Seminar

Graph-Grammatiken

Lehrstuhl für Informatik III
RWTH Aachen
Sommersemester 2004

Graphumbeschriftungssysteme und Verteilte Algorithmen

Mathias Lüstraeten und Ruben Niederhagen

Inhaltsverzeichnis

1 Motivation

Lokale Berechnungen auf Graphen sind mächtige Modelle für die Formalisierung verteilter Algorithmen und deren Korrektheitsbeweise. Ein Spezialfall von lokalen Berechnungen stellen Graphumbeschriftungssysteme dar. Als Grundlage eines Graphumbeschriftungssystems dient ein zusammenhängender, ungerichteter Graph, dessen Knoten und Kanten beschriftet sind. Die Knoten- und Kantenbeschriftung des Graphen wird *Konfiguration* genannt. Ein Algorithmus besteht aus Regeln, die Zustandsänderungen des Graphen beschreiben. Während der Ausführung des Algorithmus befindet sich der Graph zu jedem Zeitpunkt in einer bestimmten Konfiguration. Dabei bleibt bei Graphumbeschriftungssystemen zu jedem Zeitpunkt die Struktur des Graphen erhalten.

Der Algorithmus startet mit einer Anfangskonfiguration. Ein *Berechnungsschritt* entspricht einer Konfigurationsänderung. Diese Konfigurationsänderung geschieht immer lokal. Das bedeutet, dass der Algorithmus anhand eines Teilgraphen entscheidet, welcher Berechnungsschritt ausgeführt wird. In wie weit sich die Konfiguration des Graphen ändert, wird durch die *Umbeschriftungsregeln* bestimmt. Falls eine linke Regelseite im Graphen vorkommt, wird dieses *Vorkommen* durch die rechte Regelseite ersetzt. Die Umbeschriftung wird solange durchgeführt, bis keine weitere Transformation möglich ist. Die Konfiguration heißt dann *Endkonfiguration*. Eine Endkonfiguration entspricht dem Ergebnis der Berechnung. Der zugehörige Graph ist dann in *Normalform*. Zwei sequentielle Umbeschriftungsschritte heißen *unabhängig*, wenn sie auf zwei disjunkten Teilgraphen angewendet werden. In diesem Fall können die beiden Schritte in beliebiger Reihenfolge oder sogar gleichzeitig angewendet werden.

Im Folgenden werden nur lokale Berechnungssysteme betrachtet, in denen eine Konfiguration höchstens einmal auftritt. Wäre ein mehrfaches Auftreten einer Konfiguration möglich, könnte ein Umbeschriftungszyklus entstehen, und der Graphumbeschriftungsalgorithmus würde nicht terminieren. Zusätzlich zur Terminierung eines Algorithmus kann geprüft werden, ob die globale Terminierung eines Systems auch lokal festgestellt werden kann, das heißt in einer k-Umgebung eines bestimmten Knotens. In diesem Fall heißt die globale Terminierung *k-lokal erkannt*. Ein Algorithmus heißt *Auswahl-Algorithmus*, wenn er in jedem Graphen genau einen Knoten mit einer *markierenden* Beschriftung versieht. Mit *Erkennungs-Algorithmus* wird ein Algorithmus bezeichnet, der feststellt, ob ein Graph zu einer bestimmten Klasse von Graphen gehört.

Zunächst werden im Kapitel 2 einige Begriffe der Graphentheorie geklärt. An dem Problem des Spannbaums wird im Kapitel 3 die Vorgehensweise und Notation der Graphumbeschriftungssysteme demonstriert. Im Kapitel 4 wird das formale Modell von Graphumbeschriftungssystemen eingeführt. Um die Ausdrucksstärke der Graphumbeschriftungssysteme zu erhöhen, werden Kontrollmechanismen eingeführt. Im Einzelnen sind dies Graphumbeschriftungssysteme mit Prioritäten (PGRS) (vgl. Abschnitt 4.3.1) und Graphumbeschriftungssysteme mit Verbotenen Kontexten (FC-GRS) (vgl. Abschnitt 4.3.2). Es folgen Beweistechniken, mit denen die Korrektheit von Graphumbeschriftungsalgorithmen gezeigt werden kann. Außerdem wird das Konzept der lokalen Berechnungen eingeführt, das eine Verallgemeinerung der Graphumbeschriftungssysteme darstellt.

In Kapitel 7 werden *Überdeckungen* eingeführt, ein Algorithmus für Ihre Erzeugung angegeben und ihre Einschränkung zu k-Überdeckungen definiert. Überdeckun-

gen werden in den darauf folgenden Kapiteln für grundlegende Sätze benötigt. In Kapitel 8 werden Auswahl- und in Kapitel 9 Erkennungs-Algorithmen definiert und jeweils durch Beispiele erläutert. Allerdings gibt es nicht für alle Klassen von Graphen solche Algorithmen; der letzte Abschnitt dieser Kapitel zeigt, wann eine Auswahl bzw. eine Erkennung nicht möglich ist. In Kapitel 10 wird das Terminierungs-Erkennungs-Problem betrachtet und auf Basis von Überdeckungen ein Beweisverfahren für die Nicht-Erkennbarkeit der Terminierung eines Algorithmus angegeben. Im letzten Kapitel folgt eine kurze Zusammenfassung der behandelten Themen.

Der Text orientiert sich an [1].

2 Graphen

Die hier betrachteten Graphen sind endlich, ungerichtet und enthalten keine Mehrfachkanten oder Schleifen. Ein Graph ist ein Paar $(V(G), E(G))$, in dem $V(G)$ eine endliche Knotenmenge und $E(G) \subseteq \{\{v, v'\} | v, v' \in V(G), v' \neq v\}$ die Menge der Kanten ist. Die Anzahl der Knoten in einem Graphen G ist die *Größe* von G.

Sei $e = \{v, v'\}$ eine Kante. Dann ist e *inzident* mit v und v', und v' ist ein *Nachbar* von v. Die Menge der Nachbarn von v (mit dem Knoten v selbst) heißt die *Nachbarschaft* von v. Diese wird mit $N_G(v)$ bezeichnet. Zwei Kanten sind *adjazent*, falls sie einen gemeinsamen Knoten haben. Der *Grad* eines Knotens v, $d(v)$, ist die Anzahl der zu v inzidenten Kanten. Knoten mit Grad 1 heißen *Blätter*, die restlichen Knoten *innere Knoten*. Ein *Pfad* P von v_1 zu v_i in G ist eine Sequenz $P = v_1, e_1, v_2, e_2, \ldots, e_{i-1}, v_i$ mit alternierend auftretenden Knoten und Kanten, so dass für jedes j, $1 \leq j \leq i$, e_j eine zu den Knoten v_j und v_{j+1} inzidente Kante ist. Falls $v_1 = v_i$, ist P ein Kreis. Ein Pfad P ist *einfach*, falls kein Knoten doppelt in P auftritt. Zwei Knoten v und v' sind *verbunden*, falls ein Pfad von v zu v' existiert.

Ein Graph ist *zusammenhängend*, falls von jedem Knoten ein Pfad zu den anderen Knoten existiert. Seien v und v' zwei verbundene Knoten. Dann ist die *Entfernung* von v und v', $d(v, v')$, die minimale Länge eines (einfachen) Pfades von v zu v'. Die maximale Distanz $d(v, v'), v, v' \in V(G)$, heißt der *Durchmesser von G*, $D(G)$. Ein Beispiel für einen Graphen ist ein *Baum*. Ein Baum ist ein kreisfreier, zusammenhängender Graph. Folglich sind alle Knotenpaare in einem Baum durch genau einen einfachen Pfad verbunden.

Seien G und G' zwei Graphen. G' ist ein *Teilgraph* von G, falls $V(G') \subseteq V(G)$ und $E(G') \subseteq E(G)$. Sei V' eine Teilmenge von $V(G)$. Der durch V' *induzierte Teilgraph*, $G[V']$, besitzt die Knotenmenge V' und enthält alle Kanten von G, deren beide Endpunkte zu V' gehören. Sei v ein Knoten und $k \in \mathbb{N} \setminus \{0\}$. Die *Umgebung* von k mit Mittelpunkt v, in Zeichen $B_G(v, k)$, ist der Teilgraph von G, der durch $V' = \{v' \in V \mid d(v, v') \leq k\}$ induziert wird.

Ein *Homomorphismus* eines Graphen G zu einem Graphen H ist ein Abbildung $\gamma : V(G) \longrightarrow V(H)$, so dass $\{\gamma(u), \gamma(v)\}$ eine Kante von H ist, falls $\{u, v\}$ eine Kante von G ist. Da H keine Schleifen hat, gilt $\gamma(u) \neq \gamma(v)$, falls $\{u, v\}$ eine Kante in G ist. Außerdem gilt $\gamma(N_G(u)) \subseteq \gamma(N_H(u))$ für jeden Knoten u. γ ist ein *Isomorphismus*, falls γ bijektiv ist und γ^{-1} ebenfalls ein Homomorphismus ist. Zwei Graphen sind *isomorph*, in Zeichen $G \simeq H$, falls ein Isomorphismus von G nach H existiert. Eine *Klasse* von Graphen ist eine Menge von Graphen, die unter Isomorphismus abgeschlossen ist.

Notation: Für $i, j \in \mathbb{N}$, $i \leq j$ kann die Menge $\{i, i+1, \ldots, j\}$ als $[i, j]$ geschrieben werden.

Definition 2.1 (Brücke, Artikulation)
Sei $G = (V, E)$ ein zusammenhängender Graph und $G' = (V, E')$ ein Graph mit $E' = E \setminus \{e\}$. Falls G' nicht zusammenhängend ist, heißt e *Brücke* von G.

Eine Artikulation ist ein Knoten v, so dass $G'' = (V'', E'')$ mit $V'' = V \setminus \{v\}$ und $E'' = E \setminus \{e \mid e \text{ inzident mit } v\}$ nicht mehr zusammenhängend ist.

Eine Brücke eines Graphen G ist eine Kante, deren Entfernung aus G dazu führt, dass G in zwei Teile zerfällt. Eine Artikulation eines Graphen G ist ein Knoten v, dessen Entfernung aus G mit allen zu v inzidenten Kanten dazu führt, dass G nicht mehr zusammenhängend ist.

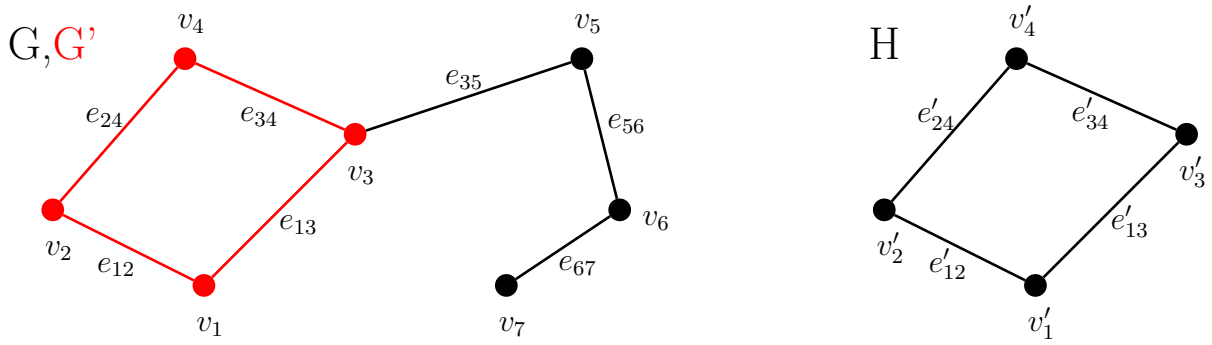


Abbildung 1: Beispiele für Graphen

Beispiel 2.2
Der in Abbildung 1 gezeigte Graph G hat die Größe 7, da G 7 Knoten hat. Kante e_{12} ist inzident mit v_1 und v_2. Die Nachbarn von v_3 sind v_1, v_4 und v_5. Die Kanten e_{35} und e_{56} sind adjazent, da sie beide mit v_5 verbunden sind. Der Knoten v_6 hat den Grad 2, da er zwei von ihm ausgehende Kanten hat. Das einzige Blatt in G ist der Knoten v_7, da er Grad 1 hat; somit sind alle anderen Knoten innere Knoten. Ein einfacher Pfad in G ist $v_1e_{12}v_2e_{24}v_4e_{34}v_3$. G ist zusammenhängend, da von jedem Knoten ein Pfad zu jedem anderen Knoten existiert. Die Entfernung von v_2 und v_5 ist 3 und der Durchmesser von G ist 5. G ist kein Baum, da in G der Kreis $v_1e_{12}v_2e_{24}v_4e_{34}v_3e_{13}v_1$ existiert. e_{35} ist eine Brücke, da G''mit $V(G'') = V(G)$ und $E(G'') = E(G) \setminus \{e_{35}\}$ nicht mehr zusammenhängend ist. v_5 ist eine Artikulation von G, da G''' mit $V(G''') = V(G) \setminus \{v_3\}$, und $E(G''') = E(G) \setminus \{e_{35}, e_{56}\}$ nicht mehr zusammenhängend ist.

G' ist der durch $V' = \{v_1, v_2, v_3, v_4\}$ induzierte Teilgraph. G' und H sind isomorph, da die Abbildung $\gamma : G' \to H$ ein bijektiver Homomorphismus ist. H gehört zur Klasse R_4, der Ringe mit 4 Knoten. Ein Ring R_n ist ein Graph $G = (V, E)$ mit $V(G) = \{v_1, \ldots, v_n\}$ und $E(G) = \{\{x, y\} \mid y = x + 1(mod\ n + 1)\}$.

Definition 2.3 (Spannbaum)
Ein Spannbaum eines zusammenhängenden Graphen $G = (V, E)$ ist ein Baum $T = (V, E')$, so dass $E' \subseteq E$.

Ein Spannbaum T eines Graphen G ist ein Baum, dessen Knotenmenge der Knotenmenge von G entspricht. Die Kanten von T sind jedoch nur eine Teilmenge der Kanten von E, so dass T ein Baum ist. Anschaulich entsteht ein Spannbaum T aus einem Graphen G, in dem so lange Kanten aus G weggelassen werden, bis G ein Baum ist.

Definition 2.4 (Restriktion einer Abbildung auf einen Teilgraph)
Seien $G = (V_1, E_1)$ und $H = (V_2, E_2)$ Graphen, $G' = (V'_1, E'_1)$ ein Teilgraph von G und $\gamma : G \to H$ eine Abbildung. Ferner sei $v_i \in V_i$ und $e_i \in E_i$ sowie $v'_1 \in V'_1$ und $e'_1 \in E'_1$. Dann ist die Restriktion von γ auf G' definiert als
$\gamma_{|G'} : G' \to H$ mit

$$v'_1 \mapsto v_2 \in \gamma_{|G'} \quad \text{, falls} \quad v_1 \mapsto v_2 \in \gamma \text{ und}$$
$$e'_1 \mapsto e_2 \in \gamma_{|G'} \quad \text{, falls} \quad e_1 \mapsto e_2 \in \gamma.$$

Eine Restriktion einer Abbildung γ auf einen Teilgraphen G' beschreibt dieselbe Abbildung γ, mit der Einschränkung, dass nur diejenigen Knoten und Kanten abgebildet werden, die in G' vorkommen. Dieser Formalismus ist nötig, da für eine Abbildung verlangt wird, dass *alle* Elemente einer Menge M abgebildet werden. Falls nur eine Teilmenge M' von M mit Hilfe von γ abgebildet werden soll, muss die Restriktion von γ auf M' verwendet werden.

3 Einführende Beispiele

Ziel dieses Kapitels ist es, den Begriff des Graphumbeschriftungssystems einzuführen. Dazu werden drei verschiedene Algorithmen zur Berechnung eines Spannbaumes auf beschrifteten Graphen vorgestellt.

Ein Algorithmus besteht aus Regeln, die Konfigurationsänderungen des Graphen beschreiben. Eine Konfiguration oder ein Zustand ist eine konkrete Knoten- und Kantenbeschriftung des gesamten Graphen. Während der Ausführung des Algorithmus befindet sich der Graph zu jedem Zeitpunkt in einer bestimmten Konfiguration. Der Graph wird immer nur in Umgebungen von so genannten aktiven Knoten umbeschriftet. Für jeden aktiven Knoten entscheidet der Algorithmus, welche Umbeschriftungsregel angewendet werden muss. Kann keine Regel mehr angewendet werden, terminiert der Algorithmus.

3.1 Sequentielle Berechnung eines Spannbaums

In folgendem Algorithmus soll ein Spannbaum in einem Graphen mit der Tiefensuche berechnet werden. Die Tiefensuche ist eine Traversierungsstrategie eines Baumes, dabei wird ausgehend von einem Knoten immer „in die Tiefe" gesucht. Von jedem Knoten wird jeweils das am weitesten links stehende Kind betrachtet. Wird ein Blatt erreicht, wird das nächste Kind seines direkten Vaterknotens v besucht. Falls kein Kind mehr existiert, wird beim Vaterknoten von v weiter gesucht. Die Tiefensuche terminiert, wenn alle Knoten besucht sind. Der gesamte Baum wird somit in einer festgelegten Reihenfolge durchlaufen.

Algorithmus 3.1 (Sequentielle Berechnung eines Spannbaumes)
Alle Knoten haben zu Anfang einen neutralen Zustand (N), bis auf genau einen Knoten, der als aktiv (A) markiert ist. Alle Kanten sind mit 0 beschriftet. Während der Laufzeit des Algorithmus ist zu jedem Zeitpunkt nur ein Knoten mit A beschriftet. In jedem Berechnungsschritt führt der Algorithmus für den mit A markierten Knoten u einen der folgenden Teilschritte aus:

1. Falls u einen mit N beschrifteten Nachbar v hat, aktiviert u seinen Nachbarn: u wird markiert (M), v wird aktiv (A) und die Kante $\{u, v\}$ wird mit 1 beschriftet.
2. Falls 1. nicht zutrifft, jedoch u genau einen mit M beschrifteten 1-Nachbar w hat, geht u in den Endzustand (F) über und w wird aktiv.

Die 1-Nachbarn eines Knotens u bezeichnen diejenigen Nachbarn von u, die mit einer 1-Kante mit u verbunden sind. Die Berechnung stoppt, falls keine der oben genannten Berechnungsregeln angewendet werden kann. In diesem Fall sind alle Nachbarn des mit A beschrifteten Knotens mit F markiert. Der Spannbaum ergibt sich aus den mit 1 markierten Kanten.

Die Bedingungen der beiden Teilschritte sind so formuliert, dass in jedem Berechnungsschritt nur genau eine Bedingung erfüllt wird.

Beispiel 3.2 (Sequentielle Spannbaumberechnung)
Abbildung 2 zeigt eine Beispielberechnung eines Spannbaumes mit Algorithmus 3.1. Der Graph befindet sich zunächst in der Anfangskonfiguration. In Schritt 2 wird gemäß Teilschritt 1 der nächste Knoten aktiviert (A), der vorherige Knoten wird markiert (M), und die Kante zwischen dem A- und dem M-Knoten wird mit 1 beschriftet. Analog dazu verlaufen auch die Schritte 2 bis 3. In Schritt 4 wird zum ersten Mal Teilschritt 2 angewendet, da der A-Knoten erstmals keinen N-Nachbar mehr hat. Somit geht der A-Knoten in den Endzustand (F) über und der markierte

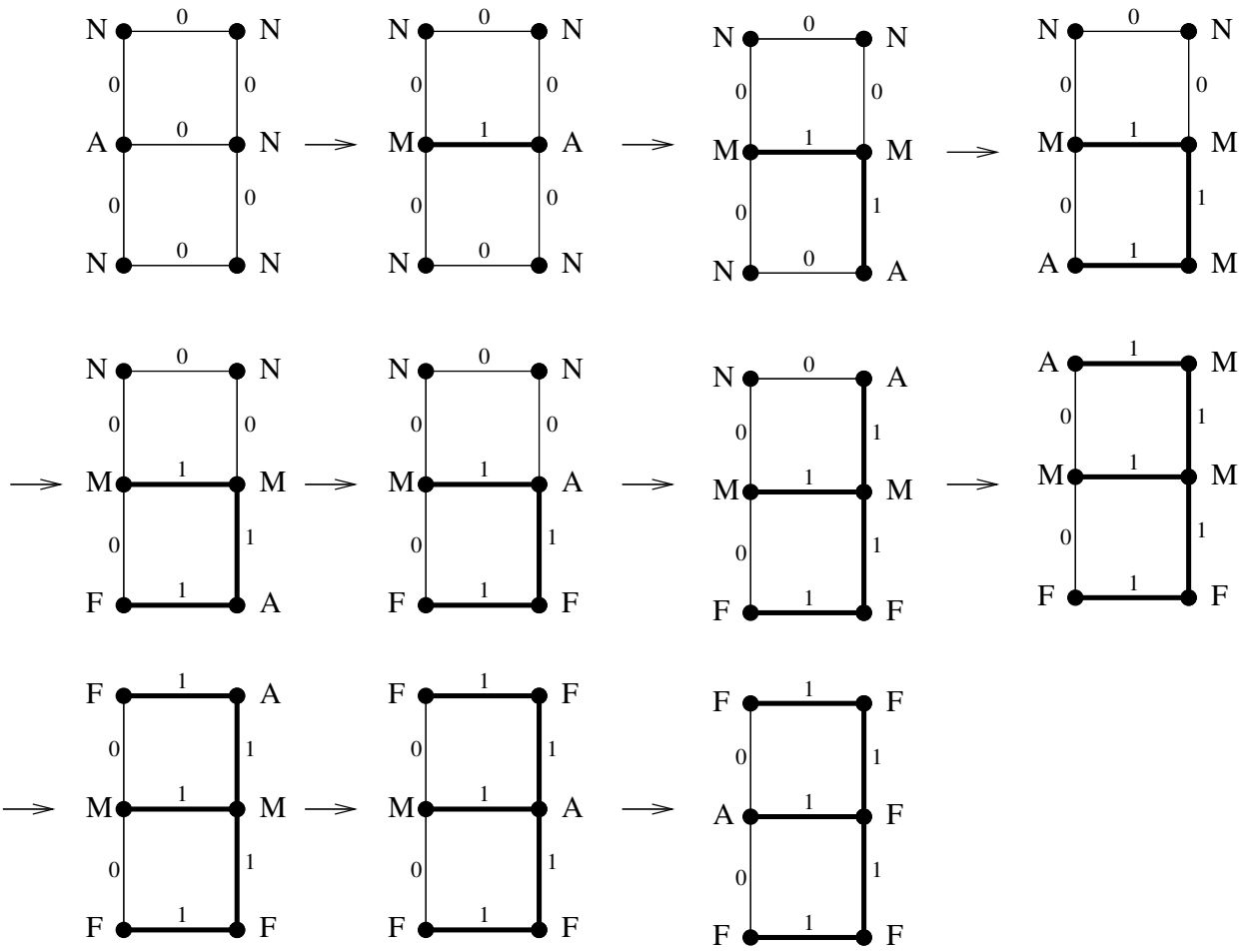


Abbildung 2: Sequentielle Berechnung eines Spannbaumes

Nachbar M wird reaktiviert, also mit A beschriftet. Im Folgenden wird je nach Situation entweder wie oben beschrieben Teilschritt 1 oder 2 angewendet. Im letzten Schritt kann keine Regel mehr angewendet werden und der Algorithmus terminiert. Der von dem A-Knoten, den F-Knoten und den 1-Kanten aufgespannte Baum ist dann der gesuchte Spannbaum.

3.2 Verteile Berechnung eines Spannbaumes ohne lokale Erkennung der globalen Terminierung

Im einführenden Beispiel wird der Spannbaum strikt sequentiell berechnet, da zu jedem Zeitpunkt nur höchstens ein Knoten aktiv ist. Es folgt ein Algorithmus, in dem einige Berechnungsschritte parallel ablaufen dürfen.

Algorithmus 3.3 (Verteile Berechnung eines Spannbaumes 1)
Der initale Graph sei wie oben konfiguriert, es sind also alle Knoten bis auf genau einen A-Knoten mit N beschriftet. Außerdem sind alle Kanten mit 0 initialisiert. In jedem Berechnungsschritt führt der Algorithmus für einen mit A markierten Knoten u folgende Regel aus: u aktiviert einen Knoten v aus der Menge seiner mit N beschrifteten Nachbarn. u behält seine Beschriftung, v wird mit A, und die Kante $\{u, v\}$ mit 1 beschriftet.

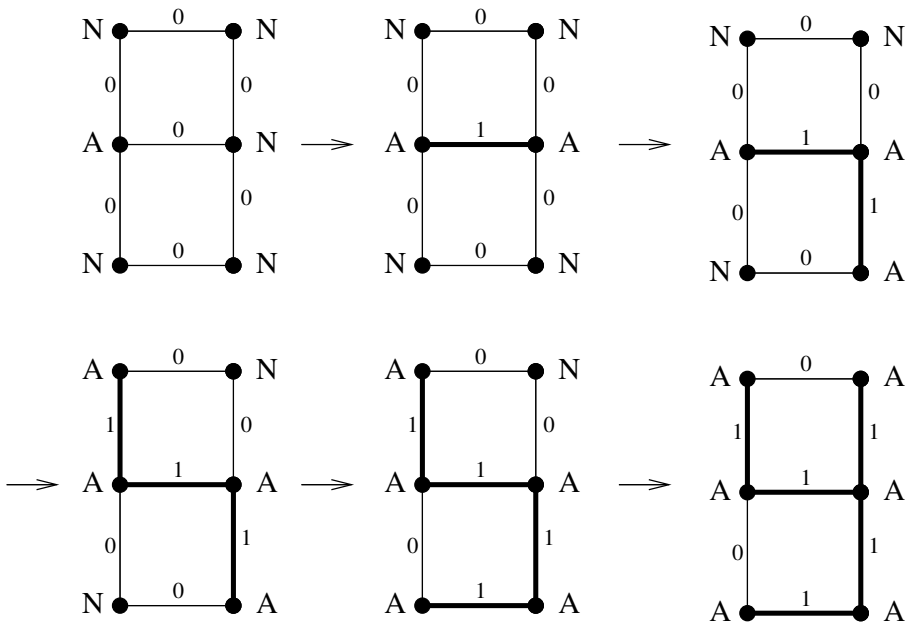


Abbildung 3: Verteilte Berechnung eines Spannbaumes 1

Bei diesem Algorithmus können zum gleichen Zeitpunkt mehrere Knoten aktiviert sein. Parallele Schritte sind erlaubt, falls sie disjunkte Mengen von Knoten betreffen. Die Berechnung stoppt, wenn alle Knoten aktiviert wurden. Der Spannbaum ergibt sich hier ebenfalls aus den 1-Kanten.

Beispiel 3.4 (Verteilte Berechnung eines Spannbaumes 1)
Abbildung 3 zeigt eine Beispielberechnung mit Algorithmus 3.3. Alle Schritte sind in der Abbildung sequentiell aufgeführt. Im ersten Schritt wird ein Knoten aus der Nachbarschaft mit N und die entsprechende Kante mit 1 beschriftet. Zu diesem Zeitpunkt kann nicht gleichzeitig der obere oder der untere N-Knoten aktiviert werden, da parallele Regelanwendung nur auf disjunkten Teilgraphen erlaubt ist. Schritt 3 und 4 sowie Schritt 4 und 5 dürfen parallel ausgeführt werden, da dort jeweils disjunkte Teilgraphen umbeschriftet werden. Der Graph ist nach der Ausführung von Schritt 5 in Normalform. Der Spannbaum ergibt sich aus den mit 1 beschrifteten Kanten.

3.3 Verteile Berechnung eines Spannbaumes mit lokaler Erkennung der globalen Terminierung

Algorithmus 3.1 ist in der Lage, lokal festzustellen, ob die Berechnung terminiert ist: Falls alle Nachbarn des aktiven Knotens mit F beschriftet sind, ist die Berechnung beendet. Algorithmus 3.3 besitzt diese Eigenschaft nicht, da es keinen ausgezeichneten Knoten mehr gibt, der die globale Terminierung lokal feststellen kann. Daher müssen alle Knoten mit einer gegebenen Beschriftung die Terminierung erkennen können. In Beispiel 3 sind in der Nachbarschaft des linken mittleren A-Knotens alle Knoten mit A beschriftet, der Algorithmus ist jedoch noch nicht terminiert. Somit kann Algorithmus 3.3 die Terminierung nicht lokal erkennen, bietet jedoch die Möglichkeit der verteilten Berechnung. Ziel ist es jetzt, einen verteilten Algorithmus zu formulieren, der die globale Terminierung lokal feststellen kann. Wie

vorher sind im initialen Graphen alle Knoten bis auf einen A-Knoten mit N beschriftet und alle Kanten 0-Kanten. Für neu aktivierte Knoten wird die Beschriftung A' gewählt, um den Startknoten A von anderen aktivierten Knoten zu unterscheiden. Sobald ein A'-Knoten nicht mehr für die Berechnung gebraucht wird, geht dieser in den Endzustand F über. Die zugehörige Kante bleibt mit 1 beschriftet. Falls der Startknoten A keinen A'-Nachbar mehr hat, ist die Berechnung beendet und A kann diese Terminierung lokal feststellen. Eine Lösung für diesen Ansatz zeigt Algorithmus 3.5.

Algorithmus 3.5 (Verteilte Berechnung eines Spannbaumes 2)
In jedem Berechnungsschritt führt jeder aktive Knoten u, also jeder mit A oder A' beschriftete Knoten, einen der Teilschritte aus:

1. Falls u einen N-Nachbar v hat, aktiviert u diesen Nachbarn: u behält seine Beschriftung, v wird mit A' beschriftet und damit aktiv. Außerdem wird die Kante $\{u, v\}$ mit 1 markiert.
2. Falls u mit A' beschriftet ist, keinen N-Nachbar hat und nur noch einen 1-Nachbar hat, der nicht mit F beschriftet ist, wird u mit F beschriftet.

Zu jedem Zeitpunkt ist der von den A-Knoten und den 1-Kanten aufgespannte Graph $\mathfrak{T}$ ein Baum. Durch Regel 1 wird der Baum aufgebaut, da diese Regel eine 1-Kante erzeugt und einen N-Knoten aktiviert. Bei der Aktivierung von neuen Knoten und Kanten kann kein Kreis entstehen: Ein Knoten, der zu einer 1-Kante inzident ist, und deshalb schon Teil des Spannbaumes ist, ist mit A oder A' beschriftet. Deshalb kann die erste Regel nicht angewendet werden.

Durch Regel 2 schrumpft der von den A- und A'-Knoten und den 1-Kanten aufgespannte Baum $\mathfrak{T}$, da ein A'-Knoten u, der Teil des Baumes $\mathfrak{T}$ ist, zu einem F-Knoten wird. Durch die Bedingungen in Regel 2 ist u nur noch mit genau einem aktivierten Knoten verbunden, und ist deshalb Blatt in $\mathfrak{T}$. Zu beachten ist, dass $\mathfrak{T}$ nicht der zu berechnende Spannbaum ist. Der gesuchte Spannbaum $\mathfrak{B}$ wird von den 1-Kanten aufgespannt. Da die 1-Kanten nie wieder mit 0 beschriftet werden, wächst $\mathfrak{B}$ nur durch Regel 1. Regel 2 hat keinen Einfluss auf $\mathfrak{B}$, da keine Kanten umbeschriftet werden. Regel 2 dient nur dazu, die globale Terminierung lokal feststellbar zu machen. Folglich läuft der Algorithmus in zwei Phasen ab, die auch überlappen können: In der ersten Phase wächst $\mathfrak{T}$ bis alle Knoten erreicht sind. In der zweiten Phase schrumpft $\mathfrak{T}$ durch Verlust seiner Blattknoten, bis nur noch der Startknoten übrig bleibt.

Falls der Algorithmus terminiert, ist der gesuchte Spannbaum der Teilgraph, der durch die 1-Kanten aufgespannt wird. Im Falle der Terminierung des Algorithmus sind alle Nachbarn des A-Knotens mit F beschriftet. Also kann der A-Knoten die Terminierung des Algorithmus lokal feststellen.

Beispiel 3.6 (Verteile Berechnung eines Spannbaumes 2)
Abbildung 4 zeigt eine Beispielberechnung eines Spannbaumes mit Algorithmus 3.5. Regel 1 des Algorithmus entspricht der Regel des Algorithmus 3.3. Deshalb laufen die ersten vier Schritte analog zu Beispiel 3.4 ab. Während dieser ersten vier Schritte hat jeder aktivierte Knoten einen N-Nachbar. Deshalb kann die zweite Regel nicht angewendet werden. Nach dem 4. Schritt hat der linke untere Knoten u keinen N-Nachbar mehr. Außerdem ist keiner seiner 1-Nachbarn mit F beschriftet. Also hat u genau einen 1-Nachbar, der nicht mit F beschriftet ist, und Regel 2 kann

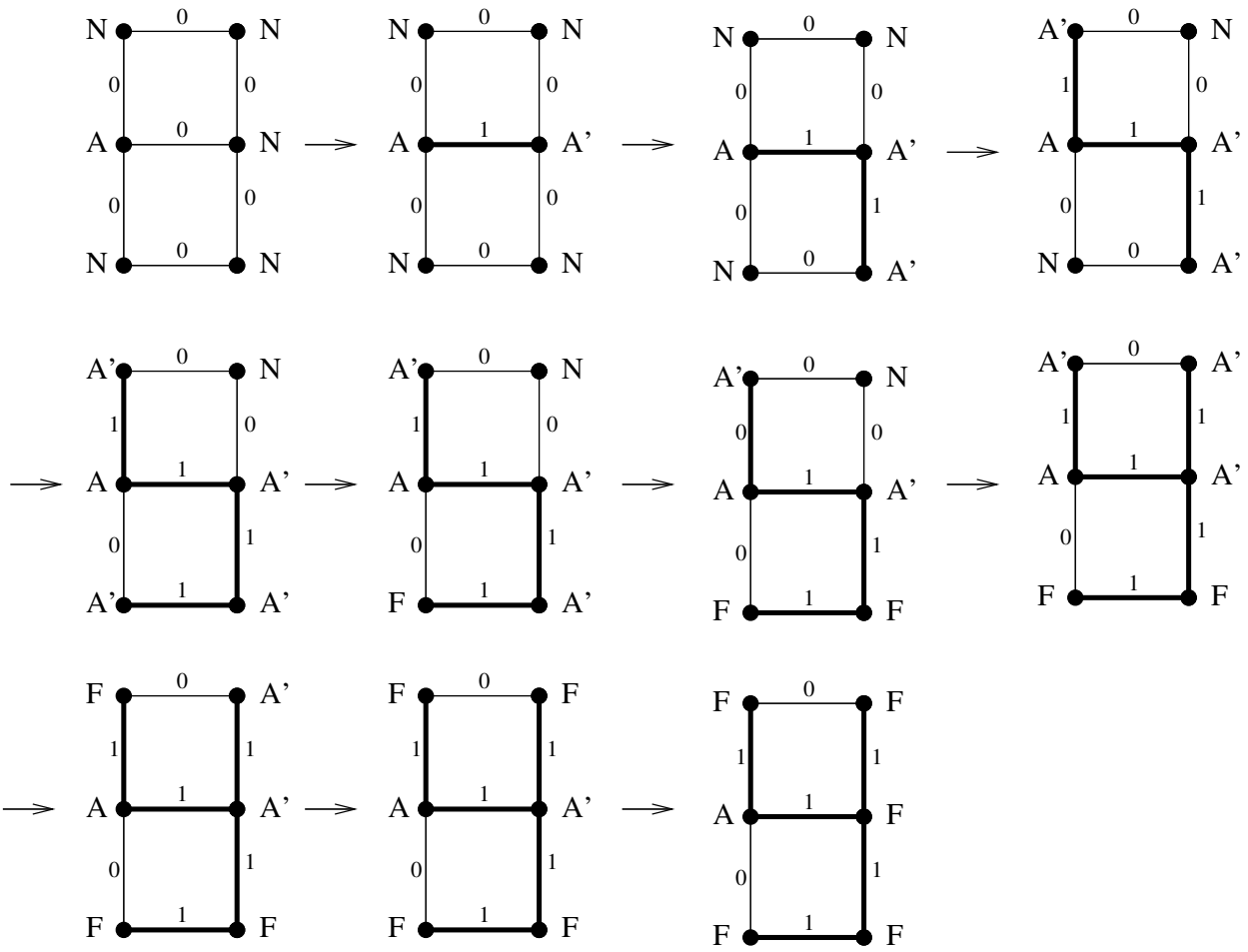


Abbildung 4: Verteilte Berechnung eines Spannbaumes 2

angewendet werden. Gemäß Regel 2 wird u somit zum F-Knoten. In Schritt 7 wird Regel 1 angewendet, in allen restlichen Schritten Regel 2.

In jedem Schritt ist nur ein Berechnungsschritt ausgeführt worden. Da jeder A- oder A'-Knoten aktiv ist, können Schritte, die disjunkte Teilgraphen betreffen, auch parallel ausgeführt werden. Zum Beispiel sind die Schritte 7 und 8 parallel ausführbar. Nach dem 10. Schritt kann keine Regel mehr angewendet werden. Der von den 1-Kanten aufgespannte Baum ist ein Spannbaum des Graphen. Alle Knoten in der Nachbarschaft des A-Knotens sind mit F beschriftet. Somit kann der A-Knoten die globale Terminierung lokal feststellen.

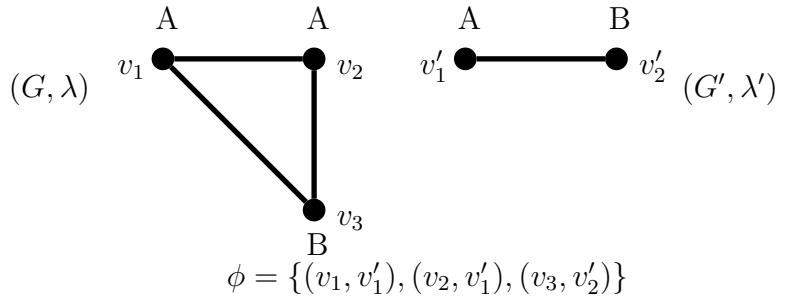


Abbildung 5: Homomorphismus $\phi : (G, \lambda) \to (G', \lambda')$

4 Graphumbeschriftungssysteme

In diesem Kapitel werden beschriftete Graphen und Graphumbeschriftungssysteme vorgestellt.

4.1 Markierte Graphen

Im Folgenden werden *L-markierte Graphen* betrachtet. Dies sind Graphen, deren Knoten und Kanten mit Buchstaben aus einem potentiell unendlichem Alphabet L beschriftet sind. Ein L-markierter Graph ist ein Paar (G, λ), wobei G ein Graph und $\lambda : V(G) \cup E(G) \longrightarrow L$ die *Markierungsfunktion* ist. Der Graph G wird *zugrundeliegender Graph* von (G, λ) genannt. Die Klasse der L-beschrifteten Graphen wird mit $\mathcal{G}_L$ bezeichnet, oder kurz $\mathcal{G}$, falls das Alphabet L aus dem Kontext ersichtlich ist.

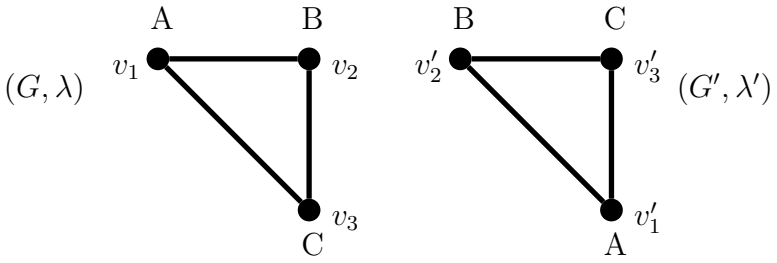
Seien (G, λ) und (G', λ') zwei beschriftete Graphen. (G, λ) ist ein Teilgraph von (G', λ'), bezeichnet durch $(G, \lambda) \subseteq (G', \lambda')$, falls G ein Teilgraph von G' und λ die Restriktion von λ' auf $V(G) \cup E(G)$ ist. Die Benutzung der Restriktion von λ' ist hier notwendig, da λ nur diejenigen Knoten und Kanten beschriften darf, die in G vorkommen. Zu beachten ist, dass sich für die Knoten und Kanten, die sowohl in G als auch in G' vorkommen, die Markierungsfunktion nicht ändert.

Eine Abbildung $\phi : V(G) \cup E(G) \longrightarrow V(G') \cup E(G')$ ist ein Homomorphismus von (G, λ) nach (G', λ'), falls ϕ ein markierungserhaltender Homomorphismus von G nach G' ist, also gilt: $\lambda'(\phi(x)) = \lambda(x) \forall x \in V(G) \cup E(G)$. Ein *Vorkommen* von (G, λ) in (G', λ') ist ein Isomorphismus ϕ zwischen (G, λ) und einem Teilgraph (H, η) von (G', λ').

Beispiel 4.1 (Homomorphismus)
Abbildung 5 zeigt einen Homomorphismus $\phi : (G, \lambda) \to (G', \lambda')$. Dabei ist $\phi : G \to G'$ ein Graphhomomorphismus: Die Kante $\{v_1', v_2'\}$ in H muss auch für die Urbilder von v_1' und v_2' existieren. Da $\phi(v_1')^{-1} = \{v_1, v_2\}$ und $\phi(v_2')^{-1} = \{v_3\}$, müssen auch die Kanten $\{v_1, v_3\}$ und $\{v_2, v_3\}$ in G existieren. Da diese Bedingung erfüllt ist, ist ϕ ein Graphhomomorphismus.

Für die Markierungen muss die Bedingung $\lambda'(\phi(x)) = \lambda(x)\ \forall x \in V(G) \cup E(G)$ gelten. Beispielsweise gilt: $\lambda'(\phi(v_1)) = \lambda'(v_1') = A = \lambda(v_1)$. Also ist ϕ ein Homomorphismus. Zu beachten ist, dass die Abbildung ϕ keine Auswirkung auf die Beschriftung hat. Die korrekte Beschriftung wird durch die Bedingung $\lambda'(\phi(x)) = \lambda(x)\ \forall x \in V(G) \cup E(G)$ sicher gestellt.

Beispiel 4.2 (Isomorphismus)
Abbildung 6 zeigt einen Isomorphismus $\phi : (G, \lambda) \to (G', \lambda')$. Ein Isomorphismus ist ein spezieller Homomorphismus (vgl. Beispiel 4.1), es muss also gelten, dass Bild



$$\phi = \{(v_1, v_1'), (v_2, v_2'), (v_3, v_3')\}$$

Abbildung 6: Isomorphismus $\phi : (G, \lambda) \to (G', \lambda')$

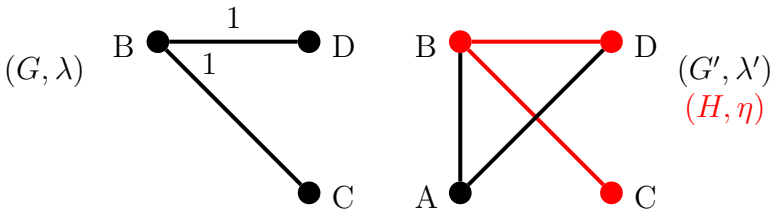


Abbildung 7: Vorkommen $\phi : (G, \lambda) \to (G', \lambda')$

und Urbild eines Knotens beziehungsweise einer Kante dieselbe Markierung haben. In einem Isomorphismus müssen die Graphen zusätzlich isomorph sein, also dieselbe Struktur haben.

Beispiel 4.3 (Vorkommen)
Der Begriff ,,Vorkommen" spiegelt das intuitive Verständnis eines Vorkommens eines Teilgraphens in einem Graphen wider. Abbildung 7 zeigt, dass (G, λ) in (G', λ') vorkommt. Dazu wird in (G', λ') ein Teilgraph (H, η) gesucht, der isomorph zu (G, λ) ist. Isomorphie bedeutet bei beschrifteten Graphen nicht nur, dass die beiden Graphen die gleiche Struktur, sondern auch die gleiche Markierung haben. Dies trifft für (G, λ) und (H, η) zu, also ist ϕ ein Vorkommen von (G, λ) in (G', λ')

4.2 Graphumbeschriftungssysteme

In diesem Abschnitt wird die formale Notation von Graphumbeschriftungssystemen vorgestellt.

Definition 4.4 (Graphumbeschriftungsregel)
Eine *(Graph-) Umbeschriftungsregel* ist ein Tripel $R = (G_R, \lambda_R, \lambda'_R)$, so dass (G_R, λ_R) und (G_R, λ'_R) zwei markierte Graphen sind. Der markierte Graph (G_R, λ_R) ist die *linke Regelseite* und (G_R, λ'_R) die *rechte Regelseite* von R.

G_R ist der Teilgraph von G, der durch die Regel R umbeschriftet wird. Bei der linken Regelseite ist der Graph mit λ_R, und bei der rechten Regelseite mit λ'_R beschriftet. Eine Graphumbeschriftungsregel beschreibt in einem Umbeschriftungsschritt eine gültige Zustandstransformation des Graphen.

Definition 4.5 (Graphumbeschriftungssystem)
Ein *Graphumbeschriftungssystem* (GRS) ist ein Tripel $\mathcal{R} = (L, I, P)$. Dabei ist L, das *Alphabet*, eine Menge von Markierungen, $I \subseteq L$ die Menge der *Anfangsbeschriftungen* und P, die *Regelmenge*, eine endliche Menge von Umbeschriftungsregeln.

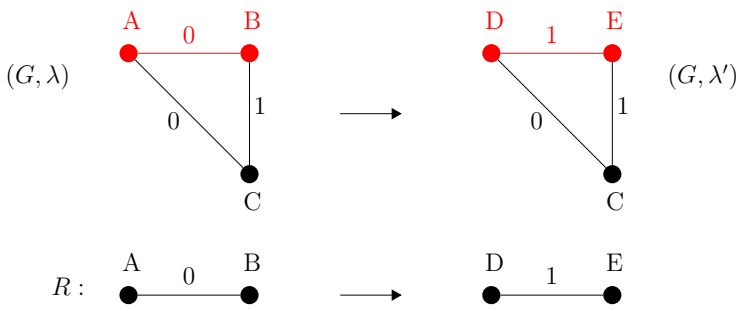


Abbildung 8: Umbeschriftungsschritt $(G, \lambda, R, \phi, \lambda')$

Einem Berechnungsschritt in einem Algorithmus entspricht dem Begriff eines Umbeschriftungsschrittes in einem Graphumbeschriftungssystem.

Definition 4.6 (Umbeschriftungsschritt)
Ein $\mathcal{R}$-*Umbeschriftungsschritt* ist ein 5-Tupel $(G, \lambda, R, \phi, \lambda')$, so dass R eine Umbeschriftungsregel in P ist und ϕ sowohl ein Vorkommen von (G_R, λ_R) in (G, λ) ist als auch ein Vorkommen von (G_R, λ'_R) in (G, λ').

Anschaulich wählt die Abbildung ϕ aus (G, λ) einen Teilgraphen aus, der der linken Regelseite (G_R, λ_R) von R entspricht. Mit der rechten Regelseite (G_R, λ'_R) wird der markierte Graph (G, λ') konstruiert, indem der durch ϕ ausgewählte Teilgraph gemäß der rechten Regelseite umbeschriftet wird. Somit entsteht (G, λ') aus (G, λ), indem ein Vorkommen der linken Regelseite von R durch die entsprechende rechte Regelseite ersetzt wird. Der zugrundeliegende Graph bleibt erhalten, da nur die Beschriftungen geändert werden. Solch ein Umbeschrifungsschritt wird bezeichnet mit $(G, \lambda) \longrightarrow_{R,\phi} (G, \lambda')$.

Beispiel 4.7 (Umbeschriftungsschritt)
Abbildung 8 zeigt einen Umbeschriftungsschritt mit der Regel R. Die Abbildung ϕ wählt den Teilgraphen aus, der der linken Regelseite entspricht und beschriftet diesen gemäß R um.

Eine Berechnung im Algorithmus entspricht einer Umbeschriftungssequenz. Eine Umbeschriftungssequenz ist eine Hintereinanderausführung mehrerer Umbeschriftungsschritte.

Definition 4.8 (Umbeschriftungssequenz)
Eine $\mathcal{R}$-*Umbeschriftungssequenz* ist ein Tupel

$$(G, \lambda_0, R_0, \phi_0, \lambda_1, R_1, \phi_1, \lambda_2, \ldots, \lambda_{n-1}, R_{n-1}, \phi_{n-1}, \lambda_n)$$

so dass für jedes i, $0 \leq i \leq n$, $(G, \lambda_i, R_i, \phi_i, \lambda_{i+1})$ ein $\mathcal{R}$-Umbeschriftungsschritt ist. Die Existenz einer solchen Umbeschriftungssequenz wird bezeichnet durch $(G, \lambda_0) \longrightarrow_R^* (G, \lambda_n)$.

Die Berechnung stoppt, wenn der Graph so beschriftet ist, dass keine Umbeschriftungsregel mehr angewendet werden kann:

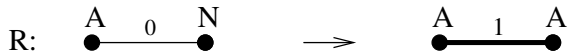


Abbildung 9: Umbeschriftungsregel R

Definition 4.9 (irreduzibel, Normalform)
Ein markierter Graph (G, λ) heißt $\mathcal{R}$-*irreduzibel*, falls für keine Umbeschriftungsregel R in P ein Vorkommen von (G_R, λ_R) in (G, λ) existiert. Ein beschrifteter Graph (G, λ) ist in *Normalform*, falls er irreduzibel ist.
Für jeden beschrifteten Graph (G, λ) in $\mathcal{G}_I$ ist $Irred_{\mathcal{R}}(G, \lambda)$ die Menge aller $\mathcal{R}$-irreduziblen beschrifteten Graphen (G, λ'), so dass $(G, \lambda) \longrightarrow_{\mathcal{R}}^{*} (G, \lambda')$.

Ein beschrifteter Graph (G, λ) ist also genau dann irreduzibel, falls keine linke Regelseite mehr in (G, λ) vorkommt. In diesem Fall kann keine Regel mehr angewendet werden. Die Menge $Irred_{\mathcal{R}}(G, \lambda)$ enthält alle terminalen Beschriftungen, die von einem I-beschrifteten Graphen durch Anwendung von Umbeschriftungsregeln in P abgeleitet werden können. $Irred_{\mathcal{R}}(G, \lambda)$ entspricht der Menge aller möglichen Ergebnisse der durch das System $\mathcal{R}$ formalisierten Berechnung.

Beispiel 4.10
Algorithmus 3.3 kann durch das folgende Graphumbeschriftungssystem $\mathcal{R}_1$ beschrieben werden:
$\mathcal{R}_1 = (L_1, I_1, P_1)$, mit dem Alphabet $L_1 = \{N, A, 0, 1\}$, den Anfangsbeschriftungen $I_1 = \{N, A, 0\}$, der Regelmenge $P_1 = \{R\}$ mit R wie in Abbildung 9.

4.3 Lokale Kontrollmechanismen

Um eine zufriedenstellende Ausdrucksstärke von Graphumbeschriftungssystemen zu erhalten, werden zwei verschiedene *lokale Kontrollmechanismen* eingeführt. Diese verhindern in bestimmten Fällen die Anwendung von Umbeschriftungsregeln.

4.3.1 Graphumbeschriftungssysteme mit Prioritäten

Der erste hier betrachtete Kontrollmechanismus benutzt eine Prioritätsrelation auf der Menge der Umbeschriftungsregeln.

Definition 4.11 (PGRS)
Ein *Graphumbeschriftungssystem mit Prioritäten* (PGRS) ist ein 4-Tupel $\mathcal{R} = (L, I, P, >)$, so dass (L, I, P) ein Graphumbeschriftungssystem und $>$ eine partielle Ordnung auf der Menge P ist. $>$ heißt dann *Prioritätsrelation*.

Ein $\mathcal{R}$-*Umbeschriftungsschritt* ist dann ein 5-Tupel $(G, \lambda, R, \phi, \lambda')$, so dass R eine Umbeschriftungsregel in P und ϕ sowohl ein Vorkommen von (G_R, λ_R) in (G, λ) als auch ein Vorkommen von (G_R, λ'_R) in (G, λ') ist. Außerdem muss für alle Vorkommen ϕ einer Umbeschriftungsregel R' mit $R' > R$ gelten, dass $V(\phi(G_R)) \cap V(\phi(G_{R'})) = \emptyset$.

Der Begriff der Umbeschriftungssequenz ist wie vorher definiert.

Die Prioritätsrelation auf den Regeln stellt sicher, dass eine Regel nur dann auf ein Vorkommen anwendbar ist, falls keine Regel mit einer höheren Priorität angewendet werden kann. Die Bedingung, dass für alle Vorkommen ϕ einer Umbeschriftungsregel R' mit $R' > R$ gelten muss, dass $V(\phi(G_R)) \cap V(\phi(G_{R'})) = \emptyset$, leistet dies. $V(\phi(G_{R'}))$ ist die Knotenmenge der linken Regelseite mit höherer Priorität, und $V(\phi(G_R))$

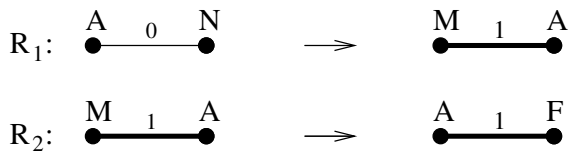


Abbildung 10: Umbeschriftungsregeln R_1 und R_2

die Knotenmenge der linken Regelseite mit niedrigerer Priorität. Durch die Forderung der Disjunktheit von $V(\phi(G_R))$ und $V(\phi(G_{R'}))$ wird sichergestellt, dass an zwei überlappenden Vorkommen von linken Regelseiten immer zuerst die Regel mit höherer Priorität angewendet werden muss.

Beispiel 4.12
Algorithmus 3.1 kann durch das folgende PGRS $\mathcal{R}_2$ codiert werden:
$\mathcal{R}_2 = (L_2, I_2, P_2, >_2)$, mit dem Alphabet $L_2 = \{N, A, M, F, 0, 1\}$, den Anfangsbeschriftungen $I_2 = \{N, A, 0\}$, der Regelmenge $P_2 = \{R_1, R_2\}$ mit R_1 und R_2 wie in Abbildung 10 und der Prioritätsrelation $R_1 >_2 R_2$.

Die Prioritätsrelation sorgt in diesem Beispiel dafür, dass die Formulierung ,,falls 1. nicht zutrifft" umgesetzt wird, da die Regel 1 eine höhere Priorität als die Regel 2 hat.

4.3.2 Graphumbeschriftungssysteme mit Verbotenen Kontexten

Der zweite hier vorgestellte Kontrollmechanismus verhindert die Anwendung einer Graphumbeschriftungsregel immer dann, wenn die linke Seite der Regel in einer besonderen Konfiguration, dem Kontext, enthalten ist. Formal werden dazu die folgenden Begriffe eingeführt.

Definition 4.13 (Kontext)
Sei (G, λ) ein markierter Graph. Ein *Kontext* von (G, λ) ist ein Tripel (H, μ, ψ), so dass (H, μ) ein markierter Graph und ψ ein Vorkommen von (G, λ) in (H, μ) ist.

(G, λ) ist ein Teilgraph von (H, μ) und (H, μ) ist der Kontext von (G, λ). Dieser Kontext gibt ein mögliches Umfeld von (G, λ) an. Die Abbildung ψ gibt an, an welcher Stelle (H, μ) in (G, λ) vorkommt.

Definition 4.14 (Umbeschriftungsregel mit verbotenen Kontexten)
Eine *Umbeschriftungsregel mit verbotenen Kontexten* ist ein 4-Tupel $R = (G_R, \lambda_R, \lambda'_R, F_R)$, so dass $(G_R, \lambda_R, \lambda'_R)$ eine Umbeschriftungsregel und F_R eine endliche Menge von Kontexten von (G_R, λ_R) ist.

Die ursprünglichen Umbeschriftungsregeln werden damit jeweils um eine Menge erweitert, deren Elemente die Kontexte sind. Diese Kontexte werden in einem Umbeschriftungsschritt als Kontrollmechanismus verwendet, indem die Anwendung der Regel nur dann möglich ist, falls die linke Regelseite in keinem Kontext der Regel vorkommt.

Definition 4.15 (Graphumbeschriftungssystem mit verbotenen Kontexten)
Ein *Graphumbeschriftungssystem mit verbotenen Kontexten* (FCGRS) ist ein Tripel $\mathcal{R} = (L, I, P)$, mit L und I wie bei GRS und P eine Menge von Umbeschriftungsregeln mit verbotenen Kontexten.

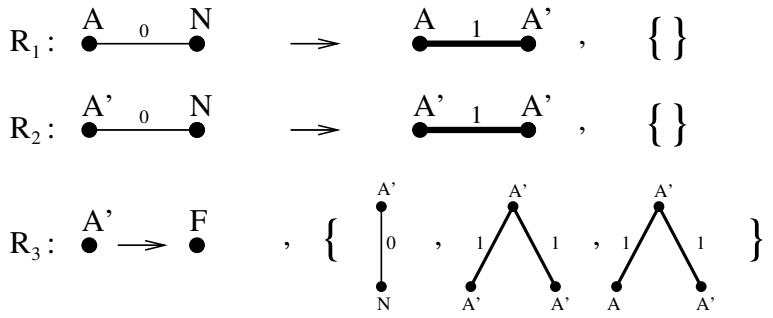


Abbildung 11: Umbeschriftungsregeln R_1, R_2 und R_3

Ein Graphumbeschriftungssystem mit verbotenen Kontexten ist ein GRS, bei dem die Regeln um die Menge der Kontexte erweitert wurde. Eine Umbeschriftungsregel mit verbotenen Kontexten kann genau dann auf ein Vorkommen angewendet werden, wenn dieses Vorkommen nicht in einem Vorkommen innerhalb seiner verbotenen Kontexte auftritt. Formal bedeutet dies:

Definition 4.16 (Umbeschriftungsschritt)
Ein $\mathcal{R}$-*Umbeschriftungsschritt* ist ein 5-Tupel $(G, \lambda, R, \phi, \lambda')$ mit folgenden Eigenschaften: R ist eine Umbeschriftungsregel mit verbotenem Kontext, ϕ sowohl ein Vorkommen von (G_R, λ_R) in (G, λ), als auch ein Vorkommen von (G_R, λ'_R) in (G, λ'), und für jeden Kontext (H_i, μ_i, ψ_i) von (G_R, λ_R) existiert kein Vorkommen ϕ_i von (H_i, μ_i), so dass $\phi_i(\psi_i(G_R, \lambda_R)) = \phi(G_R, \lambda_R)$.

Beispiel 4.17
Algorithmus 3.5 kann durch das folgende FCGRS $\mathcal{R}_3$ wie folgt codiert werden:
$\mathcal{R}_3 = (L_3, I_3, P_3)$, mit dem Alphabet $L_3 = \{N, A, M, F, 0, 1\}$, den Anfangsbeschriftungen $I_3 = \{N, A, 0\}$, und der Regelmenge $P_3 = \{R_1, R_2, R_3\}$, wobei R_1, R_2 und R_3 wie in Abbildung 11 definiert sind.

Bei den ersten beiden Regeln gibt die leere Menge hinter den eigentlichen Regeln an, dass bei der Ersetzung kein Kontext zu berücksichtigen ist. Diese beiden Regeln können also immer angewendet werden, falls die linke Regelseite im Graphen vorkommt. Die dritte Regel hat drei verbotene Kontexte. Im Kontext entspricht immer der obere A'-Knoten dem A'-Knoten auf der linken Regelseite. Der erste Kontext der Regel 1 entspricht der wörtlichen Formulierung: „falls A' keinen N-Nachbar hat". Der zweite und der dritte Kontext entsprechen der Aussage, dass der A'-Knoten keine zwei 1-Nachbarn hat, die nicht mit F beschriftet sind. Dies kann deshalb so formuliert werden, da 1-Nachbarn eines Knotens immer nur A, A' oder F-Knoten sein können.

4.3.3 Vergleich zwischen PGRS und FCGRS

Eine in der theoretischen Informatik wichtige Frage bezüglich Formalismen für Algorithmen ist die Frage der Mächtigkeit, also die Frage, welche Klasse von Problemen durch diesen Formalismus beschrieben werden kann. An dieser Stelle soll jedoch nur die Mächtigkeit von PGRS und FCGRS verglichen werden.

Satz 4.18
PGRS und FCGRS sind äquivalent.

Dies ist eine Feststellung, die nicht sofort einzusehen ist und dessen Beweis umfangreich ist (vgl. [1]). Für die Anwendung ergibt sich daraus, dass zwischen dem Formalismus PGRS und FCGRS für jedes Problem frei gewählt werden kann.

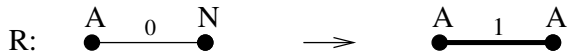


Abbildung 12: Umbeschriftungsregel R

5 Beweistechniken

Anhand des Graphumbeschriftungssystems zu einem Algorithmus lassen sich bestimmte Eigenschaften des Algorithmus feststellen. Zu diesen Eigenschaften zählen Korrektheit, Terminierung, Terminierungserkennung sowie Zeitkomplexität. Ziel dieses Kapitels ist, anhand der oben eingeführten Algorithmen zu zeigen, wie solche Eigenschaften bewiesen werden können.

Die Analogie zwischen Eigenschaften verteilter Algorithmen und Eigenschaften von Graphumbeschriftungssystemen ist wie folgt: Sei $\mathcal{A}$ ein verteilter Algorithmus, der durch ein Graphumbeschriftungssystem $\mathcal{R}$ codiert wird, dann gilt:

- Der Algorithmus $\mathcal{A}$ ist korrekt, falls für jeden beschrifteten Graph (G, λ) jeder beschrifteter Graph in $Irred_{\mathcal{R}}(G, \lambda)$ eine gültige Lösung des Algorithmus beschreibt.
- Der Algorithmus $\mathcal{A}$ terminiert genau dann, wenn jede $\mathcal{R}$-Umbeschriftungskette endlich ist.
- Die Zeitkomplexität von $\mathcal{A}$ ergibt sich aus der Betrachtung der längsten $\mathcal{R}$-Umbeschriftungskette.
- Der Algorithmus $\mathcal{A}$ hat die Eigenschaft des lokalen Erkennens der globalen Terminierung, falls eine Eigenschaft P und eine Zahl $k \in \mathbb{N} \setminus \{0\}$ mit der folgenden Eigenschaft existiert: Jeder markierte Graph, der einen Knoten v enthält, so dass die Umgebung $B(v, k)$ die Bedingung P erfüllt, ist $\mathcal{R}$-irreduzibel (In diesem Fall ist der Knoten v in der Lage, die Terminierung zu erkennen).

Die Technik zur Überprüfung der Korrektheit eines Algorithmus, beziehungsweise eines Graphumbeschriftungssystems, nutzt *invariante Eigenschaften*. Diese werden von jedem mit einer Anfangsbeschriftung versehenen Graph erfüllt und von jedem Umbeschriftungsschritt erhalten. Aus einem $\mathcal{R}$-irreduziblen Graph (G, λ) kann dann unter Zuhilfenahme der invarianten Eigenschaften die Korrektheit gefolgert werden. Im Folgenden wird die Korrektheit der Graphumbeschriftungssysteme $\mathcal{R}_1$ und $\mathcal{R}_2$ bewiesen.

5.1 Das Graphumbeschriftungssystem $\mathcal{R}_1$

Wiederholung: Das GRS $\mathcal{R}_1$ ist durch $\mathcal{R}_1 = (L_1, I_1, P_1)$ gegeben mit $L_1 = \{N, A, 0, 1\}$, $I_1 = \{N, A, 0\}$ und $P_1 = \{R\}$, wobei R die in Abbildung 12 dargestellte Graphumbeschriftungsregel ist.

Satz 5.1

1. *Jede $\mathcal{R}_1$-Umbeschriftungssequenz ist endlich.*
2. *Sei (G, λ) ein mit I_1 beschrifteter Graph, so dass genau ein Knoten mit A beschriftet ist und (G, λ') ein beliebiger mit L_1 beschrifteter Graph in $Irred_{\mathcal{R}}(G, \lambda)$. Dann induziert die Menge der mit 1 beschrifteten Kanten einen Spannbaum von G.*
3. *Die Länge einer maximalen $\mathcal{R}_1$-Umbeschriftungskette hat höchstens $|V(G)| - 1$ viele Schritte.*

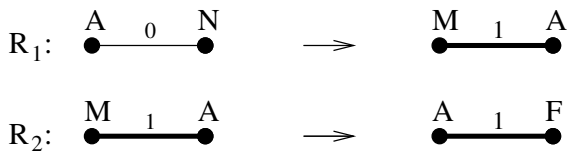


Abbildung 13: Umbeschriftungsregeln R_1 und R_2

Der erste Punkt bedeutet, dass der durch $\mathcal{R}$ formalisierte Algorithmus stets terminiert. Punkt zwei sagt aus, dass der Algorithmus korrekt ist, da jeder irreduzible Graph einen durch die 1-Kanten induzierten Spannbaum enthält. Der letzte Punkt gibt die Komplexität des Algorithmus an.

Beweis

1. Jede Anwendung der Regel R verringert die Anzahl der N-Knoten. Da die Regel R nur dann angewendet werden kann, falls ein N-Knoten in (G, λ) vorkommt, ist jede $\mathcal{R}_1$ Umbeschriftungssequenz endlich
2. Für den Beweis von Teil 2 werden folgende invariante Eigenschaften herangezogen:

   P_1 Jede zu einem mit N beschrifteten Knoten inzidente Kante ist mit 0 beschriftet.

   P_2 Jeder mit A beschriftete Knoten ist zu mindestens einer 1-Kante inzident, es sei denn, es existiert keine 1-Kante.

   P_3 Der von den 1-Kanten induzierte Teilgraph ist ein Baum.

   Diese drei Eigenschaften werden von (G, λ) erfüllt. Die Anwendung der Regel R erhält die invarianten Eigenschaften.
   Sei nun (G, λ') ein beliebiger markierter Graph in $Irred_{\mathcal{R}_1}(G, \lambda)$. Da (G, λ') irreduzibel ist, enthält es keinen mit N beschrifteten Knoten. Dann folgt mit den Eigenschaften P_1 und P_2, dass der von den 1-Kanten induzierte Teilgraph von (G, λ') ein Spannbaum von G ist.
3. Die maximale Länge einer $\mathcal{R}_1$-Umbeschriftungskette ist äquivalent zu der Anzahl der mit N beschrifteten Knoten in (G, λ). Diese beträgt $|V(G)| - 1$, da jede Anwendung der Regel R eine N-Beschriftung durch eine A-Beschriftung ersetzt. □

5.2 Das Graphumbeschriftungssystem $\mathcal{R}_2$ mit Prioritäten

Wiederholung: Das PGRS $\mathcal{R}_2$ ist durch $\mathcal{R}_2 = (L_2, I_2, P_2, >_2)$ gegeben mit $L_2 = \{N, A, M, F, 0, 1\}$, $I_2 = \{N, A, 0\}$ und $P_2 = \{R_1, R_2\}$, wobei R_1 und R_2 die in Abbildung 13 dargestellten Graphumbeschriftungsregeln sind.

Satz 5.2

1. Jede $\mathcal{R}_2$-Umbeschriftungssequenz ist endlich.

2. Sei (G, λ) ein mit I_2 beschrifteter Graph, so dass genau ein Knoten mit A beschriftet ist und (G, λ') ein beliebiger mit L_2 beschrifteter Graph in $Irred_{\mathcal{R}_2}(G, \lambda)$. Dann induziert die Menge der 1-Kanten in (G, λ') einen Spannbaum von G.

3. Die Länge einer maximalen bei (G, λ) beginnenden $\mathcal{R}_2$-Umbeschriftungskette hat höchstens $2(|V(G)| - 1)$ viele Schritte.

4. Für jede $\mathcal{R}_2$-Umbeschriftungskette von (G, λ) zu einem markierten Graph (G, λ'') ist der Graph (G, λ'') irreduzibel genau dann, wenn er einen mit A beschrifteten Knoten enthält, dessen Nachbarn alle mit F beschriftet sind.

Punkt 1 bedeutet dass der Algorithmus stets terminiert. Punkt zwei sagt aus, dass der Algorithmus für jede zulässige Eingabe einen Spannbaum von G berechnet. Der dritte Punkt gibt die Komplexität an. Punkt 4 wird für den Beweis der Terminierungserkennung benötigt. Falls diese Beziehung gilt, ist der Algorithmus genau dann terminiert, wenn alle Nachbarn des A-Knotens F-Knoten sind. Damit kann der A-Knoten die Terminierung lokal feststellen.

Beweis

1. Eine Anwendung der Regel R_1 erhöht die Zahl der M-Knoten und erniedrigt die Zahl der N-Knoten jeweils um eins. Es können daher von R_1 nur endlich viele M-Knoten erzeugt werden. Die Regel R_2 erniedrigt die Zahl der M-Knoten um eins. Damit ist jede $\mathcal{R}_2$-Umbeschriftungssequenz endlich.
2. Für den Beweis von Teil 2 werden folgende invariante Eigenschaften herangezogen:

   P_1 Jede zu einem mit N-Knoten inzidente Kante ist mit 0 beschriftet.

   P_2 Jeder mit A, M oder F beschriftete Knoten ist zu mindestens einer 1-Kante inzident, es sei denn, es existiert keine 1-Kante.

   P_3 Der von den 1-Kanten induzierte Teilgraph ist ein Baum.

   P_4 Es existiert genau ein A-Knoten.

   P_5 Die M- und A-Knoten spannen mit den 1-Kanten ein Pfad mit Endpunkt A auf.

   P_6 Jeder F-Knoten hat keinen mit N beschrifteten Nachbar.

   Diese Eigenschaften werden offensichtlich von dem Anfangsgraphen (G, λ) erfüllt. Bleibt zu zeigen, dass die Regeln R_1 und R_2 diese Eigenschaften erhalten:

   P_1 Nur die Regel R_1 erzeugt eine 1-Kante. In diesem Fall ist keiner der Endpunkte dieser Kante mit N beschriftet.

   P_2 Bei der Umbeschriftung eines Knotens zu einem $M-$, $A-$ oder $F-$ Knoten bleibt dieser nach Regel R_1 und R_2 zu einer 1-Kante inzident.

   P_3 Nur die Regel R_1 erzeugt eine 1-Kante. Einer der Endpunkte dieser Kante war mit N beschriftet und hatte gemäß P_1 keine weitere inzidente 1-Kante. Folglich kann dieser neue Knoten keinen Kreis erzeugen. Der andere Endpunkt war mit A beschriftet und gemäß P_2 schon inzident zu mindestens einer 1-Kante. Also ist der durch die 1-Kanten induzierte Teilgraph zusammenhängend.

   P_4 Bei jeder Regelanwendung bleibt die Zahl der A-Knoten gleich. Da zu Anfang genau ein A-Knoten existiert, hat nach einer Regelanwendung der Graph wieder genau einen A-Knoten.

   P_5 Im Graph existiert zu jedem Zeitpunkt ein Pfad p, der von den M-Knoten und dem A-Knoten mit den 1-Kanten aufgespannt wird. Der Endknoten dieses Pfades ist der A-Knoten. Die erste Regel verlängert den Pfad p um einen weiteren Knoten, da ein ursprünglich mit N beschrifteter Knoten zu dem neuen A-Knoten und der vorherige A-Knoten zu einem M-Knoten wird, und

außerdem die verbindende Kante mit 1 beschriftet wird. Da der neu aufgenommene Knoten ein N-Knoten war, und dieser deshalb noch zu keiner 1-Kante inzident war (P_1), ist der erweiterte Pfad wieder ein Pfad.
Regel 2 kann auf das Ende des Pfades angewendet werden, da p den Endknoten A hat und der Vorgängerknoten von A ein M-Knoten ist. Nach der Regelanwendung wird der A-Knoten mit F und der M-Knoten mit A beschriftet. Dadurch ist die Länge des Pfades p um 1 kleiner. Da p vor der Regelanwendung ein Pfad war, ist p nach der Regelanwendung immer noch ein Pfad.

P_6 Diese Eigenschaft wird durch den Prioritätsmechanismus sicher gestellt: Falls ein mit A markierter Knoten einen mit N beschrifteten Nachbar hat, kann die Regel R_2 nicht angewendet werden.

Sei nun (G, λ') ein markierter Graph in $Irred_{\mathcal{R}_2}(G, \lambda)$. (G, λ') enthalte nun einen M-Knoten. Da dieser gemäß P_5 auf einem Pfad mit Endpunkt A liegt, kann Regel 2 angewendet werden. Dies ist ein Widerspruch zu Irreduzibilität von (G, λ'). Da ein irreduzibler Graph keine M-Knoten enthalten kann, enthält (G, λ') einen mit A beschrifteten Knoten und alle anderen Knoten sind mit F oder N markiert. Durch die Eigenschaft P_6 muss jeder Nachbar eines F-Knotens wieder ein F Knoten oder der A-Knoten sein. Da der Graph zusammenhängend ist, folgt induktiv, dass keine N-Knoten existieren können. Es sind damit alle Knoten bis auf den A-Knoten mit F beschriftet. Durch die Eigenschaft P_2 sind alle Knoten, die mit A oder F beschriftet sind, zu einer 1-Kante inzident. Da mit P_3 die 1-Kanten einen Baum ergeben und alle Knoten zu einer 1-Kante inzident sind, ergibt sich der gesuchte Spannbaum aus den F-Knoten, dem A-Knoten und den 1-Kanten.

3. Zu Anfang existieren n-1 N-Knoten, da genau ein Knoten mit A beschriftet ist. Regel 1 erniedrigt die Zahl der N-Knoten um eins und erhöht die Zahl der M-Knoten um 1. Da keine Regel die Zahl der N-Knoten erhöht, kann die Regel 1 nur n-1 mal angewendet werden. Da zu Anfang kein M-Knoten existiert und Regel 1 höchstens n-1 mal angewendet werden kann, können höchstens n-1 M Knoten entstehen. Da jede Anwendung der Regel 2 die Anzahl der M-Knoten um eins erniedrigt, kann Regel 2 höchstens n-1 mal angewendet werden. Insgesamt ergibt sich eine Komplexität von $2(|V(G)| - 1)$.
4. Es wurde bereits bewiesen, dass alle Knoten bis auf den A-Knoten in einem irreduziblen Graph F-Knoten sind. Die Rückrichtung gilt ebenfalls, da falls der Graph wie oben beschrieben markiert ist, keine Umbeschriftungsregel mehr angewendet werden kann. □

Satz 5.2 sagt somit aus, das Algorithmus 3.1 ein korrekter Algorithmus zur Berechnung eines Spannbaumes ist. Der Algorithmus hat die Laufzeit $2(|V(G)|-1) \in O(n)$.

5.3 Das Graphumbeschriftungssystem $\mathcal{R}_3$ mit verbotenen Kontexten

Für Algorithmus 3.3 beziehungsweise dessen Formalisierung $\mathcal{R}_3$ kann ein ähnlicher Beweis wie in Abschnitt 5.2 angegeben werden. Dazu können wie oben invariante Eigenschaften benutzt werden, um die Korrektheit des Algorithmus zu beweisen. Die Ergebnisse sind im nachfolgenden Satz zusammengefasst.

Satz 5.3

1. Jede $\mathcal{R}_3$-Umbeschriftungssequenz ist endlich.

2. Sei (G, λ) ein mit I_3 beschrifteter Graph, so dass genau ein Knoten mit A beschriftet ist und (G, λ') ein beliebiger mit L_3 beschrifteter Graph in $Irred_{\mathcal{R}_3}(G, \lambda)$. Dann induziert die Menge der mit 1 beschrifteten Kanten in (G, λ') einen Spannbaum von G.

3. Die Länge einer maximalen $\mathcal{R}_3$-Umbeschriftungskette hat höchstens $2(|V(G)|-1)$ viele Schritte.

4. Für jede $\mathcal{R}_3$-Umbeschriftungskette von (G, λ) zu einem markierten Graph (G, λ'') ist der Graph (G, λ'') irreduzibel genau dann, wenn er einen mit A beschrifteten Knoten enthält, dessen Nachbarn alle mit F beschriftet sind.

Der erste Punkt bedeutet, dass der Algorithmus stets terminiert. Punkt zwei sagt aus, dass der Algorithmus korrekt ist, also für jede zulässige Eingabe einen Spannbaum berechnet. Punkt drei gibt die Komplexität an. In Punkt vier wird die lokale Terminierungserkennung formuliert.

In diesem Kapitel wurde gezeigt, wie invariante Eigenschaften benutzt werden können, um die Korrektheit von Graphumbeschriftungssystemen zu beweisen. Anhand eines irreduziblen Graphen wurde mit den invarianten Eigenschaften eine bestimmte Konfiguration des Graphen hergeleitet, an welcher das Berechnungsergebnis abgelesen werden konnte. Für die Terminierungserkennung wurde die Umkehrrichtung des Korrektheitsbeweises gezeigt: Falls der Graph eine bestimmte Konfiguration hatte, war er irreduzibel. Diese Konfiguration hatte die Eigenschaft, dass sie durch einen ausgezeichneten Knoten lokal erkannt werden konnte. Dieser Knoten war somit in der Lage, die globale Terminierung lokal zu erkennen. Die Komplexität der Algorithmen ergab sich aus der Länge einer maximalen Umbeschriftungskette.

6 Lokale Berechnungen

Graphumbeschriftungssysteme, wie sie in den vorhergehenden Abschnitten vorgestellt worden sind, sind Teil eines generelleren Mechanismus, der *lokale Berechnung* genannt wird.
Wiederholung: $\mathcal{G}_L$ bezeichnet die Klasse der mit L beschrifteten Graphen.

6.1 Definitionen von grundlegenden Begriffen

Im Folgenden werden Begriffe eingeführt, die für die Beschreibung von lokalen Berechnungen notwendig sind.

Definition 6.1 (Transitive Hülle)
Sei $\rightarrow \subseteq R \times R$ eine binäre Relation über einer Menge R. Die transitive Hülle von $\rightarrow$, $\rightarrow^+$, ist die kleinste binäre Relation über R, die unter Transitivität abgeschlossen ist und $\rightarrow$ enthält.

Anschaulich enthält die transitive Hülle einer Relation die Elemente, die durch mehrfaches Anwenden der Relation miteinander in Beziehung stehen.

Definition 6.2 (Graphersetzungsrelation, Ersetzungskette)
Eine *Graphersetzungsrelation* ist eine binäre Relation $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$, die unter Isomorphie abgeschlossen ist. $\mathcal{R}^+$ ist die transitive Hülle von $\mathcal{R}$. Eine $\mathcal{R}$-*Ersetzungskette* ist eine Sequenz $(G_1, \lambda_1), (G_2, \lambda_2), \ldots, (G_n, \lambda_n)$, so dass für jedes i, $1 \leq i \leq n$, gilt: $(G_i, \lambda_i)\mathcal{R}(G_{i+1}, \lambda_{i+1})$.

Unter Isomorphie abgeschlossen heißt, dass falls $(G_1, \lambda_1) \simeq (G, \lambda)$ und $(G, \lambda)\mathcal{R}(G', \lambda')$, dann existiert ein markierter Graph (G'_1, λ'_1), so dass $(G_1, \lambda_1)\mathcal{R}(G'_1, \lambda'_1)$ und $(G'_1, \lambda'_1) \simeq (G', \lambda')$.
Ein Graphumbeschriftungsschritt kann auch als binäre Relation $\rightarrow$ zwischen beschrifteten Graphen gedeutet werden. $(G, \lambda) \rightarrow (G, \lambda')$ gilt genau dann, wenn (G, λ) in einem Berechnungsschritt in (G, λ') überführbar ist. Analog zu einem Umbeschriftungsschritt kann ein Graphersetzungsschritt definiert werden, indem Änderungen des zugrundeliegenden Graphen zugelassen werden. Solch ein Graphersetzungsschritt lässt sich wiederum durch eine Graphersetzungsrelation ausdrücken. Zwei markierte Graphen (G, λ) und (G', λ') sind in der Graphersetzungsrelation, falls (G, λ) in einem Graphersetzungsschritt in (G', λ') überführt werden kann. Damit ist eine Graphersetzungsrelation eine Erweiterung einer Graphumbeschriftungsrelation, da in einem Schritt die zugrundeliegenden Graphen geändert werden können. Eine Ersetzungskette ist eine Hintereinanderreihung von markierten Graphen, die miteinander in der Graphersetzungsrelation stehen.

Definition 6.3 (k-lokal, Normalform)
Sei $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ eine Graphersetzungsrelation und $k \in \mathbb{N} \setminus \{0\}$.

1. $\mathcal{R}$ ist eine Graphumbeschriftungsrelation, falls gilt:

$$(G, \lambda)\mathcal{R}(H, \lambda') \Longrightarrow G = H$$

2. Es gelte $(G, \lambda)\mathcal{R}(G, \lambda')$. Dann heißt die Umbeschriftungsrelation $\mathcal{R}$ *k-lokal*, falls sich λ und λ' nur in einer Umgebung der Größe k unterscheiden. Formal:

$$\exists v \in V(G), \text{ so dass } \forall x \notin V(B_G(v, k)) \cup E(B_G(v, k)), \lambda(x) = \lambda'(x)$$

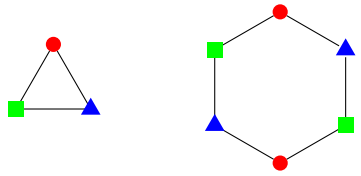
Die Relation $\mathcal{R}$ ist *lokal*, falls sie k-lokal ist für ein $k > 0$.

3. Eine $\mathcal{R}$-*Normalform* von $(G, \lambda) \in \mathcal{G}_L$ ist ein Graph (G, λ'), so dass $(G, \lambda)\mathcal{R}^+(G, \lambda')$ und es existiert kein markierter Graph (G, λ''), so dass $(G, \lambda')\mathcal{R}(G, \lambda'')$. Ein markierter Graph (G, λ) hat eine $\mathcal{R}$-Normalform, falls jede $\mathcal{R}$-Umbeschriftungssequenz endlich ist.

Punkt 1 in der Definition stellt die Beziehung zwischen einer Graphumbeschriftungsrelation und einer Graphersetzungsrelation her. Eine Graphersetzungsrelation ist eine Graphumbeschriftungsrelation, falls die beiden zugrundeliegenden Graphen gleich sind. k-lokal bedeutet, dass die Graphumbeschriftungsrelation nur innerhalb einer k-Umgebung Auswirkungen hat. Eine $\mathcal{R}$-Normalform eines Graphen (G, λ) ist dann erreicht, falls der Graph nicht mehr umbeschriftet werden kann, also irreduzibel ist.

6.2 Verteilte Berechnungen von lokalen Berechnungen

Der oben eingeführte Begriff der Umbeschriftungssequenz impliziert eine *sequentielle* Berechnung. Eine k-lokale Umbeschriftungsrelation hängt nur von einem durch k beschränkten Teilgraphen ab. Falls eine k-lokale Umbeschriftungsrelation und zwei disjunkte k-Umgebungen vorliegen, kann in beiden Teilgraphen in beliebiger Reihenfolge oder gleichzeitig ersetzt werden. Dadurch entsteht eine verteilte Berechnung. Die beiden Umbeschriftungsschritte werden in diesem Fall *kommutativ* genannt. Dies lässt sich auch auf Berechnungssequenzen anwenden: Zwei Berechnungssequenzen, von denen sich die Zweite durch eine Folge von Vertauschungen von kommutativen Berechnungsschritten aus der Ersten erzeugen lässt, führen zum gleichen markierten Graphen. Deswegen kann der Begriff der Graphumbeschriftungssequenz als *Serialisierung* einer verteilten Berechnung betrachtet werden. Dieses Modell ist asynchron: Mehrere Umbeschriftungsschritte *können* zur selben Zeit ausgeführt werden, müssen es aber nicht. Im Nachfolgenden wird immer von Sequenzen die Rede sein, auch wenn kommutative (Teil-) Sequenzen parallel berechnet werden können.



Graph H Graph G

Abbildung 14: G ist eine Überdeckung von H

7 Überdeckungen und k-Überdeckungen

In diesem Kapitel wird der Begriff *Überdeckung* definiert, der in den folgenden Kapiteln eine wichtige Rolle in Sätzen und Beweisen spielen wird. Im ersten Abschnitt werden *Überdeckungen* und einige ihrer Grundeigenschaften vorgestellt. Der Begriff der Überdeckung wird im zweiten Abschnitt auf *k-Überdeckung* eingeschränkt, um ihn für lokale Berechnungen benutzen zu können. Im dritten Abschnitt wird gezeigt, in welcher Beziehung lokale Berechnungen und *k-Überdeckung* stehen.

7.1 Überdeckungen

In diesem Abschnitt werden *Überdeckungen* definiert und ein Algorithmus angegeben, mit dessen Hilfe eine *Überdeckung* konstruiert werden kann.

7.1.1 Definition und Eigenschaften einer Überdeckung

Ein Graph G ist eine *Überdeckung* eines Graphen H, wenn eine Abbildung γ die Knoten des Graphen G surjektiv auf die des Graphen H abbildet und dabei deren Nachbarschaft erhalten bleibt. Das bedeutet, dass die Nachbarschaft jedes Knoten $v \in V(G)$ bijektiv auf die Nachbarschaft des Knoten $\gamma(v) \in V(H)$ abgebildet wird.

Abbildung 14 zeigt ein Beispiel für eine Überdeckung. Hier wird die Abbildung γ durch gleiche Symbole verdeutlicht; die ●-Knoten des Graphen G werden auf den ●-Knoten des Graphen H abgebildet, entsprechend die ■- und ▲-Knoten. Nun hat im Graphen G jeder ●-Knoten einen ■- und einen ▲-Nachbarn, jeder ▲-Knoten einen ●- und einen ■-Nachbarn sowie jeder ■-Knoten einen ▲- und einen ●-Nachbarn. Das gleiche gilt im Graphen H, somit ist G eine *Überdeckung* von H.

Formal ist eine *Überdeckung* wie folgt definiert:

Definition 7.1 (Überdeckungen)
Ein Graph G ist eine *Überdeckung* eines Graphen H, falls eine surjektive Abbildung $\gamma : G \longrightarrow H$ existiert, so dass für jeden Knoten $v \in V(G)$ die Restriktion von γ auf $N_G(v)$ eine Bijektion auf $N_H(\gamma(v))$ ist.
Eine Überdeckung heißt *strikt*, falls G und H nicht isomorph sind.

Die Knotenmenge von G ist meist mächtiger als die Knotenmenge von H. Daher ist die Abbildung γ nicht immer injektiv. Damit die Forderung gestellt werden kann, dass die Nachbarschaften surjektiv auf einander abgebildet werden, muss γ auf die Nachbarschaft von v eingeschränkt werden, so dass unter der Restriktion γ eine Bijektion zwischen den Nachbarschaften $N_G(v)$ und $N_H(\gamma(v))$ ist und somit die Nachbarschaft „erhalten“ bleibt.

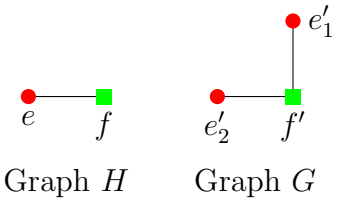


Abbildung 15: G ist keine Überdeckung von H

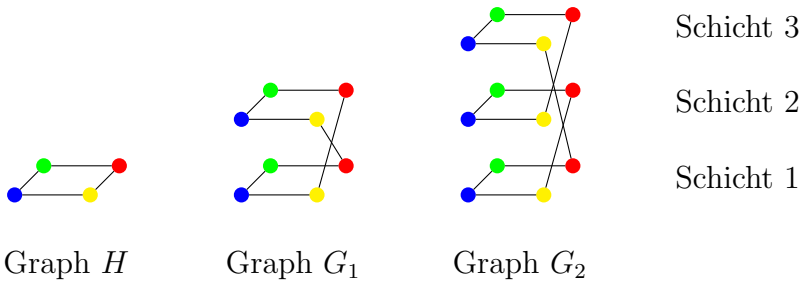


Abbildung 16: Räumliche Darstellung von Überdeckungen

Aus der Definition 7.1 folgt, dass, wenn ein Knoten $v \in V(G)$ zwei verschiedene Nachbarn v_1 und v_2 hat, auch $\gamma(v_1)$ und $\gamma(v_2)$ zwei verschiedene Knoten in H sein müssen.

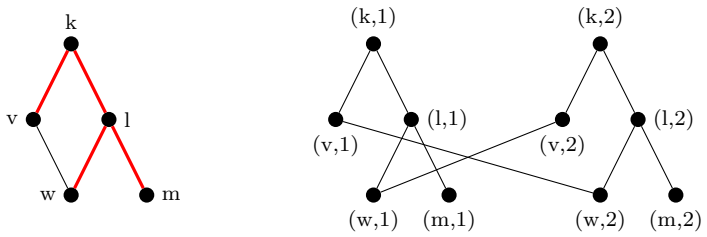
Wenn G eine Überdeckung von H ist und zwei Knoten $e'_1, e'_2 \in V(G)$ auf den gleichen Knoten $e \in V(H)$ abgebildet werden, haben sie keine gemeinsamen Nachbarn f' und somit einen Abstand größer als 2. Abbildung 15 verdeutlicht diese Aussage. Die Nachbarschaften der Knoten e'_1 und e'_2 werden jeweils bijektiv abgebildet. Die Nachbarschaft des Knoten f' wird aber nicht injektiv auf die Nachbarschaft von f abgebildet, da auf den Knoten e zwei Knoten abgebildet werden. Der Graph G ist somit keine Überdeckung des Graphen H.

Satz 7.2
Sei G eine Überdeckung von H durch γ und $v_1, v_2 \in V(G)$, $v_1 \neq v_2$. Falls $\gamma(v_1) = \gamma(v_2)$, dann gilt: $N_G(v_1) \cap N_G(v_2) = \emptyset$ und damit $d(v_1, v_2) > 2$.

Sei G eine Überdeckung eines Graphen H. Dann gilt, dass G genau n mal so viele Knoten hat wie H für ein $n \in \mathbb{N}$, und dass auf jeden Knoten von H genau n Knoten von G abgebildet werden. Dies verdeutlicht die räumliche Darstellung von Überdeckungen in Abbildung 16. Der Graph G_1 ist eine Überdeckung von H und hat acht Knoten. Jeweils zwei werden auf die vier Knoten des Graphen H abgebildet.

Satz 7.3
Sei H ein zusammenhängender Graph und sei G eine Überdeckung von H durch γ. Dann existiert ein $q \in \mathbb{N}$, so dass für alle $v \in V(G)$ gilt: $Card(\gamma^{-1}(v)) = q$.

Die Zahl q wird die Anzahl der *Schichten* einer Überdeckung von G genannt. G ist dann eine *q-schichtige Überdeckung* von H. Für $q = 1$ sind G und H isomorph. In Abbildung 16 ist der Graph G_1 eine 2-schichtige Überdeckung von H, der Graph G_2 ist 3-schichtig.



Graph H Graph $G_{T,\Sigma}$

$q = 2$ $\sigma_{\{v,w\}}(1) = 2$ $\sigma_{\{v,w\}}(2) = 1$ $\Sigma = \{\sigma_{\{v,w\}}\}$

Abbildung 17: $G_{T,\Sigma}$ ist eine 2-schichtige Überdeckung von H

7.1.2 Konstruktion von Überdeckungen

In diesem Abschnitt wird ein Algorithmus vorgestellt, der q-schichtige Überdeckungen zu einem Graphen H erzeugt.

Zur Erzeugung einer 2-schichtigen Überdeckung des Graphen H in Abbildung 17 arbeitet der Algorithmus wie folgt. Zunächst wird ein Spannbaum des Graphen H bestimmt. In der Abbildung ist dieser rot eingefärbt. Dann werden für jeden Knoten von H zwei Knoten, für jede Schicht einer, erzeugt, für den Knoten k also beispielsweise $(k, 1)$ und $(k, 2)$. Dann werden die Kanten des Spannbaumes in die Überdeckung übertragen, die Kante $\{k, l\}$ von H gibt es also zwei mal in der Überdeckung, nämlich $\{(k, 1)(l, 1)\}$ und $\{(k, 2)(l, 2)\}$. Zuletzt wird die Kante $\{v, w\}$, die nicht im Spannbaum enthalten ist zwei mal so in die Überdeckung eingefügt, dass sie die Schichten verbindet, nämlich als $\{(w, 1)(v, 2)\}$ und $\{(w, 2)(v, 1)\}$. Der so erzeugte Graph ist eine Überdeckung von H.

Definition 7.4 (Algorithmus zur Erzeugung von Überdeckungen)
Soll für einen Graphen H eine q-schichtige Überdeckung konstruiert werden, so wird zunächst ein Spannbaum T des Graphen bestimmt. Danach wird für jede Kante e, die nicht im Baum T enthalten ist, eine Permutation σ_e auf dem Intervall $[1...q]$ definiert. Sei Σ die Vereinigung dieser Permutationen. Die Überdeckung von H ist nun abhängig von T und Σ und wird daher mit $G_{T,\Sigma}$ bezeichnet. Nun wird für jeden Knoten $x \in V(H)$ in jeder Schicht der Überdeckung $G_{T,\Sigma}$ ein entsprechender Knoten erzeugt, in dem die Tupel $(x, i),\ i \in [1, q]$ gebildet werden, also gilt: $V(G_{T,\Sigma}) = \{(x, i) \mid x \in V(H), i \in [1, q]\}$. Die Kanten von $G_{T,\Sigma}$ sind zum einen Kopien der Kanten des Baumes T, also $E_1(G_{T,\Sigma}) = \{\{(x, i), (y, i)\} \mid \{x, y\} \in E(T), i \in [1, q]\}$, zum anderen die Kanten, die die einzelnen Schichten der Überdeckung verbinden. Diese entsprechen den Kanten $e \in E(H)$, die nicht im Baum T liegen. Liegt ein Knoten der Kante e in Schicht i, dann ist der andere Knoten in der Schicht $\sigma_e(i)$, es gilt: $E(G_{T,\Sigma}) = E_1(G_{T,\Sigma}) \cup \{\{(x, i), (y, \sigma_{\{x,y\}}(i))\} \mid \{x, y\} \in E(H) \backslash E(T), i \in [1, q]\}$.

Diese Konstruktion zeigt, dass es nur dann zusammenhängende Überdeckungen für einen Graphen H gibt, wenn H mindestens einen Kreis enthält, da sonst die Menge der im letzten Schritt gebildeten Kanten leer ist. Diese Kanten sind genau die Kanten, die die Schichten verbinden. Sind sie nicht vorhanden, ist der Graph

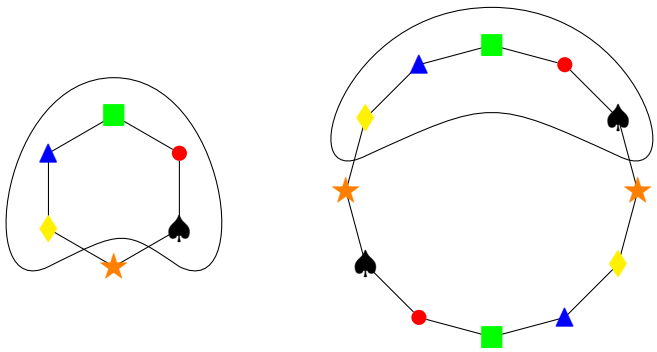


Abbildung 18: G ist eine 2-Überdeckung von H

nicht zusammenhängend. Jede mögliche Überdeckung eines Graphen H ist zu einer so konstruierten Überdeckung isomorph.

Satz 7.5 (Reidemeister)
Sei H ein Graph und T ein Spannbaum von H. Ein Graph G ist eine Überdeckung von H genau dann, wenn ein $q \in \mathbb{N}\setminus\{0\}$ und eine Menge $\Sigma = \{\sigma_e | e \in E(H)\setminus E(T)\}$ von Permutationen auf $[1,q]$ existiert, so dass G zu dem Graph $G_{T,\Sigma}$ isomorph ist, der wie folgt definiert ist:

$$\begin{aligned} V(G_{T,\Sigma}) &= \{(x,i) | x \in V(H), i \in [1,q]\}, \\ E(G_{T,\Sigma}) &= \{\{(x,i),(y,i)\} | \{x,y\} \in E(T), i \in [1,q]\} \cup \\ &\quad \{\{(x,i),(y,\sigma_{\{x,y\}}(i))\} | \{x,y\} \in E(H)\setminus E(T), i \in [1,q]\} \end{aligned}$$

Die Definition des Graphen $G_{T,\Sigma}$ entspricht genau dem vorgestellten Algorithmus.

7.2 k-Überdeckungen

In diesem Abschnitt werden Überdeckungen auf k-Überdeckungen eingeschränkt, indem gefordert wird, dass γ nicht nur Nachbarschaften bijektiv, sondern k-Umgebungen isomorph aufeinander abbildet.

Abbildung 18 zeigt ein Beispiel für eine 2-Überdeckung. Die 2-Umgebung des ■-Knoten im Graphen H ist isomorph zu den 2-Umgebungen der ■-Knoten im Graphen G. Die formale Definition für k-Überdeckungen lautet:

Definition 7.6 (k-Überdeckung)
Seien G und H zwei Graphen, γ ein surjektiver Homomorphismus von G nach H und $k \in \mathbb{N}\setminus\{0\}$. G ist eine *k-Überdeckung* von H durch γ, falls für alle $v \in V(G)$ die Restriktion von γ auf $B_G(v,k)$ ein Isomorphismus zwischen $B_G(v,k)$ und $B_H(v,k)$ ist. Eine Überdeckung heißt *strikt*, falls G und H nicht isomorph sind.

Satz 7.2 kann auch auf k-Überdeckungen angewendet werden: Wenn zwei Knoten von G auf den gleichen Knoten in H abgebildet werden, haben sie keine gemeinsamen Knoten in ihren k-Umgebungen und somit einen Abstand größer als $2k$:

Satz 7.7
Sei G eine k-Überdeckung von H durch γ und seien $v_1, v_2 \in V(G)$, $v_1 \neq v_2$. Falls $\gamma(v_1) = \gamma(v_2)$, dann gilt: $B_G(v_1,k) \cap B_G(v_2,k) = \emptyset$ und damit $d(v_1,v_2) > 2k$.

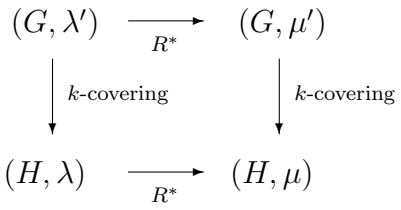


Abbildung 19: Zusammenhang zwischen Überdeckungen und lokalen Berechnungen.

Aus den Definitionen folgt, dass jede k-Überdeckung eines Graphen H auch eine Überdeckung von H ist. Die umgekehrte Richtung gilt aber nicht: Eine Umgebung mit Radius k hat einen Durchmesser von $2k+1$. Damit muss jeder Teilgraph T von G mit einem Durchmesser von höchstens $2k+1$ auf einen isomorphen Teilgraphen von H abgebildet werden. Eine solche Überdeckung G kann aber nur konstruiert werden, wenn H mindestens einen Kreis der Länge $l > 2k+1$ besitzt.

Satz 7.8
Seien G und H zwei Graphen, γ ein surjektiver Homomorphismus von G nach H und $k \in \mathbb{N} \setminus \{0\}$. Dann ist G eine k-Überdeckung von H durch γ genau dann, wenn G eine Überdeckung von H durch γ ist, und für jeden Kreis C der Länge $l \leq 2k+1$ in H das Urbild $\gamma^{-1}(C)$ eine disjunkte Vereinigung von zu C isomorphen Kreisen ist.

7.3 Lokale Berechnungen und k-Überdeckungen

Erzeugt eine Umbeschriftungs-Sequenz für einen Graphen (H, λ) einen Graphen (H, μ) und für eine k-Überdeckung (G, λ') von (H, λ) einen Graphen (G, μ'), so ist (G, μ') eine k-Überdeckung von (H, μ). Dieser Zusammenhang ist in Abbildung 19 graphisch dargestellt. Formal wird dieser Zusammenhang in Satz 7.9 ausgedrückt:

Satz 7.9
Sei $\mathcal{R}$ eine k-lokal generierte Umbeschriftungsrelation und sei (G, λ') eine k-Überdeckung von (H, λ) durch eine Abbildung γ. Sei (H, μ) ein markierter Graph, so dass gilt: $(H, \lambda)\mathcal{R}^*(H, \mu)$. Dann existiert eine Beschriftung μ von G, so dass $(G, \lambda')\mathcal{R}^*(G, \mu')$ gilt und (G, μ') eine k-Überdeckung von (H, μ) ist.

Das bedeutet, dass jede Berechnungs-Sequenz auf (H, λ) auch auf (G, λ') übertragen werden kann, indem jeder Berechnungs-Schritt auf einem Teilgraphen T von H auch auf allen Teilgraphen $\gamma^{-1}(T)$ von G durchgeführt wird. Der resultierende Graph (G, μ') ist dann eine Überdeckung eines Graphen (H, μ).

8 Das Auswahlproblem

Das Ziel einer *Auswahl* in einem Graphen ist es, genau einen der Knoten des Graphen zu markieren. Dieser Knoten heißt *ausgewählt* oder auch *Leiter* des Graphen. Dabei geht es nicht darum, dass der Algorithmus in jedem Durchlauf denselben Konten auswählt, sondern darum, dass die Beschriftung jedes Knotens diesen eindeutig als ausgewählt oder als nicht ausgewählt markiert.

Im ersten Abschnitt werden Auswahl-Algorithmen eingeführt und definiert, der zweite Abschnitt dieses Kapitels enthält zwei Beispiele für Auswahl-Algorithmen. Im dritten Abschnitt wird betrachtet, unter welcher Voraussetzungn in einer Klasse von Graphen eine Auswahl möglich ist.

8.1 Einführung und Definition

Sei L eine Menge von Beschriftungen. Alle markierten Graphen dieses Abschnittes seien L-beschriftet. $\mathcal{G}_L$ bezeichnet die Menge aller L-beschrifteten Graphen.

Ein *Auswahl-Algorithmus* ist ein Umbeschriftungs-Algorithmus zusammen mit einer *auswählenden* Markierung. Nach der Terminierung des Umbeschriftungs-Algorithmus gibt es genau einen mit der *auswählenden* Markierung beschrifteten Knoten; dieser Knoten ist der durch den Algorithmus ausgewählte Knoten. Formal ist ein *Auswahl-Algorithmus* wie folgt definiert.

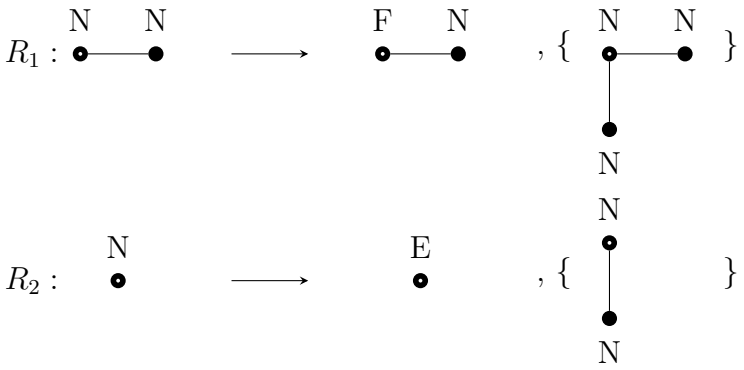
Definition 8.1 (Auswahl-Algorithmus)
Sei $\mathcal{R}$ eine Umbeschriftungs-Relation und $\ell \in L$ eine Markierung. Ein Tupel $(\mathcal{R}, \ell)$ ist ein Auswahl-Algorithmus für eine Klasse $\mathcal{C}$ von markierten Graphen, wenn für jeden markierten Graphen $(H', \lambda') \in Irred((H, \lambda) \in \mathcal{C})$ gilt: $|\lambda'^{-1}(\ell)| = 1$. Der mit ℓ markierte Knoten $\lambda'^{-1}(\ell)$ ist der ausgewählte Knoten, die Markierung ℓ heißt *auswählende* Markierung.

8.2 Beispiele

In diesem Abschnitt werden zwei Algorithmen für die Auswahl in Graphen vorgestellt. Der erste Algorithmus wählt einen Knoten in einem Baum aus, der zweite in einem vollständigen Graphen.

Beispiel 8.2
Das folgende FCGRS wählt in einem Baum genau einen Knoten aus:

Sei $\mathcal{R} = (L, I, P)$ ein FCGRS mit $L = \{N, F, E\}$, $I = \{N\}$ und $P = \{R_1, R_2\}$. Die auswählende Markierung sei E. Seien R_1 und R_2 definiert als die folgenden Umbeschriftungsregeln mit verbotenem Kontext:



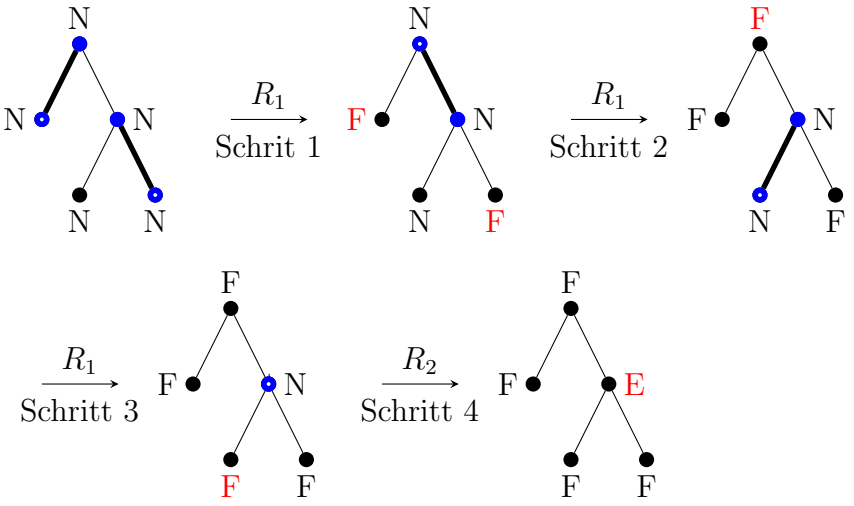


Abbildung 20: Auswahl in einem Baum

Aufgrund der Regel R_1 wird nach und nach bei den Blättern des Baumes beginnend die Beschriftung aller Knoten bis auf eines von N in F geändert. Die Regel R_2 dient dazu, diesen letzten Konten mit E als ausgewählt zu markieren.

Ein Beispiel für die Berechnung auf einem einfachen Baum ist in Abbildung 20 zu sehen. In Schritt 1 kann die Regel 1 an zwei Blättern des Baumes parallel durchgeführt werden. In Schritt 2 wird der Wurzelknoten durch die Regel 1 mit F beschriftet, in Schritt 3 das verbliebene Blatt. Da nun keine zwei adjazenten Knoten mit N beschriftet sind, wählt die Regel 2 im letzten Schritt den Median des Graphen aus.

Median bezeichnet hier einen Knoten, bei dem die Summe der Entfernungen zu allen anderen Konten des Graphen minimal ist. Für einen Knoten v ist die Summe der Entfernungen zu allen anderen Knoten definiert durch $sum(v) = \sum_{u \in V} d(v, u)$.

Für den Graphen in Abbildung 21 gilt:

$$
\begin{aligned}
sum(v_a) &= d(v_a, v_a) + d(v_a, v_b) + d(v_a, v_c) + d(v_a, v_d) + d(v_a, v_e) \\
&= \quad 0 \quad + \quad 1 \quad + \quad 2 \quad + \quad 1 \quad + \quad 2 \\
&= 6 \\
sum(v_b) &= d(v_b, v_a) + d(v_b, v_b) + d(v_b, v_c) + d(v_b, v_d) + d(v_b, v_e) \\
&= \quad 1 \quad + \quad 0 \quad + \quad 3 \quad + \quad 2 \quad + \quad 3 \\
&= 9 \\
sum(v_c) &= d(v_c, v_a) + d(v_c, v_b) + d(v_c, v_c) + d(v_c, v_d) + d(v_c, v_e) \\
&= \quad 2 \quad + \quad 3 \quad + \quad 0 \quad + \quad 1 \quad + \quad 2 \\
&= 8 \\
sum(v_d) &= d(v_d, v_a) + d(v_d, v_b) + d(v_d, v_c) + d(v_d, v_d) + d(v_d, v_e) \\
&= \quad 1 \quad + \quad 2 \quad + \quad 1 \quad + \quad 0 \quad + \quad 1 \\
&= 5 \\
sum(v_e) &= d(v_e, v_a) + d(v_e, v_b) + d(v_e, v_c) + d(v_e, v_d) + d(v_e, v_e) \\
&= \quad 2 \quad + \quad 3 \quad + \quad 2 \quad + \quad 1 \quad + \quad 0 \\
&= 8
\end{aligned}
$$

$sum(v_d)$ ist minimal und damit ist v_d der Median des Graphen.

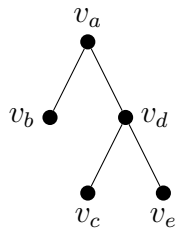


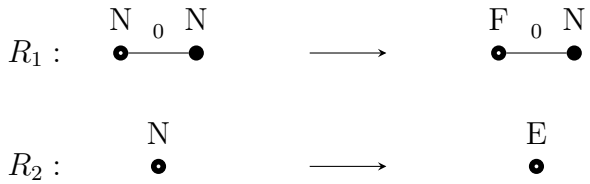
Abbildung 21: Graph G

Durch diesen Algorithmus kann jeder Knoten eines Baumes ausgewählt werden, allerdings werden die Mediane des Graphen mit der größten Wahrscheinlichkeit ausgewählt (siehe [3]).

Beispiel 8.3
In einem vollständigen Graphen ist jeder Knoten mit allen übrigen Knoten durch eine Kante verbunden.

Das folgende PGRS wählt in einem vollständigen Graphen genau einen Knoten aus:

Sei $\mathcal{R} = (L, I, P)$ ein PGRS mit $L = \{N, F, E\}$, $I = \{N\}$ und $P = \{R_1, R_2\}$. Sei E die auswählende Markierung und seien R_1 und R_2 definiert als die folgenden Umbeschriftungsregeln mit der Prioritätsrelation $R_1 > R_2$:



Da alle Knoten miteinander verbunden sind, beschriftet die Regel R_1 einen Knoten mit F, solange mindestens zwei mit N beschriftete Knoten im Graphen vorhanden sind. Gibt es keine zwei mit N beschriftete Knoten, wird durch die Regel R_2 der verbliebene Knoten mit E beschriftet. Dieser Knoten ist der ausgewählte Knoten.

Abbildung 22 zeigt eine Beispiel-Berechnung in dem vollständigen Graphen mit 5 Knoten. In Schritt 1 werden zwei Knoten parallel durch die Regel 1 umbeschriftet. In Schritt 2 und 3 wird die Markierung zweier weiterer Knoten mit Regel 1 in F geändert. Da nun Regel 1 nicht mehr angewendet werden kann, wird schließlich der verbliebene Knoten mit E beschriftet und ist damit ausgewählt.

Der Algorithmus wählt jeden Knoten eines vollständigen Graphen mit der gleichen Wahrscheinlichkeit aus.

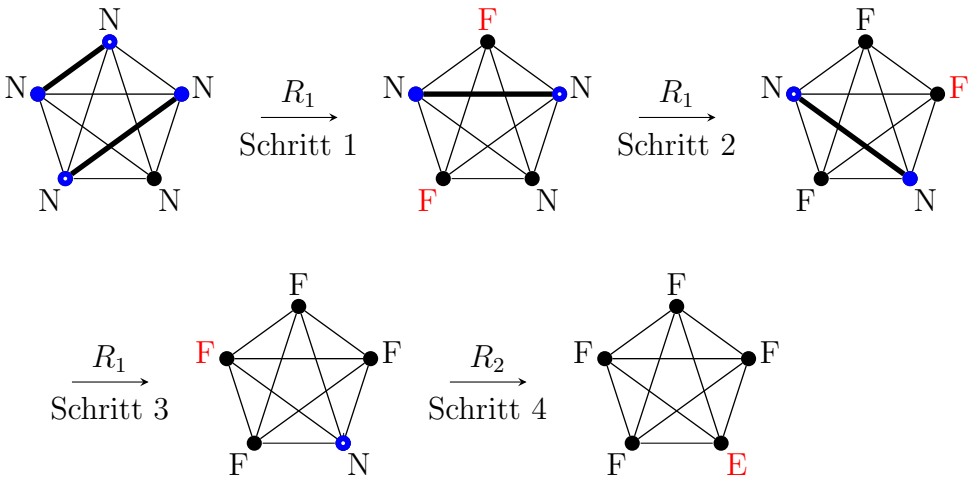


Abbildung 22: Auswahl eines Knotens in einem vollständigen Graph

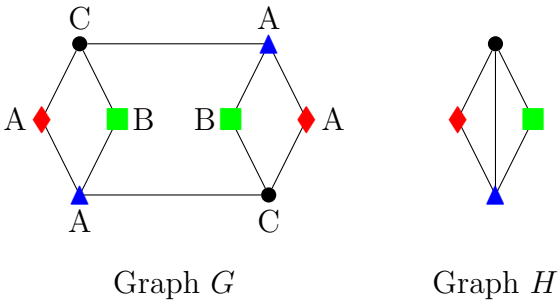


Abbildung 23: Der Graph G ist symmetrisch, er ist eine 2-Überdeckung von H

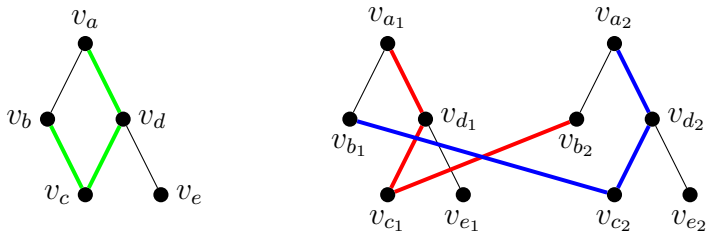
8.3 Voraussetzung für eine Auswahl

In diesem Abschnitt wird gezeigt, welche Voraussetzungen erfüllt sein müssen, damit für einen Graph bzw. eine Klasse von Graphen ein Auswahl-Algorithmus existiert. Zunächst werden einige Begriffe benötigt, um diese Voraussetzungen präzise formulieren zu können.

Definition 8.4 (Mehrdeutigkeit und Symmetrie)
Ein Graph G heißt *mehrdeutig*, wenn er eine Überdeckung eines Graphen H mit einer Abbildung γ ist. Die Markierungs-Funktion λ eines mehrdeutigen Graphen G ist *symmetrisch*, wenn für zwei Knoten $u, v \in V(G)$ mit $\gamma(u) = \gamma(v)$ gilt $\lambda(u) = \lambda(v)$. Ein markierter Graph (G, λ) ist *symmetrisch* markiert, wenn sowohl G mehrdeutig als auch λ symmetrisch ist.

Abbildung 23 zeigt ein Beispiel für einen symmetrisch markierten Graphen G. Zur Verdeutlichung ist auch der Graph H angegeben, den G überdeckt. Da die •-Knoten des Graphen G auf •-Knoten des Graphen H abgebildet werden, haben sie die gleiche Beschriftung. Das selbe gilt für die ■-, ▲- und ♦-Knoten. Markierungen des Graphen H spielen für die Symmetrie von G keine Rolle.

Definition 8.5 (γ-freier Teilgraph)
Sei H ein Graph und G ein mehrdeutiger Graph, der H mit der Abbildung γ überdeckt. Ein Teilgraph K von G ist *frei modulo* γ, oder einfach γ-*frei*, wenn $\gamma^{-1}(\gamma(K))$ eine Menge von disjunkten Teilgraphen von G ist, die alle zu K isomorph sind.



Graph H Graph G

Abbildung 24: der rot markierte Teilgraph ist γ-frei

Abbildung 24 zeigt ein Beispiel für einen γ-freien Teilgraphen einer Überdeckung. Sei K der von $v_{a_1}, v_{d_1}, v_{c_1}, v_{b_2}$ induzierte, rot markierte Teilgraph von G. Dann ist $\gamma(K)$ der durch v_a, v_d, v_c, v_b induzierte, grün markierte Teilgraph K_H in H. $\gamma^{-1}(\gamma(K))$ ist der Graph K sowie der von $v_{a_2}, v_{d_2}, v_{c_2}, v_{b_1}$ induzierte, blau markierte Teilgraph K'. Da $V(K)$ und $V(K')$ disjunkt sind, ist K γ-frei.

Definition 8.6 (γ-freier Berechnungs-Schritt)
Ein *γ-freier* Berechnungs-Schritt eines Algorithmus ist ein Berechnungs-Schritt, der lediglich Kenntnis eines γ-freien Teilgraphen benötigt, um die Beschriftungen zu verändern, und der auch nur Beschriftungen dieses Teilgraphen verändert.

Ein γ-freier Berechnungs-Schritt, der auf einem Teilgraphen K eines symmetrisch beschrifteten Graphen (G, λ) arbeitet, kann in G q-mal parallel auf den isomorphen Teilgraphen $\gamma^{-1}(\gamma(K))$ durchgeführt werden, da die Menge $\gamma^{-1}(\gamma(K))$ disjunkt ist. Das bedeutet, dass kein γ-freier Berechnungs-Schritt genau einen Knoten im symmetrischen Graphen (G, λ) markieren kann.

Abbildung 25 demonstriert dies anhand einer 2-schichtigen Überdeckung des symmetrisch beschrifteten, vollständigen Graphen mit 5 Knoten mit dem Algorithmus aus Beispiel 8.3. Im ersten Schritt wird die Regel 1 vier mal parallel angewendet, im zweiten und dritten Schritt zwei mal parallel. Im letzten Schritt markiert die Regel 2 zwei Knoten parallel mit E als ausgewählt, was ein Widerspruch zu der Forderung ist, dass genau ein Knoten ausgewählt werden soll. Da der Graph symmetrisch ist und die Berechnungs-Schritte des Algorithmus γ-frei sind, ist keine Auswahl im Graphen möglich.

Satz 8.7
Sei (G, λ) ein symmetrisch beschrifteter Graph. Dann gibt es keinen γ-freien Algorithmus, der das Auswahl-Problem für diesen Graphen löst.

Da nach Satz 8.7 keine Auswahl in einem symmetrisch markierten Graphen durchgeführt werden kann, kann auch keine Auswahl in einer Klasse von Graphen durchgeführt werden, die einen symmetrisch markierten Graphen enthält.

Satz 8.8
Es gibt keinen γ-freien Auswahl-Algorithmus für eine Klasse von Graphen, die einen symmetrisch beschrifteten Graphen enthält.

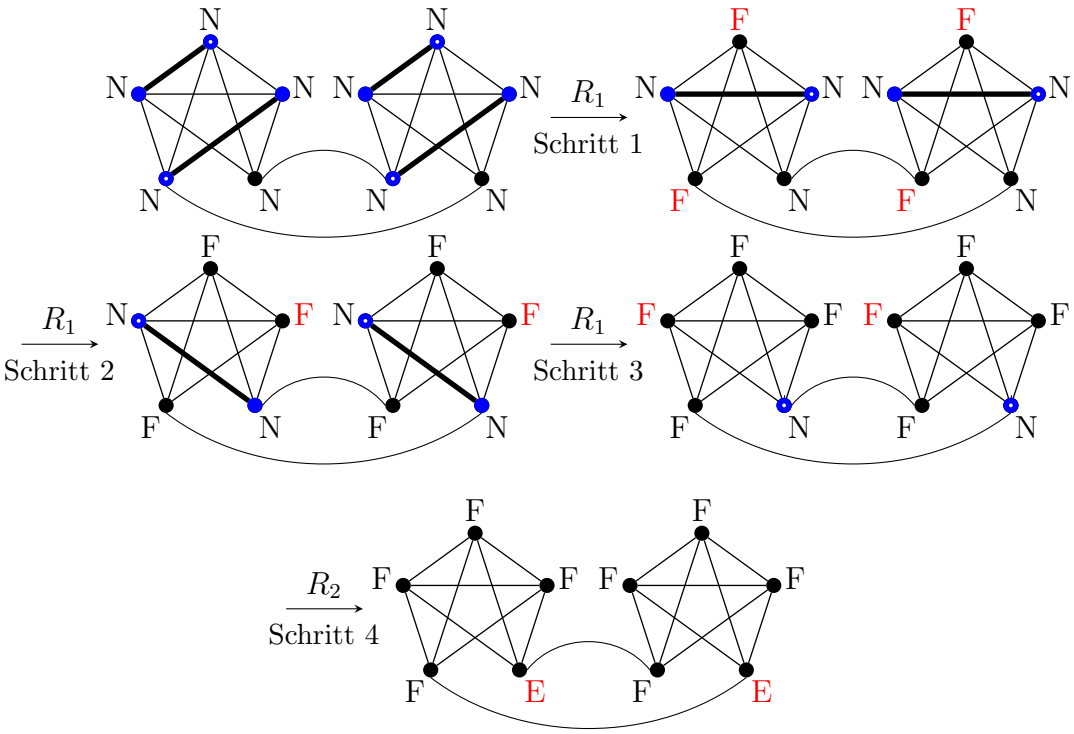


Abbildung 25: Auswahl eines Knotens in einer Überdeckung

Soll eine Auswahl auf einem mehrdeutigen Graphen G durchgeführt werden, der nicht symmetrisch beschriftet ist, so muss sicher gestellt sein, dass während der Berechnung keine symmetrische Beschriftung auftritt, da sonst wieder Satz 8.7 angewendet werden kann und eine Auswahl nicht möglich wäre.

Ein Auswahl-Algorithmus für eine Klasse von markierten Graphen $\mathcal{C}$ ist ein Algorithmus, der in jedem Graphen der Klasse $\mathcal{C}$ genau einen Knoten auswählt. Eine Auswahl kann genau dann getroffen werden, wenn der Graph, auf dem der Algorithmus arbeitet, in keiner Konfiguration symmetrisch beschriftet ist.

9 Das Erkennungsproblem

Ein *Erkennungs-Algorithmus* stellt fest, ob ein Graph G zu einer Klasse $\mathcal{C}$ von Graphen gehört, er „erkennt“ die Klasse.

Im ersten Abschnitt werden Erkennungs-Algorithmen eingeführt und definiert, im zweiten werden zwei Beispiele für die Erkennung von Graph-Klassen angegeben. Im letzten Abschnitt wird erläutert, welche Voraussetzungen eine Klasse von Graphen besitzen muss, damit sie erkannt werden kann.

9.1 Einführung und Definition

Sei L ein Menge von Beschriftungen. Alle markierten Graphen dieses Abschnittes seien L-beschriftet. $\mathcal{G}_L$ bezeichnet die Menge aller L-markierten Graphen.

Ein *Erkennungs-Algorithmus* ist ein Umbeschriftungs-Algorithmus zusammen mit einer Menge von *terminalen Graphen*. Ist nach der Terminierung des Umbeschriftungs-Algorithmus der erhaltene Graph ein Element der Menge der *terminalen Graphen*, so ist der Graph Element der Klasse. Formal ist ein *Erkennungs-Algorithmus* wie folgt definiert.

Definition 9.1 (Erkennungs-Algorithmus)
Ein *Erkennungs-Algorithmus* (GR: graph recognizer) ist ein Paar $(\mathcal{R}, \mathcal{K})$, wobei $\mathcal{R}$ eine Umbeschriftungs-Relation und $\mathcal{K}$ eine Klasse von markierten Graphen ist, deren Elemente *terminale Graphen* genannt werden. $L(\mathcal{R}, \mathcal{K})$ bezeichnet die Klasse der markierten Graphen, die von $(\mathcal{R}, \mathcal{K})$ erkannt werden, und ist definiert durch $L(\mathcal{R}, \mathcal{K}) = \{(G, \lambda) \in \mathcal{G}_L : Irred_{\mathcal{R}}((G, \lambda)) \cap \mathcal{K} \neq \emptyset\}$.

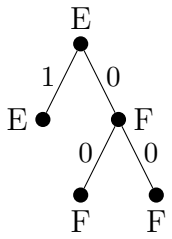
Ein GR ist *deterministisch*, wenn er auf jedem Graphen terminiert und wenn zwei verschiedene Berechnungs-Sequenzen nicht einmal zu einem terminalen Graphen führen und einmal nicht.

Definition 9.2 (Determinismus)
Ein GR $(\mathcal{R}, \mathcal{K})$ ist *deterministisch*, wenn $\mathcal{R}$ auf jedem Graphen G terminiert ist und für jeden Graph G entweder $Irred_{\mathcal{R}}(G) \subseteq \mathcal{K}$ gilt, oder $Irred_{\mathcal{R}}(G) \cap \mathcal{K} = \emptyset$.

Zu einem GR für markierte Graphen gehört zusätzlich zu der Umbeschriftungs-Relation und den terminalen Graphen noch eine Anfangsbeschriftung l_0, mit der alle Knoten und Kanten markiert werden.

Definition 9.3 (GR für unmarkierte Graphen)
Ein GR für unmarkierte Graphen ist als Tripel $(\mathcal{R}, \mathcal{K}, l_0)$ definiert, wobei $(\mathcal{R}, \mathcal{K})$ ein GR für markierte Graphen ist und $l_0 \in L$ eine Anfangsbeschriftung. Ein unmarkierter Graph G wird von $(\mathcal{R}, \mathcal{K}, l_0)$ erkannt, wenn der markierte Graph (G, λ_{l_0}) von $(\mathcal{R}, \mathcal{K})$ erkannt wird, wobei λ_{l_0} die Beschriftungsfunktion bezeichnet, die alle Knoten und Kanten mit der Markierung l_0 versieht. $L(\mathcal{R}, \mathcal{K}, l_0)$ bezeichnet die Klasse von unmarkierten Graphen, die von $(\mathcal{R}, \mathcal{K}, l_0)$ erkannt werden.

Da es nicht handlich ist, für einen GR die Menge der terminalen Graphen, die auch nicht-endlich sein kann, aufzulisten, kann diese Menge auch durch eine Formel über die Beschriftungen des Graphen definiert werden. Durch die Formel kann ausgedrückt werden, welche Beschriftungen in einem terminalen Graphen auftreten sollen und welche nicht. Eine solche Formel heißt *Endbedingung*.



Graph G

Abbildung 26: G erfüllt $\neg I$, aber nicht $F \wedge \neg 1 \wedge \neg I$

Die Formel $\neg I$ drückt aus, dass in keinem terminalen Graphen die Beschriftung I vorkommen darf. Der Graph in Abbildung 26 erfüllt diese Formel, da kein Knoten und keine Kante mit I beschriftet ist. Der Graph ist also Element der durch die Formel $\neg I$ definierten Menge von terminalen Graphen.

Die Formel $F \wedge \neg 1 \wedge \neg I$ drückt aus, dass mindestens einmal die Beschriftung F in einem terminalen Graphen vorkommen muss, aber weder die Beschriftung 1 noch die Beschriftung I vorkommen darf. Der Graph in Abbildung 26 erfüllt diese Formel nicht, da eine Kante mit 1 beschriftet ist. Der Graph ist also kein Element der durch die Formel $F \wedge \neg 1 \wedge \neg I$ definierten Menge von terminalen Graphen.

Definition 9.4 (Endbedingung)
Im Folgenden wird $\mathcal{K}$ durch *Endbedingungen* beschrieben, die auf den Beschriftungen induktiv wie folgt definiert sind:

1. $\forall \ell \in L, \ell$ ist eine Formel,
2. wenn φ und ψ Formeln sind, dann sind auch $\neg\varphi, \varphi \vee \psi$ und $\varphi \wedge \psi$ Formeln.

Für ein $\ell \in L$ erfüllt ein markierter Graph G die Formel ℓ, wenn $\lambda^{-1}(l) \neq \emptyset$, wenn also die Beschriftung ℓ mindestens einmal in G vorkommt. Per Induktion erfüllt G $\neg\varphi$, wenn G nicht φ erfüllt, $\varphi \vee \psi$, wenn G φ oder ψ erfüllt, $\varphi \wedge \psi$, wenn G φ und ψ erfüllt. Somit ermöglicht es eine Endbedingung, das Vorhandensein oder das Fehlen einer Beschriftung in einem Graphen festzustellen.

Die Aussage, dass die Beschriftung λ von G genau alle Beschriftungen einer Untermenge U von L enthält, kann durch die Endbedingung $\bigwedge_{l \in U} l \wedge \bigwedge_{l' \in L \setminus U} \neg l'$ ausgedrückt werden. Die erste Teilfomel fordert, dass alle Beschriftungen von U im Graphen auftauchen, die zweite, dass keine Beschriftung vorkommt, die nicht Element von U ist.

Endbedingungen erlauben es nicht, Knoten oder Kanten mit einer bestimmten Beschriftung zu zählen, oder ihre relativen Positionen zu überprüfen, da die Formeln lediglich die Beschriftungsmenge des Graphen betrachten, also die Menge der Beschriftungen der Knoten und Kanten. Zum Beispiel ist es nicht möglich, festzustellen, ob ein Graph genau einen mit T beschrifteten Knoten enthält, oder zwei adjazente mit T beschriftete Knoten.

Jeder Graph, der eine Formel φ erfüllt, ist Element der terminalen Graphen. Somit können die terminalen Graphen auch folgendermaßen definiert werden:

Definition 9.5 (Menge terminaler Graphen)
Sei φ eine Endbedingung. Die Menge der terminalen Graphen $\mathcal{K}(\varphi)$ ist definiert durch $\mathcal{K}(\varphi) = \{(G, \lambda) : \ (G, \lambda)\ erf\ddot{u}llt\ \varphi\}$.

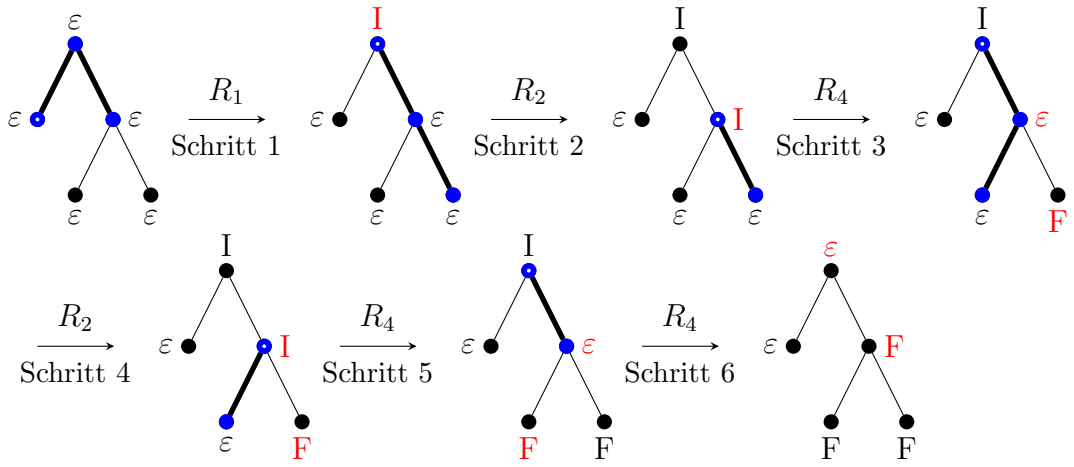
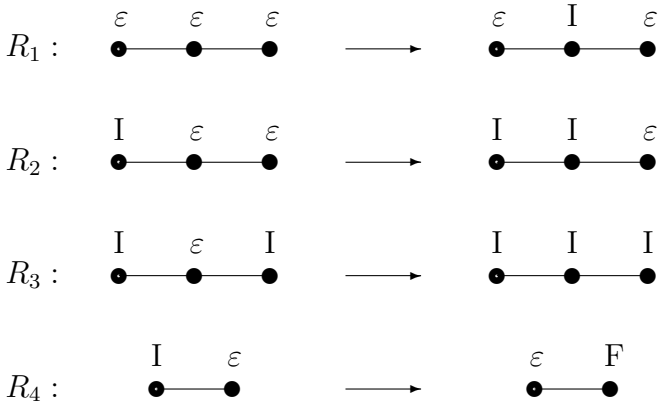


Abbildung 27: Erkennung eines Waldes.

9.2 Beispiele

In diesem Abschnitt werden zwei GR vorgestellt. Der eine erkennt Wälder, der andere vollständige Graphen. Der folgende GR erkennt die Klasse der Wälder.

Beispiel 9.6 (GR für Wälder)
Sei $W = (\mathcal{R}, \mathcal{K}(\varphi))$ ein GR, wobei φ die Endbedingung $\varphi = \neg I$ sei und $\mathcal{R} = (L, I, P, >)$ ein PGRS mit $L = \{\varepsilon, I, F, 0\}, I = \{\varepsilon, 0\}$ und $P = \{R_1, R_2, R_3, R_4\}$, der Prioritätsrelation $\{R_1, R_2, R_3\} > \{R_4\}$ und den folgenden Regeln:



Da die ersten drei Regeln eine höhere Priorität als die vierte haben, werden alle Knoten eines im Graphen enthaltenen Kreises mit I markiert, die übrigen mit F. Ist (G, λ) ein markierter Graph, dessen Knoten mit ε beschriftet sind, dann hat jeder Graph $(G, \lambda') \in Irred_{\mathcal{R}}(G, \lambda)$ genau dann keinen I-beschrifteten Knoten (und erfüllt damit φ), wenn G keinen Kreis hat. Ein Graph, der keinen Kreis hat, ist ein Baum und somit ist $(\mathcal{R}, \mathcal{K}(\varphi))$ ein deterministischer GR für die Klasse der Bäume.

Abbildung 27 zeigt ein Beispiel für die Anwendung des Algorithmus. Aus Platzgründen wird der Algorithmus lediglich an einem Baum durchgeführt, in einem Wald erfolgt die Berechnung für jeden Baum entsprechend.

Im ersten Schritt wird mit der Regel 1 die Wurzel mit I beschriftet, im zweiten mit der Regel 2 der Median. Im dritten Schritt wird mit der Regel 4 eines der Blätter

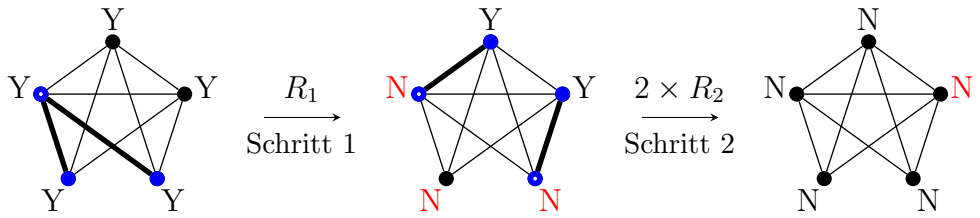


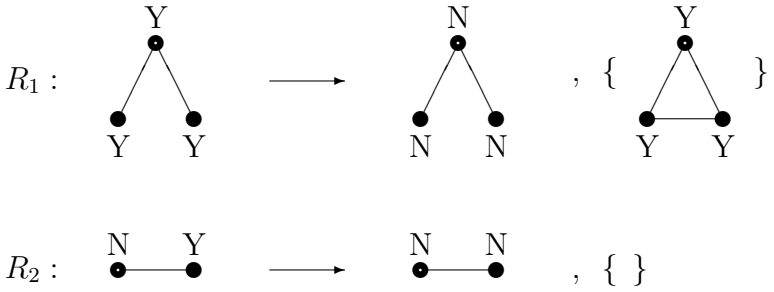
Abbildung 28: Erkennung eines vollständigen Graphen.

mit F und der Median wieder mit ε beschriftet, da hier die ersten drei Regeln nicht angewendet werden können. Nun kann auf den Wurzelknoten wieder die Regel 4 angewendet werden, die Beschriftung des Median wird wieder zu I. In den letzten beiden Schritten werden durch die Regel 4 alle I-Beschriftungen gelöscht.

Da nach der Terminierung des Algorithmus kein Knoten mit I beschriftet ist, erfüllt der Graph die Endbedingung $\neg I$ und ist somit Element der Menge $\mathcal{K}(\varphi)$ und damit ein Wald.

Im nächsten Beispiel wird ein GR für vollständige Graphen vorgestellt.

Beispiel 9.7 (GR für vollständige Graphen)
Sei $R = (L, I, P)$ das FCGRS mit $L = \{N, Y, 0\}$, $I = \{N, 0\}$ und $P = \{R_1, R_2\}$ mit den folgenden Umbeschriftungs-Regeln mit verbotenem Kontext:



Dann gilt: Ist (G, λ) ein markierter Graph, dessen Knoten mit Y beschriftet sind, dann ist (G, λ) irreduzibel genau dann, wenn G ein vollständiger Graph ist. Ist G nicht vollständig, kann Regel R_1 angewendet werden und alle Knoten des Graphen werden in Folge von Regel R_2 mit N beschriftet. Ist also die Endbedingung $\varphi = \neg N$, dann ist $(\mathcal{R}, \mathcal{K}(\varphi))$ ein deterministischer GR für die Klasse der vollständigen Graphen.

Abbildung 28 zeigt ein Beispiel für die Anwendung des Algorithmus. Im ersten Schritt wird die Regel 1 angewendet, die drei Knoten, die kein geschlossenes Dreieck bilden, mit N beschriftet. Im zweiten Schritt kann parallel die Regel 2 an den beiden verbliebenen Y-beschrifteten Knoten durchgeführt werden, die ebenfalls mit N beschriftet werden. Da nach der Terminierung des Algorithmus alle Knoten mit N beschriftet sind, erfüllt der Graph die Endbedingung $\neg N$ nicht und ist somit kein Element der Menge $\mathcal{K}(\varphi)$ und kein vollständiger Graph.

9.3 Voraussetzung für das Erkennen einer Klasse

In diesem Abschnitt wird beschrieben, welche Voraussetzung eine Klasse von Graphen erfüllen muss, um erkannt werden zu können. Eine Klasse von Graphen ist abgeschlossen unter k-Überdeckungen, wenn sie alle k-Überdeckungen ihrer Elemente enthält.

Zunächst wird im Folgenden gezeigt, dass eine Klasse der Graphen, die von einer k-lokal erzeugten Umbeschriftungs-Relation erkannt wird, unter k-Überdeckungen abgeschlossen ist.

Behauptung 9.8 (Abgeschlossenheit unter k-Überdeckungen)
Sei G eine k-Überdeckung eines Graphen H und $\mathcal{R}$ eine k-lokale Umbeschriftungs-Relation. Dann erkennt jeder GR $(\mathcal{R}, \mathcal{K})$, der H erkennt, ebenso G. Wenn $(\mathcal{R}, \mathcal{K})$ deterministisch ist, gilt darüber hinaus auch die umgekehrte Aussage, dass G erkannt wird, wenn H erkannt wird.

Beweis Sei C eine Umbeschriftungs-Sequenz, die H erkennt, also eine Sequenz, die zu einem $(H, \mu) \in \mathcal{K}$ führt. Angenommen, G sei eine q-schichtige Überdeckung von H. Für jeden Umbeschriftungs-Schritt von C auf dem Graphen H existiert eine Umgebung $B_H(v, k)$, so dass nur Beschriftungen von $B_H(v, k)$ in diesem Schritt verändert werden. Dieser Umbeschriftungs-Schritt kann somit auf die q disjunkten Umgebungen von $\gamma^{-1}(B_H(v, k))$ angewendet werden. Auf diese Weise kann mit Hilfe von γ^{-1} aus C eine Umbeschriftungs-Sequenz erzeugt werden, die den Graphen G erkennt.

Angenommen, $(\mathcal{R}, \mathcal{K})$ sei deterministisch. Dann ist $H \notin L(\mathcal{R}, \mathcal{K})$, wenn es eine Umbeschriftungs-Sequenz gibt, die H zurückweist. Wie zuvor kann diese Sequenz auf G simuliert werden, woraus sich eine Sequenz ergibt, die G zurückweist. Da aber $(\mathcal{R}, \mathcal{K})$ deterministisch ist, wird G von allen Umbeschriftungs-Sequenzen zurückgewiesen, wenn G von einer zurückgewiesen wird, somit ist $G \notin L(\mathcal{R}, \mathcal{K})$ □

Daraus folgt, dass zwei Graphen, die eine gemeinsame k-Überdeckung haben, entweder beide von einem GR erkannt oder beide verworfen werden.

Lemma 9.9
Sei $\mathcal{R}$ eine k-lokal erzeugte Umbeschriftungs-Relation und $(\mathcal{R}, \mathcal{K})$ ein deterministischer GR. Wenn zwei Graphen G und H eine gemeinsame k-Überdeckung haben, dann gilt $G \in L(\mathcal{R}, \mathcal{K})$ genau dann, wenn $H \in L(\mathcal{R}, \mathcal{K})$.

Im Folgenden werden diese Aussagen verwendet, um die Nicht-Erkennbarkeit der Klasse der Graphen zu zeigen, die genau eine ℓ-Beschriftung haben und der Klasse der Graphen, die eine ungerade Anzahl von Knoten haben. Im Allgemeinen ist nicht k-lokal überprüfbar, ob ein Graph eine bestimmte Anzahl von Beschriftungen hat.

Behauptung 9.10
Sei $\ell \in L$ eine Beschriftung. Es gibt keine lokal erzeugte Umbeschriftungs-Relation, die - weder deterministisch noch nicht-deterministisch - die Klasse der beschrifteten Graphen erkennt, die genau einen ℓ-beschrifteten Knoten haben.

Beweis Sei (G, λ) ein beschrifteter Ring mit genau einem ℓ-beschrifteten Knoten. Aus Abbildung 18 ist bekannt, dass es eine strikte k-Überdeckung (G', λ') für G gibt. Nach Satz 7.3 hat (G', λ') mindestens zwei ℓ-beschriftete Knoten (einen für

jede Schicht) und wird nach Behauptung 9.10 immer dann erkannt, wenn (G, λ) erkannt wird. Somit gibt es keinen GR, der die Klasse der beschrifteten Graphen erkennt, die genau einen ℓ-beschrifteten Knoten haben. □

Ebenso kann die Klasse der Graphen mit höchstens i ℓ-beschrifteten Knoten für ein $i \in \mathbb{N}$ nicht erkannt werden.

Jede q-schichtige Überdeckung eines Graphen H hat q-mal so viele Knoten wie H. Daher gibt es für jeden Ring mit ungerader Knotenzahl eine k-Überdeckung mit gerader Knotenzahl, nämlich gerade eine q-schichtige k-Überdeckung mit geradem q, da das Produkt einer geraden und einer ungeraden Zahl gerade ist. Daraus und aus Behauptung 9.8 folgt:

Behauptung 9.11
Es gibt keine lokal erzeugte Umbeschriftungs-Relation, die - deterministisch oder nicht-deterministisch - die Klasse der Graphen erkennt, die eine ungerade Anzahl von Knoten haben.

In [2] wird bewiesen, dass die Klasse von Graphen, die eine gerade Anzahl von Knoten haben, von einem Graph-Umbeschriftungs-System nicht-deterministisch erkennbar ist.

Ein Erkennungs-Algorithmus ist ein Algorithmus, der fest stellt, ob ein Graph zu einer Klasse von Graphen gehört, oder nicht. Eine Klasse von Graphen kann nur dann k-lokal erkannt werden, wenn sie unter k-Überdeckungen abgeschlossen ist.

10 Das Terminierung-Erkennungs-Problem

Ein wichtiges Ziel bei der Konstruktion eines verteilten Algorithmus ist es, sicherzustellen, dass es möglich ist, die Terminierung des Algorithmus zu erkennen. Das bedeutet, dass mindestens ein Knoten des Netzwerkes in der Lage ist, festzustellen, ob der Algorithmus terminiert ist.

In diesem Kapitel wird diese Eigenschaft im Rahmen von lokal erzeugten Umbeschriftungs-Relationen betrachtet. Es wird die Frage untersucht, ob ein Knoten $v \in V(G)$ unter Berücksichtigung einer Umgebung $B_G(v,k)$ in der Lage ist, festzustellen, ob der Graph G irreduzibel ist oder nicht. Die Ergebnisse dieses Kapitels wurden in [4] entwickelt.

Im ersten Abschnitt wird die Terminierungs-Erkennung formal eingeführt, im zweiten ein Beispiel zur lokalen Erkennung der globalen Terminierung angegeben. Im dritten Abschnitt wird eine Beweistechnik vorgestellt, mit der überprüft werden kann, ob die Terminierung eines Algorithmus für eine bestimmte Klasse von Graphen erkannt werden kann.

10.1 Einführung und Definition

Sei L ein Alphabet. Bezeichne $\mathcal{G} = \mathcal{G}_L$ die Menge aller L-markierten Graphen. Ein Knoten v eines Graphen G erkennt die Terminierung eines Algorithmus, wenn seine k-Umgebung $B_G(v,k)$ isomorph zu einem *charakterisierenden* Graphen ist.

Definition 10.1 (Charakterisierung von Normalformen)
Sei $\mathcal{R}$ eine k-lokal erzeugte Umbeschriftungsrelation. Sei $\mathcal{I} \subseteq \mathcal{G}$ die Klasse der *initialen Graphen* und sei $T \subseteq \mathcal{G}$, wobei alle Graphen in T zusammenhängend sind. T *charakterisiert* die Normalformen, die von $\mathcal{I}$ erreicht werden können, genau dann, wenn für jeden Graphen $(G,\lambda) \in \mathcal{I}$ mit $(G,\lambda)\ \mathcal{R}^*\ (G,\lambda')$ gilt:

$$(G,\lambda') \text{ ist Normalform} \quad \Leftrightarrow \quad \begin{array}{l}(G,\lambda') \text{ enthält einen Teilgraphen,}\\ \text{der isomorph zu einem } (K,\mu) \in T \text{ ist}\end{array}$$

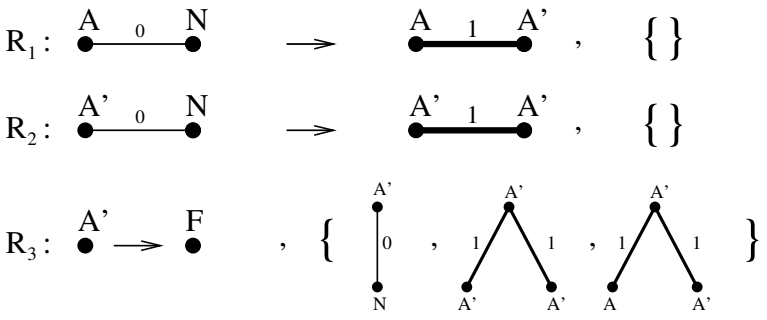
(G,λ') heißt in diesem Fall (K,μ)-*chrakterisiert*.

Sei $r \in \mathbb{N}$. Normalformen sind r-*charakterisiert*, wenn alle Elemente von T einen Radius von höchstens r haben.

10.2 Beispiel

Als Beispiel dient hier wieder der parallele Algorithmus zur Spannbaum-Berechnung aus Kapitel 4.3.2.

Sei $\mathcal{R} = (L, I, P)$ ein FCGRS mit dem Alphabet $L = \{N, A, M, F, 0, 1\}$, den Anfangsbeschriftungen $I = \{N, A, 0\}$, und der Regelmenge $P = \{R_1, R_2, R_3\}$:



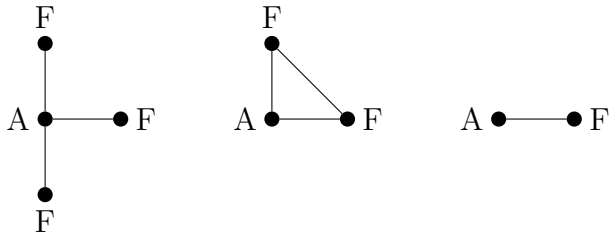


Abbildung 29: Elemente der Menge T

Die Normalformen werden durch $T = \{(K, \mu) \mid \exists v \in V(K),\ \forall v' \in V(K),\ v \neq v' : \mu(v) = A,\ \mu(v') = F,\ \{v, v'\} \in E(K)\}$ charakterisiert. Das bedeutet, dass es in jedem Graphen $(K, \mu) \in T$ genau einen mit A markierten Knoten gibt. Alle anderen Knoten sind mit F markiert und zu diesem Knoten adjazent. Abbildung 29 zeigt Beispiele für Graphen der Menge T.

Abbildung 30 zeigt eine Berechnung von $\mathcal{R}$ auf einem exemplarischen Graphen. Nur im resultierenden irreduziblen Graphen gibt es einen Teilgraphen, der in T enthalten ist, die globale Terminierung wird von dem mit A markierten Knoten erkannt.

10.3 Voraussetzung für das Erkennen der Terminierung

In diesem Abschnitt wird eine Beweistechnik auf Basis von k-Überdeckungen vorgestellt, mit der festgestellt werden kann, ob die Terminierung eines Algorithmus für eine Klasse von Graphen erkannt werden kann.

Sei $\mathcal{I}$ eine Klasse von Graphen, die sowohl einen Graphen H als auch eine strikte k-Überdeckung G von H enthält. Angenommen die Terminierung eines Algorithmus für $\mathcal{I}$ könnte durch einen Graphen K erkannt werden. Dann kann jeder k-lokale Berechnungs-Schritt eines Algorithmus, der in H angewendet wird, auch in immer der selben Schicht von G angewendet werden. Wird nach der Terminierung des Algorithmus der Graph H durch K charakterisiert, wird auch der Graph G durch K charakterisiert, und damit wäre der Algorithmus auch für G terminiert. Dies ist aber ein Widerspruch, da G nicht in Normalform ist, weil die bereits für eine Schicht durchgeführten Umbeschriftungs-Schritte auch noch in den übrigen Schichten durchgeführt werden können. Somit kann die Terminierung eines Algorithmus für die Klasse $\mathcal{I}$ nicht erkannt werden.

Abbildung 31 verdeutlicht diesen Zusammenhang anhand einer 2-schichtigen Überdeckung des symmetrisch markierten, vollständigen Graphen mit 5 Knoten mit dem Algorithmus aus Beispiel 8.3. Die Normalformen werden durch den Graphen charakterisiert, der genau einen - mit E markierten - Knoten hat. Die Umbeschriftungs-Regeln können im linken Teilgraphen angewendet werden, bis Regel 2 einen Knoten mit E markiert. In dieser Konfiguration ist der Algorithmus aber noch nicht terminiert, da die Regeln auch noch im rechten Teilgraphen angewendet werden können. Dennoch wird fälschlicherweise eine Terminierung festgestellt, da ein Knoten mit E markiert und somit der Graph als Normalform charakterisiert ist.

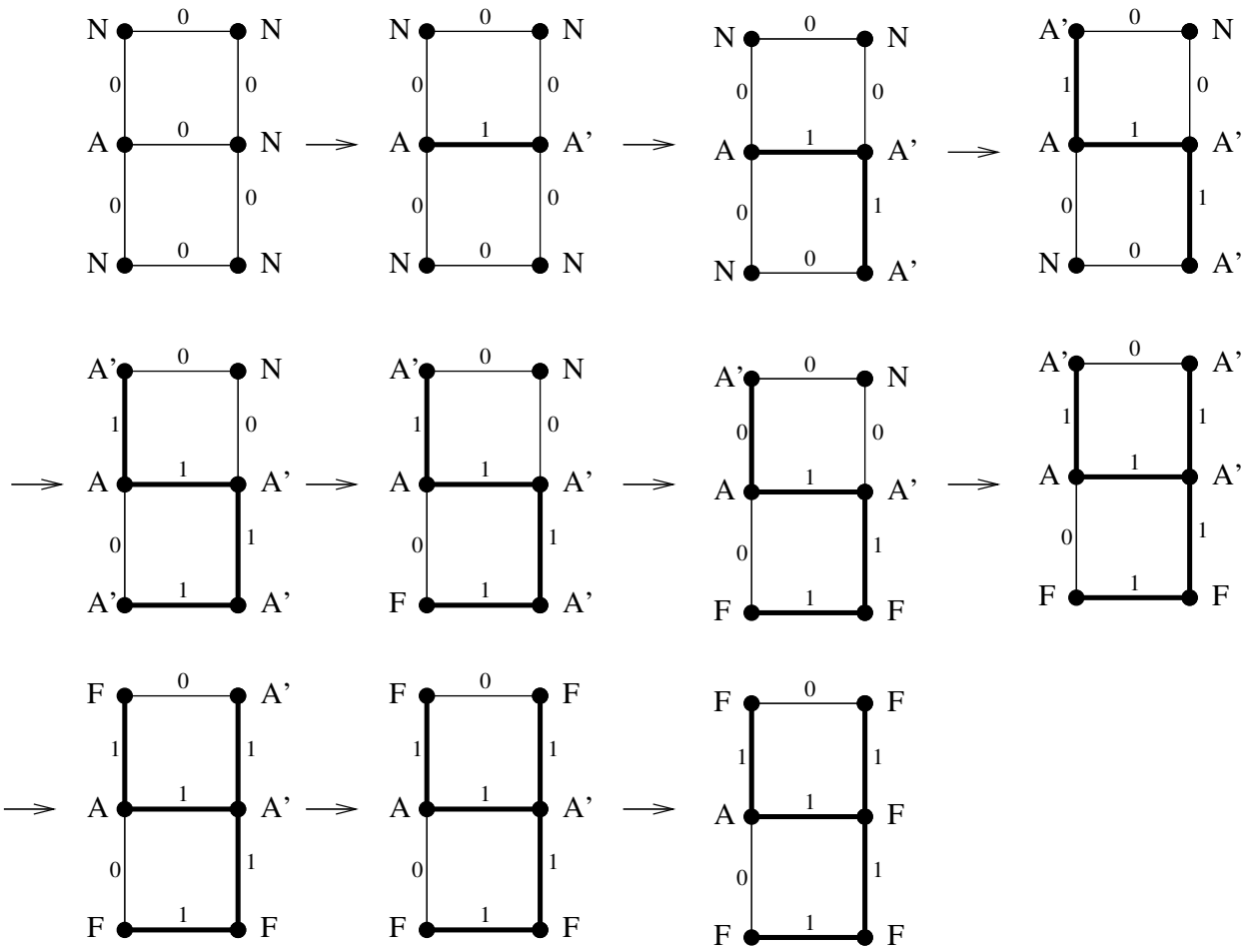


Abbildung 30: Erkennung der Terminierung

Satz 10.2
Sei $\mathcal{I} \subseteq \mathcal{G}$ eine Klasse von zusammenhängenden Graphen und sei $\mathcal{R}$ eine k-lokal erzeugte Umbeschriftungsrelation. Wenn $\mathcal{I}$ zwei Graphen (G, λ) und (H, λ') enthält, wobei (G, λ) mit der Abbildung γ eine strikte Überdeckung von (H, λ') ist, dann können die Normalformen, die von $\mathcal{I}$ erreicht werden können, für jedes $r \leq k$ nicht r-charakterisiert werden.

Die globale Terminierung eines Algorithmus kann lokal erkannt werden, wenn jeder Knoten über Informationen über die Beschriftungen der Knoten in einer bestimmten Umgebung verfügt und anhand dieser Informationen die Terminierung erkennen kann. Die Terminierung eines k-lokalen Algorithmus für eine Klasse von Graphen, die sowohl einen Graphen H als auch eine k-Überdeckung von H enthält, kann nicht erkannt werden.

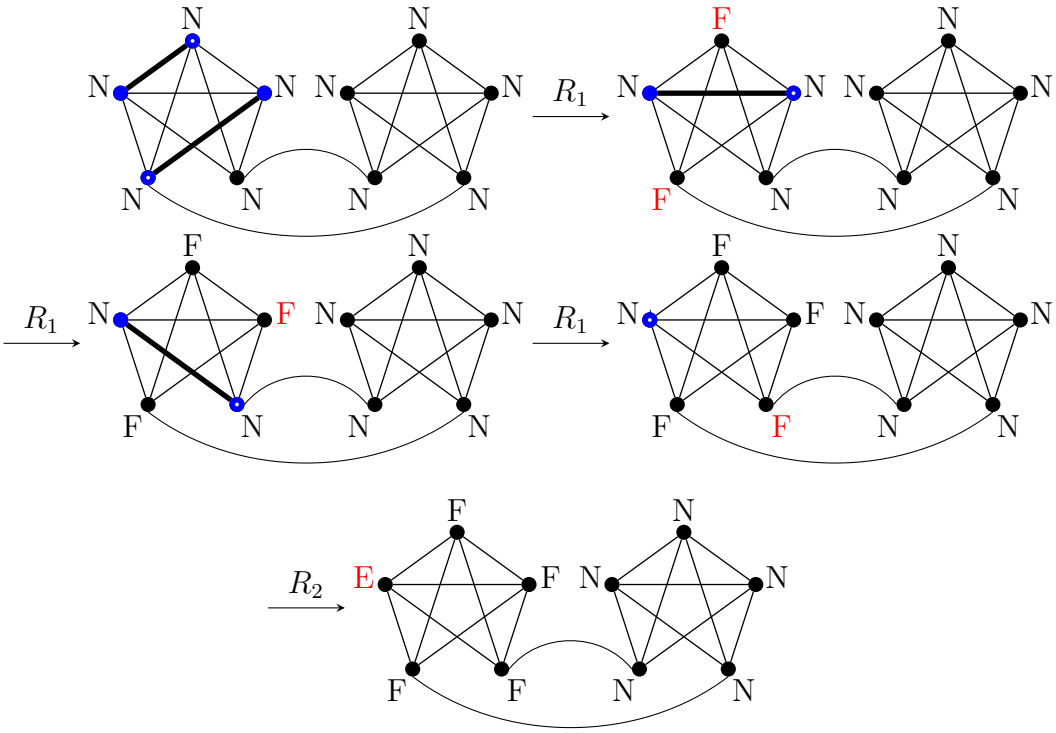


Abbildung 31: Erkennung der Terminierung in einer Überdeckung

11 Zusammenfassung

In dieser Arbeit wurden Graphumbeschriftungssysteme (GRS) als Modell zur Formalisierung verteilter Algorithmen vorgestellt. Dieses Modell ist immer dann sinnvoll, wenn ein Problem auf einem gegebenen Graphen berechnet werden soll, ohne die Graphstruktur zu verändern. Graphumbeschriftungssysteme arbeiten auf markierten Graphen, die mit Hilfe von Graphumbeschriftungsregeln umbeschriftet werden. Dabei wird ein Vorkommen einer linken Regelseite durch ein Vorkommen einer rechten Regelseite ersetzt. Damit ist ein Graphumbeschriftungssystem ein spezielles Graphersetzungssystem.

Eine Beispielanwendung von Graphersetzungssystemen ist die Formalisierung eines Algorithmus, der einen Spannbaum eines gegebenen Graphen berechnet. Eine Erweiterung von Graphumbeschriftungssystemen sind Graphumbeschriftungssysteme mit Prioritäten (PGRS). In einem solchen PGRS wird den Umbeschriftungsregeln eine Priorität zugeordnet, die die Reihenfolge der Regelanwendung auf ein Vorkommen festlegt. Falls möglich, müssen zunächst Regeln höherer Priorität angewendet werden. Nur falls dies nicht möglich ist, dürfen Regeln niedrigerer Priorität angewendet werden. Eine andere Erweiterung von Graphumbeschriftungssystemen sind Graphumbeschriftungssysteme mit verbotenen Kontexten (FCGRS). In einem FCGRS ist jede Regel um eine Menge von Kontexten erweitert. Ein Kontext ist ein beschrifteter Graph, in dem die linke Regelseite vorkommt, und somit eine Umgebung der linken Regelseite darstellt. Als Einschränkung für die Regelanwendung wird gefordert, dass jede Regel nur dann auf ein Vorkommen angewendet werden kann, wenn die Umgebung des Vorkommens keinem der Kontexte entspricht. Ein wichtiges Ergebnis ist, dass PGRS und FCGRS gleich mächtig sind.

Durch die Formalisierung eines Algorithmus als GRS, PGRS oder FCGRS können Beweise für die Korrektheit, die Terminierung, die Terminierungserkennung und die Zeitkomplexität geführt werden. Zum Beweis der Korrektheit werden invariante Eigenschaften herangezogen, die induktiv für jeden Zustand eines beschrifteten Graphen gelten. Diese werden dann benutzt, um die Korrektheit eines Algorithmus anhand eines beliebigen irreduziblen Graphen zu folgern. Das Spannbaum-Problem ist mit Zeitkomplexität $O(n)$ mit Hilfe eines GRS lösbar, falls auf die lokale Erkennung der globalen Terminierung des Algorithmus verzichtet wird. Mit Hilfe von PGRS oder FCGRS ist das Spannbaum-Problem mit der lokalen Terminierungserkennung lösbar. Im Falle des FCGRS kann der Spannbaum auch verteilt berechnet werden. Beide Algorithmen haben ebenfalls die Komplexität $O(n)$.

Für Graphumbeschriftungssysteme gibt es verschiedene Standardprobleme. Eines davon ist die Auswahl eines Knotens in einem Graphen. Ein Auswahl-Algorithmus markiert genau einen Knoten mit einer auswählenden Markierung. Das Auswahl-Problem ist lösbar, wenn der Graph nicht symmetrisch ist, er also keine Überdeckung eines anderen Graphen ist und nicht symmetrisch markiert ist. Ein Graph G ist eine Überdeckung eines Graphen H, wenn die Knoten von G surjektiv auf die Knoten von H abgebildet werden können. Dabei muss auf jeden Knoten von H die gleiche Anzahl von Knoten von G abgebildet werden. Außerdem muss für jede Kante $\{u, v\}$ in H jeder Knoten von G, der auf u abgebildet wird, mit einem Knoten, der auf v abgebildet wird durch eine Kante verbunden sein. Haben alle Knoten von G, die auf einen Knoten v von H abgebildet werden, die gleiche Markierung wie v, so ist der Graph G symmetrisch markiert. In einer Klasse von Graphen, die nicht symmetrisch markierte Graphen enthält oder unter Überdeckung abgeschlossen ist, kann keine

Auswahl durchgeführt werden.

Ein weiteres Standardproblem ist die Erkennung einer Klasse. Ein Erkennungs-Algorithmus stellt fest, ob ein Graph Element einer bestimmten Klasse von Graphen ist. Nach der Terminierung eines Erkennungs-Algorithmus wird überprüft, ob eine Endbedingung erfüllt wird. Eine Endbedingung ist eine Formel, die das Vorhandensein einer Markierung in einem irreduziblen Graphen fordert oder verbietet. Das Erkennungs-Problem ist lösbar, wenn die Klasse von Graphen unter Überdeckung abgeschlossen ist.

Da sich die Voraussetzungen für Auswahl und Erkennung widersprechen, ist es im Allgemeinen nicht möglich, zuerst die Klassenzugehörigkeit eines Graphen zu überprüfen und dann eine Auswahl in diesem Graphen zu treffen.

Ein Graphumbeschriftungsalgorithmen selbst betreffendes Problem ist das Terminierung-Erkennungs-Problem. Die Terminierung eines Algorithmus kann erkannt werden, wenn es möglich ist, eine Menge von charakterisierenden markierten Graphen anzugeben, die nur in einem irreduziblen Graphen vorkommen. Ist ein solcher charakterisierender Graph ein Teilgraph einer Konfiguration, so ist der Algorithmus terminiert. Das Terminierung-Erkennungs-Problem ist lösbar, wenn der Graph, auf den der Algorithmus angewendet wird, nicht symmetrisch ist. Somit ist die Terminierung eines Erkennungs-Algorithmus nicht feststellbar.

Graphumbeschriftungssysteme bieten Mechanismen zur präzisen formalen Formulierung von Algorithmen und ermöglichen es, für ein gegebenes Problem die Existenz bzw. Nicht-Existenz eines Algorithmus nachzuweisen.

Literatur

[1] I. Litovski, Y. Métivier, and É. Sopena. Graph relabelling systems and distributed algorithms. In *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 3, chapter 1, pages 1–56. World Scientific, 1999.

[2] I. Litovsky and Y. Metivier. Computing with graph rewriting systems with priorities. *Theoret. Comput. Sci.*, vol. 115:191–224., 1993.

[3] Y. Métivier and N. Saheb. Probabilistic analysis of an election algorithm in a tree. In *Colloqium on trees in algebra and programming*, volume 787 of *Lecture Notes in Computer Science*, pages 234–245. Springer-Verlag, 1994.

[4] A. Muscholl Y. Métivier and P.-A-Wacrenier. About the local detection of termination of local computations in graphs. In *International Colloquium on structural information and communication complexity*, pages 188–200. Carleton University Press, 1997.