Algorithmische Spieltheorie

– Grundlagen der Komplexitätstheorie –

Sven Kosub

AG Algorithmik/Theorie komplexer Systeme
Universität Konstanz

E 202 | Sven.Kosub@uni-konstanz.de | Sprechstunde: Freitag, 12:30-14:00 Uhr, o.n.V.

Sommersemester 2009

Polynomialzeit-Algorithmen

Intuition:

Berechnungsprobleme, die Polynomialzeit-Algorithmen erlauben, können als (theoretisch) effizient lösbar angesehen werden

Fragen:

- Gibt es Berechnungsprobleme, die nicht effizient lösbar sind?
- Wie können wir dies für ein konkretes Problem feststellen?
- Ist die Bestimmung von Nash-Gleichgewichten effizient lösbar?
- ...

Eingabegrößen

uniforme Eingabegröße:

- Anzahl der Komponenten zur Beschreibung der Eingabe

- $|(x_1, \ldots, x_n)|_1 = n$ für ganze Zahlen x_i
- $|G|_1 = n + m$ für Graph $G = (V, E)$ mit $\|V\| = n$ und $\|E\| = m$
- $|s_0 s_1 \ldots s_{n-1}|_1 = n$ für Buchstaben s_i

logarithmische Eingabegröße:

- Anzahl der Bits zur Beschreibung von ganzen Zahlen

- $|n|_{\log} = 1 + \lfloor \log_2 n \rfloor$

Beachte:

- $|(2^n)_{10}|_{\log} = |(10^n)_2|_1 = n + 1$ (bei binärer Repräsentierung)
- $|\underbrace{1111\ldots1111}_{2^n\text{-mal}}|_1 = 2^n$ (bei unärer Repräsentierung)

Ressourcen

Laufzeit:

- Anzahl der elementaren Operationen eines Algorithmus (in Java)
- Notation: $t_A(x)$ ist Laufzeit von Algorithmus A auf Eingabe x

Speicherplatz:

- Anzahl zusätzlicher Variablen eines Algorithmus (in Java)
- Notation: $s_A(x)$ ist Speicherplatz von Algorithmus A auf Eingabe x

(worst-case) Komplexität:

- $\text{time}_A(n) \quad =_{\text{def}} \quad \sup \{ \, t_A(x) \mid \; x \text{ ist Eingabe der Größe } |x| = n \; \}$
- $\text{space}_A(n) \quad =_{\text{def}} \quad \sup \{ \, s_A(x) \mid \; x \text{ ist Eingabe der Größe } |x| = n \; \}$

Beachte: Ressourcenaussagen hängen von Eingaberepräsentierung ab

Ressourcenschranken

- **konstant:**
  $\text{time}_A(n), \text{space}_A(n) \in O(1)$
- **logarithmisch:**
  $\text{time}_A(n), \text{space}_A(n) \in O(\log n)$
- **polynomiell:**
  $\text{time}_A(n), \text{space}_A(n) \in O(n^k)$ für geeignetes $k \in \mathbb{N}$
- **exponentiell:**
  $\text{time}_A(n), \text{space}_A(n) \in 2^{O(n^k)}$ für geeignetes $k \in \mathbb{N}$

- **superpolynomiell:**
  $\text{time}_A(n), \text{space}_A(n) \in \Omega(n^k)$ für alle $k \in \mathbb{N}$

Polynomielle Laufzeit

$$\mathrm{FP} =_{\mathrm{def}} \{\ f : M \to N \mid f \text{ wird von Algorithmus in polynomieller Laufzeit berechnet } \}$$

- $f : \mathbb{N} \to \mathbb{N} : x \mapsto 2^x$ gehört zu FP

Fakt: für $f, g \in \mathrm{FP}$ ist $f \circ g \in \mathrm{FP}$

- $f : M \to M'$ werde von A mit $\mathrm{time}_A(n) \leq p_A(n)$ berechnet
- $g : M' \to M''$ werde von B mit $\mathrm{time}_B(n) \leq p_B(n)$ berechnet
- Algorithmus $A \circ B$: wende B auf Eingabe x an mit Ergebnis y und wende dann A auf Eingabe y an
- $f \circ g : M \to M''$ wird von $A \circ B$ mit folgender Laufzeit berechnet:

$$\begin{aligned} \mathrm{time}_{A\circ B}(n) &= O(\mathrm{time}_B(n) + \mathrm{time}_A(\mathrm{time}_B(n))) \\ &= O(\underbrace{p_B(n) + p_A(p_B(n))}_{p_{A\circ B}}) = O(p_{A\circ B}(n)) \end{aligned}$$

Entscheidungsprobleme

- betrachten Teilmengen $L \subseteq M$ der Grundmenge M aller möglichen Eingaben
- **charakteristische Funktion** $c_L : M \to \{0, 1\}$ von L:

$$c_L(x) =_{\text{def}} \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{falls } x \notin L \end{cases}$$

- Algorithmus A **entscheidet** $L \iff_{\text{def}} A$ berechnet c_L

Beachte: Funktion f als Entscheidungsproblem kodierbar mittels

$$L_f =_{\text{def}} \{(x, y) \mid f(x) \geq y\}$$

Polynomielle Entscheidungsprobleme

- $L \in \mathrm{P} \iff_{\mathrm{def}} c_L \in \mathrm{FP}$
- P steht für die Klasse der effizient lösbaren Entscheidungsprobleme

Beispiele für Probleme in P:

- Menge der Palindrome, d.h. $L = \{ w \mid w \in \{0,1\}^* \text{ und } w = w^R \}$
- Menge der matchenden Wortpaare, d.h.
  $L = \{ (s,t) \mid s \text{ kommt in } t \text{ als Teilwort vor} \}$
- Menge der zusammenhängenden Graphen
- Menge der Graphen mit Euler-Kreis
- Menge der planaren Graphen
- Menge der Primzahlen, d.h. $L = \{ x \mid x \in \mathbb{N} \text{ und } x \text{ ist Primzahl} \}$

Nichtdeterministische Polynomialzeit

- $L \in \text{NP} \iff_{\text{def}}$ es gibt ein Polynom p und ein $B \in \text{P}$, so dass für alle $x \in M$ gilt:

$$x \in L \iff \text{ es gibt } y \text{ mit } |y| \leq p(|x|) \text{ und } (x, y) \in B$$

- y heißt **Lösung** (Zeuge, Zertifikat) für x

Anschauung für Probleme in NP:

- Lösungen verifizieren ist „leicht“
- richtige Lösung finden ist „schwierig“

Clique

Problem:	Clique
Eingabe:	ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es $U \subseteq V$ mit $\|U\| \geq k$ und $\{u, v\} \in E$ für alle $u, v \in U$ mit $u \neq v$?

Clique liegt in NP:

- setze $p(n) =_{\text{def}} n$
- setze $B =_{\text{def}} \{(G, k, U) \mid U \text{ ist Clique in } G \text{ und } \|U\| \geq k\}$
- c_B kann in Zeit $O(n^2)$ berechnet werden $(n = |(G, k, U)|_1)$
- $(G, k) \in \text{Clique} \iff$
  es gibt U mit $\|U\| = |U|_1 \leq |(G, k)|_1$ und $(G, k, U) \in B$

Clique

Problem:	Clique
Eingabe:	ungerichteter Graph $G = (V, E)$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es $U \subseteq V$ mit $\|U\| \geq k$ und $\{u, v\} \in E$ für alle $u, v \in U$ mit $u \neq v$?

Wie schnell lässt sich Clique (deterministisch) lösen?

- Durchmusterung: auf Eingabe (G, k) durchlaufe alle Mengen U von k Knoten aus G und teste, ob U Clique in G ist
- Laufzeit hängt wesentlich von der Zahl der Knotenmengen ab: $\binom{n}{k}$
- $\binom{n}{k}$ maximal für $k = \lfloor n/2 \rfloor$: $\binom{n}{\lfloor n/2 \rfloor} = \Theta\left(\frac{2^n}{\sqrt{n}}\right) = \Theta(2^{n-\frac{1}{2}\log n})$

Das P-NP-Problem

Fakt: $\mathrm{P} \subseteq \mathrm{NP}$

- für $L \in \mathrm{P}$ definiere $B =_{\mathrm{def}} L \times \{0,1\}$ und setze $p(n) = 1$
- $B \in \mathrm{P}$, denn: auf Eingabe (x, a) ignoriere a und wende Algorithmus für L auf x an
- Damit: $x \in L \Longleftrightarrow (x, 0) \in B$
- Also liegt L auch NP

Frage: Gilt $\mathrm{P} = \mathrm{NP}$?

- bedeutendstes offenes Problem der Informatik und Mathematik
- Vermutung: $\mathrm{P} \neq \mathrm{NP}$

Reduktionen

- $A \subseteq M$ **in polynomieller Zeit** auf $B \subseteq N$ **reduzierbar** $\iff_{\text{def}}$ es gibt eine Funktion $f : M \to N$ mit $f \in \text{FP}$, so dass für alle $x \in M$ gilt:

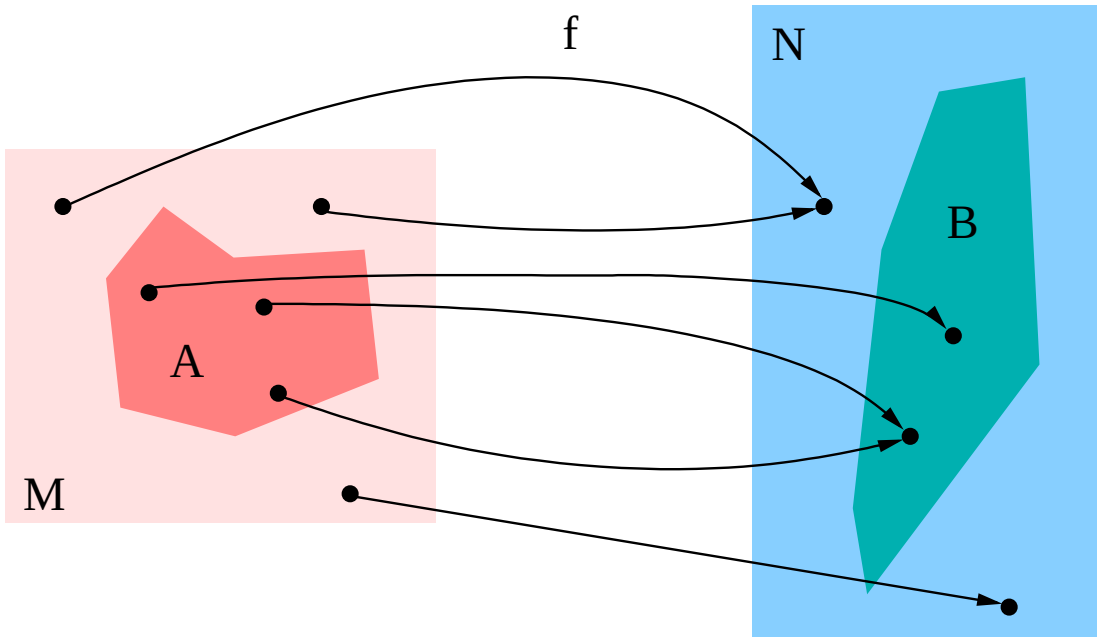
$$x \in A \iff f(x) \in B$$

- f heißt **Reduktionsfunktion** (i.A. weder injektiv noch surjektiv)
- Notation: $A \leq_m^p B$, falls A auf B in polynomieller Zeit reduzierbar
- Notation: $A \equiv_m^p B$, falls $A \leq_m^p B$ und $B \leq_m^p A$

Fakten:

- Relation $\leq_m^p$ ist reflexiv und transitiv
- Relation $\leq_m^p$ ist weder antisymmetrisch noch symmetrisch noch total
- Ist $A \leq_m^p B$ und ist $B \in \text{P}$, so ist $A \in \text{P}$
- Ist $A \leq_m^p B$ und ist $B \in \text{NP}$, so ist $A \in \text{NP}$

Reduktionen



Reduktionsfunktion f für $A \leq_m^p B$

- f bildet M total auf N ab (unabhängig von A und B)
- jedes Element von A muss nach B abgebildet werden
- jedes Element von $\overline{A}$ muss nach $\overline{B}$ abgebildet werden

NP-Vollständigkeit

- L ist **NP-schwer** $\iff_{\text{def}}$ für alle $A \in \text{NP}$ gilt $A \leq_m^p L$
- L ist **NP-vollständig** $\iff_{\text{def}}$ L ist NP-schwer und $L \in \text{NP}$
- NP-vollständige Probleme sind „schwierigste“ Probleme in NP

Fakt: Liegt ein NP-vollständiges Problem in P, so gilt $\text{P} = \text{NP}$

- m.a.W.: Ist $\text{P} \neq \text{NP}$, so liegt kein NP-vollständiges Problem in P
- Nachweis der NP-Vollständigkeit für konkretes Problem schließt polynomiellen Algorithmus für dieses Problem aus (es sei denn $\text{P} = \text{NP}$)
- Methodologie für Nachweis: Wähle irgendein NP-vollständiges Problem, welches dem gegebenen Problem ähnlich ist, und zeige Reduktion auf gegebenes Problem

Frage: Welches ist das erste NP-vollständige Problem?

Erfüllbarkeit für boolesche Formeln

- **konjunktive Normalform** (KNF, CNF) für boolesche Formel H:

$$H = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{m_i} L_{ij},$$

  wobei L_{ij} eine Variable x_ℓ oder eine negierte Variable $\neg x_\ell$ ist
- die L_{ij} heißen **Literale**
- die $\bigvee_{j=1}^{m_i} L_{ij}$ heißen **Klauseln**

- $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ ist KNF mit 4 Klauseln zu je 2 Literalen

- H heißt **erfüllbar** $\iff_{\text{def}}$ es gibt Belegung $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ mit $I(H) = 1$

- $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$ ist nicht erfüllbar

Erfüllbarkeit für boolesche Formeln

Problem:	SATISFIABILITY
Eingabe:	Menge $\{x_1, \ldots, x_n\}$ boolescher Variablen, boolesche Formel $H(x_1, \ldots, x_n)$ in KNF
Frage:	Ist H erfüllbar?

Theorem: [Cook 1971, Karp 1972, Levin 1973]
SATISFIABILITY ist NP-vollständig

Ausgewählte NP-vollständige Probleme

Problem:	Travelling Salesperson
Eingabe:	ungerichteter Graph $G = (V, E)$ mit Kantengewichten $w : E \to \mathbb{R}_+$, Knotenmenge $U \subseteq V$, Zahl $k \in \mathbb{N}$
Frage:	Gibt es einen Kreis in G mit Gesamtgewicht höchstens k, der alle Knoten U besucht?

Problem:	Subset Sum
Eingabe:	Zahlen $a_1, \ldots, a_n, b \in \mathbb{N}$
Frage:	Gibt es $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} a_i = b$?

Liste NP-vollständige Probleme

Problem:	KNAPSACK
Eingabe:	n Gegenstände mit Gewichten $g_1, \ldots, g_n \in \mathbb{N}$ und Werten $v_1, \ldots, v_n \in \mathbb{N}$, Parameter $G, W \in \mathbb{N}$
Frage:	Gibt es $I \subseteq \{1, \ldots, n\}$ mit $\sum_{i \in I} g_i \leq G$ und $\sum_{i \in I} v_i \geq W$?

Problem:	QUADRATIC DIOPHANTINE EQUATIONS
Eingabe:	Zahlen $a, b, c \in \mathbb{N}$
Frage:	Gibt es $x, y \in \mathbb{N}$ mit $ax^2 + by = c$?

Beispielreduktion

Wollen zeigen: $\textsc{Satisfiability} \leq_m^p \textsc{Clique}$

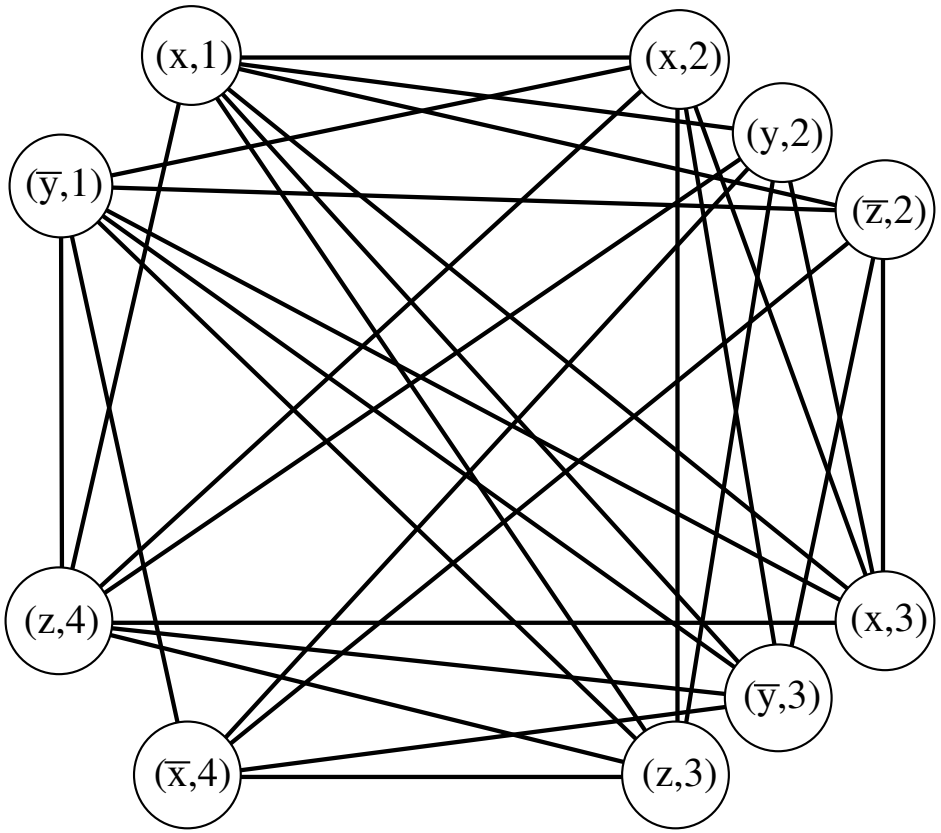
- es sei H eine Formel in KNF mit k Klauseln $C_1, \ldots, C_k$
- konstruieren Graph $G_H = (V_H, E_H)$ wie folgt:

$$V_H =_{\text{def}} \{ (L, i) \mid i \in \{1, \ldots, k\} \text{ und } L \text{ ist ein Literal in Klausel } C_i \}$$
$$E_H =_{\text{def}} \{ \{(L, i), (L', j)\} \mid i \neq j \text{ und } L \neq \neg L' \}$$

- Graph G_H kann in polynomieller Zeit aus Formel H berechnet werden
- definiere Reduktionsfunktion als

$$f(H) =_{\text{def}} (G_H, k)$$

Beispielreduktion



Graph G_H für $H =_{\text{def}} (x \vee \neg y) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee z)$
(beachte: $\overline{x}$ steht für $\neg x$)

Beispielreduktion

Fakt: H erfüllbar $\iff$ G_H enthält Clique der Größe k

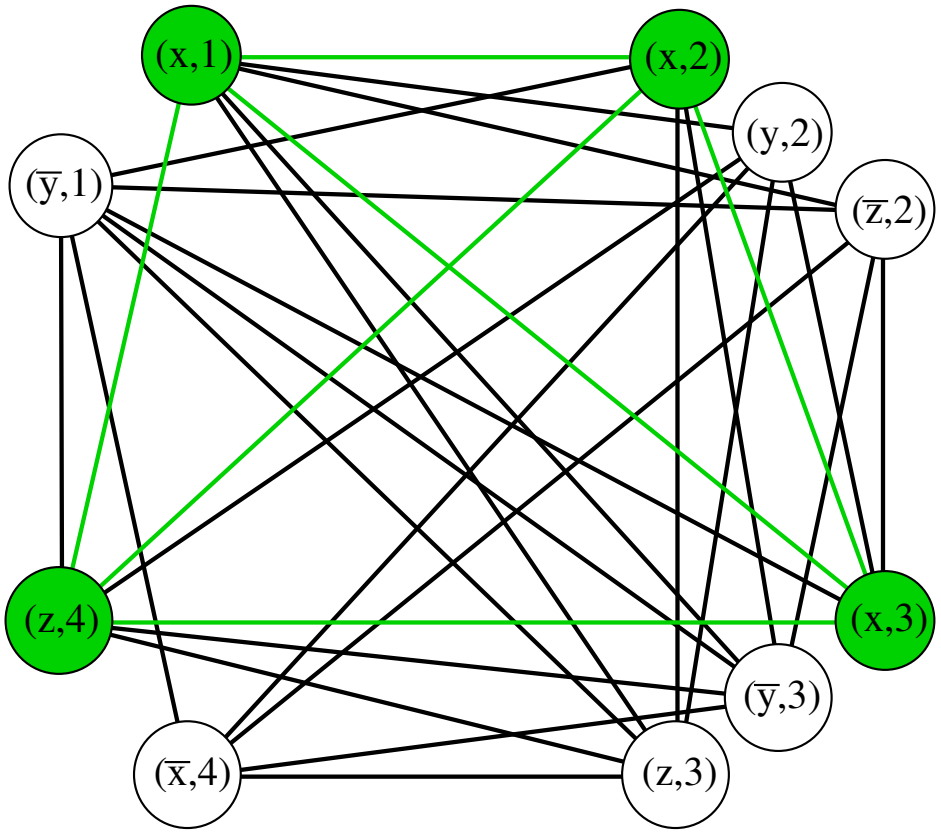
H ist erfüllbar (mittels Belegung I):

- dann gilt $I(C_1) = \cdots = I(C_k) = 1$ und in jeder Klausel wird zumindest ein Literal auf 1 gesetzt, seien $L_1, \ldots, L_k$ solche Literale
- dann gilt $L_i \neq \neg L_j$ für $i \neq j$ (sonst: $1 = I(x) = \text{NOT}\big(I(x)\big) = 0$)
- damit ist $\{(L_1, 1), \ldots, (L_k, k)\}$ Clique der Größe k in G_H

G_H enthält Clique der Größe k (nämlich $U \subseteq V_H$):

- für alle $i \in \{1, \ldots, k\}$ gibt es genau ein L_i mit $(L_i, i) \in U$
- da $L_i \neq \neg L_j$ für $i \neq j$, setze I so, dass $I(L) = 1$ für alle $(L, i) \in U$
- damit: $I(C_1) = \cdots = I(C_k) = 1$, also ist H erfüllbar

Beispielreduktion



Graph G_H für $H =_{\text{def}} (x \vee \neg y) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (\neg x \vee z)$

- $I(x) = I(y) = I(z) = 1$ ist eine erfüllende Belegung für H
- $\{(x,1),(x,2),(x,3),(z,4)\}$ ist eine Clique der Größe 4 in G_H

Nash Equilibrium

Bestimmung von Nash-Gleichgewichten als algorithmisches Problem

Besonderheiten:

- bei zwei Agenten nur rationale Gleichgewichte
- bei drei Agenten irrationale Gleichgewichte möglich
- Ausweg: ε-approximatives Nash-Gleichgewicht

Nash Equilibrium

Problem:	Nash Equilibrium
Eingabe:	Menge $A = \{a_1, \ldots, a_n\}$ von Agenten, Menge $S = \{s_1, \ldots, s_m\}$ von Strategien, Nutzenfunktionen $u_1, \ldots, u_n : S^n \to \mathbb{Z}$, Genauigkeit $\varepsilon > 0$
Ausgabe:	ε-approximatives Nash-Gleichgewicht $(p_1, \ldots, p_n)$, wobei $p_i = (p_{i,1}, \ldots, p_{i,m})$ gemischte Strategie von Agent a_i ist, d.h. a_i kann erwarteten Nutzen additiv nur um ε erhöhen durch Abweichen von gemischten Strategie

Nash Equilibrium

Bestimmung von Nash-Gleichgewichten als algorithmisches Problem

Besonderheiten:

- kein Entscheidungsproblem (damit $\leq_m^p$ nicht anwendbar)
- Ausweg: **metrische Reduktion**

$$f \leq_{\text{met}}^p g \iff_{\text{def}} (\exists h', h'' \in \text{FP})\ (\forall x \in M)\ [\ f(x) = h'(x, g(h''(x)))\ ]$$

- Relation $\leq_{\text{met}}^p$ ist reflexiv und transitiv
- Ist $f \leq_{\text{met}}^p g$ und $g \in \text{FP}$, so ist $f \in \text{FP}$

Vermutungen:

- Nash Equilibrium ist nicht in FP
- Satisfiability $\not\leq_{\text{met}}^p$ Nash Equilibrium (außer NP = coNP)

End of the Line

Problem:	End of the Line
Eingabe:	gerichteter Graph $G = (V, E)$ mit $\deg^+(v), \deg^-(v) \leq 1$ für alle $v \in V$, Knoten $u \in V$ mit $\deg^-(u) = 0$
Ausgabe:	Knoten $w \in V$ mit $\deg^+(w) = 0$

- für jede Eingabe mit Quelle u existiert eine Senke w
- mit üblicher Repäsentierung in FP (Tiefensuche)
- mit kompakter Repräsentierung in FPSPACE: $\|V\| = 2^n$ und E durch boolesche Formeln für pred, succ: $\{0,1\}^n \to \{0,1\}^n$ mit

$$\begin{aligned} \text{pred}(\text{bin}(i,n)) &=_{\text{def}} \text{bin}(j,n) && \text{falls } (v_j, v_i) \in E \\ \text{succ}(\text{bin}(i,n)) &=_{\text{def}} \text{bin}(j,n) && \text{falls } (v_i, v_j) \in E \end{aligned}$$

Brouwer

Problem:	Brouwer
Eingabe:	$F : [0,1]^m \to [0,1]^m$ repräsentiert durch effizienten Algorithmus A_F,
	Lipschitz-Konstante K mit $d(F(x_1), F(x_2)) \leq K \cdot d(x_1, x_2)$ für alle $x_1, x_2 \in [0,1]^m$,
	Genauigkeit $\varepsilon > 0$
Ausgabe:	$x \in [0,1]^m$ mit $d(F(x), x) \leq \varepsilon$

Brouwer

Theorem: [Daskalakis, Goldberg, Papadimitriou 2006]
Nash Equilibrium $\equiv^p_{\text{met}}$ Brouwer $\equiv^p_{\text{met}}$ End of the Line

Beweisschritte:

- Nash Equilibrium $\leq^p_{\text{met}}$ Brouwer
- Brouwer $\leq^p_{\text{met}}$ End of the Line
- End of the Line $\leq^p_{\text{met}}$ Nash Equilibrium

Brouwer $\leq^p_{\text{met}}$ End of the Line

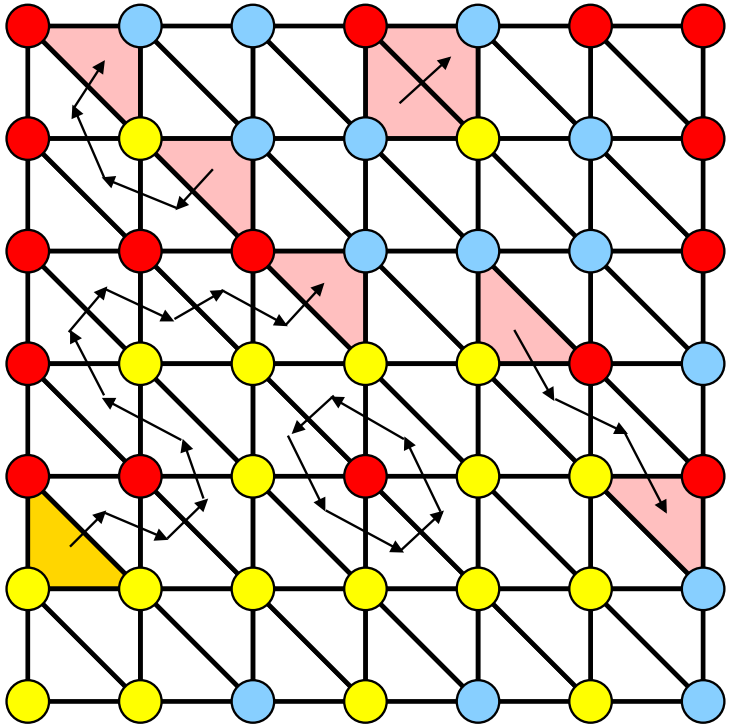
Konstruieren für Brouwer-Instanz Hilfsgraph ($m = 2$):

- legen Gitter (in Abhängigkeit von ε) über $[0,1]^2$
- jeder Knoten x bekommt eine Farbe:

$$\begin{aligned} \text{gelb:} \quad & \angle(F(x) - x, (1,0)) \in [0, \pi/2) \\ \text{blau:} \quad & \angle(F(x) - x, (1,0)) \in [\pi/2, 5\pi/4) \\ \text{rot:} \quad & \angle(F(x) - x, (1,0)) \in [5\pi/4, 2\pi) \end{aligned}$$

- es gilt:
  - keiner der unteren Gitterpunkte ist rot
  - keiner der linken Gitterpunkte ist blau
  - keiner der oberen oder rechten Gitterpunkte ist gelb

Brouwer $\leq^p_{\text{met}}$ End of the Line



- zerlegen $[0,1]^2$ in Dreiecke
- trichromatisches Dreieck ist ε-Fixpunkt
- es gibt stets trichromatische Dreiecke (Sperner-Lemma)