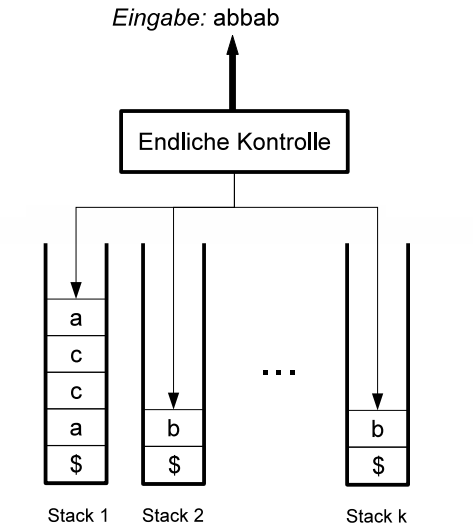
"Zwei Zähler sind universell"

Elena Weihe

January 11, 2011

1 PDAs mit mehreren Stacks (Multistack-PDAs)



Ein k-Stack-PDA ist ein deterministischer PDA mit k Stacks. Die Übergangsregeln der bereits bekannten deterministischen 1-PDAs sehen folgendermaßen aus:

$$\delta\,(q,\,a,\,X) = \{(p,\,y)\}$$

Entsprechend sehen die Übergangsregeln für einen k-PDA so aus:

$$\delta\,(q,\,a,\,X_1,\,X_2,...,X_k) = \{(p,\,y_1,y_2,...,y_k)\}$$

2 Zählermaschinen

Es gibt 2 Betrachtungsweisen für Zählermaschinen. Bei beiden besitzt die Maschine, wie auch ein PDA, ein Eingabeband und eine endliche Kontrolle. Das **erste Modell** hat k Zähler und es gilt:

- Ein Zähler kann jede natürliche Zahl annehmen.

- Jeder Zähler kann unabhängig um 1 erhöhen oder um 1 verringert werden. Ein Zähler dessen Wert 0 ist kann nicht verringert werden.
- Es kann nur zwischen Zählern die 0 sind und Zählern die nicht 0 sind unterschieden werden.

Das **zweite Modell** ist ein eingeschränkter Multistack-PDA mit folgenden Restriktionen:

- Es gibt nur 2 Stack-Symbole im Stackalphabet, diese sind \$ (als Symbol für den Boden des Stacks) und X.
- \$ steht immer ganz unten auf jedem Stack.
- \$ darf nur durch X\$ oder \$ ersetzt werden
- X darf nur durch X, XX oder ε ersetzt werden $\Rightarrow$ \$ ist immer das unterste Zeichen des Stacks und alle anderen Zeichen auf dem Stack (falls existent) sind X.

Beide Modelle sind berechenbarkeitstheoretisch äquivalent. Man kann X^i\$ als Zahl i identifizieren. Ebenso kann man \$ als 0 identifizieren. Den Zähler um 1 erhöhen entspricht dem Ersetzen von X mit XX bzw. von \$ mit X\$. Den Zähler um 1 verringern entspricht dem Herunternehmen von X vom Stack, also dem Ersetzen von von X mit ε. Für die folgenden Betrachtungen werden wir das erste Modell verwenden.

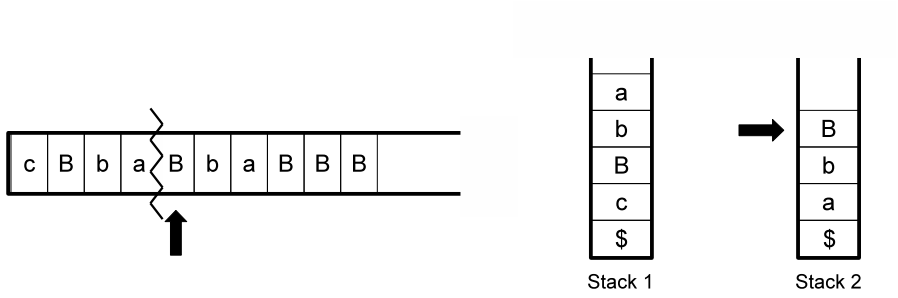
3 Eine 2-Zählermaschine ist universell

Im Folgenden wird gezeigt, wie eine 2-Zählermaschine eine Turingmaschine simulieren kann, indem wir folgende Simulationskette nutzen:

1. Ein **2-Stack PDA** kann eine **Turing-Maschine** simulieren
2. Eine **3-Zählermaschine** kann einen **2-Stack PDA** simulieren
3. Eine **2-Zählermaschine** kann eine **3-Zählermaschine** simulieren.

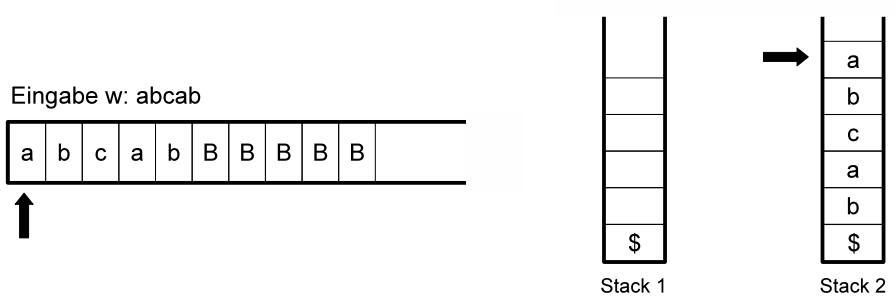
Dass die jeweils mächtigere Maschine immer auch die kleinere Maschine simulieren kann ist offensichtlich. Somit wird durch die Simulationskette gezeigt, dass eine 2-Zählermaschine berechenbarkeitstheoretisch äquivalent zur Turing-Maschine ist.

3.1 Eine 2-Stack PDA kann eine Turingmaschiene simulieren



Zur Simulation einer TM durch einen 2-Stack PDA, wird das Band der TM aufgebrochen. Die

Zeichen, die links des Lesekopfes der TM stehen, kommen auf Stack 1, wobei das Zeichen an der Bruchstelle oben und das Zeichen am linken Ende des Bandes unten liegt. Das Zeichen, auf das der Lesekopf zeigt und die Zeichen rechts davon kommen auf Stack 2.
Ab einer bestimmen Stelle auf dem Band stehen nur noch unendlich viele Blanks (B), diese werden nicht auf Stack 2 gelegt. Statt dessen interpretiert der PDA das Zeichen $ auf Stack 2 als Blank.Das oberste Zeichen auf Stack 2 ist somit das Zeichen, auf das der Lesekopf der zu simlulierenden TM aktuell zeigt.



Vor Berechnungsbeginn liest der PDA die Eingabe w ein und schreibt sie auf Stack 1, so dass w invertiert auf diesem Stack liegt. Dann wird w auf Stack 2 übertragen, so dass das erste Zeichen von w das oberste Zeichen auf Stack 2 ist. Die Ausgangssituation der TM

- Wenn die TM das Zeichen X mit Y ersetzt und anschließend nach rechts geht, legt der PDA Y auf Stack 1
- Wenn die TM X mit Y ersetzt und nach links geht, nimmt der PDA das oberste Zeichen Z von Stack 1 und X von Stack 2. Anschließend werden die Zeichen ZY auf Stack 2 gelegt.

3.2 Eine 3-Zählermaschine kann einen 2-Stack PDA simulieren

Wir nehmen an, der zu simulierende 2-Stack PDA hat $r-1$ Symbole im Stackalphabet. Jedem Symbol wird eine natürliche Zahl k von 1 bis $r-1$ zugewiesen. Nun können wir uns eine Stackbelegung $X_1X_2...X_n$ (X_i aus dem Stackalphabet) als ganze Zahl s zur Basis r vorstellen:

$$s = X_1 + X_2 \cdot r + ... + X_{n-1} \cdot r^{n-2} + X_n \cdot r^{n-1}$$

Zwei der drei Zähler nutzen wir um die beiden Stacks zu repräsentieren. Der dritte Zähler dient als Hilfszähler. Er wird genutzt um die Stackoperationen durchzuführen. Wir nennen ihn h. Zu Beginn jeder Operation wird h auf 0 gesetzt (wiederholtes Verringern um 1 bis 0 erreicht ist). Die Stackoperationen $push(x)$ und pop werden nun folgendermaßen simuliert:

pop: Bei einer pop-Operation wird das oberste Zeichen vom Stack genommen und gelesen. Um dies zu simulieren müssen wir s mit s/r ersetzen. Der Rest der ganzzahligen Division ergibt das oberste Zeichen auf dem Stack (X_1). Die Division wird auf der Zählermaschine folgendermaßen durchgeführt: der Zähler s wird wiederholt um r verringert und h jeweils um 1 erhöht. Wenn der Zähler der ursprunglich s enthielt 0 wird, haben wir X_1 gelesen. Der Zähler h enthält nun das Ergebnis der Division (s/r). Dieses wird nun noch auf s kopiert. Hierzu erhöhen wir s und verringern h um 1, bis h 0 erreicht hat.

push(X): Die push-Operation legt das Symbol X auf den Stack. Hierzu multiplizieren wir s erst mit r und addieren anschließend die Zahl k die X repräsentiert. Dafür verringern wir wiederholt s um 1 und erhöhen jeweils h um r. Wenn s 0 erreicht, enthält h $s \cdot r$. Der Wert von h wird auf s kopiert und anschließend k zu s addiert.

3.3 Eine 2-Zählermaschine kann eine 3-Zählermaschine simulieren

Die Eindeutigkeit der Primzahlzerlegung wird genutzt, um die drei Zähler a, b und c durch nur einen Zähler, wir nennen ihn i, zu repräsentieren:

$$i = 2^a \cdot 3^b \cdot 5^c$$

Der zweite Zähler h wid als Hilfszähler gebraucht. Die Operationen der Zählermaschiene werden folgendermaßen simuliert:

Erhöhen: Um beispielsweise a zu erhöhen, müssen wir $2^a \cdot 3^b \cdot 5^c$ mit $2^{a+1} \cdot 3^b \cdot 5^c = 2 \cdot 2^a \cdot 3^b \cdot 5^c$ ersetzen. Um einen der Zähler zu erhöhen, müssen wir also i mit der entsprechenden Primzahl multiplizieren. Wie man einen Zähler mit einer Konstante multipliziert, haben wir schon bei der vorherigen Simulation gesehen.

Verringern: Um einen der Zähler zu verringern müssen wir i durch die entsprechende Primzahl teilen. Auch das haben wir schon bei 3.2 demonstriert.

Test auf 0: Wenn beispielsweise s_1 0 ist, enthält der Zähler i eine Zahl der Form $2^0 \cdot 3^b \cdot 5^c = 3^b \cdot 5^c$. Der Wert von i ist somit nicht durch 2 teilbar. Um einen der Zähler auf 0 zu testen, testen wir also die Teilbarkeit von i durch die entsprechende Primzahl p. Hierzu teilen wir i durch p und überprüfen, ob die ganzahlige Division einen Rest ergibt oder nicht.

Damit ist zusammen mit 3.2 und 3.1 gezeigt: Eine 2-Zählermaschine kann eine Turingmaschine simulieren und ist damit ein universelles Berechnungsmodell.

Quelle: J.E.Hopcroft, R. Motwani, J.D.Ullman Introduction to Automata Theory, Languages, and Computation Addison-Wesley 2001