Das GAP-Theorem

Dreifach logarithmischen Platzbedarf gibt es nicht

Tobias Zech

20. September 2010

Zusammenfassung

Nicht-reguläre Sprachen haben keine obere Schranke für ihren Platzbedarf. Im Gegenteil existiert sogar eine untere Schranke für den Platzbedarf in Abhängigkeit von der Länge des Eingabewortes. Es wird gezeigt, dass diese für Einweg-Deterministische Turing Maschinen (DTM'n) logarithmisch und für Zweiwege-DTM'n doppelt logarithmisch von der Wortlänge abhängt.

1 Einführung

Reguläre Sprachen (REG) werden von endlichen Automaten (FA) erkannt. Demzufolge haben sie gar keinen oder nur konstanten Platzbedarf. (s. [Tan10] Kap.4-8,22.3). Nicht-reguläre Sprachen brauchen aber im Allgemeinen, abhängig von der Eingabelänge, mehr Platz. So braucht die Sprache $\{0^m1^n|m = n\}$ zum Beispiel logarithmisch viel Platz. Der Grund dafür ist, dass eine *deterministische* Turingmaschine (DTM) mittels eines Binärzählers erst die Nullen hoch und dann die Einsen runterzählt. Damit haben wir ein Beispiel für eine Sprache in der Platzklasse DSPACE[log n] gefunden.

Die Frage ist nun, ob es Sprachen gibt dessen Platzbedarf noch schwächer von der Eingabelänge abhängt ohne eine obere Schranke, wie REG, zu besitzen. Die Frage hat zwei Antworten: Für Einweg-DTM'n ist log n die untere Schranke für die Längenabhängigkeit, für Zweiwege-DTM's ist es log log n. Hieraus begründet sich der Titel der die Existenz von dreifach logarithmischem Platzbedarf verneint, es gibt eine GAP für die Platzfunktionen.

2 Endlichkeit führt zu Periodizität

Bevor nicht-reguläre Sprachen betrachtet werden, wird hier kurz der reguläre Fall diskutiert. Dieser entspricht genau den Sprachen, die von endlichen Automaten erkannt werden (s. [Tan10] Kap 7.3). Die Endlichkeit ihres Eingabealphabets zusammen mit nur endlich vielen möglichen internen Zuständen, führt dazu, dass es nur endlich viele verschiedene Zustandsübergänge gibt. Daraus folgt, dass falls der Automat für das selbe Eingabezeichen an verschiedenen Stellen des Wortes im selben internen Zustand ist, dann ändert das Teilwort zwischen diesen beiden Zeichen nichts an der Erkennung des Wortes. Dieses Teilwort erzeugt also ein periodisches Verhalten des Automaten. Ganz allgemein führt die Endlichkeit von *System*-Zuständen eines Systems mit deterministischer Zustandsänderung zu Periodizität.

Dieser Zusammenhang wird ausgenutzt um die maximale Länge eines Wortes, das kein periodisches Verhalten erzeugt, für festen Platzbedarf über die Beschr̈ankung der *System*-Zustände zu bestimmen. Für reguläre Sprachen liegt diesem Zusammenhang dem Pumping-Lemma (s. [Tan10] Kap. 5) zugrunde.

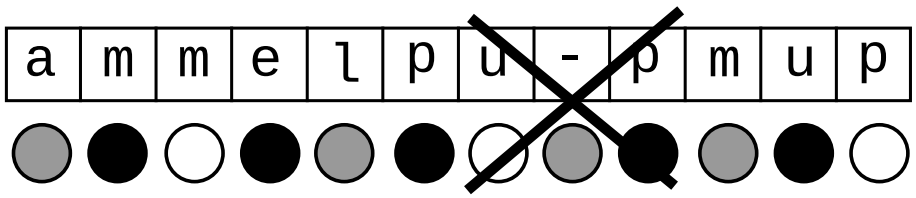

Abbildung 1: DTM mit getrenntem Eingabe- und Arbeitsband

3 Turing Maschine für L=DSPACE[log n]

Um DTM'n mit logarithmischem Platzbedarf zu untersuchen ist es zweckgemäß das Eingabeband vom Arbeitsband zu trennen. Ansonsten skaliert der Platzbedarf schon vor der Berechnung linear mit der Eingabe. Eine klare Trennung zwischen Eingabe und Platzbedarf der Verarbeitung ist dann schwierig. Das Eingabeband ist mit einem Lesekopf ausgestattet, auf das Arbeitsband kann sowohl geschrieben als auch davon gelesen werden. Zusammen mit dem internen Zustand bestimmen die Zeichen unter den beiden Köpfen den nächsten Schritt in der Berechnung. Mögliche Aktionen sind, das Schreiben auf das Arbeitsband an der Position des Kopfes, das Ändern des internen Zustandes und das Bewegen der Köpfe in beide Richtungen. Eingabe- und Arbeitsalphabet, sowie die Länge des Eingabewortes und die Menge der internen Zustände sind endlich. Die internen Zustände besitzen zwei disjunkte Teilmengen, die der akzeptierenden und der ablehnenden Zustände. Ein Wort wird akzeptiert, beziehungsweise abgelehnt, wenn sich die Maschine beim Überschreiten des Endmarkers in dem entsprechenden Zustand befindet. Die Sprache, die wir betrachten, wird von dem DTM erkannt, das heißt, dass jedes Wort der Sprache akzeptiert und der Rest abgelehnt wird. Demnach hält die Maschine für alle Eingaben.

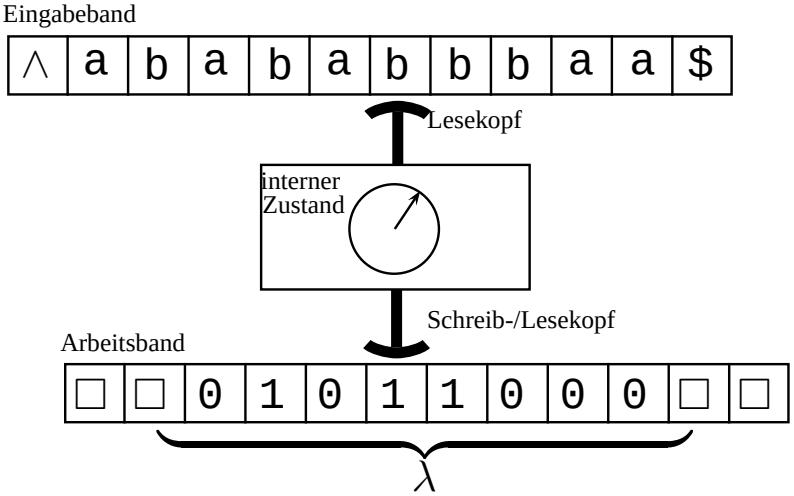

Abbildung 2: DTM mit getrenntem Eingabe- und Arbeitsband

Das Arbeitsband wird zu Beginn als unbeschrieben angenommen. Der Platzbedarf (s. λ in Abb.2) ist dann definiert als die Anzahl beschriebener Zellen auf dem Arbeitsband.

4 Einweg-DTM

Die Einschränkung auf Einweg-DTM'n, dessen Eingabeband sich nur in eine Richtung bewegen kann, vereinfacht die Identifizierung der *System*-Zustände. Die Abarbeitung des Wortes entspricht dessen Zeichenfolge. Kein Zeichen wird zweimal gelesen.

Um eine untere Schranke für den Platzbedarf zu bestimmen, wird das Problem in die Frage nach einer oberen Schranke für ein Wort von minimaler Länge bei gegebenem Platzbedarf umformuliert. Um die *System*-Zustände zu finden, werden die internen Zustände zusammengefasst zur *Speicher-Konfiguration*. Diese ist ein Tripel aus dem internen Zustand, dem String auf dem Arbeitsband und der Position des Lese-/Schreibkopfes des Arbeitsbands. Als *System*-Zustand wird die Kombination aus Speicher-Konfiguration und Zeichen unter dem Lesekopf des Eingabebands definiert. Dies macht Sinn, denn gleiche *System*-Zustände führen zu gleichem Verhalten. Da der Automat das Eingabewort nur in einer Richtung durchläuft, gehört zu jedem Zeichen genau eine Speicher-Konfiguration.

Auf der Suche nach dem kürzesten Wort w bei Platzbedarf genau λ, stellt sich heraus, dass periodisches Verhalten die Minimalität verletzt. Angenommen in $w = x_1 \dots x_r \dots x_s \dots x_n$ tritt ein Zeichen doppelt ($x_r = x_s$) auf und beim Verarbeiten ist die DTM in der gleichen Speicher-Konfiguration. Dann kann das Wort um das Teilwort zwischen den gleichen Zeichen $x_{r+1} \dots x_s$ verkürzt werden, ohne das Verhalten der DTM zu ändern. Da das neue Wort w' aber küerzer als w ist, muss es schon in weniger Platz als w anhalten. Alle Haltekonfigurationen von w brauchen aber genau λ Platz. Die Speicherkonfigurationen,die w' durchläuft, sind eine Teilmenge derer, die w' durchläuft, denn die DTM verhält sich ja gleich. Demnach hält auch w' in genau λ Platz, was im Widerspruch zur minimalen Länge von w steht.

Die Konsequenz daraus ist, dass ein Zeichen in w nur maximal so oft auftauchen darf, wie es mögliche Speicher-Konfigurationen gibt. Die maximale Länge des kürzezten Wortes ergibt sich also aus dem Produkt der Mächtigkeit des Eingabealphabets a und der Speicher-Konfigurationen mit Platzbedarf kleiner gleich λ. Für Platzbedarf genau i ergeben sich die möglichen Speicher-Konfigurationen aus dem Produkt der Anzahl interner Zustände q, Mächtigkeit des Arbeitsalphabets k hoch Stringlänge, welche gleich dem Platzbedarf ist i, und der i möglichen Positionen des Lese-/Schreibkopfes, $qk^i i$. Es gibt also $\sum_{i=1}^{\lambda} qk^i i$ mögliche Speicher-Konfigurationen. Dies lässt sich auf jeden Fall durch ein p^λ nach oben beschränken. Das Ergebnis folgt:

$$n(\lambda) \leq a \sum_{i=1}^{\lambda} qk^i \lesssim p^\lambda \quad \Rightarrow \quad \lambda \geq \log n \tag{1}$$

5 Zweiwege-DTM

Ohne die Einschränkung der Bewegungsrichtung des Eingabebandes ist es dem DTM möglich eine Zelle der Eingabe mehrmals zu lesen. Damit kann die Maschine, während der Verarbeitung des Wortes, für eine Eingabezelle mehrere unterschiedliche Speicher-Konfigurationen annehmen. Werden nun alle Zustände notiert, die die DTM während der Verarbeitung annimmt, wenn die Eingabe gerade die j-te Zelle liest, dann lässt sich jeder Zelle eine Teilmenge der Menge aller Speicher-Konfigurationen c_i zuordnen. Dies ist graphisch dargestellt in Abb.3 durch die Konfigurationen unter den Zellen des Eingabebandes. Mit Sicherheit taucht unter einer Zelle nicht zweimal die selbe Speicher-Konfiguration auf, denn dann wäre die DTM in eine Endlosschleife gegangen und würde nicht anhalten, wie gefordert.

Treten bei der Verarbeitung des Wortes $w = x_1 \dots x_r \dots x_s \dots x_n$ wieder zwei gleiche Zeichen auf, $x_r = x_s$, und sind alle Speicher-Konfigurationen, die die DTM durchläuft während sie x_r, bzw. x_s liest gleich, dann beeinflusst das Teilwort $x_{r+1} \dots x_s$ deren Verhalten nicht. Um das einzusehen, stellt man sich das Teilwort als Abbildung der Speicher-Konfigurationen bei x_r nach x_s vor (in Abb.3 angedeutet durch die Pfeile). Da alle Speicher-Konfigurationen an beiden Stellen, x_r und x_s, während der Berechnung angenommen werden, muss die Abbildung bijektiv sein und permutiert demzufolge nur die Speicherzustände. Die DTM reagiert darauf aber nur mit der Vertauschung von Berechnungspfaden, Wege durch die Speicher-Konfigurationen, lässt aber das Ergebnis unberührt.

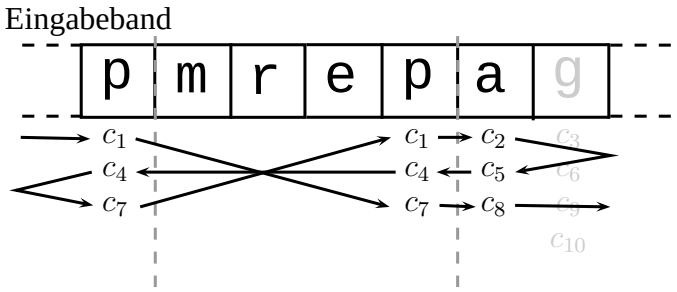

Abbildung 3: Unter den Zellen des Eingabewortes sind die Speicher-Konfigurationen, die die DTM während der Verarbeitung annimmt, aufgelistet. Bei beiden p sind diese gleich. Das Teilwort *mrep* vertauscht nur selbige.

Aus dieser Beobachtung folgt, dass für gleiche Zeichen im Eingabewort, die angenommenen Speicher-Konfigurationen ihrer Zellen schon unterschiedlich sein müssen. Ein *System*-Zustand ist mit einer Teilmenge der Speicher-Konfigurationen und dem zugehörigen Eingabezeichen zu identifizieren. Die Bestimmung der oberen Schranke der Wortlänge folgt dann dem Schema im Falle der Einweg-DTM:

Suche kürzestes Wort mit Platzbedarf genau λ. Zeige, dass gleiche *System*-Zustände nicht auftreten können, da sie die Minimalität des Wortes verletzen. Finde obere Schranke für *System*-Zustände. Diese existiert, da der Platz auf λ beschränkt wurde.

Hier sind die *System*-Zustände Teilmengen von Speicher-Konfigurationen. Deren mögliche Anzahl ist beschränkt durch die Potenzmenge aller Speicher-Konfigurationen. Eine obere Schranke für die Anzahl der Speicher-Konifigurationen haben wir schon gefunden $\leq p^\lambda$. Demnacht ist die Wortlänge beschränkt durch das Produkt aus Mächtigkeit des Eingabealphabet a und Mächtigkeit der Potenzmenge:

$$n(\lambda) \lesssim a\ 2^{p^\lambda} \quad \Rightarrow \quad \lambda \geq \log\log n$$

6 Ausblick

[Sze94] ist sehr detailliert und anschaulich in den Beweisen. In Kapitel 5 von [SHL65] wird ein formellerer Zugang über eine Äquivalenzrelation gewählt. Wie auch in [Pap94], ist diese dadurch definiert, dass Wort, die für alle λ-platzbeschränkten Berechnungen das gleiche Ergebnis hervorrufen, äquivalent sind. Zu bestimmen ist dann die Anzahl von Äquivalenzklassen.

Literatur

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[SHL65] R. E. Stearns, J. Hartmanis, and P. M. Lewis. Hierarchies of memory limited computations. In *FOCS '65: Proceedings of the 6th Annual Symposium on Switching Circuit Theory and Logical Design (SWCT 1965)*, pages 179–190, Washington, DC, USA, 1965. IEEE Computer Society.

[Sze94] Andrzej Szepietowski. *Turing Machines with Sublogarithmic Space.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.

[Tan10] Till Tantau. Theoretische informatik. http://www.tcs.uni-luebeck.de/lehre/2009-ws/ti/wiki/Vorlesung?action=AttachFile&do=get&target=2009-ws-ti.pdf, Februar 2010.