welcher einen *planaren, 3-regulären Graph* darstellt, wobei die Knoten die Punkte sind, welche besucht werden müssen und *jede Kante ein „single-use-path“* darstellt. Nach Wahl eines Startpunktes (Knoten) „S“ verbinden wir diesen mit einem weiteren (End-)Punkt, welcher nur eine Kante zum Startknoten hat und ansonsten keine weiteren Nachbarn besitzt. Der Level ist nun genau dann *lösbar*, falls ein *Hamilton-Kreis* von „S“ nach „S“ existiert.

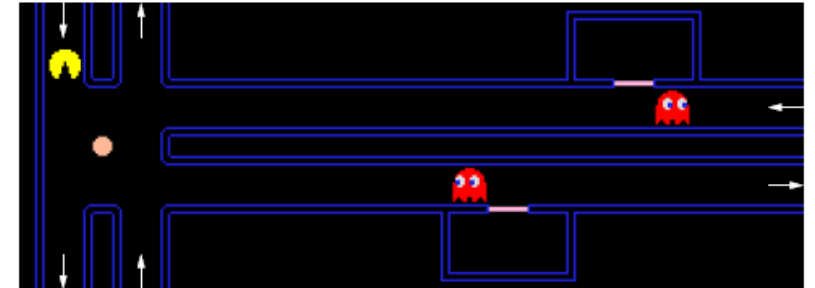
# „Gaming is hard“, Part I

**Metatheorem 2.** *A game is NP-hard if either of the following holds:*

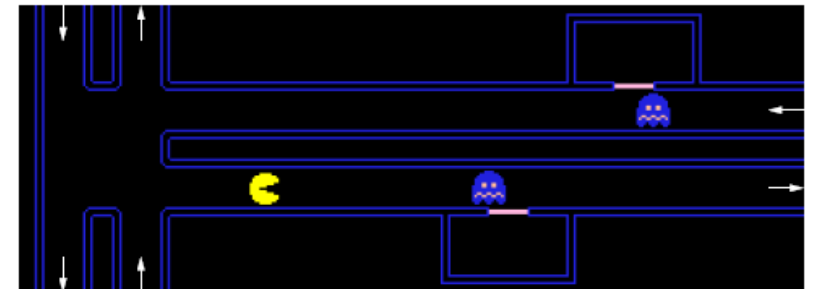
- (a) *The game features collectible tokens, toll roads, and location traversal.*
- (b) *The game features cumulative tokens, toll roads, and location traversal.*
- (c) *The game features collectible cumulative tokens, toll roads, and the avatar has to reach an exit location.*

um diesen Weg zu passieren,  
muss ein Token eingesammelt  
werden

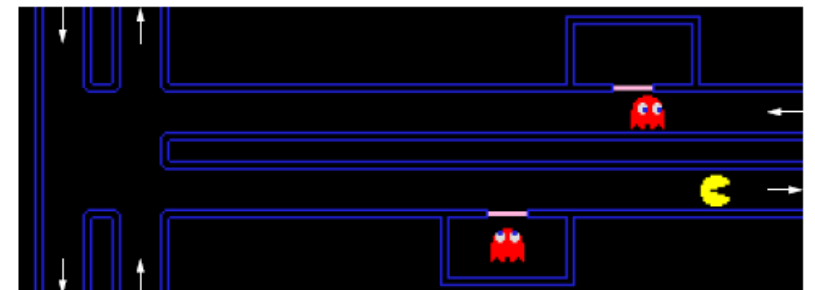
(a) und (b) wie bei MT 1: „tokens“ und „toll roads“ ergeben „single-use-paths“. Bei (b) hat der Spieler zu begin  $n+1$  „tokens“. Bei (c) plazieren wir 2 „tokens“ an jeden Punkt und jede Kante wird eine „toll road“, außer der Kante zwischen Start- und Endpunkt, welche eine Sequenz von  $n$  „toll roads“ ist.



(a) Initial configuration, approached from the top.



(b) Collecting the power pill to traverse the corridor.



(c) Exiting to the right while the ghosts recover.

# „Gaming is hard“, Part I



eine Tür wird von einem Schlüssel  
geöffnet, welcher danach als  
verbraucht gilt



**Metatheorem 3.** *A game is NP-hard if it contains **doors** and one-way paths, and either of the following holds:*

- (a) The game features **collectible keys** and location traversal.*
- (b) The game features **cumulative keys** and location traversal.*
- (c) The game features collectible cumulative keys and the avatar has to reach an exit location.*

(alles genau wie bei MT 2: „keys“ = „tokens“, „doors“ = „toll roads“)

# „Gaming is hard“, Part I

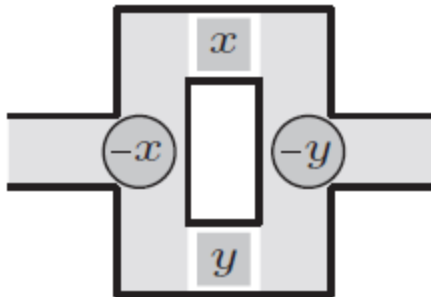


es gibt sowohl Druckplatten, welche  
Türen öffnen, als auch schließen



**Metatheorem 4.** *If a game features **doors** and **pressure plates**, and the avatar has to reach an exit location in order to win, then:*

*Even if no two pressure plates control the same door, the game is **NP-hard**.*



Alles wie bei MT 1: damit alle Punkte besucht werden  
müssen, sind vor dem Ausgang  $n$  Türen, welche mit  
Druckplatten bei jedem Punkt geöffnet werden  
müssen

„single-use-path“ mit Druckplatten

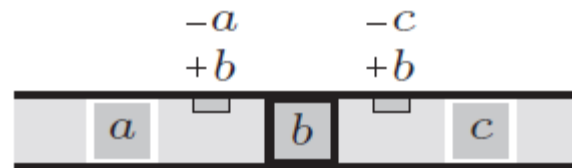
# „Gaming is hard“, Part I



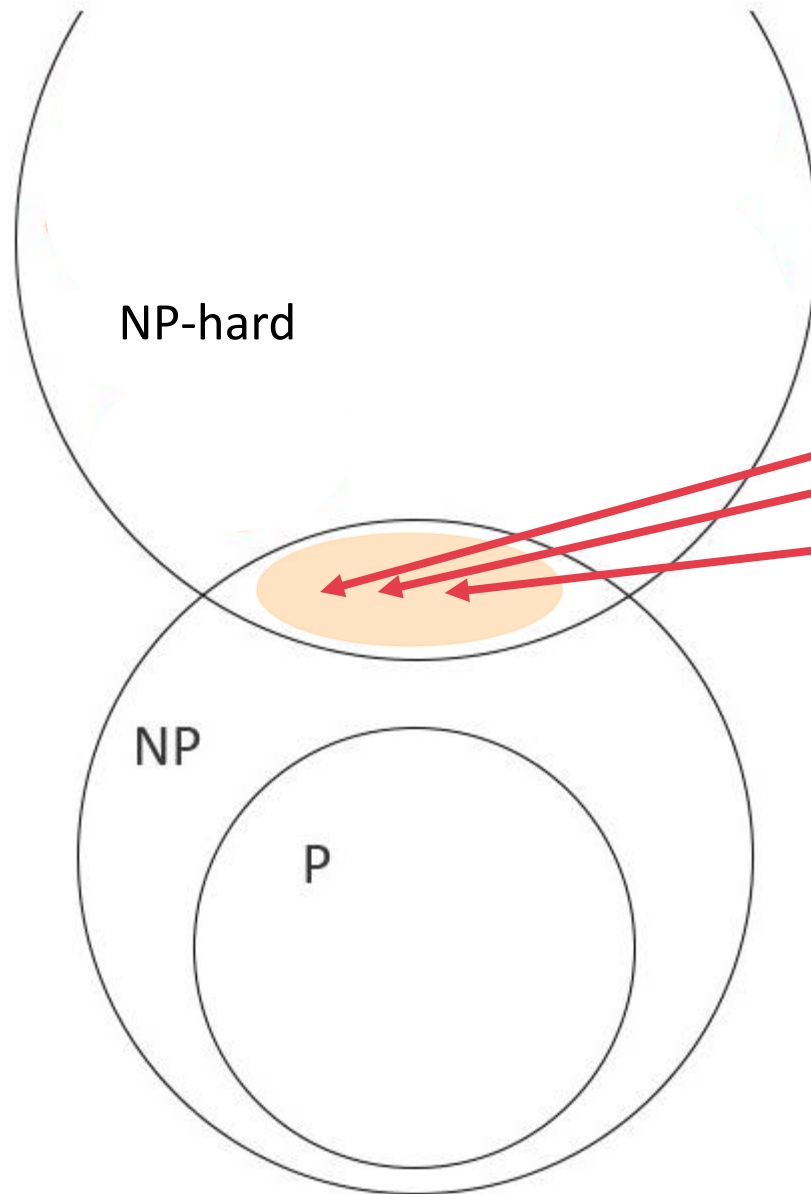
Knöpfe sind wie Druckplatten, außer dass der Spieler entscheiden kann, ob er ihn drücken will.  
Ein  $k$ -Button beeinflusst  $k$  Türen

**Metatheorem 5.** *If a game features **doors** and  **$k$ -buttons**, and the avatar has to reach an exit location in order to win, then:*

*If  $k \geq 2$ , then the game is **NP-hard**.*



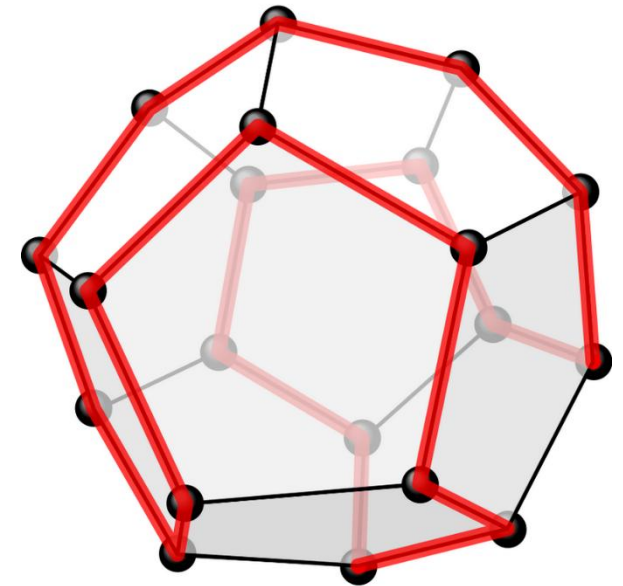
„single-use-path“ mit Türen und „2-buttons“

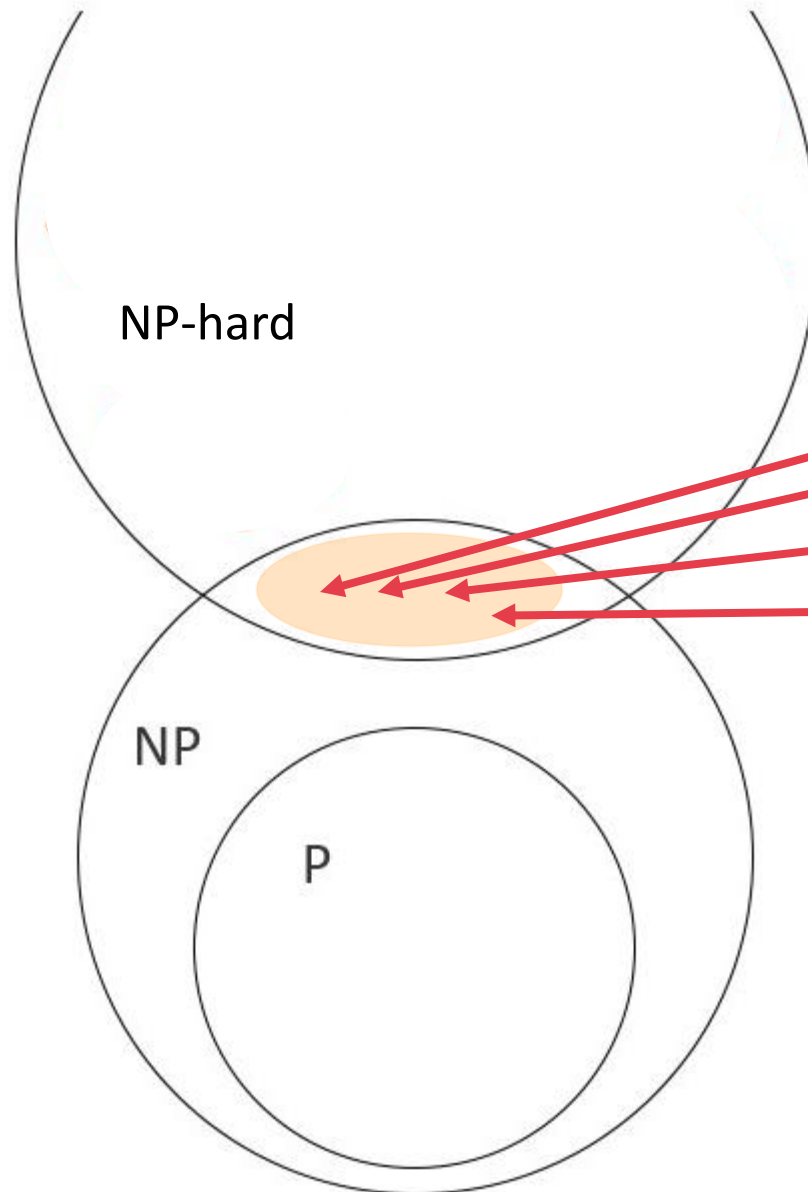


$SAT = \{F \mid F \text{ ist aussagenlogische Formel und erfüllbar}\}$

MINESWEEPER ist NP-vollständig

Hamiltonkreis ist NP-vollständig





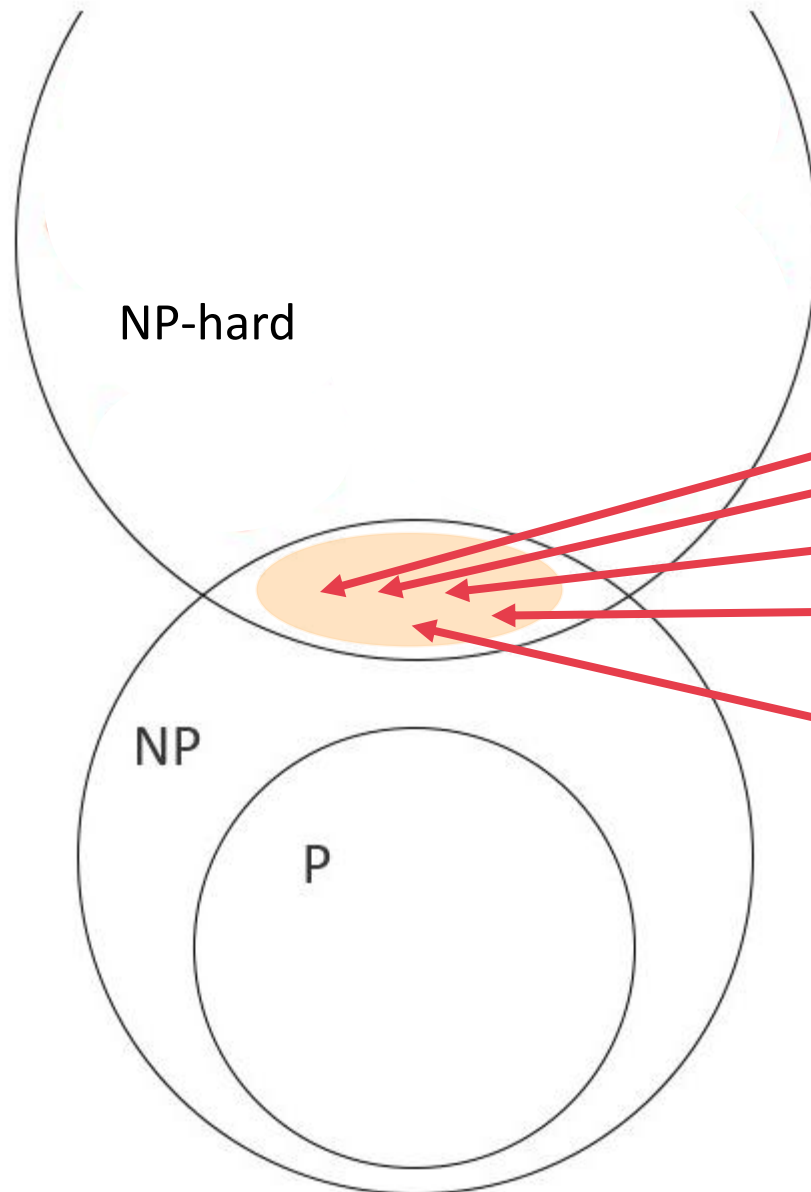
$$SAT = \{F \mid F \text{ ist aussagenlogische Formel und erfüllbar}\}$$

MINESWEEPER ist NP-vollständig

Hamiltonkreis ist NP-vollständig

Traveling Salesman ist NP-vollständig





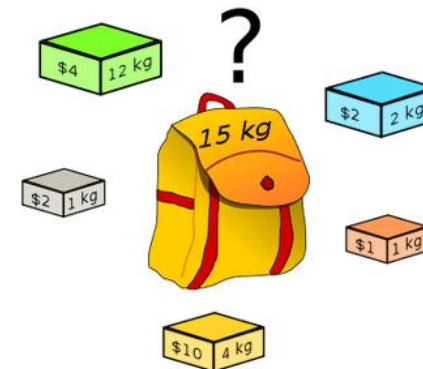
$SAT = \{F \mid F \text{ ist aussagenlogische Formel und erfüllbar}\}$

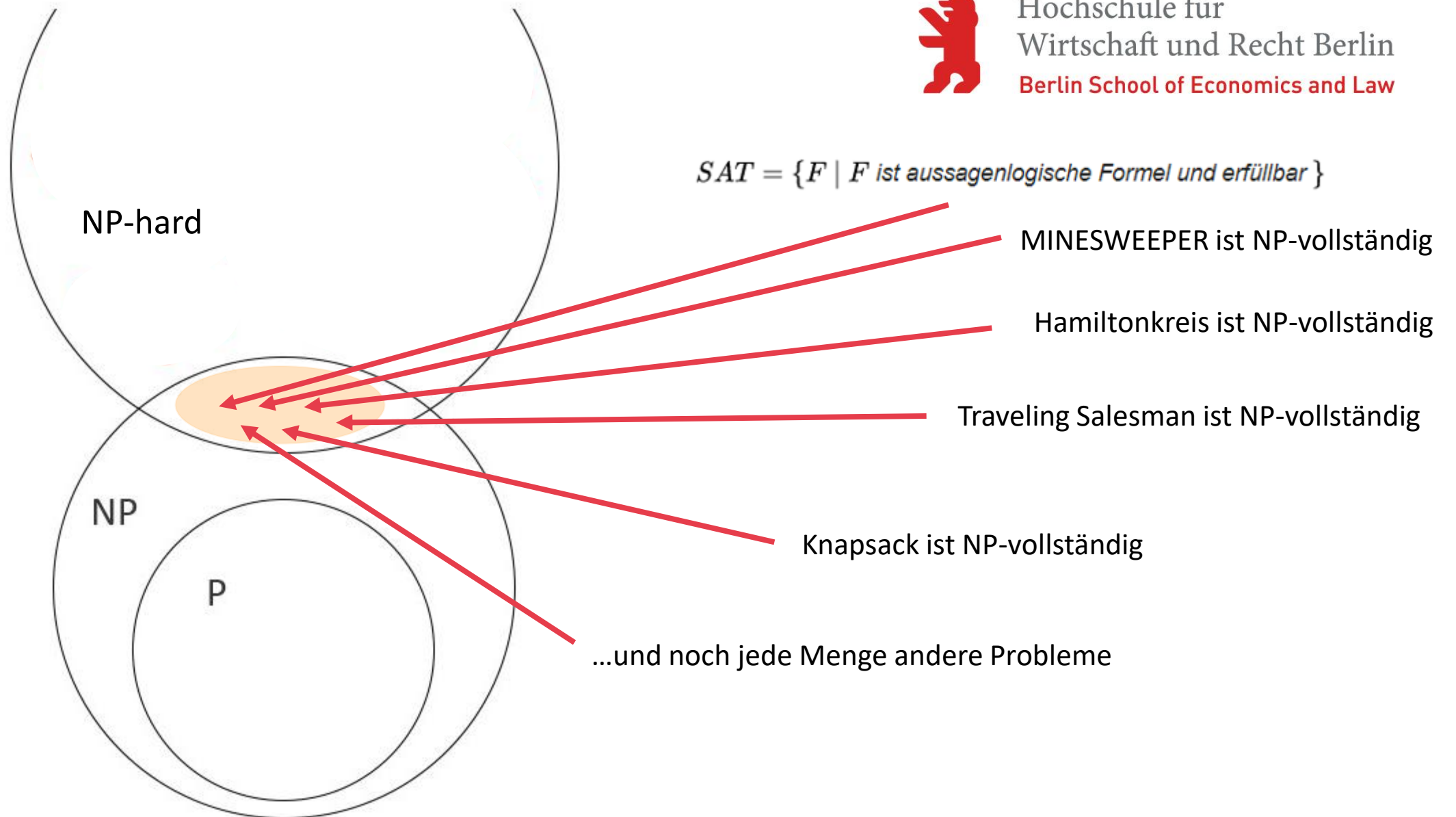
MINESWEEPER ist NP-vollständig

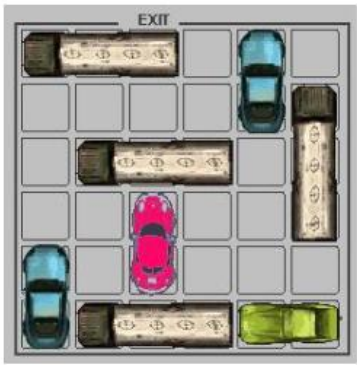
Hamiltonkreis ist NP-vollständig

Traveling Salesman ist NP-vollständig

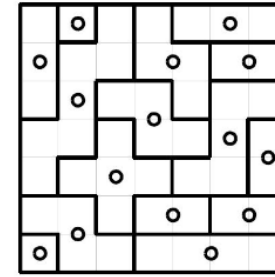
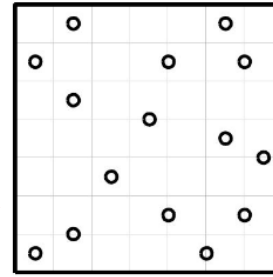
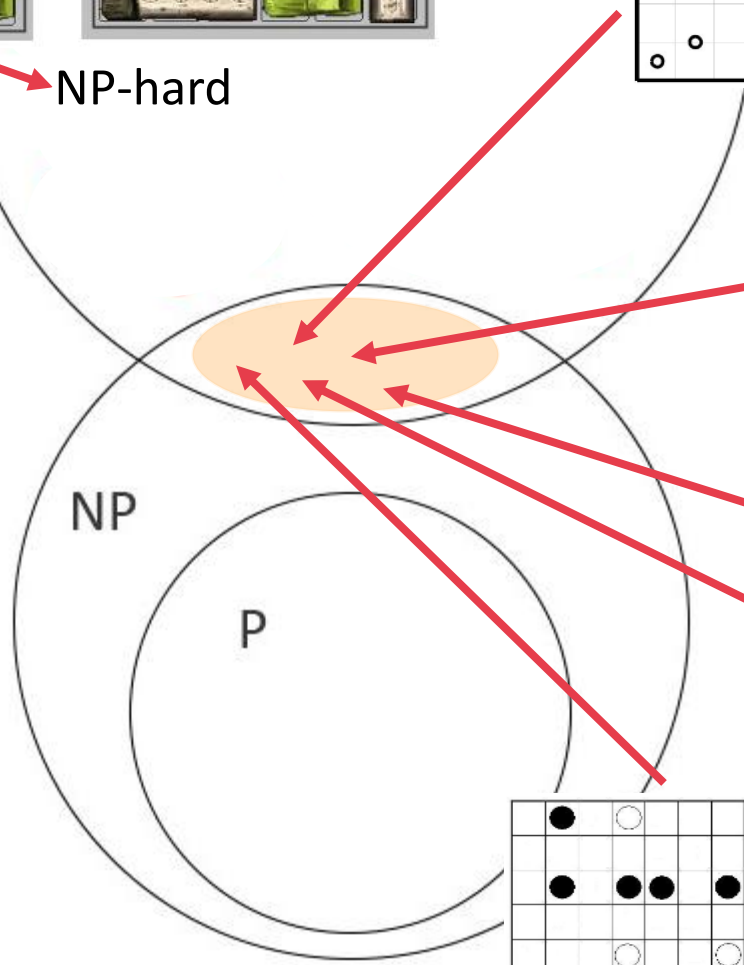
Knapsack ist NP-vollständig



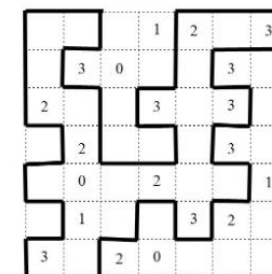
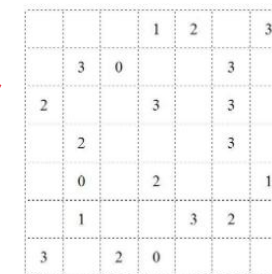
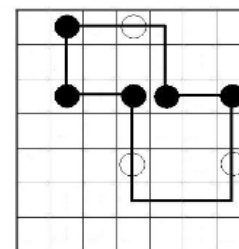
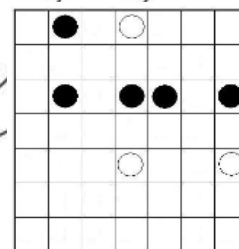
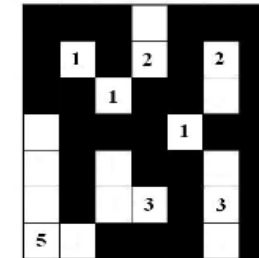
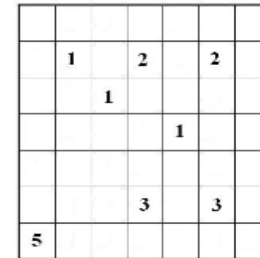
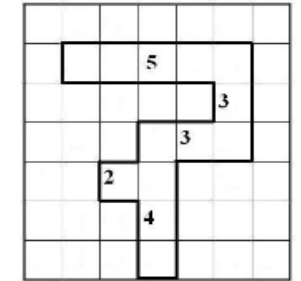
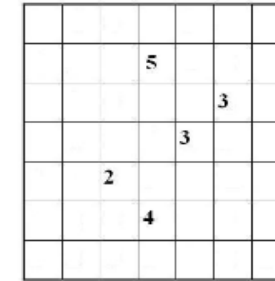




NP-hard



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law



# Karp's List

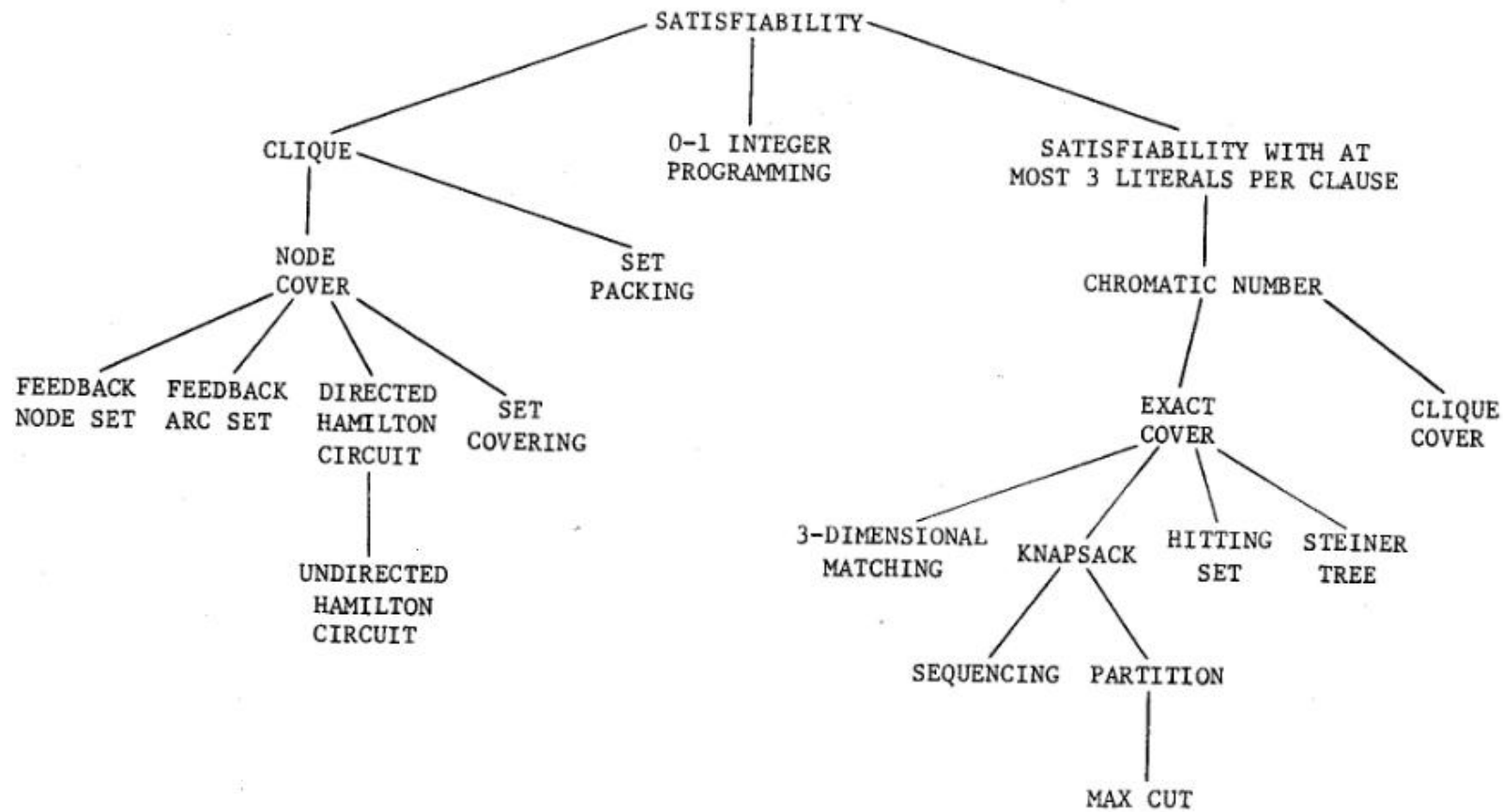


FIGURE 1 - Complete Problems



# Space Complexity

Let  $M$  be a deterministic Turing machine that halts on all inputs. The **space complexity** of  $M$  is the function  $f: \mathcal{N} \rightarrow \mathcal{N}$ , where  $f(n)$  is the **maximum number of tape cells** that  $M$  scans on any input of length  $n$ . If the space complexity of  $M$  is  $f(n)$ , we also say that  $M$  runs in space  $f(n)$ .

If  $M$  is a **nondeterministic** Turing machine wherein all branches halt on all inputs, we define its space complexity  $f(n)$  to be the maximum number of tape cells that  $M$  scans on **any branch** of its computation for any input of length  $n$ .

(in der Literatur wird hier gelegentlich angenommen, dass die TM ein Eingabe- und ein Arbeitsband besitzt und dass **nur der die Zellen auf dem Arbeitsband** gezählt werden)



# Space Complexity

Let  $f: \mathcal{N} \rightarrow \mathcal{R}^+$  be a function. The *space complexity classes*, **SPACE**( $f(n)$ ) and **NSPACE**( $f(n)$ ), are defined as follows.

$\text{SPACE}(f(n)) = \{L \mid L \text{ is a language decided by an } O(f(n)) \text{ space deterministic Turing machine}\}.$

$\text{NSPACE}(f(n)) = \{L \mid L \text{ is a language decided by an } O(f(n)) \text{ space nondeterministic Turing machine}\}.$

# Savitch's Theorem



## THEOREM

**Savitch's theorem** For any function  $f: \mathcal{N} \rightarrow \mathcal{R}^+$ , where  $f(n) \geq n$ ,  
 $\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n))$ .

*gilt auch für  
 $f(n) \geq \log(n)$*





# Savitch's Theorem

Für eine nichtdet. TM  $N$ , welche die Sprache  $A$  mit Speicher  $f(n)$  entscheidet konstruieren wir eine detem. TM  $M$ , welche die Sprache  $A$  in  $O(f(n)^2)$  entscheidet:

$M =$  "On input  $w$ :

1. Output the result of  $\text{CANYIELD}(c_{\text{start}}, c_{\text{accept}}, 2^{df(n)})$ ."

Entscheide ob Konfig.  $c_2$  von  
Konfig.  $c_1$  in  $t$  Schritten  
erreicht werden kann

Da  $f(n)$  zu Beginn nicht bekannt ist,  
muss  $M$  nacheinander  $f(n) = 1, 2, 3, \dots$   
durchtesten

$\text{CANYIELD} =$  "On input  $c_1, c_2$ , and  $t$ :

1. If  $t = 1$ , then test directly whether  $c_1 = c_2$  or whether  $c_1$  yields  $c_2$  in one step according to the rules of  $N$ . *Accept* if either test succeeds; *reject* if both fail.
2. If  $t > 1$ , then for each configuration  $c_m$  of  $N$  on  $w$  using space  $f(n)$ :
3. Run  $\text{CANYIELD}(c_1, c_m, \frac{t}{2})$ .
4. Run  $\text{CANYIELD}(c_m, c_2, \frac{t}{2})$ .
5. If steps 3 and 4 both accept, then *accept*.
6. If haven't yet accepted, *reject*."

Jeder Level benötigt  $O(f(n))$  Speicher

Die Rekursionstiefe beträgt  $O(\log t)$ ,  
und damit insgesamt  $O(f(n)^2)$

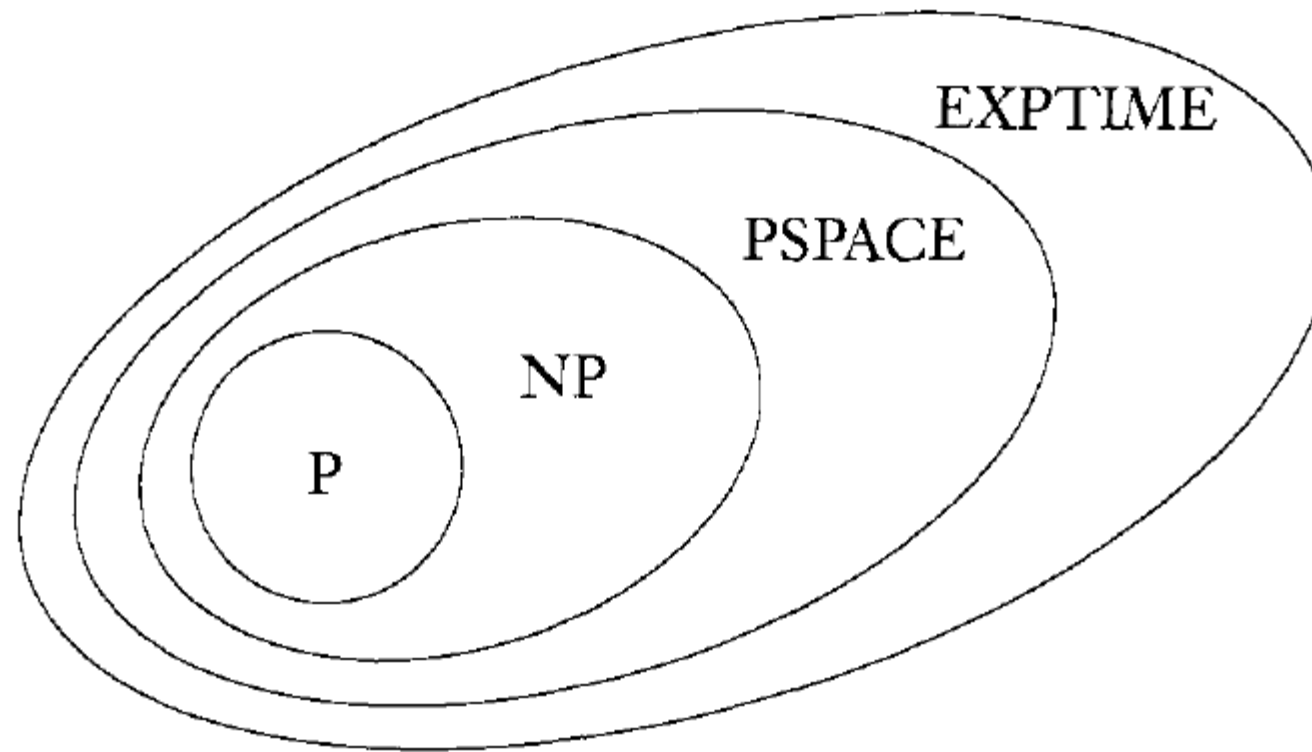
# PSPACE



**PSPACE** is the class of languages that are decidable in **polynomial space on a deterministic** Turing machine. In other words,

$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k).$$

# Time & Space



$$P \subseteq NP \subseteq PSPACE = NPSpace \subseteq EXPTIME = \bigcup_k \text{TIME}(2^{n^k})$$

# PSPACE-Vollständigkeit



A language  $B$  is **PSPACE-complete** if it satisfies two conditions:

1.  $B$  is in PSPACE, and
2. every  $A$  in PSPACE is polynomial time reducible to  $B$ .

If  $B$  merely satisfies condition 2, we say that it is **PSPACE-hard**.

# True Quantified Boolean Formulas



Hochschule für  
Wirtschaft und Recht Berlin  
Berlin School of Economics and Law

$TQBF = \{\langle \phi \rangle \mid \phi \text{ is a true fully quantified Boolean formula}\}$

## **THEOREM**

$TQBF$  is PSPACE-complete.