Prof. Dr. R. Schrader
Birgit Engels [engels@zpr.uni-koeln.de]

WS 2006/2007

Theoretische Informatik

9. Übung, Abgabe Mittwoch, 20.12.2006

Aktuelle Informationen bezüglich der Vorlesung und der Übungen finden sich unter :
http://www.zaik.uni-koeln.de/AFS/teachings/courses/ThInf/index.html und
http://www.zaik.uni-koeln.de/AFS/teachings/courses/ThInf/uebungen.html

Aufgabe 34 (Zugriffssrechte bei Allgemeinen Parallelen Systemen — 4 Punkte):

a) Wie lassen sich exklusive Lese-/Schreibrechte für parallele Prozessoren garantieren ?

b) Welche unerwünschte Situation kann dadurch auftreten ?

Aufgabe 35 (Zeitersparnis durch PRAMs — 8 Punkte):
Nehmen Sie für die Aufgabenteile a) und b) jeweils eine ausreichende Anzahl von parallelen Recheneinheiten (RAMs) an und formulieren Sie die entsprechenden Programme in sinnvollem Pseudo-Code, so dass die Parallelausführung erkennbar ist.

a) Implementieren Sie ein paralleles Programm, welches in konstanter Zeit das Minimum eines unsortierten Arrays a beliebiger Länge n ausgibt.

b) Implementieren Sie ein paralleles Programm, welches in konstanter Zeit ein unsortiertes Array a beliebiger Länge n sortiert.

c) Nehmen Sie nun an, dass die Anzahl der zur Verfügung stehenden parallelen Recheneinheiten auf k begrenzt ist. Wie ändern sich die Laufzeiten Ihrer Programme aus den Aufgabenteilen a) und b) (*Lemma von Brent*)?

Aufgabe 36 (NC — Nick's Class — 4 Punkte):
In der Vorlesung wurde die Komplexitätsklasse $NC = PRAM(poly, poly(log))$ eingeführt, wobei $PRAM(poly, poly(log))$ die Klasse aller Probleme darstellt, die von polynomiell vielen parallelen RAMs entschieden werden können, wobei die Laufzeit durch ein Polynom im Logaritmus der Länge n der Eingabe beschränkt ist. Es wird allerdings auch folgende Definition genutzt:

Bitte wenden !

Die Klasse **NC** ist die Klasse der (Entscheidungs-)Probleme, deren charakteristische Funktion von einem boolschen Schaltnetz S polylogaritmischer Tiefe ind der Länge n der Eingabe, das aus nur aus Gattern mit maximal 2 Eingängen besteht, berechnet werden kann.

Zum Verständnis: Unter einem solchen Schaltnetz S verstehen wir eine Anordnung von Bausteinen (*Gattern*) mit jeweils 2 Eingängen und einem Ausgang. Die Eingänge der Gatter können mit "1" oder "0" belegt sein (technisch: "Strom" oder "Kein Strom" :-), sie entsprechen also boolschen Variablen. Der Ausgang des Gatters entspricht je nach Typ des Gatters einer boolschen Verknüpfung der beiden Variablen. Wir erlauben dabei Gatter vom Typ **AND**, **OR** (mit je 2 Eingängen) und **NOT** (mit nur einem Eingang). Mit diesen 3 Gatter-Typen kann jede boolsche Funktion f berechnet werden. Dabei werden allerdings mehrere Gatter hintereinandergeschaltet (der Ausgang eines Gatters dient wieder als Eingabe für ein nächstes Gatter). Ein solches Schaltnetz N kann man als Binärbaum $B(N)$ betrachten, wobei das letzte Gatter, dessen Ausgang der gewünschten boolschen Funktion $f(N)$ entspricht, die Wurzel von $B(N)$ darstellt. ($B(N)$ kann unvollständig sein, falls **NOT** verwendet werden.) Die Höhe von $B(N)$ entspricht dabei der Tiefe des Schaltnetzes N. Die in der Definition betrachteten Schaltnetze S sollen also in der oben motivierten Weise eine Tiefe besitzen, die durch ein Polynom im Logarithmus der Länge der Eingabe beschränkt sind.

Um die Äquivalenz beider Definitionen zu zeigen, muss zum einen von einem Schaltnetz S (wie oben definiert) jede beliebige $PRAM(poly, poly(log))$ simuliert werden können und zum anderen muss für jedes solche S, die berechnete die Funktion f mit $f = S(x_1, \ldots, x_n)$ ($x_i \in \{0, 1\}$ sind boolsche Variablen) von einer $PRAM(poly, poly(log))$ berechnet werden können. Erstere Simulation ist in Analogie zum Satz von Cook (Mittels des *SAT*-Problems kann eine Turingmaschine simuliert werden) relativ aufwendig. Zeigen Sie daher lediglich:

Die Funktion $f = S(x_1, \ldots, x_n)$ wird von einem Schaltnetz S polylogaritmischer Tiefe aus boolschen Gattern mit je 2 Eingängen berechnet.
$\Rightarrow$
$\exists$ PRAM, die f in polylogaritmischer Zeit mit polynomieller Anzahl von Prozessoren berechnet.

Aufgabe 37 (NC und P-Vollständigkeit — 4 Punkte):
Nach dem Lemma von *Brent* (siehe Vorlesung) gilt für die Klasse P: $P = PRAM(poly, poly)$. Daher gilt $NC = PRAM(poly, poly(log)) \subseteq PRAM(poly, poly) = P$ und wir haben eine Klasse von Problemen kennengelernt, die eventuell "leichtere" Probleme umfasst als P. Daher können wir nun auch sinnvoll definieren:

Ein (Entscheidungs-)Problem L ist **P-Vollständig**, falls gilt:

$$\forall L' \in P : L' \leq_{NC} L, \; wobei$$

$$L' \leq_{NC} L \Leftrightarrow \exists f : x \in L' \Leftrightarrow f(x) \in L$$

Und f kann mit einer $PRAM(poly, poly(log))$ berechnet werden (**NC-Reduktion**)!

Bitte wenden !

In Aufgabe 36 wurden Schaltnetze S mit polylogarithmischer Tiefe vorgestellt. Das Problem, ob eine Funktion $f(N)$ für Schaltnetze N polynomieller Tiefe den Wert "1" oder "0" annimmt (**C**ircuit **V**alue **P**roblem) ist P-Vollständig.
Begründen Sie, warum das **CVP** für Schaltnetze, die ausschließlich aus **NOR**- Gattern (mit der Funktionalität $\mathbf{NOR}(x, y) = \neg x \wedge \neg y$) bestehen, ebenfalls P-vollständig ist!
(Eine formale Reduktion ist nicht nötig.)