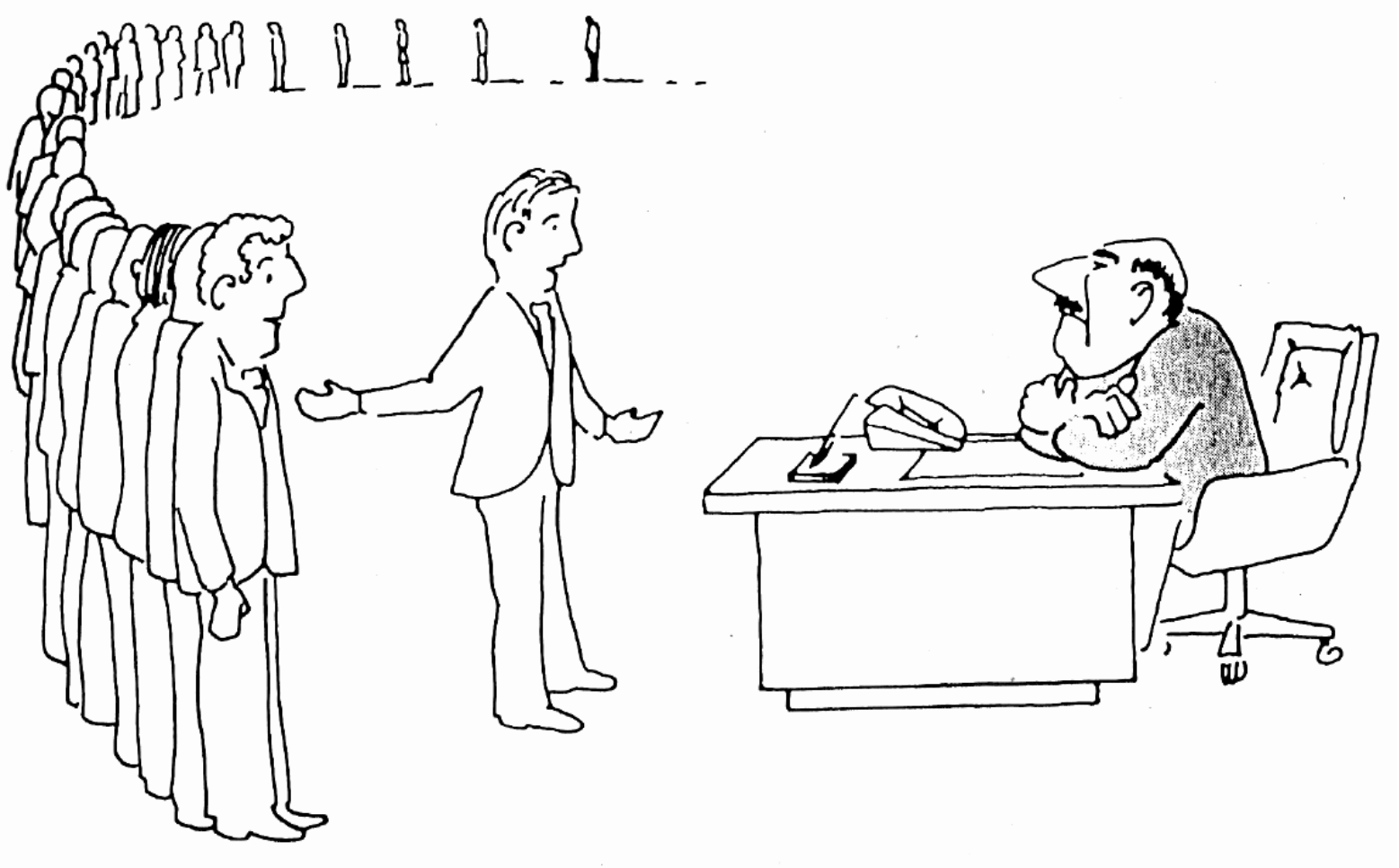
# Theoretische Informatik II

## Einheit 5.2

## Das $\mathcal{P}$–$\mathcal{NP}$ Problem

1. Nichtdeterministische Lösbarkeit
2. Sind $\mathcal{NP}$-Probleme handhabbar?
3. $\mathcal{NP}$-Vollständigkeit
4. Der Satz von Cook

# Wenn ein Problem nicht effektiv lösbar zu sein scheint



"I can't find an efficient algorithm, but neither can all these famous people."

**Vielleicht der einzig mögliche Weg**

# Welche Art von Problemen betrifft dies?

- **Travelling Salesman (TSP)** (Message Routing)
  Gibt es eine Rundreise zwischen $n$ Städten mit maximalen Kosten $B$?
- **Cliquen-Problem (CLIQUE)**
  Hat $G$ einen vollständig verbundenen Teilgraphen der Größe $k$?
- **Erfüllbarkeitsproblem (SAT)**
  Ist eine aussagenlogische Formel in KNF der Größe $n$ erfüllbar?
- **Multiprozessor-Scheduling**
  Können $n$ Prozesse derart auf eine Menge von Prozessoren verteilt werden, daß alle in Zeit $t$ abgearbeitet sind?
- **Binpacking**
  Können $n$ verschieden große Gegenstände in maximal $k$ Verpackungsbehältern untergebracht werden?

**Keine polynomielle Lösung bekannt**
**Beste Lösung ist Durchsuchen aller Möglichkeiten**

# ... aber Erfolg der Suche ist leicht zu testen

- **Travelling Salesman:** Für eine gegebene Rundreise $i_1..i_n$ können die Kosten $c_{i_1 i_2} + .. + c_{i_n i_1}$ in linearer Zeit berechnet und mit der Kostenbeschränkung $B$ verglichen werden
- **Cliquen-Problem:** Ein gegebener Teilgraph der Größe $k$ kann in polynomieller Zeit auf Vollständigkeit überprüft werden
- **Erfüllbarkeitsproblem:** Man kann in polynomieller Zeit testen, ob eine gegebene Belegung der Variablen eine Formel erfüllt
- **Multiprozessor-Scheduling**
  Man kann in polynomieller Zeit testen, ob eine gegebene Verteilung von Prozessen ein Ressourcenlimit einhält.
- **Binpacking:** Man kann in polynomieller Zeit testen, ob eine gegebene Verteilung der Gegenstände in $k$ Verpackungsbehälter paßt
- **Zusammengesetztheitstest:** Man kann in quadratischer Zeit testen, ob eine gegebene Zahl Teiler von $x$ (also $x$ keine Primzahl) ist

# Welches Modell kann diesen Effekt beschreiben?

## Der Zeitaufwand liegt in der Suche, nicht im Test

- **Orakel-Turingmaschinen (Raten und Verifizieren)**

  1. Bei Eingabe von $w \in \Sigma$ erzeugt Orakel einen Lösungsvorschlag $x$
  2. Verifizierer $V$ überprüft $w, x$ deterministisch

  OTM akzeptiert $w$, wenn es ein $x$ mit $w, x \in L(V)$ gibt

- **Berechnungsaufwand einer OTM bei Eingabe $w$**

  Maximale Rechenzeit für die Prüfung eines Lösungsvorschlags für $w$

  $=$ Ein Schritt für das Raten einer Lösung

  $+$ Konventionelle Rechenzeitdefinition für Überprüfung von $w, x$

- **OTM Modell ist äquivalent zu NTMs** §4.1

  – NTM $M$ akzeptiert, wenn ein Lösungsweg zum Erfolg führt

  – $t_M(w)$ ist maximale Zahl der Konfigurationsübergänge bis Terminierung

- **Polynomielle "Lösung" vieler schwerer Probleme**

  – Aber: deterministische Simulation von OTMs/NTMs wäre exponentiell

# Komplexität von Sprachen / Problemen

- **Zeitkomplexität:** (deterministisch & nichtdeterministisch)
  - Eine Sprache **$L$ hat deterministische Zeitkomplexität $\mathcal{O}(f)$**, falls es eine DTM $M$ mit $T_M \in \mathcal{O}(f)$ und $L = L(M)$ gibt
  - **$L$ hat nichtdeterministische Zeitkomplexität $\mathcal{O}(f)$**, falls es eine NTM $M$ (oder eine OTM) mit $T_M \in \mathcal{O}(f)$ und $L = L(M)$ gibt

  **TIME($f$)** $= \{L \mid L$ hat deterministische Zeitkomplexität $\mathcal{O}(f)\}$
  **NTIME($f$)** $= \{L \mid L$ hat nichtdeterministische Zeitkomplexität $\mathcal{O}(f)\}$

  – Statt "Sprache $L$" wird oft auch "Problem $P$" oder "Menge $M$" benutzt

- **Platzkomplexität**
  - **$L$ hat (nicht-)deterministische Platzkomplexität $\mathcal{O}(f)$**, falls $L = L(M)$ für eine DTM (bzw. NTM oder OTM) $M$ mit $S_M \in \mathcal{O}(f)$

  **SPACE($f$)** $= \{L \mid L$ hat Platzkomplexität $\mathcal{O}(f)\}$
  **NSPACE($f$)** $= \{L \mid L$ hat nichtdeterministische Platzkomplexität $\mathcal{O}(f)\}$

**Begriffe für abstrakte Algorithmen analog**

# Wichtige Komplexitätsklassen

- $\mathcal{P} = \bigcup_k TIME(n^k)$
  - Klasse der in polynomieller Zeit ($\hat{=}$ effizient) lösbaren Probleme
  - z.B. Arithmetische Operationen, Sortieren, Matrixmultiplikation, ...
- $\mathcal{NP} = \bigcup_k NTIME(n^k)$
  - Klasse der nichtdeterministisch in polynomieller Zeit lösbaren Probleme
  - z.B. TSP, CLIQUE, SAT, Multiprozessor-Scheduling, Binpacking, ...
- **Weitere Klassen und ihre Hierarchie**

$$LOGTIME \subseteq NLOGTIME \subseteq LOGSPACE \subseteq NLOGSPACE$$
$$\subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq PSPACE = NPSPACE$$
$$\subseteq EXPTIME \subseteq NEXPTIME \subseteq EXPSPACE \subseteq \ldots$$

  - Es wird vermutet, daß alle Inklusionen echt sind

**Probleme in $\mathcal{P}$ sind effizient lösbar (handhabbar)**
**Was wissen wir über Probleme in $\mathcal{NP}$ ?**

# Das $\mathcal{P}$–$\mathcal{NP}$ Problem

## Sind $\mathcal{NP}$ Probleme effizient lösbar?

- **Gilt $\mathcal{P}=\mathcal{NP}$ oder $\mathcal{P}\neq\mathcal{NP}$ ?**
  - Eines der wichtigsten offenen Probleme der TI
  - Seit mehr als 30 Jahren ungeklärt, möglicherweise unlösbar
- **Mehr als 1000 algorithmische Probleme betroffen**
  - Suchprobleme (Travelling Salesman, ...)
  - Reihenfolgenprobleme (Scheduling, Binpacking, ...)
  - Graphenprobleme (Clique, Vertex cover, ...) $\mapsto$ Operations Research
  - Logische Probleme (Erfüllbarkeit,...) $\mapsto$ Model Checking, Hardwareverifikation
  - Zahlenprobleme (Primfaktorisierung, ...) $\mapsto$ Kryptographie, IT Sicherheit
- **Indizien sprechen gegen $\mathcal{P}=\mathcal{NP}$**
  - Zu viele $\mathcal{NP}$-Probleme ohne bekannte polynomielle Lösung
  - Über 1000 äquivalente Probleme in 'schwerster Teilklasse' von $\mathcal{NP}$

# Wie analysiert man "$\mathcal{P}=\mathcal{NP}$ oder $\mathcal{P}\neq\mathcal{NP}$"?

- **Untersuche die "schwierigsten" $\mathcal{NP}$-Probleme**
  - Kann man eines davon effizient lösen?
  - Wenn ja, dann gilt $\mathcal{P}=\mathcal{NP}$
  - Wenn nein, dann gibt es ein Beispiel für $\mathcal{P}\neq\mathcal{NP}$
- **Was heißt "$L$ ist schwierigstes $\mathcal{NP}$-Problem"?**
  - Jedes andere $\mathcal{NP}$-Problem $L'$ ist nicht schwerer als $L$
  - Lösungen für $L$ könnten in Lösungen für $L'$ umgewandelt werden
  - Transformation der Lösungen muß effizient sein

$$\Downarrow$$

**Polynomielle Reduktion**

# Reduktion mit polynomiellem Zeitaufwand

- **Wiederverwendung bekannter Ergebnisse** (vgl. §4.6, F 14)
  - Zur Lösung eines Problems $L$ bzw. zum Nachweis der Unlösbarkeit
- **Methodik zum Nachweis der Unlösbarkeit**
  - Transformiere ein bekannt unlösbares Problem $L_1$ in Spezialfälle von $L$
  - Zeige, wie Lösungen für $L$ in Lösungen für $L_1$ transformiert würden
  - $L$ kann also nicht lösbar sein
- **Methodik zur Konstruktion einer Lösung**
  - Transformiere $L$ in Spezialfälle eines effizient lösbaren Problems $L_2$
  - Zeige, wie Lösung für $L$ aus der für $L_2$ entsteht
- **Formales Konzept: Polynomielle Reduzierbarkeit**
  - $\boldsymbol{L' \leq_p L}$ (**$L'$ polynomiell reduzierbar auf $L$**), falls $L' = f^{-1}(L)$ für eine totale, in polynomieller Zeit berechenbare Funktion $f$

$f$ transformiert Eingaben $x \in L'$ in $f(x) \in L$, aber das Lösungsverfahren für $L$ rückwärts(!) auf $L'$

Achtung: umgangssprachlich wird zuweilen die Problemstellung $L$ auf den Spezialfall $L'$ "reduziert"

# Polynomielle Reduktion auf Graphenproblemen

- **Cliquen Problem**
  - Gegeben ein Graph $G = (V, E)$ der Größe $n$ und eine Zahl $k \leq |V|$
  - Gibt es in $G$ eine Clique (vollständig verbundene Knotenmenge $V' \subseteq V$) der Mindestgröße $k$?

$$\boldsymbol{CLIQUE} = \{\ (G,k) \mid G{=}(V,E) \text{ Graph } \wedge\ (\exists V_c \subseteq V.\ |V_c| \geq k \wedge\ V_c \text{ Clique in } G)\ \}$$

- **Vertex Cover Problem**
  - Gegeben ein Graph $G = (V, E)$ der Größe $n$ und eine Zahl $k \leq |V|$
  - Gibt es eine Teilmenge $V' \subseteq V$ mit höchstens $k$ Elementen, so daß aus jeder Kante in $G$ mindestens eine Ecke in $V'$ liegt?

$$\boldsymbol{VC} = \{\ (G,k) \mid G \text{ Graph } \wedge\ (\exists V' \subseteq V.\ |V'| \leq k \wedge\ V' \text{ Knotenüberdeckung von } G)\ \}$$

**Probleme sind aufeinander reduzierbar**

# Reduzierbarkeit: Clique $\leq_p$ Vertex Cover

- **Analyse der Eigenschaften**

  $V_c$ ist Clique in $G = (V, E)$

  $\Leftrightarrow \forall v, v' \in V_c.\ v \neq v' \Rightarrow \{v, v'\} \in E$ (Definition)

  $\Leftrightarrow \forall \{v, v'\} \notin E.\ v \notin V_c \ \vee\ v' \notin V_c$ (Kontraposition)

  $\Leftrightarrow \forall \{v, v'\} \in E^c.\ v \in V - V_c \ \vee\ v' \in V - V_c$ (Positive Formulierung)

  $\Leftrightarrow$ $V - V_c$ Knotenüberdeckung des Komplementgraphen $G^c = (V, E^c)$

- **Transformation der Probleme**

  Wähle $f(G, k) := (G^c, |V| - k)$

  Dann ist $f$ in polynomieller Zeit $\mathcal{O}(|V|^2)$ berechenbar und es gilt

  $(G, k) \in CLIQUE$

  $\Leftrightarrow$ $G$ hat Clique $V_c$ der Mindestgröße $k$

  $\Leftrightarrow$ $G^c$ hat Knotenüberdeckung $V' = V - V_c$ der Maximalgröße $|V| - k$

  $\Leftrightarrow$ $(G^c, |V| - k) \in VC$

  also $\boldsymbol{CLIQUE = f^{-1}(VC)}$ $\checkmark$

# Entscheidung, Berechnung oder Optimierung?
## Problemvarianten sind gegenseitig "reduzierbar"

- **Löse Optimierungsproblem mit Entscheidung**

  $CLIQUE_{opt}$: *Bestimme die Größe $k$ einer maximalen Clique in $G$*

  – Beginne mit $k := |V|$ und teste ob es in $G = (V, E)$ eine $k$-Clique gibt

  – Reduziere $k$ bis der Test erfolgreich ist und gebe $k_{opt} := k$ aus

  – Zusatzaufwand linear in $|V|$

- **Löse Berechnungsproblem mit Optimierung**

  $CLIQUE_2$: *Bestimme eine Clique $C \subseteq G$ mit maximaler Größe $k$*

  – Bestimme $k_{opt}$ für $G$ und beginne mit $E_c := E$

  – Wähle Kante $e \in E$ und teste, ob es in $(V, E_c - \{e\})$ eine $k_{opt}$-Clique gibt

  · Ist dies der Fall, so setze $E_c := E_c - \{e\}$

  – Wiederhole dies iterativ für alle Kanten aus $E$

  – Das Endergebnis $E_c$ und die zugehörigen Knoten bilden die $k_{opt}$-Clique

  – Zusatzaufwand linear in $|E|$

- **Die Umkehrungen sind trivial**

**Es reicht, Entscheidungsprobleme zu analysieren**

# $\mathcal{NP}$-Vollständigkeit

- **Reduzierbarkeit bedeutet geringere Komplexität**
  - $L \leq_p L' \wedge L' \in \mathcal{P} \quad \Rightarrow \quad L \in \mathcal{P}$
  - $L \leq_p L' \wedge L' \in \mathcal{NP} \quad \Rightarrow \quad L \in \mathcal{NP}$

  **Beweis analog zu allgemeiner Reduzierbarkeit:**
  - $\chi_L(x)=1 \Leftrightarrow x \in L \Leftrightarrow f(x) \in L' \Leftrightarrow \chi_{L'}(f(x))=1 \Leftrightarrow (\chi_{L'} \circ f)(x)=1$
  - $\chi_{L'} \circ f$ ist in polynomieller Zeit berechenbar, wenn dies für $\chi_{L'}$ gilt
- **$\mathcal{NP}$-hart: nicht leichter als $\mathcal{NP}$**
  - **$L'$ ist $\mathcal{NP}$-hart**, genau dann wenn $L \leq_p L'$ für alle $L \in \mathcal{NP}$ gilt
- **$\mathcal{NP}$-vollständig: schwierigste Teilklasse in $\mathcal{NP}$**
  - **$L'$ ist $\mathcal{NP}$-vollständig**, wenn $L$ $\mathcal{NP}$-hart und $L \in \mathcal{NP}$
  - Schreibweise: $L \in \mathcal{NPC}$

# Konsequenzen von $\mathcal{NP}$-Vollständigkeit

- **Alle $\mathcal{NP}$-vollständigen Probleme sind äquivalent**
  - $L, L' \in \mathcal{NPC} \Rightarrow L' \leq_p L \wedge L \leq_p L'$
- **$\mathcal{NP}$-vollständige Probleme entscheiden '$\mathcal{P} \stackrel{?}{=} \mathcal{NP}$'**
  - $\mathcal{P}=\mathcal{NP} \Leftrightarrow \exists L \in \mathcal{NPC}.\, L \in \mathcal{P} \Leftrightarrow \forall L \in \mathcal{NPC}.\, L \in \mathcal{P}$ Satz 10.5
  - Ist $\mathcal{P}=\mathcal{NP}$ dann sind alle $\mathcal{NP}$-vollständigen Probleme in $\mathcal{P}$
  - $\mathcal{P}\neq\mathcal{NP} \Leftrightarrow \exists L \in \mathcal{NPC}.\, L \notin \mathcal{P} \Leftrightarrow \forall L \in \mathcal{NPC}.\, L \notin \mathcal{P}$
  - Ist $\mathcal{P}\neq\mathcal{NP}$ dann sind alle $\mathcal{NP}$-vollständigen Probleme nicht in $\mathcal{P}$
- **$\mathcal{NP}$-Vollständigkeit ist leicht nachweisbar, wenn ein $\mathcal{NP}$-vollständiges Problem bekannt ist**
  - $L \in \mathcal{NPC} \Leftrightarrow L \in \mathcal{NP} \wedge \exists L' \in \mathcal{NPC}.\, L' \leq_p L$ Satz 10.4
  - $L \in \mathcal{NPC} \Leftrightarrow \exists L' \in \mathcal{NPC}.\, L' \leq_p L \wedge L \leq_p L'$

**$\mathcal{NP}$-Vollständigkeit muß einmal explizit gezeigt werden**

# Wie zeigt man $\mathcal{NP}$-Vollständigkeit?

**Beweise $\mathcal{NP}$-Vollständigkeit explizit für eine Sprache $L$**

- **Codiere Berechnungen beliebiger NTMs in $L$**
  - Codierung soll zu Sprache $L$ gehören, wenn Maschine $M$ akzeptiert
  - Codierung soll nicht zu $L$ gehören, wenn $M$ nicht akzeptiert
  - Codierung 'polynomieller NTMs' muß in polynomieller Zeit geschehen

  Damit ist $L(M) \leq_p L$ für jede polynomielle NTM $M$, d.h. $L$ ist $\mathcal{NP}$-hart
- **Sprache $L$ muß selbst in $\mathcal{NP}$ liegen**
  - Ergibt zusammen mit dem obigen die $\mathcal{NP}$-Vollständigkeit von $L$
- **Welches Sprache ist ausdrucksstark genug?**
  - Idee: codiere mögliche Zustandsübergänge durch logische Formeln
  - Problemstellung: Können Zustandsübergänge so kombiniert werden, daß eine terminierende Berechnung codiert wird?
  - Erfüllbarkeitsproblem der (Aussagen-)logik ist Kandidat für $\mathcal{NPC}$

# Das Erfüllbarkeitsproblem

**Ist eine aussagenlogische Formel in KNF erfüllbar?**

*Gegeben $m$ Klauseln $k_1, .., k_m$ über $n$ Variablen $x_1, .., x_n$. Gibt es eine Belegung $a_1, .., a_n \in \{0, 1\}$ der Variablen $x_i$, welche alle Klauseln erfüllt?*

- **Klausel über den Variablen $x_1, ..x_n$**
  - Disjunktion einiger Literale der Form $x_i$ bzw. $\underline{x_i}$
- **Belegung $a_1, ..., a_n \in \{0, 1\}$ erfüllt Klausel $k_j$**
  - Auswertung von $k_j$ unter $a_1, ..., a_n$ ergibt den Booleschen Wert 1
- $$\boldsymbol{SAT = \{k_1, ..k_m \mid \; k_i \text{ Klausel über } x_1, ..x_n \;\wedge\; (\exists a_1, ..a_n \in \{0,1\}.\; \forall j \leq m.\; a_1, ..a_n \text{ erfüllt } k_j)\}}$$

Codierbar als Teilmenge der Sprache der Aussagenlogik

# Beispiele von Formeln in KNF

$(\underline{x_1} \vee x_2) \;\wedge\; (x_1 \vee \underline{x_2} \vee \underline{x_3}) \;\wedge\; \underline{x_3}$ *erfüllbar*

– Setze $x_3$=0, $x_2$=1, $x_1$ beliebig, z.B. $x_1$=0

– Auswertung: $(\underline{0}+1) * (0+\underline{1}+\underline{0}) * \underline{0} = (1+1) * (0+0+1) * 1 = 1 * 1 * 1 = 1$

$x_1 \;\wedge\; \underline{x_1}$ *nicht erfüllbar*

– Jede Belegung ergibt den Wert 0

$(x_1 \vee x_2) \;\wedge\; (\underline{x_1} \vee \underline{x_2})$ *erfüllbar, Belegung: (1,0)*

$(x_1 \vee x_2) \;\wedge\; (x_1 \vee \underline{x_2}) \;\wedge\; (\underline{x_1} \vee x_2) \;\wedge\; (\underline{x_1} \vee \underline{x_2})$

*nicht erfüllbar*

$(x_1 \vee \underline{x_2} \vee x_3) \;\wedge\; (\underline{x_1} \vee x_2 \vee \underline{x_4}) \;\wedge\; (\underline{x_1} \vee \underline{x_2} \vee \underline{x_3})$

*erfüllbar, Belegung: (1,1,0,0)*

# Lösungen für das Erfüllbarkeitsproblem

$SAT = \{k_1..k_m |\ k_i$ Klausel über $x_1..x_n \wedge \exists a_1..a_n \in \{0,1\}.a_1..a_n$ erfüllt $k_1..k_m\}$

- **Deterministische Lösung**
  - Werte Klauseln für alle möglichen Belegungen der Variablen aus bis erfüllende Belegung gefunden ist
  - Es gibt $2^n$ möglichen Belegungen von $x_1, ..x_n$
  - Auswertung linear in Größe der Formel $\mathcal{O}(m * n)$
  - **Laufzeit ist in $\mathcal{O}(2^n)$**
- **Nichtdeterministisch: Raten und Verifizieren**
  - Orakel erzeugt erfüllende Belegung der Variablen (falls es eine gibt)
  - Prüfe Belegung durch Auswertung der Formel in polynomieller Zeit

$$\Downarrow$$

$$\boldsymbol{SAT \in \mathcal{NP}}$$

# $SAT$ ist $\mathcal{NP}$-vollständig (Satz von Cook)

- **Gegeben: NTM $M$, die in polynomieller Zeit terminiert**
- **Ziel: Codiere Berechnung von $M$ bei Eingabe $w$ durch KNF-Formel, die erfüllbar ist, g.d.w. $w \in L(M)$**
  - Codierung muß in polynomieller Zeit (relativ zu $|w|$) berechenbar sein
  - Codierung darf von Kenntnissen über $L(M)$ abhängen
- **Vorgehen: Beschreibe mögliche Konfigurationsübergänge von $M$ durch aussagenlogische Klauseln**
  - Codiere Zustand, Kopfposition und Bandzellen durch Literale
  - Es werden nur polynomiell viele Literale und Klauseln benötigt
  - Formel ist erfüllbar, wenn Konfigurationsübergänge zu akzeptierender Berechnung zusammengesetzt werden können

**Aufwendiger Beweis mit sehr vielen Details**

# Satz von Cook: Grundannahmen

**Zeige $L \leq_p SAT$ für jede Sprache $L \in \mathcal{NP}$**

- **$L$ wird von $NTM$ $M$ akzeptiert**
  - $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ mit $Q=\{q_0, .., q_k\}$, $\Gamma=\{X_1, .., X_m\}$
- **$M$ zeitbeschränkt durch Polynom $p(n)$**
  - $t_M(w) \leq p(n)$ für jedes Wort $w \in \Sigma^*$ mit $|w|=n$
  - Es sind genau $p(n)$ Berechnungsschritte als Formel zu codieren

    o.B.d.A.: $M$ 'verharrt' in den Endzuständen anstatt abzubrechen

    d.h. $(u,q,v) \vdash (u,q,v)$ für $q \in F$
- **$M$ ist auch platzbeschränkt durch $p(n)$**
  - $M$ kann während der Berechnung maximal $p(n)$ Bandzellen aufsuchen

    o.B.d.A.: $M$ arbeitet mit halbseitig unendlichem Band

**Es reicht, genau die Bandzellen $1..p(n)$ zu modellieren**

–Schreibe Konfiguration $(u,q,v)$ als String $\boldsymbol{uqvB^j}$ der Länge $p(n)+1$

# Satz von Cook: zu codierende Aussagen

- **Anfangsbedingungen bei Eingabe $w$**
  - $M$ startet im Zustand $q_0$ und der Kopf ist über Bandzelle 0
  - Anfangskonfiguration ist $\boldsymbol{q_0 w_1 .. w_n B^{p(n)-(n+1)}}$
- **Übergangsbedingungen**
  - Zu jedem Zeitpunkt $t$ steht der Kopf an einer Stelle $j$ und verändert Bandinhalt und Zustand entsprechend der Tabelle von $\delta$
- **Endbedingung**
  - Nach $p(n)$ Schritten befindet sich $M$ in einem Endzustand $q_f \in F$
  - Endkonfiguration hat die Form $\boldsymbol{X_0 .. X_{j-1} q_f X_{j+1} .. X_{p(n)}}$ für ein $j$
- **Randbedingungen für eindeutiges Verhalten**
  - Zu jedem Zeitpunkt $t$ befindet sich $M$ in genau einer Konfiguration $X_0 .. X_{j-1} q X_{j+1} .. X_{p(n)}$

**Summe der Aussagen codiert NTM-Berechnung**

# Die Codierung und ihre Korrektheit (Skizze)

- **Verwende Konfigurationsvariablen $y_{t,i,A}$**
  *"Nach $t$ Schritten steht an der $i$-ten Stelle der Konfiguration ein $A$"*
- **Codiere Aussagen durch KNF-Formeln $A, \ddot{U}, E, R$**
  – Methodik ähnlich zu Codierung von Berechnungen als PKP
  – Jede Teilformel ist in der Zeit $\mathcal{O}(p(n)^3)$ konstruierbar
- **Setze $\alpha(M,w) \equiv A \wedge \ddot{U} \wedge E \wedge R$**
  – **$\alpha(M,w)$ ist in KNF**, da jede der Teilformeln ist in KNF
  – **$\alpha(M,w)$ ist in polynomieller Zeit konstruierbar**
  – **$w \in L \Rightarrow \alpha(M,w) \in SAT$**
  Ist $w \in L$, dann gibt es eine akzeptierende Berechnung $K_0,..,K_{p(n)}$ für $w$.
  Setze: $y_{t,i,A}{:=}1$, falls $A$ das $i$-te Symbol von $K_t$ ist, und sonst $y_{t,i,A}{:=}0$.
  Per Konstruktion erfüllt dies die Formel $\alpha(M,w)$, also $\alpha(M,w) \in SAT$.
  – **$\alpha(M,w) \in SAT \Rightarrow w \in L$**
  Ist $\alpha(M,w)$ erfüllbar, so kann mit $\ddot{U}$ die Belegung der Variablen in eine Konfigurationsfolge $K_0,..,K_{p(n)}$ umgerechnet werden. Wegen $R$ gibt es genau eine solche Konfigurationsfolge. Wegen $A$ und $E$ repräsentiert diese Konfigurationsfolge eine akzeptierende Berechnung für $w$. Also gilt $w \in L$.

# Satz von Cook: Zusammenfassung

- **Aufwendige Codierung von Berechnungen**
  - Formel $\alpha(M,w)$ codiert Berechnung der NTM $M$ bei Eingabe $w$
  - $\alpha(M,w)$ ist in polynomieller Zeit berechenbar (relativ zu $|w|$)
  - Es gilt $w \in L(M) \Leftrightarrow \alpha(M,w) \in SAT$
  - Es folgt $L(M) \leq_p SAT$
- **Konstruktion ist uniform für polynomielle NTMs**
  - Es folgt $\boldsymbol{L \leq_p SAT}$ **für jedes** $\boldsymbol{L \in \mathcal{NP}}$
  - Lösungen für $SAT$ sind in Lösungen für jedes $L \in \mathcal{NP}$ transformierbar
- $\boldsymbol{SAT}$ **ist selbst in** $\boldsymbol{\mathcal{NP}}$
  - Belegungen können leicht als erfüllend überprüft werden

$$\Downarrow$$

**$SAT$ ist $\mathcal{NP}$-vollständig**

# Details der Codierung: Anfangsbedingungen

**Anfangskonfiguration ist** $q_0 w_1 .. w_n B^{p(n)-(n+1)}$

- **Verwende Konfigurationsvariablen** $y_{t,i,A}$
  - Zeit $t$ und Zelle $i$ sind Zahlen zwischen 0 und $p(n)$
  - $A$ ist ein Symbol aus $\Gamma$ oder ein Zustand $(A \in \{X_1, .., X_m, q_0, .., q_k\})$
- **Codiere Anfangsbedingungen als Formel $A$ mit**

$$A \equiv y_{0,0,q_0} \wedge y_{0,1,w_1} \wedge .. \wedge y_{0,n,w_n} \wedge y_{0,n+1,B} \wedge .. \wedge y_{0,p(n),B}$$

- **$A$ ist in KNF** — Rein konjunktive Formel
- **Größe: $\mathcal{O}(p(n))$** — $p(n)+1$ Variablen
- **Berechnungsaufwand: $\mathcal{O}(p(n))$** — Bestimmung von $p(n)$

# Details der Codierung: Übergangsbedingungen

## Konfigurationsübergänge sind verträglich mit $\delta$

**Definiere Formeln $\ddot{U}(t,i)$ für Zeit $t$ und Stelle $i$**

– *Falls $M$ an Stelle $i$ steht, kann sich der Bereich $i{-}1..i{+}1$ ändern*

$$(y_{t,i-1,Z} \wedge y_{t,i,q} \wedge y_{t,i+1,X})$$
$$\Rightarrow (y_{t+1,i-1,p_1} \wedge y_{t+1,i,Z} \wedge y_{t+1,i+1,Y_1})$$
$$\vee \;..\; \vee (y_{t+1,i-1,p_l} \wedge y_{t+1,i,Z} \wedge y_{t+1,i+1,Y_l})$$
$$\vee (y_{t+1,i-1,Z} \wedge y_{t+1,i,Y'_1} \wedge y_{t+1,i+1,p'_1})$$
$$\vee \;..\; \vee (y_{t+1,i-1,Z} \wedge y_{t+1,i,Y'_r} \wedge y_{t+1,i+1,p'_r})$$

für jedes $Z \in \Gamma$, falls $\delta(q,X) = \{(p_1,Y_1,L),..,(p_l,Y_l,L),\ (p'_1,Y'_1,R),..,(p'_r,Y'_r,R)\}$

– *Falls $M$ nicht im Bereich $i{-}1..i{+}1$ steht, bleibt Stelle $i$ unverändert*

$$(\underline{y_{t,i-1,q_0}} \wedge .. \wedge \underline{y_{t,i-1,q_k}} \wedge \underline{y_{t,i,q_0}} \wedge .. \wedge \underline{y_{t,i,q_k}} \wedge \underline{y_{t,i+1,q_0}} \wedge .. \wedge \underline{y_{t,i+1,q_k}})$$
$$\Rightarrow (y_{t,i,X_1} \wedge y_{t+1,i,X_1}) \vee \;..\; \vee (y_{t,i,X_m} \wedge y_{t+1,i,X_m})$$

# Details der Codierung: Übergangsbedingungen (II)

**Konfigurationsübergänge sind verträglich mit $\delta$**

- **Kombiniere Übergangsbedingungen zu Formel $\ddot{U}$**

$$\ddot{U} \equiv \ddot{U}(0,0) \wedge .. \wedge \ddot{U}(0,p(n)) \wedge \ddot{U}(p(n),0) \wedge .. \wedge \ddot{U}(p(n),p(n))$$

Formeln werden zuvor in KNF transformiert (Standardverfahren)

- **Ü ist in $KNF$** — Alle Ü$(t,i)$ wurden normalisiert

- **Größe: $\mathcal{O}(p(n)^2)$** — $p(n)^2$ Komponentenformeln

  Je Komponente nach Normalisierung maximal $k * m * 3^{2m*k} + 3k * 2^m$ Symbole

- **Berechnungsaufwand: $\mathcal{O}(p(n)^2)$**

# Details der Codierung: Endbedingung

**Endkonfiguration hat Form $X_0..X_{j-1}q_f X_{j+1}..X_{p(n)}$**

- Sei $F=\{q_r, .., q_e\}$

  Codiere Endbedingungen als Formel $E$ mit

$$
\begin{aligned}
E = \; & (y_{p(n),0,q_r} \vee .. \vee y_{p(n),0,q_e}) \\
\vee \; & (y_{p(n),1,q_r} \vee .. \vee y_{p(n),1,q_e}) \\
\vee \; & \vdots \\
\vee \; & (y_{p(n),p(n),q_r} \vee .. \vee y_{p(n),p(n),q_e})
\end{aligned}
$$

- $E$ ist in $KNF$ — Einfache Klausel
- Größe: $\mathcal{O}(p(n))$ — $p(n) * (e-r)$ Variablen
- Berechnungsaufwand: $\mathcal{O}(p(n))$

# Details der Codierung: Randbedingungen

## Eindeutige Konfiguration $X_0..X_{j-1}qX_{j+1}..X_{p(n)}$

- **Codiere Randbedingungen als Formel $R$:**
  - Zu jedem Zeitpunkt steht an jeder Stelle genau ein Symbol
  - Zu jedem Zeitpunkt steht nur an einer Stelle ein Zustand
  - Optimierungen möglich ("*maximal eine Konfiguration*" reicht)

$$R \equiv \exists_1(y_{0,0,X_1}, .., y_{0,0,q_k}) \wedge \ldots \wedge \exists_1(y_{p(n),p(n),X_1}, .., y_{p(n),p(n),q_k})$$
$$\wedge \exists_1(y_{0,0,q_1}, .., y_{0,p(n),q_k}) \wedge \ldots \wedge \exists_1(y_{p(n),0,q_1}, .., y_{p(n),p(n),q_k})$$

Dabei steht $\exists_1(x_1, .., x_m)$ für "genau eines der $x_i$ gilt'

$$\exists_1(x_1, .., x_j) \equiv (x_1 \vee .. \vee x_j) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge .. \wedge (\bar{x}_1 \vee \bar{x}_j)$$
$$\wedge (\bar{x}_2 \vee \bar{x}_3) \wedge .. \wedge (\bar{x}_2 \vee \bar{x}_j) \wedge .. \wedge (\bar{x}_{j-1} \vee \bar{x}_j)$$

- **$R$ ist in $KNF$** — Konjunktion von $\exists_1$-Formeln
- **Größe: $\mathcal{O}(p(n)^3)$** — $p(n)^2 * (m+k)^2 + p(n) * (k * p(n))^2$ Variablen
- **Berechnungsaufwand: $\mathcal{O}(p(n)^3)$** — Bestimme $p(n)$,...