Uebungsstunde Theoretische Informatik

Thomas Briner, brinerth@iiic.ethz.ch, http://briner.computerscience.ch/ti

20.6.03

1 Nachbesprechung Uebung 5

1.1 Kernaufgaben fuers Verstaendnis:

- HW 6.5 c) Nicht abgeschlossen unter Komplement
- HW 6.5 d) L und A*-L ist erkennbar $\rightarrow$ L ist entscheidbar

1.2 HW 6.5 a)

Wichtig: "Definition heisst nicht: Ein Wort muss in endlicher Schrittzahl entschieden werden", oder so, sondern:

decidable: A language L over an alphabet A is decidable iff there is a Turing machine M(L) that accepts the strings in L and rejects the strings in the complement A*-L.
Beispiele aus Antworten:

- L = $\{w \mid w = 0^*1\}$
- L = $\{w \mid w = 0^n1\}$
- L = $\{w \mid w = \pi\}$

sind alle decidable.

1.3 HW 6.5 b)

Am verstaendlichsten und unverfaenglichsten sind wohl graphische Loesungen.

1.4 HW 6.5 c)

gleich wie b) ausser:

- Union muss jetzt parallel sein
- Komplement geht nicht. Grund: accept und reject-Lampe umgekehrt verdrahten funktioniert nicht, weil auch loop auftreten kann. Was soll mit einem solchen Wort im invertierten Fall passieren?

1.5 HW 6.6, Vergleich der beiden Definitionen von computable

Reelle Zahl ist berechenbar falls

1. Start mit leerem Band, schreibt Folge der Ziffern raus, muss nicht enden
2. Start mit k auf dem Band, schreibt k-te Zahl raus und endet dann

Beweis in beide Richtungen:

1. Aus Def 1→ Def 2:
   Zweites Band mit k, lasse Maschine auf Band 1 laufen bis k-te Ziffer erreicht, Rest loeschen
2. Aus Def 2→ Def 1:
   Zweites Band leer, iteriere ueber k bis unendlich, kopiere jedes Mal k-te Ziffer auf erstes Band

2 Vorbesprechung

2.1 Problem L ist NP-hart:

1. Jedes andere Problem L' in NP kann darauf reduziert werden (= kann auf Umweg ueber dieses Problem L geloest werden).

$$L' \leq_p L$$

2.2 Problem L ist NP-vollstaendig

1. Jedes andere Problem L' in NP kann darauf reduziert werden (= kann auf Umweg ueber dieses Problem L geloest werden).

$$L' \leq_p L$$

2. L ist in NP

2.3 Wie kann NP-Completeness fuer ein Problem L am besten bewiesen werden

(Der Beweis ueber "jedes andere Problem" ist unguenstig!)
Suche ein Problem L' von dem du weisst, dass es NP-vollstaendig ist (d.h. jedes andere Problem in NP kann auf L' reduziert werden) Reduziere dann L' auf L und zeige, dass die Umwandlungsalgorithmen in polynomieller Zeit laufen.
Nun koennte auch jedes Problem L" polynomiell auf L reduziert werden, da die Addition von zwei Polynomen immer noch ein Polynom sein muss.

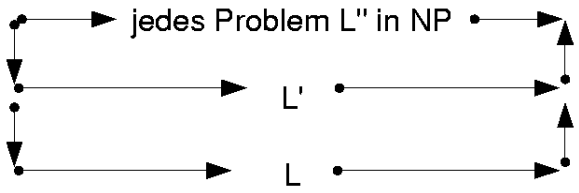

Figure 1:

3 Beispiel: Beweis der NP-Vollstaendigkeit von k-Vertex Cover

Eigentlich wollen wir:

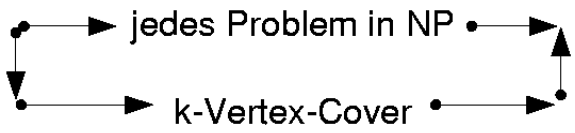

Figure 2:

Ein moeglicher Weg ist:

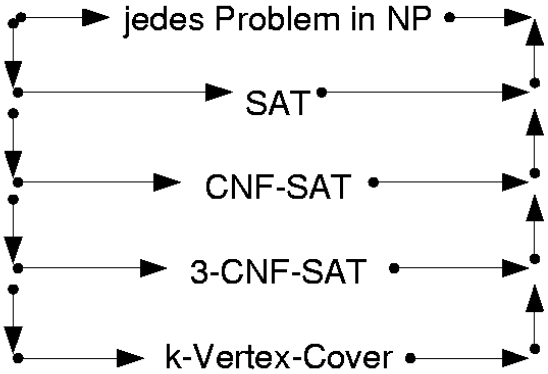

Figure 3:

3.1 Reduktion von 3-CNF-SAT auf k-Vertex Cover

- 3-CNF z.B. $(x \vee y \vee \neg z) \wedge (\neg x \vee y \vee z)$
- Anzahl Vars = n
- Anzahl Clauses = m

3.1.1 Umsetzung der 3-CNF in einen Graph:

- Jede Clause als vollstaendiger kleiner Subgraph,
- Fuer jede Variable: Je einmal positiv und einmal negiert
  $\rightarrow$ 3*m + 2*n Knoten
- Jetzt Vertex Cover loesen:
  - Fuer Clause intern alles bewachen: braucht sicher 2
  - Fuer Var intern alles bewachen: braucht sicher 1
  - Falls 2*m + n genug ist $\rightarrow$ Existiert 2*m+n Vertex Cover $\rightarrow$ 3CNF ist sat

Intuition: Annahme: Ist erfuellbar und ich kenne die erfuellende Belegung.
Nimm in jeder Clause, die 2 anderen Knoten und nicht denjenigen, der gebraucht wird, damit die Clause true wird. Falls es mehrere hat, die die Clause true machen wuerden, entscheide dich fuer eine.
Jetzt sind bereits alle Clause-internen Kanten bewacht und alle Verbindungen Variable - Clause, falls sie zu Clause-Knoten gehen, die nicht fuer die Erfuellbarkeit gebraucht werden. Nun waehle ich bei den Variablen-Knoten immer denjenigen fuers Vertex-Cover, der in der erfuellenden Belegung drin ist. Damit bewache ich erstens die Kante zwischen den Variablen und zweitens die Kante von den erfuellenden Knoten in den Clausen zu den Variablen Knoten. Damit sind alle Kanten bewacht.
Wenn ich keine $2 * m + n$ Vertex Cover finde, kann die 3-CNF nicht erfuellbar sein.

4 Ist k-Clique NP-complete?

Zwei moegliche Wege:

- $3 - CNF \leq_p k - Clique$
- $VertexCover \leq_p k - Clique$

4.1 Zusammenhang Clique - Vertex Cover

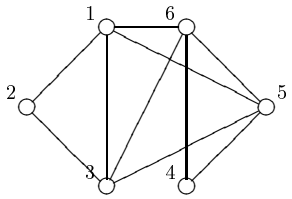

Figure 4: Graph G=(V,E)

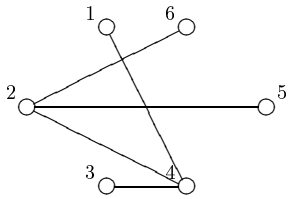

Figure 5: Graph G'=(V,$\bar{E}$)

Betrachte eine Clique in G. Wie viele der beiden Endknoten der Kanten in G' koennen in der Clique sein? 0 oder 1 (Wenn beide waeren, gaebe es in G keine Verbindung zwischen den beiden und es waere dann keine Clique).
$\rightarrow$ mindestens einer der Knoten ist nicht in der Clique dabei $\rightarrow$ Wenn ich alle die Knoten, die nicht in der Clique sind fuer das Vertex Cover waehle, ist jede Kante in $\bar{E}$ bewacht.
Um ein k-Vertex-Cover zu finden, kann ich das (n-k)-Clique Problem loesen und dann die Knoten nehmen, die nicht in der Clique enthalten sind.
Da k-Clique in NP liegt (Bedingung 2) ist damit klar, dass auch k-Clique NP-vollstaendig ist.

5 Umgangssprachliche Erklaerungen zu einigen NP-completen Problemen:

5.1 3-Sat:

Gegeben ist eine boolsche Formel, wobei jede Clause genau 3 Litterale enthaelt. Ein Litteral ist entweder eine Variable oder eine negierte Variable.

5.2 3-CNF-SAT:

Gegeben ist eine boolsche Formel in konjunktiver Normalform. $\phi = (x \vee y) \wedge (\neg y \vee z)$

5.3 Independent set:

Graph G=(V,E), waehle independent set S Teilmenge von V Keine 2 Knoten e S haben eine Kante in E

5.4 k-Set Cover:

Menge von Punkten U gegeben, viele Teilmengen von U $S_1 \dots S_n$. Waehle eine Menge von k Sets aus $S_1 \dots S_n$ so aus, dass alle Punkte von U in mindestens einem Set enthalten sind.

5.5 k-Vertex Cover:

Graph G=(V,E). Waehle ein Set L von k Knoten, so dass fuer jede Kante e E gilt: mindestens einer der beiden Endknoten ist in L enthalten.

5.6 k-Clique:

Graph G=(V,E). Waehle ein Set L von k Knoten, so dass der Subgraph aus diesen k Knoten ein vollstaendiger Graph ist. Fuer jedes Paar von Knoten e L muss also die entsprechende Kante in E enthalten sein.

5.7 und zum Schluss

Und hier gibts noch eine ganze Menge von weiteren beschriebenen Problemen:
http://www.csc.liv.ac.uk/ ped/teachadmin/COMP202/annotated_np.html