Da die Eingabe für diese Folge kürzer als n ist, können wir die Induktionshypothese anwenden und schließen, dass $a_2 \ldots a_{n-1}$ die Form yy^R hat, wobei y beliebig ist. Da $x = a_1 y y^R a_n$ und da wir wissen, dass $a_1 = a_n$, schließen wir, dass x die Form ww^R hat, wobei $w = a_1 y$.

Dies ist das Kernstück des Beweises, dass x nur dann durch Endzustand akzeptiert werden kann, wenn x für ein w gleich ww^R ist. Somit ist der »Nur-dann-Teil« des Beweises abgeschlossen, aus dem zusammen mit dem zuvor bewiesenen »Wenn-Teil« hervorgeht, dass P genau die in L_{wwr} enthaltenen Zeichenreihen durch Endzustand akzeptiert.

6.2.2 Akzeptanz durch leeren Stack

Für jeden PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ definieren wir zudem

$$N(P) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \varepsilon, \varepsilon)\}$$

für einen beliebigen Zustand q. Das heißt, $N(P)$ ist die Menge der Eingabe-Zeichenreihen w, die P einlesen kann und bei der er gleichzeitig seinen Stack leeren kann.[2]

Beispiel 6.6 Der PDA P aus dem Beispiel 6.2 leert seinen Stack nie, daher ist $N(P) = \emptyset$. Eine kleine Modifikation wird es P jedoch erlauben, L_{wwr} sowohl durch das Leeren des Stacks als auch durch den Endzustand zu akzeptieren. Statt des Übergangs $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$ verwenden wir $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$. Jetzt kann P das letzte Symbol von seinem Stack entfernen, während er akzeptiert, und $L(P) = N(P) = L_{wwr}$.

Da die Menge der akzeptierenden Zustände im Fall der Akzeptanz durch leeren Stack irrelevant ist, lassen wir gelegentlich die letzte (siebte) Komponente in der Spezifikation eines PDA P weg, wenn uns lediglich die Sprache interessiert, die P durch Leeren seines Stacks akzeptiert. Wir geben P dann als 6-Tupel $(Q, \Sigma, \Gamma, \delta, q_0, Z_0)$ an.

6.2.3 Vom leeren Stack zum Endzustand

Wir werden zeigen, dass es sich bei der Klasse von Sprachen, die für einen PDA P $L(P)$ sind, um die gleiche Klasse von Sprachen handelt, die $N(P)$ für einen im Allgemeinen anderen PDA P sind. Diese Klasse ist auch genau die Klasse der kontextfreien Sprachen, wie wir in Abschnitt 6.3 sehen werden. Unsere erste Konstruktion zeigt, wie man aus einem PDA P_N, der eine Sprache L durch Leeren seines Stacks akzeptiert, einen PDA P_F konstruiert, der L durch Endzustand akzeptiert.

Satz 6.3 Wenn für einen PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$ die Sprache $L = N(P_N)$, dann gibt es einen PDA P_F, derart dass $L = L(P_F)$.

2 In $N(P)$ steht das N für »Null-Stack«, ein Synonym für »leerer Stack«.

Beweis Der diesem Beweis zu Grunde liegende Gedankengang ist in Abbildung 6.4 dargestellt. Wir verwenden ein neues Symbol X_0, das kein Symbol aus Γ sein darf. X_0 ist sowohl das Startsymbol von P_F als auch eine Markierung am unteren Ende des Stacks, die uns mitteilt, wann P_N einen leeren Stack erreicht hat. Das heißt, wenn P_F X_0 als oberstes Symbol auf dem Stack vorfindet, dann weiß der Automat, dass P_N bei der gleichen Eingabe seinen Stack leeren würde.

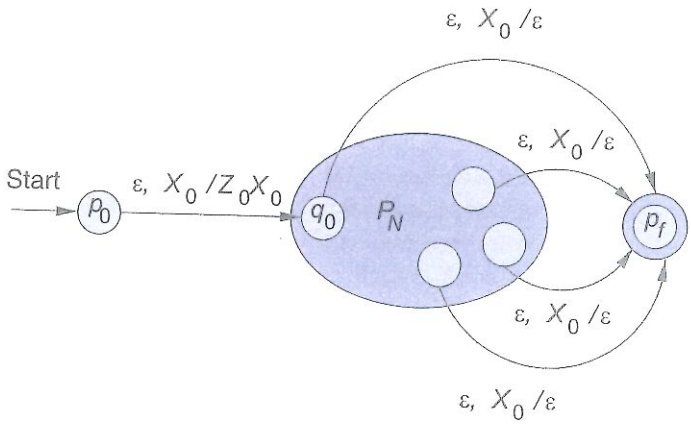

Abbildung 6.4: P_F simuliert P_N und akzeptiert, wenn P_N seinen Stack leert

Wir brauchen auch einen neuen Startzustand p_0, dessen einzige Funktion darin besteht, das Startsymbol Z_0 von P_N auf dem Stack abzulegen und in den Startzustand q_0 von P_N zu wechseln. Dann simuliert P_F P_N so lange, bis der Stack von P_N leer ist, was P_F daran erkennt, dass er X_0 als oberstes Symbol auf dem Stack vorfindet. Schließlich brauchen wir einen weiteren Zustand p_f, der den akzeptierenden Zustand von P_F repräsentiert. Dieser PDA geht in den Zustand p_f über, sobald er erkennt, dass P_N seinen Stack geleert hat.

Die Spezifikation von P_F lautet wie folgt:

$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\}).$$

wobei δ_F definiert wird durch

1. $\delta_F(p_0, \varepsilon, X_0) = \{(q_0, Z_0 X_0)\}$. Von seinem Startzustand aus geht P_F unmittelbar in den Startzustand von P_N über, wobei er das Startsymbol Z_0 auf dem Stack ablegt.
2. Für alle Zustände q aus Q, Eingabesymbole a aus Σ oder $a = \varepsilon$ und Stack-Symbole Y aus Γ enthält $\delta_F(q, a, Y)$ alle Paare aus $\delta_N(q, a, Y)$.
3. Zusätzlich zur Regel (2) enthält $\delta_F(q, \varepsilon, X_0)$ für jeden Zustand q aus Q (p_f, ε).

Wir müssen zeigen, dass w genau dann in $L(P_F)$ enthalten ist, wenn w in $N(P_N)$ enthalten ist.

(Wenn-Teil) Gegeben ist die Aussage, dass für einen Zustand $(q_0, w, Z_0) \overset{*}{\underset{P_N}{\vdash}} (q, \varepsilon, \varepsilon)$. Dank Satz 6.5 können wir X_0 am unteren Ende des Stacks einfügen und schließen, dass $(q_0, w, Z_0X_0) \overset{*}{\underset{P_N}{\vdash}} (q, \varepsilon, X_0)$. Da nach der oben genannten Regel (2) P_F sämtliche Bewegungen von P_N ausführt, können wir zudem folgern, dass $(q_0, w, Z_0X_0) \overset{*}{\underset{P_F}{\vdash}} (q, \varepsilon, X_0)$. Wenn wir diese Bewegungsfolge mit der ersten und der letzten Bewegung kombinieren, die sich aus den Regeln (1) und (3) ergeben, dann erhalten wir:

$$(p_0, w, X_0) \underset{P_F}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{P_F}{\vdash}} (q, \varepsilon, X_0) \underset{P_F}{\vdash} (p_f, \varepsilon, \varepsilon)$$

Folglich akzeptiert P_F die Zeichenreihe w durch Endzustand.

(Nur-wenn-Teil) Die Umkehrung erfordert lediglich, dass wir erkennen, dass die zusätzlichen Übergänge nach den Regeln (1) und (3) uns sehr beschränkte Möglichkeiten geben, w durch Endzustand zu akzeptieren. Wir müssen Regel (3) im letzten Schritt anwenden und wir können diese Regel nur verwenden, wenn der Stack von P_F lediglich X_0 enthält. Das Symbol X_0 kommt an keiner anderen Stelle des Stacks vor als an der untersten Position. Überdies wird Regel (1) nur im ersten Schritt angewandt und sie *muss* in diesem Schritt angewandt werden.

Folglich muss jede Berechnung von P_F, die zur Akzeptanz von w führt, wie die Folge (6.1) aussehen. Zudem muss die Mitte der Berechnung (alle Schritte außer des ersten und des letzten) auch eine Berechnung von P_N sein, wobei sich X_0 unten im Stack befinden muss. Begründet wird dies dadurch, dass, abgesehen vom ersten und letzten Schritt, P_F keine Übergänge ausführen kann, die nicht ebenso Übergänge von P_N sind, und dass X_0 dabei nur dann als oberstes Element im Stack auftreten kann, wenn die Berechnung im nächsten Schritt endet. Wir schließen daraus, dass $(q_0, w, Z_0) \overset{*}{\underset{P_N}{\vdash}} (q, \varepsilon, \varepsilon)$. Das heißt, w ist in $N(P_N)$ enthalten. ■

Beispiel 6.7 Wir wollen einen PDA entwerfen, der Folgen von `if`- und `else`-Wortsymbolen eines C-Programms verarbeitet, wobei *i* für `if` und *e* für `else` steht. Sie wissen aus Abschnitt 5.3.1, dass es problematisch ist, wenn in einem Präfix mehr `else` als `if` vorhanden sind, da es dann nicht möglich ist, jedes `else` einem voranstehenden `if` zuzuordnen. Daher werden wir das Stack-Symbol Z einsetzen, um die Differenz zwischen der Anzahl der bereits gelesenen *i* und der Anzahl der *e* zu verzeichnen. Dieser einfache PDA, der nur über einen Zustand verfügt, ist im Übergangsdiagramm aus Abbildung 6.5 dargestellt.

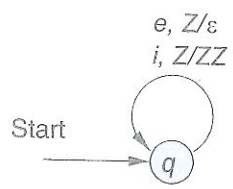

Abbildung 6.5: PDA, der `if`/`else`-Fehler durch Leeren des Stacks akzeptiert

Wir legen ein weiteres Z auf dem Stack ab, wenn ein i gelesen wird, und entfernen ein Z vom Stack, wenn ein e gelesen wird. Da wir mit einem Z auf dem Stack beginnen, halten wir uns tatsächlich an die Regel, dass n-1 mehr i als e gelesen wurden, wenn der Stack Z^n enthält. Wenn der Stack leer ist, dann lag eine um eins höhere Anzahl von e als von i vor, und die bis dahin gelesene Eingabe wurde damit zum ersten Mal unzulässig. Unser PDA akzeptiert diese Art von Zeichenreihen durch einen leeren Stack. Die formale Spezifikation von P_N lautet:

$$P_N = (\{q\}, \{i, e\}, \{Z\}, \delta_N, q, Z).$$

wobei δ_N definiert wird durch:

1. $\delta_N(q, i, Z) = \{(q, ZZ)\}$. Diese Regel bewirkt, dass ein Z auf dem Stack abgelegt wird, wenn ein i gelesen wird.
2. $\delta_N(q, e, Z) = \{(q, \varepsilon)\}$. Diese Regel bewirkt, dass ein Z vom Stack entfernt wird, wenn ein e gelesen wird.

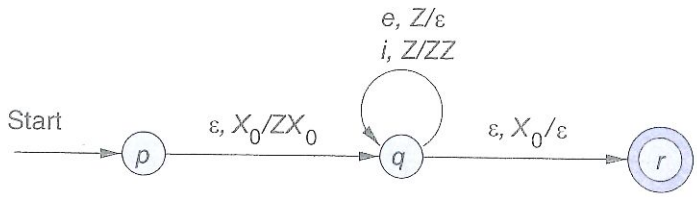

Abbildung 6.6: Konstruktion eines PDA, der durch Endzustand die Sprache des PDA aus Abbildung 6.5 akzeptiert

Lassen Sie uns nun aus P_N einen PDA P_F konstruieren, der dieselbe Sprache durch Endzustand akzeptiert. Das Übergangsdiagramm für diesen PDA P_F ist in Abbildung 6.6[3] dargestellt. Wir führen einen neuen Startzustand p und einen akzeptierenden Zustand r ein. Wir werden X_0 als Stackanfangsmarkierung verwenden. Die formale Definition von P_F lautet:

$$P_F = (\{p, q, r\}, \{i, e\}, \{Z, X_0\}, \delta_F, p, X_0, \{r\}),$$

wobei δ_F sich wie folgt zusammensetzt:

1. $\delta_F(p, \varepsilon, X_0) = \{(q, ZX_0)\}$. Diese Regel bewirkt, dass P_F mit der Simulation von P_N beginnt, wobei X_0 die Stackanfangsmarkierung darstellt.
2. $\delta_F(q, i, Z) = \{(q, ZZ)\}$. Diese Regel bewirkt, dass ein Z auf dem Stack abgelegt wird, wenn ein i gelesen wird, entsprechend dem Verhalten von P_N.
3. $\delta_F(q, e, Z) = \{(q, \varepsilon)\}$. Diese Regel bewirkt, dass ein Z vom Stack entfernt wird, wenn ein e gelesen wird; auch diese Regel simuliert P_N.
4. $\delta_F(q, \varepsilon, X_0) = \{(r, \varepsilon, \varepsilon)\}$. Diese Regel bedeutet, dass P_F akzeptiert, wenn der simulierte Automat P_N seinen Stack geleert hat.

3 Es ist nicht von Belang, dass wir hier die Zustände p und r verwenden, während in Abb. 6.4 von den Zuständen p_0 und p_f die Rede ist. Die Namen von Zuständen können beliebig gewählt werden.