Sommersemester 2004

Seminar

G r a p h g r a m m a t i k e n

Lehrstuhl für Informatik III

RWTH Aachen

Thema:

Node Replacement Graph Grammars

Verfasser: Gereon Frey, Dirk Hesse
Betreuer: Dipl.-Inform. Ulrike Ranger

Inhaltsverzeichnis

1 Einleitung

In dieser Arbeit werden *Node Replacement Graph Grammars* diskutiert, bei welchen ein Knoten eines gegebenen Graphen durch einen neuen Graphen ersetzt wird und mit dem ursprünglichen Graphen auf festgelegte Art und Weise verbunden wird. Graphgrammatiken definieren auf mathematisch präzise Weise diese Ersetzung.

Während kontextfreie Wort-Grammatiken dem formal-präzisen Aufbau und der Veränderung von Zeichenketten dienen, operieren Graphgrammatiken auf der weitaus komplexeren Menge der Graphen. Hierbei sei angemerkt, dass sich auch Zeichenketten als Graphen auffassen lassen. Zwischen beiden Klassen von Grammatiken existieren Gemeinsamkeiten, wie das endliche System von Regeln oder Produktionen sowie die Herleitung von Ableitungsbäumen oder Normalformen.

Eine mögliche Anwendung von Graphgrammatiken ist die Transformation von Programmen, d.h. die Daten- und Kontrollflüsse werden als Graphen dargestellt und können dann, den Regeln einer Grammatik folgend, verändert werden.

1.1 Grundlagen

Wie bereits erwähnt, dienen Graphersetzungssysteme der Transformation von Graphen. Diese Transformationen werden durch Produktionen beschrieben, welche durch ein Tupel (M, D, E) gegeben sind. Eine solche Produktion gibt an, dass in dem Graphen H (von engl. host) ein Teilgraph M (von engl. mother), inklusive aller Kanten, die zum Restgraphen H^- inzident sind, entfernt und durch einen Graphen D (von engl. daughter) ersetzt wird. Dieser wird dann, mit in E (von engl. to embed) definierten Regeln, in den Restgraphen H^- eingebettet. Eine Graphgrammatik besteht neben Knoten und Kanten aus einer endlichen Menge solcher Produktionen.

Grundlegend ergeben sich zwei verschiedene Möglichkeiten, einen Teilgraphen M durch D zu ersetzen: zum einen das "*gluing*", zum anderen das "*connecting*". Beim *"gluing"* werden Teile des Restgraphen H^- mit Teilen von D überlagert und dann verschmolzen. Dadurch entsteht die Verbindung zwischen H^- und D. Dem *"gluing"* steht der Ansatz des *"connecting"* gegenüber. Bei diesem werden Regeln angegeben, wie der neue Teilgraph D in den Restgraph H^- durch neue Kanten eingebunden werden soll.

Ein Ansatz des allgemeinen Themas "Graphgrammatiken" ergibt sich schon aus dem Namen dieser Arbeit. '*Node Replacement Graph Grammars*' betrachten Graphgrammatiken, bei denen nur einzelne Knoten und nicht Kanten ersetzt werden. Andere Ersetzungssysteme, wie Hyperedge-Replacement Graphgrammars, die nicht Knoten, sondern Kanten ersetzen, bzw. Systeme die Teilgraphen ersetzen, sind nicht Bestandteil dieser Arbeit. Des Weiteren wird nur die Klasse der Graphgrammatiken betrachtet, die die Eigenschaft der Konfluenz innehaben, was bedeutet, dass die Reihenfolge, in der eine Folge von Produktionen angewendet wird, keine Rolle spielt. Diese Einschränkung wird gefordert, um Notationen wie Ableitungsbäume oder Linksableitungen betrachten zu können.

Es wird hier eine Definition für Graphen angegeben, welche die nötigen Eigenschaften wie Kanten- und Knotennamen berücksichtigt.

Definition 1.1.1 (Graph): *Ein gerichteter Graph ohne Schleifen über den Alphabeten Σ (die Knotennamen) und Γ (die Kantennamen) ist ein Tupel $G = (V, E, \lambda)$, wobei V die endliche Menge der Knoten, $E \subseteq \{(v, \gamma, w) \mid v, w \in V, v \neq w, \gamma \in \Gamma\}$*

die Menge der Kanten und $\lambda : V \to \Sigma$ die Funktion ist, die den Knoten Namen zuordnet. Die Komponenten von H werden mit V_H, E_H und λ_H benannt.
Bei ungerichteten Graphen gilt zusätzlich die Annahme, dass wenn $(u, \gamma, v) \in E$, dann auch $(v, \gamma, u) \in E$ gelten muss. Graphen ohne Kanten- und/oder Knotennamen sind mit $\Sigma = \{\sharp\}$ oder $\Gamma = \{*\}$ modellierbar.
Die Menge aller Graphen über Σ und Γ wird mit $GR_{\Sigma,\Gamma}$ bezeichnet. ◇

Wichtig bei dieser Definition ist die Menge der Knoten- und Kantennamen, da diese benutzt werden, um Knoten und Kanten zu identifizieren. Ein weiterer Begriff, welcher hier durch eine Definition geklärt werden soll, ist die Isomorphie.

Definition 1.1.2 (Isomorphie): *Seien H und K zwei Graphen. Diese sind isomorph, wenn eine bijektive Abbildung $f : V_H \to V_K$ existiert, so dass gilt:*

$$E_K = \{(f(v), \gamma, f(w)) \mid (v, \gamma, w) \in E_H\}.$$

Die Menge aller Graphen, die isomorph zu einem Graphen G sind, wird mit $[G]$ bezeichnet. G wird auch als ein konkreter und $[G]$ als ein abstrakter Graph bezeichnet. ◇

Die Menge aller konkreten Graphen wird mit $GR_{\Sigma,\Gamma}$ und die Menge aller abstrakten Graphen mit $[GR_{\Sigma,\Gamma}]$ bezeichnet. Für eine Grammatik G ist $L(G) \subseteq [GR_{\Sigma,\Gamma}]$ die zugehörige Graphsprache.

1.2 Aufbau der Arbeit

Nachdem in Kapitel 1 ein Überblick über Graphen gegeben wurde, beschäftigt sich das zweite Kapitel mit den NLC-Grammatiken als eine Form von Graphgrammatiken. Bei diesen werden Knoten mit bestimmten Knotenlabeln durch die Produktionen der Grammatik adressiert und ersetzt. Das dritte Kapitel beschäftigt sich als Kernstück dieser Arbeit mit der Klasse der NCE-Grammatiken, bei welcher Knoten direkt adressiert werden können. Diese Klasse ist somit mächtiger als die Klasse der NLC-Grammatiken. Besonderes Augenmerk wird auf die Eigenschaft der Konfluenz gelegt, welche besagt, dass die Reihenfolge der Anwendung von Produktionen auf den Endgraphen keine Auswirkung hat. Des Weiteren werden in diesem Kapitel die grundlegenden Notationen wie Ableitungsbäume, Links-Ableitungen oder Normalformen beschrieben. Nachdem im vierten Kapitel auf Charakterisierungen und die Entscheidbarkeit von NCE-Grammatiken eingegangen wird, schließt eine Zusammenfassung in Kapitel 5 die Arbeit ab.

2 NLC

In diesem Kapitel werden die NLC (Node label controlled) Graphgrammatiken als eine Form einer Graphgrammatik vorgestellt. Nach der formalen Definition wird die Grammatik um einige interessante Eigenschaften erweitert, da diese eine gute Grundlage für die interessanten NCE Graphgrammatiken aus Kapitel 3 bilden.

2.1 Ersetzung basierend auf Knotennamen

Den NLC-Graphgrammatiken liegt ein ungerichteter Graph zugrunde. Anhand von Produktionsregeln werden Knoten, welche einen angegebenen Knotennamen haben, durch einen Tochtergraphen ersetzt und mittels einer Einbettungsfunktion mit dem Restgraphen verbunden. Formal gilt:

Definition 2.1.1 (die NLC Graphgrammatiken): *Eine NLC Graphgrammatik* $G = (\Sigma, \Delta, P, C, S)$ *besteht aus dem Alphabet* Σ*, welches sich aus terminalen (*$\Delta \subseteq \Sigma$*) und nichtterminalen Symbolen (*$\Sigma - \Delta$*) zusammensetzt. Die Menge*

$$P = \{f : X \to D \mid X \in \Sigma_H - \Delta_H, H \text{ sei der Hostgraph}, D \in GR_{\Sigma,\Gamma}, V_D \cap V_{H^-} = \emptyset\}$$

enthält alle Produktionen, wobei X *der Name des nichtterminalen Knotens des Graphen* H *ist, welcher durch den Graphen* D *ersetzt wird. Die Knotenmenge von* D *ist hierbei disjunkt zu der Knotenmenge des Restgraphen* H^-*.*
$C \subseteq V \times V$ *ist die Relation der Verbindungsregeln, d.h. der Instruktionen, die angeben, wie der Restgraph* H^- *und* D *verbunden werden sollen.* S *ist der initiale Graph, also das Startsymbol einer Graphgrammatik.* ◇

Eine Produktionsregel $p : X \to D$ kann also auf Knoten angewandt werden, die das Label X tragen. Dazu wird zu dem zu ersetzenden Knoten x mit Label X die Nachbarschaft N von x, d.h. die Menge $N = \{y \in V_H \mid (x, y) \in E_H\}$ ermittelt. Danach wird der Knoten x und alle Kanten, die zu x inzident sind, entfernt. Der Graph D wird eingefügt und anschließend entsprechend der Verbindungsinstruktionen mit H^- abgearbeitet.

Für jede Instruktion $(U, V) \in C$ wird geprüft, ob in der Nachbarschaft N ein Knoten mit Namen U und im Graphen D ein Knoten mit Namen V existiert. Ist dies der Fall, so wird eine Kante zwischen diesen Knoten gezogen, die auch "bridge" genannt wird, da sie eine Brücke zwischen dem alten und dem neu hinzugefügten Graphen bildet. Ist kein solcher Knoten in der Nachbarschaft, so wird die Verbindungsinstruktion nicht angewandt.

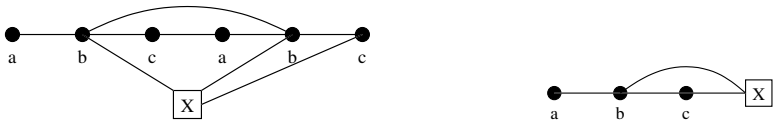
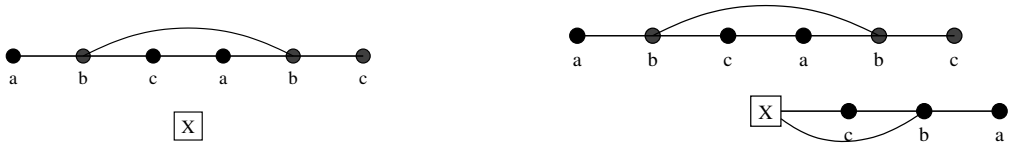


Abb. 1: Die Graphen H und D

Folgendes Beispiel soll diese Vorgehensweise erläutern, wobei aus Gründen der Übersichtlichkeit die Nichtterminale als Rechtecke dargestellt sind.

$C = \{(c, a), (b, b), (b, X)\}$ sei die Menge der Verbindunginstruktionen. Abbildung 1 stellt den initialen Graphen H sowie den Tochtergraphen D dar.

Abbildung 2 verdeutlicht die Anwendung der Produktion $p : H \to D$. Zunächst werden die drei zu X inzidenten Kanten entfernt und schließlich der Knoten X mit dem Tochtergraphen D ersetzt. Die folgende Abbildung 3 verdeutlicht die Anwendung der Verbindungsinstruktionen und zeigt den entstandenen Graphen: es sind nur Kanten zwischen dem eingesetzten Graphen und der Nachbarschaft des alten Knotens entstanden.



(a) Suche der Nachbarschaft und Entfernung der zu X inzidenten Kanten

(b) Ersetzung des Knotens X durch den Graphen D

Abbildung 2: Anwendung der Produktion $p : X \to D$

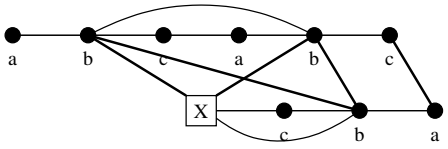


Abbildung 3: Anwendung der Verbindungsinstruktionen

2.2 Erweiterungen für NLC-Graphgrammatiken

Die so definierte NLC Graphgrammatik wird nun um einige Eigenschaften erweitert, um eine größere Anzahl von Graphen ansprechen zu können. Zum einen sollen die Graphgrammatiken nicht nur auf ungerichtete, sondern auch auf gerichtete Graphen angewendet werden können. Die Erweiterung der Definition ist einfach, da nur der Graph S, die Graphen D innerhalb der Produktionen, und die Verbindungsrelation entsprechend angepasst werden müssen. Die Nachbarschaft teilt sich dann in zwei Gruppen: zum einen die "in-Nachbarn", von denen eine Kante zu dem zu ersetzenden Knoten geht, zum anderen die "out-Nachbarn" die eine eingehende Kante vom zu ersetzenden Knoten haben. Die erweiterte Verbindungsrelation hat dann die Form $C \subseteq \Sigma \times \Sigma \times \{in, out\}$, wobei in bzw. out bestimmt, ob die Knoten in der "in-", bzw. "out-Nachbarschaft" liegen. Eine NLC Graphgrammatik, die auf gerichteten Graphen operiert, wird auch dNLC Graphgrammatik (d wie "directed") genannt.

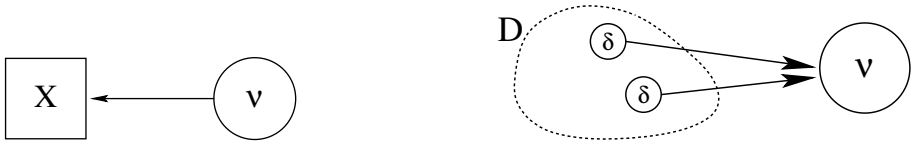
Ist $p : X \to D$ nun eine Produktion, die auf einen Knoten x mit Label X, oder in anderer Schreibweise $\lambda_H(x) = X$, angewandt wird, so besagt eine Verbindungsregel $c = (\nu, \delta, in)$, dass für einen Knoten der Nachbarschaft mit Namen ν, der eine ausgehende Kante zu x hatte, auch eine Kante zu allen Knoten des Tochtergraphen mit Label δ in dieser Richtung eingefügt wird. Abbildung 4 verdeutlicht die Idee der dNLC-Grammatiken.

(a) Der Graph H

(b) Der Graph H nach der Substitution

Abb. 4: Ein Beispiel für gerichtete Graphen im NLC Framework

Eine zusätzliche Erweiterung der dNLC-Grammatiken ist die Möglichkeit, die Richtung einer Kante während der Anwendung einer Produktion zu ändern. Dies ist durch die Erweiterung der Verbindungsrelation zu $C \subseteq \Sigma \times \Sigma \times \{in, out\} \times \{in, out\}$ möglich. Hierbei gibt das letzte $\{in, out\}$ die Richtung der neu eingefügten Kante an, d.h. wenn eine Kante von ν nach X existiert hat, so werden Kanten von allen Knoten des Tochtergraphen mit Label δ zu ν eingefügt. Abbildung 5 zeigt eine Ersetzung bei dNLC-Grammatiken mit Änderung der Kantenrichtung.

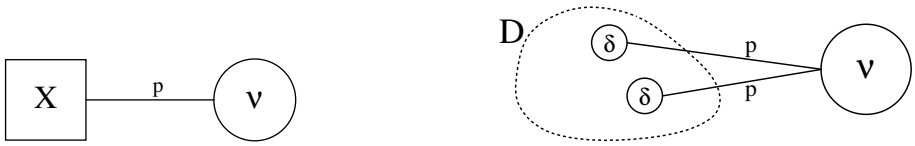


(a) Der Graph H

(b) Der Graph H nach der Substitution

Abb. 5: Ein Beispiel für gerichtete Graphen mit Richtungsänderung

Die wohl wichtigste Erweiterung ist die Berücksichtigung der Kantennamen. Um diese in den NLC Graphgrammatiken verwenden zu können, wird die Verbindungsrelation aus der Definition der NLC Graphgrammatiken um die Menge der Kantennamen erweitert. Sie hat dann die Form $C \subseteq \Sigma \times \Delta \times \Sigma$. Abbildung 6 zeigt ein Beispiel für die Einbeziehung der Kantennamen. Die Richtungen der Kanten werden bei dem Beispiel nicht betrachtet.



(a) Der Graph H

(b) Der Graph H nach der Substitution

Abb. 6: Ein Beispiel für Graphen im NLC Framework mit Kantenlabeln

Beispiel: *Sei $p : X \to D$ eine Produktion, die auf einen Knoten x mit $\lambda_H(x) = X$ angewendet wird. Dann gilt für ein $c = (\nu, p, \delta)$, dass wenn zwischen dem Knoten x und einem Knoten der Nachbarschaft mit dem Label ν eine Kante mit dem Label p war, auch eine Kante zwischen ν und allen Knoten mit Label δ des neu eingefügten Graphen D entsteht.*

NLC Graphgrammatiken mit dieser Eigenschaft werden auch eNLC Graphgrammatiken genannt.

Eine weitere Eigenschaft, die sehr leicht erreicht werden kann, ist das Abändern des Kantenlabels bei der Ausführung einer Produktion. Dazu wird die Verbindungsrelation auf $C \subseteq \Sigma \times \Delta \times \Delta \times \Sigma$ erweitert, wobei $c = (\nu, p, q, \delta)$ mit $c = (\nu, p/q, \delta)$ abgekürzt wird. Eventuell neu eingefügte Kanten haben nun nicht

mehr den Namen p, sondern den Namen q. Diese Eigenschaft wird "dynamic edge relabeling" genannt.

Natürlich sind auch Kombinationen dieser Eigenschaften möglich. Eine Graphgrammatik, welche beispielsweise auf gerichteten Graphen operiert und die Kantennamen berücksichtigt, wird also analog edNLC-Graphgrammatik genannt. Nach Einführung der NLC-Graphgrammtik und ihren Erweiterungen wird im nächsten Kapitel die mächtigere Form der NCE-Grammatiken eingeführt.

3 NCE

Ein Nachteil der NLC-Grammatiken ist, dass die Knoten eines Tochtergraphen nicht direkt adressiert werden können, sondern nur über ihre Labels ansprechbar sind. Eine direkte Adressierung hätte den Vorteil, dass Verbindungsinstruktionen wesentlich feiner angegeben werden können.

3.1 Ersetzung basierend auf der Nachbarschaft

Verwirklicht wird dieser Ansatz, durch die Definition der Verbindungsinstruktionen als (ν, x), was bedeutet, dass der Knoten x, unabhängig von seinem Label, im Tochtergraphen mit allen Knoten mit Label ν in der Nachbarschaft des Muttergraphen verbunden wird. Graphgrammatiken, die diese Art von Verbindungsinstruktionen verwenden, werden NCE Graphgrammatiken genannt, wobei NCE für *Neighbourhood Controlled Embedding* steht, was die Bedeutung der Nachbarschaft beim Einfügen verdeutlichen soll. Da die Zahl der möglichen Nachbarknoten eines Muttergraphen unbeschränkt ist, ist eine direkte Adressierung dieser nicht möglich.

Eine NCE Graphgrammatik $G = \{\Sigma, \Delta, P, S\}$ hat einen ähnlichen Aufbau wie die NLC Graphgrammatik mit dem Unterschied, dass die endliche Menge der Produktionen P als $X \to (D, C)$ definiert ist, d.h. die Verbindungsinstruktionen werden für jede Produktion einzeln und nicht einmal für alle angegeben.

3.2 Vorteile von NCE gegenüber NLC

Es stellt sich die Frage, wodurch der Schritt von NLC zu NCE Graphgrammatiken gerechtfertigt ist. Ein deutlicher Vorteil letzterer ist die direkte Adressierung der Knoten des Tochtergraphen. Weniger offensichtlich ist die erweiterte Kontextfreiheit. NLC Graphgrammatikn sind zwar insofern kontextfrei, dass Änderungen vollständing lokal erfolgen und keine Randbedingungen berücksichtigt werden, aber ein allgemeinerer Aspekt von Kontextfreiheit wird nicht erfüllt. Für NCE Graphgrammatiken gilt nämlich zudem, dass dass der Ertrag eines Ableitungsbaums (vgl. Kapitel 3.7) unabhängig ist von der Reihenfolge der Ableitungsschritte.

3.3 Formale Definition der edNCE Graphgrammatiken

Zunächst wird die Definition der edNCE Graphgrammatiken angegeben, wobei die einzelnen Komponenten und ihre Funktionsweisen anschließend Schritt für Schritt erläutert werden.

Definition 3.3.1 (edNCE Graphgrammatik): *Seien Σ, Δ, Γ und Ω Alphabete, wobei $\Delta \subseteq \Sigma$ und $\Omega \subseteq \Gamma$. Eine edNCE Graphgrammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ ist gegeben durch die in den Alphabeten enthaltenen Namen für die Knoten, bzw. die Kanten, durch eine Menge von Produktionen P und einen initialen Graphen S. Für die Alphabete gilt, dass Σ die Knoten benennt, wobei Δ die terminalen Labels enthält und $\Sigma - \Delta$ die Nichtterminalen. Analog gilt dies für Γ und Ω, die Mengen der Kantenlabels.*
Die Produktionen der edNCE Graphgrammatiken sind von der Form $X \to (D, C)$. $X \in \Sigma - \Delta$ ist das nichtterminale Symbol eines Knotens, der von der Produktion ersetzt wird durch (D, C). Das Tupel (D, C) beschreibt dabei einen Graphen mit Einbettung*, wobei D ein Graph aus $GR_{\Sigma,\Gamma}$ und $C \subseteq \Sigma \times \Gamma \times \Gamma \times V_H \times \{in, out\}$ eine Menge von Verbindungsinstruktionen ist.*
Die Menge aller Graphen mit Einbettung über Σ und Γ wird mit $GRE_{\Sigma,\Gamma}$ bezeichnet. ◇

Beispiel: *Für eine Produktion $p : X \to (D, C)$ einer edNCE Graphgrammatik gilt (vgl. Abb. 7), dass ein Knoten mit dem nichtterminalen Symbol X ersetzt wird durch den Graphen mit Einbettung (D, C) (vgl. Abb. 7(a) und 7(b)). Die Verbindungsinstruktionen aus C geben dabei vor, wie D in den Restgraphen H^-, der durch das Entfernen des Knotens mit dem Symbol X und der entsprechenden Kanten entstanden ist, integriert werden soll. Für ein $c \in C$ mit $c = (\sigma, \beta/\gamma, y, in)$ gilt also, dass wenn eine Kante mit Label β von einem Knoten der Nachbarschaft mit Label σ zum Knoten x ging, eine neue Kante mit Label γ von dem Knoten der Nachbarschaft zum Knoten y des neuen Teilgraphen D eingeführt wird (vgl. Abb. 7(c)).*

(a) Ausgangsgraph

(b) Graph nach Ersetzen von X durch D

(c) Graph nach Anwendung von c

Abb. 7: Beispiel für die Anwendung einer Produktion c auf einen Graphen

Der im Beispiel aufgezeigte Ersetzungsprozess, der bereits aus Kapitel 2 bekannt ist, soll hier noch einmal formal definiert werden.

Definition 3.3.2 (Substitution): *Seien (H, C_H) und (D, C_D) zwei disjunkte Graphen mit Einbettung aus $GRE_{\Sigma,\Gamma}$ und sei $v \in V_H$ ein Knoten von H. Die Substitution von (D, C_D) für den Knoten v aus (H, C_H) wird kurz geschrieben als $(H, C_H)[v/(D, C_D)]$ und ergibt einen Graphen mit Einbettung $(G, C_G) \in GRE_{\Sigma,\Gamma}$,*

so dass gilt:

$$\begin{aligned}
V_G &= (V_H - \{v\}) \cup V_D, \\
E_G &= \{(x,\gamma,y) \in E_H \mid x \neq v, y \neq v\} \cup E_D \\
&\quad \cup \{(w,\gamma,x) \mid \exists \beta \in \Gamma : (w,\beta,v) \in E_H, (\lambda_H(w), \beta/\gamma, x, in) \in C_D\} \\
&\quad \cup \{(x,\gamma,w) \mid \exists \beta \in \Gamma : (v,\beta,w) \in E_H, (\lambda_H(w), \beta/\gamma, x, out) \in C_D\} \\
\lambda_G(x) &= \lambda_H(x) \textit{ wenn } x \in V_H - \{v\}, \textit{ und } \lambda(x) = \lambda_D(x) \textit{ wenn } x \in V_D, \\
C_G &= \{(\sigma,\beta/\gamma,x,d) \in C_H \mid x \neq v\} \cup \\
&\quad \{(\sigma,\beta/\delta,x,d) \mid \exists \gamma \in \Gamma : (\sigma,\beta/\gamma,v,d) \in C_H, (\sigma,\gamma/\delta,x,d) \in C_D\}. \quad \diamond
\end{aligned}$$

In der Definition wird das Aussehen des Graphen G angegeben nach der Ersetzung eines Knoten v im Graphen H durch einen Graphen mit Einbettung (D, C_D). Die Forderung nach zwei disjunkten Graphen erklärt sich durch die Vereinigung der zwei Knotenmengen $V_H - v$ und V_D zur neuen Knotenmenge V_G, wobei der ersetzte Knoten v natürlich nicht mehr enthalten ist.

Die Menge der Kanten in G besteht aus allen Kanten, die in D enthalten sind, und allen Kanten aus H, die nicht zu v inzident sind. Es fehlen noch die Kanten, die durch die Verbindungsinstruktionen hinzugefügt werden, die also die Graphen D^- und H zu G vereinen. In Worten beschreibt

$$\{(x,\gamma,w) \mid \exists \beta \in \Gamma : (v,\beta,w) \in E_H, (\lambda_H(w), \beta/\gamma, x, out) \in C_D\}$$

die Menge alle Kanten (w,γ,x), die durch eine Verbindungsinstruktion $(\lambda_H(w), \beta/\gamma, x, out)$ erzeugt werden, wobei eine Kante mit dem Label $\beta \in \Gamma$ und eine Kante (v,β,w) im Ursprungsgraphen H existierte. Die Abbildung der Knotenlabels auf die Knoten wird von den Teilgraphen übernommen, wobei wieder der ersetzte Knoten v berücksichtigt werden muss.

Die Menge C_G des neuen Graphen mit Einbettung besteht aus der Menge der Verbindungsinstruktionen von C_H, wobei alle die Instruktionen herausgenommen werden, die v betreffen. Hinzu kommen Instruktionen der Form $(\sigma,\beta/\delta,x,d)$, wenn zum einen für den Mutterknoten v eine Verbindungsinstruktion $(\sigma,\beta/\gamma,v,d)$ und zum anderen in C_D eine Instruktion $(\sigma,\gamma/\delta,x,d)$ existiert.

Ein Beispiel für die Substitution ist im nächsten Abschnitt zu finden, in dem eine graphische Notation für Graphen mit Einbettung eingeführt wird, die eine intuitive graphische Darstellung der Sachverhalte erlaubt.

3.4 Notationen und Konventionen für edNCE Graphgrammatiken

Bevor weitere formale Dinge der edNCE Graphgrammatiken eingeführt werden, wird zunächst eine graphische Darstellung präsentiert, die es ermöglicht die Sachverhalte eindeutig und leicht verständlich darzustellen.

3.4.1 Graphische Notation für Graphen mit Einbettung

Seien (H, C_H) und (D, C_D) zwei Graphen mit Einbettung über die Alphabete $\Sigma = \{X, Y, a, b, \sigma, \sigma'\}$ und $\Gamma = \{\alpha, \alpha', \beta, \beta', \gamma, \gamma', \gamma_1, \gamma_2, \delta, \delta'\}$, wobei für H und D gilt:

$$\begin{aligned}
V_H &= \{u, v, w\} \\
\lambda_H(u) &= Y, \lambda_H(v) = X, \lambda_H(w) = a \\
E_H &= \{(u, \beta, v), (v, \alpha, w), (w, \beta, u)\} \\
C_H &= \{(\sigma, \beta/\gamma, u, in), (\sigma, \beta/\gamma, v, out), (\sigma', \beta'/\gamma', v, in), (\sigma', \beta'/\alpha, w, in)\} \\
V_D &= \{x, y\} \\
\lambda_D(x) &= X, \lambda_D(y) = b \\
E_D &= \{(y, \alpha, x)\} \\
C_D &= \{(\sigma, \gamma, /\delta, y, out), (\sigma', \gamma'/\delta', x, in), (Y, \beta/\gamma_1, y, in)\} \cup \\
&\quad \{(Y, \beta/\gamma_2, x, in), (a, \alpha/\alpha', x, out)\}.
\end{aligned}$$

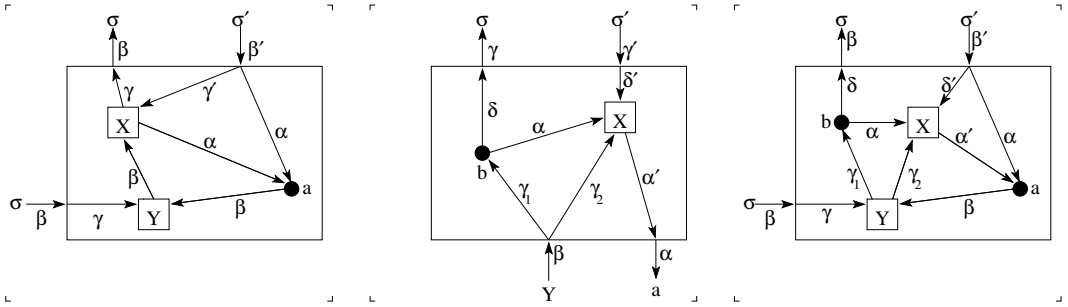
Die Definition dieser beiden Graphen ist recht umständlich und vor allem nicht sehr verständlich. Eine gute graphische Darstellung hilft, diese wesentlich besser zu verstehen. Dazu wird wie folgt vorgegangen. Ein Graph D wird in einem Kasten darstellt, der den Graphen von seiner Umgebung trennt. Die terminalen Knoten des Graphen werden durch Punkte gezeichnet, die nichtterminalen Knoten durch Vierecke, so ist in Abb. 8(a) der Graph H mit den Knoten mit Symbolen X, Y und a zu sehen. Die Kanten werden als Pfeile gezeichnet. Neu ist die Darstellung der Verbindungsinstruktionen durch Pfeile in die Umgebung des Graphen. Für die Verbindungsinstruktion $(\sigma, \beta/\gamma, u, in)$ aus C_H ist in Abb. 8(a) unten links die Kante, die von den Knoten der Umgebung mit Label σ zum Knoten u ($\lambda_H(u) = Y$) geht, zu sehen. An der Kante sind die beiden Label angegeben, wobei das innerhalb der Umgebung das alte und das ausserhalb des Systems das neue Label angibt.

In Abb. 8(c) ist die Substitution von Knoten v aus dem Graphen (H, C_H) durch den Graphen (D, C_D) (vgl. Abb. 8(b)) zu sehen, was kürzer mit $(H, C_H)[v/(D, C_D)]$ notiert wird. Der so entstandene Graph besteht aus den terminalen Knoten mit Labeln a und b und den Nichtterminalen X und Y. Neu hinzugekommen sind die Brückenkanten α', γ_1 und γ_2. Bei den Kanten in die Umgebung soll hier exemplarisch die Kante von b zu dem Knoten σ der Nachbarschaft betrachtet werden. Vor der Substitution existierte die Vorschrift, dass eine vom Knoten X ausgehende Kante β zu einem Knoten σ der Nachbarschaft durch eine Kante γ ersetzt werden soll. Im Tochtergraphen existiert weiter die Vorschrift, dass δ eine Kante γ ersetzt. Es ist dann klar, dass nach der Substitution die Verbindungsinstruktion existiert, die angibt, dass eine Kante β durch δ ersetzt wird.

3.4.2 Notationen bei Produktionen

Für Produktionen werden bestimmte Notationen verwendet, die das Beschreiben von Transformationen vereinfachen. So gibt $lhs(p)$ (lefthand-side) an, welcher Knoten durch die Produktion ersetzt wird, während $rhs(p)$ (righthand-side) den Graphen mit Einbettung, der für $lhs(p)$ eingesetzt wird, bezeichnet. Für eine Produktion $p : X \to (D, C)$ gilt also, dass $lhs(p) = X$ und $rhs(p) = (D, C)$.

Zwei Produktionen $X_1 \to (D_1, C_1)$ und $X_2 \to (D_2, C_2)$ sind isomorph, wenn $X_1 = X_2$ gilt, und wenn (D_1, C_1) und (D_2, C_2) isomorphe Graphen mit Einbettung



(a) (H, C_H) (b) (D, C_D) (c) $(H, C_H)\{v/(D, C_D)\}$

Abbildung 8: Substitution eines Graphen mit Einbettung

sind. Unter der Annahme, dass die Menge der Produktionen P einer Graphgrammatik, keine isomorphen Produktionen enthält, wird mit den Kopien von P *copy* (P) die (unendliche) Menge aller Produktionen beschrieben, die isomorph zu einer Produktion aus P sind.

3.4.3 Ableitungsschritte

Im Folgenden wird erklärt, wie die Menge der Produktionen benutzt wird, um nichtterminale Knoten eines Graphen zu ersetzen. Sei $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ eine edNCE Graphgrammatik, H und H' Graphen aus $GRE_{\Sigma,\Gamma}$, $v \in V_H$ und sei $p : X \to (D, C)$ eine Kopie einer Produktion von G, so dass D und H disjunkt sind. Wenn gilt $\lambda_H(v) = X$ und $H' = H[v/(D, C)]$ schreibt man $H \Rightarrow_{v,p} H'$, oder kurz $H \Rightarrow H'$. $H \Rightarrow_{v,p} H'$ beschreibt einen *Ableitungsschritt*, bei dem der Knoten v des Graphen H von der Produktion p ersetzt wird. Wichtig ist hier die Verwendung von Kopien von Produktionen, da diese dann mehrmals angewand werden können und nicht "verbraucht" werden, was der Fall wäre, wenn die Produktionen immer direkt angewendet würden. Das ist so zu verstehen, dass die Knoten, die im Graphen der linken Seite einer Produktion p ($lhs(p)$) sind, nicht mehrmals im gesamten Graphen vorhanden sein dürfen.

Eine Folge solcher Ableitungsschritte wird eine *Ableitung* genannt. Eine Ableitung $H_0 \Rightarrow_{v_1,p_1} H_1 \Rightarrow_{v_2,p_2} \ldots \Rightarrow_{v_n,p_n} H_n$, für ein $n \geq 0$, heisst produktiv ("creative"), wenn die Graphen H_0 und $rhs(p_i)$ für $1 \leq i \leq n$, wechselseitig disjunkt sind. Es gilt also, dass eine Produktion ein Ergebnis hat und ein Nichtterminal nicht einfach auf dasselbe Nichtterminal abgebildet wird (z.B. $p : X \to X$); eine Ableitung bringt somit eine Veränderung des Graphen mit sich. Da im Folgenden nur noch produktive Ableitungen betrachtet werden, wird $H \Rightarrow^* H'$ geschrieben, wenn eine solche, wie oben angegeben, existiert, wobei $H_0 = H$ und $H_n = H'$.

$sn(X, x)$ steht für einen "einfachen" Graphen mit Einbettung, der nur einen Knoten x mit Label X, keine Kanten und eine leere Menge von Verbindungsinstruktionen hat. Im Folgenden wird für jede edNCE Graphgrammatik angenommen, dass wenn $S \to (D, C)$ eine Produktion aus P ist, $C = \emptyset$ gilt. Jede Ableitung dieser Grammatik beginnt also mit $sn(S, s)$. Es ist klar, dass diese Annahme nicht einschränkt, da sich jede edNCE Graphgrammatik G durch Hinzufügen eines neuen initialen Nichtterminals S' und der Produktion $S' \to (D, \emptyset)$ in eine Grammatik G'

transformieren läßt, die diese Annahme erfüllt. Der Grund dieser Annahme liegt in der Möglichkeit Ableitungsbäume aufeinander aufzubauen. Ableitungsbäume sind Bestandteil des Unterapitles 3.7.

Ein *gültiges Ableitungsprodukt* ("sentential form") einer edNCE Graphgrammatik G ist ein Graph H, so dass gilt $sn(S, z) \Rightarrow^* H$ für ein beliebiges z, wobei H ein gegebener Graph ist ($H \in GR_{\Sigma,\Gamma}$).

3.4.4 Die von einer Graphgrammatik definierte Sprache

Für eine Graphgrammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ gilt, dass

$$L(G) = \{[H] \mid H \in GR_{\Delta,\Omega}, sn(S, z) \Rightarrow^* H, \text{für ein z}\}$$

die von G erzeugte *Graphsprache* ist. Für eine Graphgrammatik G ist die Graphsprache $L(G)$ also die Menge aller Graphen, welche nur terminale Knotenlabel und finale Kantenlabel besitzen und sich durch eine Ableitung aus $sn(S, z)$ bilden lassen.

Zwei edNCE Graphgrammatiken G und G' sind äquivalent, wenn für die Sprachen gilt

$$L(G) - \{\Lambda\} = (G') - \{\Lambda\},$$

wobei Λ den leeren Graphen bezeichnet.

3.4.5 Assoziativität der Substitution von Graphen mit Einbettung

Seien K, H und D drei wechselseitig disjunkte Graphen mit Einbettung. Sei weiter w ein Knoten von K und v ein Knoten von H, dann ist $K[w/H][v/D] = K[w/H[v/D]]$. Die Assoziativität der Substitution ist eins der zwei wichtigen Merkmale für kontextfreie Graphgrammatiken. Das zweite Merkmal, die Konfluenz, wird im folgenden Kapitel näher erläutert.

3.5 Konfluenz

Bei einigen edNCE Graphgrammatiken lassen sich Eigenschaften feststellen, welche die Grammatik als nicht-kontextfrei klassifizieren. Eine dieser Eigenschaften, auf welche in dieser Arbeit eingegangen wird, ist das Vorhandensein von blockierenden Kanten.

3.5.1 blockierende Kanten

Bei edNCE Grammatiken kann es vorkommen, dass zwei terminale Knoten mit einer nicht-terminalen Kante verbunden sind. Diese kann durch eine Produktionsregel nicht mehr entfernt werden. Der so entstandene Graph kann nun nicht mehr Teil einer Graphsprache sein, da in solchen Sprachen nur terminale Graphen, also Graphen mit ausschließlich terminalen Kanten und Knoten, vorhanden sein dürfen. Ein Lösungsansatz wäre, per Definition solche blockierenden Kanten zu verbieten. Eine nichtblockierende Graphgrammatik ist durch folgende Definition gegeben.

Definition 3.5.1 (Eine nichtblockierende Graphgrammatik): *Eine edNCE Graphgrammatik G heißt nichtblockierend, wenn für jedes gültige und terminale Ableitungsprodukt H von G gilt, dass in H nur finale Kanten enthalten sind.* ◇

Bevor im weiteren Verlauf dieser Diskussion die blockierenden Kanten erneut aufgegriffen werden, folgt zunächst die formale Definition der Eigenschaft der Konfluenz von Graphgrammatiken.

3.5.2 Statische und dynamische Konfluenz

Eine weitere, wichtige Eigenschaft welche edNCE Graphgrammatiken besitzen können, ist die Konfluenz. Wie schon erwähnt, bedeutet Konfluenz, dass die Reihenfolge, in der eine Menge von Produktionen angewandt wird, für den resultierenden Graphen keine Rolle spielt. Formal ausgedrückt heißt dies:

Definition 3.5.2 (Konfluente edNCE Graphgrammatiken): *Eine edNCE Graphgrammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ heißt konfluent, oder C-edNCE Graphgrammatik, wenn für alle Produktionen $X_1 \to (D_1, C_1)$ und $X_2 \to (D_2, C_2)$ in P, alle Knoten $x_1 \in V_{D_1}$ und $x_2 \in V_{D_2}$ und alle Kantennamen $\alpha, \delta \in \Gamma$, die folgende Äquivalenz gilt:*

$$\begin{array}{rcl} \exists \beta \in \Gamma : (X_2, \alpha/\beta, x_1, out) \in C_1 \quad und \quad (\lambda_{D_1}(x_1), \beta/\delta, x_2, in) \in C_2 \\ \Longleftrightarrow \\ \exists \gamma \in \Gamma : (X_1, \alpha/\gamma, x_2, in) \in C_2 \quad und \quad (\lambda_{D_2}(x_2), \gamma/\delta, x_1, out) \in C1 \end{array} \qquad \diamond$$

Um die Wichtigkeit dieser Eigenschaft zu unterstreichen, wird im folgenden Beispiel eine edNCE Graphgrammatik betrachtet, welche diese Eigenschaft nicht besitzt.

Beispiel: *Gegeben sei die edNCE Grammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ mit $\Sigma = \{S, A, B, a, b\}, \Delta = \{a, b\}, \Gamma = \{\alpha, \beta, \delta, \delta'\}, \Omega = \{\delta, \delta'\}$ und den drei Produktionen $X \to (D, C)$ (vgl. Abb. 9):*

$$\begin{array}{rcl} p_S & : & X = S, V_D = \{u, v\}, E_D = \{(u, \alpha, v)\}, \lambda_D(v) = B \text{ und} \\ & & C = \emptyset, \\ p_A & : & X = A, V_D = \{x\}, E_D = \emptyset, \lambda_D(x) = a \text{ und} \\ & & C = \{(B, \alpha/\beta, x, out), (b, \beta/\delta', x, out)\}, \\ p_B & : & X = B, V_D = \{y\}, E_D = \emptyset, \lambda_D(y) = b \text{ und} \\ & & C = \{(A, \alpha/\beta, y, in), (a, \beta/\delta, y, in)\}. \end{array}$$

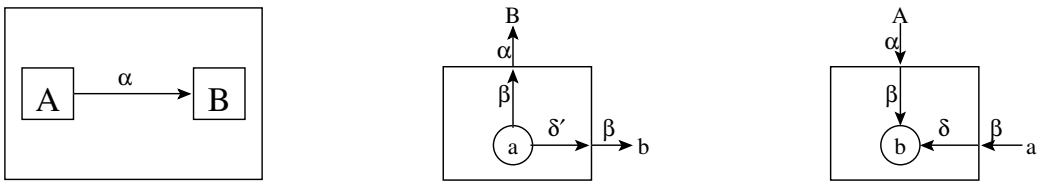


Abb. 9: Die Graphen S, A und B

Die Anwendung der Produktionen in der Reihenfolge p_S, p_A, p_B führt zu der Ableitung $sn(S, z) \Rightarrow_{z,p_S} H \Rightarrow_{u,p_A} H_1 \Rightarrow_{v,p_B} H_{12}$ (vgl. Abb. 10). Durch Änderung der Reihenfolge der Produktionen in p_S, p_B, p_A ensteht die Ableitung $sn(S, z) \Rightarrow_{z,p_S} H \Rightarrow_{u,p_B} H_2 \Rightarrow_{v,p_A} H_{21}$ (vgl. Abb. 11).

Abb. 10: Bildung des Graphen H_{12}

Abb. 11: Bildung des Graphen H_{21}

Die Unabhängigkeit der Reihenfolge der Regelanwendung gibt folgender Satz wieder.

Satz 3.5.1 (statische Konfluenz): *Eine edNCE Graphgrammatik* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *ist konfluent, dann und nur dann, wenn für jeden Graphen* $G \in GR_{\Sigma,\Gamma}$ *(oder* $H \in GRE_{\Sigma,\Gamma}$*) gilt: wenn* $H \Rightarrow_{u_1,p_1} H_1 \Rightarrow_{u_2,p_2} H_{12}$ *und* $H \Rightarrow_{u_2,p_2} H_2 \Rightarrow_{u_1,p_1} H_{21}$ *produktive Ableitungen in* G *sind und* $u_1, u_2 \in V_H$ *und* $u_1 \neq u_2$*, dann gilt* $H_{12} = H_{21}$*.* ◇

Beweis:
"⇒" Für diese Richtung ist zu zeigen, dass

$$H\left[u_1/\left(D_1, C_1\right)\right]\left[u_2/\left(D_2, C_2\right)\right] = H\left[u_2/\left(D_2, C_2\right)\right]\left[u_1/\left(D_1, C_1\right)\right]$$

gilt, wobei $p_i : X_i \to (D_i, C_i)$ *für* $1 \leq i \leq 2$*, und* H*,* D_1 *und* D_2 *wechselseitig disjunkt sind. Es muss also die Identität der Knoten und Kanten gezeigt werden.*

Aus der Definition der Substitution folgt, dass beide Graphen die gleiche Menge an Knoten besitzen. Bei beiden Graphen fallen sowohl u_1 *als auch* u_2 *aus Knotenmenge hinaus, da sie ersetzt werden. Vervollständigt werden die beiden Mengen jeweils um die Knoten aus den eingesetzten Tochtergraphen. Kanten* (v, γ, w) *mit* $(v, w), (w, v) \notin V_{D_1} \times V_{D_2}$ *sind ebenfalls identisch, da sie von den Ersetzungen nicht betroffen sind.*

Aus der obigen Definition 3.5.2 folgt zudem, dass sie dieselben Kanten (v, γ, w) *mit* $v \in V_{D_1}, w \in V_{D_2}$ *oder* $v \in V_{D_2}, w \in V_{D_1}$ *besitzen.*

"⇐" Um die Eigenschaft aus Definition 3.5.2 zu beweisen, wird ein Graph H *mit den Knoten* u_1 *und* u_2 *angenommen, wobei* u_i *mit* X_i *bezeichnet wird; Die beiden Knoten sind mit einer einzelnen Kante* (u_1, α, u_2) *verbunden. Sei* p_i *die Produktion* $p_i \to (D_i, C_i)$*. Da* H_{12} *und* H_{21} *als gleich gegeben sind, folgt aus der Definition der Konfluenz, dass es sich bei der gegebenen Grammatik um eine konfluente Graphgrammatik handelt.* □

In vielen anderen Lehrbüchern wird nicht eine statische sondern eine dynamische Konfluenz angegeben. Der Vollständigkeit halber wird auch diese hier definiert.

Lemma 3.5.1 (dynamische Konfluenz): *Eine edNCE Graphgrammatik* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *ist dynamisch konfluent, dann und nur dann, wenn für jede gültige Ableitung von* H *aus* G *gilt, dass wenn* $H \Rightarrow_{u_1,p_1} H_1 \Rightarrow_{u_2,p_2} H_{12}$ *und* $H \Rightarrow_{u_2,p_2} H_2 \Rightarrow_{u_1,p_1} H_{21}$ *produktive Ableitungen in* G *sind und* $u_1, u_2 \in V_H$ *und* $u_1 \neq u_2$ *gilt, dann* $H_{12} = H_{21}$. ◇

Es ist offensichtlich, dass jede statisch konfluente Graphgrammatik auch dynamisch konfluent ist, da bei der statischen Konfluenz gefordert ist, dass die Konfluenzeigenschaft für alle Graphen der Grammatik gelten muss, während bei der dynamischen Konfluenz die Konfluenzeigenschaft nur für jedes Ableitungsprodukt gefordert ist. Es gilt nicht notwendigerweise umgekehrt, dass jede dynamisch-konfluente Grammatik statisch konfluent ist, da jedes gültige Ableitungsprodukt in G auch ein Graph über den gleichen Alphabeten von Kanten- und Knotennamen ist, aber nicht jeder Graph notwendigerweise ein gültiges Ableitungsprodukt sein muss.

Zu Beginn des Unterkapitels wurde bereits das Problem der blockierenden KAnten angesprochen. Die Klasse der nichtblockierenden edNCE Graphgrammatiken stellt keine Lösung dar, wie in ([1] S. 33)gezeigt wird.

Bei C-edNCE Graphgrammatiken lässt sich eine Konstruktion angeben (vgl. [1] S.53), die für Graphgrammatiken, welche blockierende Kanten erzeugen, eine äquivalente Grammatik erzeugen, bei welcher dies nicht mehr der Fall ist.

3.5.3 Abgeschlossenheit unter Veränderung von Kantennamen

Wie auch bei Grammatiken über Zeichenketten, gelten für die Klasse der C-edNCE Graph-Sprachen die aus C-edNCE Grammatiken erzeugt werden, verschiedene Abgeschlossenheitsbedingungen. Diese sind besonders bei Beweisführungen ein wichtiges Mittel.

Exemplarisch wird hier die Abgeschlossenheit unter Kantenumbenennung gezeigt.

Sei ρ eine Kantenumbenennung, d.h. eine Abbildung $\rho : \Omega \to \Omega'$ bei welchem Ω und Ω' Kantennamen-Alphabete sind. Für einen Graphen $H \in R_{\Delta,\Omega}$ wird $\rho(H) \in GR_{\Delta,\Omega'}$ als der Graphe (V_H, E, λ_H) mit $E = \{(v, \rho(\gamma), w) \mid (v, \rho, w) \in E_H\}$ definiert. Für eine Graph-Sprache L gilt außerdem: $\rho(L) = \{[\rho(H)] \mid [H] \in L\}$.

Satz 3.5.2 (Abgeschlossenheit unter Kantenumbenennung): *Die Klasse der C-edNCE ist unter Kantenumbenennung abgeschlossen. Wenn* ρ *eine Kantenumbenennung ist und* $L \in$ *C-edNCE, dann gilt:* $\rho(L) \in$ *C-edNCE.* ◇

Der detailierte Beweis ist in der Literatur (vgl. [1] S. 37) zu finden.

3.6 Die Linksableitung und ihre Eigenschaften

Eine wichtige Eigenschaft von kontextfreien Grammatiken über Zeichenketten ist die Äquivalenz einer jeden Ableitung mit einer Linksableitung. Im Folgenden werden Linksableitungen im Kontext der Graphgrammatiken eingeführt und ihre Bedeutung in Verbindung zu C-edNCE-Grammatiken beschreiben.

3.6.1 lineare Ordnung

Da Zeichenketten eine natürliche Ordnung besitzen, dies bei Graphen aber nicht der Fall ist, wird vor der formalen Definition der Linksableitung eine lineare Ordnung

für Graphen eingeführt. Es ist offensichtlich, dass bei einer Definition der linearen Ordnung eine Ordnung der nichtterminalen Knoten ausreichend ist, da genau diese Knoten bei Ersetzungen betroffen sind. Aus Gründen der Übersichtlichkeit werden jedoch auch terminale Knoten mit in die lineare Ordnung einbezogen. Ein *geordneter Graph* ist ein Graph, dessen Knoten eine lineare Ordnung haben.

Beispiel: *Gegeben seien zwei Graphen G und H. Die Knoten von G seien geordnet durch $(v_1, \ldots, v_n)$, die Knoten von H durch $(w_1, \ldots, w_m)$. Die Ordnung des Graphen $G[v_i/H]$ ergibt sich dann zu $(v_1, \ldots, v_{i-1}, w_1, \ldots, w_m, v_{i+1}, \ldots, v_n)$.*

3.6.2 Linksableitung

Für die Definition der Linksableitung wird im Folgenden immer angenommen, dass die rechten Seiten der Produktionen der edNCE-Graphgrammatik geordnete Graphen sind. Jedes gültige Ableitungsprodukt ist also ebenfalls eine geordnete Graphgrammatik.

Definition 3.6.1 (Linksableitung): *Sei G eine edNCE Graphgrammatik. Für einen geordneten Graphen H ist ein Ableitungsschritt $H \Rightarrow_{v,p} H'$ aus G ein* links abgeleiteter Schritt*, wenn v in der Ordnung von H, der erste, mit einem nichtterminalen Symbol gekennzeichnete Knoten ist. Von einer Linksableitung wird gesprochen, wenn in allen Schritten links abgeleitet wurde. Eine Linksableitung von H zu H' wird mit $H \Rightarrow_{lm}^* H'$ bezeichnet.*
Die von solchen Grammatiken erzeugte Sprache ist $L_{lm}(G) = \{[H] \mid H \in GR_{\Delta,\Omega}$ und $sn(S, z) \Rightarrow_{lm}^* H$ für gegebene $z\}$, wobei die abstrakten Graphen $[H]$ keine Ordnung mehr enthalten.
Mit L-edNCE wird die Klasse der Graphsprachen beschrieben, welche durch Linksableitungen von edNCE Grammatiken erhalten werden. ◇

Den abstrakten Graphen wird keine Ordnung mehr zugewiesen, weil diese nur für die Linksableitung nötig war.

3.6.3 Äquvialenz der Linksableitung

Analog zu Linksableitungen von Zeichenketten gilt auch hier, dass jede Ableitung einer konfluenten edNCE-Grammatik in eine Linksableitung transformiert werden kann.

Satz 3.6.1 (Äquivalenz der Linksableitung): *Für jede C-edNCE Grammatik G gilt: $L_{lm}(G) = L(G)$* ◇

Beweis: *Sei $sn(S, z) \Rightarrow^* H$ eine Ableitung, welche keine Linksableitung ist und bei welcher H der Terminalgraph ist. Die Ableitung kann als*

$$sn(S, z) \Rightarrow \cdots \Rightarrow H_{i-1} \Rightarrow_{v_i,p_i} H_i \Rightarrow \cdots \Rightarrow H_{j-1} \Rightarrow_{v_j,p_j} H_j \Rightarrow \cdots \Rightarrow H$$

ausführlich geschrieben werden. Da es sich bei der Ableitung um eine nicht-Linksableitung handelt, wurde bei mindestens einem Ableitungsschritt nicht links abgeleitet. Sei nun $H_{i-1} \Rightarrow H_i$ die erste nicht-Linksableitung und v_j der Nichtterminal-Knoten von H_{i-1}, welcher am weitesten links steht. Würde es sich

bei der Ableitung um eine Linksableitung handeln, müsste also als nächstes genau dieser Knoten abgeleitet werden.

Durch j-i Anwendungen der dynamischen Konfluenz-Eigenschaft, beginnend mit $H_{j-2} \Rightarrow_{v_{j-1},p_{j-1}} H_{j-1} \Rightarrow_{v_j,p_j} H_j$ *entsteht eine Ableitung von* H *der gleichen Länge , bei welcher der erste nicht-Linksableitungs-Schritt nach der i-ten Ableitung auftritt. Der Knoten wurde also durch sukzessive Vertauschung zweier benachbarter Knoten, was aufgrund der Konfluenzeigenschaft möglich ist, an die erforderliche Stelle gebracht.*

Da es möglich sein kann, dass bei mehreren Nichtterminalen nicht links-abgeleitet wurde, führt die wiederholte Anwendung dieser Vertauschung schließlich zu einer Linksableitung von H*.* □

3.6.4 C-edNCE = L-edNCE

In Satz 3.6.1 wurde deutlich, dass zu jeder C-edNCE Grammatik eine äquivalente Linksableitung gebildet werden kann. Es gilt also C-edNCE $\subseteq$ L-edNCE. Nun gilt sogar, dass L-edNCE $\subseteq$ C-edNCE, was bedeutet, dass zu jeder edNCE Grammatik G eine C-edNCE Grammatik G' existiert, für die gilt $L(G') = L_{lm}(G)$.

Die Restriktion auf Linksableitungen ist also äquivalent zu der Restriktion auf konfluente Grammatiken, wobei Linksableitungen der übersichtlichen Notation dienen.

Satz 3.6.2: *L-edNCE = C-edNCE* ◇

Mit Satz 3.6.2 wurde eine erste Charakterisierung der Klasse C-edNCE erhalten. Weitere Charakterisierungsmölichkeiten finden sich in Kapitel 4.1.

3.7 Ableitungsbäume

Unter der Annahme, dass die rechten Seiten der Produktionen einer Graphgrammatik geordnet sind, können Ableitungsbäume analog zu denen von Grammatiken über Zeichenketten definiert werden. Bei letzteren stehen in den Knoten allerdings die abgeleiteten Symbole, bei Ableitungsbäumen von Graphgrammatiken aber die angewendeten Produktionen, bzw. Kopien dieser.

3.7.1 c-Knoten Ableitungsbäume

Im Folgenden werden c-Knoten Ableitungsbäumen eingeführt, in deren Knoten immer genau die Kopie der Produktion steht, welche angewendet werden soll. Im Gegensatz dazu stehen die p-Knoten Ableitungsbäume bei denen in einem Knoten angegeben wird welche Produktion angewendet werden soll.

Wichtig ist, dass im Kontext der Ableitungsbäume gut zwischen Knoten des Graphen und den Knoten des zugehörigen Ableitungsbaumes unterschieden wird.

Definition 3.7.1 (Der c-Knoten Ableitungsbaum): *Für eine edNCE Graphgrammatik* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *ist ein c-Knoten Ableitungsbaum* t *ein geordneter Baum mit Wurzel, bei dem in den Knoten Kopien von Produktionen aus* $copy(P)$ *stehen. Es gilt, dass die rechten Seiten der Kopien der Produktionen für jeden Knoten des Baumes wechselseitig disjunkt sein müssen, da in einem Graphen aus* $GR_{\Sigma,\Gamma}$ *kein Knoten zweimal vorkommen darf. Wenn ein Knoten* v *des Baumes* t *das*

Label $X \to (D, C)$ hat, dann sind die Kinder von v die nichtterminalen Knoten von D, wobei ihre Ordnung in t der Ordnung in D entspricht. Für jeden Nachfolger w eines Knotens gilt, dass die linke Seite der Produktion, die w im Baum t markiert, der linken Seite in D entspricht. ◇

Wenn die Wurzel eines Ableitungsbaumes t mit der Produktion $X \to (D, C)$ versehen ist, dann heißt t auch der Ableitungsbaum für X, wobei X nicht notwendigerweise ein initiales Nichtterminal sein muss. Ein Ableitungsbaum für X entspricht also einer Ableitung, die bei einem Nichtterminal X beginnt, die aber Teil einer größeren Ableitung sein kann. Es ist also zweckmäßig, dass X das Label eines Knotens eines Graphen mit Einbettung ist, der, durch die vom Ableitungsbaum t vorgegebenen Ableitung, einen Graphen mit Einbettung erzeugt.

$sin(X, x)$ bezeichnet den Graphen mit Einbettung (D, C_D), der nur einen Knoten $x \in V_D$ mit $\lambda_D(x) = X$ hat, dessen Kantenmenge E_D leer ist und dessen Menge der Verbindungsinstruktionen $C_D = \{(\sigma, \gamma/\gamma, x, d) \mid \sigma \in \Sigma, \gamma \in \Gamma, d \in \{in, out\}\}$ sei. $sin(X, x)$ ist die Erweiterung des initialen Graphen $sn(S, z)$, der in Kapitel 3.4.3 definiert wurde, um eine Menge von Verbindungsinstruktionen. Man hat mit $sin(X, x)$ die Möglichkeit, bei X beginnend, eine Ableitung durchzuführen und den resultierenden Graphen dann in eine "größere" Ableitung einzubetten. Trotz der Erweiterung von $sn(S, z)$ auf $sin(S, z)$ müssen $L(G)$ und $L_{lm}(G)$ nicht neu definiert werden, da die einzige Änderung die Hinzunahme von Verbindungsinstruktionen ist. Deswegen gilt $L(G) = \{[H] \mid H \in GR_{\Delta,\Omega}, sin(S, z) \Rightarrow^* H$ für gegebenes z$\}$. Es ist zu beachten, dass wenn ein Graph K einen Knoten v mit dem Label X hat, dass $K[v/sin(X, x)]$ isomorph zu K ist.

Es stellt sich jetzt die Frage nach dem Zusammenhang von Ableitungen und Ableitungsbäumen. Im Folgenden werden Konstruktionen gezeigt, die zwischen den beiden Darstellungen von Transformationen eines Graphen eine Verbindung aufbauen. Zu einer Ableitung $sin(X, v_1) \Rightarrow_{v_1,p_1} H_1 \Rightarrow_{v_2,p_2} \cdots \Rightarrow_{v_n,p_n} H_n$, wobei H_n ein terminaler Graph mit Einbettung ist, wird ein c-Knoten Ableitungsbaum t mit Knoten $v_1, \ldots, v_n$, die mit $p_1, \ldots, p_n$ bezeichnet werden, gebildet. Dabei sind die Kinder von v_i genau die geordneten, nichtterminalen Knoten von $rhs(p_i)$.

Für den Umwandlung des Ableitungsbaums in eine Ableitung muss zuerst der Baum traversiert werden, was in Preorder-Reihenfolge geschehen soll. Das bedeutet, dass der Baum rekursiv durchlaufen wird, wobei erst die Kindknoten betrachtet werden, bevor der Knoten selbst betrachtet wird (vgl. Abb. 12).

Abbildung 12: Ein Baum und seine Preorder-Traversierung

$v_1, \ldots, v_k$ wird als die Sequenz der Knoten des c-Knoten Ableitungsbaumes t angenommen, die erhalten wird, wenn der Baum in Preorder-Reihenfolge rekursiv durchlaufen wird. Dabei ist p_i das Label des Knoten v_i und $X = lhs(p_1)$. So kann auf dieselbe Art und Weise wie bei kontextfreien Grammatiken über Zeichenketten gezeigt werden, dass eine Linksableitung $sin(X, v_1) \Rightarrow_{v_1,p_1} H_1 \Rightarrow_{v_2,p_2} \cdots \Rightarrow_{v_k,p_k} H_k$ mit bestimmten $H_1, \ldots, H_k$ existiert, wobei H_k terminal sein soll.

Wie im Falle der kontextfreien Grammatiken über Zeichenketten kann gezeigt werden, dass die Komposition der beiden, oben angegebenen Abbildungen, von Linksableitungen auf c-Knoten Ableitungsbäume und umgekehrt, genau die Identität ergibt und diese sogar bijektiv ist. Das deutet auf die enge Verbindung zwischen Linksableitungen und c-Knoten Ableitungsbäumen für beliebige edNCE Graphgrammatiken hin.

Im Folgenden wird der Ertrag eines Ableitungsbaumes definiert, der genau das Ergebnis der zugehörigen Linksableitung eines Ableitungsbaumes liefert.

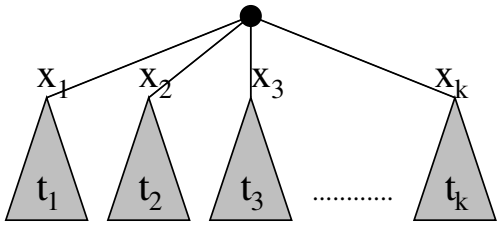


Abb. 13: Die Folge der Unterbäume des Baumes t

Definition 3.7.2 (Der Ertrag einer Ableitung): *Sei t ein c-Knoten Ableitungsbaum einer edNCE Graphgrammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ und sei $p \in copy(p)$ das Label der Wurzel des Baumes t. Ferner seien $x_1, \ldots, x_k$ die nichtterminalen Knoten von $rhs(p)$, in derselben Ordnung. Seien $t_1, \ldots, t_k$ die direkten Unterbäume von t, also die Unterbäume, die ihre Wurzel in $x_1, \ldots, x_k$ haben (vgl. Abb. 13). Dann ist der* Ertrag *des Baumes t der Graph mit Einbettung aus $GRE_{\Delta,\Gamma}$, der rekursiv definiert ist durch $yield(t) = rhs(p)[x_1/yield(t_1)] \ldots [x_k/yield(t_k)]$. Ferner ist*

$$Y(G) = \{[yield(t)] \in [GRE_{\Delta,\Omega}] \mid t \text{ ist ein c-Knoten Ableitungsbaum für } G \text{ von } S\}$$

der Ertrag einer Grammatik, also die Menge aller Graphen, die sich mit dieser Grammatik erzeugen lassen. ◇

Der Ertrag eines Baumes ergibt sich also rekursiv aus den Erträgen seiner Unterbäume. Zu beachten ist, dass der Ertrag eines Ableitungsbaum nicht in $GRE_{\Delta,\Omega}$ liegen muss, da blockierende Kanten vorhanden sein können.

Für einen gegebenen Ableitungbaum, führt die Linksableitung die Substitutionen in einem top-down Ansatz durch, während die Ertragsfunktion diese bottom-up ausführt. Dass diese beiden Ansätze denselben Graphen ergeben, liegt an der Assoziativität der Substitution (vgl. Lemma 3.4.5). Im Beweis des folgenden Lemmas, das die Einbettung einer Linksableitung in einen größeren Kontext zeigt, wird diese Assoziativität angewendet.

Lemma 3.7.1: *Sei $sin(X, x) \Rightarrow_{v_1,p_1} H_1 \Rightarrow_{v_2,p_2} \cdots \Rightarrow_{v_n,p_n} H_n$ eine Linksableitung einer edNCE Graphgrammatik G (wobei $v_1 = x$ und $rhs(p_1) = H_1$). Weiter sei H ein geordneter Graph, so dass x der erste nichtterminale Knoten von H ist und $\lambda_H(x) = X$. Dann gilt, dass auch $H \Rightarrow_{v_1,p_1} H[x/H_1] \Rightarrow_{v_2,p_2} H[x/H_2] \Rightarrow_{v_3,p_3} \cdots \Rightarrow_{v_n,p_n} H[x/H_n]$ eine Linksableitung von G ist.* ◇

BEWEIS: *Es gilt, dass* $H \Rightarrow_{v_1,p_1} H[x/H_1]$, *da* $rhs(p_1) = H_1$. *Wird der Linksableitungschritt* $H_{j-1} \Rightarrow_{v_j,p_j} H_j$, *für* $2 \leq j \leq n$ *betracht, dann gilt* $H_j = H_{j-1}[v_j/rhs(p_j)]$. *Es folgt, dass wegen der Assoziativität der Ableitung* $H[x/H_{j-1}] \Rightarrow_{v_j,p_j} H[x/H_{j-1}][v_j/rhs(p_j)] = H[x/H_{j-1}[v_j/rhs(p_j)]]$ *gilt. Dies entspricht einem Linksableitungsschritt, da für den Knoten* x *des Graphen* H *die Bedingungen der Linksableitung eingehalten wurden.* □

Dieses Lemma besagt also, dass eine Linksableitung, die bei $sin(X, x)$ beginnt, auch in eine größere Linksableitung integriert werden kann, wenn für den Graphen H gilt, dass x der erste nichtterminale Knoten in der Ordnung von H ist.

Auf die Verknüpfung von Ableitungsbäumen und Linksableitungen wurde schon eingegangen, folgender Satz zeigt die Korrektheit.

Satz 3.7.1: *Sei* $sin(X, v_1) \Rightarrow_{v_1,p_1} H_1 \Rightarrow_{v_2,p_2} \cdots \Rightarrow_{v_n,p_n} H_n$ *eine Linksableitung einer edNCE Graphgrammatik mit terminalem* H_n, *und sei* t *der zugehörige c-Knoten Ableitungsbaum. Dann gilt, dass* $yield(t) = H_n$. ◇

Der Ertrag eines Ableitungsbaumes entspricht also genau dem Graphen, der erhalten wird, wenn die zugehörige Linksableitung durchgeführt wird. Die Aussage des Satzes ist sogar auf $L_{lm}(G) = Y(G)$ erweiterbar, wobei G eine beliebige edNCE Graphgrammatik ist. Es gilt also, dass die Sprache einer Grammatik, die Linksableitungen verwendet, genau den Erträgen aller Ableitungsbäume für G entspricht. Insbesondere gilt nach Satz 3.6.1 auch, dass für jede C-edNCE Graphgrammatik G $L(G) = Y(G)$ ist. Für C-edNCE Graphgrammatiken hält Satz 3.7.1 sogar für beliebige Ableitungen, da Ableitungen die mit $sin(X, v)$ beginnen, in eine Linksableitung umgewandelt werden können, die denselben Ableitungsbaum hat.

Es stellt sich in diesem Zusammenhang die Frage, ob es für eine C-edNCE Graphgrammatik wichtig ist, dass in der Definition von $yield(t)$ die Graphen $yield(t_1) \cdots yield(t_k)$ von links nach rechts substituiert werden. Das folgende Lemma zeigt, dass dies nicht notwendig ist.

Lemma 3.7.2: *Sei* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *eine C-edNCE Graphgrammatik. Seien* u_1 *und* u_2 *zwei nichtterminale Knoten von* $H \in GRE_{\Sigma,\Gamma}$ *mit den Labeln* X_1 *und* X_2. *Seien* $K_1, K_2 \in GRE_{\Sigma,\Gamma}$, *so dass* $sin(X_i, u_i) \Rightarrow^* K_i$. *Dann gilt, dass* $H[u_1/K_1][u_2/K_2] = H[u_2/K_2][u_1/K_1]$. ◇

Mit diesem Lemma kann ein weiteres Lemma aufgestellt werden, das die Existenz von Ableitungsbäumen zum Ausdruck bringt, ohne dass diese explizit genannt werden. Aus diesem Grund wird es auch als das Lemma der Kontextfreiheit bezeichnet.

Lemma 3.7.3 (Das Lemma der Kontextfreiheit): *Sei* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *eine C-edNCE Graphgrammatik. Der Graph* $H \in GRE_{\Sigma,\Gamma}$ *habe die nichtterminalen Knoten* $v_1, \ldots, v_k$ *mit den Knotenlabels* $X_1, \ldots, X_k$ *und* $F \in GRE_{\Delta,\Gamma}$ *sei ein terminaler Graph. Dann gilt, dass* $H \Rightarrow^* F$ *dann und nur dann, wenn es Graphen mit Einbettung* $F_1, \ldots, F_k \in GRE_{\Delta,\Gamma}$ *gibt, so dass* $F = H[v_1/F_1] \ldots [v_n/F_n]$ *und* $sin(X_i, v_i) \Rightarrow^* F_i$ *für alle* $1 \leq i \leq k$. ◇

In diesem Abschnitt wurden die wichtige c-Knoten Ableitungsbäume und ihre Relevanz in Verbindung mit Ableitungsschritten näher betrachtet. Der Vollständigkeit halber werden nun ebenfalls die p-Knoten Ableitungsbäume kurz erwähnt, die aber heutzutage keine große Bedeutung mehr besitzen.

3.7.2 p-Knoten Ableitungsbäume

Definition 3.7.3 (p-Knoten Ableitungsbäume): *Sei* $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ *eine edNCE Graphgrammatik. Ein p-Knoten Ableitungsbaum von* G *ist ein geordneter Baum* t *mit Wurzel, bei dem die Knoten mit den Produktionen von* P *beschriftet sind. Es gilt, dass wenn der Knoten* v *von* t *das Label* p *hat, und* $rhs(p)$ *die nichtterminalen Knoten* $x_1, \ldots, x_k$ *in dieser Ordnung hat, dann hat* v *die Kinder* $v_1, \ldots, v_k$*. Die linke Seite der Label von* v_i *von* t *entspricht den Labeln von* x_i *in* $rhs(p)$*.*
Mit $D(G)$ *wird die Menge aller abstrakten p-Knoten Ableitungsbäume von* G *für* S *beschrieben.* ◇

Aus einem gegebenen p-Knoten Ableitungsbaum t_p kann ein c-Knoten Ableitungsbaum t_c gebildet werden durch die Wahl wechselseitig disjunkter Kopien der Produktionen, mit denen die Knoten von t beschriftet sind. Es sei angemerkt, dass alle c-Knoten Ableitungsbäume auf diese Art und Weise gebildet werden können. Es ist definiert, dass der Ertrag eines p-Knoten Ableitungsbaumes t durch den abstrakten Graph $yield(t) = [yield(ct)]$ gegeben ist. Diese Definition hängt nicht von der Wahl von t_c ab, da, wenn H, mit nichtterminalen Knoten $x_1, \ldots, x_k$ und H', ebenfalls mit nichtterminalen Knoten $x'_1, \ldots, x'_k$, isomorphe geordnete Graphen aus $GRE_{\Sigma,\Gamma}$ sind, $H[x_1/H_k] \ldots [x_k/H_k]$ und $H'[x'_1/H'_k] \ldots [x'_k/H'_k]$ isomorph sind. Dies gilt, wenn für $1 \leq i \leq k$, H_i und H'_i isomorphe Graphen aus $GRE_{\Sigma,\Gamma}$ gegeben sind. Für eine Produktion $p \in P$ gilt sogar, dass der Operator p als eine Operation über abstrakte Graphen aufgefasst werden kann, und der Ertrag eines abstrakten p-Knoten Ableitungsbaums t dann der Wert eines Ausdrucks t ist.

3.7.3 Anwendung von Ableitungsbäumen

Ableitungsbäume finden insbesondere Anwendung bei formalen Beweisen über edNCE Graphgrammatiken. Diese benötigen oft, im Gegensatz zu Beweisen welche den Schritten einer Ableitung direkt folgen, eine bottom-up Argumentation. Anstelle von Ableitungsbäumen kann auch das Lemma 3.7.3 benutzt werden. Beispiele für Beweise, die diese Methoden anwenden, finden sich in der Literatur (vgl. [1] S. 53f).

3.8 Unterklassen

Ähnlich wie bei Grammatiken über Zeichenketten, gibt es auch bei den konfluenten edNCE-Grammatiken Unterklassen, welche von besonderer Bedeutung sind. In dieser Arbeit werden die zwei wichtigen Klassen boundary-edNCE (B-edNCE) und linear-edNCE (lin-edNCE) sowie eine Erweiterung der B-edNCE näher behandelt. Es sei darauf hingewiesen, dass weitere Unterklassen, wie beispielsweise die Klasse der apex-edNCE (A-edNCE) existieren, hier aber nicht näher betrachtet werden. Besitzt eine Grammatik mehrere der erwähnten Eigenschaften, werden die Abkürzungen der Eigenschaften mit dem Zeichen $*$ verbunden. Eine Grammatik X*Y-edNCE besitzt also die Eigenschaft X und die Eigenschaft Y.

3.8.1 B-edNCE

Die Idee der Unterklasse der *boundary-edNCE* ist, dass keine nichtterminal-Symbole direkt miteinander verbunden sind. Jedes Nichtterminal hat somit eine Umgrenzung

(engl. *boundary*) von Terminalen in seiner direkten Nachbarschaft.

Definition 3.8.1 (B-edNCE): *Eine edNCE Grammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ heißt boundary oder B-edNCE Grammatik, wenn für jede Produktion $X \to (D, C)$ gilt:*

(B1) D besitzt keine Kanten zwischen nichterminalen Knoten und

(B2) C besitzt keine Verbindungsanweisungen $(\sigma, \beta/\gamma, x, d)$, für welche σ nichtterminal ist. ◇

Es kann gezeigt werden, dass eine der beiden Bedingungen ausreichend ist, um zu zeigen, dass es sich bei einer gegebenen Grammatik um eine B-edNCE Grammatik handelt. Zu einer edNCE Grammatik, welche eine der beiden Bedingungen erfüllt, kann also eine äquivalente Grammatik konstruiert werden, welche beiden Bedingungen genügt.

Aufbauend auf diese Unterklasse ist die Klasse der nichtterminalen-Nachbarschafts-deterministischen B-edNCE definiert.

Definition 3.8.2 (B_{nd}-edNCE): *Eine B-edNCE Grammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ heißt nichtterminal-Nachbarschafts-deterministisch oder B_{nd}-edNCE Grammatik, wenn für jede Produktion $X \to (D, C)$, jeden nichtterminalen Knoten $x \in V_D$ und alle $\gamma \in \Gamma$ und $\sigma \in \Delta$ gilt:*

(1) $\{y \in V_D | (y, \gamma, x) \in E_D \text{ und } \lambda_D(y) = \sigma\} \cup \{\beta \in \Gamma | (\sigma, \beta/\gamma, x, in) \in C\}$ ist einelementig oder leer sowie

(2) $\{y \in V_D | (y, \gamma, x) \in E_D \text{ und } \lambda_D(y) = \sigma\} \cup \{\beta \in \Gamma | (\sigma, \beta/\gamma, x, out) \in C\}$ ist einelementig oder leer. ◇

Eine B_{nd}-edNCE Grammatik besitz also neben den Eigenschaften der B-edNCE Grammatik zusätzlich die Eigenschaft, jeder Knoten nur mit unterschiedlich benannten Kanten mir anderen Knoten verbunden ist.

Idee hinter dieser Eigenschaft ist, dass die Nachbarn y eines nichtterminalen Knotens v eindeutig durch Beschriftung und Richtung der Kante zwischen v und y sowie das Label von y bestimmt werden können. Jeder Nachbar von v kann also bei einem Überschreiben von v durch den Einbettungsprozess unterschieden werden.

3.8.2 LIN-edNCE

Eine Unterklasse, welche auch bei Grammatiken über Zeichenketten oft untersucht wird, ist die Klasse der linearen Grammatiken.

Definition 3.8.3 (LIN-edNCE): *Eine edNCE Grammatik $G = (\Sigma, \Delta, \Gamma, \Omega, P, S)$ heißt linear oder LIN-edNCE Grammatik, wenn für jede Produktion $X \to (D, C)$ der Tochtergraph D maximal einen nichtterminalen Knoten besitzt.* ◇

Der Vorteil von linearen Graph-Grammatiken ist, analog zu linearen Zeichenketten-Grammatiken, Ableitungen auf übersichtliche Art und Weise darzustellen, da in jeder Produktionsregel D nur einen nichtterminalen Knoten enthält. Aufgrund dieser Eigenschaft ist jede LIN-edNCE Grammatik eine B-edNCE Grammatik.

3.9 Normalformen

Eine wichtige Eigenschaft von Grammatiken über Zeichenketten ist die eindeutige Darstellung eines Wortes durch Normalformen. Solche Darstellungen existieren auch für Graphgrammatiken, wie im Folgenden zu sehen ist..

Eine Vorstufe für Normalformen sind **reduzierte** C-edNCE Graphgrammatiken, für die gilt, dass sie nur die Nichtterminale enthalten, die in Ableitungen auch verwendet werden und die keine Λ- und Kettenproduktionen enthalten. Produktionen $p : X \to \Lambda$, wobei Λ der leere Graph ist, werden Λ-Produktionen genannt; Kettenproduktionen sind Produktionen der Form $p : X \to Y$, wobei $X, Y \in \Sigma - \Delta$. Der folgende Satz formuliert, dass sich zu jeder C-edNCE Graphgrammatik eine solche reduzierte Form bilden lässt.

Lemma 3.9.1 (Reduktion von C-edNCE): *Für jede C-edNCE Graphgrammatik kann eine äquivalente C-edNCE Graphgrammatik konstruiert werden, die weder Λ-, noch Kettenproduktionen enthält. Dasselbe gilt für alle Unterklassen von C-edNCE.* ◇

Ein Beweis für dieses Lemma, ist in der Literatur (vgl. [1] S. 62) zu finden.

3.9.1 Die Chomsky-Normalform

Die erste Normalform, welche hier betrachtet wird, ist die Chomsky-Normalform, welche von den Grammatiken über Zeichenketten übernommen wird. Eine C-edNCE Graphgrammatik G ist in Chomsky-Normalform, wenn für jede Produktion p von G die rechte Seite $rhs\,(p)$ entweder zwei Knoten, (wobei höchstens einer ein terminales Label tragen darf), oder einen terminalen Knoten hat (vgl. Abb. **??**).

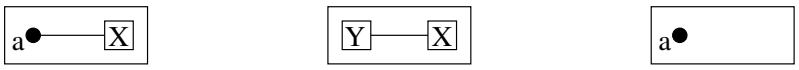
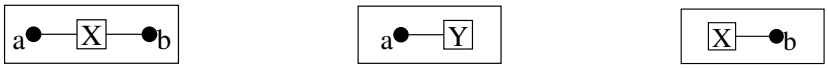


Abb. 14: $rhs(p)$ von Produktionen p in Chomsky-Normalform (ohne Verbindungsinstruktionen)

Satz 3.9.1 (Chomsky-Normalform): *Zu jeder C-edNCE Graphgrammatik läßt sich eine äquivalente C-edNCE Graphgrammatik in Chomsky-Normalform bilden. Dasselbe gilt für die Unterklassen B, B_{nd}, LIN und $LIN * B_{nd}$.* ◇

Im Folgenden wird kurz die Idee der Konstruktion erläutert, die eine beliebige C-edNCE Graphgrammatik in eine solche in Chomsky-Normalform abbildet. Für die ausführliche Konstruktion sei auf die Literatur hingewiesen (vgl. [1] S.63f). Sei G eine C-edNCE Graphgrammatik, die nicht in Chomsky-Normalform ist. Zunächst werden Λ- und Kettenproduktionen, wie in Lemma 3.9.1 beschrieben, entfernt. Für eine Produktion $p : X \to (D, C)$, die die Regeln der Chomsky-Normalform verletzt, werden durch die Konstruktion zwei neue Produktionen $p_1 : X \to (D_1, C_1)$ und $p_2 : Y \to (D_2, C_2)$ eingefügt, wobei p_1 die gewünschte Form hat und D_2 sich durch die Entfernung eines Knotens von D ergibt (vgl. Abb. 15). Falls nötig wird auf D_2 diese Konstruktion erneut (Rekursion) angewendet.



(a) D (b) D_1 (c) D_2

Abb. 15: Die Graphen D, D_1 und D_2

3.9.2 Die Operator-Normalform

Eine C-edNCE Graphgrammatik G ist in Operator-Normalform (ONF), wenn die rechten Seiten aller Produktionen von G genau einen terminalen Knoten enthalten. Für eine Produktion $p : X \to (D, C)$ gilt also, dass in λ_D genau ein Knoten auf ein terminales Symbol abbildet ($\lambda_D(x) = \delta$, mit $x \in V_D$ und $\delta \in \Delta$). Für einen Graphen $H \in L(G)$, wobei H in ONF sei, gilt also, dass wenn $|V_H| = n$ genau n Ableitungsschritte gemacht wurden.

Auf den ersten Blick kann vermutet werden, dass für jede Graphgrammatik G aus C-edNCE ein Äquivalent in ONF exisitiert. Diese Vermutung ist bis jetzt nicht bewiesen. Für die B-edNCE Graphgrammatiken dagegen gibt es eine bewiesene Konstruktion, die die B_{nd} Eigenschaft erhält. Bei linearen Graphgrammatiken ist ersichtlich, dass jede Grammatik aus LIN-edNCE in Chomsky-Normalform, automatisch auch in ONF ist.

3.9.3 Die Nachbarschaft erhaltende Normalform

Die Nachbarschaft erhaltende Normalform fordert genau das, was ihr Name ausdrückt. Die Kanten, welche zum Mutterknoten inzident sind, gehen durch die Substitution des Mutterknotens nicht verloren, d.h. nach der Substitution ist die Nachbarschaft genauso gross, wie vorher.

Die Existenz dieser Normalform wurde von verschiedenen Wissenschaftlern für B-NLC, B-edNCE und C-edNCE bewiesen. Zudem kann für jede edNCE Graphgrammatik aus einer dieser Klassen auch eine äquivalente C-edNCE Graphgrammatik konstruiert werden, die die Eigenschaft hat, dass Nachbarschaften erhalten bleiben.

Die hier vorgestellten Normalformen dienen, in Analogie zu Normalformen bei Zeichenketten, der eindeutigen Darstellung von Grammatiken. Durch Algorithmen, welche gegebene Grammatiken in die jeweiligen Normalformen umwandeln, lassen sie sich leicht bestimmen und sind auch bei Beweisen von großer Bedeutung.

4 Eigenschaften von NCE

In diesem Kapitel wird zunächst ein Überblick über verschiedene Beschreibungsmöglichkeiten der konfluenten edNCE-Grammatiken gegeben. Diesen Charakterisierungen folgend, wird in Abschnitt 4.2 die Komplexität des Zuordnungsproblems einer edNCE Graphsprache kurz erläutert.

4.1 Charakterisierungen

Es gibt verschiedene Arten C-edNCE Grammatiken zu charakterisieren. Neben den bereits in Abschnitt 3.6 beschriebenen Linksableitungen, gibt es noch die Beschreibungsmethode des regulären Pfads, die logische Beschreibung, das Handle Replacement sowie die Beschreibung durch Graph-Ausdrücke.

Bei der Charakterisierung anhand von regulären Pfaden, wird der baumähnliche Aufbau der C-edNCE Graphen ausgenutzt. Dazu werden die Knoten des Graphen H als Untermenge der Knotenpunkte eines regulären Baumes t definiert. Der Baum darf also nur durch Verbindungen zwischen Vorfahren und Nachfahren gegeben sein. Eine Kante zwischen zwei Knoten u und v aus H ist definiert durch das Wort der Baumknoten des kürzesten Pfades zwischen u und v, welches einer regulären Sprache angehört.

Eine andere Möglichkeit die Klasse der C-edNCE-Grammatiken zu beschreiben, ist die logische Charakterisierung mit Hilfe der monadischen Logik zweiter Ordnung (MSO: *monadic second order logic*). Bei dieser Beschreibungsform werden Variablen, Terme sowie Formeln verwendet, welche rekursiv definiert sind Diese Beschreibungsart ist sehr nützlich, da sie zu einigen Abgeschlossenheitsbedingungen und Entscheidbarkeitsregeln für C-edNCE führt. Des Weiteren können mit der MSO Funktionen auf Graphen beschrieben werden.

Die S-HH (*seperated handle hypergraph*) Grammatik stellt eine weitere Möglichkeit dar, C-edNCE Grammatiken über Graph-Grammatiken zu charakterisieren. Im Gegensatz zur NCE-Grammatik, werden bei der S-HH Grammatik nicht Knoten, sondern sogenannte *handle* ersetzt. Ein *handle* ist dabei eine Kante mit den beiden verbundenen Knoten, sowie alle Kanten, die mit diesen beiden Knoten verbunden sind. Somit arbeitet die S-HH Grammatik sowohl mit Knoten- als auch mit Kantenersetzung.

Als letzte Charakterisierungsmöglichkeit gibt es Graph-Ausdrücke. Bei diesen beschreibt eine reguläre Baumsprache die C-edNCE-Grammatik. Die Grammatik wird hierbei genau von der Graphenmenge beschrieben, die in der Algebra aller abstrakten Graphen mit Enbettung enthalten sind.

4.2 Entscheidbarkeit

Ein wichtiges Problem der Entscheidbarkeit ist, analog zu Grammatiken über Zeichenketten das Erkennungs- oder Wortproblem. Bei Zeichenkettengrammatiken ist für ein gegebenes Wort zu entscheiden, ob es aus einer gegebenen Grammatik konstruierbar bzw. innerhalb einer gegebenen Sprache liegt. In Analogie wird bei diesem Problem in der Graphentheorie entschieden, ob ein gegebener Graph aus einer gegebenen Graphgrammatik konstruierbar ist.

Zur Lösung dieses Problems wird im Folgenden von einer C-edNCE Grammatik ohne Λ-Produktionen sowie ohne Kettenproduktionen ausgegangen. Jede Ableitung eines Graphen mit n Knoten hat also maximal eine Länge von $2n$. Jede C-edNCE liegt laut Literatur (vgl [1] S.82) demzufolge in $NPTIME$. Eine gegebene Ableitung zu schätzen benötigt analog n^2 Speicher. Es handelt sich demzufolge um einen polynomiellen Algorithmus.

5 Zusammenfassung

In dieser Arbet wurden Grapphgrammatiken untersucht, bei welchen Knoten durch einen neuen Untergraphen ersetzt werden.

Nach einer Einführung der Notation an Hand der NLC-Grammatik (*Node Labeled Controlled*), wurden Erweiterungen dieser Graphgrammatik besprochen und in Kapitel 3 die wichtige Klasse der NCE-Grammatik (*Neighborhood Controlled Embedding*) eingeführt. Bei dieser Grammatik können zu ersetzende Knoten durch Berücksichtigung der Nachbarschaft genauer angegeben werden. Mit dieser Klasse von Graphgrammatiken wurden zentrale Begriffe wie Produktionen und Ableitungsschritte erläutert. In Abschnitt 3.5 wurde die Eigenschaft der Konfluenz erläutert. Konfluente NCE-Grammatiken sind hinsichtlich der Reihenfolge der Ersetzungen gleich und bilden die wichtigste Klasse dieser Arbeit. Neben der statischen Definition wurde die dynamische Definition der Konfluenz betrachtet und die Abgeschlossenheitsbedingungen von konfluenten Sprachen unter Kantenumbennenung exemplarisch eingeführt. In Abschnitt 3.6 wurde die Linksableitung als Charakterisierung von konfluenten edNCE-Grammatik eingeführt. Die Äquivalenz von Linksableitungen zu C-edNCE Grammatiken wurde ebenfalls erläutert. Im Bereich der Ableitungsbäume wurden sowohl die c-Knoten Ableitungsbäume als auch die p-Knoten Ableitungsbäume betrachtet und Ihre Wichtigkeit unterstrichen. In Abschnitt 3.8 wurden als wichtige Unterklasse die boundary-edNCE als C-edNCE ohne Kanten zwischen Nichtterminalen, sowie die Unterklasse der linearen-edNCE betrachtet. Das Kapitel wurde mit der Erläuterung verschiedener Normalformen abgeschlossen. Hier wurden neben der Chomsky- und der Operator-Normalform auch die Nachbarschaft-erhaltende-Normalform diskutiert.

Die Ausarbeitung wurde von einem Ausblick auf verschiedene Charakterisierungsformen der C-edNCE sowie der Komplexität der C-edNCE-Grammatiken abgeschlossen.

Literatur

[1] **Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1-3**; H. Ehrig, H.-J. Kreowski, U. Montanari, G. Rozenberg; World Scientific; ISBN 981-02-4021-X;

[2] **Skript zur Vorlesung Graphgrammatiken**; Prof. Dr. Franz J. Brandenburg; Passau, 2002; http://www.infosun.fmi.uni-passau.de/br/lehrstuhl/Kurse/gg_02/