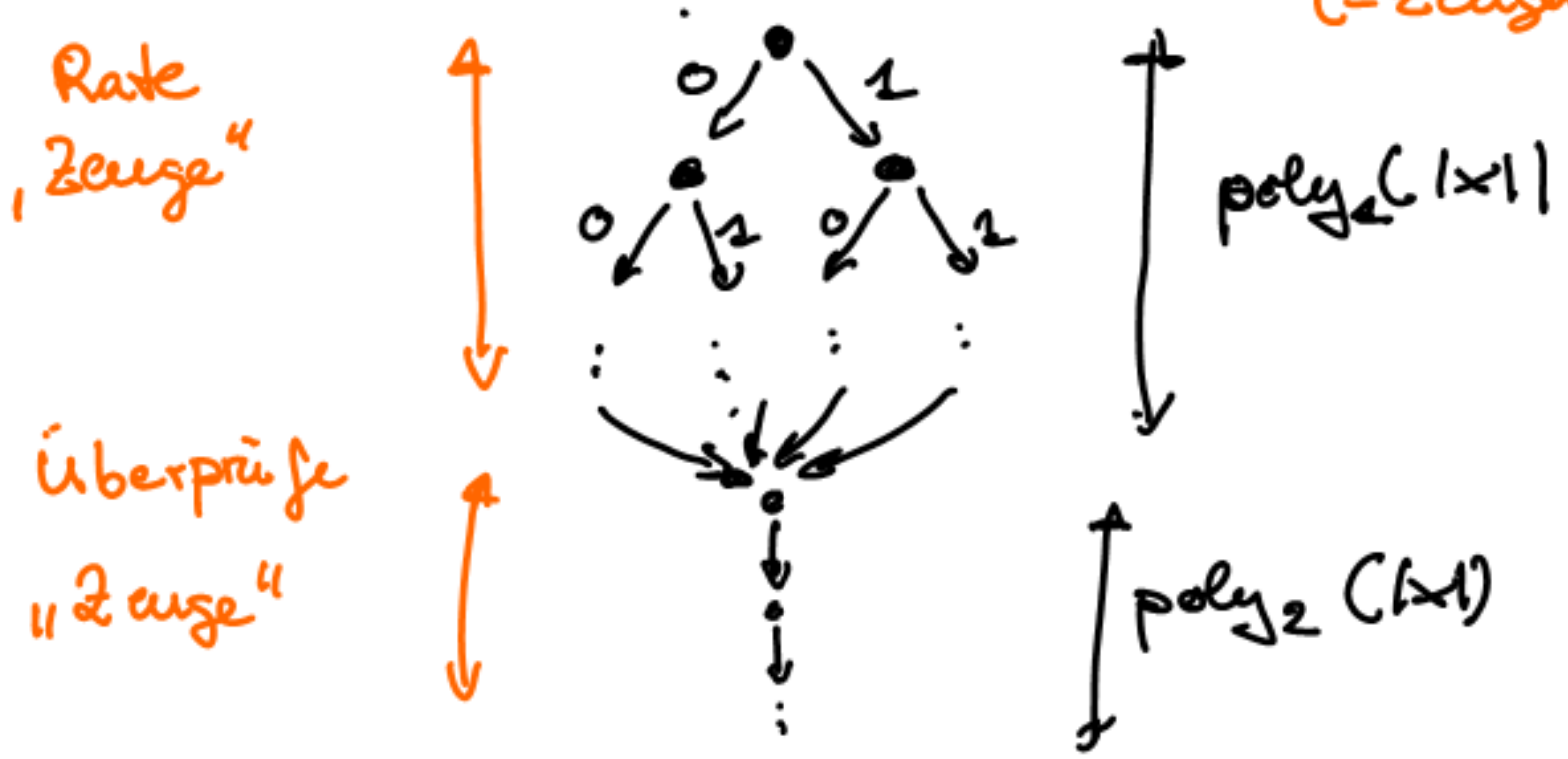
NP ≙ Menge aller Entscheidungsprobleme, die durch NTM in Polynomialzeit $\bigcup_{k \in \mathbb{N}} O(n^k)$ entschieden werden können

≙ Menge aller Entscheidungsprobleme mit in Polynomialzeit überprüfbaren Zertifikaten (= Zeugen)



„Die Mutter aller NP-Probleme" SAT

Gegeben: Aussagenlogische Formel F

$F ::= V \mid \neg F \mid (F \vee F) \mid (F \wedge F)$

$V ::= A \mid V'$ mit $A_0 := A$, $A_1 := A'$, $A_2 := A''$, ...

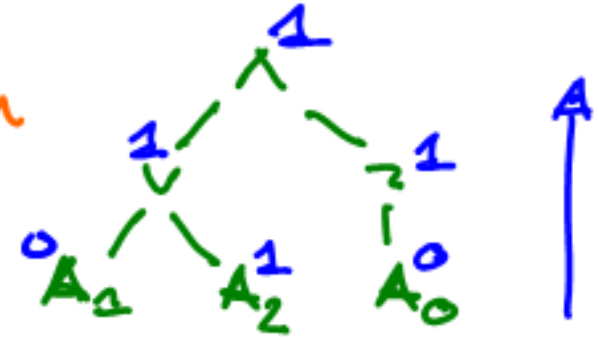
oder $x := A$, $y := A'$, $z := A''$, ...

Frage: Ist F erfüllbar?

Zeuge: erfüllende Belegung σ $\rightsquigarrow$ maximal $|F|$ Bits
$\rightsquigarrow$ polynomiell in $|F|$

Überprüfen: Einsetzen und Auswerten
$\rightsquigarrow$ bottom-up Syntaxbaum auswerten
$\rightsquigarrow$ in $\mathcal{O}(|F|)$

Syntaxbaum mittels CYK in PTIME →



$\sigma(A_0) = 0, \sigma(A_1) = 0, \sigma(A_2) = 1$

6.3 NP-Vollständigkeit

1. Polynomielle Reduzierbarkeit $\leq_p$
2. NP-vollständige Probleme = härteste Probleme in NP,
   alle anderen Probleme in NP darauf polynomiell reduzierbar
3. Satz: SAT ist NP-vollständig

Definition 6.10

Sei $A \subseteq \Sigma^*$ und $B \subseteq \Gamma^*$.
Dann heißt A polynomiell reduzierbar auf B, $A \leq_p B$,
gdw es eine totale und von einer DTM in polynomieller Zeit berechenbare Funktion $f : \Sigma^* \to \Gamma^*$ gibt, so dass für alle $w \in \Sigma^*$

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

Da $q(p(n))$ ein Polynom ist falls p und q Polynome sind:

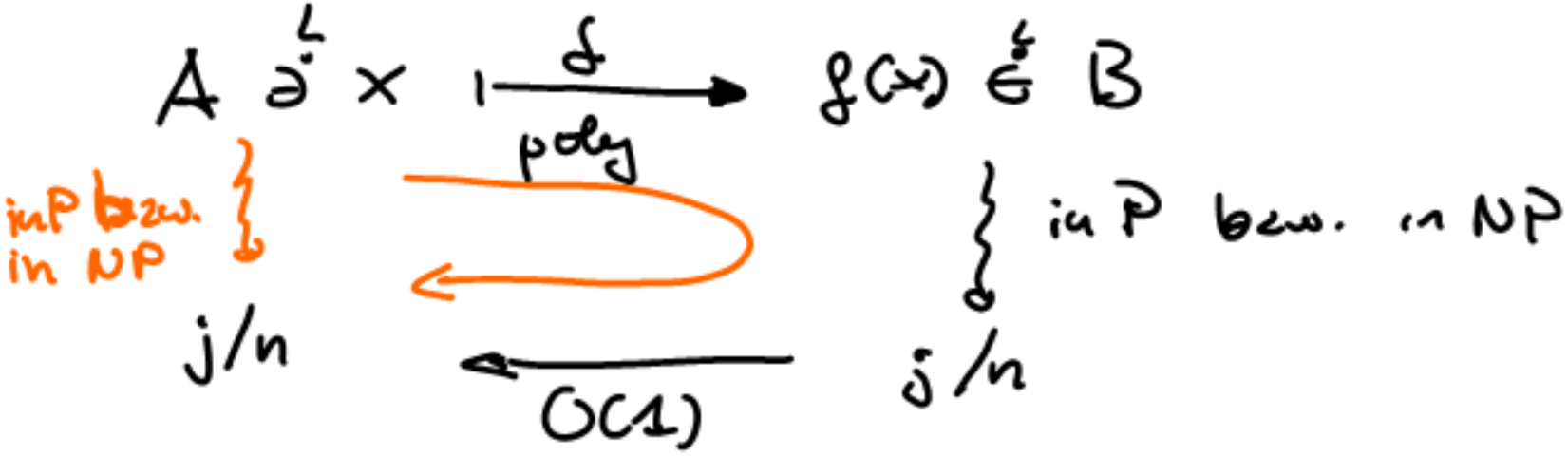
Lemma 6.11

Die Relation $\leq_p$ ist transitiv.

Lemma 6.12

Die Klassen P und NP sind unter polynomieller Reduzierbarkeit nach unten abgeschlossen:

$$A \leq_p B \in P/NP \implies A \in P/NP$$



da: $A \ni x$ gdw. $f(x) \in B$

Lemma 6.12

Die Klassen P und NP sind unter polynomieller Reduzierbarkeit nach unten abgeschlossen:

$$A \leq_p B \in P/NP \implies A \in P/NP$$

Beweis:

- Sei $A \leq_p B$ mittels f, die von DTM M_f berechnet wird.
  Polynom p begrenzt Rechenzeit von M_f.
- Sei $B \in$ P mittels DTM M.
  Polynom q begrenzt Rechenzeit von M.

Damit ist $M_f; M$ polynomiell zeitbeschränkt in $|w|$:

- $M_f[w]$ macht $\leq p(|w|)$ Schritte.
- Ausgabe $f(w)$ von M_f hat Länge $\leq |w| + p(|w|)$.
- M macht $\leq q(|f(w)|) \leq q(|w|+p(|w|))$ Schritte (q monoton)

Daher macht $(M_f; M)[w]$ maximal $p(|w|) + q(|w| + p(|w|))$ Schritte, ein Polynom in $|w|$.

Analog: $A \leq_p B \in$ NP $\implies A \in$ NP

Englisch: NP-hard
Deutsch: NP-schwer

Ein Problem ist NP-hart,
wenn es mindestens so hart wie alles in NP ist:

Definition 6.13

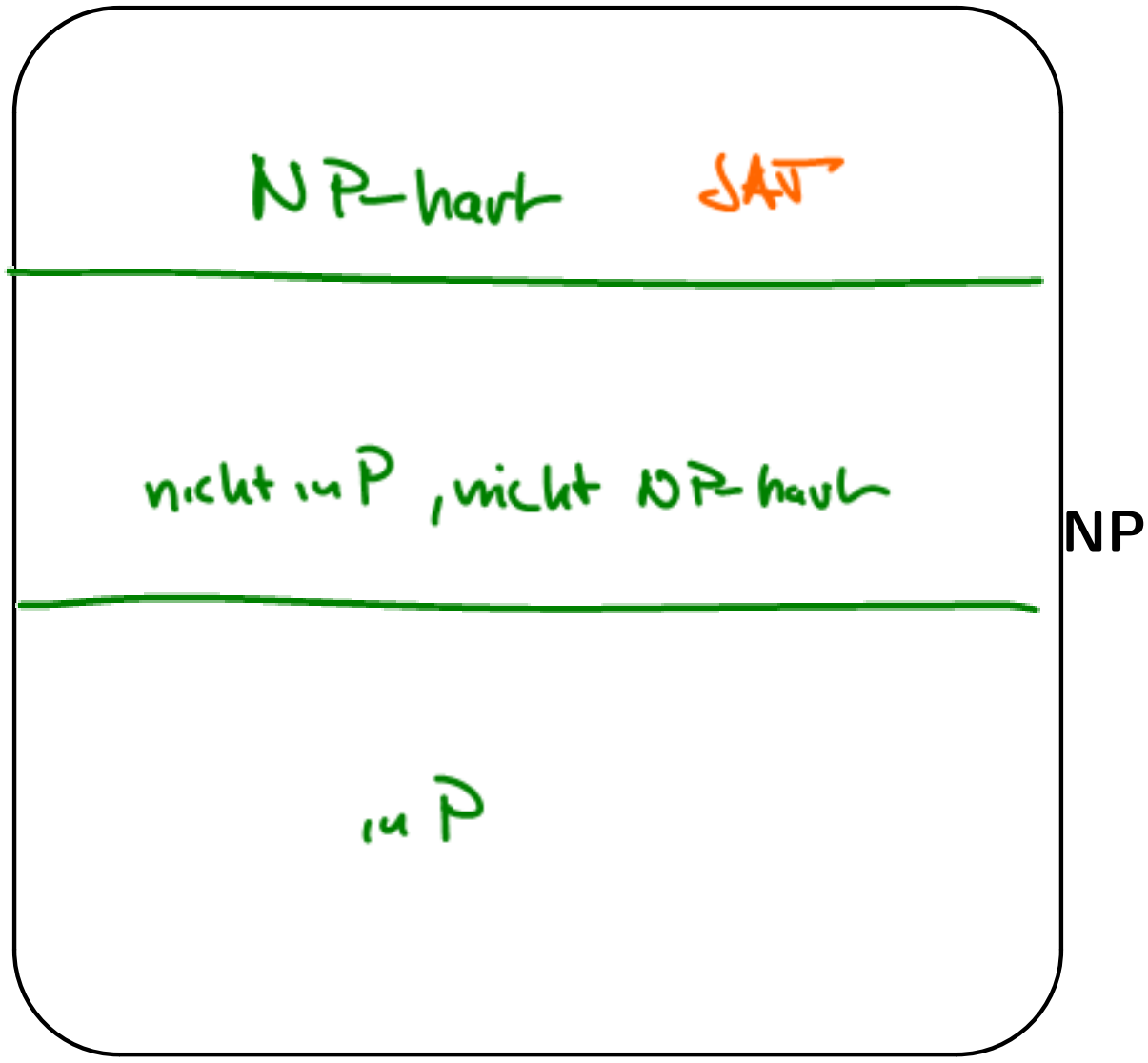
Eine Sprache L heißt NP-hart gdw $A \leq_p L$ für alle $A \in$ NP.

Definition 6.14

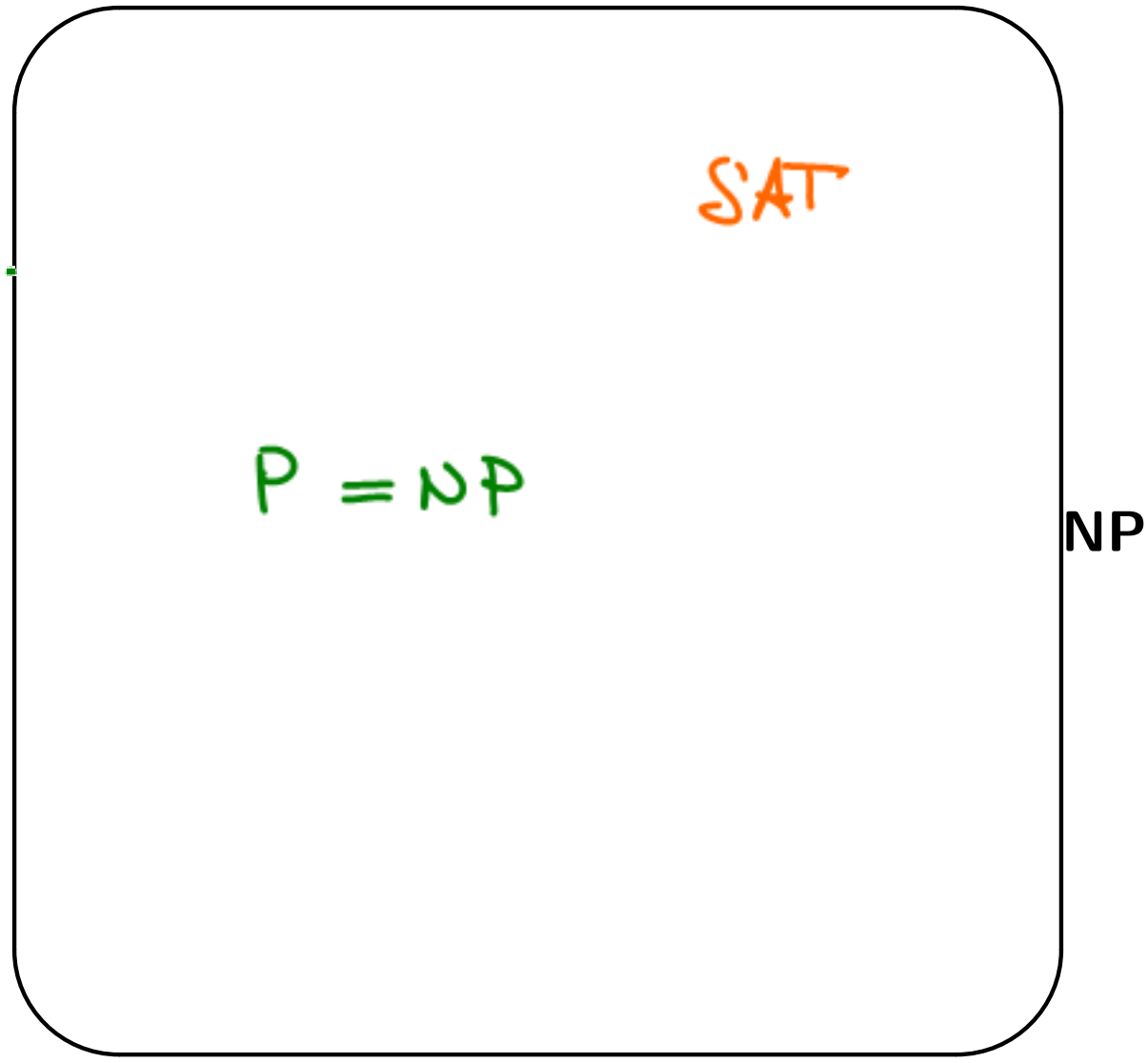
Eine Sprache L heißt NP-vollständig gdw
L NP-hart ist und $L \in$ NP.

Intuition: NP-vollständige Probleme sind die schwierigsten Probleme in NP:
alle Probleme in NP sind polynomiell auf sie reduzierbar.

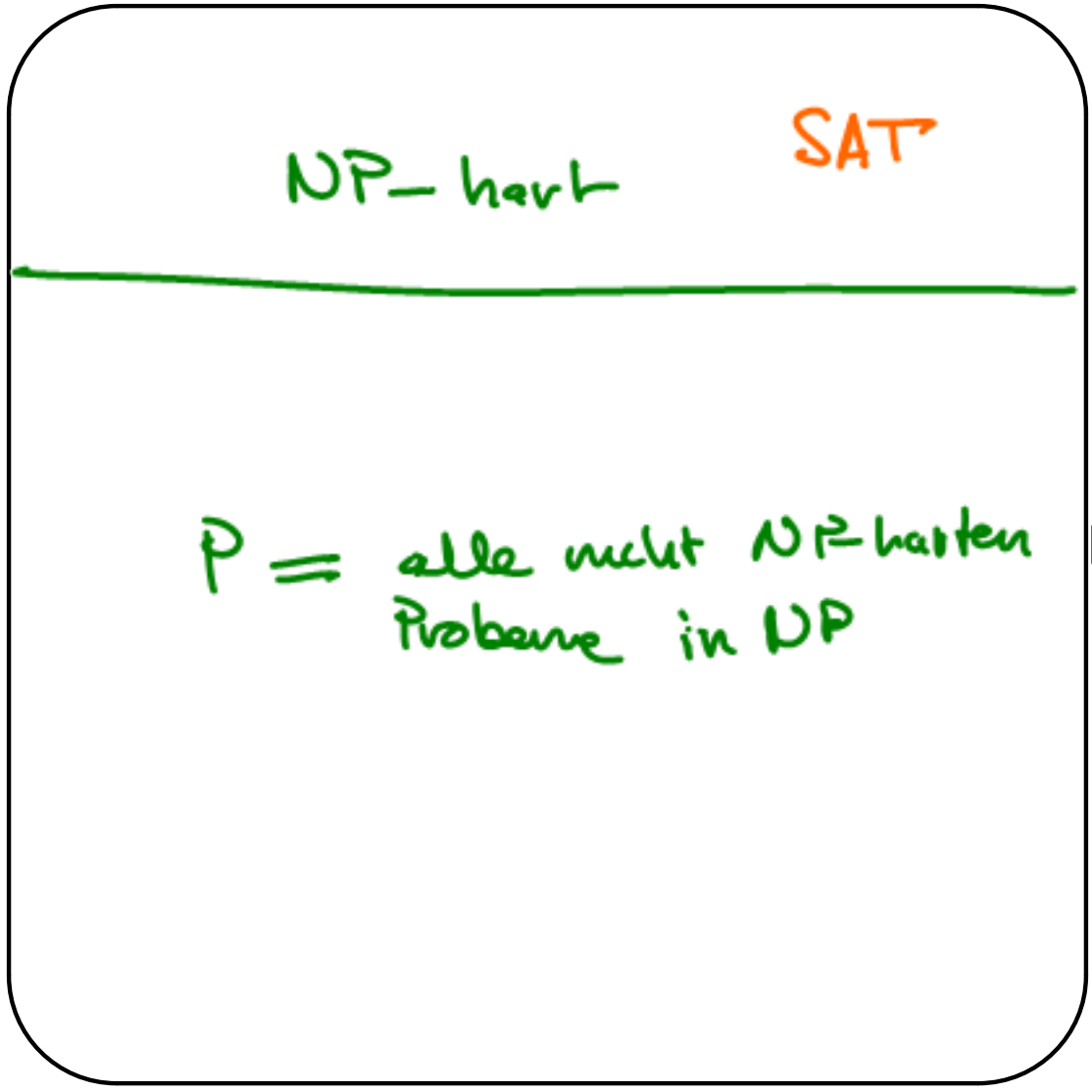
Mögliche Szenarien für $P \subseteq NP \supseteq$ NP-vollständig



Mögliche Szenarien für $P \subseteq NP \supseteq$ NP-vollständig



NICHT möglich ~ siehe z.B. Papadimitriou



Wie man $P \stackrel{?}{=} NP$ lösen kann:

nicht: NP-hart

Lemma 6.15

Es gilt P=NP gdw ein NP-vollständiges Problem in P liegt.

„⇐" $A \in NP$, $B \in P$ mit $\forall C \in NP : C \leq_p B$

also

„⇒" trivial

Wie man P$\stackrel{?}{=}$NP lösen kann:

Lemma 6.15

Es gilt P=NP gdw ein NP-vollständiges Problem in P liegt.

Beweis:

„$\Rightarrow$“: Falls P=NP, so liegt jedes NP-vollständige Problem in P.

„$\Leftarrow$“: Sei L ein NP-vollständiges Problem in P. Dann gilt P $\supseteq$ NP: Ist $A \in$ NP, so gilt $A \leq_p L$ (da L NP-hart) und nach Lemma 6.12 $A \in$ P (da $L \in$ P). $\square$

Starke Vermutung:

- P $\neq$ NP
- dh kein NP-vollständiges Problem ist in P.

Aber gibt es überhaupt NP-vollständige Probleme?

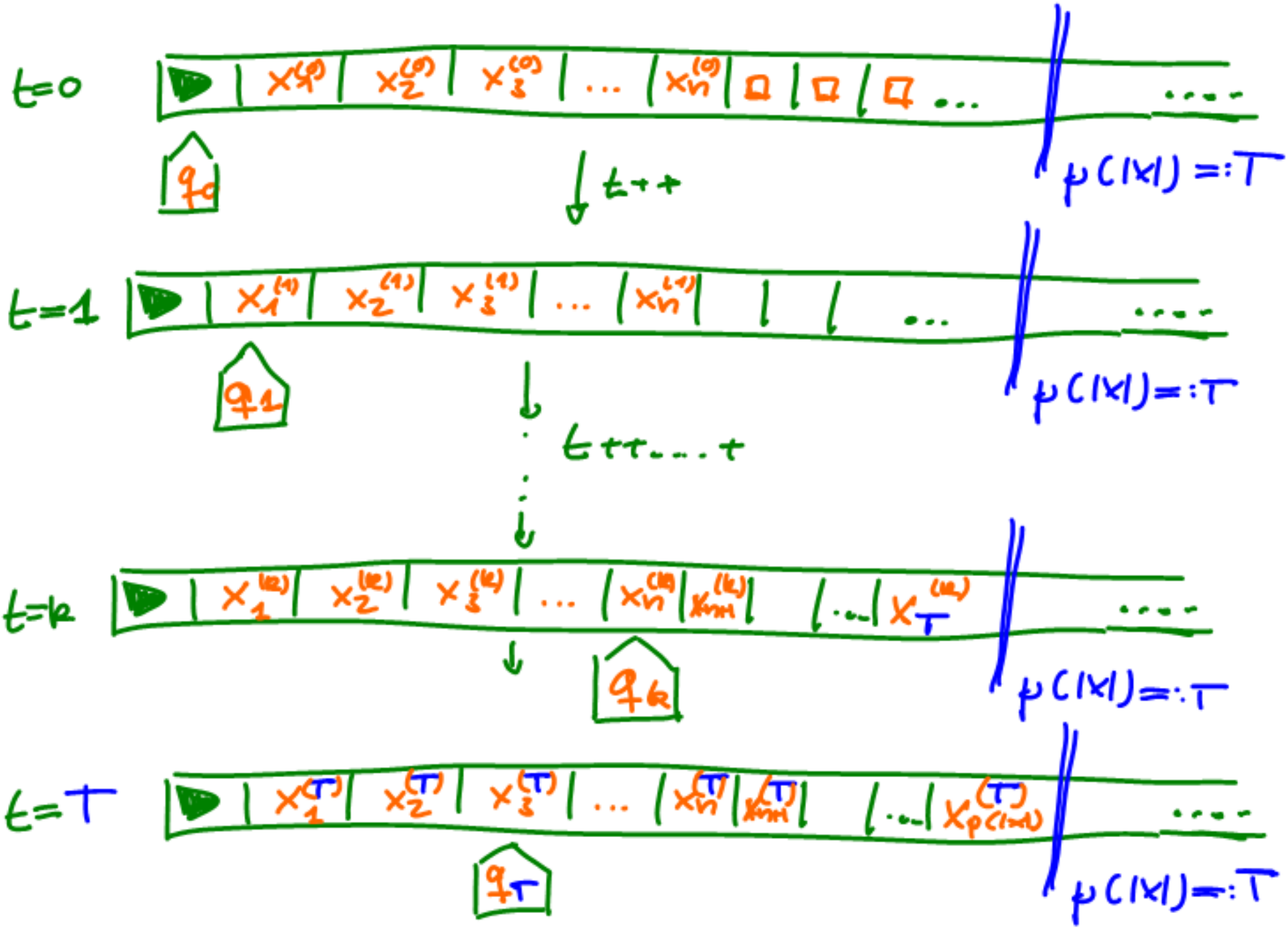
OBdA: 1 Band, nur nach rechts offen

Satz 6.17 (Cook 1971)

SAT ist NP-vollständig.

- Gegeben $A \in NP$ zeige NTM M, Polynom p so dass $x \in A$ gdw. $M[x] = 1$ und alle Abläufe von $M[x]$ bestehen aus genau $p(|x|)$ Schritten
- also kann sich Kopf von M um maximal $p(|x|)$ Positionen bewegen

- Idee: Beschreibe alle möglichen Abläufe $M[x]$ mittels aussagenlogischer Formel der Länge poly $(p(|x|))$


t=0
$x_1^{(0)}$ $x_2^{(0)}$ $x_3^{(0)}$ ... $x_n^{(0)}$
q_0
t++
$p(|x|) =: T$
t=1
$x_1^{(1)}$ $x_2^{(1)}$ $x_3^{(1)}$ ... $x_n^{(1)}$
q_1
$p(|x|) =: T$
t++...+
t=k
$x_1^{(k)}$ $x_2^{(k)}$ $x_3^{(k)}$ ... $x_n^{(k)}$ $x_{n+1}^{(k)}$ ... $x_T^{(k)}$
q_k
$p(|x|) =: T$
t=T
$x_1^{(T)}$ $x_2^{(T)}$ $x_3^{(T)}$ ... $x_n^{(T)}$ $x_{n+1}^{(T)}$ ... $x_{p(|x|)}^{(T)}$
q_T
$p(|x|) =: T$

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf 1 zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i, j, t \in \{0, \dots, p(|x|)\}$, $q \in Q$, $a \in \Gamma$

Beschreibe

① Anfangskonfiguration : F_A

② akzeptierende Endkonfiguration : F_E

③ korrekten Übergang von t nach $t+1$: F_U

④ Randbedingung: Immer genau eine Möglichkeit F_R

z.B. $\neg(P_{0,0} \wedge P_{0,1})$ muss erfüllt sein

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i, j, t \in \{0, \dots, p(|x|)\}$, $q \in Q$, $a \in \Gamma$

Hilfsformel: $G(X)$:= "genau eine Variable aus X mit 1 belegt"

$$G(X) = \left(\bigvee_{x \in X} x \right) \wedge \bigwedge_{\substack{x,y \in X \\ x \neq y}} \neg(x \wedge y)$$

$|G(X)| \in \mathcal{O}(|X| + |X|^2) = \mathcal{O}(|X|^2)$

↑ mindestens eine Variable wahr

↑ nie zwei Variablen wahr

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i, j, t \in \{0, \dots, p(|x|)\}$, $q \in Q$, $a \in T$

(6) Randbedingung Zu jedem Zeitpunkt genau ein Zustand, genau eine Position, genau ein Zeichen je Position

$$F_k = \bigwedge_{t=0}^{p(|x|)} \Big(G(\{Z_{t,q} \mid q \in Q\}) \wedge G(\{P_{t,i} \mid i = 0 \dots p(|x|)\}) \wedge \bigwedge_{j=0}^{p(|x|)} G(\{B_{t,j,a} \mid a \in T\}) \Big)$$

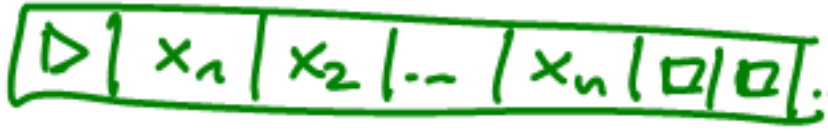
$|F_k| \in O(p(|x|) \cdot (|Q|^2 + p(|x|)^2 + |T|^2))$

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i, j, t \in \{0, \dots, p(|x|)\}$, $q \in Q$, $a \in \Gamma$

① Anfangskonfiguration



$$F_A = Z_{0,q_0} \wedge P_{0,0} \wedge B_{0,0,\triangleright}$$

(t=0) $\wedge \bigwedge_{i=1}^{|x|} B_{0,i,x_i}$ $\quad |F_A| = \mathcal{O}(p(|x|))$

$\wedge \bigwedge_{i,\ |x|+1}^{p(|x|)} B_{0,i,\square}$

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i, j, t \in \{0, \dots, p(|x|)\}$, $q \in Q$, $a \in \Gamma$

(2) akzeptierende Endkonfiguration

$$F_E = \bigvee_{q \in Q_F} Z_{p(|x|),q}$$

Variablen

- $Z_{t,q}$:= "Zustand zum Zeitpunkt t ist q"
- $P_{t,i}$:= "Position von Kopf zum Zeitp. t ist i"
- $B_{t,j,a}$:= "Bandinhalt zum Zeitp. t an Position j ist a."

$i,j,t \in \{0,\dots,p(|x|)\}$, $q \in Q$, $a \in \Gamma$

switch (..) {
case: .... break;
}

③ korrekter Übergang:

$$F_U = \bigwedge_{\substack{t,q,i,a \\ t<p(|x|) \\ q \neq \bar{q} \\ i=0 \dots p(|x|) \\ a \in \Gamma}} \Big(Z_{t,q} \wedge P_{t,i} \wedge B_{t,i,a} \longrightarrow \quad \Big)$$

$$\wedge \bigwedge_{t,j} \Big(\neg P_{t,j} \longrightarrow \bigwedge_{a \in \Gamma} \big(B_{t,j,a} \longrightarrow B_{t+1,j,a} \big) \Big)$$

Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT "reduziert" werden

von Entscheidungsproblem : F erfüllbar? (SAT)

zu Berechnungsproblem : Bestimme erfüllende Belegung

~> Idee SAT als „Orakel", ob erfüllende Belegung im Fall $x=0$ bzw. $x=1$ existiert.

#Var $\leq |F|$ also $\mathcal{O}(|F|)$ SAT-Aufrufe

- Falls $F[x:=0]$ erfüllbar, setze $\sigma(x):=0$ und berechne erfüllende Belegung an $F[x:=0]$, ansonsten $\sigma(x):=1$ und analog für $F[x:=1]$

(„Selbstreduktion von SAT")

Von der Lösbarkeit zur Lösung

Die Berechnung einer erfüllenden Belegung kann auf SAT "reduziert" werden

Sei F eine Formel mit den Variablen $x_1, \ldots, x_k$:

if $F \notin$ SAT **then** output("nicht lösbar")
else
  for $i := 1$ **to** k **do**
    if $F[x_i := 0] \in$ SAT **then** $b := 0$ **else** $b := 1$
    output(x_i "=" b)
    $F := F[x_i := b]$

wobei $F[x := b] = F$ mit x ersetzt durch b.

Entscheidung von SAT in Zeit $O(f(n))$
$\Longrightarrow$ Berechnung einer erf. Bel. in Zeit $O(n \cdot (f(n) + n))$
(falls es eine gibt.)

$f(n)$ polynomiell $\Longrightarrow$ $n \cdot (f(n) + n)$ polynomiell
$f(n)$ exponentiell $\Longrightarrow$ $n \cdot (f(n) + n)$ exponentiell

Von NP-hart zu „NP-leicht“

- Bis vor ca. 15 Jahren:

  NP-vollständig = Todesurteil

- In den letzten 15 Jahren:
  Spektakuläre Fortschritte bei *Implementierung* von
  SAT-Lösern (*SAT-solver*): `http://satcompetition.org`
  Stand der Kunst: Erfüllbar: bis 10^5 Variablen
  Unerfüllbar: bis 10^3 Variablen

- Jetzt:

  NP-vollständig = Hoffnung durch SAT

- Paradigma:

  SAT (Logik!) als universelle Sprache
  zur Kodierung kombinatorischer Probleme

  Reduktion auf SAT manchmal schneller als problemspezifische Löser! Und fast immer einfacher.

Beispiel: Reduktion von 3-Färbbarkeit (3COL) auf SAT.

3COL

Gegeben: Ein ungerichteter Graph

Problem: Gibt es eine Färbung der Knoten, so dass keine benachbarten Knoten gleich gefärbt sind?

Lineare Reduktion von Graph $G = (V, E)$ auf SAT:

- Variablen $= V \times \{1, 2, 3\}$. Notatation: x_{vi}
- Interpretation: $x_{vi} = 1$ gdw Knoten v hat Farbe i

Formel:

genau ein Farbe pro Knoten

$$\bigwedge_{v \in V} G(x_{v1}, x_{v2}, x_{v3}) \wedge \bigwedge_{(u,v) \in E} \neg(x_{u1} \wedge x_{v1} \vee x_{u2} \wedge x_{v2} \vee x_{u3} \wedge x_{v3})$$

benachbarte Knoten haben verschiedene Farben

Bemerkungen

- Zeigt 3COL $\leq_p$ SAT und damit 3COL $\in$ NP.
- Zeigt nicht, dass 3COL NP-vollständig ist.

SAT auf 3COL mittels 3SAT.

3SAT kurz für 3-KNF-SAT

kurz für: entscheide, ob Formel in 3-KNF erfüllbar

3-KNF: $F = \bigwedge_{i=1}^{n} \underbrace{(L_{i,1} \vee L_{i,2} \vee L_{i,3})}_{\text{Klausel}}$

mit $L_{i,j} \in \{A, \neg A \mid A \text{ Variable}\}$ Literal

bel. Formel $\rightsquigarrow$ 3-KNF

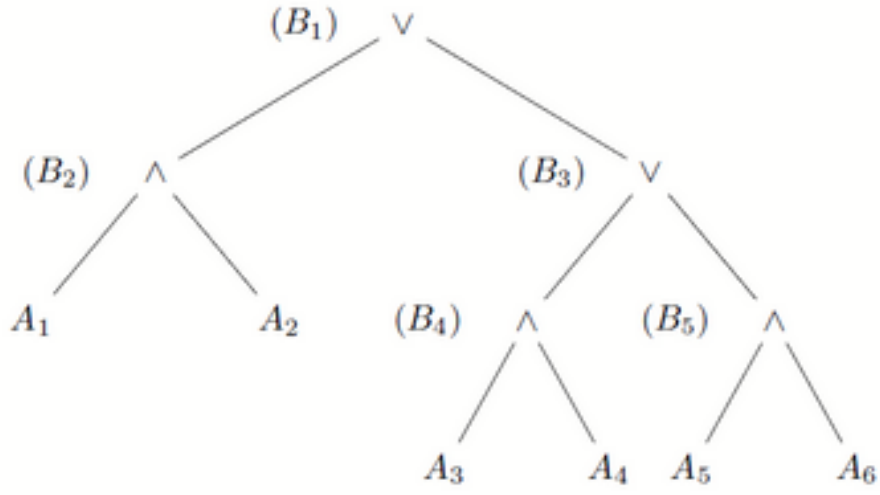
Erinnerung:

DS und nächstes Mal

$F = \begin{array}{c} \text{op} \\ G \quad H \end{array} \equiv_e (B_F \leftrightarrow (B_G \text{ op } B_H)) \wedge (B_G \leftrightarrow G) \wedge (B_H \leftrightarrow H)$

$\rightsquigarrow B_F$ Hilfsvariable, speichert Wert von Teilformel F

siehe DS TA 5.2 (2015)



Für F_3 erhält man so:

$$G_3 = B_1 \wedge (B_1 \leftrightarrow (B_2 \vee B_3)) \wedge (B_2 \leftrightarrow (A_1 \wedge A_2)) \wedge (B_3 \leftrightarrow (B_4 \vee B_5)) \wedge (B_4 \leftrightarrow (A_3 \wedge A_4)) \wedge (B_5 \leftrightarrow (A_5 \wedge A_6))$$

Es lässt sich zeigen, dass F_3 genau dann erfüllbar ist, wenn G_3 erfüllbar ist (kurz: F und G sind *erfüllbarkeitsäquivalent*):

- Ist β eine minimale, F_3 erfüllende Belegung, so kann man β zu einer G_3 erfüllenden Belegung erweitern.
- Gilt $[G_3](\beta) = 1$, dann auch gilt auch $[F_3](\beta) = 1$.

(b) Zeigen Sie für beliebige aussagenlogische Formeln F, G, H:

- $(F \leftrightarrow (G \wedge H)) \equiv (F \vee \neg G \vee \neg H) \wedge (\neg F \vee G) \wedge (\neg F \vee H)$
- $(F \leftrightarrow (G \vee H)) \equiv (\neg F \vee G \vee H) \wedge (F \vee \neg G) \wedge (F \vee \neg H)$

$SAT \leq_p 3SAT$

$F \longmapsto F' = \bigwedge_{i=1}^{n} (L_{i,1} \vee L_{i,2} \vee L_{i,3})$ (poly)

op (G, H) $\rightsquigarrow (B_F \leftrightarrow (B_G \; op \; B_H)) \wedge (B_G \leftrightarrow \ldots) \wedge \ldots$

mit $F \equiv_e F'$

$\rightsquigarrow$ 3SAT bereits NP-vollständig (2SAT aber in P)

$3SAT \leq_p 3COL$

$F' = \bigwedge_{i=1}^{n} (L_{i,1} \vee L_{i,2} \vee L_{i,3}) \longmapsto$? Graph ?

3SAT $\leq_p$ 3COL

Literale $\longmapsto$ Knoten

Beziehungen zw Literalen $\longmapsto$ Kanten

+ Hilfsknoten & -kanten

Belegung $\longleftarrow$ Färbung

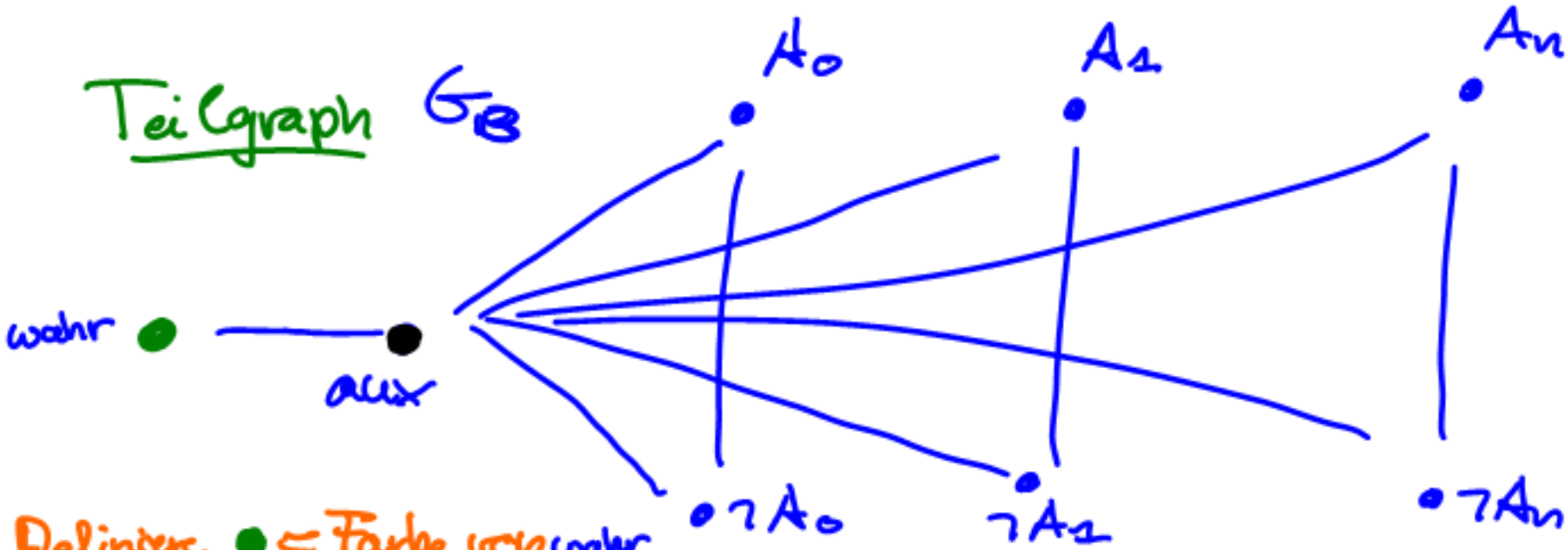
1. Ziel: Korrekte Färbung muss Belegung codiren

3SAT $\leq_p$ 3COL

1. Ziel: Korrekte Färbung muss A und $\neg A$ verschieden färben

und immer dieselben Farben für 1 und 0

Teilgraph G_B



Definiere ● = Farbe von wahr

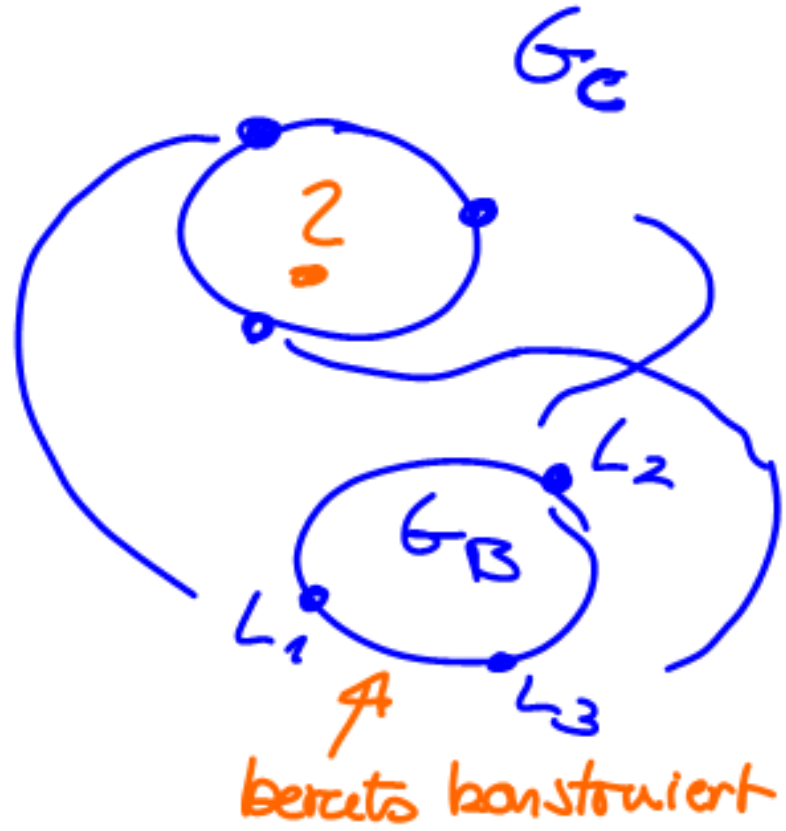
● = Farbe von aux

● = verbleibende Farbe (falsch)

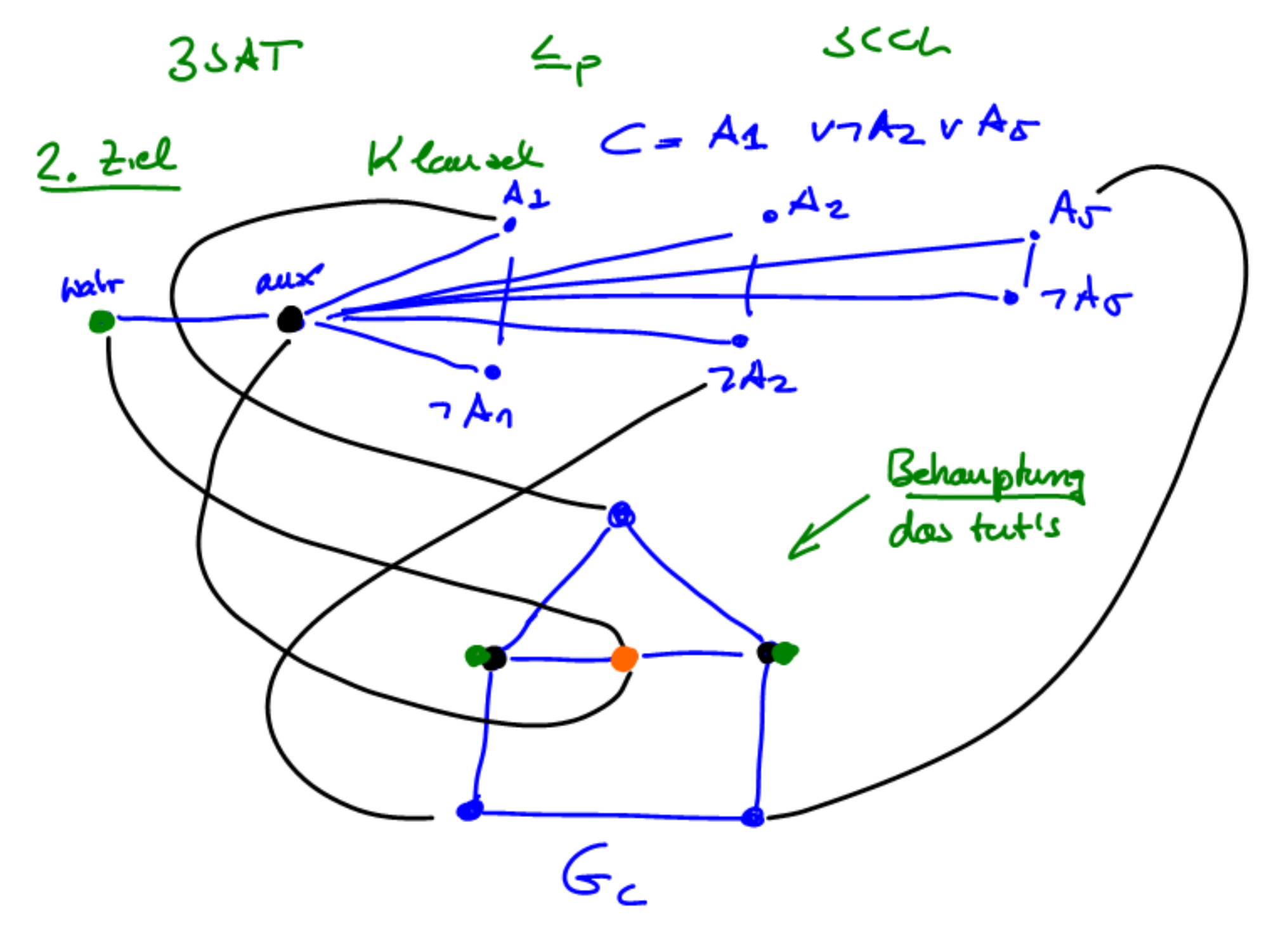
3SAT $\leq_p$ 3CCh

2. Ziel $\forall$ Klausel C

$C = L_1 \vee L_2 \vee L_3$ ⇝ G_C



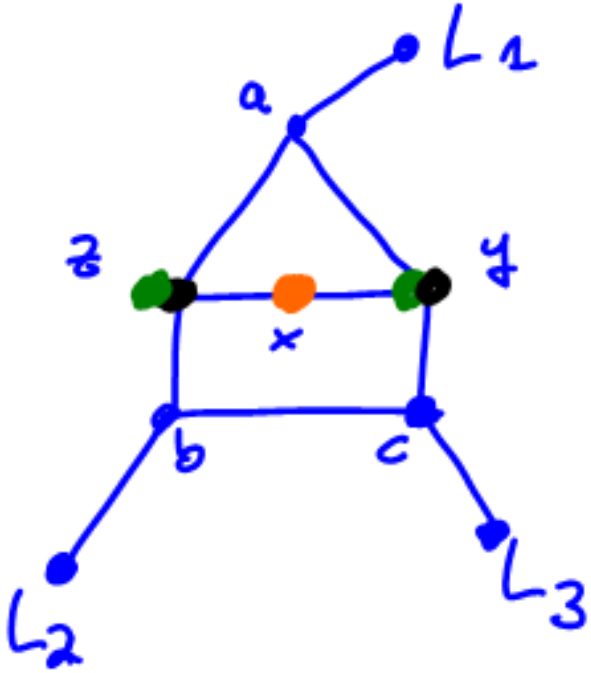
Idee Falls Färbung von G_B eine erfüllende Belegung von C codiert, dann und nur dann soll G_C korrekt gefärbt werden können.


3SAT
≤p
SCCh
2. Ziel
Klausel
C = A1 ∨ ¬A2 ∨ A5
A1
A2
A5
wahr
aux
¬A5
¬A1
¬A2
Behauptung
das tut's
GC

Intuition zu G_C mit $C = L_1 \vee L_2 \vee L_3$

Fallunterscheidung nach L_1:

- Falls L_1 grün, dann Farben von L_2 und L_3 egal
- sonst mindestens L_2 oder L_3 grün,
- ansonsten darf keine Färbung möglich sein.



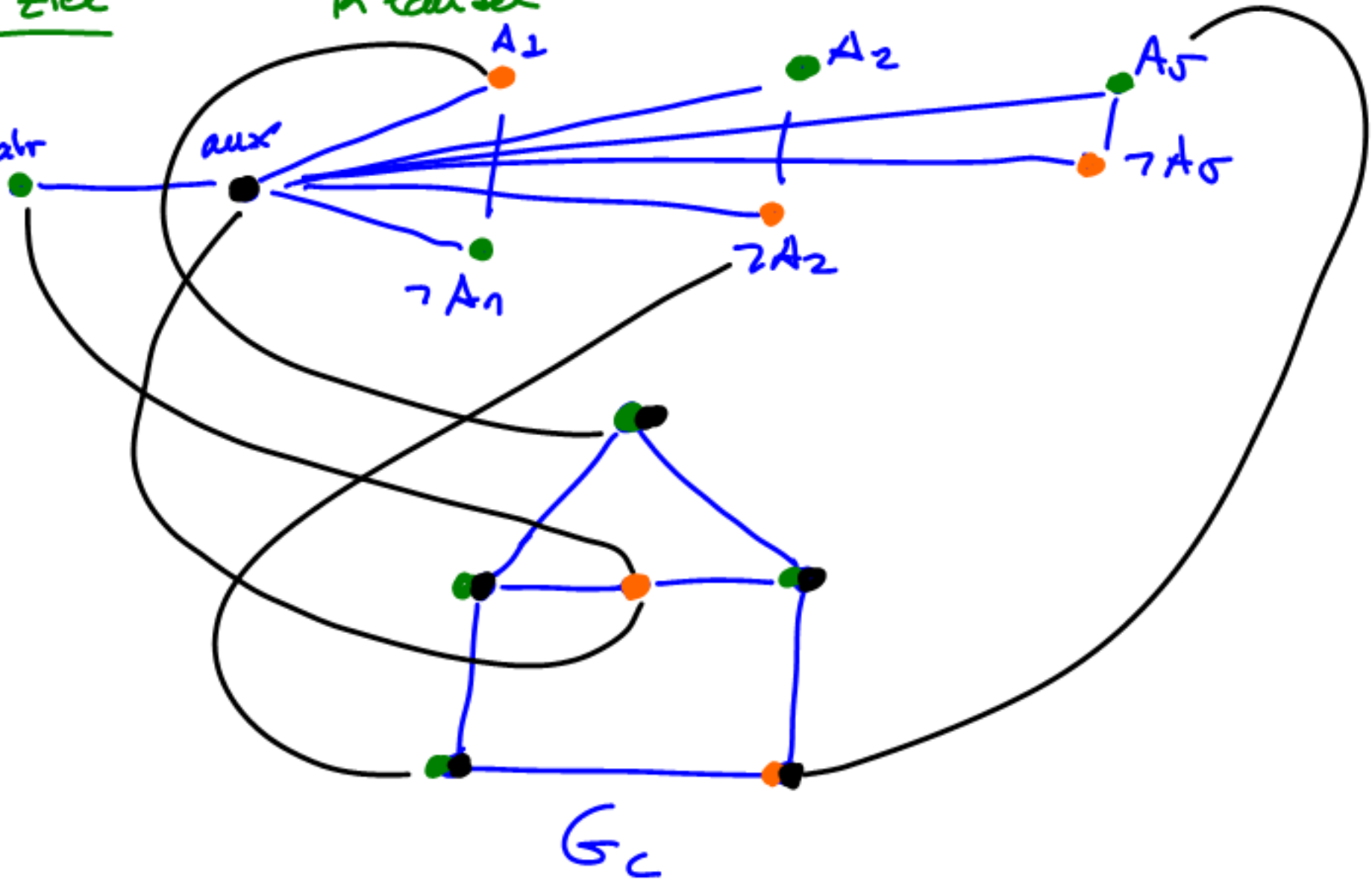
Erzwinge, dass

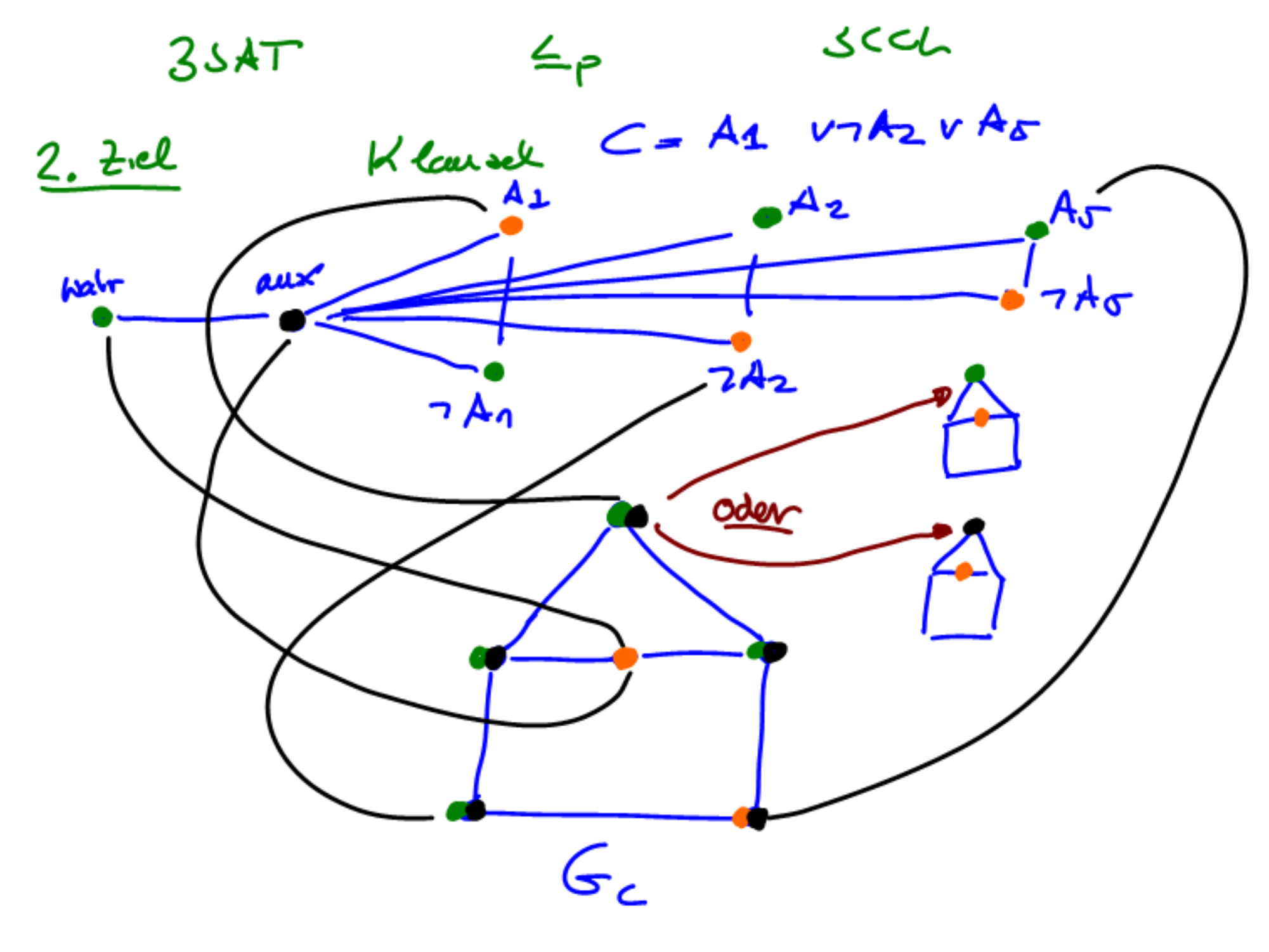
x stets ● mittels

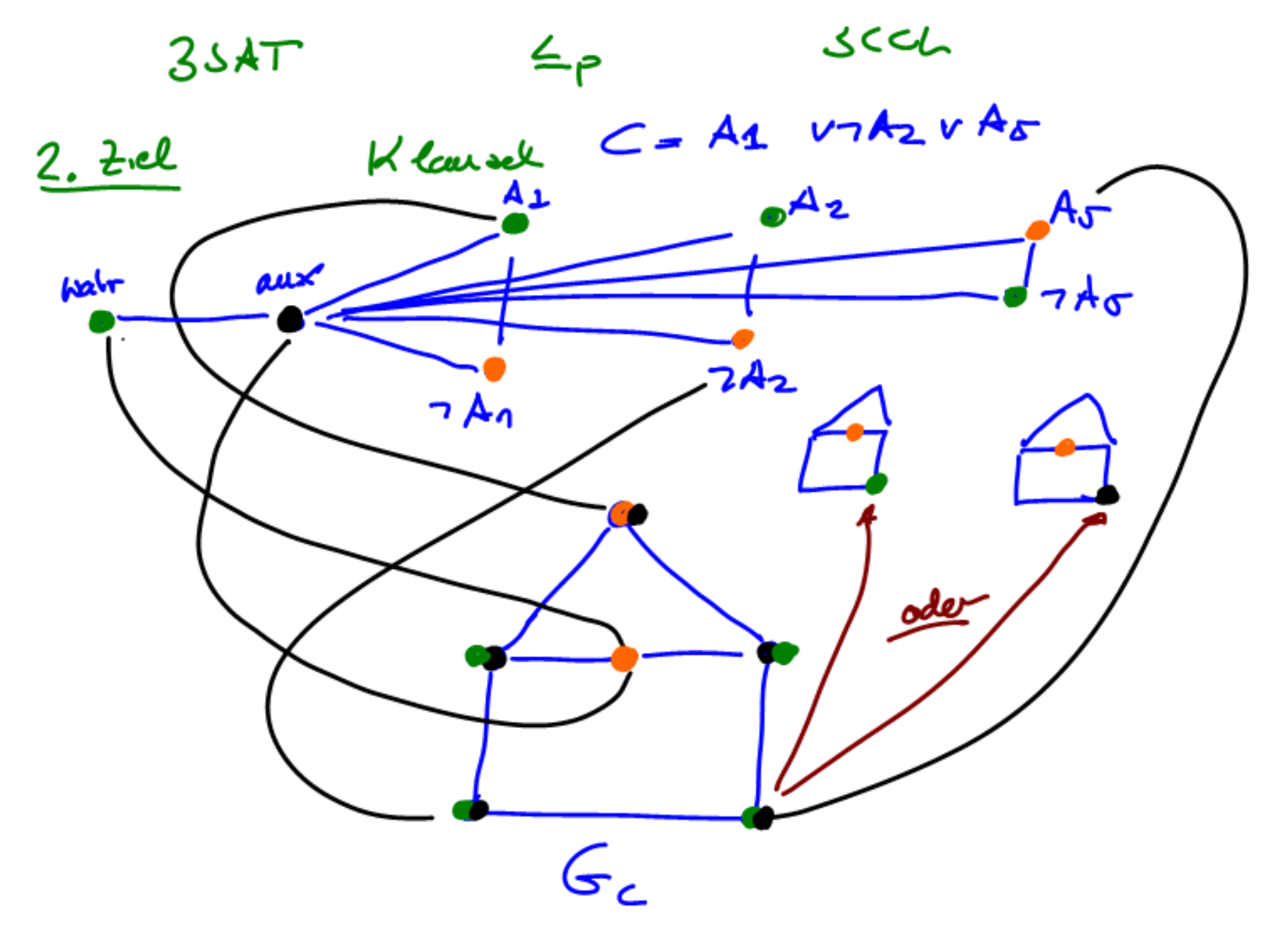
3SAT $\leq_p$ SCC

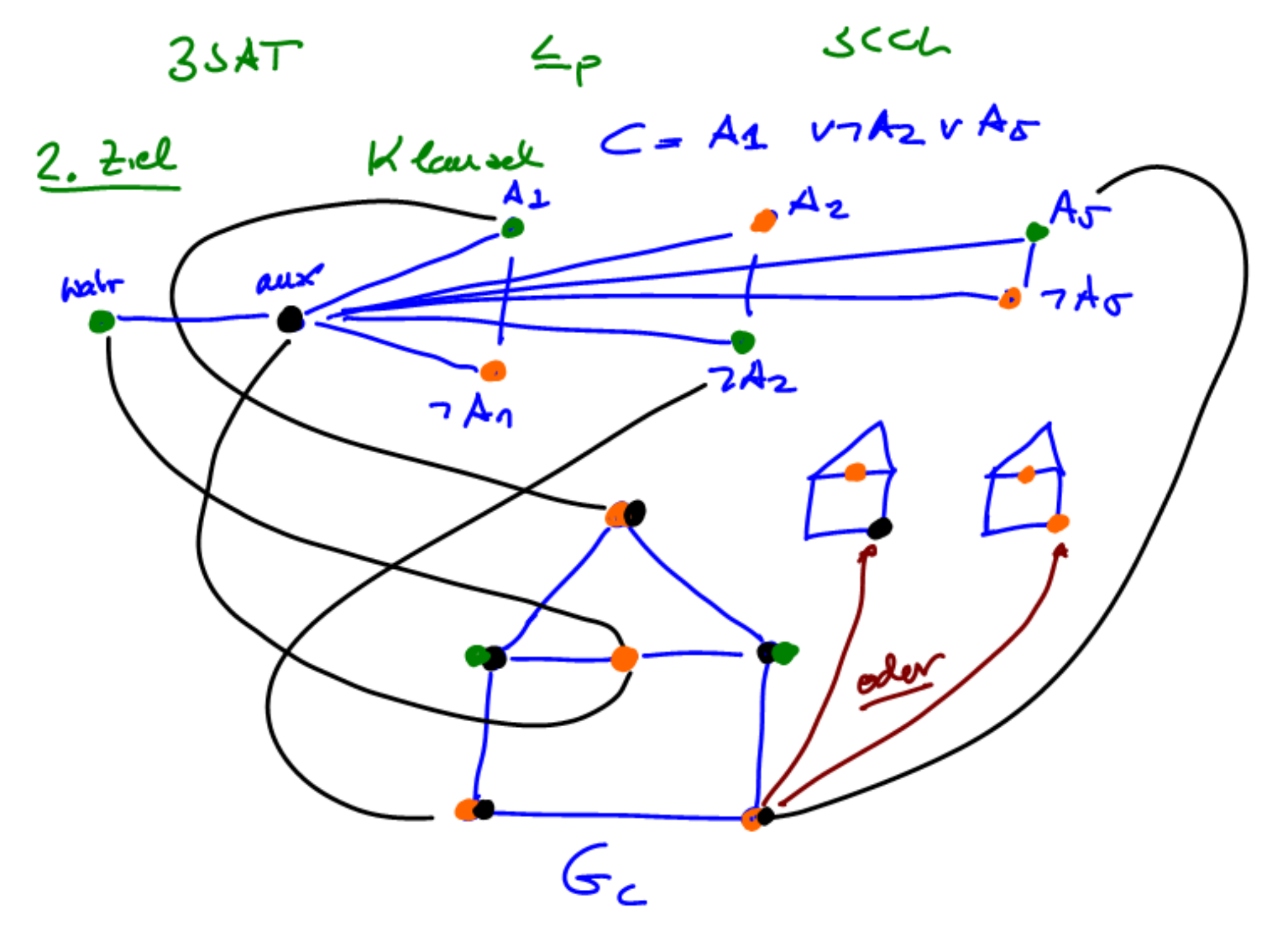
2. Ziel

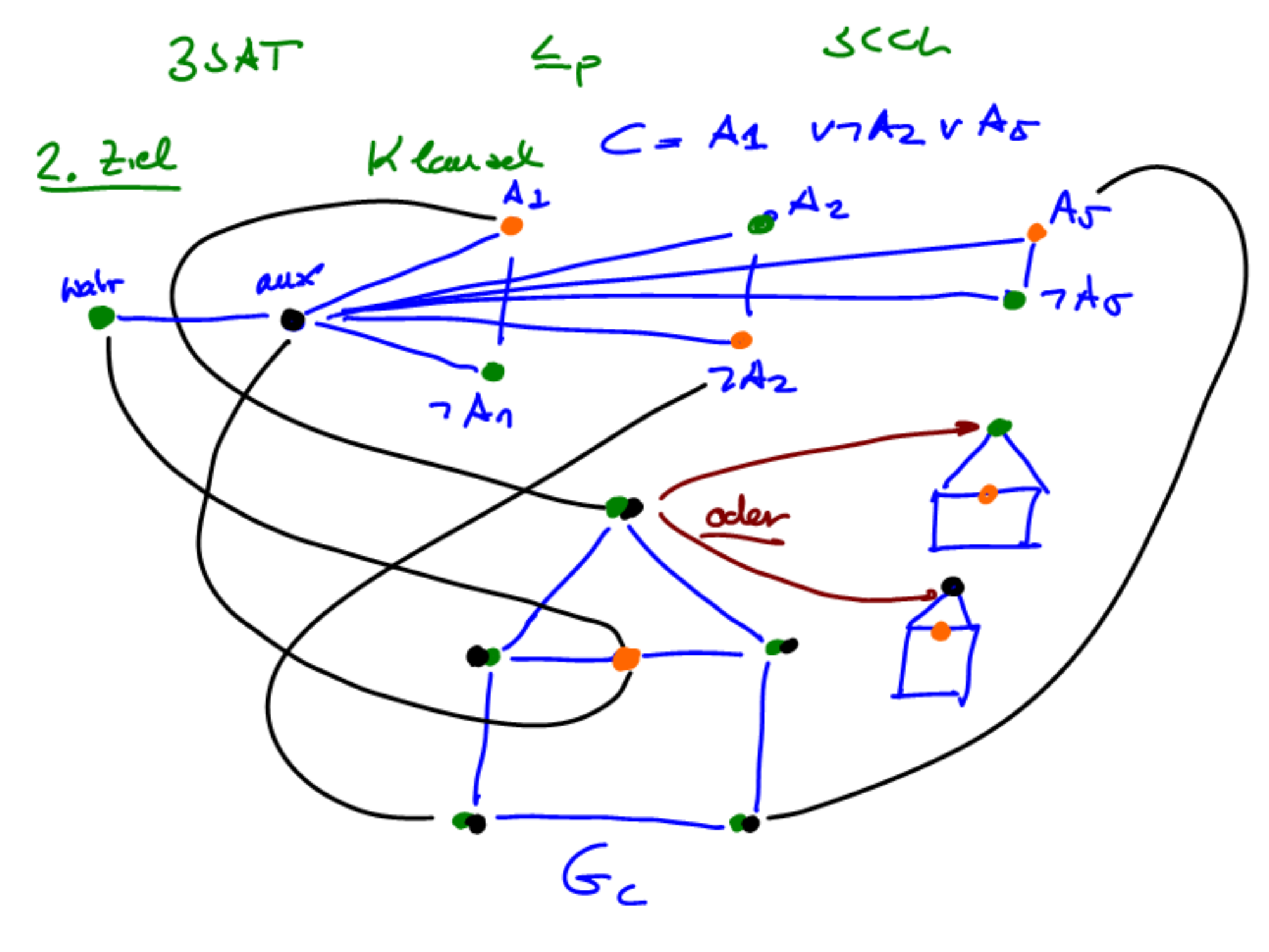
Klausel $C = A_1 \vee \neg A_2 \vee A_5$

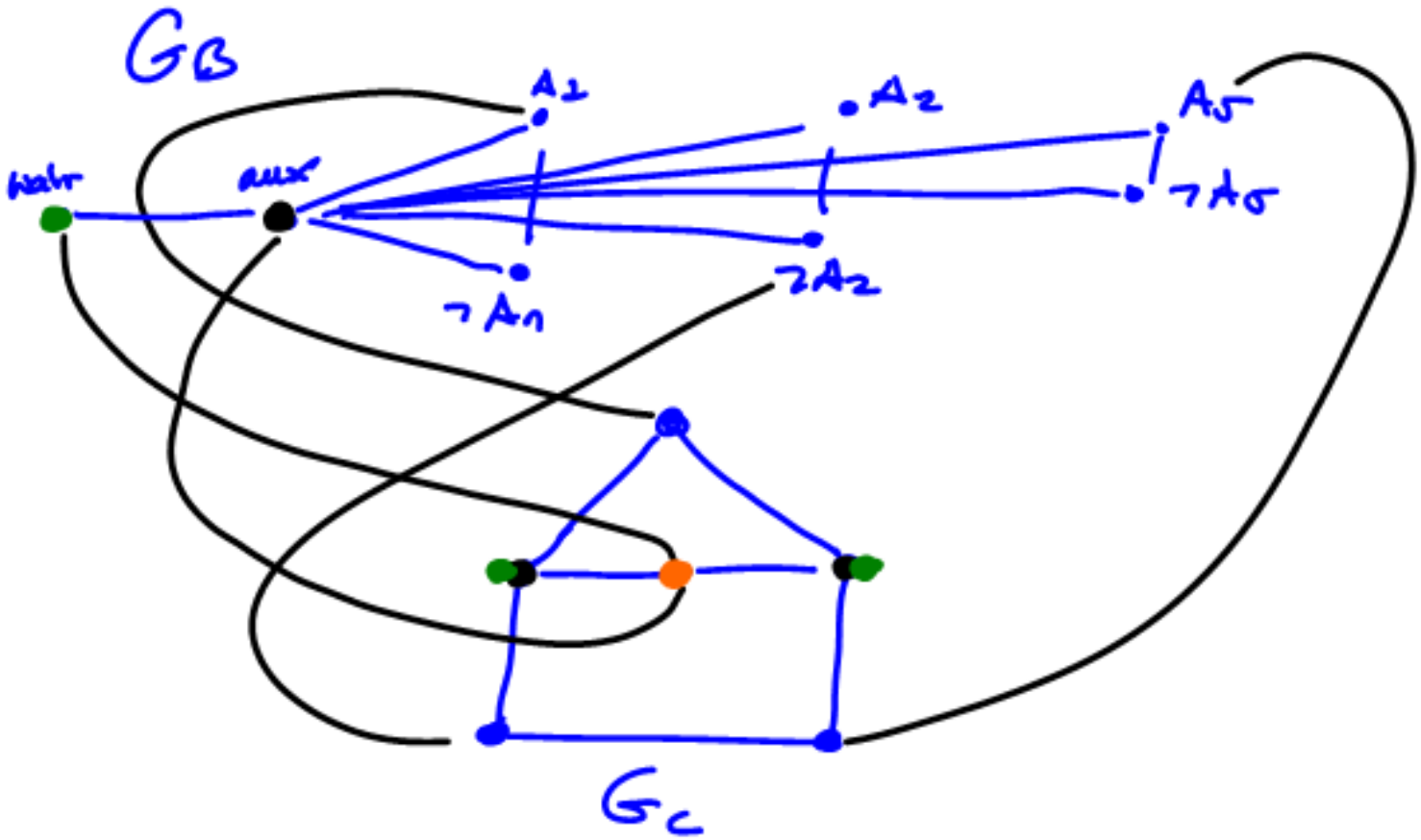



3SAT
≤p
SCCh
2. Ziel
Klausel
C = A1 ∨ ¬A2 ∨ A5
A1
A2
A5
wahr
aux
¬A5
¬A1
¬A2
oder
GC

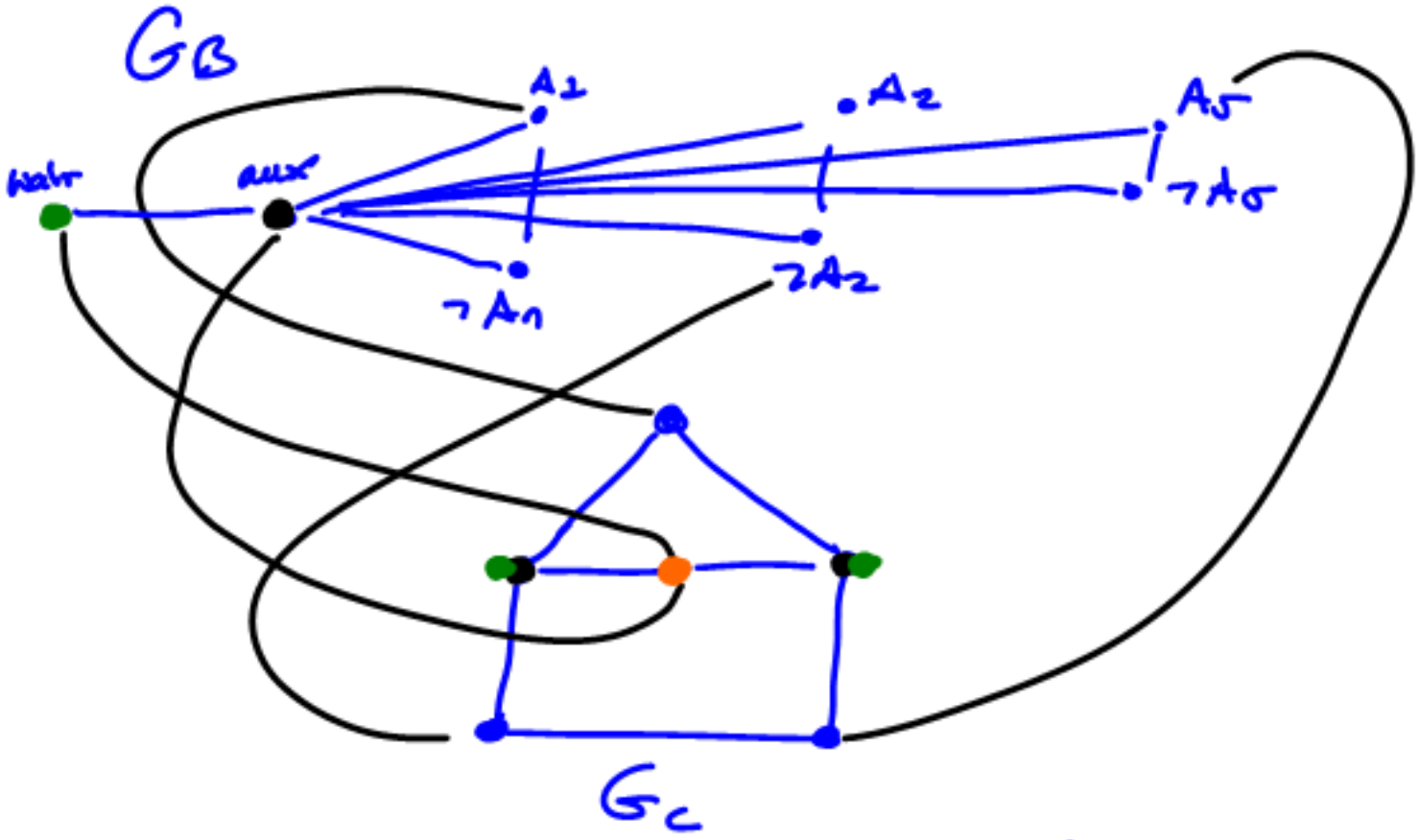

3SAT
≤p
SCCh
2. Ziel
Klausel
C = A1 ∨ ¬A2 ∨ A5
A1
A2
A5
wahr
aux
¬A5
¬A1
¬A2
oder
GC


3SAT
≤p
SCC
2. Ziel
Klausel
C = A1 ∨ ¬A2 ∨ A5
A1
A2
A5
wahr
aux
¬A5
¬A1
¬A2
oder
GC


3SAT
≤p
SCCh
2. Ziel
Klausel
C = A1 ∨ ¬A2 ∨ A5
A1
A2
A5
wahr
aux
¬A5
¬A1
¬A2
oder
GC



- Codiert Färbung von G_B eine erfüllende Belegung von C_i, dann kann G_C korrekt gefärbt werden (auf 2 Arten)
- Codiert Färbung von G_B eine passende nicht erfüllende Bel., dann kann G_C nicht korrekt gefärbt werden.



$$F = \bigwedge_{i=1}^{n} (L_{i1} \vee L_{i2} \vee L_{i3}) \xmapsto{\text{poly}} G_F$$

$$|G_F| = |G_B| + n \cdot (|G_C| + 4)$$

$$= 2n+2 \quad \text{mit} \quad |G_C| = 13$$

Was wäre, wenn 3SAT in PTIME auf 2COL reduzierbar?

Industrielle Anwendungen von SAT

1. Äquivalenztest von Schaltkreisen.

Schaltung	$\rightarrow$	Boole'sche Formel
Eingabe	$\rightarrow$	Belegung
Eingabe mit 1-Ergebnis	$\rightarrow$	Erfüllende Belegung
Funktionale Äquivalenz	$\rightarrow$	Logische Äquivalenz

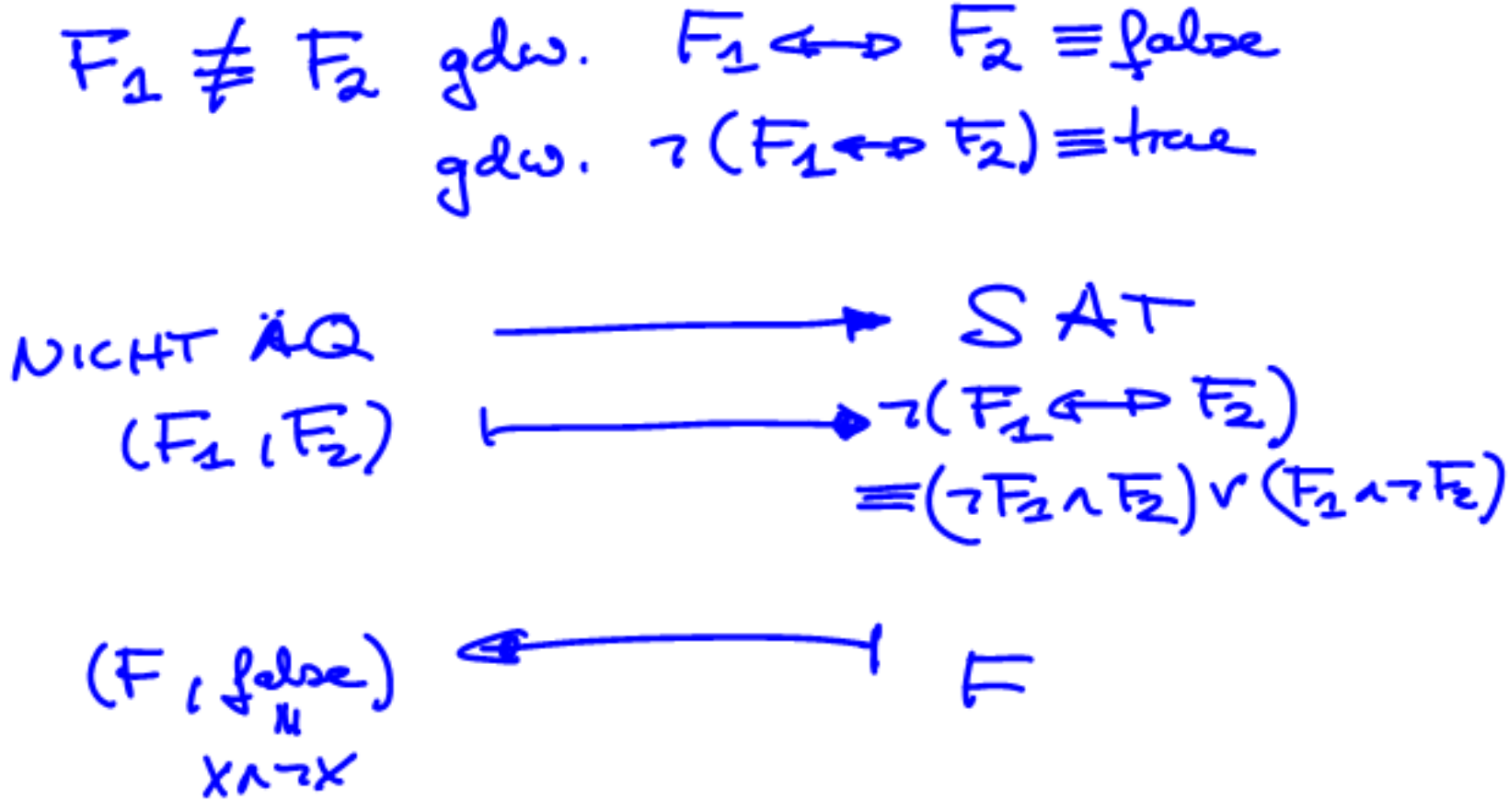
NICHTÄQUIVALENZ

Gegeben: Zwei aussagenlogische Formeln F_1, F_2 über Variablenmengen X_1, X_2.

Problem: Gibt es eine Belegung σ von $X_1 \cup X_2$ mit $\sigma(F_1)\neg\sigma(F_2)$?

Lemma 6.18

NICHTÄQUIVALENZ $\leq_p$ *SAT und SAT* $\leq_p$ *NICHTÄQUIVALENZ.*



2. Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für alle Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

2. Bounded Model Checking

Hardware Entscheide, ob eine Schaltung mit Zustand für alle Eingaben innerhalb von 10 Zyklen ein bestimmtes Verhalten (nicht) hat.

Software Entscheide, ob ein Programm für alle Eingaben innerhalb von 10 Schritten ein bestimmtes Verhalten (nicht) hat. Variablen müssen auf sehr kleine Wertebereiche eingeschränkt werden!

Aufgabe 4.3

Notation: $\mathbb{Z}_n := \{0, 1, 2, \ldots, n-1\}$ ist Menge der möglichen Reste bei Division durch n ($n \in \mathbb{N}$).

(a) Beschreiben Sie folgende Funktionen entsprechend TA3.3 durch aussagenlogische Formeln $\varphi_f, \varphi_g, \varphi_h$:

$$f\colon \mathbb{Z}_4 \to \mathbb{Z}_4 : x \mapsto (x-1) \bmod 4 \qquad g\colon \mathbb{Z}_4 \to \mathbb{Z}_4 : x \mapsto (x^2) \bmod 4 \qquad h\colon \mathbb{Z}_4 \to \mathbb{Z}_4 : x \mapsto (3 \cdot x) \bmod 4$$

(b) Beschreiben Sie folgendes Programm als aussagenlogische Formel φ. Verwenden Sie die Formeln aus (a) und Variablenumbenennung (vgl. Tutorübungen).

```
x = (x - 1) % 4  # (x − 1) mod 4
if x == 3:
  x = (x * x) % 4  # (x²) mod 4
else:
  x = (3*x) % 4  # (3 · x) mod 4
return x
```

Der initiale Werte der Progammvariablen x soll durch die aussagenlogischen Variablen x_0, x_1 beschrieben werden.

3. Programmoptimierung. Beispiel: Registerverteilung

> *Kann man in einem Programmstück n Variablen so auf k Register verteilen, dass jede Variable so lange in einem Register bleibt, wie sie lebendig ist?*

Variable ist an einem Punkt lebendig
gdw sie später noch gelesen wird, ohne vorher überschrieben worden zu sein.

Reduktion auf k-Färbbarkeit (und damit auf SAT):

Knoten	$=$	Variable
u und v verbunden	$=$	$u \neq v$ und es gibt einen Programmpunkt, an dem u und v lebendig sind
Farbe	$=$	Register
k-Färbung	$=$	Zuordnung eines Registers zu jeder Variablen

Sowohl k-Färbbarkeit als auch Registerverteilung ist für $k \geq 3$ NP-vollständig.

Mehr Information: Vorlesung *Programmoptimierung*

DS 2013
TA 13.1