Rheinisch Westfälische Technische Hochschule Aachen
Lehr- und Forschungsgebiet Theoretische Informatik
Seminar Programmverifikation

IP=PSPACE

Joachim Kneis

Gliederung

IP=PSPACE

Teil 0 Einführung und Motivation

Teil 1

- Alternative Beschreibung von PSPACE
- IP $\subseteq$ PSPACE

Teil 2

- QBF
- LFKN Protokoll
- PSPACE $\subseteq$ IP

Teil 0: Einführung / Geschichte

- QBF ist PSPACE vollständig, Stockmeyer und Meyer, *word problems requiring exponential time*, 1973 STOC
- Alternative Beschreibung von PSPACE, Papadimitriou, *games against nature*, 1983
- Einführung von IP, Goldwasser, Micali und Rackoff, *the knowledge complexity of interactive proof-systems*, 1985 STOC
- Einführung von AM Spielen, Babai, *trading group theory for randomness*, 1985 STOC
- IP=PSPACE , Shamir, 1989

Teil 0: Arthur und Merlin (1)

- König Arthur möchte seine 150 Ritter mit 150 Hofdamen verheiraten
- Arthur beauftragt Hofzauberer Merlin passende Paare zu finden
- (P_1: Perfektes Matching in bipartiten Graphen)
- Merlin kann Paare raten, falls Lösung existiert ($P_1 \in NP$)
- Problem: Was tun, falls keine Lösung existiert? (Arthur hat nur polynomielle Zeit)

Teil 0: Arthur und Merlin (1)

- König Arthur möchte seine 150 Ritter mit 150 Hofdamen verheiraten
- Arthur beauftragt Hofzauberer Merlin passende Paare zu finden
- (P_1: Perfektes Matching in bipartiten Graphen)

- Merlin kann Paare raten, falls Lösung existiert ($P_1 \in NP$)
- Problem: Was tun, falls keine Lösung existiert? (Arthur hat nur polynomielle Zeit)

Satz von König: $\exists$ k Ritter die alle die gleichen $k-1$ Hofdamen heiraten wollen $\Leftrightarrow$ ex keine Lösung für P_1

- Merlin rät k Ritter und $k-1$ Hofdamen $(P_1 \in co-NP)$

Teil 0: Arthur und Merlin (2)

- König Arthur möchte seine 150 Ritter so am Tisch plazieren, daß keine Gefechte ausbrechen
- Arthur beauftragt wieder Merlin, einen passenden Sitzplan zu finden
- (P_2: Hamiltonkreis)
- Merlin kann Hamiltonkreis raten, falls Lösung existiert ($P_2 \in NP$)
- Aber: Kein Verfahren bekannt, um Nichtexistenz eines Hamiltonkreises (in NP) zu beweisen

$\Rightarrow$ Arthur sperrt Merlin ins Verlies, bis er eine Lösung findet ...

Teil 0: Motivation

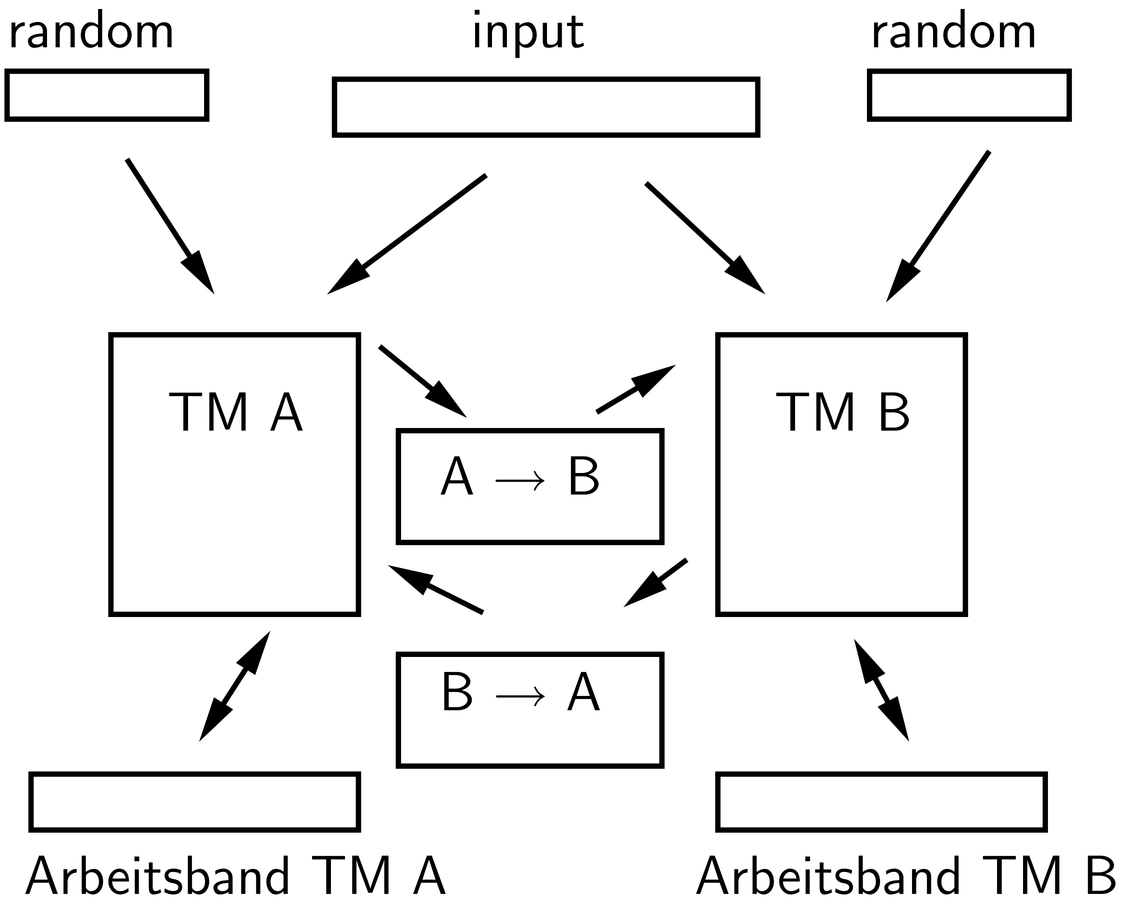
- Idee: IP kleine Erweiterung von NP
- Vermutung: co-NP $\not\subseteq$ AM (=IP)

Teil 0: Motivation

- Idee: IP kleine Erweiterung von NP
- Vermutung: co-NP $\not\subseteq$ AM (=IP)

- Aber: IP=PSPACE
- Folgerung: Unterschied NP $\leftrightarrow$ PSPACE klein

Teil 0: interactive proof-system



Teil 0: Definition IP

Gegeben: eine Sprache L

(A,B) ist IPS für L falls:

- A *prover*: unbeschränkt, nichtdeterministisch
- B *verifier*: polynomiell, deterministisch
- $x \in L$ als Eingabe für (A,B) $\Rightarrow$ B akzeptiert mit W'keit $\geq \frac{2}{3}$
- $x \notin L$ oder anderer Prover $\Rightarrow$ B akzeptiert mit W'keit $< \frac{1}{3}$

Def: $L \in IP \Leftrightarrow$ existiert IPS für L

Teil 0: Definition PSPACE

DEF: $L \in \mathbf{PSPACE} \Leftrightarrow$ existiert TM M mit L(M)=L und L ist polynomiell platzbeschränkt.

Problem: Wie die verschiedenen Modelle vergleichen?

1. Wie Kommunikation und Zufallbits simulieren?

2. Wie jedes Problem in $\mathbf{PSPACE}$ mit IPS lösen?

Teil 1: PPSPACE

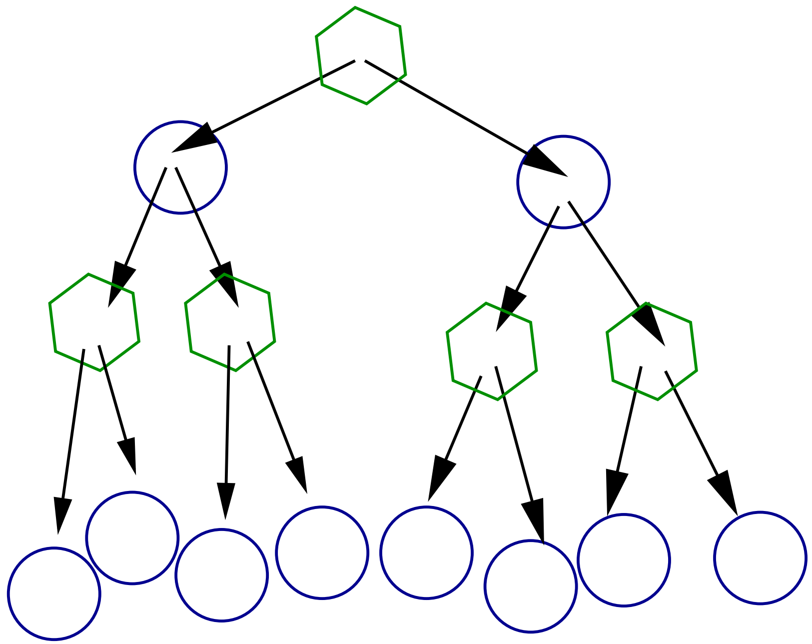
Papadimitriou: probabilistische Turingmaschine S

- ohne Einschränkung: immer nur zwei Wahlmöglichkeiten
- ungerade Schritte: Zufällige Wahl
- gerade Schritte: nichtdeterministische Wahl

S akzeptiert Eingabe $x \Leftrightarrow$ existiert ein akzeptierender (Teil-)Baum im Berechnungsbaum für x auf S

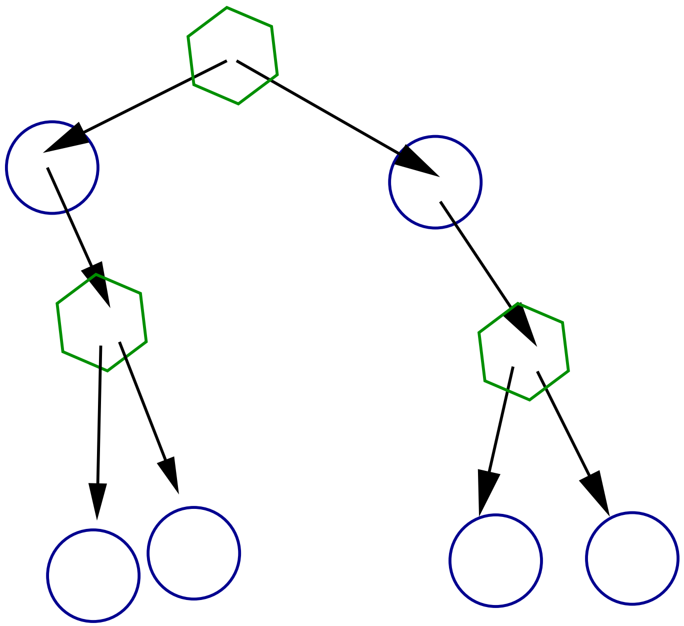
DEF: $L \in \mathbf{PPSPACE} \Leftrightarrow$ existiert STS S mit $L(S) = S$

Teil 1: PPSPACE



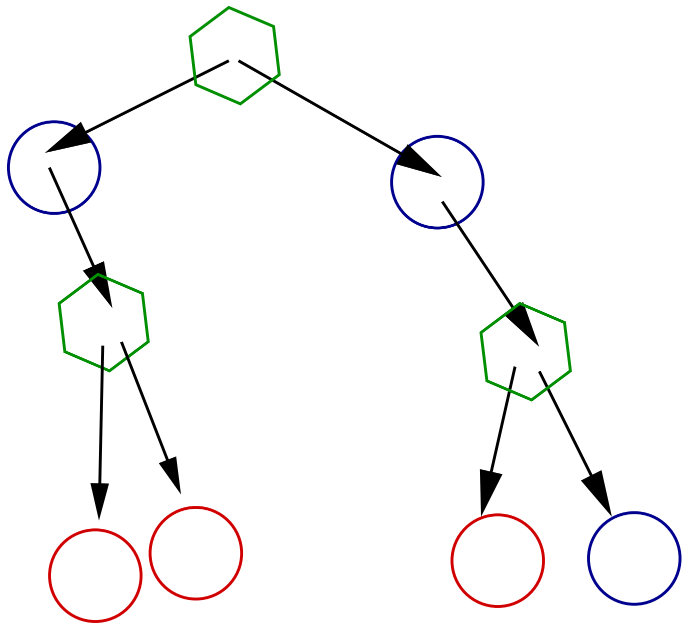
grün: Zufall , blau: Wahlmöglichkeit

Teil 1: PPSPACE



grün: Zufall , blau: Wahlmöglichkeit

Teil 1: PPSPACE



grün: Zufall , blau: Wahlmöglichkeit , rot: akzeptierende Zustände

Teil 1: $\mathbf{PPSPACE} \subseteq \mathbf{PSPACE}$

Gegeben: STM S und Sprache L mit $L(S) = L$

Gesucht: NTM A mit $L(A) = L$, A polynomiell platzbeschränkt

Idee: A simuliert S auf polynomiellen Platz:

- A rät Entscheidungen von S in ungeraden Schritten
- $\Rightarrow A$ erhält Teilbaum T
- A zählt, ob mehr akzeptierende als verwerfende Blätter in T

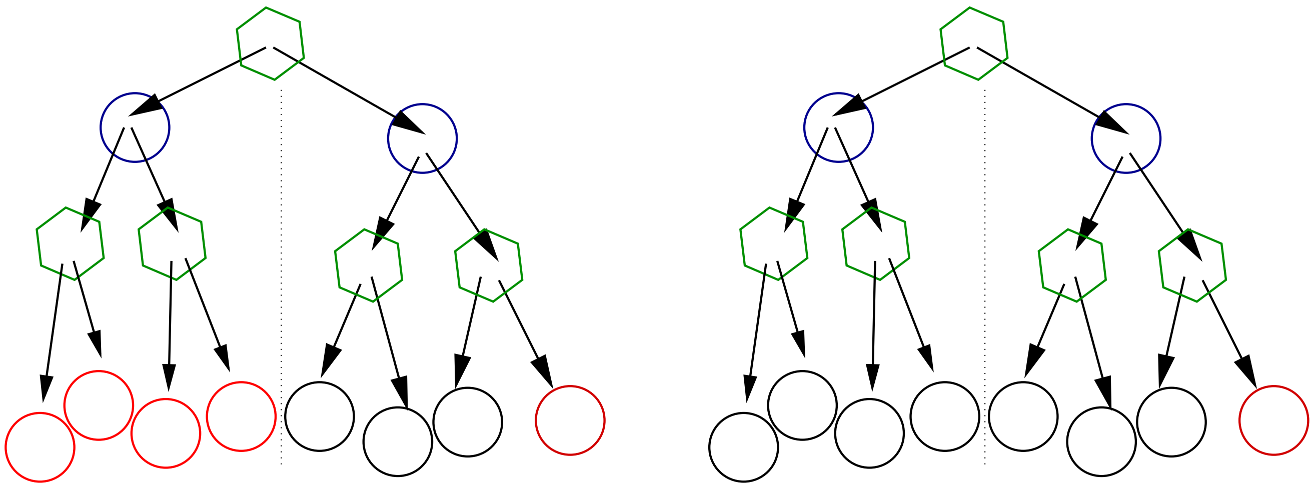
Teil 1: $\mathbf{PSPACE} \subseteq \mathbf{PPSPACE}$

Gegeben: TM A und Sprache L mit $L(A) = L$
Gesucht: STM S mit $L(S) = L$

Konstruktion für S:

- erster Schritt ist zufälliger Schritt
- linker Teilbaum simuliert auf jedem Pfad A
- rechter Teilbaum gleich groß wie linker Teilbaum, enthält genau ein akzeptierendes Blatt

Teil 1: PSPACE $\subseteq$ PPSPACE



akzeptierender Lauf / verwerfender Lauf

Teil 1: $\mathrm{IP} \subseteq \mathrm{PPSPACE}$

Umformulierung PPSPACE:

- *grüne* und *blaue* Knoten $\Rightarrow$ zwei kommunizierende TM (A , B)
- TM B nur abhängig von Zufallsbits
- TM A unbeschränkt
- Eingabe x wird akzeptiert, falls mit W'keit $\geq \frac{1}{2}$ akzeptierender Zustand erreicht wird

Unterschied zu IP:

- TM B abhängig von Zufallsbits und deterministischer Strategie
- Eingabe x wird akzeptiert, falls mit W'keit $\geq \frac{2}{3}$ akzeptierender Zustand erreicht wird

Teil 2: $\mathbf{PSPACE} \subseteq \mathbf{IP}$

Vorgehen:

- $\mathbf{QBF}$ ist $\mathbf{PSPACE}$-vollständig
- Einführung $\mathbf{LFKN}$-Protokoll
- $\mathbf{LFKN}$ und $\mathbf{QBF}$

Teil 2: $\mathbf{QBF}$

Anschaulich: QBF ist Erweiterung von SAT um Quantoren

Formal: $\mathbf{QBF} = \{\varphi \in \Sigma^+_{logic} | \varphi = Q_1 x_1 \dots Q_m x_m \psi, Q_i \in \{\forall, \exists\}, \varphi \text{ erfüllbar}\}$

- $\mathbf{QBF} \in \mathbf{PSPACE}$
  - TM kann alle Möglichkeiten ausprobieren
- (vermutlich) $\mathbf{QBF} \notin \mathbf{NP}$
  - eine Lösung raten reicht nicht
  - bisher kein anderer Ansatz als Ausprobieren bekannt

Teil 2: QBF ist PSPACE-vollständig

Sei M eine TM, polynomiell platzbeschränkt, konstruiere Formel für M wie folgt:

- Variablen für Zustände, Band, Kopf ...
- Formeln für Schritte, Korrektheit, Startzustand $(I(U))$, akzeptierender Zustand $F(U)$, ...
- $A_0(U, V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, ex. Transition $U \vdash V$

Teil 2: QBF ist PSPACE-vollständig

Sei M eine TM, polynomiell platzbeschränkt, konstruiere Formel für M wie folgt:

- Variablen für Zustände, Band, Kopf ...
- Formeln für Schritte, Korrektheit, Startzustand $(I(U))$, akzeptierender Zustand $F(U)$, ...
- $A_0(U,V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, ex. Transition $U \vdash V$
- $A_k(U,V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, $U \vdash^{2^k} V$

Teil 2: QBF ist PSPACE-vollständig

Sei M eine TM, polynomiell platzbeschränkt, konstruiere Formel für M wie folgt:

- Variablen für Zustände, Band, Kopf ...
- Formeln für Schritte, Korrektheit, Startzustand $(I(U))$, akzeptierender Zustand $F(U)$, ...
- $A_0(U, V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, ex. Transition $U \vdash V$
- $A_k(U, V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, $U \vdash^{2^k} V$
  $A_{k+1} = \exists W \forall Y \forall Z \left[(U = Y \wedge W = Z) \vee (W = Y \wedge V = Z)\right] \rightarrow A_k(Y, Z)$

Teil 2: QBF ist PSACE-vollständig

Sei M eine TM, polynomiell platzbeschränkt, konstruiere Formel für M wie folgt:

- Variablen für Zustände, Band, Kopf ...
- Formeln für Schritte, Korrektheit, Startzustand ($I(U)$), akzeptierender Zustand $F(U)$, ...
- $A_0(U,V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, ex. Transition $U \vdash V$
- $A_k(U,V) = 1 \Leftrightarrow$ U,V gültige Konfigurationen, $U \vdash^{2^k} V$
  $A_{k+1} = \exists W \forall Y \forall Z \, [(U = Y \wedge W = Z) \vee (W = Y \wedge V = Z)] \rightarrow A_k(Y,Z)$

Dann gilt: $x \in L(M) \Leftrightarrow \exists U \exists V \, (I(U) \wedge F(V) \wedge A_t(U,V))$

$t =$ Logarithmus $TIME_M(x)$. M platzbeschränkt $\Rightarrow TIME_M(x) \leq 2^{p(x)}$, p Polynom

Teil 2: Permanente

Gegeben: quadratische Matrix A. A_{ij} entsteht aus A durch Streichen der Zeile i und Spalte j.

Def: Permanente $Per(A)$ einer Matrix A:
falls $A \in \mathbb{R}^{2\times 2}$, dann ist $Per(A) = a_{11}a_{22} + a_{12}a_{21}$
falls $A \in \mathbb{R}^{n\times n}$, dann ist $Per(A) = \sum_{i=1}^{n} a_{1i}Per(A_{1,i})$
(Determinante: $(Det(A)) = \sum_{i=1}^{n}(-1)^{i}a_{1i}Det(A_{1,i})$)

Keine andere Berechnung bekannt, daher Zeilenentwicklung notwendig. Berechnung in $\mathbf{PSPACE}$ möglich.

Frage: Existiert IPS für $Per(A)$?

Teil 2: LFKN Idee

Grobe Idee für LFKN Protokoll:

- Merlin veröffentlicht für alle $i = 1, \ldots, n$ Permanente einer $i \times i$ Matrix A_i.
  $A_n = A$
- A_i hängt von A_{i+1} und Zufallsbits von Arthur ab
- Falls $Per(A_i)$ korrekt $\Rightarrow$ $Per(A_{i-1})$ korrekt
- Falls $Per(A_i)$ falsch $\Rightarrow$ $Per(A_{i-1})$ falsch

A_{i+1}: Verschmelzung der einzelnen Summanden $\sum_{i=1}^{n} a_{1i} Per(A_{1,i})$

Teil 2: LFKN Vorbemerkung

Gegeben: A, B $n \times n$-Matrix

Setze $D(x) := x \cdot A + (1-x)B \Rightarrow Per(D(x))$ ist Polynom vom Grad $\leq n$

- Merlin veröffentlicht $\mathcal{A} = Per(A), \mathcal{B} = Per(B)$ und Koeffizienten von $\mathcal{D}(x) = Per(D(x))$
- Falls Merlin betrügt ($\mathcal{A} \neq Per(a)$), dann gilt auch $\mathcal{D}(x) \neq Per(D(x))$
- Arthur wählt $\bar{x} \in X$, weitere Berechnung von $Per(D(\bar{x}))$

Beachte: Merlin sendet immer $Per(D(x))$, Arthur berechnet unabhängig $D(x)$!!!

Teil 2: LFKN Vorbemerkung

Gegeben: A, B $n \times n$-Matrix

Setze $D(x) := x \cdot A + (1-x)B \Rightarrow Per(D(x))$ ist Polynom vom Grad $\leq n$

- Merlin veröffentlicht $\mathcal{A} = Per(A), \mathcal{B} = Per(B)$ und Koeffizienten von $\mathcal{D}(x) = Per(D(x))$
- Falls Merlin betrügt ($\mathcal{A} \neq Per(a)$), dann gilt auch $\mathcal{D}(x) \neq Per(D(x))$
- Arthur wählt $\bar{x} \in X$, weitere Berechnung von $Per(D(\bar{x}))$

Beachte: Merlin sendet immer $Per(D(x))$, Arthur berechnet unabhängig $D(x)$!!!

Wie hoch W'keit, daß Betrug nicht auffällt ?

$Per(D(\bar{x})) = \mathcal{D}(\bar{x})$, aber $Per(D(x)) \neq \mathcal{D}(x)$

$Per(D(x)) - \mathcal{D}(x) \neq 0$, Polynom vom Grad $\leq n$. $\Rightarrow$ maximal n Nullstellen.

$\Rightarrow$ Betrug gelingt mit W'keit $\leq \frac{n}{|X|}$

Teil 2: LFKN Protkoll

Sei: $Per(A) = a_{11}Per(A_{11}) + \ldots + a_{1n}Per(A_{1n})$

- Merlin veröffentlicht alle Permanenten der Gleichung
- Arthur prüft, ob obige Summengleichung erfüllt ist
- Matritzen $A_{11}, \ldots A_{1n}$ werden wie gezeigt zu D verschmolzen
- falls $Per(A)$ falsch $\Rightarrow$ $Per(D)$ mit hoher W'keit falsch
- D ist $(n-1) \times (n-1)$ Matrix, falls n klein genug, testet Arthur, ob Merlin betrogen hat ($Per(D)) \neq \mathcal{A}$)

Aufwand:

- In einer Runde maximal n Verschmelzungen
- Maximal $n-1$ Runden

Einschub: Arthur und Merlin (3)

Gegeben: Graph $G = V, E)$

Valiant: Es existiert Matrix M_G, mit $Per(M) = k \cdot |\{C|C$ ist HC von $G\}|$

M_G kann effizient berechnet werden

$\Rightarrow$

- Arthur und Merlin berechnen M_G
- Berechnung von $Per(M_G)$ mit LFKN Protokoll
- Falls $Per(M_G) = 0$ existiert kein Hamiltonkreis in G

$\Rightarrow$ Merlin ist wieder frei und kann neue Probleme für Arthur lösen

Teil 2: Alternative Sichweise

- Im LFKN Protokoll keine Eigenschaften der Permanente ausgenutzt
- Nur $Per(A) = a_{11}Per(A_{11}) + \ldots + a_{1n}Per(A_{1n})$ verwendet
- Statt $Per(A)$ auch Terme $(\sum(a_{ij} \cdot \sum b_{kl}(\ldots)))$ einsetzbar

Idee für QBF: Wandle Formel φ in Term $\tilde{\varphi}$ um. φ erfüllbar $\Leftrightarrow$ Term $\tilde{\varphi} \neq 0$

Teil 2: Arithmetisierung

Wandle Formel (mit freien Variablen) in Polynom um.

Arithmetisierung von $\varphi \rightarrow \tilde{\varphi}$:

- $\varphi(x) = x_i \rightarrow \quad \tilde{\varphi}(x) = x_i$
- $\varphi(x) = \bar{\psi} \rightarrow \quad \tilde{\varphi}(x) = 1 - \tilde{\psi}$
- $\varphi(x) = \psi_1 \wedge \psi_2 \rightarrow \quad \tilde{\varphi}(x) = \tilde{\psi}_1 \cdot \tilde{\psi}_2$
- $\varphi(x) = \exists x_1, \ldots, x_s \psi(x_1, \ldots, x_s) \rightarrow$
  $\tilde{\varphi}(x) = \sum_{x_1 \in \{0,1\}}, \ldots, \sum_{x_s \in \{0,1\}} \tilde{\psi}(x_1, \ldots, x_s)$
- $\varphi(x) = \forall x_1, \ldots, x_s \psi(x_1, \ldots, x_s) \rightarrow$
  $\tilde{\varphi}(x) = \prod_{x_1 \in \{0,1\}}, \ldots, \prod_{x_s \in \{0,1\}} \tilde{\psi}(x_1, \ldots, x_s)$

Es gilt offensichtlich: φ erfüllbar $\Leftrightarrow \tilde{\varphi} \neq 0$

Teil 2: Erster Schritt im Protokoll

$$
\begin{aligned}
f_0 &:= \sum_{x_1 \in \{0,1\}} \prod_{x_2 \in \{0,1\}} \prod_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(x_1, \ldots, x_n) \\
f_1(x_1) &:= \prod_{x_2 \in \{0,1\}} \prod_{x_3 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(x_1, \ldots, x_n)
\end{aligned}
$$

- Merlin veröffentlicht f_0, und Koeffizienten von Polynom $f_1(x_1)$
- Arthur prüft, ob $f_0 = f_1(0) + f_1(1)$
- Arthur wählt r_1 zufällig, nächste Runde mit $f_1(r_1)$

- Falls Merlin bei f_0 betrügt, dann auch bei $f_1(x_1)$
- Arthur kann $f_1(x)$ konstruieren, aber nicht auswerten

Teil 2: allgemeiner Schritt im Protokoll

$$
\begin{aligned}
f_0 &:= \sum_{x_1\in\{0,1\}} \prod_{x_2\in\{0,1\}} \prod_{x_3\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} \tilde{\varphi}(x_1,\ldots,x_n) \\
f_1(x_1) &:= \prod_{x_2\in\{0,1\}} \prod_{x_3\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} \tilde{\varphi}(x_1,\ldots,x_n) \\
f_i(x_1,\ldots,x_i) &:= \sum_{x_{i+1}\in\{0,1\}} \prod_{x_{i+2}\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} \tilde{\varphi}(x_1,\ldots,x_n)
\end{aligned}
$$

- Arthur hat $f_i(r_1,\ldots,r_i)$ in Runde vorher berechnet, Merlin sendet Koeffizienten von Polynom $f_i(r_1,\ldots,r_i,x_{i+1})$
- Arthur prüft, ob $f_i = f_{i+1}(0) + f_{i+1}(1)$
- Wenn f_i klein, prüft Arthur direkt
- Sonst wählt Arthur r_{i+1} zufällig, nächste Runde mit $f_{i+1}(r_1,\ldots,r_{i+1})$

Teil 2: Auftretende Probleme

Arthur muss $f_i(y)$ an Stelle 0 und 1 berechnen, also z.B:

$$f_i(r_1, \ldots, r_{i-1}, y) \;\; := \;\; \prod_{x_{i+1} \in \{0,1\}} \prod_{x_{i+2} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} (x_n \cdot y)$$

Probleme:

1. f_i in beliebiger Form

2. Grad f_i kann exponentiell sein

Teil 2: Auftretende Probleme

Arthur muss $f_i(y)$ an Stelle 0 und 1 berechnen, also z.B:

$$f_i(r_1, \ldots, r_{i-1}, y) \;:=\; \prod_{x_{i+1}\in\{0,1\}} \prod_{x_{i+2}\in\{0,1\}} \cdots \sum_{x_n\in\{0,1\}} (x_n \cdot y)$$

Probleme:

1. f_i in beliebiger Form
2. Grad f_i kann exponentiell sein

Lösung:

1. Merlin sendet f_i in normierter Form
2. keine Variable oft hinter einem Allquantor

Teil 2: Vorberechnung

Ziel: Keine Variable mehr als zweimal hinter Allquantor

$$\varphi(x_1, \ldots, x_n) \;=\; \exists x_1 \exists x_2 \forall x_3 \psi(x_1, \ldots x_n)$$

Teil 2: Vorberechnung

Ziel: Keine Variable mehr als zweimal hinter Allquantor

Idee: Substitution von Variablen

$$
\begin{aligned}
\varphi(x_1, \ldots, x_n) &= \exists x_1 \exists x_2 \forall x_3 \psi(x_1, \ldots x_n) \Rightarrow \\
\varphi'(x_1, \ldots, x_n) &= \exists x_1 \exists x_2 \forall x_3 \exists (x'_1, \ldots, x'_n)(x_1 = x'_1) \wedge \ldots \wedge (x_n = x'_n) \wedge \psi(x'_1, \ldots x'_n)
\end{aligned}
$$

Klar: $\varphi \equiv \varphi'$

Aufwand: Jeder Allquantor fügt n neue Variablen ein $\Rightarrow |\varphi'| \leq |\varphi|^2$

Teil 2: Vorberechnung

Ziel: Keine Variable mehr als zweimal hinter Allquantor

Idee: Substitution von Variablen

$$\begin{aligned}\varphi(x_1,\ldots,x_n) &= \exists x_1 \exists x_2 \forall x_3 \psi(x_1,\ldots x_n) \Rightarrow \\ \varphi'(x_1,\ldots,x_n) &= \exists x_1 \exists x_2 \forall x_3 \exists (x_1',\ldots,x_n')(x_1 = x_1') \wedge \ldots \wedge (x_n = x_n') \wedge \psi(x_1',\ldots x_n')\end{aligned}$$

Klar: $\varphi \equiv \varphi'$

Aufwand: Jeder Allquantor fügt n neue Variablen ein $\Rightarrow |\varphi'| \leq |\varphi|^2$

Kodierung von $(x_i = x_i')\psi$:

$$(x_i = x_i')\psi \rightarrow [(x_i x_i') + (1 - x_i)(1 - x_i')]\tilde{\psi}$$

Teil 2: Vorberechnung

Letztes Problem: Auftretende Werte exponentiell

Lösung: modulo p rechnen

- Merlin sendet Primzahl p + Beweis, daß p Primzahl
- Alle Berechnungen in F_p

$\Rightarrow$ PSPACE $\subseteq$ IP

$\Rightarrow$ **PSPACE = IP**

Teil 2: Zusammenfassung

- IP $\subseteq$ PSPACE
  - probabilistische Turingmaschinen
  - PPSPACE
- PSPACE $\subseteq$ IP
  - QBF
  - LFKN Protokoll
  - Arithmetisierung

Zusammenhang mit Programmverifikation

Beispiel: Permanente einer Matrix A

- instance checking: LFKN Protokoll für A gegen sich selbst
- mehrfache Anwendung: Programm korrekt in mehr als $1 - n^{-2}$ Fällen
- self-correcting:
  1. Berechne $Per(A + iB)$ für $i = 1, \ldots, n+1$, B zufällig
  2. Interpolation von $Per(A + xB)$
  3. Berechnen von $Per(A + 0B)$

FINIS