

# Komplexitätstheorie

## WiSe 2011/12 in Trier

Henning Fernau  
Universität Trier  
fernau@uni-trier.de

## **Komplexitätstheorie** Gesamtübersicht

- Organisatorisches / Einführung  
Motivation / Erinnerung / Fragestellungen
- Diskussion verschiedener Komplexitätsklassen:  
Zeitkomplexität  
Platzkomplexität
- zugehörige Reduktionsbegriffe
- vollständige Probleme
- Anpassung von Klassenbegriffen und Reduktionen

## P- und NP-Vollständigkeit

Ein *And/Or-Graph* ist ein Paar  $(G, L)$  aus

—Graph  $G = (V, E)$  und

—*Label-Funktion (Beschriftung)*  $L: V \rightarrow \{\text{and}, \text{or}\}$

Darstellung eines And/Or-Graphen als  $(g)(l_1 \dots l_n)$

— $g$  übliche Adjazenzmatrix von  $G$

— $l_i \in \{0, 1\}$  für Label (Beschriftung)  $L(i)$  der Knoten  $i$

$$l_i = 0 \Leftrightarrow L(i) = \text{and}$$

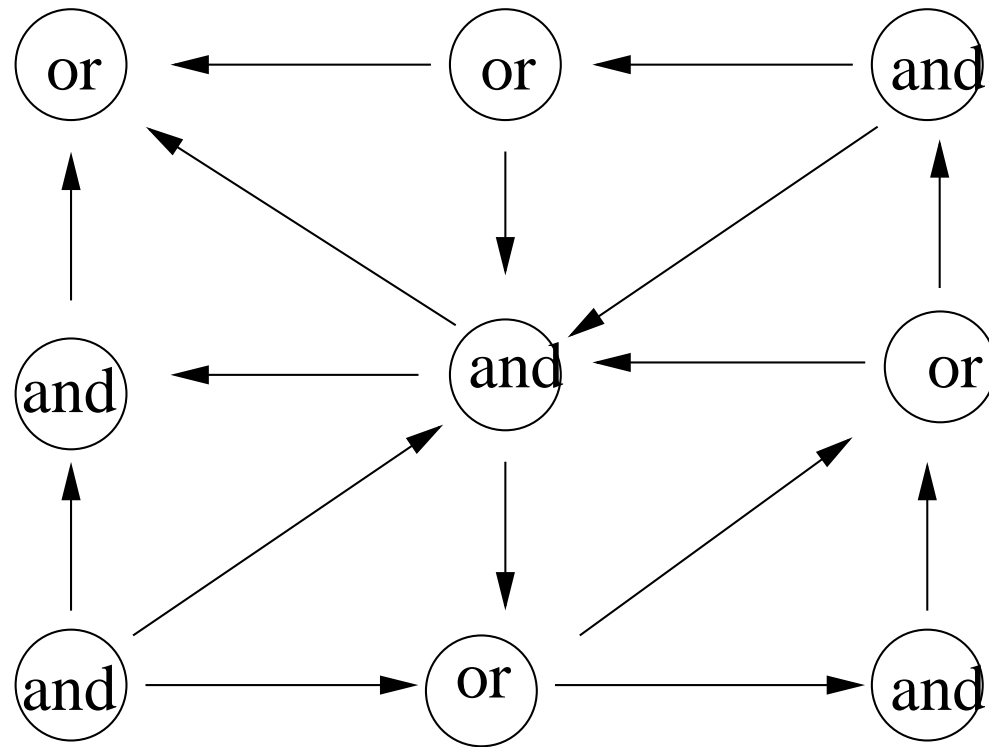
## Variante des Pebble-Spiels:

- Ein Knoten ohne Vorgänger kann jederzeit markiert werden.
- Ein Knoten mit Beschriftung `and` kann markiert werden, wenn `sämtliche` Vorgänger dieses Knotens markiert sind.
- Ein Knoten mit Beschriftung `or` kann markiert werden, wenn `mindestens einer` seiner Vorgänger markiert ist.

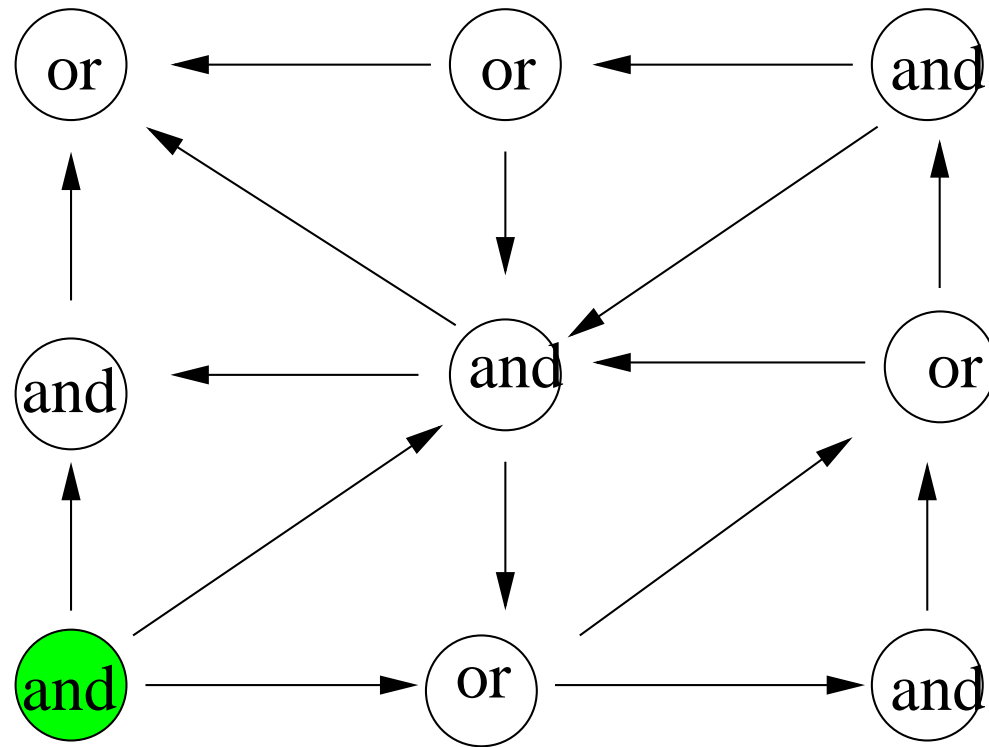
~> *Problem AGAP (And/Or-Graph Accessibility Problem):*

- Gegeben sei ein And/Or Graph  $(G, L)$ .
- **Frage:** Kann mit dem obigen Pebble-Spiel ein Knoten ohne Nachfolger markiert werden?

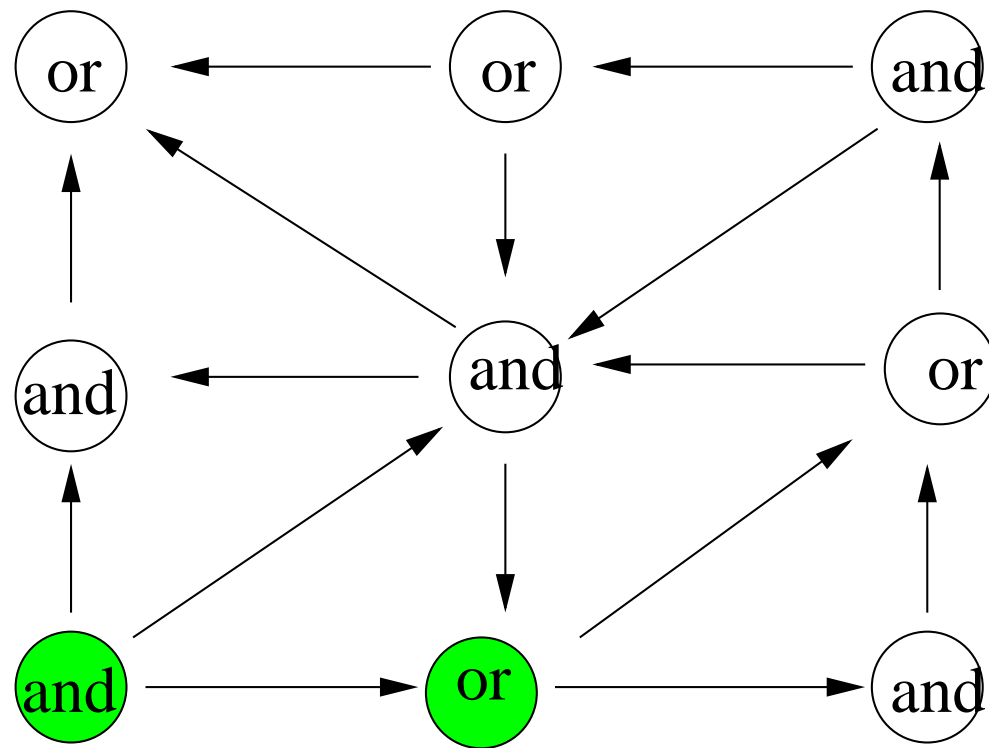
Beispiel:



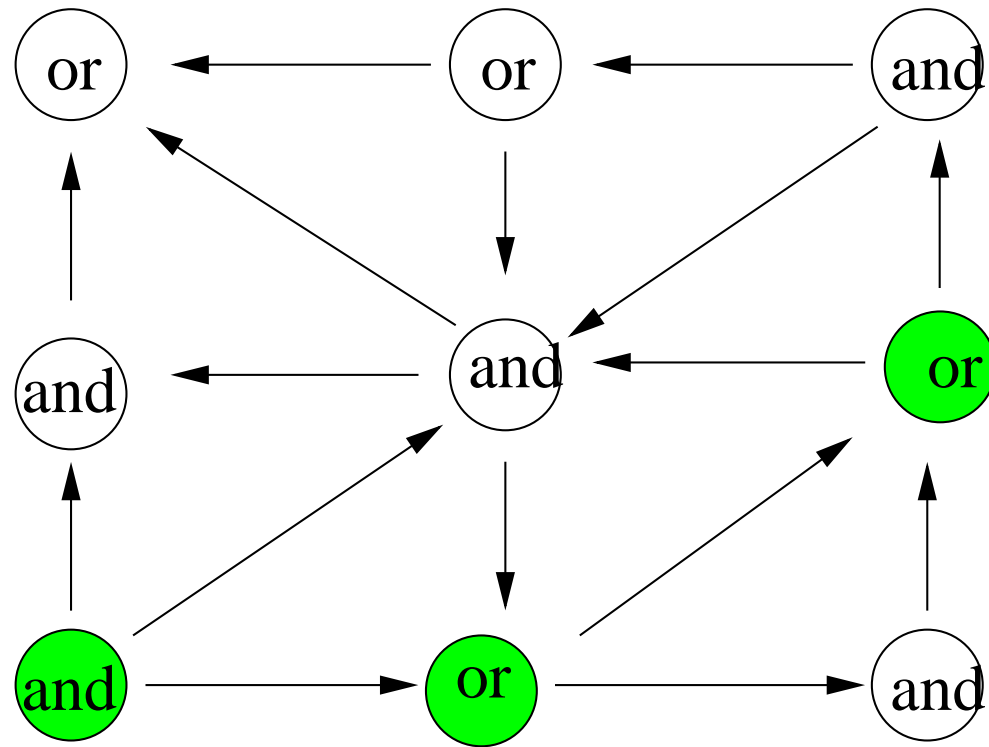
Beispiel:



Beispiel:

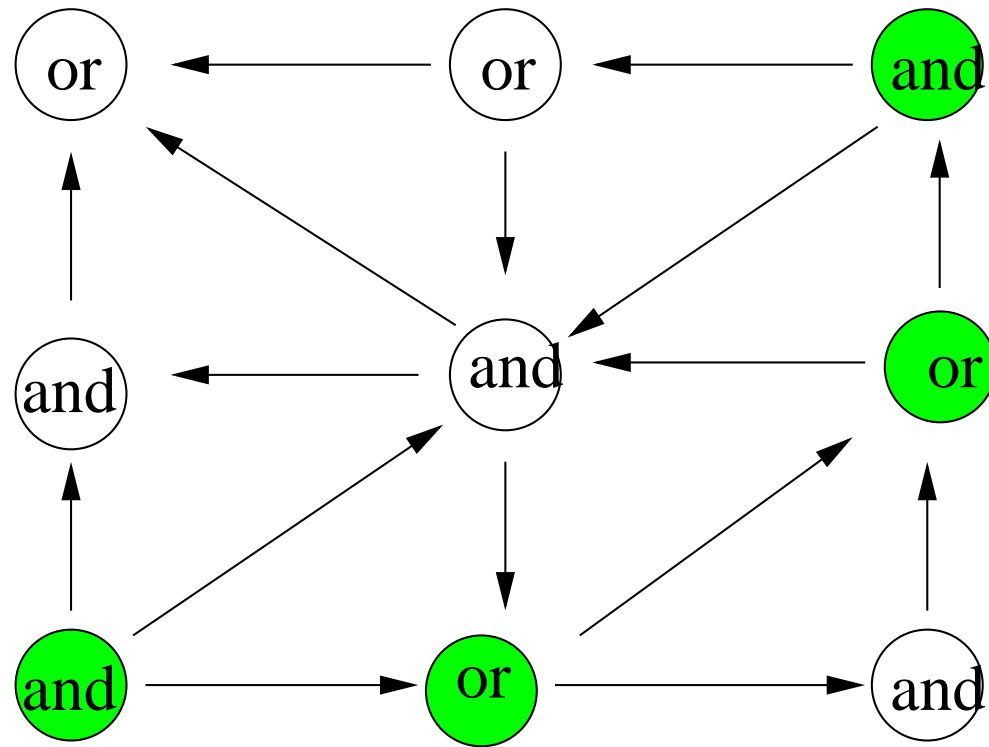


Beispiel:

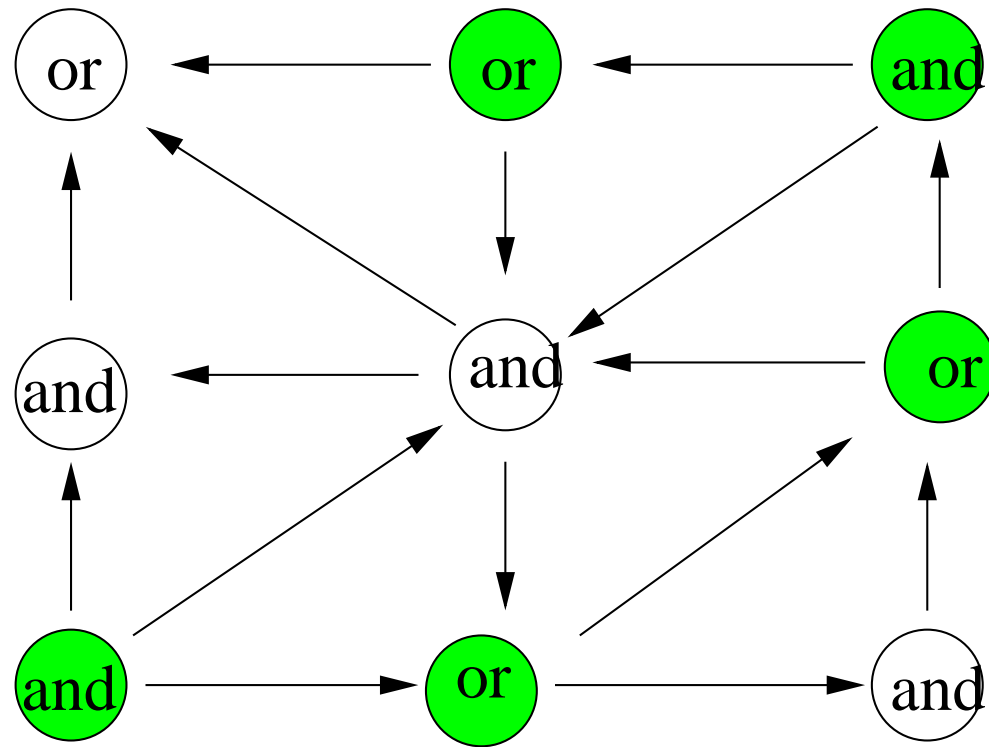




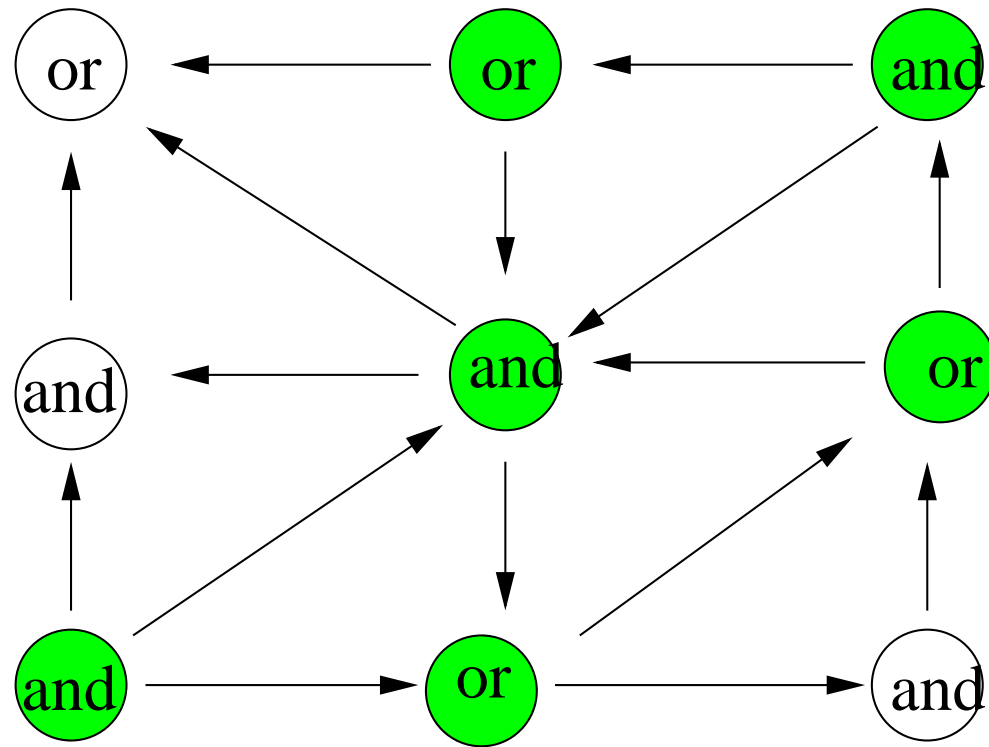
Beispiel:



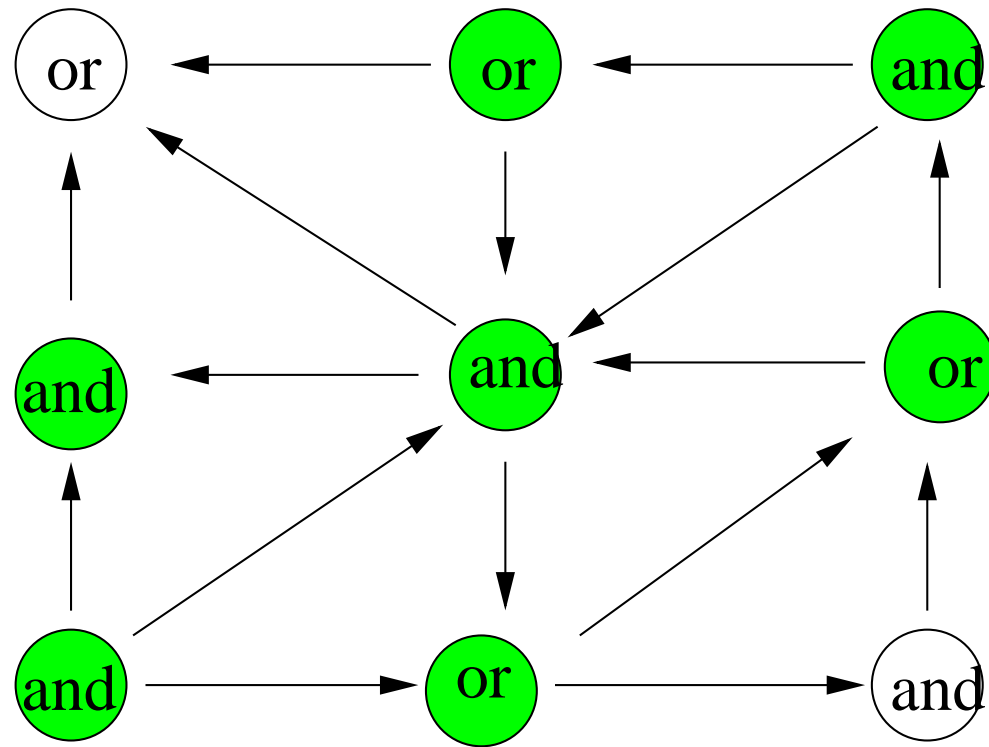
Beispiel:



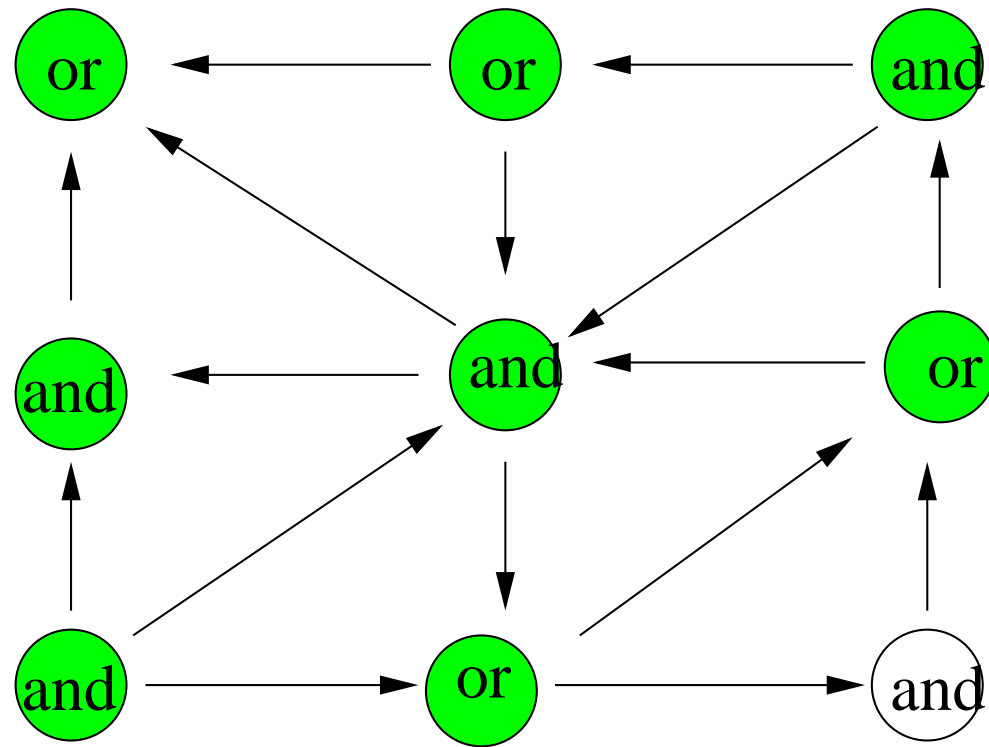
Beispiel:



Beispiel:



Beispiel:



**Lemma 1**  $AGAP \in \mathbf{P}$

Beweis: “Solange möglich, finde neue Markierungsmöglichkeit” liefert (ganz einfach) quadratischen Aufwand für das Problem.

**Satz 2**  $AGAP$  ist  $\mathbf{P}$ -vollständig (bzgl.  $\leq_{\log}$ ).

Ein ausführlicher Beweis für die Härte würde sicher 15 Folien kosten, daher folgt ein Abriss eines Beweises.

## AGAP ist P-vollständig: Grundidee

Betrachte  $L \in \mathbf{P}$  bel.  $L \subseteq W(\Sigma)$  wird von det. TM  $M$  akzeptiert mit

$T_M(x) \leq c \lg(x)^k + c =: t(x)$  für geeignete Konstanten  $k, c$ .

O.E.:  $M$  hat nur ein Arbeitsband und beschreibt nie das Ausgabeband.

$\leadsto$  Konfigurationen darstellbar durch  $(z, i, w_\ell, a, w_r)$  (s. NL-Vollständigkeit für GAP)

In jeder Konfig., die von  $I_M(x)$  in  $t \leq t(x)$  Schritten erreichbar ist, gilt:

$\lg(w_\ell) + \lg(w_r) \leq t(x)$ .

Konstruktion eines AND/OR-Graphen  $(G_x, L_x)$  mit  $x \in L \iff (G_x, L_x) \in \text{AGAP}$ .

Jeder Knoten entspricht (i.W.) einer Aussage über eine von  $I_M(x)$  aus erreichbare Konfig.

$(G_x, L_x)$  wird so konstruiert, dass nur Knoten für wahre Aussagen markiert werden können.

## Grundaufbau von $G_x = (V_x, E_x)$

Es gibt genau einen Knoten **START** ohne Vorgänger.

Von START aus können Knoten für  $I_M(x)$  markiert werden.

Es gibt genau einen Knoten **ZIEL** ohne Nachfolger.

ZIEL kann markiert werden, wenn ein markierter Knoten aussagte: Endkonfig. wurde erreicht.

Es gibt einen weiteren Knoten **KEINSTART**, der Vorgänger aller Knoten außer START ist.

Insbesondere ist KEINSTART Vorgänger von KEINSTART.

Es gibt einen weiteren Knoten **KEINZIEL**, der Nachfolger aller Knoten außer ZIEL ist.

Insbesondere ist KEINZIEL Nachfolger von KEINZIEL.

Nach Def. des Pebble-Spiels muss die Markiererei bei Knoten vom Eingangsgrad Null beginnen.  $\rightsquigarrow$  Nur START kommt dafür in Frage.

Ebenso ist die Markiererei nur dann "erfolgreich", wenn ein Knoten vom Ausgangsgard Null erreicht wird.  $\rightsquigarrow$  Nur ZIEL kommt dafür in Frage.

Die bislang aufgeführten Knoten werden **OR-beschriftet**.



## Hauptknoten von $V_x I$ ,

denen ein Zeitpunkt  $t \leq t(x)$  zugeordnet werden kann.

$S(t, p, \gamma)$ : Nach  $0 \leq t \leq t(x)$  Schritten steht  $\gamma$  auf Position  $p$ ,  $-t(x) \leq p \leq t(x)$ , auf dem Arbeitsband.

Die Position  $p$  versteht sich *relativ zur aktuellen Kopfposition*.

$P(t, i)$ : Nach  $0 \leq t \leq t(x)$  Schritten steht der Eingabekopf auf dem  $i$ -ten Zeichen von  $x$ ,  $0 \leq i \leq \lg(x) + 1$ .

Eingangs  $P(0, 0)$  markierbar,  $P(0, i)$  jedoch nicht für  $i > 0$ .

$Z(t, z)$ : Nach  $0 \leq t \leq t(x)$  Schritten wurde Zustand  $z$  erreicht.

Die voranstehenden Knoten werden mit **OR beschriftet**.

Damit für  $t = 0$  alle Knoten markiert werden können, die wahren Aussagen über  $I_M(x)$  entsprechen, werden die Kanten  $(\text{START}, Z(0, z_0))$ ,  $(\text{START}, P(0, 0))$  und  $(\text{START}, S(0, p, B))$  für  $-t(x) \leq p \leq t(x)$  in  $E_x$  aufgenommen.

Außerdem werden  $(Z(t, z_e), \text{ZIEL})$  für Endzustände  $z_e$  Kanten.

## Hauptknoten von $V_x$ II,

denen ein Zeitpunkt  $t \leq t(x)$  zugeordnet werden kann.

$B(t, b)$ : Im Schritt  $t$  wird Befehl  $b$  ausgeführt.

Nach unserer TM-Def. hängt der Befehl nur vom Zustand ab, daher gibt es (im positiven Fall) Kante von  $Z(t-1, z)$  nach  $B(t, b)$ .  $B(t, b)$  wird mit **OR beschriftet**.

$B \wedge S(t, b, p, \gamma)$ : Soll markiert werden können, wenn  $B(t, b)$  und  $S(t-1, p, \gamma)$  “wahr” sind.

Also: Vor Schritt  $t$  steht  $\gamma$  an der  $p$ -ten Stelle des Arbeitsbands und es soll der Befehl  $b$  ausgeführt werden.  $\rightsquigarrow$  **AND-Beschriftung**.

$B \wedge P(t, b, i)$ : Soll markiert werden können, wenn  $B(t, b)$  und  $P(t-1, i)$  “wahr” sind.  $\rightsquigarrow$  **AND-Beschriftung**.

Dann kann z.B.  $S(t, p, \gamma)$  mit Hilfe von  $B \wedge S(t, b, p, \gamma)$  richtig gesetzt werden.

## Welche Kanten müssen eingefügt werden?

$Z(t, z)$  soll markiert werden können, wenn

- $B(t, b)$  markiert werden kann für ein  $b \in \{(1, L, z), (1, R, z), (2, L, z), (2, R, z)\} \cup \{2, \gamma, z \mid \gamma \in \Gamma\}$ ; oder
- $B(t, b)$  und  $S(t - 1, 0, \gamma)$ , also  $B \wedge S(t, b, 0, \gamma)$ , markiert sind für  $g, \gamma$  mit  $b \in \{(2, \gamma, z, z') \mid z' \in Z\}$ ; oder
- $B(t, b)$  und  $S(t - 1, 0, \gamma)$ , also  $B \wedge S(t, b, 0, \gamma)$ , markiert sind für  $g, \gamma$  mit  $b \in \{(2, \gamma', z', z) \mid z' \in Z, \gamma' \neq \gamma\}$ ; oder
- $B(t, b)$  und  $P(t - 1, i)$ , also  $B \wedge P(t, b, i)$ , markiert sind für  $b, i$  mit  $b \in \{(1, a_i, z, z') \mid z' \in Z\}$ , wobei  $a_i$  das  $i$ -te Zeichen der Eingabe ist; oder
- $B(t, b)$  und  $P(t - 1, i)$ , also  $B \wedge P(t, b, i)$ , markiert sind für  $b, i$  mit  $b \in \{(1, b, z', z) \mid z' \in Z, b \neq a_i\}$ .

## Welche Kanten müssen eingefügt werden?

$P(t, i)$  soll markiert werden können, wenn

- $P(t - 1, i)$  und  $B(t, b)$ , also  $B \wedge P(t, b, i)$ , markiert sind für einen Befehl  $b$ , der den Eingabekopf nicht bewegt; oder
- $B \wedge P(t, b, i + 1)$  markiert ist für  $b = (1, L, z)$ ; oder
- (im Spezialfall  $i = 0$ )  $B \wedge P(t, b, 0)$  markiert ist für  $b = (1, L, z)$ ; oder
- $B \wedge P(t, b, i - 1)$  markiert ist für  $b = (1, R, z)$ ; oder
- (im Spezialfall  $i = \lg(x) + 1$ )  $B \wedge P(t, b, i)$  markiert ist für  $b = (1, R, z)$ .

## Welche Kanten müssen eingefügt werden?

$S(t, p, \gamma)$  soll markiert werden können, wenn

- $B \wedge S(t, b, p, \gamma)$  markiert ist, wobei  $b$  auf Band 1 operiert oder auf Band 2 testet; oder
- $B \wedge S(t, b, p, \gamma)$  markiert ist, wobei  $b$  auf Band 2 schreibt und  $p \neq 0$  ist; oder
- $B(t, b)$  markiert ist für  $b \in \{2\} \times \{\gamma\} \times Z$  und  $p = 0$ ; oder
- $B \wedge S(t, b, p - 1, \gamma)$  markiert ist für  $b \in \{2\} \times \{L\} \times Z$ ; oder
- $B \wedge S(t, b, p + 1, \gamma)$  markiert ist für  $b \in \{2\} \times \{R\} \times Z$ ; oder
- $B(t, b)$  markiert ist für  $b \in \{2\} \times \{L\} \times Z$  sowie  $p = -t(x)$ ,  $\gamma = B$  (Band links mit Blanks gefüllt) oder für  $b \in \{2\} \times \{R\} \times Z$  sowie  $p = t(x)$ ,  $\gamma = B$  (Band rechts mit Blanks gefüllt).

## Zur Komplexität der Konstruktion

Die Knotenarten enthalten allesamt Zahlen zur genauen Angabe; diese können daher in  $FSPACE(\log)$  erzeugt werden.

Ähnliches gilt für die Kanten.

Beachte: Reduktionsmaschine kennt  $x$ .

## Zur Korrektheit der Konstruktion

Durch Induktion über  $t$  zeigt man folgende Aussagen:

- (1) Für jedes  $t$  und jedes  $p$  gibt es maximal ein  $\gamma$ , für das  $S(t, p, \gamma)$  markiert werden kann.
- (2) Für jedes  $t$  gibt es maximal ein  $i$  und ein  $z$ , sodass (a)  $P(t, i)$  und (b)  $Z(t, z)$  markiert sind.
- (3) Für jedes  $t \geq 1$  gibt es maximal ein  $b$ , sodass  $B(t, b)$  markiert ist.
- (4) Ist zu einem  $t$  ein Knoten  $Z(t, z)$  markiert, so ebenfalls ein  $P(t, i)$  und für jedes  $p$  ein  $S(t, p, \gamma)$ .
- (5) Ist für ein  $t$  ein  $Z(t, z)$  oder  $S(t, p, \gamma)$  oder  $P(t, i)$  markiert, so ist auch für jedes  $t' < t$  ein  $Z(t', z')$  markiert.

Die Menge aller in  $G_x$  markierbaren Knoten definiert damit eindeutig eine Folge von Konfigurationen  $K_0, \dots, K_m$  mit  $K_0 = I_M(x)$ ,  $K_m$  ist Endkonfiguration,  $K_i \rightarrow_M K_{i+1}$ ,  $m \leq t(x)$ .

Die markierten Knoten entsprechen wahren Aussagen über die Konfigurationen, und kein Knoten mit Index  $t' > t(x)$  wurde markiert.

Umgekehrt gilt Entsprechendes.

Kleine Modifikation von Problem + Beweis

⇒ **NP-Vollständigkeit!**

Problem **AGAP mit verbotenen Paaren** (kurz:  $\text{AGAP}_{\text{vP}}$ ):

- **Gegeben seien** ein And/Or Graph  $(G, L)$  und eine Menge

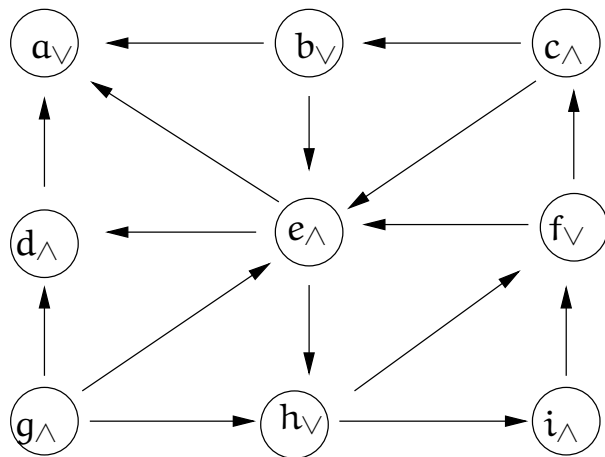
$$\{(x_1, y_1), \dots, (x_m, y_m)\}$$

von Paaren von Knoten (die **verbotenen Paare**)

- **Frage:** Kann mit dem obigen Pebble-Spiel ein Knoten ohne Nachfolger markiert werden, wenn **von jedem verbotenen Paar maximal ein Knoten markiert** werden darf?



## Beispiel zu $AGAP_{\forall P}$



Menge der verbotenen Paare:  $(b, i), (h, d)$ .

Möglicher Ablauf des Pebble-Spiels:

$g \rightarrow h \rightarrow f \rightarrow c \rightarrow b \rightarrow a$ .

**Beobachte:**  $d$  und  $i$  werden nicht markiert.

**Achtung:** Im Ggs. zu  $AGAP$  darf nicht jederzeit jeder markierbare Knoten auch markiert werden. Es muss also **richtig geraten** werden.

Beispiel:  $g \rightarrow h \rightarrow i \rightarrow f \rightarrow c$  bleibt stecken.

**Folgerung 3**  $AGAP_{\forall P}$  ist **NP**-vollständig (bzgl.  $\leq_{\log}$ ).

Beweisidee: Zu gegebener nichtdet. TM  $M$  zur Akzeptanz von  $L \in \mathbf{NP}$  mit  $T_M(x) \leq c \lg(x)^k + c$  konstruiere “wie zuvor bei  $AGAP$ ” nun  $AGAP_{\forall P}$ -Instanz.

$M$  nichtdeterministisch

$\leadsto$  In Zustand  $z$  gibt es mehrere mögliche ausführbare Befehle  $b \in \delta(z)$ .

$\leadsto$  Definiere zu  $z$  als Menge verbotener Paare alle  $(B(t, b), B(t, b'))$  mit  $b \neq b'$ .

Klar: Konstruktion geht in Logspace.

Gibt es “Akzeptanzweg”, d.h., Konfigurationsfolge für  $x$  durch  $M$ ,  
so gibt es auch eine Markierungsfolge in konstruierter  $AGAP_{\forall P}$ -Instanz  
und umgekehrt.

## Umfangreiche Liste NP-vollständiger Probleme

- Garey, M.R., Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco 1979 sowie als Fortsetzung davon:
- Johnson, D.S., The NP-completeness column: an ongoing guide, seit 1981 in der Zeitschrift “Journal of Algorithms”, später in “ACM Transactions on Algorithms”

Noch einmal als Merksatz:

Solange es nicht gelungen ist,  $P = NP$  zu beweisen, ist für keines der NP-vollständigen Probleme ein praktisch verwendbarer Algorithmus bekannt!

Ähnlich zu AGAP / AGAGvP: **Boole'sche Schaltkreise**

### Schaltkreis:

- gerichteter Graph  $(V, E)$
- markierte (s.u.) Knoten  $V = \{1, 2, \dots, n\}$
- $x < y$  bei jeder Kante  $(x, y) \in E$

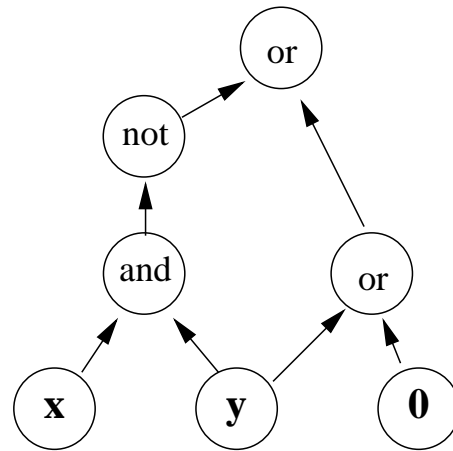
$\leadsto$  Graph zyklensfrei, bereits topologisch sortiert.

### Markierungen:

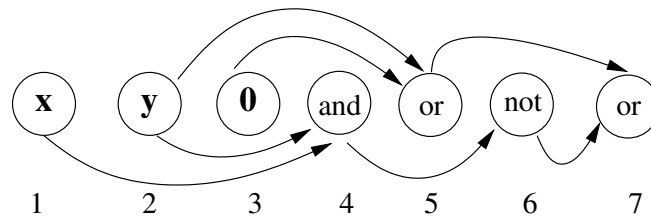
- In-Grad 0: Konstanten 0, 1, boole'sche Variablen  $x_1, x_2, \dots$
- In-Grad  $> 0$ : logische Operatoren  $\vee, \wedge$
- Ingrad 1: evtl. Negation  $\neg$

Boole'sche Variablen mit logischen Werten belegt

$\Rightarrow$  Wert des Knotens  $n$  als *Ausgabewert*



topologisch sortiert:



- CIRCUIT VALUE:

**Gegeben** sei ein Boole'scher Schaltkreis über  $\{\vee, \wedge, \neg, 0, 1\}$  *ohne Variablen*.

**Frage:** Ist der Ausgabewert gleich 1?

- MONOTONE CIRCUIT VALUE:

**Gegeben** sei ein Boole'scher Schaltkreis über  $\{\vee, \wedge, 0, 1\}$  *ohne Negation und ohne Variablen*.

**Frage:** Ist der Ausgabewert gleich 1?

- CIRCUIT SAT:

**Gegeben** sei ein Boole'scher Schaltkreis über  $\{\vee, \wedge, \neg, 0, 1\}$  *mit Variablen*.

**Frage:** Gibt es eine Variablen-Belegung mit Ausgabewert 1?

**Folgerung 4** Sowohl CIRCUIT VALUE *als auch* MONOTONE CIRCUIT VALUE sind **P**-vollständig, CIRCUIT SAT *ist* **NP**-vollständig.

## MONOTONE CIRCUIT VALUE **ist P-hart**

Kleine Modifikation zum  $AGAP$ -Beweis:

Entferne die Schlingen an den Knoten KEINSTART und KEINZIEL

$\rightsquigarrow$  So modifizierter Graph  $G'_x$  ist zyklensfrei.

ZIEL sei in topologischer Sortierung der letzte Knoten und legt so den Ausgabe-  
wert fest.

Die beiden einzigen Knoten mit Eingangsgrad Eins sind START und KEINSTART;  
START bekommt den Wert 1 zugewiesen und KEINSTART 0 (als Eingabe).

Alternativ hätte man KEINSTART auch gänzlich löschen können.

**Offenbar:**  $G_x \in AGAP \iff G'_x \in MONOTONE\ CIRCUIT\ VALUE.$

## Mehr Argumente für die Folgerung

**Klar:** Die Auswertung eines Schaltkreises kann mit einem deterministischen Polynomzeitalgorithmus erfolgen.

~> Sowohl `CIRCUIT VALUE`  
als auch `MONOTONE CIRCUIT VALUE` sind **P**-vollständig.

`CIRCUIT SAT` ist **NP**-vollständig:

(1) Härte: Die Befehlsauswahl (bei  $AGAP_{\forall P}$ ) im  $t$ -ten Schritt kann durch Variablenbelegung simuliert werden.

(2) in **NP**: Rate zunächst Variablenbelegung;  
dann erfolgt Überprüfung (`CIRCUIT VALUE`)

*Guess and Check* ist übliche Strategie für **NP**-Algorithmen.



Im Folgenden **NP**-vollständige Probleme aus verschiedenen Problembereichen:

Logik, Graphentheorie, Mengen- und Zahlentheorie