Grundlagen Theoretischer Informatik 3

SoSe 2010 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Grundlagen Theoretischer Informatik 3 Gesamtübersicht

- Organisatorisches; Einführung
- Algorithmenanalyse: Komplexität und Korrektheit
- Zur Behandlung NP-schwerer Probleme: Ausblicke
- Randomisierte Algorithmen und ihre Analyse

Organisatorisches

Vorlesungen FR 8.30-10.00 im HS 13

Übungsbetrieb in Form von einer Übungsgruppe
BEGINN: in der zweiten Semesterwoche
FR 10-12, HS 13

Dozentensprechstunde DO 13-14 in meinem Büro H 410 (4. Stock)

Mitarbeitersprechstunde (Stefan Gulan) DO 13-14 H 413

Tutorensprechstunde DI 13.30-14.30 & MI 14-15 H 407/H412

Datenstrukturen

Aus “Algorithmen und Datenstrukturen” bekannt:
Es gibt viele weitere “sinnvolle” Datentypen, abgesehen von Zahlen und Booleschen Werten.
Beispiele: Felder, Listen, Mengen, Graphen, . . .

Insbesondere mit Hilfe der indirekten Adressierung lassen sich diese Datentypen (sowie die zugehörigen Operationen) auf Registermaschinen abbilden.

Ohne auf Einzelheiten einzugehen, werden wir im Folgenden manchmal solche Datenstrukturen als “vorhanden” annehmen.

NP-Härte: Wiederholungen und Ergänzungen

NP: Klasse von Problemen (formal: Sprachklasse), die mit nichtdeterministischen Turing-Maschinen in Polynomzeit gelöst werden können.

Anstelle von Turing-Maschinen können auch WHILE-Programme als Sprachakzeptoren dienen:
Erinnerung: Wörter "sind" Zahlen (logarithmische Kosten!),
Akzeptanz bei "Ausgabe" von **true**.

Einfachere Vorstellung:
WHILE-Programme (wie bislang betrachtet) mit einer Zusatzfunktion **guess**,
die einen Booleschen Wert zurückliefert.
Der Aufruf dieser Zusatzfunktion benötigt lediglich einen Zeitschritt.
Ein Wort w (also eine Zahl) wird von so einem WHILE-Programm akzeptiert,
wenn es geeignete Rückgaben von **guess** gibt,
sodass das Programm hält und **true** zurückgibt.

Erfüllbarkeit Das NP-Problem ist SAT.

Erfüllbarkeitsproblem der Aussagenlogik

$SAT := \{code(F)$ aus $E^* \mid F$ erfüllbare Formel der Aussagenlogik $\}$

(Eine Formel F heiße *erfüllbar*, wenn es Belegung ϕ gibt mit $\phi(F) = 1$.)

Wichtige Spezialfälle:

Klausel: Disjunktion von Literalen; *konjunktive Normalform* (**KNF**): Konjunktion von Klauseln

$KNF - SAT$: SAT, eingeschränkt auf Formeln in KNF.

Ein BA in KNF heißt in (schwacher/starker) *3-SAT-NF*, falls jede Klausel höchstens/genau drei Literale umfasst.

$3 - SAT/3 - SAT'$: zugehörige Probleme

Satz von Cook: Wäre SAT ($3 - SAT/3 - SAT'$) in Polynomzeit (ohne **guess**) lösbar, so gälte das für alle Probleme in NP. Vermutung: $P \neq NP$.

Erfüllbarkeit

Warum liegt $KNF - SAT$ in NP?

WHILE-Programm-Skizze (Pseudo-Code):
Lies Formel F ein.
Extrahiere Anzahl n Boolescher Variablen und normiere durch Umbenennung Variablennamen $x[1], \ldots, x[n]$.
Speichere F in geeigneter interner Struktur, z.B. als Menge $\mathcal{F}$ von Mengen C von Literalen.
Ein Literal könnte als Zahl aus $\{\pm 1, \ldots, \pm n\}$ gespeichert werden.
For $i \leftarrow 1$ **to** n **do** $x[i] \leftarrow$ **guess od**
$res \leftarrow$ **true**; Ausgabevariable richtig für leere Formel
Foreach $C \in F$ **do**
    $res' \leftarrow$ **false**
    Foreach $l \in C$ **do**
        If $((l < 0) \wedge x[-l] =$ **false**$) \vee ((l > 0) \wedge x[l] =$ **true**$)$ **then** $res' \leftarrow$ **true fi od**
    $res \leftarrow res \wedge res'$
od
$output(res)$
Guess & Check Paradigma
Bsp.: $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \rightsquigarrow \mathcal{F} = \{\{1,2,3\},\{-1,-2,3\}\}$

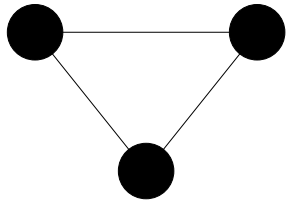
NP-Härte

SAT gehört zu den schwierigsten Problemen in NP.
Das heißt: Wenn SAT in P läge, so auch alle anderen Probleme aus NP.

Ein Problem A heißt NP-*hart*, wenn Folgendes gilt:
Kann man $A \in P$ zeigen, so lägen alle NP-Probleme in P.
Wie beweist man NP-Härte für ein Problem A?
Man zeigt: Wäre es möglich, A in Polynomzeit zu lösen, so wäre es auch möglich, ein bekanntermaßen NP-hartes Problem B in Polynomzeit zu lösen und mithin alle Probleme aus NP.
Man *reduziert* B auf A, i.Z.: $B \leq_P A$.

Formaler: Seien A und B Sprachen über einem Alphabet E.
Dann heißt B auf A *polynomial reduzierbar*,
wenn es eine in Polynomzeit berechenbare Funktion $f : E^* \to E^*$ gibt,
so dass für alle $w \in E^*$ gilt: $w \in B \iff f(w) \in A$. Schreibweise: $B \leq_p A$.

Sei $G = (V, E)$ ungerichteter Graph:
$V' \subseteq V$ ist *Knotenüberdeckung*, wenn jede Kante aus E mindestens einen Endpunkt in V' hat.



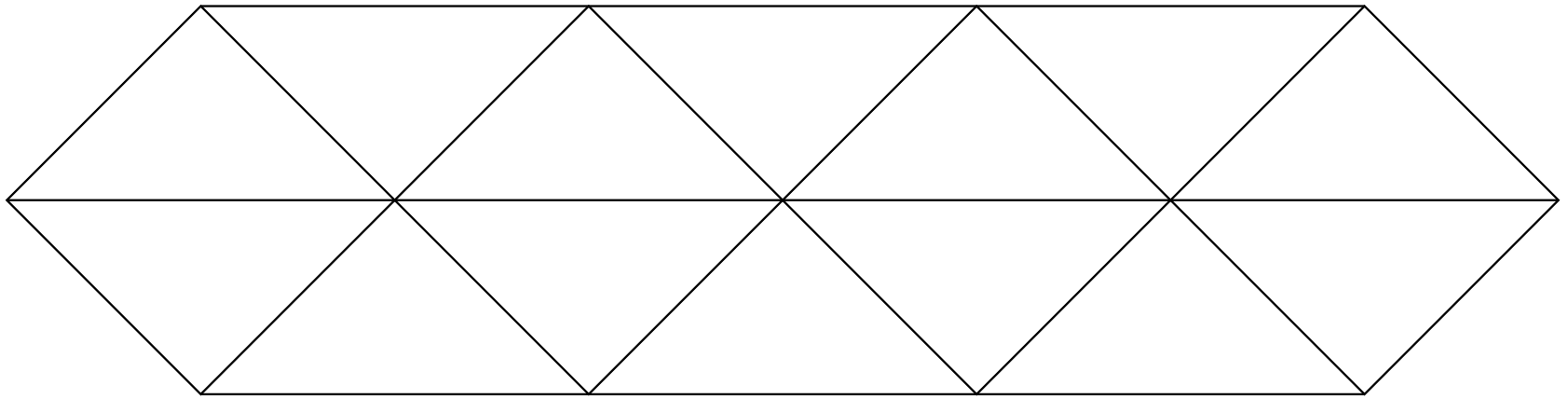
Hier sind zwei Knoten nötig!

$\Rightarrow$ VERTEX COVER als graphentheoretisches Überdeckungsproblem:
VERTEX COVER: Gegeben seien ein ungerichteter Graph $G = (V, E)$ und eine natürliche Zahl k.
Frage: Gibt es eine Menge $V' \subseteq V$ aus höchstens k Knoten, die eine Knotenüberdeckung bildet?

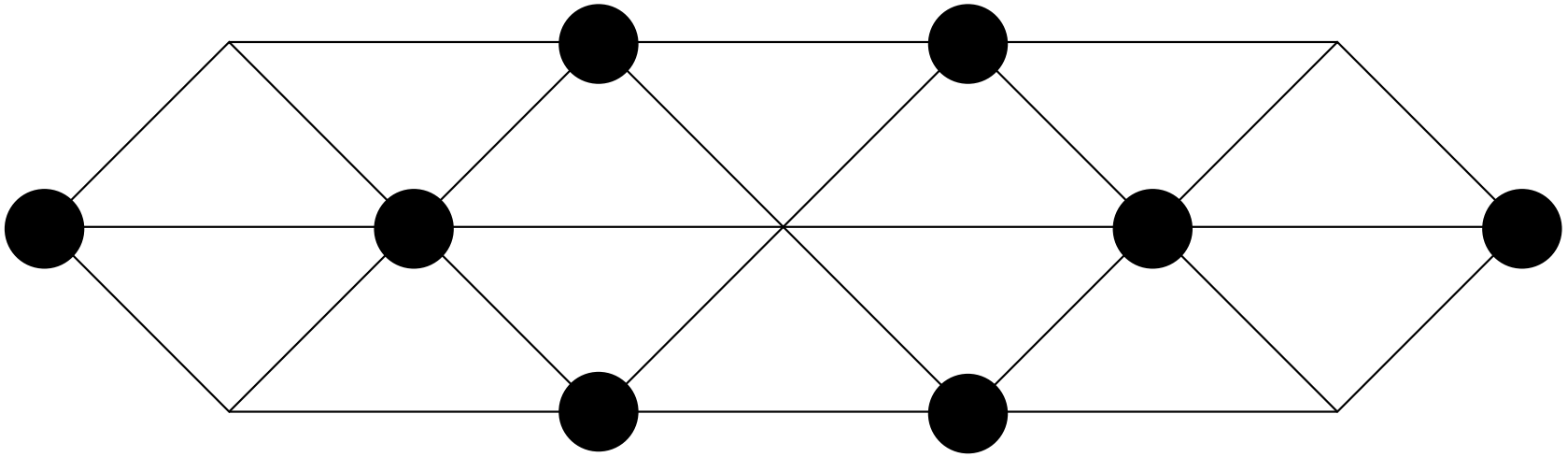
Satz: VERTEX COVER ist *NP*-hart.

Hinweis: Hierzu Graphen "geeignet" zu codieren!

Beispiel für die Begriffe: Betrachte folgenden Graphen:

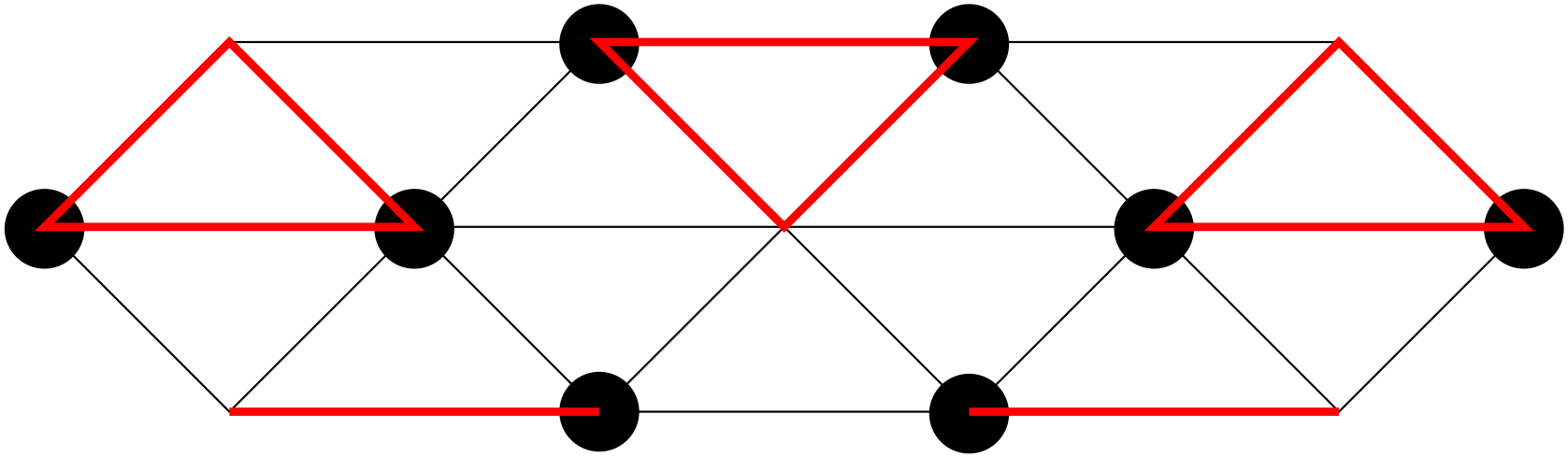


Beispiel: Lösung zu VERTEX COVER mit $k \geq 8$:



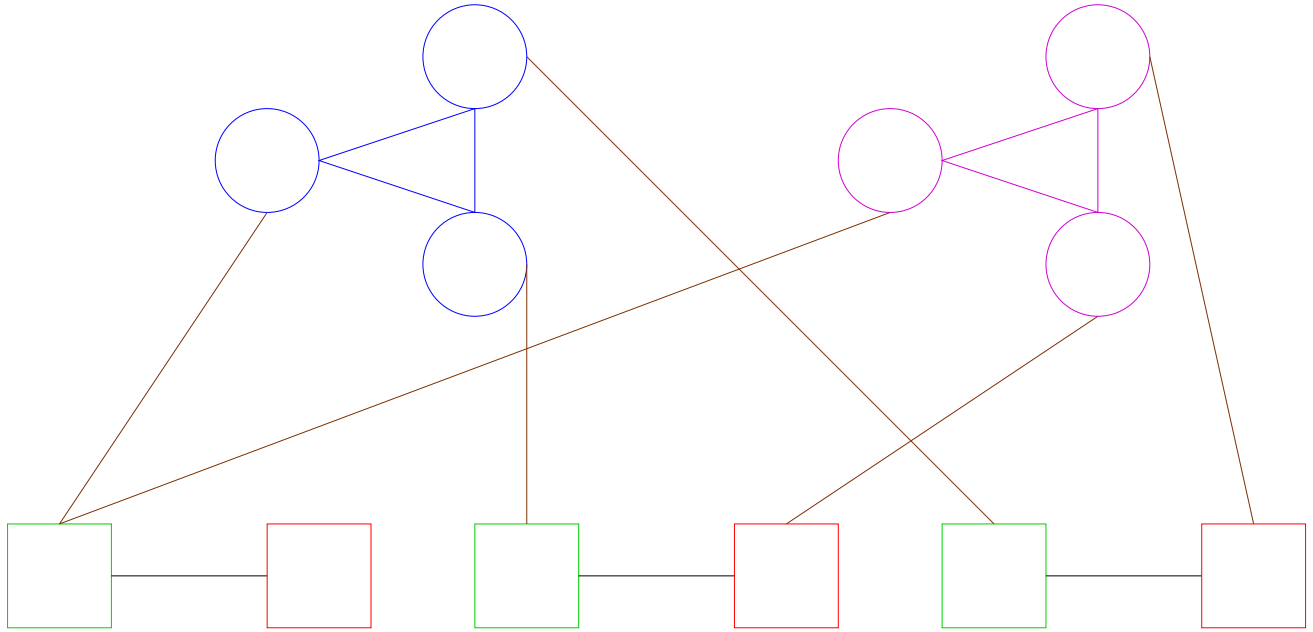
Warum ist Lösung kleinstmöglich? Betrachte Dreiecke und Kanten!

Beispiel: Lösung zu VERTEX COVER mit $k \geq 8$:

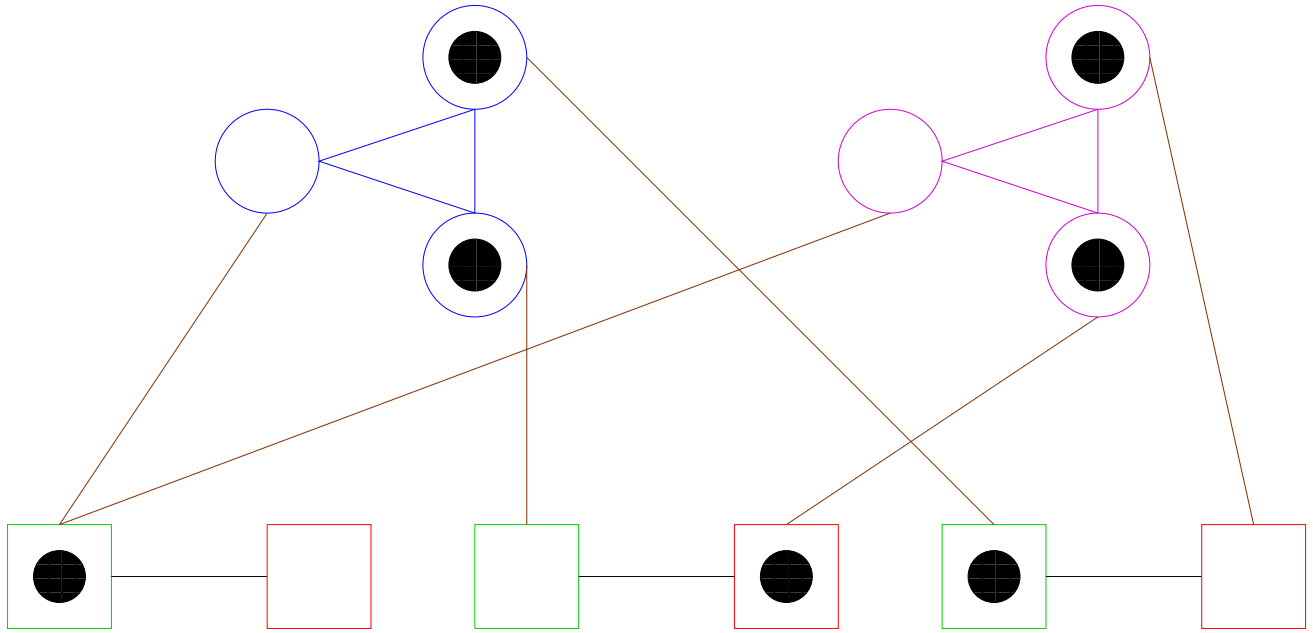


Warum ist Lösung kleinstmöglich? Betrachte Dreiecke und Kanten!

Grundidee: Codiere $3-SAT'$ (o.ä.) durch "Gadgets" für Variablen und Klauseln.
"Oben": $3m$ Knoten für die m Klauseln, "unten" $2n$ Knoten für die n Variablen.
Kanten zwischen Klauselknoten und Literalknoten kennzeichnen Vorkommen.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$. Allg.: Formel $w \mapsto G(w)$.



Konstruktion: $w \in 3-SAT'$ gdw.
$(G(w), 2m + n) \in$ VERTEX COVER.
Im Beispiel: $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee \bar{z})$.



Punkte markieren die Knotenüberdeckung.
Wegen der Dreiecke oben und Kanten unten folgt Minimalität.

Was wäre zu zeigen?

— Die skizzierte Transformation $w \mapsto G(w)$ arbeitet in Polynomzeit.

— Korrektheit der Reduktion.

Dazu im Einzelnen einige Überlegungen zu bel. Überdeckung C:

— Zum Abdecken eines Dreiecks benötigt man (mind.) zwei Knoten in C.
— Zum Abdecken einer Kante benötigt man (mind.) einen Knoten in C.
— Der Graph $G(w)$ enthält eine Kollektion von m Dreiecken (Klausel-Gadgets) und n Kanten (Variablen-Selektions-Gadget), welche, aufgefasst als Knotenmengen, paarweise disjunkt sind. Daher gilt: $|C| \geq 2m + n$.
— Eine Knotenüberdeckung C^* der Größe $2m + n$ von $G(w)$ enthält aus jedem der angesprochenen m Dreiecke $2m$ Knoten und von jeder Variablen-Selektions-Kante n weitere Knoten.
— Setzt man die Variablen auf "wahr", deren "grüne Knoten" in der Überdeckungsmenge C^* liegen, so erhält man eine gültige Variablenbelegung, die die Formel w wahr macht.
— Existiert keine Knotenüberdeckung der Größe $2m + n$ in $G(w)$, so gibt es auch keine erfüllende Belegung (durch Kontraposition).

Was tun, wenn (Alltags-)Problem *NP*-hart?

"Chef, das geht nicht. . . " liefert eben doch keine Job-Garantie. . .

Mögliche Idee: Vielleicht sind die "tatsächlich vorkommenden" Instanzen nicht von der Allgemeinheit, wie sie der Härte-Beweis fordert.

Beispiel SAT: Gibt es vielleicht nur zwei Literale pro Klausel?

Satz: Das folgende Problem liegt in P: **2KNF-Sat**

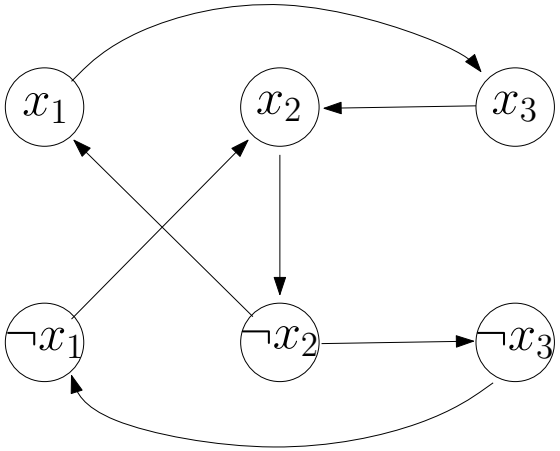
- *G*egeben: Eine Boolesche Formel F in konjunktiver Normalform mit höchstens 2 Literalen pro Klausel.
- *Gefragt:* Ist F erfüllbar?

Beweisidee

- Gegeben 2-KNF-Formel F mit Variablen $x_1, x_2, ..., x_n$.
- O.B.d.A.: zwei Literale pro Klausel (statt (a) verwende $(a \vee a)$).
- Klausel $(a \vee b)$ entspricht Implikation $(\neg a \rightarrow b)$ und $(\neg b \rightarrow a)$.
- Betrachte gerichteten Graphen $G_F = (V, E)$ mit Knotenmenge $V = \{x_1, ..., x_n, \neg x_1, ..., \neg x_n\}$ (*Literalgraph*).
- Zu E: Für Klausel $(a \vee b)$ verwende Kanten $(\neg a, b)$ und $(\neg b, a)$.
- Damit: Klausel mit einem Literal (a) ergibt Einzelkante $(\neg a, a)$.

Beispiel: Graph G_F zur folgenden Formel F:

$$F = \underbrace{(x_1 \vee x_2)}_{1} \wedge \underbrace{(\neg x_1 \vee x_3)}_{2} \wedge \underbrace{(x_2 \vee \neg x_3)}_{3} \wedge \underbrace{(\neg x_2)}_{4}$$



F ist nicht erfüllbar!

in Formel F: x_2 falsch $\overset{1}{\rightsquigarrow}$ x_1 wahr $\overset{2}{\rightsquigarrow}$ x_3 wahr $\overset{3}{\rightsquigarrow}$ x_2 wahr $\overset{4}{\rightsquigarrow}$ x_2 falsch

im Graphen: $\neg x_2 \longrightarrow x_1 \longrightarrow x_3 \longrightarrow x_2 \longrightarrow \neg x_2$

Also: Nutze Erreichbarkeit im Graphen G_F!

Durchführung der Idee

Betrachte transitive Hülle $G_F^* = (V, E^*)$ des Graphen G_F,
d.h. $(a, b) \in E^* \Leftrightarrow$ es gibt in E Pfad von a nach b.
E^* kann in polynomialer Zeit berechnet werden
(z.B. mit Warshall-Algorithmus, kubische Komplexität in n, s.u.).

Lemma: Eine 2-KNF-Formel F ist genau dann erfüllbar, wenn für kein i
in G_F ein Kreis der Form $x_i \to \cdots \to \neg x_i \to \cdots \to x_i$ existiert,
d.h. wenn in G_F^* nie beide Kanten $(x_i, \neg x_i)$ und $(\neg x_i, x_i)$ existieren.

Beweis des Lemmas: Beweis von '$\Longrightarrow$':
$(a \vee b)$ entspricht Kanten $(\neg a, b), (\neg b, a) \in E$
Bei einer erfüllenden Belegung von F gilt damit:

1. Ist Literal a wahr, sind alle Literale b wahr mit $(a, b) \in E^*$.

2. Ist Literal b falsch, sind alle Literale a falsch mit $(a, b) \in E^*$.

3. Ist $(\neg a, a) \in E^*$ für Literal a, so muss a wahr sein.

Damit direkt "F erfüllbar $\Rightarrow \neg\exists$ Kreis".

Beweis des Lemmas; Beweis von '$\Longleftarrow$':
Für die Rückrichtung definiere 'Belegung' (mit $\{0, 1\}$) wie folgt:
(1) Für alle Literale mit $(\neg a, a) \in E^*$ setze $a \mapsto 1$ (damit $\neg a \mapsto 0$).
(2) Dann ergänze für diese a:

- Setze alle b mit $(a, b) \in E^*$ auf 1.
- Setze alle b mit $(b, \neg a) \in E^*$ auf 0.

(3) Solange noch nicht alle Literale einen Wahrheitswert haben:

- Wähle beliebiges(!) noch nicht gesetztes Literal a, setze es auf 1.
- Setze wieder alle b mit $(a, b) \in E^*$ auf 1.
- Setze wieder alle b mit $(b, \neg a) \in E^*$ auf 0.

Gibt es keine Kanten $(x_i, \neg x_i)$ und $(\neg x_i, x_i)$ in E^*,
so ist die Belegung wohldefiniert und erfüllt F:
Nie folgt auf 1 im Graphen G eine 0, d.h. alle Klauseln sind erfüllt.

Das Knotenüberdeckungsproblem auf Hypergraphen (HITTING SET)

Ein *Hypergraph* $G = (V, E)$ ist gegeben durch eine endliche Menge V von Knoten und eine Menge E von *Hyperkanten*, d.h., Teilmengen von V.

Beobachtung: Jeder ungerichtete Graph ist ein Hypergraph.
Denn: Jede (ungerichtete) Kante ist eine zweielementige Knotenmenge.

Eine *Knotenüberdeckung* in einem Hypergraphen $G = (V, E)$ ist eine Knotenmenge C mit der Eigenschaft: $\forall e \in E : e \cap C \neq \emptyset$. Jede Kante wird also "getroffen" (Hitting Set).
Beim Knotenüberdeckungsproblem fragt man bei vorgelegtem Hypergraphen und Zahl k, ob es eine Knotenüberdeckung der Größe k gibt.

Lemma: HITTING SET ist NP-vollständig.
Beweis: Zur Härte siehe "Beobachtung"; Mitgliedschaft in NP "klar".

Das Mengenüberdeckungsproblem (SET COVER)

Eingabe: Eine endliche Grundmenge X und ein Mengensystem $\mathcal{S} \subseteq 2^X$ sowie eine Zahl k.
Frage: Gibt es eine *Mengenüberdeckung* genannte Teilmenge $\mathcal{C} \subseteq \mathcal{S}$ mit höchstens k Elementen, für die gilt: $\bigcup_{C \in \mathcal{C}} C = X$?

Satz: SET COVER ist NP-vollständig.
Beweis: Zur Härte siehe unten; Mitgliedschaft in NP "klar".
Wir zeigen: HITTING SET $\leq_p$ SET COVER.
Sei $G = (V, E)$ Hypergraph.
Assoziiere hierzu Mengensystem $\mathcal{S}$ über Grundmenge E:
Knoten v entspricht der Menge $S_v = \{e \in E \mid v \in e\}$.
Beobachte: $C \subseteq V$ ist Knotenüberdeckung von G gdw. $\mathcal{C} = \{S_v \mid v \in C\}$ ist Mengenüberdeckung von E.
$\forall e \in E : e \cap C \neq \emptyset$ gdw. $\forall e \in E \exists v \in C : v \in e$ gdw. $\forall e \in E \exists v \in C : e \in S_v$ gdw. $\forall e \in E \exists S_v \in \mathcal{C} : e \in S_v$ gdw. $\bigcup_{C \in \mathcal{C}} C = E$.
Die Begriffe Knoten und Hyperkante sind bei Hypergraphen "austauschbar".

Das Dominierungsproblem (DOMINATING SET)

Eine Knotenmenge $D \subseteq V$ eines Graphen $G = (V, E)$ heißt *dominierend*, wenn jeder Knoten entweder zu D gehört oder aber Nachbar eines Knotens von D ist: $\forall v \in V \exists d \in D : v = d \vee \{v, d\} \in E$.

Satz: DOMINATING SET ist NP-vollständig.
Beweis: Zur Härte siehe unten; Mitgliedschaft in NP "klar".
Wir zeigen: SET COVER $\leq_p$ DOMINATING SET.

Betrachte Mengensystem $\mathcal{S} \subseteq 2^X$, $X \neq \emptyset$.
Sei $V = X \cup \mathcal{S}$ Knotenmenge eines Graphen $G = (V, E)$.
Führe zwei Arten von Kanten ein, sodass stets $\{S_1, S_2\} \in E$ gilt für alle $S_1, S_2 \in \mathcal{S}$ sowie $\{x, S\}$ gilt für $x \in X$ und $S \in \mathcal{S}$, sofern $x \in S$.
Ist $\mathcal{C}$ eine Mengenüberdeckung von $(X, \mathcal{S})$, so gilt wegen $X \neq \emptyset$ auch $\mathcal{C} \neq \emptyset$.
Daher ist $\mathcal{C}$ eine dominierende Menge in G.
Ist $D \subseteq V$ dominierend, so gibt es $D' \subseteq \mathcal{S}$, $|D'| \leq |D|$, die ebenfalls dominierend ist.
D' ist dann Mengenüberdeckung von $(X, \mathcal{S})$.

Bäume sind leicht...

Was ist, wenn wir überdeckende oder dominierende Knotenmengen nur für ganz spezielle Graphen ausrechnen müssen, z.B. für Bäume?
Erinnerung: Ein Graph $G = (V, E)$ heißt (ungerichteter, unverwurzelter) *Baum* wenn es zwischen zwei Knoten u, v genau einen *Weg* gibt, d.h., eine Knotenfolge $x_0 = u, x_1, \ldots, x_r = v$, $r \geq 0$, mit $\{x_{i-1}, x_i\} \in E$ für alle $i = 1, \ldots, r$.
Grundidee:
Wähle beliebige Wurzel r; assoziiere Zustände und Teilbäume zu den Knoten. Berechne dann von den Blättern zur Wurzel hin die bestmögliche Lösungsgröße in den Teilbäumen für jeden Zustand und notiere diese Größe.
Sie kann (im Sinne des dynamischen Programmierens) für die Berechnung im Vaterknoten benutzt werden.

Zustände für Knotenüberdeckungsproblem: $0, 1$;
für Dominierungsproblem: $(0, ☺), (0, ☹), 1$.
Mehr an der Tafel!