

# Die Klasse $P$

Die Klasse  $P$  enthält alle (Entscheidungs-)Probleme mit einer **deterministischen polynomiellen** Lösung.

## Beispiele

*insort*, *quicksort*, *mergesort*, Matrizenmultiplikation,  
Wortproblem für kontextfreie Sprachen, . . .

# Die Klasse *NP*

Die Klasse *NP* enthält alle (Entscheidungs-)Probleme mit einer nichtdeterministischen polynomiellen Lösung.

## Beispiele

Traveling Salesperson, Hamiltonsche Kreise, Rucksack, Stundenplanung, Maschinenbelegung, Sudoku, . . .

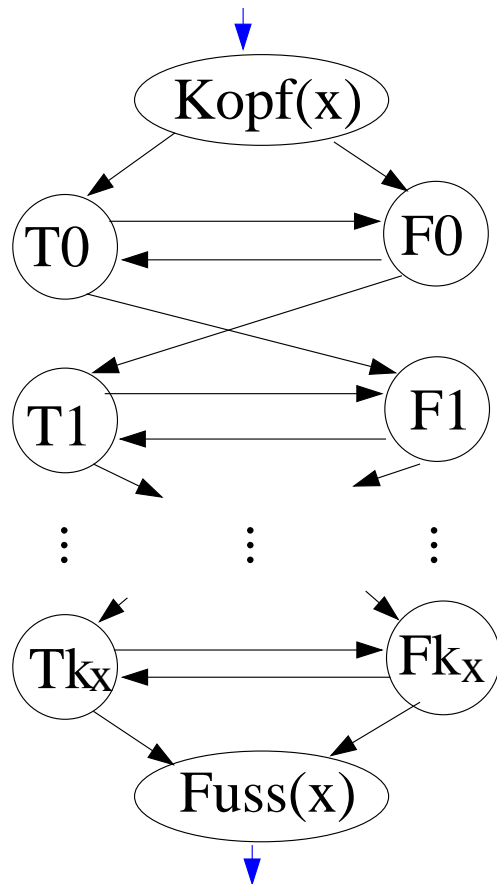
## Theorem

DHC ist  $NP$ -vollständig.

**Beweisidee:** Reduziere 3SAT auf DHC.

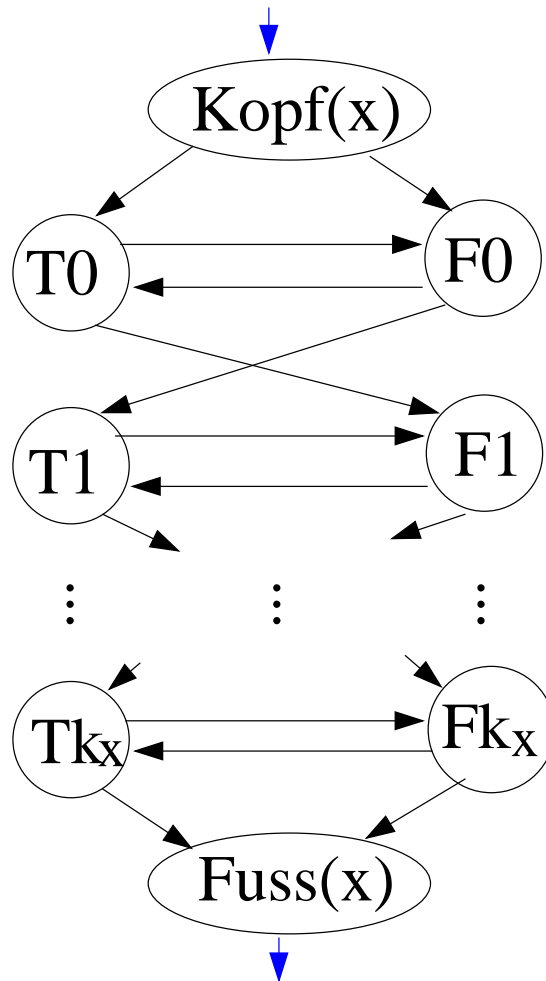
# Reduktion von 3SAT auf DHC

- $f$ : Eingabeformel für 3SAT mit den Variablen  $\{x_1, \dots, x_n\}$
1. Konstruiere für jede Variable  $x$  in  $f$  den Graphen  $G(x)$ :



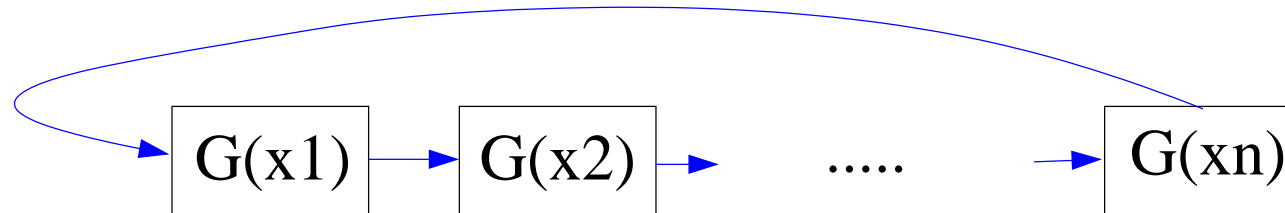
mit  $k_x = \max(\text{count}(x, f), \text{count}(\neg x, f))$

## Idee



Durchlaufe den Graphen auf einem Weg von Kopf bis Fuß. Besuche dabei jeden Knoten genau einmal. Gehe nach dem Kopf zuerst zu  $T_0$ , falls  $x$  mit  $1$  belegt ist; sonst gehe zuerst zu  $F_0$ .

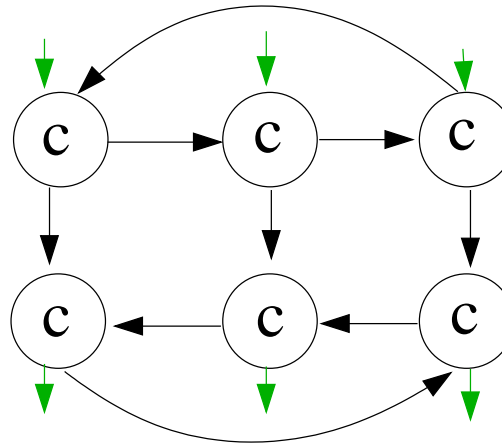
2. Verbinde diese Graphen miteinander zu  $G'$ :



### Beobachtung

Für jede Belegung der Variablen existiert ein **Hamiltonscher Kreis** in  $G'$  (d.h. ein Rundweg, der jeden Knoten genau einmal besucht).

3. Konstruiere für jede Klausel  $c$  in  $f$  den **Klausel-Graphen**  $G(c)$ :



### Beobachtung

Läuft man in den  $i$ -ten oberen Knoten hinein, muss man aus dem  $i$ -ten unteren Knoten wieder herauslaufen, wenn man jeden Knoten aus  $G(c)$  genau einmal besuchen will ( $i = 1, 2, 3$ ).

4. Konstruiere  $G(f)$ , indem jeder Graph  $G(c)$  mit dem Rest wie folgt verbunden wird:

$$\text{Sei } c = L_1 \vee L_2 \vee L_3.$$

Falls  $L_i = x$ :

(a) Wähle ein  $F_j$  aus  $G(x)$ , für das gilt:

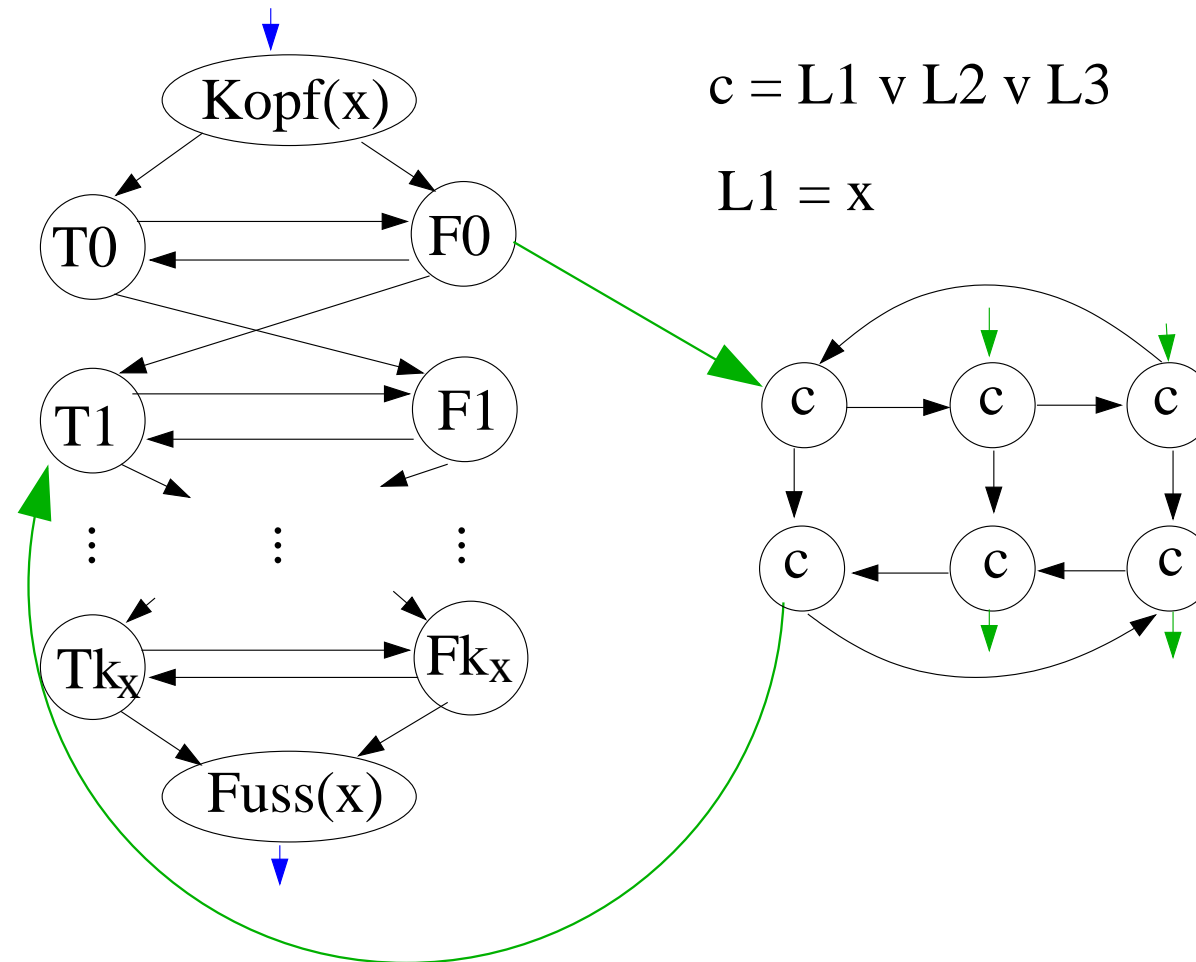
▷ Es gibt keine Kante zwischen  $F_j$  und einem Klausel-Graphen.

▷  $F_j$  ist nicht der unterste  $F$ -Knoten in  $G(x)$ .

(b) Ziehe eine Kante von  $F_j$  zu dem  $i$ -ten oberen Knoten in  $G(c)$  und eine Kante von dem  $i$ -ten unteren Knoten in  $G(c)$  zu  $T_{j+1}$  in  $G(x)$ .



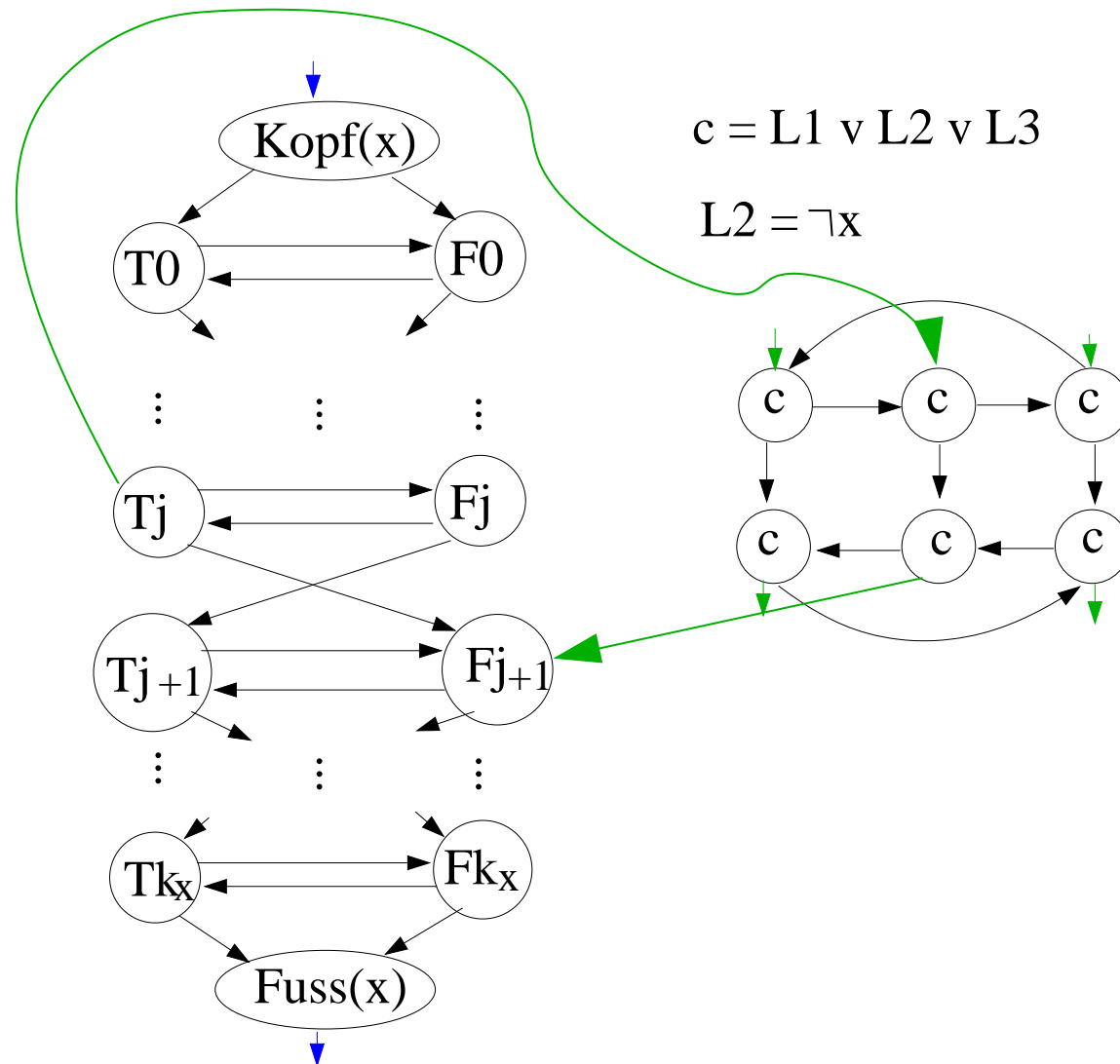
# Beispiel ( $L_1 = x$ )



Falls  $L_i = \neg x$ :

1. Wähle ein  $T_j$  aus  $G(x)$ , für das gilt:
  - Es gibt keine Kante zwischen  $T_j$  und einem Klausel-Graphen.
  - $T_j$  ist nicht der unterste  $T$ -Knoten in  $G(x)$ .
2. Ziehe eine Kante von  $T_j$  zu dem  $i$ -ten oberen Knoten in  $G(c)$  und eine Kante von dem  $i$ -ten unteren Knoten in  $G(c)$  zu  $F_{j+1}$  in  $G(x)$ .

# Beispiel ( $L_2 = \neg x$ )



## Beobachtung

Der Hamiltonsche Kreis in  $G'$  für eine Belegung der Variablen kann einen **Umweg** über  $G(c)$  machen gdw  $c \in T$  ist.

## Beobachtung

- DHC ist in  $NP$ .
- $f$  ist erfüllbar gdw es einen Hamiltonschen Weg durch  $G(f)$  gibt.
- Die Konstruktion von  $G(f)$  hat einen **polynomiellen Zeitaufwand**.

# Polynomieller Platzbedarf

**NPSPACE:**

Probleme mit nichtdeterministischen Lösungs-  
algorithmen, die polynomiellen Speicherplatz  
brauchen (im Verhältnis zur Größe der Eingabe)

**PSPACE:**

Analog für deterministische Lösungen

# Zusammenhänge zwischen Komplexitätsklassen

$$P \subseteq NP \subseteq_{(1)} NPSPACE \stackrel{(2)}{=} PSPACE$$

- (1) Konstanter Speicherplatz pro Rechenschritt
- (2) Backtracking

Offenes Problem:  $NPSPACE \subseteq NP$

# Wie weit reicht P und was kommt dahinter?



- $NP$ : Erfüllbarkeitsproblem, TSP u.v.a.m.
- $PSPACE$ : Wortproblem monotoner Sprachen
- $EXP$ : Drachenkurve, hoffnungslos für große Eingaben
- Berechenbares: Interpreter für CE-S u. ä.; Aufwand oft nicht definiert (**Berechnung unendlich**)
- Unberechenbares: Halteproblem u.v.a.m.

# Chomsky-Grammatiken

- ▶ Ursprünglich von Chomsky in den 1950er Jahren eingeführt zur Beschreibung natürlicher Sprachen.
- ▶ Enge Verwandtschaft zu Automaten
- ▶ Grundlage wichtiger Softwarekomponenten
- ▶ Enthalten außer den rechtslinearen und den kontextfreien weitere Grammatiktypen



# Chomsky-Grammatik (Typ 0)

$G = (N, T, P, S)$  mit

- $N$ : endl. Menge **nichtterminaler Zeichen**,
- $T$ : endl. Menge **terminaler Zeichen** mit  $N \cap T = \emptyset$ ,
- $P \subseteq (N \cup T)^* N (N \cup T)^* \times (N \cup T)^*$ : endliche Menge von **Produktionen**
- $S \in N$  : **Startsymbol**

► Schreibweise für Produktionen  $(u, v) \in P$ :  $u ::= v$

# Direkte Ableitung

$$w = xuy \xrightarrow[p]{} xvy = w'$$

mit  $w, w', x, y, u, v \in (N \cup T)^*$ ,  $p = (u ::= v)$ .

1. Suche  $u$  als Teilwort eines Wortes
2. Ersetze  $u$  durch  $v$

► **Schreibweise:**  $w \xrightarrow[P]{} w'$ ,

falls  $P$  eine Menge von Produktionen ist mit  $p \in P$ .

# Ableitung (Iteration direkter Ableitungen)

$$w_0 \xrightarrow{p_1} w_1 \xrightarrow{p_2} \cdots \xrightarrow{p_n} w_n$$

für  $w_0, \dots, w_n \in (N \cup T)^*$  und Produktionen  $p_1, \dots, p_n$   
 ( $n \geq 1$ )

## Schreibweisen:

- $w_0 \xrightarrow{P} \cdots \xrightarrow{P} w_n$  oder  $w_0 \xrightarrow[n]{P} w_n$  oder  $w_0 \xrightarrow{P}^* w_n$ ,  
 falls  $p_1, \dots, p_n \in P$ .
- $w \xrightarrow{*} w'$ , falls  $P$  aus dem Kontext klar ist.

# Nullableitung

$$w \xrightarrow[P]{0} w$$

für alle  $w \in (N \cup T)^*$ .

# Erzeugte Sprache

Sei  $G = (N, T, P, S)$  eine Chomsky-Grammatik.

Erzeugte Sprache

$$L(G) = \{w \in T^* \mid S \xrightarrow[P]{*} w\}$$