Masterarbeit
Abschlusseigenschaften für Graph-Sprachen mit Anwendungen auf Terminierungsanalyse

Sebastian Küpper

10. August 2012

Betreuer: Prof. Dr. Barbara König
Dr. H.J. Sander Bruggink
Tag der Anmeldung: 05. April 2012
Tag der Abgabe: 13. August 2012

Inhaltsverzeichnis

Teil I

Grundlegende Definitionen

Kapitel 1

Einleitung

Graphtransformationssysteme sind ein mächtiges Werkzeug zur Systemmodellierung und -analyse in der Informatik [18]. Aber auch abseits der Systemmodellierung haben Graphtransformationssysteme interessante Anwendungen, so können sie beispielsweise zur Analyse neuronaler Netzwerke [7] verwendet werden oder zur Optimierung im Compilerbau, sofern eine Graph-basierte Zwischensprache verwendet wird [5]. Außerhalb der Informatik wird Graphtransformation ebenfalls, beispielsweise zur Beschreibung von Strukturen in der Mikrobiologie verwendet [17].

Graphtransformationssysteme bieten Regelsysteme, die veränderliche Graphstrukturen beschreiben können. So kann man beispielsweise den Zustand eines Programms als Graphen ausdrücken und ein Graphtransformationssystem angeben, das die Verhaltensweise eines Programms auf einem Zustandsgraphen beschreibt. Die Ausdrucksmächtigkeit von Graphtransformationssytemen ist hoch, was sie zu angenehmen Modellierungswerkzeugen macht. Allerdings geht mit der hohen Ausdrucksmächtigkeit auch eine hohe Komplexität einher. Graphtransformationssysteme sind Turing-vollständig, das heißt, man kann auf Graphtransformationssystemen (zusammen mit sequentieller Komposition der Regeln und der Möglichkeit der Iteration) allgemeine Turingmaschinen simulieren, wie Annegret Habel und Detlef Plump, [10], gezeigt haben.

Das hat zur Folge, dass viele Fragen bezogen auf ein Graphtransformationssystem unentscheidbar sind. So ist zu einem gegebenen Graphtransformationssystem und einem Startgraphen die Frage, ob dieses System auf dem Startgraphen jemals terminiert, also keine Regelanwendung mehr möglich ist, unentscheidbar, auch viele andere Unentscheidbarkeitsresultate lassen sich auf den Fall der Graphtransformationssysteme übertragen. Um dennoch relevante Aussagen über Graphtransformationssysteme treffen zu können, bedarf es approximativer Ansätze. Statt eines Entscheidungsverfahrens, ob ein Graphtransformationssystem auf einem gegebenen Startgraphen jemals

terminiert, kann man beispielsweise ein Semientscheidungsverfahren oder ein Verfahren, das nur einseitig Fehler macht, untersuchen.

Für Stringersetzungssysteme wurde von Geser et al., [8], ein erfolgreiches Semi-Entscheidungsverfahren umgesetzt. Ziel dieser Arbeit ist es nun, die Möglichkeiten der Umsetzung dieses Verfahrens auf das Setting der Graphtransformationssysteme zu untersuchen. Der Vorteil dieses Ansatzes ist es, dass für Graphtransformationssysteme viele zu Stringersetzungssystemen sehr ähnliche Konzepte und Eigenschaften existieren. Daher wird in dieser Arbeit die Möglichkeit untersucht, ein Verfahren in Anlehnung an das Verfahren aus [8] zu entwerfen, das für ein gegebenes Graphtransformationssystem nachweist, dass es Match-beschränkt ist. Gelingt der Nachweis, dass ein System Match-beschränkt ist, so folgt hierdurch, dass das System auch terminierend ist. Die Konstruktion im Stringersetzungsfall basiert allerdings auf einem Satz, der den Erhalt der Eigenschaft der Regularität einer Sprache betrifft. Der Begriff der regulären Sprache wurde auf den Graphenfall verallgemeinert durch den Begriff der Erkennbarkeit.

Diese Arbeit ist in drei Teile unterteilt. Im ersten Teil werden wir die Grundlagen legen, insbesondere werden wir den Begriff der erkennbaren Pfeilsprache und der erkennbaren Graphsprache kennen lernen. Hierzu werden wir einige Grundbegriffe der Kategorientheorie verwenden, die ebenfalls im ersten Teil definiert werden.

Betrachtet man den Algorithmus aus [8], so basiert er stark auf der Eigenschaft des Erhalts der Regularität. Um eine vergleichbare Eigenschaft im Graphen-Fall zu erhalten, müssen wir jedoch zunächst einmal eine Reihe von Abschlusseigenschaften untersuchen. Für reguläre Sprachen ist bekannt, dass diese abgeschlossen sind unter Substitution, der Restriktion und der Anwendung invers kontextfreier Regeln - diese drei Abschlusseigenschaften werden in [12] verwendet, um den Erhalt der Regularität zu beweisen. Weiterhin sind reguläre Sprachen abgeschlossen unter Konkatenation, Anwendung löschender Regeln, sowie unter der Anwendung des Kleene-Sterns [3]. In der Vergangenheit [4] wurde bereits gezeigt, dass die erkennbaren Graphsprachen abgeschlossen sind unter Vereinigung, Schnitt (und somit auch Restriktion) und Komplementbildung. Die Konstruktion ist in diesem Fall ähnlich zu den bekannten Konstruktionen für reguläre Sprachen, so dass in dieser Arbeit nicht näher auf die entsprechenden Beweise eingegangen wird. Im zweiten Teil der Arbeit untersuchen wir daher, welche der Abschlusseigenschaften sich auch auf den Fall erkennbarer Graphsprachen übertragen lassen.

Im dritten Teil werden wir schließlich zunächst den Algorithmus zur Terminierungsanalyse für Stringersetzungssysteme kennen lernen. Hierzu werden wir die Begriffe der löschenden und der Match-beschränkten Stringerset-

zungssysteme kennen lernen und alle für den Analyseansatz wichtigen Konstruktionsschritte und Sätze zitieren. Da die Beweise der Aussagen allerdings bereits in [8] und [12] vorgestellt wurden, verzichten wir in diesem Abschnitt auf die Beweise der Aussagen.

Im zweiten Kapitel des dritten Teils werden wir dann die Begriffe der löschenden und Match-beschränkten Graphtransformationssysteme in Analogie zu löschenden und Match-beschränkten Stringersetzungssystemen definieren und einige Eigenschaften dieser untersuchen. Schließlich werden wir untersuchen, ob löschende Graphtransformationssysteme die Erkennbarkeit erhalten, was im String-Setting eine wesentliche Voraussetzung dafür ist, dass das vorgestellte Verfahren zur Überprüfung auf Terminierung ein Semi-Entscheidungsverfahren ist.

Kapitel 2

Grundbegriffe der Kategorientheorie

Die in dieser Arbeit verwendeten Graphsprachen basieren auf Begriffen aus der Kategorientheorie. Diese dient unter Anderem als Verallgemeinerung der Mengenlehre. Zunächst ist zu klären, was eine Kategorie überhaupt ist, außerdem benötigen wir zumindest die Definitionen des Pushouts und des Cospans, um schließlich den Begriff der Erkennbarkeit definieren zu können. Die Definitionen in diesem Kapitel folgen [16], die Darstellung der ersten drei Definitionen ist nahezu wörtlich [14] entnommen.

Definition 2.1 ***(Kategorie)***
Eine Kategorie $\boldsymbol{C}$ *ist ein Tupel aus einer Sammlung von Objekten* O*, einer Sammlung von Pfeilen* P *(oder auch Morphismen) und einem Kompositionsoperator* $\circ$ *für die gilt:*
Jedem Pfeil f *ist ein Definitionsbereich* $dom(f) \in O$ *und ein Bildbereich* $cod(f) \in O$ *zugeordnet. Für einen Pfeil* f *mit* $dom(f) = A$ *und* $cod(f) = B$ *schreiben wir auch* $f : A \to B$*. Der Kompositionsoperator* $\circ$ *verknüpft zwei Pfeile* f *und* g *genau dann als* $g \circ f$*, wenn* $cod(f) = dom(g)$*. Für den Kompositionsoperator gilt das Assoziativgesetz, das heißt für drei Pfeile* $f : A \to B$*,* $g : B \to C$*,* $h : C \to D$ *gilt*

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

Zudem gibt es für jedes Objekt $A \in O$ *einen Identitätspfeil* $id_A : A \to A$*. Die Identitätspfeile genügen dem Identitätsgesetz, das heißt es gilt für jeden Pfeil* $f : A \to B$*:*

$$id_B \circ f = f$$

und

$$f \circ id_A = f.$$

Eines der wichtigsten Konzepte der Kategorientheorie ist der Funktor, eine strukturerhaltende Abbildung zwischen verschiedenen Kategorien.

Definition 2.2 ***(Funktor)***
Seien $\boldsymbol{C}$ *und* $\boldsymbol{D}$ *Kategorien, dann ist ein Funktor* $\mathcal{F} : \boldsymbol{C} \to \boldsymbol{D}$ *eine Abbildung, die jedes* $\boldsymbol{C}$*-Objekt* A *auf ein* $\boldsymbol{D}$*-Objekt* $\mathcal{F}(A)$ *abbildet und jeden* $\boldsymbol{C}$*-Pfeil* $f : A \to B$ *auf einen* $\boldsymbol{D}$*-Pfeil* $\mathcal{F}(f) : \mathcal{F}(A) \to \mathcal{F}(B)$ *abbildet, so dass für alle* $\boldsymbol{C}$*-Objekte* A *und konkatenierbare* $\boldsymbol{C}$*-Pfeile* f *und* g *gilt:* $\mathcal{F}(id_A) = id_{\mathcal{F}(A)}$ *und* $\mathcal{F}(g \circ f) = \mathcal{F}(g) \circ \mathcal{F}(f)$.

Definition 2.3 ***(Pushout)***
Seien $f : A \to B$ *und* $g : A \to C$ *Pfeile einer Kategorie. Ein Pushout dieser Pfeile ist ein Objekt* P *- das Pushoutobjekt - und zwei Pfeile* $f' : C \to P$ *und* $g' : B \to P$*, so dass* $f' \circ g = g' \circ f$ *und wenn es ein Objekt* X *und zwei Pfeile* $f'' : C \to X$ *und* $g'' : B \to X$ *gibt, für die ebenfalls* $f'' \circ g = g'' \circ f$ *gilt, dann gibt es genau einen Pfeil* $k : P \to X$*, so dass* $f'' = k \circ f'$ *und* $g'' = k \circ g'$*. Vgl. auch folgende Abbildung:*

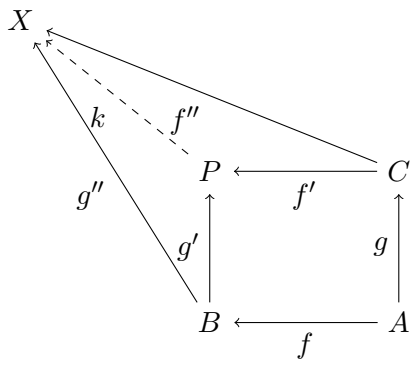


Abbildung 2.1: Pushout

Der hierzu duale Begriff des Pullbacks wird in dieser Arbeit ebenfalls eine wichtige Rolle spielen.

Definition 2.4 ***(Pullback)***
Seien $f : A \to C$ *und* $g : B \to C$ *Pfeile einer Kategorie. Ein Pullback dieser Pfeile ist ein Objekt* P *- das Pullbackobjekt - und zwei Pfeile* $f' : P \to B$ *und* $g' : P \to A$*, so dass* $f \circ g' = g \circ f'$ *gilt. Weiterhin, wenn es ein Objekt* X *und zwei Pfeile* $f'' : X \to B$ *und* $g'' : X \to A$ *gibt, für die ebenfalls* $g \circ f'' = f \circ g''$ *gilt, dann gibt es genau einen Pfeil* $k : X \to P$*, so dass* $f'' = f' \circ k$ *und* $g'' = g' \circ k$*. Vgl. auch folgende Abbildung:*

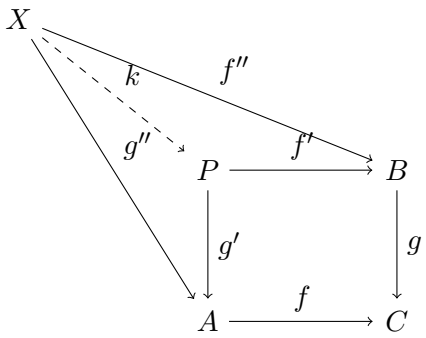


Abbildung 2.2: Pullback

Definition 2.5 ***(Pushout-Komplement)***
Gegeben seien zwei Pfeile $f : A \to B$ und $g : B \to C$. Wir nennen zwei Pfeile $h : A \to D$ und $k : D \to C$ das Pushout-Komplement der Pfeile f und g, falls das Diagramm, das die vier Pfeile enthält, einen Pushout bildet.

Wichtig für das Verständnis von Graphsprachen ist schließlich noch der Begriff des Cospans. Zur Definition des Cospan-Begriffs benötigen wir allerdings noch den Begriff des Isomorphismus, vgl. [16]:

Definition 2.6 ***(Isomorphismus)***
Einen Pfeil $f : A \to B$ nennen wir einen Isomorphismus, wenn es einen inversen Pfeil (oder kurz ein Inverses) $f^{-1} : B \to A$ gibt, so dass $f^{-1} \circ f = id_B$ sowie $f \circ f^{-1} = id_A$ richtig sind.
Weiterhin nennen wir zwei Objekte A und B isomorph, wenn es einen Isomorphismus $f : A \to B$ gibt.

Definition 2.7 ***(Kommutierendes Diagramm)***
Wir sagen ein Diagramm (eine Darstellung von Objekten und Pfeilen einer Kategorie) kommutiert, falls für jedes Paar von dargestellten Sequenzen von konkatenierbaren Pfeilen $f_1, ..., f_n$ und $g_1, ..., g_m$ mit $dom(f_1) = dom(f_2)$ und $cod(f_n) = cod(g_m)$ gilt

$$f_n \circ f_{n-1} \circ ... \circ f_1 = g_m \circ g_{m-1} \circ ... \circ g_1.$$

Die nachfolgende Definition ist [4] entnommen.

Definition 2.8 ***(Cospan und Cospan-Kategorie)***
Sei $\boldsymbol{C}$ eine Kategorie in der für alle Paare von Pfeilen mit gleichem Definitionsbereich der Pushout existiert. Ein Cospan ist ein Paar von Pfeilen aus dieser Kategorie (c_L, c_R) mit dem gleichen Bildbereich, wir schreiben:

$$c : J \overset{c_L}{\to} G \overset{c_R}{\leftarrow} K.$$

Wir nennen zwei Cospans

$$c_1 : J \overset{c_{1_L}}{\rightarrow} G \overset{c_{1_R}}{\leftarrow} K$$

und

$$c_2 : J \overset{c_{2_L}}{\rightarrow} H \overset{c_{2_R}}{\leftarrow} K$$

äquivalent wenn es einen Isomorphismus h zwischen G und H gibt, so dass das folgende Diagramm kommutiert:

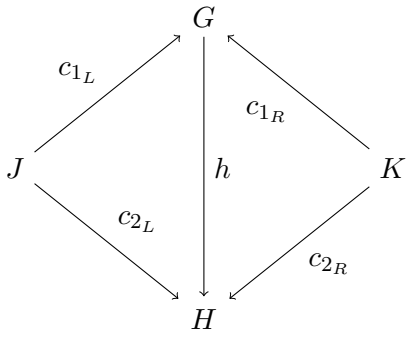


Abbildung 2.3: Isomorphie-Bedingung

Wir identifizieren im Folgenden die Äquivalenzklassen von Cospans, die durch die Isomorphie-Definition gebildet werden, mit ihren Repräsentanten.
Die Konkatenation zweier Cospans $c_1 : J \overset{c_{1_L}}{\rightarrow} G \overset{c_{1_R}}{\leftarrow} M$ und $c_2 : M \overset{c_{2_L}}{\rightarrow} H \overset{c_{2_R}}{\leftarrow} K$ bilden wir wie folgt. Sei M' das Pushoutobjekt des Pushouts von c_{2_L} und c_{1_R}, f der Pfeil von G nach M' und g der Pfeil von H nach M'. Dann ist die Konkatenation von c_1 und c_2 der Cospan

$$J \overset{f \circ c_{1_L}}{\rightarrow} M' \overset{g \circ c_{2_R}}{\leftarrow} K.$$

Mit dieser Definition ist auch $\mathbf{Cospan}(\mathbf{C})$*, die Kategorie mit den Objekten von* $\mathbf{C}$ *als Objekten, den Äquivalenzklassen der Cospans über* $\boldsymbol{C}$ *als Pfeilen und der oben definierten Konkatenation, tatsächlich eine Kategorie. Die Identitätspfeile dieser Kategorie sind die Cospans die aus Paaren von zwei Identitätspfeilen aus* $\mathbf{C}$ *gebildet werden. Wir nennen* $\mathbf{Cospan}(\mathbf{C})$ *die Cospan-Kategorie von* $\mathbf{C}$*.*

Bemerkung 2.9 *Für die Konkatenation verwenden wir je nach Kontext zwei verschiedene Schreibweisen. Statt $f \circ g$ schreiben wir mit gleicher Bedeutung auch manchmal $g\,;f$.*

Um in dem folgenden Unterkapitel die adhäsiven Kategorien zu definieren und wichtige Eigenschaften adhäsiver Kategorien zu verstehen, benötigen wir noch drei weitere Grundbegriffe der Kategorientheorie.

Definition 2.10 ***(Monomorphismus)***
Ein Monomorphismus ist ein Pfeil $f : X \to Y$, für den gilt: Für je zwei Pfeile $g : Z \to X$ und $h : Z \to X$ mit $f \circ g = f \circ h$ gilt $g = h$.

Definition 2.11 ***(Produkt)***
Das Produkt zweier Objekte A, B ist ein Objekt $A \times B$ zusammen mit zwei Pfeilen $\pi_1 : A \times B \to A$ und $\pi_2 : A \times B \to B$, so dass für jedes Objekt C und Paar von Pfeilen $f : C \to A$ und $g : C \to B$ genau ein vermittelnder Pfeil h existiert, so dass das folgende Diagramm kommutiert, so dass also gilt $\pi_1 \circ h = f$ und $\pi_2 \circ h = g$:

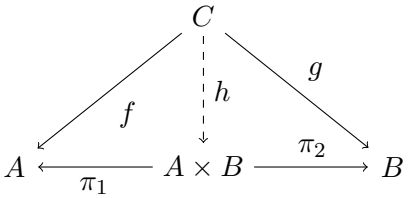


Abbildung 2.4: Produkt

Analog zu Pushout und Pullback gibt es auch einen dualen Begriff zum Produkt, das Coprodukt, der in der nachfolgenden Definition vorgestellt wird.

Definition 2.12 ***(Coprodukt)***
Das Coprodukt zweier Objekte A, B ist ein Objekt $A + B$ zusammen mit zwei Pfeilen $\pi_1 : A \to A + B$ und $\pi_2 : B \to A + B$, so dass für jedes Objekt C und Paar von Pfeilen $f : A \to C$ und $g : B \to C$ genau ein vermittelnder Pfeil h existiert, so dass das folgende Diagramm kommutiert, so dass also gilt $h \circ \pi_1 = f$ und $h \circ \pi_2 = g$:

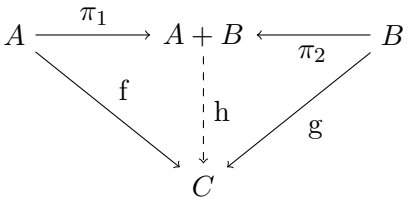


Abbildung 2.5: Coprodukt

Kapitel 3

Erkennbare Graphsprachen

Die Definitionen dieses Kapitels folgen, wenn nicht anders angegeben, [4]. Zunächst definieren wir die Kategorie **HGraph** von Hypergraphen und Graphmorphismen.

Definition 3.1 ***(Die Kategorie* HGraph*)***
Es sei Λ ein festes Alphabet von Labels. Ein (Hyper-) Graph (über Λ) ist ein Vier-Tupel $G = (V_G, E_G, att_G, lab_G)$. V_G ist die Menge der Knoten, E_G die Menge der Kanten, $att : E_G \to V_G^*$ die Verknüpfungsfunktion (wobei V_G^* die Menge der Sequenzen von Elementen aus V_G ist) und $lab_G : E_G \to \Lambda$ die Label-Funktion. Wir nennen die Knoten $att(k)$ die zu k inzidenten Knoten und nennen zwei Knoten adjazent in einem Graphen, wenn es eine Kante im Graphen gibt, die mit beiden Knoten inzidiert. Ein Graph G ist zusammenhängend, falls für jedes Paar von Knoten u, v des Graphen G eine Sequenz von Knoten $u = v_1, v_2, ..., v_n = v$ existiert, so dass für $i = 1, 2, ..., n-1$ gilt: v_i und v_{i+1} sind adjazent.

Ein Graphmorphismus f zwischen zwei Graphen G und H ist ein Paar von Funktionen (f_V, f_E) mit

$$f_V : V_G \to V_H$$

und

$$f_E : E_G \to E_H$$

so dass gilt:

$$lab_H \circ f_E = lab_G$$

und

$$att_H \circ f_E = f_V^* \circ att_G.$$

Dabei ist f_V^* die Erweiterung von f_V auf Sequenzen, die sich ergibt, wenn man f_V elementweise auf Sequenzen anwendet.
Den Graphen mit leerer Knoten- und Kantenmenge nennen wir $\emptyset$.
Die Kategorie **HGraph** *ist schließlich die Kategorie, die Hypergraphen als Objekte und Graphmorphismen als Pfeile hat.*

Wichtig ist vor allem die Kategorie **Cospan**(**HGraph**), also die Kategorie die Cospans von Hypergraphen als Pfeile hat. Mit dieser Vorbereitung können wir nun die Begriffe Automaten-Funktor, erkennbare Pfeilsprachen und schließlich erkennbare Graphsprachen definieren.

Definition 3.2 ***(Automaten-Funktor)***
Es sei $\mathbf{C}$ *eine Kategorie und* $\mathcal{A} : \mathbf{C} \to \mathbf{Rel}$ *ein Funktor, der jedes Objekt* X *von* $\mathbf{C}$ *auf eine endliche Menge* $\mathcal{A}(X)$ *abbildet und jeden Pfeil* $f : X \to Y$ *auf eine Relation* $R(f) \subseteq \mathcal{A}(X) \times \mathcal{A}(Y)$*. Wir nennen das Bild eines Objekts* X *von* $\mathbf{C}$ *unter* $\mathcal{A}$ *die Menge der Zustände von* X*. Für jede Menge* $\mathcal{A}(X)$ *gebe es eine ausgezeichnete Menge* $I_X^{\mathcal{A}}$ *von Startzuständen und eine Menge* $F_X^{\mathcal{A}}$ *der Endzustände. Dann nennen wir* $\mathcal{A}$ *einen Automaten-Funktor.*

Definition 3.3 ***(Erkennbarkeit)***
Es sei $\mathcal{A}$ *ein Automaten-Funktor in der Kategorie* $\mathbf{C}$ *und* J, K *zwei Objekte von* $\mathbf{C}$*. Die* (J, K)*-Sprache* $L_{J,K}(\mathcal{A})$ *ist definiert als die Menge, die alle solchen Pfeile* $f : J \to K$ *enthält, für die ein* $s \in I_J^{\mathcal{A}}$ *und ein* $t \in F_K^{\mathcal{A}}$ *existieren, die durch* $\mathcal{A}(f)$ *in Relation gesetzt werden. Eine Menge* $L_{J,K}$ *von Pfeilen von* J *nach* K *ist erkennbar in* $\mathbf{C}$*, wenn sie die* (J, K)*-Sprache eines Automaten-Funktors* $\mathcal{A} : \mathbf{C} \to \mathbf{Rel}$ *ist.*

Definition 3.4 ***(Erkennbare Graph-Sprachen)***
Eine Menge L *von Graphen mit innerem Interface* J *und äußerem Interface* K *ist erkennbar, wenn*

$$L_{I,J} = \{[G] : J \to G \leftarrow K | G \in L\}$$

eine erkennbare Sprache in der Kategorie **Cospan**(**HGraph**) *ist.*

Die letzten vier Definitionen sind aus [4] entnommen und folgen stellenweise wörtlich der Darstellung in [14].

Eine erkennbare Graphsprache ist also eine erkennbare Pfeilsprache in der Kategorie **Cospan**(**HGraph**).
Erkennbare Graphsprachen sind aber nicht die einzigen erkennbaren Sprachen. In einigen Fällen lassen sich Eigenschaften gleich für eine ganze Familie von erkennbaren Sprachen beweisen. In dieser Arbeit wird unter anderem gezeigt, dass die Konkatenation zweier erkennbarer Graphsprachen wieder eine erkennbare Graphsprache ergibt. Dieser Satz wird allerdings zunächst ohne direkten Bezug auf die Kategorie **HGraph** bewiesen. Stattdessen betrachten wir allgemeiner adhäsive Kategorien. Die Kategorie **HGraph** selbst ist eine adhäsive Kategorie, es gibt allerdings auch andere adhäsive Kategorien. Wir wollen im Folgenden nun den Begriff der adhäsiven Kategorie definieren.
Die folgende Definition ist nach [15].

Definition 3.5 ***(Van-Kampen-Viereck)***
Es sei ein Pushout P wie in der folgenden Abbildung gegeben:

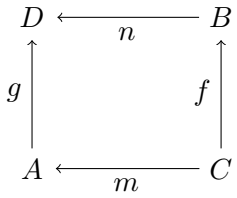


Abbildung 3.1: Van-Kampen-Viereck

Falls dieser Cospan für jeden kommutierenden Würfel wie in Abbildung 3.2 (in der Abbildung wird der Würfel von oben betrachtet) in dem der Cospan den Boden des Würfels bildet und in dem die die beiden hinteren Seiten des Würfels Pullbacks sind die folgende Bedingung erfüllt:

- *Die vorderen beiden Seiten sind Pullbacks, genau dann, wenn die obere Seite ein Pushout ist*

so nennen wir diesen Pushout P ein van-Kampen-Viereck.

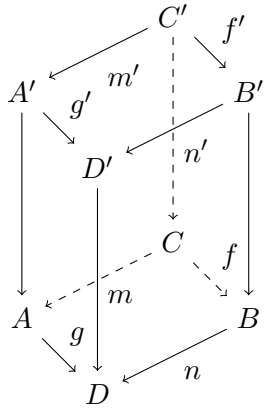


Abbildung 3.2: Würfel

Die folgende Definition ist [15] entnommen.

Definition 3.6 ***(Adhäsive Kategorien)***
Eine Kategorie **C** *nennen wir adhäsiv, wenn die folgenden drei Bedingungen erfüllt sind:*

- **C** *hat Pushouts entlang von Monomorphismen, das heißt, entlang Monomorphismen existiert immer ein Pushout*

- **C** *hat Pullbacks, das heißt, gegeben zwei Pfeile f und g wie in der Definition des Pullbacks, existiert immer ein Pullback.*
- *Pushouts entlang Monomorphismen sind van-Kampen-Vierecke.*

Adhäsive Kategorien haben einige angenehme Eigenschaften, die im Folgenden kurz dargestellt werden. Die Aussagen sind [15] entnommen und dort auch bewiesen.

Satz 3.7 *Gegeben sei eine Kategorie $\boldsymbol{K}$ und die Monomorphismen $A \to B$ und $B \to C$. Wenn es ein Pushout-Komplement zu $A \to B$ und $B \to C$ gibt, ist dieses bis auf Isomorphie eindeutig.*

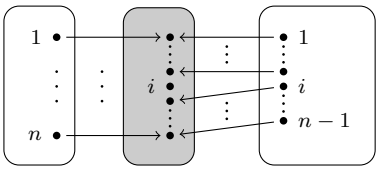
Satz 3.8 *Gegeben sei eine adhäsive Kategorie $\mathbf{C}$ und ein Objekt $Z \in \mathbf{C}$. Wir betrachten die Kategorie $\mathbf{Sub}(\mathbf{Z})$, die alle Monomorphismen nach Z in $\mathbf{C}$ als Pfeile hat und alle Objekte, für die es einen Monomorphismus nach Z gibt, als Objekte hat. Dann gibt es in $\mathbf{C}$ für jedes Paar von Objekten A, B ein Produkt-Objekt D, das wir als Pullback von A und B über Z gewinnen. Wir schreiben $D = A \cap B$. Weiterhin hat jedes Paar von Objekten A, B auch ein Coprodukt-Objekt E, wir schreiben $A \cup B = E$. In $\mathbf{C}$ gilt das Distributivgesetz für $\cup$ und $\cap$, es gilt also*

$$(A \cup B) \cap X = (A \cap X) \cup (B \cap X)$$

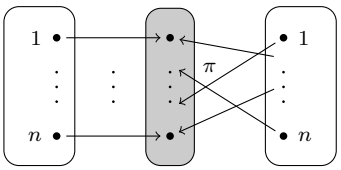
für alle A, B, X, die Objekte in $\mathbf{Sub}(\mathbf{Z})$ sind.

Definition 3.9 ***(Atomare Cospan-Menge)*** *[1]*
Die atomare Cospan-Menge enthält für alle Kantenlabel A und alle natürlichen Zahlen n die Alphabetssysmbole $connect^n_{A,\theta}$, $vertex^n_i$, $perm^n_\pi$ und res^n_i, die die folgenden Cospans umsetzen:

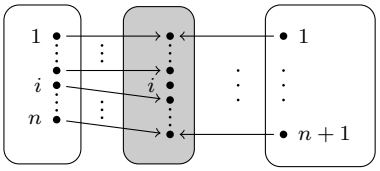
- *res^n_i: Der i-te Knoten wird verschattet. Der Cospan $I \to G \leftarrow J$ hat die folgende Form: $G = J$ und für $j = 1, ..., i-1$ wird der j-te Knoten von I auf den j-ten Knoten von G abgebildet, für $j = i+1, ..., n$ wird der j-te Knoten von I auf den $j+1$-ten Knoten von G abgebildet.*



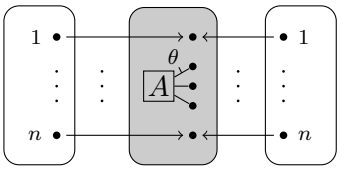
- *$perm^n_\pi$: π ist hierbei eine Permutation der Zahlen $1, ..., n$, diese Operation ändert die Reihenfolge der Knoten im Interface. Es gilt $I = G = J$ und für $i = 1, ..., n$ wird der i-te Knoten von I auf den i-ten Knoten von G abgebildet. Der i-te Knoten von J wird auf den $\pi(i)$-ten Knoten von G abgebildet.*



- *$vertex_i^n$: Fügt einen Knoten an i-ter Position zum Interface hinzu. Der Cospan $I \to G \leftarrow J$ hat die folgende Form: $G = J$, J enthält $n+1$ Knoten und I enthält n Knoten. Für $j = 1, ..., i-1$ wird der j-te Knoten aus I auf den j-ten Knoten aus G abgebildet, für $j = i, ..., n$ wird der j-te Knoten aus I auf den $j+1$-ten Knoten aus G abgebildet und der i-te Knoten hat kein Urbild in I. Für $j = 1, ..., n$ wird der j-te Knoten aus J auf den j-ten Knoten aus G abgebildet.*



- *$connect_{A,\theta}^n$: Fügt eine Kante mit dem Label A hinzu. Es ist $I = J$ und G enthält genau die Knoten aus I, aber zusätzlich eine Kante mit Label A, die die in θ angegebenen Knoten verbindet. θ ist eine Sequenz von Zahlen von 1 bis n. Für $i = 1, ..., n$ wird der i-te Knoten von I auf den i-ten Knoten aus G abgebildet und der i-te Knoten von J auf en i-ten Knoten aus G abgebildet.*



Wir nennen die Einschränkung der Kategorie **Cospan**(**HGraph**) *auf die Pfeile aus der Menge der atomaren Cospans* **ACospan**(**HGraph**).
ACospan(**HGraph**) *hat also als Objekte die Hypergraphen und als Pfeile die atomaren Cospans. Es ist allerdings zu beachten, dass es sich bei* **ACospan**(**HGraph**) *nicht um eine Kategorie handelt, da die Konkatenation zweier Pfeile in* **ACospan**(**HGraph**) *nicht wieder einen Pfeil in* **ACospan**(**HGraph**) *ergibt.*

Definition 3.10 ***(Zerlegung von Cospans)***
Gegeben sei ein Cospan c der Kategorie **Cospan**(**HGraph**). *Wir nennen eine Sequenz $c_1, c_2, ..., c_n$ von konkatenierbaren Cospans der Kategorie* **Cospan**(**HGraph**) *eine Zerlegung des Cospans c, falls der Cospan*

$d = c_1 ; c_2 ; ... ; c_n$ der durch Konkatenation der Cospans $c_1, c_2, ..., c_n$ entsteht gleich c ist, also $c = c_1 ; c_2 ; ... ; c_n$ gilt.

Definition 3.11 ***(Graph-Automat über der Menge der atomaren Cospans)***
Gegeben sei eine Abbildung $\mathcal{A} : \mathbf{ACospan}(\mathbf{HGraph}) \to \mathbf{Rel}$, die jedes Objekt X von $\mathbf{ACospan}(\mathbf{HGraph})$ auf eine endliche Menge $\mathcal{A}(X)$ abbildet und jeden Pfeil $f : X \to Y$ von $\mathbf{ACospan}(\mathbf{HGraph})$, auf eine Relation $R(f) \subseteq \mathcal{A}(X) \times \mathcal{A}(Y)$. Wir nennen das Bild eines Objektes X von $\mathbf{ACospan}(\mathbf{HGraph})$ unter $\mathcal{A}$ die Menge der Zustände von X. Für jede Menge $\mathcal{A}(X)$ gebe es eine ausgezeichnete Menge $I_X^{\mathcal{A}}$ von Startzuständen und eine Menge $F_X^{\mathcal{A}}$ von Endzuständen.
Die Abbildung $\mathcal{A}$ verallgemeinern wir auf Sequenzen von konkatenierbaren atomaren Cospans mit Hilfe der folgenden beiden Gleichungen:

$$\mathcal{A}(\epsilon) = id$$

$$\mathcal{A}(\sigma_1; \sigma_2) = \mathcal{A}(\sigma_1); \mathcal{A}(\sigma_2)$$

Falls für jeden Cospan der Kategorie $\mathbf{Cospan}(\mathbf{HGraph})$ und jedes Paar von Zerlegungen dieses Cospans in Sequenzen von atomaren Cospans σ_1 und σ_2 die Gleichung $\mathcal{A}(\sigma_1) = \mathcal{A}(\sigma_2)$ gilt, nennen wir $\mathcal{A}$ einen Graph-Automaten über der Menge der atomaren Cospans.
Ein Graph wird von einem Graph-Automaten über der Menge der atomaren Cospans akzeptiert, falls es für eine Zerlegung σ des Graphen in atomare Cospans einen Pfad im Automaten gibt, der einen Startzustand mit einem Endzustand verbindet. Die Menge aller von einem Graph-Automaten über der Menge der atomaren Cospans akzeptiertierten Graphen nennen wir die Sprache von $\mathcal{A}$ oder in Zeichen $L(\mathcal{A})$.

Satz 3.12 *Wenn L eine erkennbare Graph-Sprache ist, dann gibt es einen Graph-Automaten über der Menge der atomaren Cospans, der L akzeptiert. Umgekehrt, falls es einen Graph-Automaten über der Menge der atmoaren Cospans $\mathcal{A}$ gibt, so ist $L(\mathcal{A})$ eine erkennbare Graph-Sprache.*

Wir bemerken noch, dass die über den atomaren Cospan-Mengen definierten Automaten im Grunde keine Automatenfunktoren sind, da sie die Funktor-Eigenschaft in der angegebenen Form nicht erfüllen. Um einen Funktor zu erhalten, muss man allerdings nur jede Folge von Zustandsübergängen $z_1, z_2, ..., z_k$ mittels der Operatorfolge $(op_1, op_1, ..., op_{k-1})$ im Automaten durch einen Übergang von z_1 nach z_k mit dem Cospan, der sich als Pushout der Operatoren op_1 bis op_{k-1} ergibt, ergänzen.
Wir wollen uns die Definitionen dieses Kapitels nun an einem Beispiel verdeutlichen.

Beispiel 3.13 *In diesem Beispiel werden wir einen Graph-Automaten über der atomaren Cospan-Menge konstruieren, der die Sprache aller zusammenhängenden Graphen mit beliebigen Interface-Größen J und K akzeptiert. Wir werden uns in diesem Automaten merken, welche verschiedenen Komponenten es gibt und erlauben das Verschatten eines Knotens immer nur dann, wenn es noch einen anderen Knoten in dieser Komponente gibt, oder wenn es sich um den einzigen Knoten im Interface handelt und anschließend kein Knoten mehr eingelesen wird. Wir können dies fordern, denn wenn ein Knoten, der noch nicht zu einem anderen Knoten im Interface adjazent ist verschattet wird, so kann dieser nicht mehr in die gleiche Komponente gelangen wie die übrigen im Interface befindlichen Knoten, es kann also kein zusammenhängender Graph mehr entstehen. Gibt es nach Einlesen des gesamten Graphen nur noch eine Äquivalenzklasse im Interface, so müssen also alle Knoten in der gleichen Zusammenhangskomponente liegen.*
Zu jedem Interface I mit mindestens einem Knoten wählen wir als Zustände alle Äquivalenzrelationen auf der Knotenmenge in I. Falls das Startinterface J leer ist, gibt es weiterhin einen Zustand Z_s zum leeren Interface, falls das Zielinterface K leer ist, gibt es einen Zustand Z_e zum leeren Interface.
Als Startzustände wählen wir Z_s, falls J leer ist, sonst wählen wir den Zustand zum Interface J, bei dem die zugehörige Äquivalenzrelation $\equiv$ die Gleichheit ist. Als Endzustände wählen wir Z_e, falls K leer ist, ansonsten den Zustand zum Interface K, in dem alle Knoten zueinander äquivalent sind.
Wir erlauben nun einen Zustandsübergang von einem Zustand $\equiv_a$ zum Interface I_a in einen Zustand $\equiv_b$ zum Interface I_b mit der Operation op, falls:

- *$op = perm^n_\pi$: Falls $x \equiv_a y \Leftrightarrow x \equiv_b y$*
- *$op = vertex^n_i$: Falls der i-te Knoten im Interface I_b nur zu sich selbst äquivalent ist und für alle übrigen Knoten gilt $x \equiv_a y \Leftrightarrow x \equiv_b y$*
- *$op = connect^n_{A,s}$: Die Äquivalenzklassen modulo $\equiv_b$ außer denen, in denen Knoten aus s liegen, sind die gleichen wie die Äquivalenzklassen modulo $\equiv_a$, die Äquivalenzklassen der Knoten in s werden hingegen zu einer Äquivalenzklasse zusammengeführt.*
- *$op = res^n_i$: Falls die Äquivalenzklasse C des i-ten Knotens v in I nicht nur v enthält, ist diese Operation erlaubt, dann sind alle Äquivalenzklassen außer C von $\equiv_a$ auch Äquivalenzklassen in $\equiv_b$ und $C \setminus \{v\}$ ist die einzige weitere Äquivalenzklasse in $\equiv_b$. Falls K leer ist, erlauben wir den Übergang auch dann, wenn v der einzige Knoten im Interface ist, dann muss der Zielzustand Z_e sein.*

Wir zeigen nun zunächst, dass es sich hierbei um einen Graphautomaten über der atomaren Cospanmenge handelt. Zu zeigen ist, dass für zwei Zerlegungen des gleichen Cospans in Sequenzen von atomaren Cospans σ_1 und σ_2

die gleich Zustandsmenge erreichbar ist. Nun ist es aber so, dass durch die Zerlegung des Cospans in σ_1 der erreichte Zustand bereits eindeutig festgelegt ist, wir müssen nur noch zeigen, dass der erreichte Zustand mit σ_1 und σ_2 identisch ist. Angenommen, σ_1 gelange in den Zustand $\equiv_a$ und σ_2 in den Zustand $\equiv_b$. Wäre $\equiv_a$ ungleich $\equiv_b$, so gäbe es o.B.d.A. ein Paar von Knoten (u, v) im erzeugten Graphen G, der unter $\equiv_a$ äquivalent ist, aber nicht unter $\equiv_b$. Da $u \equiv_a v$ richtig ist, muss es in σ_1 eine connect$^n_{A,\theta}$-Operation geben mit $u \in \theta$ und $v \in \theta$. Dann muss es aber eine Kante geben, die u und v in G verbindet. Also muss diese Kante auch in σ_2 erzeugt werden. Wenn diese Kante erzeugt wird, müssen u und v allerdings in die gleiche Äquivalenzklasse gelangen. Da es keine Operation gibt, bei der u und v wieder in unterschiedliche Äquivalenzklassen gelangen könnten, muss auch $u \equiv_b v$ gelten. Das steht im Widerspruch zur Annahme, dass $a \not\equiv_b v$ richtig ist, also müssen $\equiv_a$ und $\equiv_b$ identisch sein.

Bleibt zu zeigen, dass der Automat auch genau die Sprache der zusammenhängenden Graphen akzeptiert. Ist ein Graph zusammenhängend, so gibt es zwischen zwei Knoten x und y immer einen (ungerichteten) Pfad im Graphen. Die Erzeugung der zugehörigen Kanten vereinigt so die Äquivalenzklassen von x und y. Also liegen alle Knoten in der gleichen Äquivalenzklasse, also wird jeder zusammenhängende Graph akzeptiert.

Andersherum, wird ein Graph akzeptiert, so muss jeder Knoten, der auf dem Weg erzeugt wird, in dieselbe Äquivalenzklasse fallen, denn jeder Knoten kann nur verschattet werden, wenn es noch einen Knoten der gleichen Äquivalenzklasse gibt und akzeptiert wird nur, wenn es am Ende nur noch eine Äquivalenzklasse gibt. Insgesamt ist also die Sprache der zusammenhängenden Graphen erkennbar.

Kapitel 4

Graphtransformationssysteme und kontextfreie Regeln

Die folgenden drei Definitionen sind [13] entnommen. Wir beginnen damit, Isomorphie zwischen zwei Graphen zu definieren. Hierbei handelt es sich um die Isomorphien der Kategorie **HGraph**, da wir die Isomorphie konkret für diese Kategorie allerdings benötigen werden, geben wir diese hier an.

Lemma 4.1 ***(Isomorphie)***
Wir nennen zwei Graphen G, H isomorph, falls es einen Morphismus $\varphi : G \to H$ gibt, dessen Komponenten φ_V und φ_E bijektiv sind. Wir schreiben auch $G \cong H$.

Definition 4.2 ***(Verkleben von Graphen)***
Es seien I, G_1, G_2 Graphen mit Graphmorphismen $\varphi_1 : I \to G_1$, $\varphi_2 : I \to G_2$. Wir nennen den Graphen $I = (V_I, E_I, att_I, lab_I)$ das Interface und nehmen an, dass die Knoten- und Kantenmengen aller drei Graphen disjunkt sind. Es sei $\equiv$ die kleinste Äquivalenzrelation auf $V_1 \cup E_1 \cup V_2 \cup E_2$, die $\varphi_1(x) \equiv \varphi_2(x)$ für alle $x \in V_I \cup E_I$ erfüllt. Dann bezeichnen wir als Verklebung der Graphen G_1 und G_2 über dem Interface I den Graphen $G = (V, E, att, lab)$ der die folgenden vier Bedingungen erfüllt:

- $V = (V_1 \cup V_2)/\equiv$
- $E = (E_1 \cup E_2)/\equiv$
- $att : E \to V^*$ *mit* $att([e]_\equiv) = [v_1]_\equiv ... [v_k]_\equiv$, *wobei*

$$v_1 ... v_k = \begin{cases} att_1(e) & , falls\ e \in E_1 \\ att_2(e) & , falls\ e \in E_2 \end{cases}$$

- $lab : E \to \Lambda$ *mit*

$$lab([e]_\equiv) = \begin{cases} lab_1(e) & , falls\ e \in E_1 \\ lab_2(e) & , falls\ e \in E_2 \end{cases}$$

Wir schreiben $G = G_1 +_{\varphi_1,\varphi_2} G_2$.

Die Verklebung zweier Graphen kann auch als Pushout von φ_1 und φ_2 verstanden werden, das Verklebe-Interface ist dann das gemeinsame Interface der beiden Cospans. Vergleiche auch folgende Abbildung.

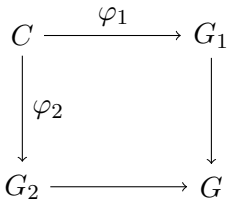


Abbildung 4.1: Verklebung

Definition 4.3 ***(Graphtransformationssysteme)***

- *Eine* Graphtransformationsregel *besteht aus drei Graphen* L, I, R *und injektiven Morphismen* φ_L, φ_R. *Wir schreiben eine solche Graphtransformationsregel als*
$$L \xleftarrow{\varphi_L} I \xrightarrow{\varphi_R} R.$$
- *Gegeben sei nun eine Graphtransformationsregel* $T = L \xleftarrow{\varphi_L} I \xrightarrow{\varphi_R} R$. *Wir sagen, ein Graph* G *wird zu einem Graphen* H *transformiert, wenn es einen Graphen* C, *den so genannten Kontext, und einen Graphmorphismus* $\psi : I \to C$ *gibt mit*
$$G \cong L +_{\varphi_L,\psi} C$$
$$H \cong R +_{\varphi_R,\psi} C.$$
Wir schreiben $G \Rightarrow_T H$.
- *Ein Graphtransformationssystem ist ein Tupel* (G_0, R) *wobei* G_0 *eine Menge von Graphen, die initialen Graphen oder Startgraphen, ist und* R *eine Menge von Graphtransformationsregeln.*

Die Anwendung einer Graphtransformationsregel entspricht einem doppelten Pushout, daher nennt man diesen Ansatz auch Double-Pushout-Ansatz. Vergleiche die folgende Abbildung:

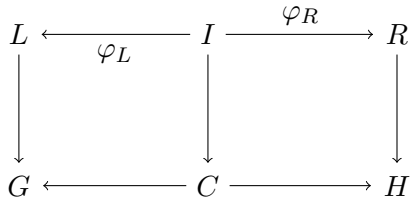


Abbildung 4.2: Double-Pushout

Im Kontext der Kategorientheorie wird die Anwendung einer Regel auf einen Graphen in der Regel durch eine Präfix-Ersetzung umgesetzt. Ist also eine Regel $L \to I \leftarrow R$ gegeben, so gehören zwei Cospans zu dieser Regel, die Cospans

$$\emptyset \leftarrow L \to I$$

und

$$\emptyset \to R \leftarrow I.$$

Für einen gegebenen Graphen G wird dann untersucht, ob sich der Cospan

$$\emptyset \to G \leftarrow \emptyset$$

schreiben lässt als

$$\emptyset \to L \leftarrow I\,; I \to C \leftarrow \emptyset.$$

Falls das der Fall ist, wird der Cospan ersetzt durch

$$\emptyset \to R \leftarrow I\,; I \to C \leftarrow \emptyset.$$

Hierbei nennen wir den Graphen C den Kontext, in dem die Regel angewandt wird.

Im Folgenden wollen wir von dieser Standard-Betrachtungsweise ein wenig abrücken. Im Wesentlichen benötigen wir für den Beweis der ersten Abschlusseigenschaft für Graphsprachen eine Betrachtungsweise, in der nicht zwingend ein Präfix ersetzt wird, sondern in der auch vor dem Vorkommen der linken Seite ein Kontext zu betrachten ist. Hierzu beweisen wir die folgenden zwei Sätze.

Satz 4.4 *Gegeben sei ein Cospan $\emptyset \to G \leftarrow I$. Es gilt*

$$\emptyset \to G \leftarrow I = \emptyset \to I \xleftarrow{id_I} I\,; I \to G \leftarrow I.$$

Beweis: Wir betrachten

$$\emptyset \to I \xleftarrow{id_I} I\,; I \to G \leftarrow I.$$

Die Konkatenation der beiden Cospans erhalten wir, indem wir den Pushout von G und I über dem Interface I bilden. Es ergibt sich wiederum G, also insgesamt der Cospan $\emptyset \to G \leftarrow I$ wie behauptet. □

Definition 4.5 *Es seien* $c : C_1 \xrightarrow{\alpha_1} C \xleftarrow{\alpha_2} C_2$ *und* $d : D \xrightarrow{id_D} D \xleftarrow{id_D} D$ *zwei Pfeile der Kategorie* **Cospan**(**HGraph**)*. Die Interfaces* C_1*,* C_2 *und* D *sind diskrete Graphen (nicht zwingend die gleichen), enthalten also nur Knoten und keine Kanten. Wir können die Knotenmengen eines Interfaces identifizieren mit der Menge* $\{0, 1, 2, ..., n\} = [n]$*. Da bei Interfaces die Reihenfolge der Knoten wichtig ist, verwenden wir diese Darstellung. Wir nennen im Folgenden also* C_1 $[n_1]$*,* C_2 $[n_2]$ *und* D $[m]$*. Dann ist* $c \oplus d$ *eine Menge von Cospans der Kategorie* **Cospan**(**HGraph**)*, die wie folgt definiert sei. Es gilt* $c \oplus d \ni e$ *mit* $e = E_1 \xrightarrow{\gamma_1} E \xleftarrow{\gamma_2} E_2$*, falls folgende Bedingungen erfüllt sind:*

- $E_1 = [n_1 + m]$
- $E_2 = [n_2 + m]$
- *Es gibt eine injektive Abbildung* $[n_1] \xrightarrow{f_1} [n_1 + m]$*, jeder Knoten in* C_1 *hat also einen anderen Bildknoten in* E_1*.*
- *Es gibt eine injektive Abbildung* $[m] \xrightarrow{g_1} [n_1 + m]$*, jeder Knoten in* D *hat also einen anderen Bildknoten in* E_1*.*
- f_1 *und* g_1 *sind gemeinsam surjektiv. Dadurch wird sichergestellt, dass jeder Knoten in* E_1 *auch einen zugehörigen Urbildknoten aus* C_1 *oder* D *hat.*
- $i < j \implies f_1(i) < f_1(j)$*, diese Bedingung stellt sicher, dass die Reihenfolge der Knoten aus* C_1 *in* C_1 *und* E_1 *gleich ist.*
- $i < j \implies g_1(i) < g_1(j)$*, diese Bedingung stellt sicher, dass die Reihenfolge der Knoten aus* D *in* D *und* E_1 *gleich ist.*
- *Es gibt einen injektiven Graphmorphismus* f *von* C *nach* E
- *Es gibt einen injektiven Graphmorphismus* g *von* D *nach* E
- f *und* g *sind gemeinsam surjektiv.*
- $codom(f) \cap codom(g) = \emptyset$ *Durch die letzten vier Bedingungen wird sichergestellt, dass* E *genau den Graphen repräsentiert, der durch die disjunkte Vereinigung von* C *und* D *entsteht.*
- *Es gibt eine injektive Abbildung* $[n_2] \xrightarrow{f_2} [n_2 + m]$*, jeder Knoten in* C_2 *hat also einen anderen Bildknoten in* E_2*.*
- *Es gibt eine injektive Abbildung* $[m_2] \xrightarrow{g_2} [n_2 + m]$*, jeder Knoten in* D *hat also einen anderen Bildknoten in* E_2*.*

- *f_2 und g_2 sind gemeinsam surjektiv. Dadurch wird sichergestellt, dass jeder Knoten in E_2 auch einen zugehörigen Urbildknoten aus C_2 oder D hat.*
- *$i < j \implies f_2(i) < f_2(j)$, diese Bedingung stellt sicher, dass die Reihenfolge der Knoten aus C_2 in C_2 und E_2 gleich ist.*
- *$i < j \implies g_2(i) < g_2(j)$, diese Bedingung stellt sicher, dass die Reihenfolge der Knoten aus D in D und E_2 gleich ist.*
- *Das folgende Diagramm kommutiert:*

$$\begin{array}{ccccc}
C_1 & \xrightarrow{\alpha_1} & C & \xleftarrow[\alpha_2]{} & C_2 \\
\downarrow{\scriptstyle f_1} & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f_2} \\
E_1 & \xrightarrow{\gamma_1} & E & \xleftarrow[\gamma_2]{} & E_2 \\
\uparrow{\scriptstyle g_1} & & \uparrow{\scriptstyle g} & & \uparrow{\scriptstyle g_2} \\
D & \xrightarrow{id_D} & D & \xleftarrow[id_D]{} & D
\end{array}$$

Wir können den Operator anschaulich so interpretieren, dass er auf alle Cospans abbildet, die die Knoten aus M irgendwie – aber in fester Reihenfolge – in den Cospan $I \to G \leftarrow I$ einordnet. Wenn wir im Folgenden sagen, ein Cospan sei $c : (M \to M \leftarrow M) \oplus (J \to G \leftarrow I)$, dann ist damit gemeint, dass es sich um irgendeinen Cospan aus dieser Menge handelt.

Lemma 4.6 *Gegeben seien drei Cospans der Kategorie* **HGraph**

$$c_1 : \emptyset \to G_1 \xleftarrow{m} M$$

$$c_2 : M \to G_2 + M \xleftarrow{\varphi_1} I + M \in (M \xrightarrow{id_M} M \xleftarrow{id_M} M) \oplus (\emptyset \to G_2 \leftarrow I)$$

und

$$d_1 : \emptyset \to G_2 \leftarrow I$$

wobei

$$G_1 \cap G_2 = \emptyset$$

gilt und φ_1 id_M enthält. Dann gilt für jeden Cospan der Form

$$d_2 : I \to G_1 + I \xleftarrow{\varphi_2} I + M \in (I \xrightarrow{id_I} I \xleftarrow{id_I} I) \oplus (\emptyset \to G_1 \xleftarrow{m} M)$$

mit

$$M + I \xrightarrow{\varphi_1} (M + G_2)|_{Im(\varphi_1)} = M + I \xrightarrow{\varphi_2} (I + G_1)|_{Im(\varphi_2)}$$

die Gleichung $c_1 ; c_2 = d_1 ; d_2$.

Beweis: Wir konkatenieren die Cospans c_1 und c_2 mittels Pushout. Der Pushout von G_1 und $G_2 \cup M$ über M ist $G_1 \cup G_2 \cup M$ und da $M \subseteq G_1$ richtig ist, ergibt sich der Cospan

$$c_1 ; c_2 = \emptyset \to (G_1 \cup G_2) \leftarrow (I \cup M).$$

Weiterhin konkatenieren wir die Cospans d_1 und d_2 mittels Pushout. Der Pushout von G_2 und $G_1 \cup I$ ist $G_2 \cup G_1 \cup I$ und wegen $I \subseteq G_2$ ergibt sich der Cospan

$$d_1 ; d_2 = \emptyset \to (G_1 \cup G_2) \leftarrow (I \cup M).$$

Das gilt für alle möglichen Wahlen von d_2. Da das linke Interface das leere Interface ist, der Graph der gleiche ist und die Gleichheit des rechten Interfaces durch die Bedingung an φ_1 und φ_2 sichergestellt ist, handelt es sich um die gleichen Cospans. □

Lemma 4.7 *Es seien drei Cospans der Kategorie* **HGraph** *gegeben, bei denen alle Pfeile injektiv sind:*

$$c_1 : J \to C_1 \leftarrow K$$

$$c_2 : J \to C_2 \leftarrow K$$

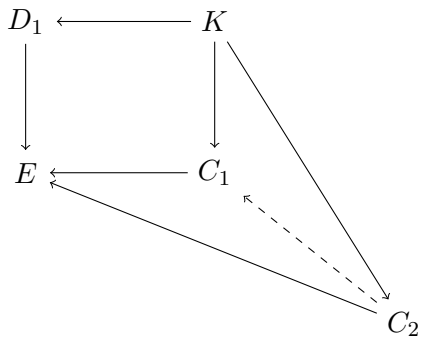
$$d : K \to D_1 \leftarrow M.$$

Falls $c_1 ; d = c_2 ; d$ richtig ist, folgt $c_1 = c_2$.

Beweis: Es sei E das Pushoutobjekt von D_1 und C_1 über K. Da $c_1 ; d = c_2 ; d$ ist, ist E auch das Pushoutobjekt von D_1 und C_2 über K. Wegen $c_1 ; d = c_2 ; d$ muss insbesondere auch gelten

$$J \to C_2 \to E = J \to C_1 \to E.$$

Da bei injektiven Morphismen das Pushout-Komplement, wenn es existiert, eindeutig ist, gibt es einen Isomorphismus zwischen C_1 und C_2 und das folgende Diagramm kommutiert:



Damit können wir dann berechnen

$$J \to C_2 \to C_1 \to E = J \to C_2 \to E = J \to C_1 \to E_1.$$

Damit gilt also die Behauptung. □

Satz 4.8 *Es sei c ein Cospan und l ein Cospan, so dass wir l und c zu $l\,;c$ konkatenieren können. Weiterhin sei M ein diskreter Graph, sowie c_1 und c_2 Cospans, so dass $l\,;c \in c_1\,;(l \oplus id_M)\,;c_2$ gilt. Ist nun $L \leftarrow I \to R$ eine Graphersetzungsregel und $l = (\emptyset \to L \leftarrow I)$ sowie $r = (\emptyset \to R \leftarrow I)$, dann können wir schreiben*

$$l\,;c = c_1\,;(M \to L + M \xleftarrow{\varphi_l} I + M)\,;c_2.$$

Dann gilt

$$r\,;c = c_1\,;(M \to R + M \xleftarrow{\varphi_r} I + M)\,;c_2$$

für alle $(M \to R + M \xleftarrow{\varphi_r} I + M) \in r \oplus id_M$ mit

$$M \to L + M|_{Im(\varphi_l)} \leftarrow I + M = M \to R + M|_{Im(\varphi_r)} \leftarrow I + M$$

Beweis: Da $c_1\,;(l \oplus id_M)\,;c_2 \ni l\,;c$ richtig ist, lässt sich c schreiben als $c_1'\,;c_2$, wir müssen in dieser Schreibweise also zeigen dass

$$c_1\,;(M \to R + M \xleftarrow{\varphi_r} I + M)\,;c_2 = r\,;c_1'\,;c_2$$

richtig ist. Da c_2 ein gemeinsames Suffix der beiden Cospans ist, bleibt aber nur noch zu zeigen

$$c_1\,;(M \to R + M \xleftarrow{\varphi_r} I + M) = (\emptyset \to R \leftarrow I)\,;c_1'.$$

Dabei beachten wir, dass wir bereits wissen, dass gilt:

$$c_1\,;(M \to L + M \xleftarrow{\varphi_l} I + M) = (\emptyset \to L \leftarrow I)\,;c_1'.$$

Auf Grund dieser Feststellung wissen wir bereits, dass in beiden Fällen das linke Interface $\emptyset$ und das rechte Interface $M \cup I$ sein muss. Wir betrachten nun also noch einmal die Gleichung $c_1\,;(M \to L + M \xleftarrow{\varphi_l} I + M) = l\,;c_1'$. Wir stellen fest, dass links die I-Knoten erst im Teil des Typs $l \oplus id_M$ erzeugt werden, insbesondere also nicht adjazent zu irgendeinem Knoten in c_1 sein können. Dann kann aber auch c_1' nur Knoten und Kanten erzeugen, die nicht zu einem der Knoten aus I adjazent sind. Dann ist also c_1' vom Typ $\widehat{c}_1 \oplus id_I$. Mit unserem Lemma 4.6 sehen wir außerdem ein, dass

$$c_1\,;(M \to L + M \xleftarrow{\varphi_r} I + M)$$

mit

$$c_1 = \emptyset \to C_1 \leftarrow M$$

sich schreiben lässt als

$$l\,;(\emptyset + I \to C_1 + I \leftarrow M + I)$$

also auch, mit Hilfe von Lemma 4.7 $\widehat{c}_1 = c_1$ gilt. Damit muss weiterhin

$$r\,;(\emptyset + I \to C_1 + I \leftarrow M + I) = r\,;c_1'$$

richtig sein.

Wir wissen nun also, dass gilt

$$r\,;c_1' = (\emptyset \to R \leftarrow I)\,;(I \to C_1 + I \leftarrow M + I)$$

und

$$c_1\,;(M \to R + M \xleftarrow{\varphi_2} M + I) = (\emptyset \to C_1 \leftarrow M)\,;(M \to R + M \xleftarrow{\varphi_2} M + I).$$

Dass beide Ausdrücke gleich sind, sehen wir nun wieder mittels Lemma 4.6 ein. □

Folgerung 4.9 *Der Graphersetzungsformalismus, der sich ergibt, wenn man nach Zerlegungen des Cospans des Graphen sucht, die in der Form*

$$\emptyset \to G_1 \leftarrow I + M\,;I + M \xrightarrow{\alpha} L + M \xleftarrow{\alpha} I + M\,;I + M \to G_2 \leftarrow \emptyset$$

sind und den Teilcospan $I \xrightarrow{l} L \xleftarrow{l} I$ durch $I \xrightarrow{r} R \xleftarrow{r} I$ ersetzt, ist äquivalent zur Präfix-Ersetzung.

Beweis: Wir wenden zunächst den Satz 4.8 an und können schreiben

$$\emptyset \to R \leftarrow I\,;I \to C \leftarrow \emptyset =$$
$$\emptyset \to C_1 \leftarrow M\,;M \to R \leftarrow M + I\,;M + I \to M + C_2 \leftarrow \emptyset$$

und anschließend den Satz 4.4 und erhalten

$$\begin{aligned}\emptyset \to C_1 \leftarrow M\,;M \to M + R \xleftarrow{\beta} M + I\,;M + I \to M + C_2 \leftarrow \emptyset \\ = \emptyset \to C_1 \leftarrow M\,;M \to M + I \xleftarrow{id_{M+I}} M + I\,; \\ M + I \xrightarrow{\beta} M + R \xleftarrow{\beta} M + I\,;M + I \to M + C_2 \leftarrow \emptyset \\ = \emptyset \to C_1 + I \leftarrow M + I\,;M + I \xrightarrow{\beta} M + R \xleftarrow{\beta} M + I\,; \\ M + I \to M + C_2 \leftarrow \emptyset\end{aligned}$$

Für den Fall, dass in dem Präfix-Cospan die Knoten aus I nur erzeugt werden und anderweitig nicht verwendet werden, haben wir also bereits Gleichheit.

Wir können aber schließlich auch beliebige Teil-Cospans des Suffix-Cospans, die das Interface $I + M$ erhalten, in den Präfix-Cospan ziehen, denn es gilt:

$$\emptyset \to A \leftarrow B \,;B \xrightarrow{\beta_1} C \xleftarrow{\beta_1} B \,;B \xrightarrow{\beta_2} D \xleftarrow{\beta_2} B \,;B \to B \leftarrow E$$
$$= \emptyset \to A \leftarrow B \,;B \xrightarrow{\beta_2} D \xleftarrow{\beta_2} B \,;B \xrightarrow{\beta_1} C \xleftarrow{\beta_1} B \,;B \to B \leftarrow E$$

für beliebige Cospans von Hypergraphen $B \xrightarrow{\beta_2} D \xleftarrow{\beta_2} B$ und $B \xrightarrow{\beta_1} C \xleftarrow{\beta_1} B$. Das sieht man ein durch Abgleich der sich ergebenden Cospans, wenn man die Pushouts bildet.
Weiterhin sehen wir ein, dass Präfix-Ersetzung ein Spezialfall unserer Interpretation der Graphersetzung ist (der erste Cospan ist einfach der leere Cospan $\emptyset \to \emptyset \leftarrow \emptyset$).
Andersherum: Wenn es eine Zerlegung eines Cospans c gibt mit

$$c = \emptyset \to A \leftarrow B + I \,;B + I \to B + L \leftarrow B + I \,;B + I \to C \leftarrow D,$$

dann gibt es eine Zerlegung des Cospans $\emptyset \to A \leftarrow B + I$ zu

$$\emptyset \to A' \leftarrow B \,;B \to A'' \leftarrow B + I$$

und dieser kann wieder weiter zerlegt werden zu

$$\emptyset \to A' \leftarrow B \,;B \to B + I \leftarrow B + I \,;B + I \to A'' \leftarrow B + I.$$

Anschließend kann der letzte dieser Cospans in das Suffix gezogen werden, so dass wir die Gesamtsituation erhalten:

$$\emptyset \to \widehat{A} \leftarrow B \,;B \to B + L \leftarrow B + I \,;B + I \to C' \leftarrow D$$

Da nun im mittleren Cospan die B-Knoten isolierte Knoten sind, können wir wieder nach Lemma 4.6 den mittleren Cospan nach vorne ziehen und erhalten die Zerlegung

$$\emptyset \to L \leftarrow I \,;I \to \widehat{A} + I \leftarrow B + I \,;B + I \to C' \leftarrow D.$$

Also ist auch immer dann, wenn in der neuen Interpretation der Graphersetzung eine Regelanwendung möglich ist, eine Regelanwendung in der Präfix-Ersetzung möglich. □

In der Theorie formaler Sprachen bezeichnet man Regeln dann als kontextfrei, wenn ein einzelnes Symbol auf der linken Seite der Regel steht und die rechte Seite der Regel mindestens ein Symbol enthält. Analog wollen wir nun den Begriff der kontextfreien Graphersetzungsregel definieren.

Definition 4.10 ***(Kontextfreie Graphersetzungsregeln)****[9]*
Eine Graphersetzungsregel $L \leftarrow I \to R$ nennen wir kontextfrei, wenn sie die folgenden Bedingungen erfüllt:

- *L ist ein zusammenhängender Hypergraph mit nur einer Kante.*
- *I ist ein Graph der genau so viele Knoten wie L enthält, aber keine Kante hat.*

Teil II

Abschlusseigenschaften von Graphsprachen

In diesem Teil der Arbeit untersuchen wir erkennbare Graphsprachen auf ihre Abschlusseigenschaften. Für reguläre Sprachen gelten eine ganze Reihe von Abschlusseigenschaften, manche hiervon – Abschluss unter Schnitt, Vereinigung und Komplement – lassen sich recht problemlos auf erkennbare Graphsprachen übertragen, andere Eigenschaften bedürfen einer genaueren Betrachtung. Wir werden zunächst sehen, dass die erkennbaren Graphsprachen abgeschlossen sind unter der Anwendung invers kontextfreier Graphersetzungsregeln.

Anschließend betrachten wir die Frage der Abgeschlossenheit unter Konkatenation ein wenig allgemeiner. Wir werden zeigen, dass die erkennbaren Pfeilsprachen auf adhäsiven Kategorien unter Konkatenation abgeschlossen sind. Eine konkrete Konstruktion für den Fall der erkennbaren Graphsprachen werden wir dann im darauf folgenden Kapitel kennen lernen.

Leider lassen sich aber nicht alle Abschlusseigenschaften der regulären Sprachen auf die erkennbaren Graphsprachen übertragen. Wir werden sehen, dass die erkennbaren Graphsprachen nicht abgeschlossen sind unter der Anwendung löschender Graphersetzungsregeln, also solchen Regeln, die einfach das Entfernen von Kanten bestimmter Labels ermöglichen. Im letzten Kapitel werden wir durch Angabe eines Gegenbeispiels zeigen, dass erkennbare Graphsprachen nicht abgeschlossen sind unter der Anwendung von Substitutionen und auch nicht unter Anwendung des Kleene-Sterns. Diese drei Abschlusseigenschaften werden von regulären Sprachen erfüllt.

Kapitel 5

Abschluss unter invers kontextfreien Regeln

Ziel dieses Abschnitts ist es, zu zeigen, dass erkennbare Graphsprachen unter invers kontextfreien Regeln abgeschlossen sind. Dafür müssen wir natürlich zunächst einmal definieren, was wir unter invers kontextfreien Regeln verstehen.

Definition 5.1 ***(Invers kontextfreie Graphersetzungsregel)*** *Eine Regel $L \leftarrow I \rightarrow R$ ist eine invers kontextfreie Graphersetzungsregel, wenn $R \leftarrow I \rightarrow L$ eine kontextfreie Graphersetzungsregel ist.*

Wir werden im Folgenden hin und wieder von einem Cospan sprechen, der L repräsentiert und einem Cospan, der R repräsentiert. Hierzu wählen wir die Cospans $I \rightarrow L \leftarrow I$ und $I \rightarrow R \leftarrow I$.

Satz 5.2 *Es sei $\mathcal{A}$ ein Graph-Automat, der die Graphsprache S akzeptiert. Weiterhin sei $T = \{T_i = (L_i \leftarrow I_i \rightarrow R_i) : 1 \leq i \leq t\}$ eine Menge invers kontextfreier Regeln, das heißt, dass jedes R_i genau eine Kante k_i mit Label lab_i enthält. Dann gilt, dass die Sprache*

$$S' = \{H : \exists G : G \in L \wedge G \Rightarrow_T^* H\}$$

ebenfalls eine erkennbare Graphsprache ist. S' entsteht also aus S, indem man S mit der Menge aller Graphen vereinigt, die durch beliebig häufige Anwendung von Regeln aus T auf einen Graphen aus S entstehen.

Beweis: Wir können annehmen, dass $\mathcal{A}$ ein Automat ist, der nur Übergänge der atomaren Cospan-Menge verwendet. Dann konstruieren wir einen Automaten $\mathcal{A}'$, der genau die Sprache S' akzeptiert und zeigen, dass dieser tatsächlich die korrekte Sprache akzeptiert und wiederum ein Graphautomat über der atomaren Cospan-Menge ist.
Wir definieren $\mathcal{A}'$ wie folgt: $\mathcal{A}'$ enthalte genau die Zustände von $\mathcal{A}$ und alle

Übergänge von $\mathcal{A}$. Weiterhin fügen wir allerdings neue Übergänge zu $\mathcal{A}'$ hinzu. Wir betrachten für alle Knotenmengen M alle Pfade im Automaten $\mathcal{A}'$, deren Operatorfolgen σ einen Cospan

$$I_i + M \xrightarrow{\alpha} L_i + M \xleftarrow{\alpha} I_i \in I_i \to L_i \leftarrow I_i \oplus id_M,$$

im Weiteren kurz $L_i \overline{\oplus} id_M$ genannt, erzeugen. Dabei können wir uns dank der Funktor-Eigenschaft des Automaten $\mathcal{A}$ für jede linke Seite L_i auf eine fixe Zerlegung in atomare Operatoren beschränken. Wir können diesen Cospan schreiben als $A \to B \leftarrow A$, wir nennen die Zustände, die durch den Pfad σ miteinander verbunden werden z und z', es gilt also

$$z \to_\sigma z'.$$

Für jede solche Sequenz σ fügen wir einen neuen Übergang

$$z \xrightarrow{connect^{|A|}_{lab_i, f_{I_i}}} z'$$

hinzu. Dabei gibt f_{I_i} die Position der I_i-Knoten im A-Interface an.

Wir müssen nun drei Dinge zeigen:

- $\mathcal{A}'$ ist ein Graphautomat über der atomaren Cospan-Menge
- $\mathcal{A}'$ akzeptiert alle Graphen in S'
- Alle von $\mathcal{A}'$ akzeptierten Graphen sind in S'.

Wenn im Folgenden von der Zahl benötigter neuer Übergänge gesprochen wird, so ist dies im Fall dessen, dass die Kantenlabel der Kanten auf der rechten Seite der T_i nicht auf den linken Seite der T_i vorkommt, einfach die Zahl der neuen Übergänge in der Zustandsfolge. Wenn hingegen auf der linken Seite einer T-Regel das Kantenlabel der Kante auf einer rechten Seite einer T-Regel vorkommt, so ist dies nicht unbedingt identisch. Dann ist es möglich, dass ein neuer Übergang eine Operatorfolge ersetzt, die bereits einen oder mehrere neue Übergänge beinhaltet.
Formal schreiben wir für die Zahl benötigter Übergänge $b(\sigma)$ eines Pfades σ durch den Automaten $\mathcal{A}'$ σ:

$$b(\sigma) = \begin{cases} 0 & falls \sigma auch Pfad in \mathcal{A} ist. \\ \sum_{op \in \sigma} b(op) & , sonst \end{cases}$$

und für die Zahl benötigter Übergänge $b(op)$ eines neu hinzugefügten Übergangs op zur Abkürzung des Pfades σ

$$b(op) = b(\sigma) + 1.$$

Rekursiv ist die Zahl der benötigten neuen Übergänge eines neuen Übergangs also 1, falls keine neuen Übergänge hierdurch abgekürzt werden, sonst ist es die Summe der benötigten neuen Übergänge aller durch die Operatorfolge verwendeten neuen Übergänge plus 1.

1. Um zu zeigen, dass $\mathcal{A}'$ ein Graphautomat über der atomaren Cospan-Menge ist, betrachten wir zwei beliebige Sequenzen σ_1 und σ_2, die den gleichen Cospan erzeugen und zeigen, dass jeder Zustand, der von Sequenz σ_1 erreicht werden kann, auch von Sequenz σ_2 erreicht werden kann. Da die Wahl von σ_1 und σ_2 beliebig ist, folgt dann, dass die Menge der erreichbaren Zustände unabhängig von der gewählten Zerlegung des Cospans ist, was die erste Eigenschaft beweist.
   Wir betrachten hierzu eine fixe Zustandsfolge Z_1 zu σ_1 und zeigen, dass der finale Zustand z_f von Z_1 mit der Sequenz σ_2 erreichbar ist. Durch Z_1 und σ_1 ist ein Pfad durch den Automaten festgelegt, die durch diesen Pfad benötigten neuen Übergänge nennen wir im Folgenden die von Z_1 benötigten neuen Übergänge.
   Wir zeigen dies per Induktion über die Zahl n der benötigten neuen Zuständsübergänge in Z_1. Wenn im Folgenden vom $n+1$-ten Übergang die Rede ist, ist die Wahl des benötigten Übergangs als $n+1$-ten Übergang beliebig, es muss allerdings ein Übergang gewählt werden, der tatsächlich auf dem Pfad liegt.
   Induktionsanfang ($n = 0$): Wenn für Z_1 keine neuen Übergänge verwendet werden müssen, so handelt es sich um eine Zustandssequenz in $\mathcal{A}$. Dann ist z_f auch von σ_2 in $\mathcal{A}$ erreichbar, insbesondere also auch in $\mathcal{A}'$ erreichbar, da alle Übergänge aus $\mathcal{A}$ auch Übergänge in $\mathcal{A}'$ sind.
   Induktionsvoraussetzung: Für alle $m \leq n$ gilt: Falls Z_1 genau m neue Übergänge benötigt, ist z_f durch σ_2 erreichbar.
   Induktionsschritt ($n \to n+1$): Es sei

$$z \xrightarrow{connect^{y}_{lab_i, f_{I_i}}} z'$$

   der $n+1$-te benötigte neue Übergang in Z_1. Dann lässt sich σ_1 schreiben als

$$\sigma_1 = (\sigma_1^a \,;\, connect^{y}_{lab_i, f_{I_i}} \,;\, \sigma_1^b).$$

   Da σ_1 und σ_2 die gleichen Cospans erzeugen, gibt es zudem eine *connect*-Operation op, die gemäß der Übereinstimmung der beiden Cospans die gleiche Kante erzeugt und σ_2 lässt sich schreiben als

$$\sigma_2 = (\sigma_2^a \,;\, op \,;\, \sigma_2^b).$$

   Wir bilden nun aus σ_1 eine neue Sequenz

$$\sigma_1' = (\sigma_1^a \,;\, \sigma \,;\, \sigma_1^b),$$

in der wir den $connect^{y}_{lab_i,f_I}$-Übergang ersetzen durch die zugehörige alte Sequenz σ, die durch den neuen $connect^{y}_{lab_i,f_{I_i}}$-Übergang abgekürzt wurde. Dann ist insbesondere z_f durch σ'_1 erreichbar.
Analog bilden wir die Sequenz σ'_2 gemäß

$$\sigma'_2 = (\sigma_2^a \,;\, \sigma' \,;\, \sigma_2^b)$$

in der wir *op* durch eine Sequenz σ' ersetzen, die wir wie folgt bilden. σ lässt sich schreiben als $L_i \overline{\oplus} id_M$ für eine Knotenmenge M. Weiterhin ist $op = connect^{x}_{lab_i,\theta}$. Nun sei N eine Knotenmenge mit $x - |\sigma|$ Knoten, dann wählen wir schließlich $\sigma' = L_i \overline{\oplus} id_N$. Dann bilden σ'_1 und σ'_2 den gleichen Cospan. Da σ'_1 allerdings nur n neue Übergänge verwendet, sind alle Zustände, die durch σ'_1 erreichbar sind, auch durch σ'_2 erreichbar. Insbesondere ist somit z_f durch σ'_2 erreichbar. Wir betrachten nun eine Zustandsfolge in $\mathcal{A}'$ für σ'_2, mit der z_f erreicht wird und betrachten die Zustände $\widehat{x}$ und $\widehat{x'}$, die in der Sequenz durch σ' verbunden werden. Nach Konstruktion von $\mathcal{A}'$ kann man auch mit *op* von $\widehat{x}$ nach $\widehat{x'}$ gelangen, da das Präfix und Suffix bei σ_2 und σ'_2 identisch sind, ist also schließlich z_f auch erreichbar mit σ_2.

2. Um zu zeigen, dass $\mathcal{A}'$ alle Graphen in S' akzeptiert, betrachten wir einen beliebigen Graphen G in S'. Wenn $G \in S'$ richtig ist, dann entsteht G aus einem Graphen $\widehat{G} \in S$ durch n-fache Anwendung von Regeln aus T. Wir führen eine Induktion durch nach der Zahl n.
Induktionsanfang ($n = 0$): In diesem Fall ist $G \in S$ und wird somit von $\mathcal{A}'$ akzeptiert.
Induktionsvoraussetzung: Für alle $m \leq n$ gilt: Entsteht G durch genau m Regelanwendungen von Regeln aus T auf einen Graphen $\widehat{G} \in S$, so wird der Graph durch $\mathcal{A}'$ akzeptiert.
Induktionsschritt ($n \to n+1$):

   Wir betrachten eine Operatorsequenz σ, die G bildet. Die $n+1$-te Regelanwendungen einer Regel T_i aus T auf einen Graphen G', der durch n-fache Regelanwendung von Regeln aus T auf $\widehat{G}$ entsteht, korrespondieren zu einer *connect*-Operationen *op*, so dass

$$\sigma = (\sigma_1 \,;\, op \,;\, \sigma_2)$$

   richtig ist. Wir ersetzen nun in der obigen Operatorfolge σ das *op* durch eine passende Sequenz σ'_1, so dass die sich ergebende Operatorfolge

$$\sigma' = (\sigma_1 \,;\, \sigma'_1 \,;\, \sigma_2)$$

   den Graphen G' erzeugt. Da G' nach Induktionsvoraussetzung von $\mathcal{A}'$ akzeptiert wird, muss nach Definition des Automaten $\mathcal{A}'$ auch σ durch $\mathcal{A}'$ akzeptiert werden, G wird also von $\mathcal{A}'$ akzeptiert.

3. Wir betrachten nun eine beliebige Operatorsequenz σ, die von $\mathcal{A}'$ akzeptiert wird. Um zu zeigen, dass diese Operatorsequenz auch einen Graphen in S' erzeugt, führen wir eine Induktion über die Zahl der benötigten neuen Regelanwendungen für eine zu σ gehörige akzeptierende Zustandsfolge Z durch.
Induktionsanfang ($n = 0$): Wenn keine neue Regel in der Zustandsfolge angewendet werden muss, dann wird die Operatorsequenz auch von $\mathcal{A}$ akzeptiert, der durch σ erzeugte Graph ist also in $S \subseteq S'$.
Induktionsvoraussetzung: Für alle $m < n$ gilt: Wenn genau m neue Regelanwendungen für die Zustandsfolge notwendig sind, dann erzeugt die Operatorsequenz einen Graphen der in S' liegt.
Induktionsschritt ($n \to n + 1$): Wir betrachten die $n + 1$-te benötigte T-Regelanwendung und bemerken, dass diese zu einer *connect*-Operation *op* gehört, so dass sich σ schreiben lässt als

$$\sigma = (\sigma_1 \,;\, op \,;\, \sigma_2).$$

Weiterhin sei G' ein Graph, der durch Anwendung der $n+1$-ten benötigten T-Regelanwendung zu dem Graphen G, der durch σ gebildet wird, transformiert wird. Ersetzen wir die *op*-Operation durch eine Operatorfolge σ'_1, die das Vorkommen der linken Seite in G', das zur $n + 1$-ten T-Regelanwendung gehört, bildet, so erhalten wir eine Operatorfolge

$$\sigma' = (\sigma_1 \,;\, \sigma'_1 \,;\, \sigma_2),$$

die genau den Graphen G' bildet und für die nur n T-Regelanwendungen notwendig sind. Per Definition des Automaten $\mathcal{A}'$ wird der Endzustand, der von σ erreicht wird, auch von σ' erreicht. Da σ' allerdings nur n T-Regelanwendungen benötigt, folgt aus der Induktionsannahme, dass $G' \in S'$ ist und da G aus G' durch T-Regelanwendung entsteht, ist auch $G \in S'$.

□

Beispiel 5.3 *Wir wollen nun ein Beispiel für unsere oben angegebene Konstruktion vorführen. Wir beginnen mit einem Graph-Automaten für die Sprache L aller Graphen mit einem Knoten und gerade vielen einstelligen A-Kanten. Der Automat habe die vier Zustände z_0, z_g, z_u, z_e und – abgesehen von einstelligen Permutationen – die Übergänge*

- $z_0 \xrightarrow{vertex_1^0} z_g$
- $z_g \xrightarrow{connect^1_{[1],A}} z_u$
- $z_u \xrightarrow{connect^1_{[1],A}} z_g$

- $z_g \xrightarrow{res^1_{\{1\}}} z_e$

Weiterhin verwenden wir die Menge $\{z_0\}$ als Startzustandsmenge und die Menge $\{z_e\}$ als Endzustandsmenge.
Wir können diesen Automaten grafisch analog zu nichtdeterministischen endlichen Automaten für Wortsprachen darstellen.

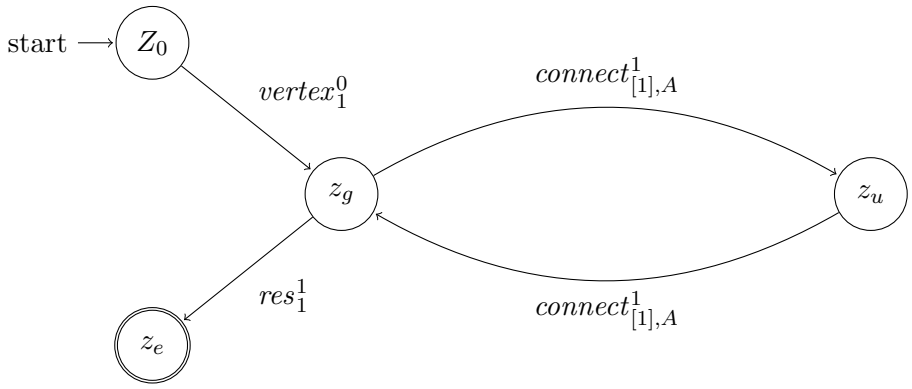
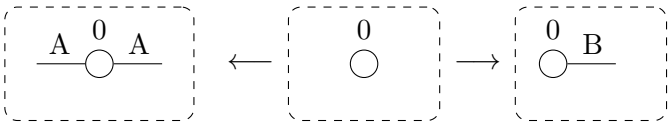


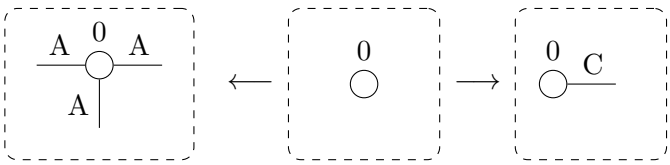
Abbildung 5.1: Eingabe-Automat

Als Regelmenge verwenden wir die Menge, die die folgenden zwei Regeln enthält:

- *Zwei einstellige A-Kanten die mit dem gleichen Knoten inzidieren, können durch eine einstellige B-Kante ersetzt werden.*



- *Drei einstellige A-Kanten die dem gleichen Knoten inzidieren können durch eine einstellige C-Kante ersetzt werden.*



Gemäß unserer Konstruktion müssen wir nun nach Operatorfolgen suchen, die $R-I$ im Automaten erzeugen. I ist in beiden Regeln ein einzelner Knoten, wir suchen also für die erste Regel nach Folgen von zwei A-Kanten und für die zweite Regel nach Folgen von drei A-Kanten. Wir finden hierfür jeweils zwei Pfade. Für Regel 1 finden wir die folgenden Pfade:

- $z_u \xrightarrow{connect^1_{[1],A}} z_g \xrightarrow{connect^1_{[1],A}} z_u$
- $z_g \xrightarrow{connect^1_{[1],A}} z_u \xrightarrow{connect^1_{[1],A}} z_g$

und fügen entsprechend die Übergänge

$$z_u \xrightarrow{connect^1_{[1],B}} z_u$$

und

$$z_g \xrightarrow{connect^1_{[1],B}} z_g$$

hinzu. Für Regel 2 finden wir die folgenden Pfade:

- $z_u \xrightarrow{connect^1_{[1],A}} z_g \xrightarrow{connect^1_{[1],A}} z_u \xrightarrow{connect^1_{[1],A}} z_g$
- $z_g \xrightarrow{connect^1_{[1],A}} z_u \xrightarrow{connect^1_{[1],A}} z_g \xrightarrow{connect^1_{[1],A}} z_u$

und fügen entsprechend die Übergänge

$$z_u \xrightarrow{connect^1_{[1],C}} z_g$$

und

$$z_g \xrightarrow{connect^1_{[1],C}} z_u$$

hinzu. Den sich ergebenden Automaten können wir grafisch analog zu nicht-deterministischen Automaten wie folgt darstellen:

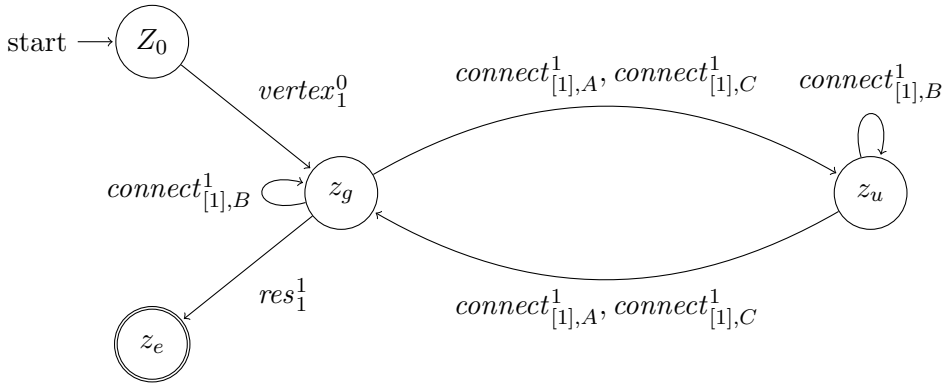


Abbildung 5.2: Ergebnis-Automat

Untersuchen wir diese Darstellung des Automaten, so sehen wir ein, dass der Automat die Sprache aller Graphen akzeptiert, die genau einen Knoten besitzen, sowie

- *Gerade viele A-Kanten enthält*
- *Gerade viele C-Kanten enthält*
- *Beliebig viele B-Kanten enthält*

oder aber

- *Ungerade viele A-Kanten enthält*
- *Ungerade viele C-Kanten enthält*
- *Beliebig viele B-Kanten enthält*

Kapitel 6

Abschluss unter Konkatenation

Ziel des Kapitels ist der Beweis des nachfolgenden Satzes.

Satz 6.1 *Die Menge der erkennbaren Sprachen über adhäsiven Kategorien ist abgeschlossen unter Konkatenation, das heißt, wenn L_1 eine erkennbare N, K-Sprache ist und L_2 eine erkennbare K, M-Sprache ist, dann ist $L_2 \circ L_1$ eine erkennbare N, M-Sprache.*

Im Fall der regulären Sprachen ist diese Aussage sehr einfach zu beweisen. Durch die Äquivalenz regulärer Sprachen zu den Sprachen, die sich durch reguläre Ausdrücke bilden lassen, sieht man ein, dass die Konkatenation zweier regulärer Sprachen wieder regulär ist. Aber auch wenn man im Automatenmodell arbeiten möchte, kann man eine Konstruktion einfach angeben. Man nimmt einfach einen Automaten A_1, der die Sprache L_1 akzeptiert und einen Automaten A_2, der die Sprache L_2 akzeptiert. Für jeden Übergang in Automat A_1, der von einem Zustand z in A_1 in einen Endzustand führt und jeden Startzustand z' von A_2, fügt man einen entsprechenden Übergang ein, der von z nach z' führt.

Im Fall der erkennbaren Graphsprachen ist die Konstruktion allerdings leider nicht so einfach möglich. Der Grund hierfür ist, dass in einem Graphautomaten der Graph in beliebiger Reihenfolge eingelesen werden kann. Wir können also nicht verlangen, dass zunächst der Teil des Graphen, der zu L_1 gehört, eingelesen wird und anschließend der Teil des Graphen, der zu L_2 gehört. Wir müssen demnach einen anderen Ansatz wählen, um die Aussage zu beweisen. Das gleiche Problem ergibt sich im Fall der adhäsiven Kategorien, auch hier müssen die Cospans in beliebiger Weise zerlegt werden können und wir können keine Reihenfolge verlangen, die uns ermöglichen würde, Automat $\mathcal{A}_2$ strikt nach Automat $\mathcal{A}_1$ auszuführen.

Um den Satz zu beweisen, geben wir zunächst die Definition eines Konkatenationsautomaten an, anschließend beweisen wir, dass dieser tatsächlich das Gewünschte leistet.

Definition 6.2 ***(Konkatenationsautomat)***
Gegeben seien ein Automat $\mathcal{A}_1 = (A_1, I_1, F_1)$ und ein Automat $\mathcal{A}_2 = (A_2, I_2, F_2)$, wobei $L(\mathcal{A}_1)$ eine A, K-Sprache ist und $L(\mathcal{A}_2)$ eine K, B-Sprache. Wir definieren einen Automaten $\mathcal{A} = (A, I, F)$ für den wir anschließend zeigen, dass er die Konkatenation $L(\mathcal{A}_2) \circ L(\mathcal{A}_1)$ akzeptiert. Wir definieren zunächst die Zustände. Die Zustände des Automaten $\mathcal{A}$ haben die Form

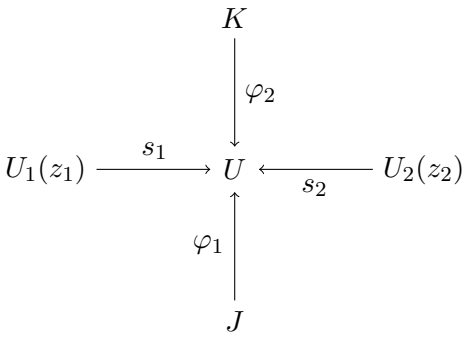


Abbildung 6.1: Zustände im Automaten $\mathcal{A}$

Dabei ist J das Interface, das zum Zustand in $\mathcal{A}$ gehört und K ist das Interface, das am Ende des Automaten $\mathcal{A}_1$ erreicht werden soll und von dem aus der Automat $\mathcal{A}_2$ beginnen soll. Weiter sei

$$U = J \cup K.$$

z_1 ist ein Zustand des Automaten $\mathcal{A}_1$ und U_1 ist das zugehörige Interface im Automaten $\mathcal{A}_1$. Es gilt also

$$z_1 \in \mathcal{A}_1(U_1).$$

Analog ist z_2 ein Zustand des Automaten $\mathcal{A}_2$ und U_2 das zugehörige Interface im Automaten $\mathcal{A}_2$. Es gilt also

$$z_2 \in \mathcal{A}_2(U_2).$$

Wir verlangen für einen Zustand, dass die Abbildungen φ_1, φ_2, s_1 und s_2 injektive Funktionen sind. Weiterhin sind φ_1 und φ_2 gemeinsam surjektiv - das Bild von φ_1 vereinigt mit dem Bild von φ_2 ergebe also ganz U - und s_1 und s_2 sind ebenfalls gemeinsam surjektiv. Schließlich sei noch

$$U_1 \cap U_2 = J \cap K.$$

Als Startzustände verwenden wir diejenigen Zustände, für die $K = U_2$ und $J = U_1$ gelten. Weiterhin verlangen wir $\varphi_1 = s_2$, $\varphi_2 = s_2$ sowie dass z_1 ein Startzustand in $\mathcal{A}_1$ ist und z_2 ein Startzustand in $\mathcal{A}_2$ ist. Für die Startzustände gelte also

$$z_1 \in I_1,\ z_2 \in I_2.$$

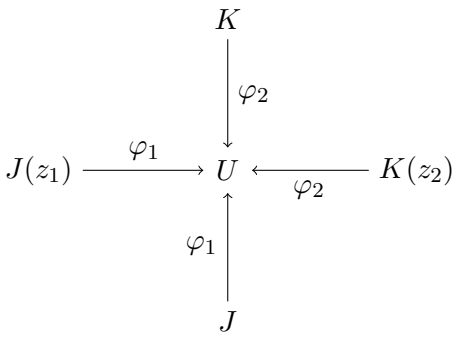


Abbildung 6.2: Startzustände im Automaten $\mathcal{A}$

Als Endzustände wählen wir all die Zustände, die die Form haben, dass $U_1 = K$, $U_2 = J$, $s_1 = \varphi_2$, $s_2 = \varphi_1$ sind und z_1 ein Endzustand in $\mathcal{A}_1$, z_2 ein Endzustand in $\mathcal{A}_2$ ist, bei denen also gilt

$$z_1 \in F_1,\ z_2 \in F_2.$$

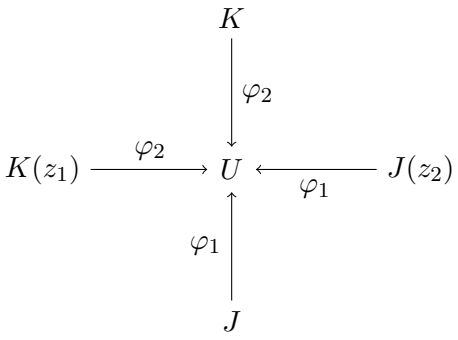


Abbildung 6.3: Endzustände im Automaten $\mathcal{A}$

Mit einem Cospan c ermöglichen wir nun einen Zustandsübergang von einem Zustand z in einen Zustand z', der aussieht wie in der Abbildung 6.4, genau dann wenn die folgenden fünf Bedingungen erfüllt sind:

1. $d_1 \cup d_2 = d$

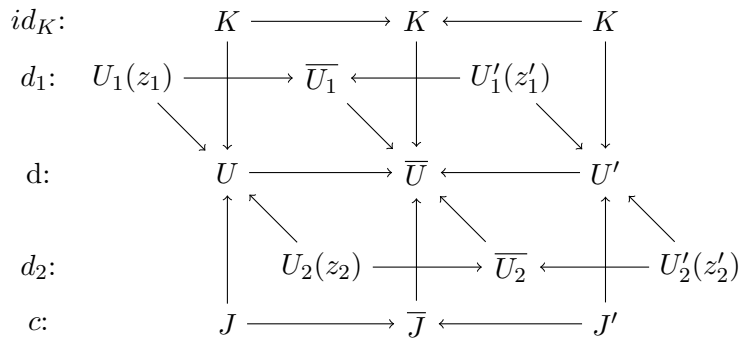


Abbildung 6.4: Zustandsübergänge im Automaten $\mathcal{A}$

2. $d_1 \cap d_2 = c \cap id_K$

3. $c \cup id_K = d$
   Diese ersten drei Bedingungen setzen unsere Bedingungen an die Zustände auf den Übergängen um.

4. d_1 *ist ein Cospan mit* $z_1 \mathcal{A}_1(d_1) z_1'$ *und* d_2 *ist ein Cospan mit* $z_2 \mathcal{A}_2(d_2) z_2'$. d_1 *ist also ein Cospan, der im Automaten* $\mathcal{A}_1$ *einen Zustandsübergang von* z_1 *nach* z_1' *erlaubt und* d_2 *ist ein Cospan, der im Automaten* $\mathcal{A}_2$ *einen Zustandsübergang von* z_2 *nach* z_2' *erlaubt.*

5. $\overline{U}_1 \cap K = U_1' \cap K$ *und* $\overline{U}_2 \cap K = U_2 \cap K$,
   wir verlangen also eine gewisse Monotonie von d_1 *und* d_2 *bezüglich der Inklusion von* K. U_1 *darf also bezüglich* K *nur wachsen, wohingegen* U_2 *bezüglich* K *nur schrumpfen kann. Betrachtet man Start- und Endzustände, so wird klar, wieso das der Fall ist. Um in einen Endzustand zu gelangen, muss* U_2 *genau* K *enthalten, um in einem Startzustand zu sein, muss* U_1 *genau* K *enthalten.*

Wir müssen nun zweierlei Dinge zeigen, einerseits dass der angegebene Automat tatsächlich ein Automat ist, andererseits, dass dieser Automat genau die Sprache akzeptiert, die die Konkatenation $L(\mathcal{A}_2) \circ L(\mathcal{A}_1)$ liefert. Diese Bedingungen lassen sich aufspalten in vier Teilbedingungen, die wir im Folgenden separat beweisen.

1. $A(c' \circ c) \supseteq \mathcal{A}(c') \circ \mathcal{A}(c)$
2. $\mathcal{A}(c' \circ c) \subseteq \mathcal{A}(c') \circ \mathcal{A}(c)$
3. $L(\mathcal{A}_2) \circ L(\mathcal{A}_1) \subseteq L(\mathcal{A})$
4. $L(\mathcal{A}_2) \circ L(\mathcal{A}_1) \supseteq L(\mathcal{A})$

Punkte 1 und 2 beweisen die Funktoreigenschaft von $\mathcal{A}$, Punkte 3 und 4, dass $\mathcal{A}$ genau die gewünschte Sprache akzeptiert. Wir beweisen diese Eigenschaften nun in den folgenden vier Sätzen, Eigenschaft 1 beweisen wir in Satz 6.3, Eigenschaft 2 in Satz 6.4, Eigenschaft 3 in Satz 6.6 und Eigenschaft 4 in Satz 6.7.

Satz 6.3 *Mit dem Konatenationsautomaten $\mathcal{A}$ wie in Definition 6.2 definiert und einem Cospan c, der sich schreiben lässt als $c = c_1 ; c_2$ für zwei Cospans c_1 und c_2, gilt: Falls es einen Zustand $\overline{z}$ gibt, so dass es einen $\mathcal{A}$-Übergang für c_1 von z nach $\overline{z}$ und einen $\mathcal{A}$-Übergang für c_2 von $\overline{z}$ nach z' gibt, gibt es auch einen $\mathcal{A}$-Übergang für c von z nach z'.*

Beweis: Gegeben sind $\mathcal{A}$-Übergänge für c_1 und c_2, die in den beiden folgenden Abbildungen dargestellt sind:

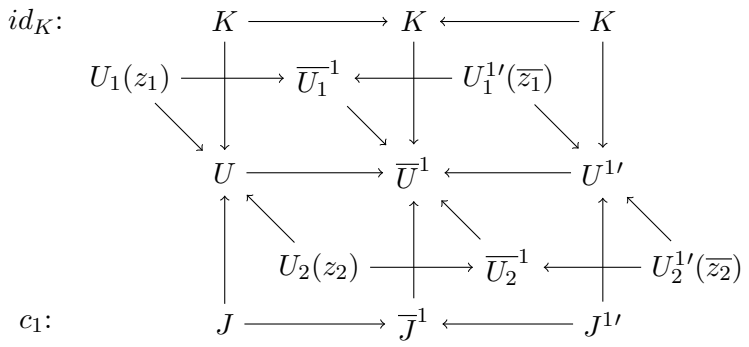


Abbildung 6.5: Zustandsübergang für c_1 im Automaten $\mathcal{A}$

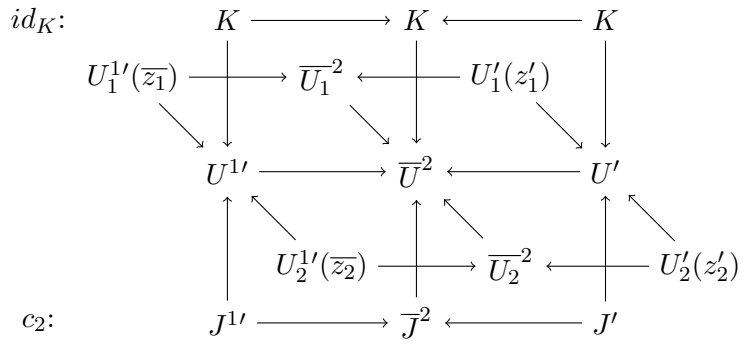


Abbildung 6.6: Zustandsübergang für c_2 im Automaten $\mathcal{A}$

Es gilt, da c, c_1 und c_2 Cospans sind:

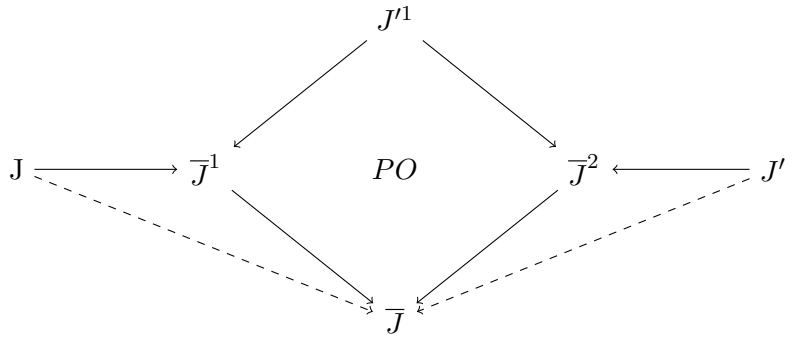


Abbildung 6.7: Pushout-Eigenschaft für $\overline{J}$

Analog bilden wir per Pushout die Objekte $\overline{U}$, $\overline{U}_1$ und $\overline{U}_2$:

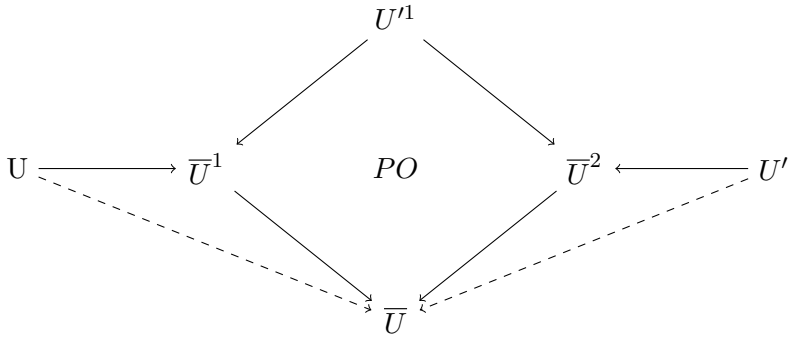


Abbildung 6.8: Pushout zur Bildung von $\overline{U}$

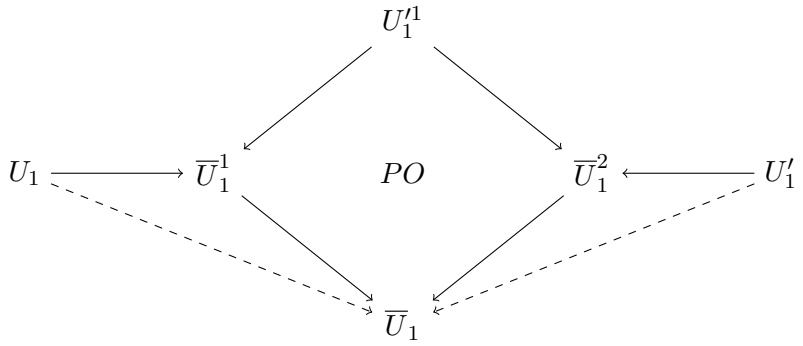


Abbildung 6.9: Pushout zur Bildung von $\overline{U}_1$

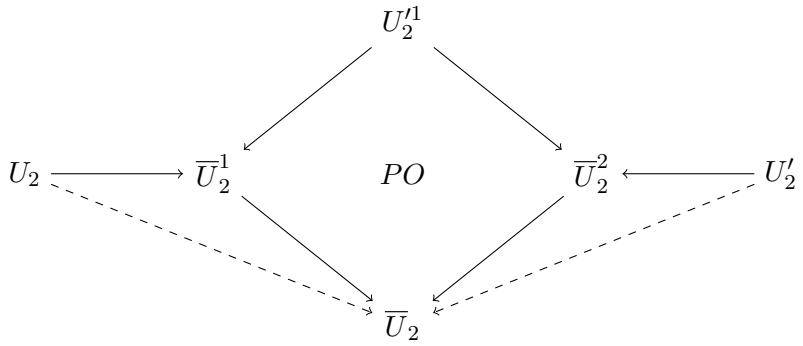


Abbildung 6.10: Pushout zur Bildung von $\overline{U}_2$

Wir bemerken gleich, dass es für $U_1 \to \overline{U}_1 \leftarrow U_1'$ einen Übergang von z_1 nach z_1' in $\mathcal{A}_1$ gibt, da $\mathcal{A}_1$ als Automat die Funktor-Eigenschaft besitzt. Ebenso gibt es für $U_2 \to \overline{U}_2 \leftarrow U_2'$ einen Übergang von z_2 nach z_2' in $\mathcal{A}_2$.
Die fehlenden Pfeile zwischen $\overline{J}$ und $\overline{U}$, zwischen $\overline{U}_1$ und $\overline{U}$, zwischen $\overline{U}_2$ und $\overline{U}$ sowie zwischen K und $\overline{U}$ entstehen als vermittelnde Morphismen (der eindeutige Morphismus, so dass das entstehende Diagramm kommutiert), wir geben hier nur an, wie man den Pfeil zwischen $\overline{J}$ und $\overline{U}$ erhält, die Situation für die anderen Paare ergeben sich analog durch Einsetzen der entsprechenden Pushouts.

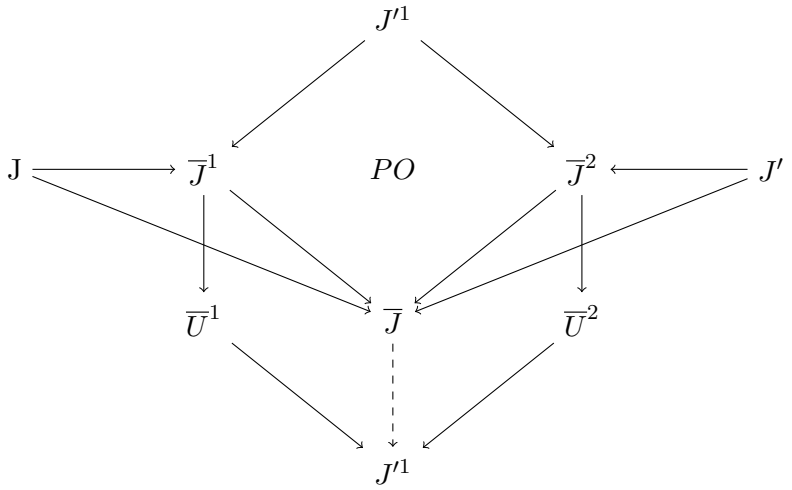


Abbildung 6.11: Gestrichelt: Der vermittelnde Morphismus zwischen $\overline{J}$ und $\overline{U}$

Die vermittelnden Morphismen existieren auf Grund des Pushouts, weil das Diagramm kommutiert. Sie sind zudem injektiv, wie Tobias Heindel in seiner Dissertation [11] gezeigt hat, da wir mit adhäsiven Kategorien arbeiten.
Wir haben nun den Zustandsübergang vollständig angegeben, es bleibt aber zu zeigen, dass es sich tatsächlich um einen Zustandsübergang in $\mathcal{A}$ handelt. Hierzu müssen wir die fünf Eigenschaften zeigen, die wir für die Existenz eines Zustandsübergangs vorausgesetzt hatten. Die *vierte* Eigenschaft haben wir bereits diskutiert und ist gegeben.

Die *fünfte* Eigenschaft ist ebenfalls einfach einzusehen.
Für den Automaten $\mathcal{A}_1$ berechnen wir zunächst:

$$\overline{U}_1 \cap K \underset{(\overline{U}_1 = \overline{U}_1^1 \cup \overline{U}_1^2)}{=} (\overline{U}_1^1 \cap K) \cup (\overline{U}_1^2 \cap K) \underset{(K-Monotonie)}{=} (U_1^{1\prime} \cap K) \cup (U_1' \cap K).$$

Da $U_1^{1\prime} \subseteq \overline{U}_1^2$ gilt, ist weiterhin

$$(U_1^{1\prime} \cap K) \subseteq (\overline{U}_1^2 \cap K) = (U_1' \cap K).$$

Also gilt insgesamt

$$\overline{U}_1 \cap K = (U_1' \cap K).$$

Für den Automaten $\mathcal{A}_2$ berechnen wir zunächst:

$$\overline{U}_2 \cap K = (\overline{U}_2^1 \cap K) \cup (\overline{U}_2^2 \cap K) = (U_2^{1\prime} \cap K) \cup (U_2 \cap K).$$

Da $U_2^{1\prime} \subseteq \overline{U}_2^1$ gilt, ist weiterhin

$$(U_2^{1\prime} \cap K) \subseteq (\overline{U}_2^1 \cap K) = (U_2 \cap K).$$

Also gilt insgesamt

$$\overline{U}_2 \cap K = (U_2 \cap K).$$

Die fünfte Eigenschaft gilt also.

Weil die beiden Übergänge, von denen wir ausgegangen sind, legale Übergänge in $\mathcal{A}$ sind, wissen wir bereits, dass $U_1 \cup U_2 = U$ und $U_1' \cup U_2' = U'$ richtig sind. Die *erste* Eigenschaft ($d_1 \cup d_2 = d$) bedarf daher nur noch des Beweises, dass $\overline{U}_1 \cup \overline{U}_2 = \overline{U}$ richtig ist.

$$\overline{U}_1 \cup \overline{U}_2 = (\overline{U}_1^1 \cup \overline{U}_1^2) \cup (\overline{U}_2^1 \cup \overline{U}_2^2) = (\overline{U}_1^1 \cup \overline{U}_2^1) \cup (\overline{U}_1^2 \cup \overline{U}_2^2) = \overline{U}^1 \cup \overline{U}^2 = \overline{U}.$$

Also ist Eigenschaft 1 ebenfalls bewiesen.

Ganz analog beweisen wir die *dritte* Eigenschaft ($c \cup id_K = d$). Wir wissen wieder, dass $J \cup K = U$ und $J' \cup K = U'$ richtig sein müssen, da die gegebenen Übergänge für c_1 und c_2 legale Zustandsübergänge in $\mathcal{A}$ sind. Es bleibt also nur noch zu zeigen, dass auch $\overline{J} \cup K = \overline{U}$ richtig ist:

$$\overline{J} \cup K = \overline{J}^1 \cup \overline{J}^2 \cup K = (\overline{J}^1 \cup K) \cup (\overline{J}^2 \cup K) = \overline{U}^1 \cup \overline{U}^2 = \overline{U}.$$

Demnach gilt auch die dritte Eigenschaft und es bleibt nur die zweite Eigenschaft zu zeigen.

Ein wenig komplizierter ist der Beweis der zweiten Eigenschaft ($d_1 \cap d_2 = c \cap id_K$). Wir müssen noch zeigen, dass $\overline{U}_1 \cap \overline{U}_2 = \overline{J} \cap K$ richtig ist, dass diese Eigenschaften für den linken und den rechten Zustand gelten ist wieder aus den gegebenen Übergängen für c_1 und c_2 klar. Wir teilen diese Aussage in zwei Teilaussagen:

- $\overline{J} \cap K \subseteq \overline{U}_1 \cap \overline{U}_2$: Wir berechnen:

  $$\overline{U}_1 \cap \overline{U}_2 = (\overline{U}_1^1 \cup \overline{U}_1^2) \cap (\overline{U}_2^1 \cup \overline{U}_2^2)$$
  $$= (\overline{U}_1^1 \cap \overline{U}_2^1) \cup (\overline{U}_1^1 \cap \overline{U}_2^2) \cup (\overline{U}_1^2 \cap \overline{U}_2^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^2).$$

  Wir betrachten nun hiervon die Teilmenge $(\overline{U}_1^1 \cap \overline{U}_2^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^2)$ und berechnen:

  $$(\overline{U}_1^1 \cap \overline{U}_2^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^2) = (\overline{J^1} \cap K) \cup (\overline{J^2} \cap K) = (\overline{J^1} \cup \overline{J^2}) \cap K = \overline{J} \cap K.$$

  Also gilt diese Richtung.

- $\overline{J} \cap K \supseteq \overline{U}_1 \cap \overline{U}_2$: Nach der Überlegung im ersten Teil ist noch zu zeigen, dass $(\overline{U}_1^1 \cap \overline{U}_2^2) \subseteq \overline{J} \cap K$ und $(\overline{U}_1^2 \cap \overline{U}_2^1) \subseteq \overline{J} \cap K$ richtig sind.

– $(\overline{U}_1^1 \cap \overline{U}_2^2) \subseteq \overline{J} \cap K$: Hierzu beachten wir die K-Monotonie, die wir für unsere Konstruktion bereits bewiesen haben und beachten

$$\overline{U}_1^1 \cap \overline{U}_2^2 = (\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1) \cup (\overline{U}_1^1 \cap \overline{U}_2^2 \cap K).$$

Das gilt, weil $\overline{U}_1^1 \subseteq \overline{J}^1 \cup K$ richtig ist. Wir berechnen:

$$\overline{U}_1^1 \cap \overline{U}_2^2 \cap K = \overline{U}_1^1 \cap U_2^{1\prime} \cap K \subseteq \overline{U}_1^1 \cap \overline{U}_2^1 = \overline{J}^1 \cap K \subseteq \overline{J} \cap K$$

sowie

$$\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1 = (\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1 \cap K)$$

Diese Umformung erfolgt unter Berücksichtigung von

$$\overline{U}_2^2 \subseteq \overline{J}^2 \cup K$$

Der zweite Teilausdruck, $(\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1 \cap K)$, ist in $\overline{J}^1 \cap K \subseteq \overline{J} \cap K$ enthalten. Betrachte also nur noch

$$\begin{aligned}
(\overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{J}^1 \cap \overline{J}^2) &= \overline{U}_1^1 \cap \overline{U}_2^2 \cap J^{1\prime} \\
&\underset{(J^{1\prime} \subseteq J^{1\prime} \cup K = U_2^{1\prime} \cup U_1^{1\prime})}{=} (\overline{U}_1^1 \cap \overline{U}_2^2 \cap J^{1\prime} \cap U_2^{1\prime}) \cup (\overline{U}_1^1 \cap \overline{U}_2^2 \cap J^{1\prime} \cap U_1^{1\prime}) \\
&\subseteq (\overline{U}_1^1 \cap \overline{U}_2^2 \cap J^{1\prime} \cap \overline{U}_2^1) \cup (\overline{U}_1^1 \cap \overline{U}_2^2 \cap J^{1\prime} \cap \overline{U}_1^2) \\
&\underset{(\overline{U}_1^1 \cap \overline{U}_2^1 = \overline{J}^1 \cap K \wedge \overline{U}_1^2 \cap \overline{U}_2^2 = \overline{J}^2 \cap K)}{\subseteq} (\overline{J}^1 \cap K) \cup (\overline{J}^2 \cap K) = \overline{J} \cap K.
\end{aligned}$$

Insgesamt gilt demnach die untersuchte Inklusion.

– $(\overline{U}_1^2 \cap \overline{U}_2^1) \subseteq \overline{J} \cap K$: Analog berechnen wir zunächst:

$$\overline{U}_1^2 \cap \overline{U}_2^1 = \overline{U}_1^2 \cap \overline{U}_2^1 \cap (\overline{J} \cup K) = (\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap K)$$

und untersuchen:

$$\begin{aligned}
&\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J} \\
&= (\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^2) \\
&= ((\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^1 \cap \overline{J}^2) \cup \underbrace{(\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^1 \cap K)}_{\subseteq \overline{J} \cap K}) \\
&\cup ((\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^2 \cap \overline{J}^1) \cup \underbrace{(\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^2 \cap K)}_{\subseteq \overline{J} \cap K})
\end{aligned}$$

Wir betrachten also nur noch den ersten und den dritten Teilterm, die zudem beide identisch sind. Dabei beachten wir noch

$J^{1\prime} \subseteq U_1^{1\prime} \cup U_2^{1\prime}$.

$$\begin{aligned}
\overline{U}_1^2 \cap \overline{U}_2^1 \cap \overline{J}^1 \cap \overline{J}^2 &= \overline{U}_1^2 \cap \overline{U}_2^1 \cap J^{1\prime} \\
&= (\overline{U}_1^2 \cap \overline{U}_2^1 \cap J^{1\prime} \cap U_1^{1\prime}) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap J^{1\prime} \cap U_2^{1\prime}) \\
&\subseteq (\overline{U}_1^2 \cap \overline{U}_2^1 \cap J^{1\prime} \cap \overline{U}_1^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap J^{1\prime} \cap \overline{U}_2^2) \\
&\subseteq (\overline{J}^1 \cap K) \cup (\overline{J}^2 \cap K) = \overline{J} \cap K.
\end{aligned}$$

Weiterhin berechnen wir unter Beachtung von $K \subseteq U_1^{1\prime} \cup U_2^{1\prime}$:

$$\begin{aligned}
&\overline{U}_1^2 \cap \overline{U}_2^1 \cap K \\
&= (\overline{U}_1^2 \cap \overline{U}_2^1 \cap K \cap U_1^{1\prime}) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap K \cap U_2^{1\prime}) \\
&\subseteq (\overline{U}_1^2 \cap \overline{U}_2^1 \cap K \cap \overline{U}_1^1) \cup (\overline{U}_1^2 \cap \overline{U}_2^1 \cap K \cap \overline{U}_2^2) \\
&= (\overline{J}^1 \cap K) \cup (\overline{J}^2 \cap K) = \overline{J} \cap K.
\end{aligned}$$

Also gilt auch diese Inklusion.

Damit haben wir alle Eigenschaften und somit auch den Satz bewiesen. □

Satz 6.4 *Mit dem Konkatenationsautomaten $\mathcal{A}$ wie in Definition 6.2 definiert und einem Cospan c, der sich schreiben lässt als $c = c_2 \circ c_1$ für zwei Cospans c_1 und c_2, gilt: Falls es einen $\mathcal{A}$-Übergang für c von z nach z' gibt, dann gibt es einen Zustand $\overline{z}$, so dass es einen $\mathcal{A}$-Übergang für c_1 von z nach $\overline{z}$ und einen $\mathcal{A}$-Übergang für c_2 von $\overline{z}$ nach z' gibt.*

Beweis: Gegeben ist also ein Übergang wie in der folgenden Abbildung zu sehen:

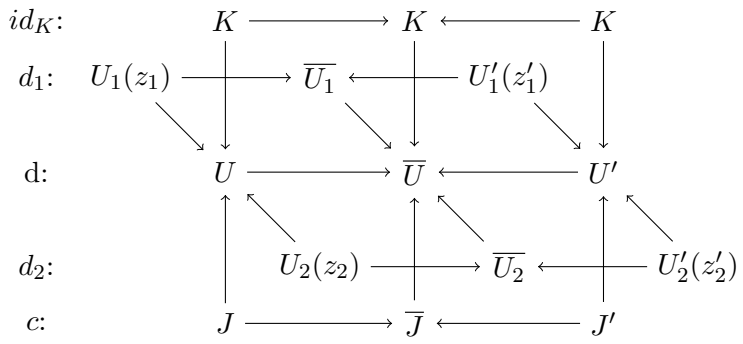


Abbildung 6.12: Zustandsübergänge im Automaten $\mathcal{A}$

Sowie zwei Cospans $c_1 : J \to \overline{J}^1 \leftarrow J^{1\prime}$ und $c_2 : J^{1\prime} \to \overline{J}_2 \leftarrow J'$ mit $c = c_1 \,;\, c_2$. Wir müssen nun jeweils einen Übergang in $\mathcal{A}$ für c_1 und c_2 finden, die konkateniert werden können und die gleichen Zustände in Relation setzen wie obiger Übergang für c. Hierzu bilden wir $\overline{U}^1$ und $\overline{U}^2$ per Pushout:

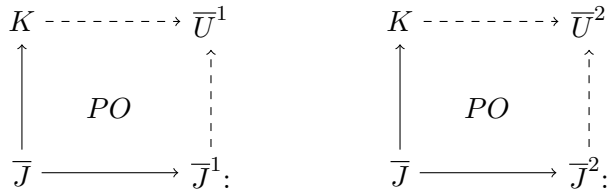


Abbildung 6.13: $\overline{U}^1$ und $\overline{U}^2$

Wir bilden auch $\overline{U}_1^1$, $\overline{U}_1^2$, $\overline{U}_2^1$ und $\overline{U}_2^2$ mit Hilfe von Pushouts und Pullbacks. Allerdings ist die Konstruktion komplizierter. Daher ist die Konstruktion nur mittels $\cup$ (Pushout) und $\cap$ (Pullback) angegeben.

$$\overline{U}_1^1 = (\overline{U}_1 \cap \overline{J}^1) \cup U_1$$

$$\overline{U}_1^2 = (\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K)$$
$$\overline{U}_2^1 = (\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K)$$

$$\overline{U}_2^2 = (\overline{U}_2 \cap \overline{J}^2) \cup U_2'$$

Per Pullback erhalten wir schließlich $U^{1\prime}$, $U_1^{1\prime}$ und $U_2^{1\prime}$:

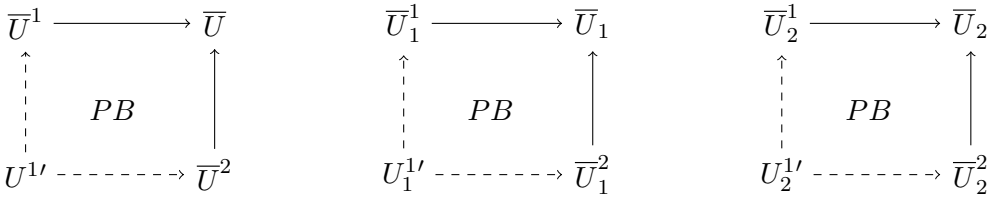


Abbildung 6.14: $\overline{U}^{1\prime}$, $\overline{U}_1^{1\prime}$ und $\overline{U}_2^{1\prime}$

Zunächst wollen wir *Eigenschaft 4* beweisen. Dazu zeigen wir, dass der d_1-Cospan des c-Übergangs heraus kommt, wenn man den d_1-Cospan des c_1-Übergangs und den d_1-Cospan des c_2-Übergangs konkateniert. Das gleiche gilt analog für den d_2-Cospan. Dass die Cospans jeweils konkatenierbar sind, folgt aus der Wahl des Zwischeninterfaces $U_1^{1\prime}$ respektive $U_2^{1\prime}$. Wir müssen also nur zeigen, dass gilt $\overline{U}_1^1 \cup \overline{U}_1^2 = \overline{U}_1$ und $\overline{U}_2^1 \cup \overline{U}_2^2 = \overline{U}_2$.

$$\begin{aligned} \overline{U}_1^1 \cup \overline{U}_1^2 &= (\overline{U}_1 \cap \overline{J}^1) \cup U_1 \cup (\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K) \\ = (\overline{J}^1 \cup \overline{J}^2) \cap \overline{U}_1 \cup U_1 \cup (\overline{U}_1 \cap K) &= (\overline{J} \cap \overline{U}_1) \cup (\overline{U}_1 \cap K) \cup U_1 \\ = \overline{U}_1 \cap (\overline{J} \cup K) \cup U_1 = \overline{U}_1 \cap \overline{U} \cup U_1 &= \overline{U}_1 \cup U_1 = \overline{U}_1 \end{aligned}$$

Weiterhin:

$$\begin{aligned}\overline{U}_2^1 \cup \overline{U}_2^2 &= (\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K) \cup (\overline{U}_2 \cap \overline{J}^2) \cup U_2' \\ = (\overline{J}^1 \cup \overline{J}^2) \cap \overline{U}_2 \cup (\overline{U}_2 \cap K) \cup U_2' &= (\overline{J} \cap \overline{U}_2) \cup (K \cap \overline{U}_2) \cup U_2' \\ &= \overline{U} \cap \overline{U}_2 \cup U_2' = \overline{U}_2 \cup U_2' = \overline{U}_2.\end{aligned}$$

Also gilt die vierte Eigenschaft, denn die Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$ erfüllen die Funktoreigenschaft.

Die *fünfte Eigenschaft* müssen wir für die sich ergebenden Teilübergänge separat beweisen. Wir beginnen mit dem c_1-Übergang.

- Zu zeigen ist einerseits $U_2 \cap K = \overline{U}_2^1 \cap K$ und andererseits $\overline{U}_1^1 \cap K = U_1^{1\prime} \cap K$.
  - $U_2 \cap K = \overline{U}_2^1 \cap K$: Die K-Monotonie für den c-Übergang liefert $U_2 \cap K = \overline{U}_2 \cap K$. Da außerdem $U_2 \subseteq \overline{U}_2^1 \subseteq \overline{U}_2$ gilt, muss auch $U_2 \cap K = \overline{U}_2^1 \cap K$ gelten.
  - $\overline{U}_1^1 \cap K = U_1^{1\prime} \cap K$: Wir beachten zunächst, dass gilt $U_1^{1\prime} = \overline{U}_1^1 \cap \overline{U}_1^2$, es ist also zu zeigen $\overline{U}_1^1 \cap K = \overline{U}_1^1 \cap K \cap \overline{U}_1^2$. Die Inklusion $\supseteq$ ist offensichtlich, wir zeigen also nur noch die Inklusion $\subseteq$. Diese gilt, wenn $\overline{U}_1^1 \cap K \subseteq U_1^2$ richtig ist. Daher berechnen wir:

    $$\overline{U}_1^1 \cap K = (\overline{U}_1 \cap \overline{J}^1 \cap K) \cup (U_1 \cap K)$$

    und

    $$\overline{U}_1^2 = (\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K).$$

    Wir stellen nun fest, dass gilt

    $$\overline{U}_1 \cap \overline{J}^1 \cap K \subseteq \overline{U}_1 \cap K$$

    und

    $$U_1 \cap K \subseteq \overline{U}_1 \cap K,$$

    also gilt die Inklusion.

- Für den c_2-Übergang sind die Relationen $U_1' \cap K = \overline{U}_1^2 \cap K$ und $\overline{U}_2^2 \cap K = U_2^{1\prime} \cap K$ zu beweisen.
  - $U_1' \cap K = \overline{U}_1^2 \cap K$: Wieder wegen der K-Monotonie des c-Übergangs gilt $U_1' \cap K = \overline{U}_1 \cap K$. Weiterhin gilt $U_1' \subseteq \overline{U}_1^2 \subseteq \overline{U}_1$, also gilt $U_1' \cap K = \overline{U}_1^2 \cap K$.

– Zeige nun: $\overline{U}_2^2 \cap K = U_2^{1\prime} \cap K$. Hierzu beachten wir wieder dass gilt

$$U_2^{1\prime} = \overline{U}_2^1 \cap \overline{U}_2^2.$$

Wir müssen also folgendes zeigen: $\overline{U}_2^2 \cap K = \overline{U}_2^1 \cap \overline{U}_2^2 \cap K$. Die Inklusion $\supseteq$ ist wiederum offensichtlich, wir zeigen also nur die Inklusion $\subseteq$. Hierzu reicht es, zu zeigen, dass $\overline{U}_2^2 \cap K \subseteq \overline{U}_2^1$ richtig ist. Da gilt (Einsetzen der Definition von $\overline{U}_2^2$ und „Durchmultiplizieren“ des Schnitts)

$$\overline{U}_2^2 \cap K = (\overline{U}_2 \cap \overline{J}^2 \cap K) \cup (U_2' \cap K)$$

und

$$\overline{U}_2^1 = (\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K)$$

merken wir an, dass

$$\overline{U}_2 \cap \overline{J}^2 \cap K \subseteq \overline{U}_2 \cap K$$

und

$$U_2' \cap K \subseteq \overline{U}_2 \cap K$$

richtig sind, die Inklusion also gilt und somit die gesamte Relation korrekt ist.

Damit haben wir Eigenschaft 5 ebenfalls bewiesen.

Für Eigenschaft 1, $d_1 \cup d_2 = d$ müssen wir insgesamt fünf Gleichungen zeigen:

1. $U_1 \cup U_2 = U$
2. $\overline{U}_1^1 \cup U_2^1 = \overline{U}_1$
3. $U_1^{1\prime} \cup U_2^{1\prime} = U$
4. $\overline{U}_1^2 \cup \overline{U}_2^2 = \overline{U}^2$
5. $U_1' \cup U_2' = U'$

Gleichungen 1 und 5 sind bereits klar, weil der c-Übergang ein legaler Zustandsübergang in $\mathcal{A}$ ist, wir zeigen also nur Gleichungen 2 bis 4.
Für Gleichung 2 berechnen wir:

$$\begin{aligned}
\overline{U}_1^1 \cup \overline{U}_2^1 &= (\overline{U}_1 \cap \overline{J}^1) \cup U_1 \cup (\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K) \\
&= \overline{J}^1 \cap (\overline{U}_1 \cup \overline{U}_2) \cup \overline{U}_2 \cap (\overline{J}^1 \cup K) \cup U_1 \\
&= (\overline{J}^1 \cap \overline{U}) \cup (\overline{J}^1 \cup K) \cap \overline{U}_2 \cup U_1 \\
&= \overline{J}^1 \cup (\overline{J}^1 \cup K) \cap \overline{U}_2 \cup U_1 \\
&\underset{(\overline{J}^1 \cap \overline{U}_2 \subseteq \overline{J}^1 \wedge U_1 = U_1 \cup (U_1 \cap K))}{=} \overline{J}^1 \cup (K \cap \overline{U}_2) \cup (U_1 \cap K) \cup U_1 \\
&\underset{K-Monotonie}{=} \overline{J}^1 \cup (K \cap U_2) \cup (K \cup U_1) \cup U_1 \\
&= \overline{J}^1 \cup (K \cap (U_1 \cup U_2)) \cup U_1 \\
&= \overline{J}^1 \cup (K \cap (U)) \cup U_1 \\
&= \overline{J}^1 \cup K \cup U_1 \\
&= \overline{U}^1 \cup U_1 = \overline{U}^1
\end{aligned}$$

Gleichung 3 berechnen wir wie folgt:

$$\begin{aligned}
U_1^{1\prime} \cup U_2^{1\prime} &= \overline{U}_1^2 \cap \overline{U}_1^1 \cup \overline{U}_2^1 \cap \overline{U}_2^2 \\
&= (((\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K)) \cap ((\overline{U}_1 \cap \overline{J}^1) \cup U_1)) \\
&\cup (((\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K)) \cap ((\overline{U}_2 \cap \overline{J}^2) \cup U_2')) \\
&= \underline{(\overline{U}_1 \cap \overline{J}^1 \cap \overline{J}^2)} \cup (\overline{U}_1 \cap U_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K \cap \overline{J}^1) \cup (\overline{U}_1 \cap U_1 \cap K) \\
&\cup \underline{(\overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2)} \cup (\overline{U}_2 \cap \overline{J}^1 \cap U_2') \cup (\overline{U}_2 \cap K \cap \overline{J}^2) \cup (\overline{U}_2 \cap K \cap U_2')
\end{aligned}$$

Der Übersicht halber erfolgt nun eine kleine Nebenrechnung, die die beiden unterstrichenen Terme zusammenfasst.

$$\begin{aligned}
&(\overline{U}_1 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \\
&= (\overline{U}_1 \cup \overline{U}_2) \cap (\overline{J}^1 \cap \overline{J}^2) \\
&= \overline{U} \cap (\overline{J}^1 \cap \overline{J}^2) = \overline{J}^1 \cap \overline{J}^2
\end{aligned}$$

Das Ergebnis der Nebenrechnung setzen wir in unsere obige Gleichung ein und fassen außerdem alle Terme zusammen, die ein K enthalten. Wir erhalten:

$$\begin{aligned}
&(\overline{J}^1 \cap \overline{J}^2) \cup K \cap ((\overline{J}^1 \cap \overline{U}_1) \cup (\overline{U}_1 \cap U_1) \cup (\overline{U}_2 \cap \overline{J}^2) \cup (\overline{U}_2 \cap U_2')) \\
&\qquad \cup \underbrace{(\overline{U}_1 \cap U_1 \cap \overline{J}^2)}_{\subseteq (\overline{J}^2 \cap \overline{J}^1)} \cup \underbrace{(\overline{U}_2 \cap \overline{J}^1 \cap U_2')}_{\subseteq (\overline{J}^2 \cap \overline{J}^1)} \\
&= (\overline{J}^1 \cap \overline{J}^2) \cup K \cap (\underbrace{\overline{U}_1 \cap (\overline{J}^1 \cup U_1)}_{\supseteq U_1} \cup \underbrace{\overline{U}_2 \cap (\overline{J}^2 \cup U_2))}_{\supseteq U_2} \\
&\qquad \underset{(U_1 \cup U_2 = U \supseteq K)}{=} \overline{J}^1 \cap \overline{J}^2 \cup K
\end{aligned}$$

Andererseits gilt

$$\begin{aligned}\overline{U}^1 \cap \overline{U}^2 = (K \cup \overline{J}^1) \cap (K \cup \overline{J}^2) = \\ K \cup \underbrace{(K \cap \overline{J}^2)}_{\subseteq K} \cup \underbrace{(\overline{J}^1 \cap K)}_{\subseteq K} \cup (\overline{J}^1 \cap \overline{J}^2) \\ = K \cup (\overline{J}^1 \cap \overline{J}^2)\end{aligned}$$

Also sind beide Ausdrücke gleich.

Bleibt schließlich die vierte Gleichung zu zeigen. Wir berechnen:

$$\begin{aligned}\overline{U}_1^2 \cup \overline{U}_2^2 &= (\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K) \cup (\overline{U}_2 \cap \overline{J}^2) \cup U_2' \\ &= (\overline{U}_1 \cup \overline{U}_2) \cap \overline{J}^2 \cup (\overline{U}_1 \cap K) \cup U_2' \\ &= \overline{U} \cap \overline{J}^2 \cup (\overline{U}_1 \cap K) \cup U_2' \\ &= \overline{J}^2 \cup (\overline{U}_1 \cap K) \cup U_2' \\ &\underset{(U_2' \subseteq U' = J' \cup K \subseteq \overline{J}^2 \cup K)}{=} \overline{J}^2 \cup K \cap (\overline{U}_1 \cup U_2' \cup \overline{J}^2).\end{aligned}$$

Da definitionsgemäß $\overline{U}^2 = \overline{J}^2 \cup K$ gilt, muss für Gleichheit der beiden Ausdrücke folgendes gezeigt werden: $K = K \cap (\overline{U}_1 \cup U_2' \cup \overline{J}^2)$.
Es gilt wegen der K-Monotonie

$$\overline{K} \cap (U_1 \cup U_2') = K \cap (U_1' \cup U_2') = K \cap (J' \cup K) = K.$$

Also ist $L \cap (\overline{U}_1 \cup U_2') \subseteq K$ und damit

$$K \cap (\overline{U}_1 \cup U_2' \cup \overline{J}^2) \subseteq K \cap K = K.$$

Insgesamt gilt Eigenschaft 1 also.

Um die zweite Eigenschaft zu beweisen, sind wiederum fünf Gleichungen zu beweisen:

1. $U_1 \cap U_2 = J \cap K$
2. $\overline{U}_1^1 \cap \overline{U}_2^1 = \overline{J}^1 \cap K$
3. $J^{1\prime} \cap K = J^{1\prime} \cap K$
4. $\overline{J}^2 \cap K = \overline{U}_1^2 \cap \overline{U}_2^2$
5. $J' \cap K = U_1' \cap U_2'$

Gleichungen 1 und 5 sind bereits vom c-Übergang her bekannt, wir zeigen also nur die Gleichungen zwei bis vier.
Für die zweite Gleichung berechnen wir:

$$\begin{aligned}\overline{U}_1^1 \cap \overline{U}_2^1 &= (\overline{U}_1 \cap \overline{J}^1 \cup U_1) \cap ((\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K)) \\ &= \underline{(\overline{U}_1 \cap \overline{J}^1 \cap \overline{U}_2) \cup (\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap K)} \cup (U_1 \cap \overline{U}_2 \cap \overline{J}^1) \cup (U_1 \cap \overline{U}_2 \cap K)\end{aligned}$$

Wir können die beiden unterstrichenen Teilterme zusammenfassen, wenn wir beachten, dass $\overline{U}_1 \cap \overline{U}_2 = \overline{J} \cap K$. Dann fallen nämlich beide Teilterme zu $\overline{J}^1 \cap K$ zusammen und es ergibt sich:

$$(\overline{J}^1 \cap K) \cup (U_1 \cap \overline{U}_2 \cap \overline{J}^1) \cup (U_1 \cap \overline{U}_2 \cap K)$$

Nun ist aber $U_1 \subseteq \overline{U}_1$ und somit ist $U_1 \cap \overline{U}_2 \subseteq \overline{U}_1 \cap \overline{U}_2 = \overline{J}^1 \cap K$. Insgesamt fällt der obige Term also zusammen zu $\overline{J}^1 \cap K$, die zweite Gleichung ist also korrekt.

Wir berechnen:

$$\begin{aligned}U_1^{1\prime} \cap U_2^{1\prime} &= \overline{U}_1^2 \cap \overline{U}_1^1 \cap \overline{U}_2^2 \cap \overline{U}_2^1 \\ &= ((\overline{U}_1 \cap \overline{J}^1) \cup U_1) \cap ((\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K)) \\ &\cap ((\overline{U}_2 \cap \overline{J}^1) \cup (\overline{U}_2 \cap K)) \cap ((\overline{U}_2 \cap \overline{J}^2) \cup U_2') \\ &= ((\overline{U}_1 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap \overline{J}^1 \cap K) \cup (U_1 \cap \overline{U}_1 \cap \overline{J}^2) \cup (U_1 \cap \overline{U}_1 \cap K)) \\ &\cap ((\overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_2 \cap \overline{J}^1 \cap U_2') \cup (\overline{U}_2 \cap K \cap \overline{J}^2) \cup (\overline{U}_2 \cap U_2' \cap K))\end{aligned}$$

Da $U_1 \subseteq \overline{U}_1$ und $U_2' \subseteq \overline{U}_2$, können wir einige Teilterme ein wenig vereinfachen:

$$\begin{aligned}&= ((\overline{U}_1 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap \overline{J}^1 \cap K) \cup (U_1 \cap \overline{J}^2) \cup (U_1 \cap K)) \\ &\cap ((\overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \cup (U_2' \cap \overline{J}^1) \cup (\overline{U}_2 \cap K \cap \overline{J}^2) \cup (U_2' \cap K)) \\ &= (\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \cup \underline{(\overline{U}_1 \cap \overline{J}^1 \cap \overline{J}^2 \cap U_2')} \\ &\cup \underline{(\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2 \cap K)} \cup \underline{(\overline{U}_1 \cap U_2' \cap \overline{J}^1 \cap \overline{J}^2 \cap K)} \\ &\cup \underline{(\overline{U}_1 \cap \overline{J}^1 \cap K \cap \overline{U}_2 \cap \overline{J}^2)} \cup (\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2') \cup \underline{(\overline{U}_1 \cap \overline{J}^1 \cap K \cap \overline{J}^2 \cap \overline{U}_2)} \\ &\cup (\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2') \cup \underline{(U_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2)} \cup \underline{(U_1 \cap U_2' \cap \overline{J}^1 \cap \overline{J}^2)} \\ &\cup (U_1 \cap \overline{J}^2 \cap \overline{U}_2 \cap K) \cup (U_1 \cap U_2' \cap \overline{J}^2 \cap K) \cup \underline{(U_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2 \cap K)} \\ &\cup (U_1 \cap U_2' \cap \overline{J}^1 \cap K) \cup \underline{(U_1 \cap \overline{U}_2 \cap K \cap \overline{J}^2)} \cup (U_1 \cap U_2' \cap K)\end{aligned}$$

Nun können wir die unterstrichenen Teilterme mit dem ersten Teilterm zusammenlegen, denn alle diese Teilterme sind im hervorgehobenen Teilterm

enthalten. Wir beachten hierbei, dass $\overline{J}^1 \cap \overline{J}^2 \cap K \subseteq \overline{J} \cap K \subseteq \overline{U}_1 \cap \overline{U}_2$ richtig ist und erhalten:

$$\begin{aligned} &= (\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \cup \underline{(\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2')} \\ &\cup \underline{(\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2')} \cup (U_1 \cap \overline{J}^2 \cap \overline{U}_2 \cap K) \\ \cup (U_1 \cap U_2' \cap \overline{J}^2 \cap K) &\cup \underline{(U_1 \cap U_2' \cap \overline{J}^1 \cap K)} \cup (U_1 \cap U_2' \cap K) \end{aligned}$$

Hier und im Folgenden lassen sich die unterstrichenen Terme zusammenfassen.

$$\begin{aligned} &= (\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2') \cup \underline{(U_1 \cap \overline{J}^2 \cap \overline{U}_2 \cap K)} \\ &\cup \underline{(U_1 \cap U_2' \cap \overline{J}^2 \cap K)} \cup (U_1 \cap U_2' \cap K) \\ &= (\underbrace{\overline{U}_1 \cap \overline{U}_2}_{\overline{U}_1 \cap \overline{U}_2 = K \cap \overline{J} \subseteq K, K \cap \overline{J} \supseteq K \cap J^{1\prime}} \cap \underbrace{\overline{J}^1 \cap \overline{J}^2}_{J^{1\prime}}) \cup (\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2') \\ &\cup (U_1 \cap \overline{J}^2 \cap \overline{U}_2 \cap K) \cup (U_1 \cap U_2' \cap K) \\ = (K \cap J^{1\prime}) &\cup (\overline{U}_1 \cap \overline{J}^1 \cap K \cap U_2') \cup (U_1 \cap \overline{J}^2 \cap \overline{U}_2 \cap K) \cup (U_1 \cap U_2' \cap K) \end{aligned}$$

Wir wollen zeigen, dass dieser Ausdruck gleich $(K \cap J^{1\prime})$ ist. Hierzu zeigen wir dreierlei:

1. $\overline{U}_1 \cap U_2' \cap \overline{J}^1 \subseteq J^{1\prime}$
2. $U_1 \cap \overline{U}_2 \cap \overline{J}^2 \subseteq J^{1\prime}$
3. $U_1 \cap U_2' \subseteq J^{1\prime}$

Zur ersten Inklusion: Die K-Monotonie liefert $\overline{U}_1 \cup U_2' = U_1' \cap U_2'$, also gilt

$$\overline{U}_1 \cap U_2' \cap \overline{J}^1 = U_1' \cap U_2' \cap \overline{J}^1 \subseteq \overline{U}_1^2 \cap \overline{U}_2^2 \cap \overline{J}^1 \subseteq \overline{J}^2 \cap \overline{J}^1 = J^{1\prime}$$

Also gilt die erste Inklusion.
Zur zweiten Inklusion: Die K-Monotonie liefert $U_1 \cap \overline{U}_2 = U_1 \cap U_2$, also gilt

$$U_1 \cap \overline{U}_2 \cap \overline{J}^2 = U_1 \cap U_2 \cap \overline{J}^2 \subseteq \overline{U}_1^1 \cap \overline{U}_2^1 \cap \overline{J}^2 = \overline{J}^1 \cap \overline{J}^2 \cap K \subseteq \overline{J}^1 \cap \overline{J}^2 = J^{1\prime}$$

Zur dritten Inklusion: Die K-Monotonie liefert, unter der Beachtung, dass $U_1 \subseteq \overline{U}_1$, $U_2 \subseteq \overline{U}_2$ und $\overline{U}_1 \cap \overline{U}_2 \subseteq K$ richtig sind:

$$\begin{aligned} U_1 \cap U_2' &\underset{(\subseteq K)}{=} U_1 \cap U_2' \cap K \underset{(U_2' \subseteq \overline{U}_2)}{\subseteq} U_1 \cap \overline{U}_2 \cap K \\ &\underset{(K-Monotonie)}{=} U_1 \cap U_2 \cap K = J \cap K \subseteq \overline{J}^1 \end{aligned}$$

weiterhin folgt ganz analog:

$$U_1 \cap U_2' \subseteq \overline{U}_1 \cap U_2' \cap K = U_1' \cap U_2' \cap K \subseteq \overline{J}^2.$$

Insgesamt gilt also:

$$U_1 \cap U_2' \subseteq \overline{J}^1 \cap \overline{J}^2$$

wie zu zeigen war.

Die dritte Gleichung gilt damit also insgesamt auch. Bleibt die vierte Gleichung zu beweisen:

$$\begin{aligned}\overline{U}_1^2 \cap \overline{U}_2^2 &= ((\overline{U}_1 \cap \overline{J}^2) \cup (\overline{U}_1 \cap K)) \cap ((\overline{U}_2 \cap \overline{J}^2) \cup U_2') \\ = \underline{(\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^2)} \cup (\overline{U}_1 \cap U_2' \cap \overline{J}^2) &\cup \underline{(\overline{U}_1 \cap \overline{U}_2 \cap \overline{J}^2 \cap K)} \cup (\overline{U}_1 \cap K \cap U_2')\end{aligned}$$

Für die beiden unterstrichenen Teilterme beachten wir: $\overline{U}_1 \cap \overline{U}_2 = \overline{J} \cap K$ sowie $\overline{J} \cap K \cap \overline{J}^2 = \overline{J}^2 \cap K$, beide Teilterme fallen also zusammen zu $\overline{J}^2 \cap K$. Es ergibt sich:

$$(\overline{J}^2 \cap K) \cup (\overline{U}_1 \cap U_2' \cap \overline{J}^2) \cup (\overline{U}_1 \cap K \cap U_2')$$

Für die rechten beiden Teilterme berechnen wir nun:

$$(\overline{U}_1 \cap U_2' \cap \overline{J}^2) \subseteq (\underbrace{\overline{U}_1 \cap \overline{U}_2}_{\overline{J} \cap K} \cap \overline{J}^2) = (\overline{J}^2 \cap K)$$

sowie

$$\overline{U}_1 \cap K \cap U_2' \subseteq (\overline{U}_1 \cap \overline{U}_2 \cap K) = (\overline{J}^2 \cap K)$$

Also fällt der gesamte Term zusammen zu $\overline{J}^2 \cap K$. Das ist genau das was zu zeigen war. Die vierte Gleichung - und damit auch die zweite Eigenschaft - gilt also.

Zum Beweis der dritten Eigenschaft, $c \cup id_K = d$ bemerken wir, dass $J \cup K = U$ und $J' \cup K = U'$ bereits vom c-Übergang her bekannt sind, weiterhin gelten $\overline{J}^1 \cup K = \overline{U}^1$ und $\overline{J}^2 \cup K = \overline{U}^2$ definitionsgemäß. Bleibt also nur zu zeigen dass auch $U^{1\prime} = J^{1\prime} \cup K$ richtig ist. Wir berechnen:

$$\begin{aligned}U^{1\prime} = \overline{U}^1 \cap \overline{U}^2 &= (\overline{J}^1 \cup K) \cap (\overline{J}^2 \cup K) \\ = \overline{J}^1 \cap \overline{J}^2 \cup K \cup (\overline{J}^1 \cap K) &\cup (\overline{J}^2 \cap K) \\ = J^{1\prime} \cup K \cup \underbrace{\overline{J}^1 \cap K}_{\subseteq K} \cup \underbrace{\overline{J}^2 \cap K}_{\subseteq K} &= J^{1\prime} \cup K.\end{aligned}$$

Da also alle fünf Eigenschaften gelten, ist der Satz bewiesen. Insgesamt folgt:

Folgerung 6.5 *Der in Definition 6.2 beschriebene Automat $\mathcal{A}$ erfüllt die Funktor-Eigenschaft, ist also tatsächlich ein Automat*

Wir werden nun in den folgenden zwei Sätzen beweisen, dass der konstruierte Automat auch tatsächlich die korrekte Sprache akzeptiert.

Satz 6.6 *Es sei $c \in L(\mathcal{A}_1)\,;L(\mathcal{A}_2)$ gegeben, dann ist $c \in L(\mathcal{A})$*

Beweis: Wenn $c \in L(\mathcal{A}_1)\,;L(\mathcal{A}_2)$ gilt, dann gibt es ein $c_1 \in L(\mathcal{A}_1)$ und ein $c_2 \in L(\mathcal{A}_2)$, so dass $c = c_1\,;c_2$ gilt. Wir betrachten nun zunächst c_1.
c_1 hat die Form $c_1 = U_1 \to \overline{U} \leftarrow K$, wir bilden den Übergang, der in der folgenden Abbildung zu sehen ist. Dabei ist z_{s_1} der Startzustand des c_1-Übergangs in $\mathcal{A}_1$, z_{s_2} der Startzustand des C_2-Übergangs in $\mathcal{A}_2$, z_{e_1} der Endzustand des c_1-Übergangs in $\mathcal{A}_1$ und z_{e_2} der Endzustand des c_2-Übergangs in $\mathcal{A}_2$.

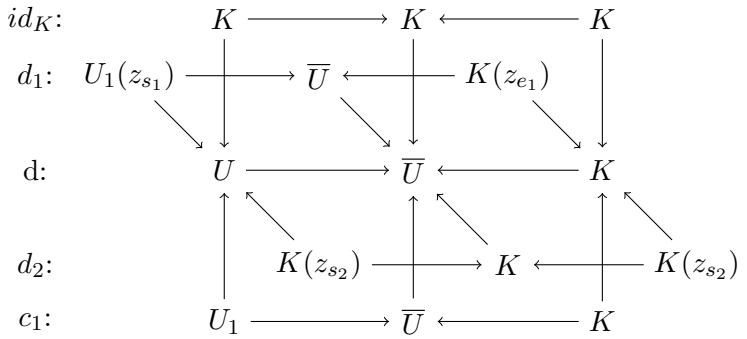


Abbildung 6.15: Zustandsübergang im Automaten $\mathcal{A}$

Außer U sind alle Objekte und zugehörigen Pfeile bereits bekannt, U bilden wir als Pushout:

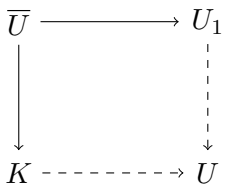


Abbildung 6.16: Pushout für U

Wir zeigen nun die fünf Eigenschaften, die erfüllt sein müssen, damit es sich hierbei um einen Übergang in $\mathcal{A}$ handelt:

1. $d_1 \cup d_2 = d$: Da $K \subseteq \overline{U}$ richtig ist, gilt $\overline{U} = \overline{U} \cup K$, $K \cup K = K$ gilt ebenfalls und wir haben U als Pushout von U_1 und K definiert, also ist auch $U = U_1 \cup K$.

2. $d_1 \cap d_2 = c_1 \cap id_K$: $K \cap K = K \cap K$ ist klar, ebenso $\overline{U} \cap K = \overline{U} \cap K$ und $K \cap U_1 = K \cap U_1$.

3. $c_1 \cup id_K = d$: Wir haben U als Pushout von K und U_1 definiert, also gilt $U = K \cup U_1$, da $K \subseteq \overline{U}$ richtig ist, gilt auch $\overline{U} \cup K = \overline{U}$ und $K \cup K = K$ ist klar.

4. d_1 ist ein Cospan mit $z_{s_1}\mathcal{A}_1(c_1)z_{e_1}$ nach Voraussetzung. Da der Cospan d_2 schlicht die Identität ist und keine Zustandsänderung vom inneren zum äußeren Interface stattfindet, gilt auch, dass d_2 ein Cospan ist, für den $z_{s_1}\mathcal{A}_2(d_2)z_{s_1}$ gilt.

5. $\overline{U} \cap K = K \cap K$ folgt aus $\overline{U} \supseteq K$ und $K \cap K = K \cap K$ ist klar.

Also gelten alle Eigenschaften, die ein Übergang in $\mathcal{A}$ erfüllen muss für diesen Übergang und es handelt sich somit um einen legalen Zustandsübergang.

Ganz analog bilden wir nun einen Zustandsübergang für c_2 in $\mathcal{A}$ wie folgt:

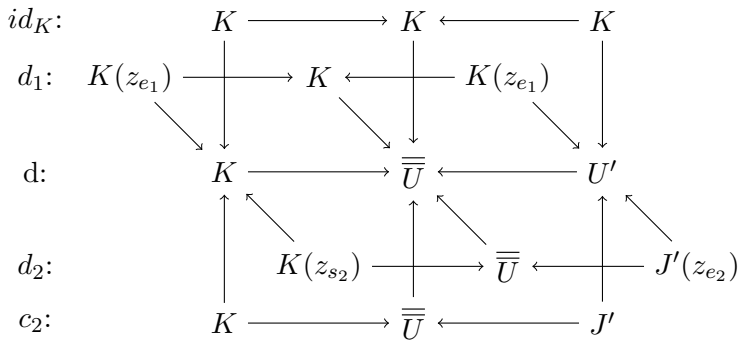


Abbildung 6.17: Zustandsübergang im Automaten $\mathcal{A}$

Wiederum sind alle Objekte und Pfeile mit einer Ausnahme bekannt. U' bilden wir analog zu U im ersten Übergang als Pushout-Objekt von J' und K:

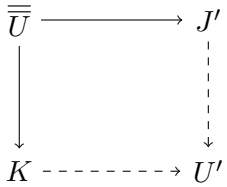


Abbildung 6.18: Pushout für U'

Auch hier zeigen wir, dass die fünf Bedingungen gelten, die an einen Zustandsübergang in $\mathcal{A}$ gestellt werden.

1. $d_1 \cup d_2 = d$: Da $K \subseteq \overline{\overline{U}}$ richtig ist, gilt $\overline{\overline{U}} = \overline{\overline{U}} \cup K$, $K \cup K = K$ gilt ebenfalls und wir haben U' als Pushout von J' und K definiert, also ist auch $U' = J' \cup K$.

2. $d_1 \cap d_2 = c_2 \cap id_K$: $K \cap K = K \cap K$ ist klar, ebenso $\overline{\overline{U}} \cap K = \overline{\overline{U}} \cap K$ und $K \cap J' = K \cap J'$.

3. $c_2 \cup id_K = d$: Wir haben U' als Pushout von K und J' definiert, also gilt $U' = K \cup J'$, da $K \subseteq \overline{\overline{U}}$ richtig ist, gilt auch $\overline{\overline{U}} \cup K = \overline{\overline{U}}$ und $K \cup K = K$ ist klar.

4. d_1 ist ein Cospan mit $z_{e_1}\mathcal{A}_1(c_2)z_{e_1}$, weil es sich um die Identität handelt und die beiden Zustände identisch sind. d_2 ist ein Cospan mit $z_{s_2}\mathcal{A}_2(c_2)z_{e_2}$ nach Voraussetzung.

5. $\overline{\overline{U}} \cap K = K \cap K$ folgt aus $\overline{\overline{U}} \supseteq K$ und $K \cap K = K \cap K$ ist klar.

Beide Zustandsübergänge sind also legale Zustandsübergänge in $\mathcal{A}$, da wir bereits wissen, dass $\mathcal{A}$ die Funktor-Eigenschaft erfüllt und $c = c_1 ; c_2$ ist, werden der Startzustand vom ersten definierten Übergang und der Endzustand vom letzten definierten Übergang durch $\mathcal{A}$ in Relation gesetzt. Da diese Zustände (wie man durch einfaches Abgleichen mit der Definition von Start- und Endzuständen ersehen kann) tatsächliche Start- und Endzustände sind, akzeptiert $\mathcal{A}$ also den Cospan c. □

Satz 6.7 *Es sei $c \in L(\mathcal{A})$ gegeben, dann ist $c \in L(\mathcal{A}_1) ; L(\mathcal{A}_2)$*

Beweis: Sei ein akzeptierender c-Übergang in $\mathcal{A}$ gegeben:

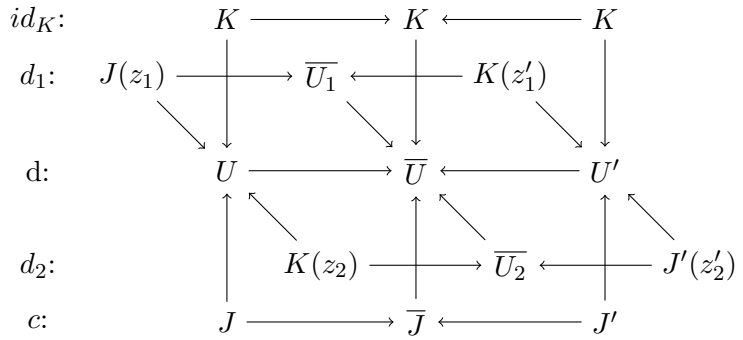


Abbildung 6.19: Akzeptierender Zustandsübergang im Automaten $\mathcal{A}$

Wir zeigen, dass gilt $c = d_1 ; d_2$. Da c der untersuchte von $\mathcal{A}$ akzeptierte Cospan ist, d_1 ein von $\mathcal{A}_1$ akzeptierter Cospan und d_2 ein von $\mathcal{A}_2$ akzeptierter Cospan, folgt daraus, dass $c \in L(\mathcal{A}_1) ; L(\mathcal{A}_2)$. Da das äußere Interface von d_1 gleich dem inneren Interface von d_2 ist, sind d_1 und d_2 konkatenierbar, falls $\overline{U}_1 \cap \overline{U}_2 = K$ gilt.
Wir halten zunächst fest, dass gilt $\overline{U} = \overline{J}$. Das gilt, denn $c \cap id_K = d_1 \cap d_2$. Da K das äußere Interface von d_1 ist, ist $K \subseteq \overline{U}_1$ und da K das innere Interface von d_2 ist, ist $K \subseteq \overline{U}_2$. Also ist $K \subseteq \overline{U}_1 \cap \overline{U}_2 = \overline{J} \cap K$. Damit folgt

$K \subseteq \overline{J}$. Da $\overline{U} = \overline{J} \cup K$ richtig ist, gilt also insgesamt $\overline{U} = \overline{J}$. Wir haben nun vier Relationen zu zeigen:

1. $\overline{U}_1 \cap \overline{U}_2 = K$. Wir wissen bereits, dass $K \subseteq \overline{U}_1 \cap \overline{U}_2$ gilt, wir müssen also noch die umgekehrte Inklusion, $\overline{U}_1 \cap \overline{U}_2 \subseteq K$, zeigen. Es gilt $c \cap id_K = d_1 \cap d_2$, also insbesondere $\overline{U}_1 \cap \overline{U}_2 = \overline{J} \cap K$. Da Gleichheit wechselseitige Inklusion impliziert, gilt also insbesondere auch $\overline{U}_1 \cap \overline{U}_2 \subseteq \overline{J} \cap K \subseteq K$.

2. $\overline{J} = \overline{U}_1 \cup \overline{U}_2$: Wir wissen bereits $\overline{U} = \overline{J}$. Weiterhin gilt $d_1 \cup d_2 = d$, also insbesondere $\overline{U}_1 \cup \overline{U}_2 = \overline{U} = \overline{J}$.

3. $J' \subseteq \overline{U}_2$: Da der rechte Zustand ein Zielzustand ist, ist J' in $\overline{U}_2$ enthalten.

4. $J \subseteq \overline{U}_1$: Da der linke Zustand ein Startzustand ist, ist J in $\overline{U}_1$ enthalten.

Insgesamt haben wir also bewiesen, dass $c = d_1 \,;\, d_2$ gilt und damit ist jeder Cospan, der in der Sprache von $\mathcal{A}$ liegt auch in $L(\mathcal{A}_1) \,;\, L(\mathcal{A}_2)$. □

Folgerung 6.8 *Der Konkatenationsautomat $\mathcal{A}$ erkennt genau die Konkatenation der Sprachen $L(\mathcal{A}_1)$ und $L(\mathcal{A}_2)$.*

Damit haben wir insgesamt folgenden Satz bewiesen:

Satz 6.9 *Die Konkatenation zweier erkennbarer Pfeilsprachen ist eine erkennbare Pfeilsprache.*

Kapitel 7

Abschluss unter Konkatenation für Graphsprachen

Wir wollen nun für den Spezialfall der Graphautomaten über der Menge atomarer Cospans eine Konstruktion angeben.

Definition 7.1 ***(Konkatenationsautomat im Graphenfall)***
Wir konstruieren einen Graphautomaten aus einem Graph-Automaten $\mathcal{A}_1$ für L_1 und einem Graph-Automaten $\mathcal{A}_2$ für L_2 wie folgt:
Wir können davon ausgehen, dass $\mathcal{A}_1$ und $\mathcal{A}_2$ über dem Alphabet

$$\{vertex_i^n, perm_\pi^n, connect_{A,\theta}^n, res_i^n : i < n, Im(\theta) \in \{1, 2, ..., n\}\}$$

definiert sind, also ebenfalls nur Cospans aus der Menge der atomaren Cospans enthält. Zur Definition der Zustände betrachten wir, dass ein Zustand eine zugehörige Interface-Größe $|I|$ hat. Das Start-Interface des Automaten $\mathcal{A}_2$ habe die Größe $|K|$. Schließlich enthält die Vereinigung des Interfaces des momentanen Zustandes und des Start-Interfaces des Automaten $\mathcal{A}_2$ $|U| \in \{\max\{|I|, |K|\}, \max\{|I|, |K|\} + 1, ..., |I| + |K|\}$ viele Knoten. Wir definieren nun die Komponenten des Graph-Automaten $\mathcal{A}$ wie folgt.

- ***Zustände:*** *Die Zustände des Automaten sind von der Form*

  $$(z_1, z_2, s_1, s_2, \varphi_1, \varphi_2)$$

  Zu jedem Zustand gehört implizit eine Menge U, die die Knoten aus dem momentanen Interface vereinigt mit der Menge der Knoten aus dem Start-Interface des Automaten $\mathcal{A}_2$ repräsentiert. Wenn x die Zahl der Start-Interface-Knoten von $\mathcal{A}_2$ ist, die nicht in der Menge der Knoten aus dem momentanen Interface vorkommen, dann gilt $|U| = |I| + x$.
  Weiterhin ist z_1 ein Zustand der Interface-Größe A aus $\mathcal{A}_1$, z_2 ein Zustand der Interface-Größe B aus $\mathcal{A}_2$.

- $s_1 : \{1, 2, ..., A\} \to U$ *zeigt an, wo sich die Knoten die zum Zustand* z_1 *gehören in* U *finden.*
- $s_2 : \{1, 2, ..., B\} \to U$ *zeigt an, wo sich die Knoten die zum Zustand* z_2 *gehören in* U *befinden.*
- s_1 *und* s_2 *sind gemeinsam surjektiv, das bedeutet, dass* $Im(s_1) \cup Im(s_2) = U$ *gilt.*
- $\varphi_1 : \{1, 2, ..., |I|\} \to U$ *zeigt an, wo sich die Knoten aus dem momentanen Interface in* U *finden.*
- $\varphi_2 : \{1, 2, ..., |K|\} \to U$ *zeigt an, wo sich die Knoten aus dem Ziel-Interface des Automaten* $\mathcal{A}_2$ *in* U *befinden.*
- φ_1 *und* φ_2 *sind gemeinsam surjektiv, das bedeutet, dass* $Im(\varphi_1) \cup Im(\varphi_2) = U$ *gilt.*

Schließlich müssen die Zustände die folgende Gleichung erfüllen:

$$K \cap J = U_1 \cap U_2.$$

Zustände, die bis auf Isomorphie der zugehörigen U *identisch sind, fassen wir zu einem Zustand zusammen.*

- ***Startzustände:*** *Startzustände des Automaten* $\mathcal{A}$ *sind all jene Zustände* $(z_1, z_2, s_1, s_2, \varphi_1, \varphi_2)$ *für die gilt:*
  - z_1 *ist ein Startzustand in* $\mathcal{A}_1$
  - z_2 *ist ein Startzustand in* $\mathcal{A}_2$
  - $s_1 = \varphi_1$
  - $s_2 = \varphi_2$
- ***Endzustände:*** *Endzustände des Automaten* $\mathcal{A}$ *sind all jene Zustände* $(z, z_2, s_1, s_2, \varphi_1, \varphi_2)$ *für die gilt:*
  - z_2 *ist ein Endzustand von* $\mathcal{A}_2$.
  - $\varphi_1 = s_2$
  - $\varphi_2 = s_1$
- ***Übergänge:*** *Es gibt einen Übergang*

$$(z_1, z_2, s_1, s_2, \varphi_1, \varphi_2) \xrightarrow{op} (z_1', z_2', s_1', s_2', \varphi_1', \varphi_2'),$$

hierbei sei $U = Im(s_1) \cup Im(s_2)$ *und* $U' = Im(s_1') \cup Im(s_2')$, *genau dann wenn:*

– *$op = perm^n_\pi$: Es gilt $U = U'$. Es gibt eine Permutation π_1 und eine Permutation π_2, so dass*

$$z_1 \xrightarrow{perm_{\pi_1}} z_1'$$

ein Übergang in $\mathcal{A}_1$ ist und

$$z_2 \xrightarrow{perm_{\pi_2}} z_2'$$

ein Übergang in $\mathcal{A}_2$ ist. Weiterhin gilt

$$s_1' = s_1 \circ \pi_1^{-1}$$

$$s_2' = s_2 \circ \pi_2^{-1}$$

$$\varphi_1' = \varphi_1 \circ \pi^{-1}$$

$$\varphi_2' = \varphi_2$$

– *$op = res^n_i$: Es gibt eine Permutation π_1 und eine Permutation π_2, so dass einer der folgenden drei Fälle gilt:*

1. *Es gilt $U \setminus \{\varphi_1(i)\} = U'$.*

$$z_1 \xrightarrow{res_j} \cdot \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ und

$$z_2 \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$. Weiterhin gelte $s_1(j) = \varphi_1(i)$, $\varphi_1(i) \notin Im(s_2) \cup Im(\varphi_2)$. Für s_1', s_2', φ_1' und φ_2' fordern wir schließlich:

$$s_1'(x) = \begin{cases} s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < j \\ s_1(\pi_1^{-1}(x) + 1) & ,falls\ \pi_1^{-1}(x) \geq j \end{cases}$$

$$s_2'(x) = s_2(\pi_2^{-1}(x))$$

$$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < i \\ \varphi_1(x+1) & ,falls\ x \geq i \end{cases}$$

$$\varphi_2'(x) = \varphi_2(x)$$

Dieser Fall beschreibt also das Verschatten eines Knotens, der zum Automaten $\mathcal{A}_1$ gehört und nicht im Endinterface K liegen soll.

2. *Es gilt* $U = U'$.

$$z_1 \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ *und*

$$z_2 \xrightarrow{res_j} \cdot \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$. *Weiterhin gilt* $s_2(\pi_2^{-1}(j)) = \varphi_1(i)$, $\varphi_1(i) \in Im(s_1) \cap Im(\varphi_2)$. *Schließlich fordern wir für* s_1', s_2', φ_1' *und* φ_2', *dass gilt:*

$$s_1'(x) = s_1(\pi_1^{-1}(x))$$

$$s_2'(x) = \begin{cases} s_2(\pi_2^{-1}(x)) & ,falls\ \pi_2^{-1}(x) < j \\ s_2(\pi_2^{-1}(x) + 1) & ,falls\ \pi_2^{-1}(x) \geq j \end{cases}$$

$$\varphi_1'(x) = \varphi_1(x)$$

$$\varphi_2'(x) = \varphi_2(x)$$

Dieser Fall beschreibt also das Verschatten eines Knotens, der zum Automaten $\mathcal{A}_1$ *gehört und aus dem Startinterface* K *stammen soll. In diesem Fall muss* φ_1 *angepasst werden, da der Knoten aus* U *verschwindet.*

3. *Es gilt* $U \setminus \{\varphi_1(i)\} = U'$.

$$z_1 \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ *und*

$$z_2 \xrightarrow{res_j} \cdot \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$. *Weiterhin gilt* $s_2(\pi_2^{-1}(j)) = \varphi_1(i)$, $\varphi_1(i) \notin Im(s_1) \cup Im(\varphi_2)$. *Schließlich fordern wir für* s_1', s_2', φ_1' *und* φ_2', *dass gilt:*

$$s_1'(x) = s_1(\pi_1^{-1}(x))$$

$$s_2'(x) = \begin{cases} s_2(\pi_2^{-1}(x)) & ,falls\ \pi_2^{-1}(x) < j \\ s_2(\pi_1^{-1}(x) + 1) & ,falls\ \pi_2^{-1}(x) \geq j \end{cases}$$

$$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < i \\ \varphi_1(x+1) & ,falls\ x \geq i \end{cases}$$

$$\varphi_2'(x) = \varphi_2(x)$$

Dieser Fall beschreibt also das Verschatten eines Knotens, der zum Automaten $\mathcal{A}_1$ gehört und nicht in dessen Startinterface K liegen soll. Anders als in den beiden vorangegangenen Fällen muss hier φ_1 nicht angepasst werden, denn der verschattete Knoten fällt zwar aus dem momentanen Interface heraus, bleibt aber in U, da er in K liegt und K über alle Zustandsübergänge hinweg gleich bleibt.

- *$op = vertex_i^n$: Es gibt eine Permutation π_1 und eine Permutation π_2, so dass einer der folgenden drei Fälle gilt:*
  1. *Es gilt $U = U'$.*

$$z_1 \xrightarrow{vertex_j} \cdot \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ und

$$z_2 \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$. Weiterhin gibt es einen Index m, für den gilt $\varphi_2(m) \notin Im(s_1)$. Dann fordern wir, dass für s_1', s_2', φ_1' und φ_2' gilt:

$$s_1'(x) = \begin{cases} \varphi_2(m) & ,falls\ \pi_2^{-1}(x) = j \\ s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < j \\ s_1(\pi_1^{-1}(x) - 1) & ,falls\ \pi_1^{-1}(x) > j \end{cases}$$

$$s_2'(x) = s_2(\pi_2^{-1}(x))$$

$$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < i \\ \varphi_1(x-1) & ,falls\ x > i \\ \varphi_2(m) & ,falls\ x = i \end{cases}$$

$$\varphi_2'(x) = \varphi_2(x)$$

Dieser Übergang ermöglicht also das Einfügen eines Knotens (des Knotens $\varphi_2^(m)$) aus dem Zielinterfaces des Automaten $\mathcal{A}_1$ zum Automaten $\mathcal{A}_1$. Da dieser Knoten bereits vorher in K vorhanden sein muss, muss er auch bereits vorher in U liegen, wir müssen also U nicht verändern. Allerdings wird der Knoten erst durch diese Operation in das tatsächliche Interface des Automaten $\mathcal{A}$ aufgenommen, φ_1 muss also angepasst werden.

  2. *Es gilt $U' = U \cup \{v\} \neq U$.*

$$z_1 \xrightarrow{vertex_j} \cdot \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ *und*

$$z_2 \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$*. Dann fordern wir, dass für* s_1'*,* s_2'*,* φ_1' *und* φ_2' *gilt:*

$$s_1'(x) = \begin{cases} v & ,falls\ \pi_2^{-1}(x) = j \\ s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < j \\ s_1(\pi_1^{-1}(x) - 1) & ,falls\ \pi_1^{-1}(x) > j \end{cases}$$

$$s_2'(x) = s_2(\pi_2^{-1}(x))$$

$$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < i \\ \varphi_1(x-1) & ,falls\ x > i \\ v & ,falls\ x = i \end{cases}$$

$$\varphi_2'(x) = \varphi_2(x)$$

Dieser Übergang ermöglicht das Hinzufügen eines Knotens im Automaten $\mathcal{A}_1$*, der nicht zum Zielinterface* K *gehören soll.* s_1 *muss also um den neuen Knoten erweitert werden und auch* φ_1 *muss nun den neuen Knoten referenzieren. Hingegen wird keine Änderung an* K*-Knoten durchgeführt und auch keine Änderung an* $\mathcal{A}_2$*-Knoten durchgeführt, also beliben* φ_2 *und* s_2 *– abgesehen von Umordnungen – unverändert.*

3. *Es gilt* $U' = U \cup \{v\} \neq U$*.*

$$z_1 \xrightarrow{perm_{\pi_1}} z_1'$$

ist ein Übergang in $\mathcal{A}_1$ *und*

$$z_2 \xrightarrow{vertex_j} \cdot \xrightarrow{perm_{\pi_2}} z_2'$$

ist ein Übergang in $\mathcal{A}_2$*. Weiterhin gilt*

$$s_1'(x) = s_1(\pi_1^{-1}(x))$$

$$s_2'(x) = \begin{cases} s_2(\pi_2^{-1}(x)) & ,falls\ \pi_2^{-1}(x) < j \\ s_2(\pi_2^{-1}(x) - 1) & ,falls\ \pi_2^{-1}(x) > j \\ v & ,falls\ \pi_2^{-1}(x) = j \end{cases}$$

$$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < i \\ \varphi_1(x-1) & ,falls\ x > i \\ v & ,falls\ x = i \end{cases}$$

$$\varphi_2'(x) = \varphi_2(x)$$

Durch diese Operation kann ein Knoten im Automaten $\mathcal{A}_2$ hinzugefügt werden. Es kann sich dann nicht um einen Knoten aus K handeln, also bleibt φ_2 unverändert. Hingegen wird der Knoten in das neue Gesamt-Interface von $\mathcal{A}$ und in das Interface des Automaten $\mathcal{A}_2$ aufgenommen, also muss der neue Knoten in φ_1 und s_2 referenziert werden. s_1 bleibt – abgesehen von Umordnungen – unverändert, weil keine Änderung am $\mathcal{A}_1$-Interface durchgeführt wird.

- *$op = connect^n_{A,\theta}$: Es gilt $U = U'$. Wir bezeichnen die Kardinalität einer A-Kante mit $|A|$. Es gibt eine Permutation π_1 und eine Permutation π_2, so dass einer der folgenden zwei Fälle gilt:*

  1. $$z_1 \xrightarrow{connect_{A,\theta'}} \cdot \xrightarrow{perm_{\pi_1}} z_1'$$

     ist ein Übergang in $\mathcal{A}_1$ und

     $$z_2 \xrightarrow{perm_{\pi_2}} z_2'$$

     ist ein Übergang in $\mathcal{A}_2$. Weiterhin gilt für alle $i \in \{1, ..., |A|\}$: $\varphi_1(\theta(i)) = s_1(\theta'(i))$ und wir fordern $s_1' = s_1 \circ \pi_1^{-1}$, $s_2' = s_2 \circ \pi_2^{-1}$, $\varphi_1' = \varphi_1$ und $\varphi_2' = \varphi_2$. Dieser Zustandsübergang ermöglicht das Hinzufügen einer Kante im Automaten $\mathcal{A}_1$, an den Knotenmengen ändert sich hierdurch nichts, s_1, s_2, φ_1 und φ_2 bleiben also – abgesehen von Umordnungen – gleich.

  2. $$z_1 \xrightarrow{perm_{\pi_1}} z_1'$$

     ist ein Übergang in $\mathcal{A}_1$ und

     $$z_2 \xrightarrow{connect_{A,\theta'}} \cdot \xrightarrow{perm_{\pi_2}} z_2'$$

     ist ein Übergang in $\mathcal{A}_2$. Weiterhin gilt für alle $i \in \{1, ..., |A|\}$: $\varphi_1(\theta(i)) = s_2(\theta'(i))$ und wir fordern $s_2' = s_2 \circ \pi_2^{-1}$, $s_1' = s_1 \circ \pi_1^{-1}$, $\varphi_1' = \varphi_1$ und $\varphi_2' = \varphi_2$.
     Dieser Zustandsübergang ermöglicht das Hinzufügen einer Kante im Automaten $\mathcal{A}_2$, an den Knotenmengen ändert sich hierdurch nichts, s_1, s_2, φ_1 und φ_2 bleiben also – abgesehen von Umordnungen – gleich.

Wir haben nun also einen Automaten für die Konkatenation zweier Graphsprachen mit atomaren Cospans beschrieben. Um zu zeigen, dass dieser Automat korrekt ist und das Gewünschte leistet, reicht es, zu zeigen, dass dieser Automat ein Spezialfall des Konkatenationsautomaten aus der im vorherigen Kapitel angegebenen Definition 6.2 ist.

Satz 7.2 *Der in Definition 7.1 beschriebene Automat $\mathcal{A}$ zur Konkatenation zweier Graph-Automaten ist ein Konkatenationsautomat.*

Beweis: Wir setzen $U_1 = \{1, 2, ..., A\}$, $U_2 = \{1, 2, ..., B\}$. $J = \{1, 2, ..., |I|\}$ und $K' = \{1, 2, ..., K\}$, dann lässt sich ein Zustand in $\mathcal{A}$ schreiben als

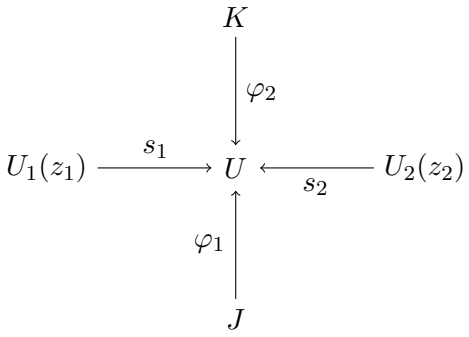


Abbildung 7.1: Zustände im Automaten $\mathcal{A}$

Also genau wie gewünscht.
Für Startzustände fordern wir, dass $s_1 = \varphi_1$ gilt. Das impliziert, dass $U_1 = J$ ist. Weiterhin fordern wir, dass $s_2 = \varphi_2$ ist, woraus wiederum folgt, dass $U_2 = K$ ist. Da wir ebenfalls fordern, dass z_1 ein Startzustand in $\mathcal{A}_1$ ist und z_2 ein Startzustand in $\mathcal{A}_2$, sind die Startzustände also genau die Zustände der Form, die beim Konkatenationsautomat gefordert wird.
Analog beobachten wir, dass die Forderung für die Endzustände, dass $\varphi_1 = s_2$ ist, impliziert, dass $J = U_2$ ist und $\varphi_2 = s_1$ impliziert, dass $U_1 = K$ ist. Also sind auch die Endzustände genau die Zustände der Form, die beim Konkatenationsautomat gefordert wird.
Nun müssen wir noch zeigen, dass die Zustandsübergänge genau diejenigen sind, die beim Konkatenationsautomat angegeben sind. Dies beweisen wir in zwei Schritten:

- Wir zeigen, dass alle für $\mathcal{A}$ angegebenen Übergänge die korrekte Form eines Übergangs in einem Konkatenationsautomaten haben. c ist dann der durch die jeweilige Operation *op* gegebene Zustand und $d = c \cup id_K$. Wir unterscheiden nun alle Fälle für *op* und sehen, dass die erste Bedingung nach Definition immer erfüllt ist, ebenso wie Bedingung vier, definitionsgemäß, da für einen Zustandsübergang verlangt wird, dass die Übergänge auf U_1 und U_2 legale Übergänge in $\mathcal{A}_1$ und $\mathcal{A}_2$ sind.

  - $op = perm^n_\pi$: $d_1 = \pi_1$, $d_2 = \pi_2$. Die fünf Bedingungen werden erfüllt, da $\pi \cup id_K = \pi_1 \cup \pi_2$ nach Definition richtig ist, die K-Monotonie bei Permutationen dadurch erfüllt ist, dass linkes

Interface, Graph und rechtes Interface die gleichen Knoten enthalten und die übrigen Bedingungen schon dadurch garantiert sind, dass wir in einem legalen Zustand starten.

- $op = res_i^n$: Im Fall 1 aus der obigen Definition 7.1 setzen wir $d_1 = res_j ; \pi_1$ und $d_2 = \pi_2$. $res_i^n \cup id_K = res_j ; \pi_1 \cup \pi_2$ gilt ebenfalls. Da im Fall 1 keine K-Knoten verschattet werden, sind Bedingung 2 und die K-Monotonie erfüllt. Im zweiten und dritten Fall ist $d_1 = \pi_1$ und $d_2 = res_j ; \pi_2$. Die dritte Bedingung gilt wieder, weil $res_i^n \cup id_K = res_j ; \pi_2 \cup \pi_1$ gilt, für Bedingungen 5 und 2 betrachten wir getrennt, ob der verschattete Knoten aus K ist oder nicht. Im Fall dessen, dass der verschattete Knoten nicht aus K stammt, wird der Schnitt von U_1 und U_2 sowie von U_1 mit K und U_2 mit K überhaupt nicht berührt, die Bedingungen gelten also. Stammt der Knoten hingegen aus K, so verschwindet der Knoten aus J, also auch aus dem Schnitt $J' \cap K$. Da er auch aus U_2 verschwindet, also auch nicht mehr im Schnitt $U_2' \cap U_1'$ liegt, ist Bedingung 2 weiterhin erfüllt. Die K-Monotonie gilt, denn der Schnitt von U_1 mit K wird nicht angefasst.
- $op = vertex_i^n$: Im ersten und zweiten Fall aus der Definition des *vertex*-Übergangs in $\mathcal{A}$ ist $d_1 = vertex_j ; \pi_1$ und $d_2 = \pi_2$. $vertex_i^n \cup id_K = vertex_j ; \pi_1 \cup \pi_2$ liefert wieder Bedingung 3. Im Fall dessen, dass der neue Knoten kein Knoten aus K ist, sind Bedingungen 2 und 5 (K-Monotonie) klar. Wird hingegen ein Knoten aus K hinzugefügt, so ist dieser Knoten im Vergleich zu $J \cap K$ zusätzlich in $J' \cap K$, allerdings ist er auch zusätzlich in $U_1' \cap U_2'$. Weiterhin ist der Knoten im Graphen $\overline{U}_1$, aber er taucht ebenso in U_1' auf, so dass die K-Monotonie ebenfalls erhalten bleibt.
  Im dritten Fall ist $d_1 = \pi_2$ und $d_2 = vertex_j ; \pi_2$ und Bedingung 3 wieder klar wegen $vertex_i^n \cup id_K = \pi_1 \cup vertex_j ; \pi_2$. In diesem Fall kann kein Knoten aus K hinzugefügt worden sein. Bedingung 2 ist also ebenfalls erfüllt. Ohne Änderungen an dem Schnitt von U_1 und K oder U_2 und K gilt auch die K-Monotonie.
- $op = connect_{A,\theta}^n$: Im ersten Fall setzen wir $d_1 = connect_{A,\theta'} ; \pi_1$ und $d_2 = \pi_2$ und stellen wieder fest, dass Bedingung 3 erfüllt ist wegen $vertex_i^n \cup id_K = connect_{A,\theta'} ; \pi_1 \cup \pi_2$. Es wird kein Knoten zu irgendeiner Menge hinzugefügt oder entfernt, die Bedingungen 2 und 5 gelten also ebenfalls weiterhin. Analog ist im zweiten Fall $d_1 = \pi_1$ und $d_2 = connect_{A,\theta'} ; \pi_2$ und Bedingung 3 erfüllt wegen $vertex_i^n \cup id_K = \pi_1 \cup connect_{A,\theta'} ; \pi_2$. Auch in diesem Fall wird kein Knoten zu einer Menge hinzugefügt oder entfernt, Bedingungen 2 und 5 gelten also wiederum weiter.

- Wir zeigen nun andersherum, dass jeder Übergang, der für einen Konka-

tenationsautomaten erlaubt ist und der atomar ist, auch im Automaten $\mathcal{A}$ vorhanden ist. Wir unterscheiden abermals nach dem Typ der gewählten Operation.

- *perm*-Operation: Wegen $d = c \cup id_K$ muss d eine *perm*-Operation sein und da weiter gilt $d = d_1 \cup d_2$, müssen auch d_1 und d_2 *perm*-Operationen sein.
- *res*-Operation: Falls der verschattete Knoten in $J \cap K$ liegt muss er auch in $U_1 \cap U_2$ liegen. Da er hinterher immernoch in K liegt, muss er entweder in U_1' oder in U_2' liegen. Für den Fall, dass er in U_1' liegt, ist dieser Übergang in $\mathcal{A}$ enthalten, anderenfalls ist aber $U_1' \cap K \neq \overline{U}_1 \cap K$, also ist ein solcher Übergang auch nicht im Konkatenationsautomaten enthalten.
  Ist der Knoten andersherum nicht in $J \cap K$, so ist er entweder in U_1 oder in U_2. Da $c \cup id_K = d_1 \cup d_2$ gelten muss und es entsprechende Übergänge in $\mathcal{A}_1$ und $\mathcal{A}_2$ geben muss, damit der Übergang im Konkatenationsautomaten vorhanden ist, ist der Übergang von $\mathcal{A}$ erfasst.
- *vertex*: Wir unterscheiden wieder, ob der neue Knoten in $J' \cap K$ liegt oder nicht. Liegt der neue Knoten in $J' \cap K$, so liegt der neue Knoten auch in $U_1' \cap U_2'$. Da K sich im Laufe der Operation nicht ändert, war der Knoten bereits in $J \cup K$, also auch in $U_1 \cup U_2$, er lag also vorher schon in U_1 oder in U_2 (aber nicht in beiden, sonst wäre er auch in J). Lag er vorher schon in U_2, aber nicht in U_1, so ist die Operation erfasst. Lag er hingegen in U_1 und nicht in U_2, so würde $U_2 \cap K \neq \overline{U}_2 \cap K$ gelten, die Operation würde also die K-Monotonie nicht erfüllen und ist somit auch keine Operation im Konkatenationsautomaten.
  Ist der neue Knoten umgekehrt nicht in $J' \cap K$, so ist er entweder in U_1' oder in U_2'. Ist er in U_1', so muss d_1 auch eine *vertex*-Operation sein und d_2 eine Permutation, die Operation ist also von $\mathcal{A}$ erfasst. Analog, ist er in U_2', so muss d_2 auch eine *vertex*-Operation sein und d_1 eine Permutation. Auch in diesem Fall ist die Operation von $\mathcal{A}$ erfasst.
- *connect*: Da $c \cap id_K$ ausschließlich Identitäten enthält und ergo keine Kante hinzufügen kann und $d_1 \cap d_2 = c \cap id_K$ richtig ist, muss die Kante entweder komplett im U_1-Graphen oder komplett im U_2 Graphen hinzugefügt werden, die Operation wird also durch $\mathcal{A}$ erfasst.

Insgesamt ist also der oben angegebene Automat $\mathcal{A}$ tatsächlich ein Konkatenationsautomat. □

Um die Konstruktion des Konkatenationsautomaten zu veranschaulichen, wollen wir diese an einem einfachen Graphen-Beispiel einmal durchführen.

Beispiel 7.3 *Beispiel eines Konkatenationsautomaten „Haus des Nikolaus“ Wir werden zunächst einen Automaten $\mathcal{A}_1$ angeben, der drei Knoten erstellt, diese so mit zwei Kanten verbindet, dass ein zusammenhängender Graph entsteht. Der Knoten, der in der Mitte des entstehenden Weges liegt wird verschattet und die beiden äußeren Knoten liegen im Zielinterface des Automaten. Zur Definition unseres Automaten definieren wir allerdings zunächst zwei Funktionen:*

- *Sei $i \in \mathbb{N}$, dann ist die Funktion $p_i : \mathcal{P}(\mathcal{P}(\mathbb{N})) \to \mathcal{P}(\mathcal{P}(\mathbb{N}))$ definiert zu $p_i(X)\{\{a+1 : a \in A\} : A \in X\}$*
- *Sei π eine Permutation, dann ist die Funktion $t_\pi : \mathcal{P}(\mathcal{P}(\mathbb{N})) \to \mathcal{P}(\mathcal{P}(\mathbb{N}))$ definiert zu*
  $t_\pi(X)\{\{\pi(i) : i \in A\} : A \in X\}$

Die Zustände des Automaten sind alle Paare der Form

$$z \in \{0,1,2,3,2'\} \times \{\emptyset, \{\{1,2\}\}, \{\{1,3\}\}, \{\{2,3\}\},$$
$$\{\{1,2\},\{1,3\}\}, \{\{1,2\},\{2,3\}\}, \{\{1,3\},\{2,3\}\}\}$$

Wir merken uns in der ersten Komponente des Zustands, wie viele Knoten bereits erstellt wurden, die Zahlen 0 bis 3 bedeuten einfach, dass 0 bis 3 Knoten bereits eingelesen wurden. $2'$ bedeutet hingegen, dass bereits alle drei Knoten eingelesen wurden und zudem der mittlere Knoten bereits verschattet wurde. In der zweiten Komponente merken wir uns schließlich, welche Paare von Knoten bereits durch Kanten verbunden sind. Der Startzustand ist $(0, \emptyset)$. Als Endzustände wählen wir den Zustand $(2', \emptyset)$.
Als Zustandsübergänge wählen wir alle Tripel aus zwei Zuständen und zugehöriger Operation, die eine der folgenden Bedingungen erfüllt:

- $(i, X) \xrightarrow{vertex^i_j} (i+1, p_j(X))$, *falls* $i \in \{0,1,2\}$
- $(3, X) \xrightarrow{res^3_i} (2', \emptyset)$, *falls* $|\{a \in X : i \in a\}| = 2$
- $(i, X) \xrightarrow{perm^i_\pi} (i, t_\pi(X))$, $i \in \{0,1,2,3\}$
- $(2', \emptyset) \xrightarrow{perm^2_\pi} (2', \emptyset)$
- $(i, X) \xrightarrow{connect^i_{A,\theta}} (i, X \cup \{\theta\})$, $\theta \notin X$

Weiterhin definieren wir einen Automaten $\mathcal{A}_2$, der ausgehend von einem Startinterface, das zwei Knoten enthält, den vollständigen Graphen K_4 bildet. Als Zustände des Automaten $\mathcal{A}_2$ wählen wir alle Paare der Form

$$z \in \{2,3,4\} \times \mathcal{P}(\{\{a,b\} : \{a,b\} \in \{1,2,3,4\} \times \{1,2,3,4\} \wedge a \neq b\})$$

Wir merken uns hierbei wiederum in der ersten Komponente, wie viele Knoten bereits eingelesen wurden und in der zweiten Komponente, welche Knoten aus dem Interface bereits durch eine Kante verbunden werden.
Als Startzustand wählen wir $(2, \emptyset)$.
Als Endzustand wählen wir

$$(4, \{\{a, b\} : \{a, b\} \in \{1, 2, 3, 4\} \times \{1, 2, 3, 4\} \wedge a \neq b\}).$$

Als Zustandsübergänge wählen wir alle Tripel aus zwei Zuständen und zugehöriger Operation, die eine der folgenden Bedingungen erfüllen:

- $(i, X) \xrightarrow{vertex_j^i} (i + 1, p_j(X))$, *mit* $i \leq 3$
- $(i, X) \xrightarrow{perm_\pi^i} (i, t_\pi(X))$
- $(i, X) \xrightarrow{connect_{A,\theta}^i} (i, X \cup \{\theta\})$, *falls* $\theta \notin X$.

Nun können wir entsprechend unserer Konstruktion den Konkatenationsautomaten für diese beiden Automaten angeben. Der Konkatenationsautomat $\mathcal{A}$ *hat alle Zustände der Form*

$$(z_1, z_2, s_1, s_2, \varphi_1, \varphi_2)$$

mit

$$z_1 \in \{0, 1, 2, 3, 2'\} \times \{\emptyset, \{\{1, 2\}\}, \{\{1, 3\}\}, \{\{2, 3\}\}, \{\{1, 2\}, \{1, 3\}\}, \{\{1, 2\}, \{2, 3\}\}, \{\{1, 3\}, \{2, 3\}\}\}$$

und

$$z_2 \in \{2, 3, 4\} \times \mathcal{P}(\{\{a, b\} : \{a, b\} \in \{1, 2, 3, 4\} \times \{1, 2, 3, 4\} \wedge a \neq b\})$$

sowie $\varphi_1 \cup \varphi_2 = s_1 \cup s_2 = U$.
Die Startzustände sind die Zustände der Form

$$((0, \emptyset), (2, \emptyset), \emptyset, id_{\{1,2\}}, \emptyset, id_{\{1,2\}}).$$

Die Endzustände sind von der Form

$$((2', X), (4, \{\{a, b\} \in \{1, 2, 3, 4\} \times \{1, 2, 3, 4\} : a \notin b\}), s_1, s_2, s_2, s_1),$$

wobei $|X| = 2$ *ist.*
Schließlich geben wir die Übergänge an. Die Angabe der Zustandsübergänge erfolgt genau in der Reihenfolge, in der sie in der Definition des Konkatenationsautomaten für den Graphenfall angegeben werden.

- $((i, X), (j, Y), s_1, s_2, \varphi_1, \varphi_2) \xrightarrow{perm_\pi^{|I|}} ((i, t_{\pi_1}(X)), (j, t_{\pi_2}(Y)), s_1 \circ \pi_1^{-1}, s_2 \circ \pi_2^{-1}, \varphi_2 \circ \pi^{-1}, \varphi_2)$

- *Im zweiten Automaten darf überhaupt nicht verschattet werden, es greift also nur Fall 1 der Fallunterscheidung für die res-Operation.*

  $((3,X),(i,Y),s_1,s_2,\varphi_1,\varphi_2) \xrightarrow{res^{|I|}_{\varphi_1^{-1}(s_1(j))}} ((2',t_{\pi_1}(X)),(i,t_{\pi_2}(Y)),s_1',s_2\circ \pi_2^{-1},\varphi_1',\varphi_2)$
  mit

  $|\{a \in X : j \in a\}| = 2,\ \varphi_1' = \begin{cases}\varphi_1(x) & ,falls\ x < j\\ \varphi_1(x+1) & ,falls\ x \geq j\end{cases}$ *und*

  $s_1'(x) = \begin{cases}s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < \varphi_1^{-1}(s_1(j))\\ s_1(\pi_1^{-1}(x)+1) & ,falls\ \pi_1^{-1}(x) \geq \varphi_1^{-1}(s_1(j))\end{cases}$

- *Der erste vertex-Fall beschreibt das Hinzufügen eines K-Interface-Knotens. Hierfür sei m so gewählt, dass $\varphi_2(m) \notin Im(\varphi_1)$*

  $$((a,X),(b,Y),s_1,s_2,\varphi_1,\varphi_2)$$
  $$\xrightarrow{vertex_j^{|I|}} ((a+1,p_b(t_{\pi_1}(X))),(b,t_{\pi_2}(Y)),s_1',s_2\circ\pi_2^{-1},\varphi_1',\varphi_2)$$

  mit $a \in \{0,1,2\}, b \in \{0,...,a+1\}$,

  $$s_1'(x) = \begin{cases}\varphi_2(m) & ,falls\ \pi_2^{-1}(x) = b\\ s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < b\\ s_1(\pi_1^{-1}(x)-1) & ,falls\ \pi_1^{-1}(x) > b\end{cases}$$

  und $\varphi_1'(x) = \begin{cases}\varphi_1(x) & ,falls\ x < j\\ \varphi_1(x-1) & ,falls\ x > j\\ \varphi_2(m) & ,falls\ x = j\end{cases}$

- *Im zweiten vertex-Fall wird ein Knoten v hinzugefügt, der nur zum ersten Automaten, nicht zum zweiten Automaten gehört.*

  $$((i,X),(j,Y),s_1,s_2,\varphi_1,\varphi_2)$$
  $$\xrightarrow{vertex_k^{|I|}} ((i+1,p_a(t_{\pi_1}(X))),(j,t_{\pi_2}(Y)),s_1',s_2,\varphi_1',\varphi_2)$$

  mit $i \in \{0,1,2\},\ a \in \{0,...,i+1\}$,

  $$s_1'(x) = \begin{cases}v & ,falls\ \pi_2^{-1}(x) = a\\ s_1(\pi_1^{-1}(x)) & ,falls\ \pi_1^{-1}(x) < a\\ s_1(\pi_1^{-1}(x)-1) & ,falls\ \pi_1^{-1}(x) > a\end{cases}$$

  und $\varphi_1'(x) = \begin{cases}\varphi_1(x) & ,falls\ x < k\\ \varphi_1(x-1) & ,falls\ x > k\\ v & ,falls\ x = k\end{cases}$

- *Im dritten vertex-Fall wird ein Knoten hinzugefügt, der nur zum zweiten Automaten, nicht zum ersten Automaten gehört.*
  $((i,X),(j,Y),s_1,s_2,\varphi_2,\varphi_2) \xrightarrow{vertex_k^{|I|}} ((i,t_{\pi_1}(X)),(j{+}1,p_a(t_{\pi_2}(Y))),s_1\circ \pi_1^{-1},s_2',\varphi_1',\varphi_2)$ *mit* $j \in \{2,3\}$, $a \in \{0,1,2,...,j+1\}$,
  $$s_2'(x) = \begin{cases} s_2(\pi_2^{-1}(x)) & ,falls\ \pi_2^{-1}(x) < a \\ s_2(\pi_2^{-1}(x)-1) & ,falls\ \pi_2^{-1}(x) > a\ \ und \\ v & ,falls\ \pi_2^{-1}(x) = a \end{cases}$$
  $$\varphi_1'(x) = \begin{cases} \varphi_1(x) & ,falls\ x < k \\ \varphi_1(x-1) & ,falls\ x > k \\ v & ,falls\ x = k \end{cases}$$

- *Der erste connect-Fall ermöglicht das Hinzufügen einer Kante, die zu Automat 1 gehört.*
  $((i,X),(j,Y),s_1,s_2,\varphi_1,\varphi_2) \xrightarrow{connect_{A,\theta}^{|I|}} ((i,t_{\pi_1}(X)\cup\{(s_1\circ\pi_1^{-1})^{-1}(\varphi_1(k)) : k\in\theta\}),(j,t_{\pi_2}(Y)),s_1\circ\pi_1^{-1},s_2\circ\pi_2^{-1},\varphi_1,\varphi_2)$
  mit $\varphi_1(\theta) \subseteq Im(s_1\circ\pi_1^{-1})$ *und* $\{(s_1\circ\pi_1)^{-1}(\varphi_1(k)) : k\in\theta\} \nsubseteq t_{\pi_1}(X)$.

- *Der zweite connect-Fall ermöglicht das Hinzufügen einer Kante, die zu Automat 2 gehört.*
  $((i,X),(j,Y),s_1,s_2,\varphi_1,\varphi_2) \xrightarrow{connect_{A,\theta}^{|I|}} ((i,t_{\pi_1}(X)),(j,t_{\pi_2}(Y)\cup\{(s_2\circ\pi_2^{-1})^{-1}(\varphi_1(k))) : k\in\theta\},s_1\circ\pi_1^{-1},s_2\circ\pi_2^{-1},\varphi_1,\varphi_2)$
  mit $\varphi_1(\theta) \subseteq Im(s_2\circ\pi_2^{-1})$ *und* $\{(s_2\circ\pi_2^{-1})^{-1}(\varphi_1(k)) : k\in\theta\} \nsubseteq t_{\pi_2}(Y)$.

Dieser Automat sollte nun, wenn man sich die Funktionalität der ursprünglichen Automaten in Erinnerung ruft, genau die Graphen akzeptieren, die sich als Haus des Nikolaus zeichnen lassen, also einem Vollständigen Graphen K_4 , der mit einem weiteren Knoten verbunden ist. Dass kein anderer Graph akzeptiert wird, lässt sich durch Abgleichen der Start- und Endzustände einsehen. Wir geben nun einen Pfad durch den Automaten an, der tatsächlich den gewünschten Graphen ergibt. Um die Zustandsfolge leichter verständlich zu machen, sei zuvor aber ein Bild des resultierenden Graphen angegeben. Jeder Knoten ist mit drei Zahlen gelabelt, die erste Zahl ist die Nummer des Knotens im Automaten $\mathcal{A}$, die zweite Nummer die Nummer im Automaten $\mathcal{A}_1$ und die dritte Nummer die Nummer im Automaten $\mathcal{A}_2$. Die Kanten sind in der Reihenfolge nummeriert, in der sie in der Operatorsequenz erzeugt werden.

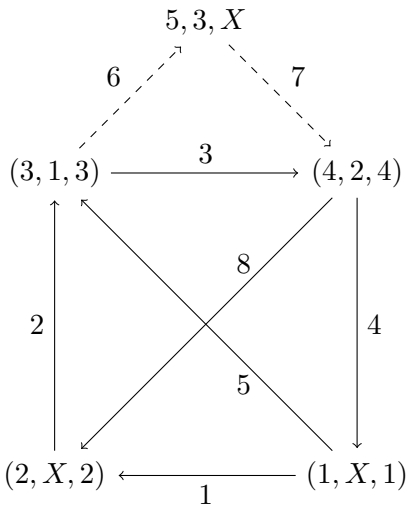


Abbildung 7.2: Haus des Nikolaus

$$((0,\emptyset),(2,\emptyset),\emptyset,id_{\{1,2\}},\emptyset,id_{\{1,2\}})$$

$$\downarrow vertex_1^0$$

$$((0,\emptyset),(3,\emptyset),\emptyset,[1\to 1;2\to 2;3\to 3],[1\to 1],[1\to 2;2\to 3])$$

$$\downarrow vertex_1^1$$

$$((0,\emptyset),(4,\emptyset),\emptyset,[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1,2\to 2],[1\to 3;2\to 4])$$

$$\downarrow connect^2_{[1,2],A}$$

$$((0,\emptyset),(4,\{\{1,2\}\}),\emptyset,[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1,2\to 2],[1\to 3;2\to 4])$$

$$\downarrow vertex_3^2$$

$$((1,\emptyset),(4,\{\{1,2\}\}),[1\to 3],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3],[1\to 3;2\to 4])$$

$$\downarrow connect^3_{[2,3],A}$$

$$((1,\emptyset),(4,\{\{1,2\},\{2,3\}\}),[1\to 3],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3],[1\to 3;2\to 4])$$

$$\downarrow vertex_4^3$$

$$((2,\emptyset),(4,\{\{1,2\},\{2,3\}\}),[1\to 3;2\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 3;2\to 4])$$

$$\downarrow connect^4_{[3,4],A}$$

$$((2,\emptyset),(4,\{\{1,2\},\{2,3\},\{3,4\}\}),[1\to 3;2\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 3;2\to 4])$$

$$\downarrow connect^4_{[4,1],A}$$

$$((2,\emptyset),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\}\}),[1\to 3;2\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 3;2\to 4])$$

$$\downarrow connect^4_{[1,3],A}$$

$$((2,\emptyset),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\}\}),[1\to 3;2\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 3;2\to 4])$$

$$\downarrow vertex^4_5$$

$$((3,\emptyset),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\}\}),[1\to 3;2\to 4;3\to 5],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4;5\to 5],[1\to 3;2\to 4])$$

$$\downarrow connect^5_{[3,5],A}$$

$$((3,\{\{1,3\}\}),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\}\}),[1\to 3;2\to 4;3\to 5],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4;5\to 5],[1\to 3;2\to 4])$$

$$\downarrow connect^5_{[5,4],A}$$

$$((3,\{\{1,3\},\{3,2\}\}),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\}\}),[1\to 3;2\to 4;3\to 5],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4;5\to 5],[1\to 3;2\to 4])$$

$$\downarrow connect^5_{[4,2],A}$$

$$((3,\{\{1,3\},\{3,2\}\}),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\},\{4,2\}\}),[1\to 3;2\to 4;3\to 5],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4;5\to 5],[1\to 3;2\to 4])$$

$$\downarrow res^5_5$$

$$((2',\{\{1,3\},\{3,2\}\}),(4,\{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\},\{4,2\}\}),[1\to 3;2\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 1;2\to 2;3\to 3;4\to 4],[1\to 3;2\to 4])$$

Der letzte Zustand ist ein Endzustand, denn die erste Komponente,

$$(2', \{\{1,3\},\{3,2\}\}),$$

ist ein Endzustand in $\mathcal{A}_1$, die zweite Komponente,

$$(4, \{\{1,2\},\{2,3\},\{3,4\},\{4,1\},\{1,3\},\{4,2\}\}),$$

ist ein Endzustand in $\mathcal{A}_2$, die dritte Komponente ist gleich der sechsten Komponente und die vierte Komponente ist gleich der fünften Komponente. Anhand dieses Ablaufs sehen wir auch, dass der Graph in beliebiger Reihenfolge eingelesen werden kann. Die Kanten werden in einer Reihenfolge eingelesen, in der das Haus des Nikolaus in einem Strich durchgezeichnet werden könnte, das wäre bei einer Hintereinander-Ausführung des ersten und des zweiten Automaten nicht möglich, denn es gibt keine Möglichkeit, das Haus des Nikolaus in einem Strich zu zeichnen und dabei mit dem Dach zu beginnen oder aufzuhören.

Bemerkung 7.4 *Das Ergebnis, dass erkennbare Graphsprachen unter Konkatenation abgeschlossen sind, ist für sich nicht neu, schon im Jahre 1994 hat Bruno Courcelle in [6] die Abschlusseigenschaften erkennbarer Graphsprachen untersucht. Er hat in dieser Arbeit die erkennbaren Graphsprachen mit Hilfe einer äquivalenten Definition (deren Äquivalenz wurde von König et al. in [4] bewiesen) untersucht, ob erkennbare Graphsprachen mit einer ausgezeichneten Teilknotenmenge, die er Quelle genannt hat unter der Operation // abgeschlossen sind. Hat man zwei Graphen G_1 und G_2 gegeben, die eine ausgezeichnete (und gleiche) Quelle C haben, so verklebt der Operator // die beiden Graphen an dem Interface C. Angewandt auf erkennbare Mengen von Graphen ergibt sich so ein Konkatenationsbegriff, der mit unserem weitgehend übereinstimmt. Da die Graphen allerdings nur ein einzelnes Interface besitzen und nach dem Verschmelzen dieses Interface verwendet wurde, ist wiederholte Konkatenation zunächst nur für das immergleiche gemeinsame Interface definiert. Insofern unterscheidet sich die in dieser Arbeit vorgestellte Sicht auf erkennbare Graphsprachen von der von Courcelle verwendeten. Courcelles Beweis konstruiert weiterhin keinen Konkatenationsautomaten, die hier vorgestellte Konstruktion kann allerdings potentiell für die Entwicklung von Algorithmen auf Graph-Automaten verwendet werden, so dass das Ergebnis durchaus Verwendung abseits der bereits bekannten Aussage des Abschlusses unter Konkatenation finden kann. Weiterhin ist das Ergebnis für adhäsive Kategorien allgemeiner als das bereits bekannte Ergebnis für erkennbare Graphsprachen.*

Kapitel 8

Abschluss unter Kanten-Löschung

Eine Abschlusseigenschaft, die sich für reguläre (Wort-) Sprachen sehr einfach beweisen lässt, ist der Abschluss unter Zeichen-Löschung. Für eine gegebene reguläre Sprache L ist die Sprache, die sich ergibt, indem man die Sprache L vereinigt mit allen Worten, die aus Worten aus der Sprache L entstehen, indem man beliebig viele Vorkommen eines fixen Zeichens a entfernt, also wiederum regulär. Der Beweis dieser Eigenschaft ist mit Hilfe eines nichtdeterministischen Automaten relativ einfach. Man fügt einfach einen Epsilon-Übergang zusätzlich für jeden a-Übergang in den Automaten ein und erhält einen NFA, der die gewünschte Sprache akzeptiert.

Wir wollen nun untersuchen, ob eine analoge Aussage auch für erkennbare-Graphsprachen gilt. Ziel ist es also, herauszufinden, ob erkennbare Graphsprachen abgeschlossen sind unter Kanten-Löschung. Das heißt wir wollen herausfinden, ob für eine gegebene erkennbare Graph-Sprache S und eine Regel $T = (L \leftarrow I \rightarrow R)$ der folgenden Form:

- I hat genau so viele Knoten wie L
- L ist zusammenhängend.
- $E(L) = \{k\}$, wir nennen das Label der Kante k *lab*.
- R hat genau so viele Knoten wie I
- $E(R) = \emptyset$

die Sprache $S' = \{G' : \exists G \in S : G \Rightarrow_T^* G'\}$, die entsteht, indem man S mit der Menge vereinigt, die für jeden Graphen, der $n > 0$ Kanten mit Label *lab* enthält, alle Graphen enthält, die für $1 \leq i \leq n$ durch i-fache Anwendung der Regel T entstehen, erkennbar ist.

Wir stellen zunächst einmal fest, dass wir die Konstruktion aus dem Fall der Wortsprachen nicht einfach adaptieren können. Das Problem ist nämlich die beliebige Zerlegbarkeit eines Graphen. Sei beispielsweise die zu löschende Kante eine dreistellige Kante des Labels A. Dann müsste ein Graph, der eine A-Kante enthält, sicher so zerlegt werden, dass sich die drei Knoten, die mit der Kante verbunden werden, in einem Interface befinden - sonst könnte die Kante niemals eingelesen werden. Löscht man die A-Kante, so müssen aber nicht mehr unbedingt irgendwann drei Knoten im Interface liegen. Führt man nun eine Abkürzung im Automaten ein, um das Hinzufügen einer A-Kante zu überspringen, so müsste man beachten, dass gegebenenfalls die Funktoreigenschaft hierdurch zerstört wird. Wie wir im Folgenden sehen werden, ist das Problem allerdings noch grundlegender Natur, die erkennbaren Graphsprachen sind nämlich *nicht* abgeschlossen unter Kanten-Löschung.

Satz 8.1 *Die erkennbaren Graphsprachen sind nicht abgeschlossen unter Kantenlöschung*

Beweis: Wir betrachten die Sprache L_{BC}, die für $i = 1, 2, ...$ jeweils den Graphen G^i_{BC} enthält. Das linke und rechte Interface enthalten jeweils zwei Knoten und besitzen die gleiche injektive Abbildung vom Interface in den Graphen. Der Graph G^i_{BC} hat die Form einer Kette von $2 \cdot i$ vielen Knoten, die linear durch A-Kanten miteinander verbunden sind. Zählen wir die Kette von einem Ende zum anderen Ende ab, so haben alle Knoten mit gerader Nummer eine einstellige B-Kante und alle Knoten mit ungerader Nummer eine einstellige C-Kante, weitere Knoten und Kanten gibt es im Graphen G^i_{BC} nicht. Beispielhaft sind in den folgenden beiden Abbildungen die Graphen G^1_{BC} und G^2_{BC} abgebildet. Da die Richtung der Kanten unerheblich ist, wird sie in den Abbildungen nicht dargestellt.

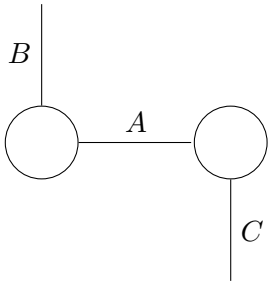


Abbildung 8.1: G^1_{BC}

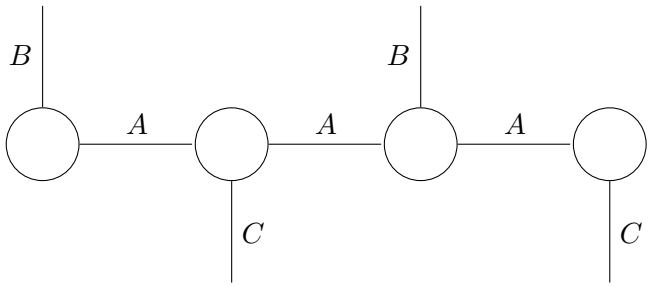


Abbildung 8.2: G^2_{BC}

Die Sprache L_{BC} ist erkennbar, wie wir in Lemma 8.2 zeigen werden. Weiterhin betrachten wir die Graphersetzungsregel T, die es erlaubt, A-Kanten zu löschen. Nehmen wir nun an, dass die Sprache L', die sich durch Kanten-Löschung mit der Regel T aus L_{BC} ergibt, erkennbar wäre, so wäre auch die Sprache L'' erkennbar, die sich aus L' wie folgt ergibt. Wir betrachten die Sprache K aller Graphen die beliebig viele Knoten enthalten und bei denen jeder Knoten adjazent ist zu genau einer Kante des Labels B oder einer Kante des Labels C. Dass diese Sprache erkennbar ist, zeigen wir in Lemma 8.3. Dann ergibt sich L'' als

$$L'' = L' \cap K.$$

Dann müsste also L'' erkennbar sein, denn wie König et al. bereits in [4] bewiesen haben, sind die erkennbaren Graphsprachen abgeschlossen unter Schnitt. Allerdings ist L'' die Sprache aller Graphen, die für ein $n \in \mathbb{N}$ n Knoten enthalten, die adjazant zu genau einer B-Kante sind und n Knoten enthalten, die adjazent zu genau einer C-Kante sind. Dass diese Sprache aber nicht erkennbar ist, zeigen wir in Lemma 8.4. Also ergibt sich ein Widerspruch und die erkennbaren Graphsprachen können somit nicht abgeschlossen sein unter Kanten-Löschung. □

Lemma 8.2 *Die Sprache L_{BC} ist erkennbar.*

Beweisskizze: Wir konstruieren einen Graphautomaten $\mathcal{A}$, der die Sprache L_{BC} akzeptiert. Zu einem Interface I enthalte $\mathcal{A}$ die Zustände

$$(f_B, f_C, f_A, L, R)$$

für die gilt

- $f_B \subseteq I$: Wir merken uns, welche Knoten bereits eine B-Kante tragen.
- $f_C \subseteq I$: Wir merken uns, welche Knoten bereits eine C-Kante tragen.
- $f_B \cap f_C = \emptyset$: Kein Knoten darf B- und C-Kante haben.

- $f_A \subseteq I \times (I \cup \{X_B, X_C, Y_B, Y_C\})$. f_A merkt sich, welche Knoten bereits durch eine A-Kante verbunden sind. Ist ein Knoten mit einem Knoten verbunden, der nicht mehr im Interface ist, wird durch X_B bzw. X_C gesichert, ob dieser Knoten eine B- oder C-Kante hat. Ist ein Knoten mit einem zweiten Knoten verbunden, der nicht mehr im Interface ist, wird durch Y_B bzw. Y_C gesichert, ob dieser Knoten eine B- oder C-Kante hat.

- $\{L, R\} \subseteq I$ mit $L \neq R$: Wir merken uns den linken und rechten Randknoten

Der Startzustand sei der Zustand der Form $I = \{L, R\}$, $f_A = f_B = f_C = \emptyset$. Als Endzustände wählen wir die Zustände der Form $I = \{L, R\}$, $f_B = \{L\}$, $f_C = \{R\}$, $f_A \in \{\{(L, R), (R, L)\}, \{(L, X_C), (R, X_B)\}\}$.
Schließlich erlauben wir einen Übergang von

$$(f_B, f_C, f_A, L, R)$$

nach

$$(f'_B, f'_C, f'_A, L, R)$$

mit den folgenden Operationen:

- $vertex_i^n$: Wenn $f'_B = f_B$, $f'_A = f_A$, $f'_C = f_C$ richtig sind.

- res_i^n: Sei y der i-te Knoten im Interface I, dann erlauben wir einen solchen Übergang, falls $y \neq L$, $y \neq R$, $y \in f_B \cup f_C$ und es ein x und ein z gibt, so dass $x \neq y \neq z$ und $x \neq z$ gelten und $\{(y, x), (y, z)\} \subseteq f_A$ richtig ist. Wir setzen $X := X_B$ und $Y := Y_B$, falls $y \in f_B$, falls hingegen $y \in f_C$, so setzen wir $X := X_C$ und $Y := Y_C$. Dann müssen weiterhin $f'_C = f_B \cap I'$, $f'_C = f_C \cap I'$ und

$$\begin{aligned} f'_A = (f_A \setminus \{(y, a), (a, y) : a \in I \cup \{X_B, Y_B, X_C, Y_C\}\}) \\ \cup\{(a, X) : a \in \{x, z\} \wedge (a, y) \in f_A \wedge a \notin \{X_B, Y_B, X_C, Y_C\}\} \\ \cup\{(a, Y) : a \in \{x, z\} \wedge (a, X) \in f_A \wedge a \notin \{X_B, Y_B, X_C, Y_C\}\} \end{aligned}$$

  richtig sein. Dabei ist anzumerken, dass X markiert, dass der Knoten x respektive z bereits mit einer A-Kante verbunden ist, allerdings ist diese Kante mit einem Knoten, der nicht mehr im Interface liegt, verbunden. Analog markiert Y, dass x respektive z bereits mit zwei A-Kanten verbunden ist, die beide nicht mehr im Interface liegen. Hierdurch stellen wir sicher, dass die Randknoten nur eine und alle übrigen Knoten nur zwei inzidente A-Kanten besitzen. Der Index B beziehungsweise C schließlich zeigt an, ob die verschatteten adjazenten Knoten eine B- oder eine C-Kante haben. Das müssen wir uns merken, um sicherzustellen, dass nicht zwei B- oder C-Kanten aufeinanderfolgen.

- $perm^n_\pi$: $f'_B = f_B$, $f'_C = f_C$, $f'_A = f_A$.
- $connect^n_{A,[i,j]}$: Es sei x der i-te Knoten und y der j-te Knoten in I, dann ist dieser Übergang erlaubt, wenn gilt: $x \neq y$, $\{x,y\} \not\subseteq f_B$, $\{x,y\} \not\subseteq f_C$, $(x,y) \notin f_A$ - hierdurch stellen wir sicher dass keine zwei Knoten miteinander verbunden werden, die beide eine Kante des Labels B oder beide eine Kante des Labels C haben. Weiterhin fordern wir, um sicherzustellen, dass die Knoten L und R niemals adjazent zu mehr als einem anderen Knoten sind und die übrigen Knoten niemals adjazent zu mehr als zwei anderen Knoten sind:

$$|f_A \cap \{(x,z) : z \in I \cup \{X_B, X_C, Y_B, Y_C\}\}| \leq \begin{cases} 1 & , falls x \notin \{L,R\} \\ 0 & , falls x \in \{L,R\} \end{cases}$$

$$|f_A \cap \{(y,z) : z \in I \cup \{X_B, X_C, Y_B, Y_C\}\}| \leq \begin{cases} 1 & , falls y \notin \{L,R\} \\ 0 & , falls y \in \{L,R\} \end{cases}$$

  Dann ist $f'_B = f_B$, $f'_C = f_C$, $f'_A = f_A \cup \{(x,y),(y,x)\}$
- $connect^n_{B,[i]}$: Es sei x der i-te Knoten im Interface I, dann ist dieser Übergang erlaubt, wenn $x \neq R$ ist und für alle y mit $(x,y) \in f_A$ gilt $y \notin f_B \cup \{X_B, Y_B\}$. Dann ist $f'_B = f_B \cup \{x\}$, $f'_C = f_C$, $f'_A = f_A$.
- $connect^n_{C,[i]}$: Es sei x der i-te Knoten im Interface I, dann ist dieser Übergang erlaubt, wenn $x \neq L$ ist und für alle y mit $(x,y) \in f_A$ gilt $y \notin f_C \cup \{X_C, Y_C\}$. Dann ist $f'_C = f_C \cup \{x\}$, $f'_B = f_B$, $f'_A = f_A$.

Der dargestellte Automat $\mathcal{A}$ akzeptiert tatsächlich jeden Graphen der Sprache L_{BC}, sei nämlich ein beliebiger Graph G^i_{BC}, $i > 1$, gegeben, so ist

$$\begin{aligned} &connect^2_{B,[1]} \,;\, connect^2_{C,[2]} \,;\, vertex^2_3 \,;\, connect^3_{C,[3]} \,;\, vertex^3_4 \,;\, connect^4_{B,[4]} \,; \\ &connect^4_{A,[1,3]} \,;\, connect^4_{A,[3,4]} \,;\, res^4_3 \,;(vertex^3_4 \,;\, connect^4_{C,[4]} \,;\, vertex^4_5 \,; \\ &connect^5_{B,[5]} \,;\, connect^5_{A,[3,4]} \,;\, connect^5_{A,[4,5]} \,;\, res^5_3 \,;\, res^4_3 \,;)^{i-2} connect^3_{A,[2,3]} \,;\, res^3_3 \end{aligned}$$

eine Zerlegung des zugehörigen Cospans in Operatoren. Abgleichen mit den Zustandsübergängen des Automaten ergibt, dass diese akzeptiert wird. Für $i = 1$ nimmt man statt obiger Operatorfolge die Folge

$$connect^2_{B,[1]} \,;\, connect^2_{C,[2]} \,;\, connect^2_{A,[1,2]}$$

Andersherum ist auch jeder von $\mathcal{A}$ akzeptierte zusammenhängende Graph ein G^i_{BC}, denn wenn keine neuen Knoten hinzugefügt werden, kann nur akzeptiert werden, wenn $f_A = \{(L,R),(R,L)\}$, $f_B = \{L\}$, $f_C = \{R\}$ ist, was nach Abgleich der erlaubten Übergänge von einem Startzustand aus nur erreicht werden kann, wenn der G^1_{BC} eingelesen wird. Werden andererseits in einer

akzeptierten Operatorsequenz neue Knoten erzeugt, so müssen diese wieder verschattet werden. Ein Knoten kann nur verschattet werden, wenn er genau zu zwei anderen Knoten per A-Kante verbunden ist und entweder selbst mit einer B-Kante verbunden ist und die beiden adjazenten Knoten mit einer C-Kante verbunden sind oder aber selbst mit einer C-Kante verbunden ist und die beiden adjazenzten Knoten mit einer B-Kante verbunden sind. Also handelt es sich auch dann um einen Graphen der Form G^i_{BC}. $L(\mathcal{A})$ - eingeschränkt auf die zusammenhängenden Graphen, vergleiche hierzu Beispiel 3.13 - ist also genau die Sprache L_{BC}. Dass es sich auch tatsächlich um einen Graphautomaten handelt sieht man ein, indem man beachtet, dass eine Entsprechung zwischen den Zuständen und dem bislang eingelesenen Graphen existiert. Wenn man einen Graphen auf zwei unterschiedliche Weisen zerlegt und einliest, muss man im gleichen Zustand auskommen, denn wäre das nicht der Fall, so gäbe es zwei Zustände (f_A, f_B, f_C, L, R) und $(f'_A, f'_B f'_C, L, R)$ die nicht gleich sind, aber durch unterschiedliche Zerlegungen des gleichen Cospans erreicht werden. Wäre nun $f_C \neq f'_C$, so wäre (beachte, wie f_C verändert wird, nämlich nur durch Hinzufügen einer Kante und durch Entfernen eines Knoten) in f_C ein Knoten der nicht in f'_C liegt - also ein Knoten im ersten Graph mit einer C-Kante versehen, der es im zweiten Graph nicht ist oder andersherum, beides wäre ein Widerspruch. Analog muss auch $f_B = f'_B$ sein. Wäre nun $f_A \neq f'_A$, so muss einer der folgenden Fälle eintreffen:

- f_A enthält ein Paar aus $I \times I$, das in f'_A nicht liegt. Dann müssten aber (beachte, dass in f_A nur durch Hinzufügen einer Kante und Löschen eines Knotens Änderungen vorgenommen werden) zwei Knoten im ersten Graphen durch eine A-Kante verknüpft sein, die es im zweiten Graphen nicht sind (analog für vertauschte Rollen von f_A und f'_A).
- f_A enthält ein Paar aus $I \times \{X_B, X_C\}$, das f'_A nicht enthält, dann ist im ersten Graphen ein Knoten durch eine A-Kante zu einem Knoten verbunden, der nicht mehr im Interface liegt und eine B-Kante (respektive C-Kante) hat, im zweiten Graphen jedoch nicht (analog für vertauschte Rollen von f_A und f'_A).
- f_A enthält ein Paar aus $I \times \{Y_B, Y_C\}$, das f'_A nicht enthält, dann ist im ersten Graphen ein Knoten durch eine A-Kante zu zwei Knoten verbunden, die nicht mehr im Interface liegen und eine B-Kante (respektive C-Kante) haben, im zweiten Graphen jedoch nicht (analog für vertauschte Rollen von f_A und f'_A).

In jedem Fall ergibt sich ein Widerspruch dazu, dass die Operatorfolgen den gleichen Cospan bilden, also handelt es sich bei $\mathcal{A}$ tatsächlich um einen Graphautomaten. □

Lemma 8.3 *Die Sprache L aller Graphen, die beliebig viele Knoten enthalten und bei denen jeder Knoten adjazent ist zu genau einer Kante des Labels B oder einer Kante des Labels C, ist erkennbar.*

Beweisskizze: Wir konstruieren abermals einen Graphautomaten $\mathcal{A}$, der die Sprache L akzeptiert. Die Konstruktion ist ähnlich der, die wir für das Lemma 8.2 durchgeführt haben. Zu einem Interface I gibt es die Zustände (f_B, f_C) wobei $f_B \subseteq I$, $f_C \subseteq I$ und $f_B \cap f_C = \emptyset$ gilt. Als Startzustand wählen wir den Zustand mit $|I| = 2$ und $f_B = f_C = \emptyset$, als Endzustand die Zustände mit $|I| = 2$ und $f_B \cup f_C = I$. Wir ermöglichen schließlich einen Übergang von (f_B, f_C) nach (f'_B, f'_C) mit den folgenden Operationen:

- $vertex^n_i$: $f'_B = f_B$ und $f'_C = f_C$.
- res^n_i: Es sei x der i-te Knoten in I, dann ist dieser Übergang erlaubt, falls $x \in f_C \cup f_B$, sowie $f'_B = f_B \setminus \{x\}$ und $f'_C = f_C \setminus \{x\}$.
- $perm^n_\pi$: $f'_B = f_B$ und $f'_C = f_C$.
- $connect^n_{B,[i]}$: Es sei x der i-te Knoten im Interface I, dann ist dieser Übergang erlaubt, falls $x \notin f_B \cup f_C$ ist und $f'_B = f_B \cup \{x\}$ sowie $f'_C = f_C$ richtig ist.
- $connect^n_{C,[i]}$: Es sei x der i-te Knoten im Interface I, dann ist dieser Übergang erlaubt, falls $x \notin f_B \cup f_C$ ist und $f'_C = f_C \cup \{x\}$ sowie $f'_B = f_B$ richtig ist.

□

Lemma 8.4 *Die Sprache L aller Graphen, die für ein $n \in \mathbb{N}$ n Knoten enthalten, die adjazent zu genau einer B-Kante sind und n Knoten enthalten, die adjazent zu genau einer C-Kante sind, ist nicht erkennbar.*

Beweis: Angenommen es gäbe einen Graphautomaten $\mathcal{A}$ über der atomaren Menge von Cospans, der L akzeptiert. Dann muss es für $i = 1, 2, ...$ jeweils paarweise verschiedene Zustände geben, deren Interface-Größe Null ist, denn für $i = 1, 2, ...$ ist

$$\underbrace{vertex^2_1 ; connect^3_{B,[1]} ; res^3_1}_{i-Mal}$$

ein gültiges Präfix einer Zerlegung eines Graphen in L in atomare Cospans. Würden nun für ein $i \neq j$ zwei Präfixe dieser Form der Länge i bzw. j in den gleichen Zustand führen, so müsste

$$\underbrace{vertex^2_1 ; connect^3_{C,[1]} ; res^3_1}_{i-Mal} ; connect^2_{C,[1]} ; connect^2_{B,[2]}$$

von diesem Zustand aus in einen Endzustand führen. Da es aber egal ist, ob man dieses Suffix nun an das eine oder das andere Präfix anhängt und man in beiden Fällen in den gleichen Endzustand geraten kann, würde also ein Graph akzeptiert werden, der nicht in L liegt. Also muss es unendlich

viele Zustände zum Interface der Größe Null geben, dann ist $\mathcal{A}$ aber nicht lokal endlich und somit kein Graphautomat. Das steht im Widerspruch zur Annahme, dass $\mathcal{A}$ ein Graphautomat sei und somit haben wir die Aussage bewiesen. □

Kapitel 9

Abschluss unter Substitution

Wir werden in diesem Kapitel sehen, dass die erkennbaren Graphsprachen nicht unter (Kanten-)Substitution abgeschlossen sind. Das ist insofern ein interessantes Resultat, als dass reguläre Sprachen unter Substitution (eines Alphabetssysmbols) abgeschlossen sind. Wir wollen zunächst definieren, was Substitution genau bedeutet.

Definition 9.1 ***((Kanten-)Substitution)***
Gegeben sei eine Menge von Graphen L, ein Kantenlabel A der Stelligkeit k und ein fester Graph G in dem k Knoten als Verschmelzknoten ausgezeichnet sind. Dann bezeichnen wir als die Menge L', die durch Substitution von Kanten des Labels A durch den Graphen G entsteht die Menge, die alle Graphen enthält, die wie folgt aus einem Graphen G' aus L entstehen:

- *Entferne alle A-Kanten aus G'*
- *Füge an Stelle jeder A-Kante den Graphen G ein. Die Verschmelzknoten von G werden hierbei identifiziert mit den Knoten, die mit der A-Kante verbunden waren.*

Satz 9.2 *Die erkennbaren Graphsprachen sind nicht abgeschlossen unter Substitution.*

Beweis: Wir zeigen dies durch Angabe eines Gegenbeispiels. Es sei zunächst L die Sprache, die genau die Graphen akzeptiert, die einen Knoten v besitzen und beliebig viele einstellige A-Kanten besitzt (die dann offensichtlich adjazent sein müssen zu v). Als Substitutionsgraph verwenden wir den Graphen G, der aus genau einem Knoten v besteht, der mit genau einer einstelligen B-Kante und einer einstelligen C-Kante verbunden ist.
Wir müssen nun zeigen, dass L erkennbar ist. Dies tun wir durch Angabe eines Graph-Automaten unter Verwendung der atomaren Cospan-Menge.

Als Zustände verwenden wir die Menge $\{z_0, z_1\}$, z_0 ist der (einzige) Startzustand und z_1 der (einzige) Endzustand. Als Übergänge erlauben wir einen

Übergang mit $vertex_1^0$ von z_0 nach z_1 sowie mit $connect^1_{A,[1]}$ von z_1 nach z_1. Der so angegebene Automat ist offensichtlich ein Graphautomat und erkennt die Sprache L.

Wir betrachten nun die Menge L' die entsteht, indem wir auf L die Substitution aller A-Kanten durch G anwenden. Es entsteht die Menge, die genau die Graphen enthält, die gleich viele B- wie C-Kanten enthält und nur einen Knoten. Wollten wir nun einen Graph-Automaten für L' angeben, so müsste dieser die Graphen in beliebiger Zerlegung akzeptieren. Insbesondere muss ein solcher Automat Graphen in der Zerlegung akzeptieren, in der zunächst alle B-Kanten und dann alle C-Kanten erzeugt werden. Das bedeutet aber, dass es unendlich viele Äquivalenzklassen für Graphen mit genau einem Knoten im Interface gibt, nämlich für $i = 0, 1, 2, ...$ die Äquivalenzklasse „i B-Kanten mehr als C-Kanten“ und für $j = 1, 2, ...$ die Äquivalenzklasse „j C-Kanten mehr als B-Kanten“. Also kann es keinen Graphautomaten geben, der L' akzeptiert, L' ist also nicht erkennbar. Insgesamt ist die Klasse der erkennbaren Graphsprachen nicht abgeschlossen unter Substitution. □

Wir haben im vorangegangenen Kapitel bereits gezeigt, dass die erkennbaren Graphsprachen abgeschlossen sind unter Konkatenation. Durch wiederholtes Anwenden der Konstruktion ist offensichtlich, dass sie auch unter wiederholter Konkatenation abgeschlossen sind. Mit einem sehr ähnlichen Gegenbeispiel wie für die Substitution können wir aber noch zeigen, dass erkennbare Graphsprachen nicht unter dem Kleene-Stern abgeschlossen sind. Die Definition des Kleene-Sterns ist allerdings nicht eindeutig für erkennbare Graphsprachen (vgl. [6]), daher definieren wir zunächst, was wir unter einem Kleene-Stern verstehen wollen.

Definition 9.3 *Gegeben sei eine erkennbare K, K-Graphsprache L und ein Automat $\mathcal{A}$ der L erkennt. Dann bezeichnen wir als L^* die folgende Sprache:*

$$L^* = \{G : \exists n \in \mathbb{N}(K \to G \leftarrow K = \sigma_1 ; ... ; \sigma_n \wedge \forall 1 \leq i \leq n(\sigma_i \in L(\mathcal{A})))\}$$

L^* ist also die Sprache aller Graphen, deren Cospan sich so zerlegen lässt, dass jeder Teilcospan von $\mathcal{A}$ akzeptiert wird.

Satz 9.4 *Die erkennbaren Graphsprachen sind nicht abgeschlossen unter dem Kleene-Stern, also unter der beliebig häufigen Konkatenation.*

Beweis: Wir betrachten die Graphsprache L, die als Startinterface einen einzelnen Knoten hat und genau den Graphen akzeptiert, der einen Knoten hat und jeweils eine damit verbundene einstellige A-Kante und B-Kante. Der akzeptierte Graph wird in der Abbildung 9.1 dargestellt.

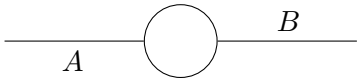


Abbildung 9.1: Graph in der Sprache L

Dass diese Sprache erkennbar ist, lässt sich durch Angabe eines einfachen Automaten über der atomaren Cospan-Menge zeigen. Der Automat hat vier Zustände z_0, z_a, z_b und z_e. z_0 ist der einzige Startzustand, z_e der einzige Endzustand. Es gibt (abgesehen von den Identitätspfeilen für die einstellige Permutation) folgende Übergänge:

- $z_0 \xrightarrow{connect^1_{A,[1]}} z_a$
- $z_0 \xrightarrow{connect^1_{B,[1]}} z_b$
- $z_b \xrightarrow{connect^1_{A,[1]}} z_e$
- $z_a \xrightarrow{connect^1_{B,[1]}} z_e$

Simples Abgleichen der Übergänge zeigt, dass der Automat tatsächlich die gewünschte Sprache akzeptiert. Wendet man nun auf diese Sprache den Kleene-Stern an, erhält man die Sprache aller Graphen, die einen Knoten enthalten und genau gleich viele A- wie B-Kanten. Dass diese Sprache nicht erkennbar ist, haben wir aber bereits im vorangegangenen Beweis zu Satz 9.2 gezeigt. Also sind die erkennbaren Graphsprachen nicht abgeschlossen unter dem Kleene-Stern. □

Teil III

Terminierungsanalyse

In diesem Teil der Arbeit werden wir einen Ansatz zur Terminierungsanalyse für Stringersetzungssysteme kennen lernen und untersuchen, inwiefern sich dieser Ansatz auf Graphtransformationssysteme übertragen lässt. Wir werden im ersten Kapitel dieses Abschnitts zunächst löschende und Matchbeschränkte Stringersetzungssysteme definieren und wichtige Eigenschaften dieser Systeme aufzeigen. Die wichtigste Eigenschaft der löschenden Stringersetzungssysteme für die Terminierungsanalyse ist, dass sie Regularität erhalten. Der Beweis dieser Eigenschaft wurde in [12] konstruktiv erbracht. Diese Konstruktion wurde anschließend genutzt um einen Algorithmus zur Terminierungsanalyse für Stringersetzungssysteme zu entwerfen.

Im zweiten Kapitel dieses Teils wollen wir hieran anknüpfend untersuchen, ob die vorgestellte Konstruktion sich auf Graphtransformationssysteme übertragen lässt. Aus dem vorangegangenen Teil wissen wir bereits, dass die erkennbaren Graphsprachen nicht unter Substitution abgeschlossen sind, für einen Beweis, dass sie unter der Anwendung löschender Graphtransformationssysteme abgeschlossen sind, müssten wir also eine andere Konstruktion wählen, als im Fall der Stringersetzungssysteme. Wir werden allerdings anhand eines Gegenbeispiels sehen, dass die Suche nach einem anderen Beweis hierfür nicht zielführend sein kann, denn die erkennbaren Graphsprachen sind nicht abgeschlossen unter der Anwendung löschender Graphtransformationssysteme.

Kapitel 10

Löschende und Match-beschränkte Stringersetzungssysteme

Wir wollen in diesem Kapitel ein Verfahren zur Untersuchung von Stringersetzungssysteme auf Terminierung kennen lernen, das von Alfons Geser, Dieter Hofbauer und Johannes Waldmann in [8] entwickelt wurde. Ziel ist es, für ein gegebenes Stringersetzungssystem festzustellen, ob es auf jeder Zeichenkette einer gegebenen regulären Sprachen terminiert. Ein Stringersetzungssystem terminiert auf einer Zeichenkette, wenn nur endlich viele Regelanwendungen auf der Zeichenkette möglich sind. Zunächst werden wir löschende und Match-beschränkte Stringersetzungssysteme defininieren, anschließend stellen wir die Konstruktion vor, mit der Alfons Geser, Dieter Hofbauer und Johannes Waldmann in [8] bewiesen haben, dass Match-beschränkte Stringersetzungssysteme Regularität erhalten und schließlich können wir auf dieser Grundlage ein Verfahren angeben, um zu überprüfen, ob ein Stringersetzungssystem auf einer gegebenen regulären Sprache terminiert.
Das Verfahren überprüft im Wesentlichen, ob ein gegebenes Stringersetzungssystem S Match-beschränkt ist. Dafür konstruiert man für $c = 0, 1, ...$ einen nichtdeterministischen endlichen Automaten für das Match-System $Match_c(S)$. Findet man ein c für das $Match_{c+1}(S) = Match_c(S)$ gilt, so ist S Match-bechränkt durch S und terminiert somit, da Match-beschränkte Systeme löschend sind und löschende Systeme terminierend sind. Die Aussagen in diesem Kapitel werden wir nicht beweisen, sie sind allerdings in [12], respektive [8] bewiesen. Wir definieren zunächst, was wir unter einem löschenden Stringersetzungssystem verstehen. Die folgenden Definitionen und Sätze sind [12] entnommen.

Definition 10.1 *Ein Stringersetzungssystem S über einem Alphabet Σ nennen wir $>$-löschend, falls folgendes gilt:*

- *$>$ ist eine strenge Ordnung auf Σ, das heißt $>$ ist transitiv und für $a, b \in \Sigma$ gilt $a > b \implies b \not> a$*
- *Für jede Regel $l \to r$ ist die Bedingung $l > r$ richtig.*

Dabei erweitern wir die Relation $>$ von einzelnen Zeichen auf Zeichenketten wie folgt. Es sei L die Multimenge der in l enthaltenen Zeichen und R die Multimenge der in r enthaltenen Zeichen, dann gelte die Äquivalenz:

$$l > r \Leftrightarrow L \neq R \wedge \forall k \in R \setminus L \exists k' \in L \setminus R : k' > k$$

Dabei sei $\setminus$ die Multimengen-Differenz und es gilt $a \in L$ genau dann wenn es mindestens ein a in der Multimenge L gibt.
Wir nennen ein Stringersetzungssystem S schließlich einfach löschend*, falls es eine Ordnung $>$ gibt, so dass S $>$-löschend ist.*

Satz 10.2 *Die löschenden Stringersetzungssysteme haben linear beschränkte Berechnungskomplexität. Das bedeutet, dass die Zahl der möglichen Ableitungen eines löschenden Stringersetzungssystems auf einem gegebenen Wort mit n Zeichen beschränkt ist durch $\mathcal{O}(n)$. Insbesondere sind also alle löschenden Stringersetzungssysteme terminierend auf allen Mengen von Wörtern, die über ihrem Alphabet definiert sind.*

Wenn wir also einen Algorithmus angeben, der für ein gegebenes Stringersetzungssystem überprüft, ob es löschend ist, so erhalten wir hiermit auch einen Algorithmus mit einseitigem Fehler zur Überprüfung, ob das Stringersetzungssystem terminierend ist. Wir werden im Folgenden die wesentlichen Konstruktionsschritte, die zum Beweis führen, dass löschende Stringersetzungssysteme die Regularität erhalten, vorstellen.

Bemerkung 10.3 *Erweiterung des Alphabets*
Wir werden im Folgenden oftmals ein Alphabet Δ um neue Buchstaben erweitern müssen. Wir gehen hierbei von einem fixen endlichen Grundalphabet Σ aus und definieren das unendliche Alphabet Σ' wie folgt. Es gilt $\Sigma' = \Sigma \times \mathbb{N}^*$, ein Zeichen $x' \in \Sigma'$ hat also die Form $x' = (x, [n_1, ..., n_k])$ mit $x \in \Sigma$ und $n_j \in \mathbb{N}$ für $j = 1, ..., k$. Wir identifizieren weiterhin $(x, [])$ mit x und bezeichnen für ein Zeichen $(x, [n_1, ..., n_k]) = x' \in \Sigma'$ mit x'_j das Zeichen $(x, [n_1, ..., n_k, j])$. Wenn wir also ein endliches Alphabet $\Delta \supseteq \Sigma$ erweitern, indem wir sagen „Es sei $a \in \Delta$ und a_1 ein neues Zeichen, das nicht in Δ enthalten ist“, dann bezeichnen wir mit a_1 ein neues Zeichen aus Σ', das nicht in Δ enthalten ist.
Eine partielle Ordnung $>$ auf Σ erweitern wir zu einer partiellen Ordnung auf Σ' als

$$(x, \overline{m}) > (y, \overline{n}) \Leftrightarrow x > y \vee (x = y \wedge \overline{m} \sqsubset \overline{n})$$

Dabei bedeutet $\overline{m} \sqsubset \overline{n}$, dass $\overline{m}$ ein echtes Präfix von $\overline{n}$ ist.

Wir werden in den folgenden beiden Lemmata die zwei wesentlichen Umformungsschritte kennen lernen, die von der Konstruktion zum Beweis des Erhalts der Regularität durch löschende Stringersetzungssysteme genutzt werden.

Lemma 10.4 *Es sei R ein Stringersetzungssystem über dem Alphabet Σ und $a \in \Sigma$, dann gilt*

$$R^* = \overline{R}_a^* \circ R_a^*$$

mit

$$R_a = \{l \to r : (l \to r) \in R, l = a\},$$
$$\overline{R}_a = \{l \to r' : (l \to r) \in R \setminus R_a, r' \in R_a^*(r)\}$$

Wenn a nicht auf der rechten Seite irgendeiner Regel in R_a vorkommt, dann ist R_a eine endliche Substitution und $\overline{R}_a$ ein endliches Stringersetzungssystem.

Lemma 10.5 ***(Splitting)***
Es sei R ein Stringersetzungssystem über dem Alphabet Σ und $(x_a x_2 \to y_1 z y_2) \in R$ mit $a \in \Sigma$ und $x_i, y_i, z \in \Sigma^*$. Weiterhin seien a_1 und a_2 zwei neue Zeichen, die nicht in Σ vorkommen, dann bilden wir das Stringersetzungssystem R' aus R, indem wir die obige Regel entfernen und durch die folgende Menge an Regeln ersetzen:

$$\{a \to a_1 z a_2, x_1 a_1 \to y_1, a_2 x_2 \to y_2\}$$

Dann ist $R^* = R'^*|_\Sigma$. Weiterhin, falls R $>$-löschend ist und $a > d$ für alle $d \in \Sigma(y_1 z y_2)$ richtig ist, ist R' löschend.

Wir merken noch an, dass wir davon ausgehen können, dass jedes untersuchte Stringersetzungssystem die Eigenschaft hat, dass jedes Alphabetssymbol in jeder Regel höchstens ein Mal auf der linken Seite vorkommt. Kommt ein Zeichen a zwei Mal in einer Regel auf der linken Seite vor, so können wir das erste Vorkommen von a durch einen neuen Buchstaben a_i ersetzen und die Regel $a \to a_i$ hinzufügen. Wenn wir diese Überlegung iteriert anwenden, erhalten wir ein Stringersetzungssystem in der gewünschten Form.

Wir verwenden nun die oben angegebenen Konstruktionen, um zu zeigen, dass sich jedes löschende Stringersetzungssystem R schreiben lässt als eine Folge von Substitutionen, gefolgt von invers kontextfreien Regeln und schließlich einer Restriktion auf das ursprüngliche Alphabet. Da reguläre Sprachen abgeschlossen sind unter Substitution, der Anwendung invers kontextfreier Regeln und der Restriktion, folgt damit, dass löschende Stringersetzungssysteme Regularität erhalten.

Satz 10.6 *löschende Stringersetzungssysteme erhalten Regularität, das heißt, wenden wir ein löschendes Stringersetzungssystem auf eine reguläre Sprache an, so erhalten wir wieder eine reguläre Sprache.*

Wir nehmen an, dass auf jeder linken Seite einer Regel jedes Zeichen maximal ein Mal vorkommt und konstruieren eine Folge $(S_i, M_i, N_i)_{i\in\mathbb{N}}$, so dass für alle i gilt: S_i ist eine endliche Substitution, M_i ist ein invers kontextfreies Stringersetzungssystem (also die rechte Seite einer Regel enthält genau ein Zeichen) und N_i enthält nur Regeln, die nicht invers kontextfrei sind. Für alle i soll dann gelten

$$R^* = ((M_i \cup N_i)^* \circ (S_i \circ ... \circ S_1))|_\Sigma.$$

Wir starten mit S_0 als Identität, M_0, der Menge der invers kontextfreien Regeln in R und N_0, der Menge der nicht invers kontextfreien Regeln in R. Wir iterieren nun über i, bis wir ein i erreicht haben, für das $N_i = \emptyset$ gilt, denn dann haben wir eine Zerlegung des Stringersetzungssystems R gefunden, die nur aus Substitutionen, gefolgt von invers kontextfreien Regeln und einer Restriktion besteht. Gelangen wir zu einem i, bei dem diese Bedingung noch nicht erfüllt ist, erhalten wir $(S_{i+1}, M_{i+1}, N_{i+1})$ aus (S_1, M_i, N_i) wie folgt:

- Wir wählen eine beliebige Regel $(l \to r) \in N_i$ und bestimmen das gemäß der Ordnung $>$ größte Zeichen a in l. Wir nennen $l \to r$ die Pivotregel und a das Pivotzeichen.

- $l \to r$ ist nicht invers kontextfrei, die rechte Seite hat also mindestens zwei Zeichen. Wir können also r schreiben als $b_1 z b_2$, wobei b_1 und b_2 jeweils ein Zeichen sind und z eine (möglicherweise leere) Zeichenkette. Weiterhin können wir die linke Seite der Regel schreiben als $l = l_1 a l_2$ wobei l_1 und l_2 (möglicherweise leere) Zeichenketten sind und a das Pivotzeichen. Wir wenden nun Lemma 10.5 an, um die Regel $l \to r$ zu zerlegen in die drei Regeln $\{a \to a_1 z a_2, l_1 a_1 \to b_1, a_2 l_2 \to b_2\}$. Dabei sind a_1 und a_2 frische Buchstaben. Wir erhalten nun N_i' aus N_i indem wir die Regel $l \to r$ durch die obigen drei Regeln ersetzen.

- Es sei nun $R_i' = M_i \cup N_i'$ und wir bestimmen gemäß des Lemmas 10.4 $(R_i')_a$ und $\overline{(R_i')_a}$. $(R_i')_a$ ist eine Substitution, die wir als S_{i+1} wählen. M_{i+1} definieren wir zu den invers kontextfreien Regeln von $\overline{(R_i')_a}$ und N_{i+1} zu den nicht invers kontextfreien Regeln von $\overline{(R_i')_a}$.

Im Folgenden werden wir einsehen, wieso die gerade beschriebene Konstruktion zur Terminierungsanalyse verwendet werden kann. Der Rest dieses Kapitels folgt nun den Darstellungen aus [8]. Wie im Fall der löschenden Stringersetzungssysteme werden wir die Beweise der Aussagen hier nicht besprechen, die Beweise finden sich allerdings in [8]. Wir beginnen damit, die Klasse der Match-beschränkten Stringersetzungssysteme zu definieren.

Hierzu führen wir zunächst Funktionen ein, die dazu dienen, sich für einen Buchstaben in einem String zu merken, wie viele Regelanwendungen benötigt wurden, um diesen Buchstaben zu generieren.

Definition 10.7 ***(Match-System)***
Gegeben sei ein Alphabet Σ. Wir definieren die Morphismen $lift_c$, base und height:

$$lift_c : \Sigma^* \to (\Sigma \times \mathbb{N})^* : a \mapsto (a, c)$$

$$base : (\Sigma \times \mathbb{N})^* \to \Sigma^* : (a, c) \mapsto a$$

$$height : (\Sigma \times \mathbb{N})^* \to \mathbb{N}^* : (a, c) \mapsto c$$

Auf dieser Basis können wir nun zu einem gegebenen Stringersetzungssystem R das zugehörige Match-System $match(R)$ definieren zu

$$match(R) = \{l' \to lift_c(r) : (l \to r) \in R, base(l') = l, c = 1 + min(height(l'))\}$$

Hiermit können wir nun definieren, was wir unter einem Match-beschränkten Stringersetzungssystem verstehen.

Definition 10.8 *Ein Stringersetzungssystem R über dem Alphabet Σ nennen wir Match-beschränkt für eine Sprache $L \subseteq \Sigma^*$ durch $c \in \mathbb{N}$, falls gilt:*

- $c \notin lhs(R)$
- $max(height(x)) \leq c$ *für alle* $x \in match(R)^*(lift_0(L))$

Dabei ist $lhs(R)$ die Menge aller linken Seiten von Regeln in R.

Die Klasse der Match-beschränkten Stringersetzungssysteme ist nun auf Grund zweier Eigenschaften für uns besonders interessant.

Satz 10.9 *Es gelten die folgenden beiden Aussagen:*

1. *Für alle Stringersetzungssysteme R mit $\epsilon \notin lhs(R)$ und alle $c \in \mathbb{N}$ ist das System $match_c(R)$ löschend.*
2. *Zusammen mit dem Ergebnis aus Satz 10.2, dass löschende Stringersetzungssysteme terminierend sind, folgt, dass falls R Match-beschränkt für L ist, R auch terminierend für L ist.*

Analog erhalten wir zudem:

Satz 10.10 *Ist R löschend, so ist R Match-beschränkt.*

Folgerung 10.11 *Falls R Match-beschränkt für eine reguläre Sprache L ist, so ist auch $R^*(L)$ regulär*

Auf Grund dieser Nähe der Begriffe der Match-Beschränktheit und der löschenden Stringersetzungssysteme, können wir nun aus der Konstruktion zu Satz 10.6 ein Verfahren entwerfen, zu entscheiden, ob ein Stringersetzungssystem Match-beschränkt durch eine Konstante ist. Da wir wissen, dass aus der Match-Beschränktheit Terminierung folgt, haben wir somit ein Verfahren mit einseitigem Fehler gefunden, um festzustellen, ob ein gegebenes Stringersetzungssystem auf einer gegebenen regulären Sprache terminiert.

Satz 10.12 *Gegeben seien ein Stringersetzungssystem R, eine reguläre Sprache L und ein $c \in \mathbb{N}$. Dann ist die Frage, ob R für L durch c Match-beschränkt ist entscheidbar*

Beweis: Wir konstruieren einen endlichen Automaten für die Sprache

$$L_{c+1} = match_{c+1}(R)^*(lift_0(L))$$

mittels des Verfahrens aus dem Satz 10.6. Dann ist R Match-beschränkt durch c für L, genau dann wenn $max(height(L_{c+1})) \leq c$ richtig ist. Ob diese Bedingung erfüllt ist, kann man im Automaten ablesen, indem man überprüft, ob es einen akzeptierenden Pfad im Automaten gibt, auf dem ein Zustandsübergang mit einem Alphabetssysmbol der Höhe $c + 1$ liegt. □
Wendet man dieses Verfahren für $c = 1, 2, ...$ an, so erhält man ein Semi-Entscheidungsverfahren für das Problem, ob R auf L terminierend ist. Auf Grund der Unentscheidbarkeit dieses Problems gibt es kein Entscheidungsverfahren, so dass dieses Ergebnis bereits ein gutes Ergebnis ist.

Kapitel 11

Löschende und Match-beschränkte Graphtransformationssysteme

Wir haben im vergangenen Kapitel gesehen, wie die Begriffe der löschenden und Match-beschränkten Stringersetzungssysteme zu einem guten Algorithmus geführt haben, um Terminierung nachzuweisen. Wir wollen in diesem Kapitel untersuchen, ob sich dieser Ansatz für den Graphen-Kontext adaptieren lässt. Dazu müssen wir die Begriffe der löschenden und Match-beschränkten Systeme natürlich auf den Graphenfall verallgemeinern. Wir werden sehen, dass sich eine Reihe von Eigenschaften mit strukturell sehr ähnlichen Beweisen wie im Stringersetzungsfall beweisen lassen, allerdings werden wir auch sehen, wieso dieser Ansatz leider nicht zu einem Ergebnis vergleichbarer Güte führen kann, wie im Stringersetzungsfall. Die folgenden Definitionen, Sätze und Beweise orientieren sich strukturell stark an den in [12] vorgestellten Ergebnissen im Fall der Stringersetzung.

Definition 11.1 *Ein Graphtransformationssystem (GTS) S über einer Kantenlabelmenge Λ ist $>$-löschend für eine strenge Ordnung $>$ auf Λ, falls für jede Regel $L \leftarrow I \rightarrow R \in S$ gilt $L > R$, wobei $>$ wie folgt auf Graphen L und R verallgemeiner wird:*
Es bezeichne l die Multimenge, die die Label der Kanten in L (mit ihrer Vielfachheit gezählt) enthält und r die Multimenge, die die Label der Kanten in R (mit ihrer Vielfachheit gezählt) enthält. Dann gilt

$$L > R \Leftrightarrow l \neq r \wedge \forall k \in r \setminus l \exists k' \in l \setminus r : k' > k$$

Dabei ist $\setminus$ die Multimengen-Differenz.
Wir nennen ein GTS S schließlich einfach löschend, wenn es eine strenge Ordnung $>$ gibt, so dass S $>$-löschend ist.

Satz 11.2 *löschende Graphtransformationssysteme sind linear in ihrer Berechnungskomplexität beschränkt, das heißt, dass ausgehend von einem Graphen mit n Kanten höchstens $\mathcal{O}(n)$ viele Regelanwendungen möglich sind.*

Korollar 11.3 *Aus obigem Satz folgt direkt, dass löschende Graphtransformationssysteme terminierend sind, es also keine unendliche Folge von Regelanwendungen auf einen gegebenen Graphen geben kann.*

Beweis: Es sei S ein löschendes GTS. Dann gibt es eine strenge Ordnung $>$, so dass R $>$-löschend ist. Im Folgenden sei nun $g : \mathbf{HGraph} \rightarrow \mathbf{MSet}$ die Funktion, die einen Hypergraphen auf die Multimenge seiner Kantenlabels abbildet. Nach Definition eines $>$-löschenden GTS muss für jede Regel $L \rightarrow R \in S$ gelten, dass $g(L) > g(R)$. Seien also G_i, G_{i+1} Hypergraphen mit $G_i \rightarrow_S G_{i+1}$, so muss auch $g(G_i) > g(G_{i+1})$ gelten. Weiterhin muss $|g(G_{i+1}) \setminus g(G_i)| \leq m$ gelten, mit $m = \max\{|h(R)| : L \leftarrow I \rightarrow R \in S\}$. Wir bezeichnen für ein Kantenlabel k mit $h(k)$ die Höhe dieses Kantenlabels, also die Zahl der Kantenlabels, die kleiner als k sind:

$$h(k) = |\{k' \in \Lambda : k > k'\}|$$

Es sei nun G ein beliebiger Graph mit $g(G) = \{k_1, k_2, ..., k_n\}$. Wir wollen die Zahl der möglichen Transformationsschritte für G abschätzen. Hierzu nehmen wir an, dass $h(k_1) = ... = h(k_i) = ... = h(k_n)$ richtig ist.

Anderenfalls, falls also $h(k_1) = ... = h(k_i) = ... = h(k_n)$ nicht gilt, überschätzen wir die Zahl der möglichen Transformationsschritte, indem wir den Wert von $h(k_i)$, $1 \leq i \leq n$, auf $k := \max\{h(k_i) : 1 \leq i \leq n\}$ setzen, indem wir Λ um Hilfslabel erweitern wie folgt.
Sei i so gewählt, dass $h(k_i) < k$, dann fügen wir $k - h(k_i)$ Hilfslabel k_{i_1}, ... $k_{i_{k-h(k_i)}}$ ein, erweitern $>$ um die Beziehungen $k_{i_{k-h(k_i)}} > ... > k_{i_1} > k_i$ und fügen zu S die Transformationsregeln

$$G(k_{i_{k-h(k_i)}}) \leftarrow I \rightarrow G(k_i)$$

hinzu, für die $G(k_{i_{k-h(k_i)}})$ ein zusammenhängender Graph ist, der genau eine Kante des Labels $k_{i_{k-h(k_i)}}$ enthält und $G(k_i)$ aus $G(k_{i_{k-h(k_i)}})$ entsteht, indem man die Kante mit dem Label k_i versieht. I bezeichnet den zugehörigen Interface-Graphen, einen diskreten Graphen, der genauso viele Knoten enthält wie $G(k_i)$. Schließlich ersetzen wir in G k_i durch $k_{i_{k-h(k_i)}}$. Führen wir dies für alle $i \in \{1, 2, ..., n\}$ durch, so erhalten wir einen Graphen der gewünschten Form, für den nach Anwendung der neu hinzugefügten Regeln an jeder möglichen Stelle genau die Transformationen möglich sind, die auch für den ursprünglichen Graphen möglich waren. Wir haben also sichergestellt, dass wir die Zahl der maximal möglichen Transformationen auf diese Weise nicht gesenkt haben.

Es gelte nun also $g(G) = \{k_1, k_2, ..., k_n\}$ und $h(k_1) = ... = h(k_i) = ... = h(k_n)$. Für jedes Kantenlabel in $g(G)$ lässt sich die Zahl der möglichen Transformationen maximieren, indem wir annehmen, dass wir stets nur Transformationen der Art durchführen, dass die linke Seite genau eine Kante k_i aus G enthält und die rechte Seite m neue Kanten mit sich bringt (mehr sind nicht möglich, denn wir haben m so gesetzt, dass es die maximale Anzahl an Kanten auf der rechten Seite einer Regel ist), die alle die höchstmögliche Höhe haben, es gilt also für jede neue Kante k_i': $h(k_i') = h(k_i) - 1$. Diese Abschätzung liefert tatsächlich die maximale Zahl der Regelanwendungen, denn umfasst die linke Seite einer angewandten Regel mehr als eine Kante, so kann die Gesamtzahl an Kanten nach der Regelanwendung nur um weniger als $m-1$ größer sein als vor der Regelanwendung, wir verlieren also potentielle Regelanwendungen. Insgesamt sind je Kante k_i also maximal $1+m+m^2+...+m^{h(k_i)} = \sum_{j=0}^{h(k_i)} m^j$ Regelanwendungen möglich. Wir zeigen nun unter (B), dass $\sum_{j=0}^{h(k_i)} m^j \leq (m+1)^{h(k_i)}$ richtig ist, dann gilt nämlich mit $c = (m+1)^k$, wobei $k := h(k_1) = ... = h(k_n)$ gesetzt ist, dass die Zahl der anwendbaren Transformationen beschränkt ist durch $\sum_{i=1}^{n} c = cn \in \mathcal{O}(n)$. Diese Summe entspricht der Summe aller möglicher Regelanwendungen ausgehend von den Labels von k_1, ..., k_n.
Zunächst benötigen wir eine Hilfsaussage:
(A) Es gilt für alle $x \geq 0$ und $n \in \mathbb{N}_0$ die Ungleichung $(x+1)^{n+1} \geq x^{n+1} + (x+1)^n$. Diese zeigen wir induktiv:
Induktionsanfang $(n = 0)$: $(x+1)^{0+1} = x + 1 \geq x^{0+1} + (x+1)^0$
Induktionsvoraussetzung: Die Aussage gilt für alle Werte bis zu einem gewissen $n \in \mathbb{N}_0$.
Induktionsschritt $(n \to n+1)$:

$$\begin{aligned}(x+1)^{(n+1)+1} &= (x+1)(x+1)^{n+1} \geq_{(IV)} (x+1)(x^{n+1} + (x+1)^n) \\ &= x^{n+2} + (x+1)^{n+1} + x^{n+1} \geq x^{n+2} + (x+1)^{n+1}\end{aligned}$$

Damit beweisen wir nun die Aussage (B), wiederum induktiv:
(B) Es gilt für alle $x \geq 0$ und $n \in \mathbb{N}_0$ die Ungleichung $\sum_{k=0}^{n} x^k \leq (x+1)^n$.
Induktionsanfang $(n = 0)$: $\sum_{k=0}^{0} x^k = 1 = (x+1)^0$
Induktionsvoraussetzung: Die Aussage gilt für alle Werte bis zu einem gewissen $n \in \mathbb{N}_0$.
Induktionsschritt $(n \to n+1)$:

$$\sum_{k=0}^{n+1} x^k = x^{n+1} + \sum_{k=0}^{n} x^k \leq_{(IV)} x^{n+1} + (x+1)^n \leq_{(A)} (x+1)^{n+1}$$

□

Im Folgenden wollen wir die Labelmenge, über der ein Graphtransformationssystem definiert ist, erweitern. Sei Λ die Labelmenge, die wir erweitern wollen. Wir wollen zu einem Label $a \in \Lambda$ neue Label a_i, a_{ij}, ... einführen. Hierzu

betrachten wir das unendliche Alphabet $\Lambda' = \Lambda \times \mathbb{N}^*$, dann hat jedes Label in Λ' die Form $x' = (x, [n_1, ..., n_k])$ mit $x \in \Lambda$ und $n_j \in \mathbb{N}$ für $j = 1, ..., k$. Auch wenn wir an dieser Stelle ein unendliches Alphabet definieren, werden wir später sehen, dass wir nur eine endliche Teilmenge dieses Alphabets benötigen werden, um unsere Konstruktionen ausgehend von einem endlichen Alphabet umzusetzen. Wir bezeichnen weiterhin mit der Basis eines Labels $x' = (x, [n_1, ..., n_k]) \in \Lambda'$ das zugrundeliegende Label x, in Zeichen $base((x, [n_1, ..., n_k])) = x$. Weiterhin schreiben wir für $x' = (x, [n_1, ..., n_k])$ die Variable $(x, [n_1, ..., n_k, j])$ auch als x'_j. Für den Fall, dass $k = 0$ ist, also die Sequenz der natürlichen Zahlen leer ist, identifizieren wir $x' = (x, [])$ mit x.
Wir erweitern nun eine partielle Ordnung $>$ auf Λ zu einer partiellen Ordnung auf Λ' wie folgt:

$$(x, [m_1, m_2, ..., m_k]) > (x, [n_1, n_2, ..., n_l]) \Leftrightarrow x > y \vee (x = y \wedge \bar{l} > k \wedge \bigwedge_{1 \leq i \leq k} (n_i = m_i))$$

Es ist also $(x, m) > (y, n)$ genau dann, wenn $x > y$ ist, oder aber wenn $x = y$ ist und die Sequenz m ein (echtes) Präfix der Sequenz n ist.

Wir fahren nun fort mit der Definition der Match-beschränkten Graphtransformationssysteme. Die folgenden Definitionen, Sätze und Beweise orientieren sich stark am analogen Fall für Stringersetzungssysteme, wie sie in [8] untersucht wurden.

Definition 11.4 ***(Match-GTS)***
Es sei S ein GTS über der Labelmenge Λ. Wir definieren das zu S gehörige Match-GTS $Match(S)$ wie folgt.

1. *Wir definieren für $c \in \mathbb{N}_0$ die Abbildung $lift_c$: **HGraph** $\rightarrow$ **HGraph** gemäß*

   $$lift_c((V, E, src, tgt, lab)) = (V, E, src, tgt, \{(x, (lab, c)) : (x, lab) \in lab\})$$

2. *Wir definieren weiterhin die Abbildung $base$: **HGraph** $\rightarrow$ **HGraph** gemäß*

   $$base((V, E, src, tgt, lab)) = (V, E, src, tgt, \pi_1 \circ lab)$$

   Hierbei ist π_1 die Projektionsfunktion, die das erste Element eines Tupels zurück gibt.

3. *Schließlich definieren wir die Abbildung $height$: **HGraph** $\rightarrow \mathbb{N}_0$ zu*

   $$height(V, E, src, tgt, lab) = \{n : n \in \pi_2(\pi_2(x)), x \in lab\}$$

4. *Das Match-GTS zu S wird nun über der – zunächst unendlichen – Labelmenge $\Lambda \times \mathbb{N}_0$ definiert zu*

$$match(S) = \{L' \leftarrow I \rightarrow lift_c(R) : (L \leftarrow I \rightarrow R) \in S, base(L') = L, c = 1 + \min(height(L'))\}$$

Dabei ist min *die Minimumsfunktion.*

Zwar betrachten wir im allgemeinen Fall eine unendliche Labelmenge, wir können uns aber für den Fall eines Match-beschränkten Graphtransformationssystems auf eine endliche Labelmenge zurückziehen.

Definition 11.5 ***(Match-beschränkte GTS)***
Es sei S ein Graphtransformationssystem über der endlichen Labelmenge Λ. Wir nennen S Match-beschränkt durch $c \in \mathbb{N}$ für eine Graphsprache L, falls jede linke Seite einer Transformationsregel in S mindestens eine Kante enthält und folgendes gilt:

$$\max(height(x)) \leq c \forall x \in match(S)^*(lift_0(L))$$

Wir nennen S einfach Match-beschränkt (durch c), wenn S für die Sprache aller Graphen über der Labelmenge Λ Match-beschränkt (durch c) ist.

Wissen wir, dass ein System S Match-beschränkt durch c ist, so können wir die Labelmenge des Match-GTS einschränken auf $\Lambda \times \{0, 1, ..., c\}$ und so wieder eine endliche Labelmenge erhalten. Wir nennen das auf das Alphabet $\Lambda \times \{0, 1, ..., c\}$ eingeschränkte Match GTS zu S $Match_c(S)$. Wir verwenden diese Einschränkung auf ein Alphabet auch dann, wenn S nicht Match-beschränkt ist oder wir jedenfalls nicht wissen, ob S Match-beschränkt ist. Allerdings ist dann nicht sichergestellt, dass für $n > c$ gilt

$$S^n|_L = (base \circ match_c(S)^n \circ lift_0)_L$$

ist S hingegen Match-beschränkt bezüglich L mit c, so gilt diese Gleichung für alle $n \in \mathbb{N}_0$.

Lemma 11.6 *Für alle Graphtransformationssysteme S über der Labelmenge Λ, die keine Regel mit linker Seite ohne Kante enthalten, ist $match_c(S)$ für alle $c \in \mathbb{N}_0$ löschend.*

Beweis: Wir definieren die Ordnung $>$ auf $\Lambda \times \{0, 1, ..., c\}$ gemäß $(a, m) > (b, n) \Leftrightarrow m < n$. Wir zeigen nun, dass $match_c(S)$ $>$-löschend ist. Hierzu betrachten wir eine beliebige Regel $(L \leftarrow I \rightarrow R) \in match_c S$ und zeigen, dass $L > R$ richtig ist. Zunächst einmal ist klar, dass die Multimenge der Kantenlabel von L und R nicht identisch sein kann, auf Grund der Definition des Match-Graphen, denn die geringste Höhe eines Kantenlabels auf der linken Seite ist genau um 1 geringer als die Höhe eines jeden Kantenlabels

auf der rechten Seite. Wir betrachten nun also eine Kante $k \in R$, die ein Label hat, das keine Kante in L hat, gibt es eine solche Kante nicht, ist die Bedingung für ein $>$-löschendes GTS trivial erfüllt. Wir suchen nun eine Kante k' in L, die ein Label hat, das bezüglich der Ordnung $>$ ein größeres Label hat, als k. Das ist nach Definition von $>$ äquivalent dazu, eine Kante k' in L zu suchen, die eine kleinere Höhe hat, als k. Hierzu wählen wir k' als eine Kante mit der minimalen Höhe in L, nach Definition hat jede Kante in R eine um 1 größere Höhe als k', also hat auch k eine größere Höhe als k'.□
Da wir bereits bewiesen haben, dass löschende Graphtransformationssysteme terminierend sind, folgt direkt:

Korollar 11.7 *Für alle Graphtransformationssysteme S über der Labelmenge Λ, die keine Regel mit linker Seite ohne Kante enthalten, ist $match_c(S)$ für alle $c \in \mathbb{N}_0$ terminierend.*

Schließlich können wir auch beweisen, dass Match-beschränkte GTS terminierend sind.

Satz 11.8 *Ist S ein für L durch c Match-beschränktes GTS, so ist S terminierend auf L*

Beweis: Wir zeigen dies durch Widerspruch. Angenommen, es gäbe eine unendliche S-Ableitung, die bei einem Element von L beginnt, so kann diese Ableitung in eine unendliche $match(S)$-Ableitung umgeformt werden, die bei einem Element von $lift_0(L)$ beginnt. Da aber S für L durch c Match-beschränkt ist, muss diese Ableitung auch eine $match_c(S)$-Ableitung sein. Das steht aber im Widerspruch zum gerade bewiesenen Lemma, das besagt, dass $match_c(S)$ terminiernd ist. □

Korollar 11.9 *Jedes Match-beschränkte GTS ist linear in seiner Berechnungskomplexität beschränkt*

Das lässt sich genauso wie obiger Satz beweisen, mit der zusätzlichen Anmerkung, dass wir nicht nur gezeigt haben, dass löschende GTS terminierend sind, sondern sogar, dass sie linear beschränkte Berechnungskomplexität besitzen.
Nachdem wir bereits eingesehen haben, dass Match-beschränkte GTS löschend sind, wollen wir noch zeigen, dass auch jedes löschende GTS Match-beschränkt ist.

Satz 11.10 *Ist S ein löschendes GTS über Λ, so ist S auch Match-beschränkt*

Beweis: Es sei $>$ so gewählt, dass S $>$-löschend ist. Sei nun k die Länge der größten $>$-Kette in Λ, dann ist S Match-beschränkt durch k. □
Wir sehen also, dass ganz analog zum Stringersetzungs-Fall Match-beschränkte Graphtransformationssystem terminierend sind. Gelänge es also, in Analogie zum String-Fall ein Semientscheidungs-Verfahren zu entwickeln, das

untersucht, ob ein gegebenes Graphtransformationssystem Match-beschränkt ist, so erhielten wir gleichermaßen ein Verfahren, um ein Graphtransformationssystem auf Terminierung zu untersuchen. Ziel wäre es nun, die Konstruktion auf den Graphenfall zu adaptieren und zu zeigen, dass Match-beschränkte Graphtransformationssysteme Erkennbarkeit erhalten. Genau die gleiche Konstruktion können wir hierfür nicht verwenden, denn wir haben bereits gezeigt, dass die erkennbaren Graphsprachen nicht unter Substitution abgeschlossen sind. Wir werden nun allerdings leider auch noch einsehen, dass erkennbare Graphsprachen grundsätzlich nicht unter der Anwendung von Match-beschränkten Graphtransformationssystemen abgeschlossen sind.

Satz 11.11 *Die erkennbaren Graphsprachen sind nicht unter der Anwendung Match-beschränkter Graphtransformationssysteme abgeschlossen*

Beweis: Wir verwenden das Beispiel aus Satz 9.2 und zeigen nur noch, dass das Graphtransformationssystem, das nur aus der Regel besteht, die eine A-Kante durch eine B- und eine C-Kante ersetzt, löschend ist. Wir betrachten die Ordnung $A > B > C$, dann ist die einzige Regel des Graphtransformationssystems löschend, denn $A > B$ und $A > C$ ist richtig und auf der linken Seite befindet sich eine A-Kante, auf der rechten Seite nur B- und C-Kanten. Also ist das Graphtransformationssystem tatsächlich löschend. Wie in Satz 9.2 sehen wir aber ein, dass die Sprache aller Graphen mit einem Knoten und beliebig vielen A-Kanten erkennbar ist, wohingegen die Sprache aller Graphen mit einem Knoten und beliebig vielen A-Kanten und gleich vielen B- wie C-Kanten nicht erkennbar ist. □

Kapitel 12

Fazit und Ausblick

In dieser Arbeit haben wir eine Reihe von Abschlusseigenschaften für erkennbare Graphsprachen nachweisen können, andere aber auch widerlegen können. Bereits im Vorfeld war bekannt, dass die erkennbaren Graphsprachen abgeschlossen sind unter Vereinigung, Schnitt und Komplement. Diese Eigenschaften wurden bereits in [4] nachgewiesen. In dieser Arbeit haben wir zusätzlich bewiesen, dass die erkennbaren Graphsprachen abgeschlossen sind unter Konkatenation, sowie unter der Anwendung invers kontextfreier Graphersetzungsregeln. Negative Ergebnisse haben wir für die Substitution, die Anwendung von Regeln, die Kanten löschen und den Kleene-Stern erhalten. Erkennbare Graphsprachen sind also nicht abgeschlossen unter Substitution, löschende Graphersetzungsregeln und der Anwendung des Kleene-Sterns.

Der Beweis des Abschlusses unter Konkatenation erfolgte im allgemeineren Setting der adhäsiven Kategorien und durch den Nachweis im Spezialfall der Graphsprachen, dass der Abschluss unter Anwendung des Kleene-Sterns nicht gegeben ist, wissen wir dies natürlich auch im Allgemeinen für adhäsive Katgorien. Wir haben also für erkennbare Pfeilsprachen auf adhäsiven Kategorien ebenfalls Einblicke in die Abschlusseigenschaften gewonnen.

Wir haben weiterhin gesehen, dass eine Möglichkeit zur Terminierungsanalyse für Graphtransformationssysteme darin besteht, zu überprüfen, ob das Graphtransformationssystem ein Match-beschränktes System ist. Auf dieser Basis wurde in vergangenen Arbeiten bereits ein in praktischen Tests erfolgreiches System zur Terminierungs-Prüfung von Stringersetzungssystemen ausgearbeitet. Die wichtigste Grundlage für diesen Algorithmus ist ein Satz, der den Erhalt der Regularität einer Sprache betrifft. Leider mussten wir bereits feststellen, dass die erkennbaren Graphsprachen, anders als die regulären Sprachen, nicht unter Substitution abgeschlossen sind. Dies war allerdings eine wichtige Voraussetzung für die Anwendbarkeit des Algorithmus' im Stringersetzungsfall, so dass gewiss eine andere Zerlegung gewählt

werden müsste. Leider hat sich zudem herausgestellt, dass löschende Graphtransformationssysteme die Regularität nicht erhalten. Damit ist allerdings eine Adaption des Algorithmus vom String-Fall auf das Graphen-Setting unmöglich. Nichtsdestotrotz haben wir gesehen, dass der Nachweis, dass ein Graphtransformationssystem löschend oder Match-beschränkt ist, generell zur Terminierungsanalyse verwendet werden kann. Ein anderer Ansatz, diese Eigenschaften nachzuweisen könnte also zu einem guten Verfahren zur Terminierungsanalyse führen.

Doch auch unabhängig von dieser Betrachtung können auf dieser Arbeit aufbauend weitere Erkenntnisse gesammelt werden. Die Abschlusseigenschaften, die in dieser Arbeit bewiesen wurden, können auch weiteren Untersuchungen der erkennbaren Graphsprachen dienlich sein. Neben den bereits bekannten und den in dieser Arbeit bewiesenen Abschlusseigenschaften für Graphsprachen bietet sich zudem an, weitere Operationen zu untersuchen und herauszufinden, ob erkennbare Graphsprachen unter Anwendung dieser Operationen abgeschlossen sind oder nicht.

Für die Terminierungsanalyse für Graphtransformationssysteme kann man durchaus versuchen, die Match-Beschränktheit nachzuweisen, Dr. H.J. Sander Bruggink hat diesbezüglich in [2] bereits vor einigen Jahren einen Ansatz vorgestellt, der allerdings oftmals nicht zu einem Ergebnis führt. Mit den neuen Erkenntnissen über erkennbare Graphsprachen wäre es aber durchaus denkbar, dass man das Verfahren verfeinern könnte, auch wenn ähnlich gute Ergebnisse wie im Stringfall möglicherweise nicht erreicht werden können.

Literaturverzeichnis

[1] Christoph Blume, H.J. Sander Bruggink, and Barbara König. Treewidth, pathwidth and cospan decompositions. In *Proc. of GT-VMT '11 (Workshop on Graph Transformation and Visual Modeling Techniques)*, 2011.

[2] H.J. Sander Bruggink. Towards a systematic method for proving termination of graph transformation systems. In *Proceedings of GT-VC '07*, 2007.

[3] H.J. Sander Bruggink. Automaten und formale Sprachen, 2011.

[4] H.J. Sander Bruggink and Barbara König. On the recognizability of arrow and graph languages. In *Proc. of ICGT '08 (International Conference on Graph Transformation)*, pages 336–350. Springer, 2008. LNCS 5214.

[5] Sebastian Buchwald and Edgar Jakumeit. Compiler optimization: A case for the transformation tool contest. In *Transformation Tool Contest*, pages 6–16, 2011.

[6] Bruno Courcelle. Recognizable sets of graphs: Equivalent definitions and closure properties. *Mathematical Structures in Computer Science*, 4(1):1–32, 1994.

[7] Ingrid Fischer, Manuel Koch, and Michael R. Berthold. Proving properties of neural networks with graph transformations. In *IEEE International Joint Conference on Neural Networks*, volume 1, pages 441–446, 1998.

[8] Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-bounded string rewriting systems. In Branislav Rovan and Peter Vojtas, editors, *Mathematical Foundations of Computer Science 2003*, volume 2747 of *Lecture Notes in Computer Science*, pages 449–459. Springer Berlin / Heidelberg, 2003.

[9] Annegret Habel. Hyperedge Replacement: Grammars and Languages. *Lecture Notes in Computer Science*, 643, 1992.

[10] Annegret Habel and Detlef Plump. Computational completeness of programming languages based on graph transformation. In Furio Honsell and Marino Miculan, editors, *Foundations of Software Science and Computation Structures*, volume 2030 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001.

[11] Tobias Heindel. *A category theoretical approach to the concurrent semantics of rewriting: adhesive categories and related concepts*. PhD thesis, Universität Duisburg-Essen, 2009.

[12] Dieter Hofbauer and Johannes Waldmann. Deleting string rewriting systems preserve regularity. *Theor. Comput. Sci.*, 327(3):301–317, November 2004.

[13] Barbara König. Modellierung nebenläufiger Systeme, 2011.

[14] Sebastian Küpper. Bachelorarbeit Algorithmen für Baum- und Pfadzerlegungen von Graphen, Universität Duisburg-Essen, 2010.

[15] Stephen Lack and Pawel Sobocinski. Adhesive and quasiadhesive categories. *ITA*, 39(3):511–545, 2005.

[16] Benjamin C. Pierce, editor. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[17] Francesc Rosselló and Gabriel Valiente. Graph transformation in molecular biology. In *Formal Methods in Software and Systems Modeling*, pages 116–133, 2005.

[18] Eduardo Zambon and Arend Rensink. Using graph transformations and graph abstractions for software verification. *ECEASST*, 38, 2011.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Duisburg, den 13.08.2012