Graphentheoretische Konzepte und Algorithmen

Thema 07
Graphersetzung

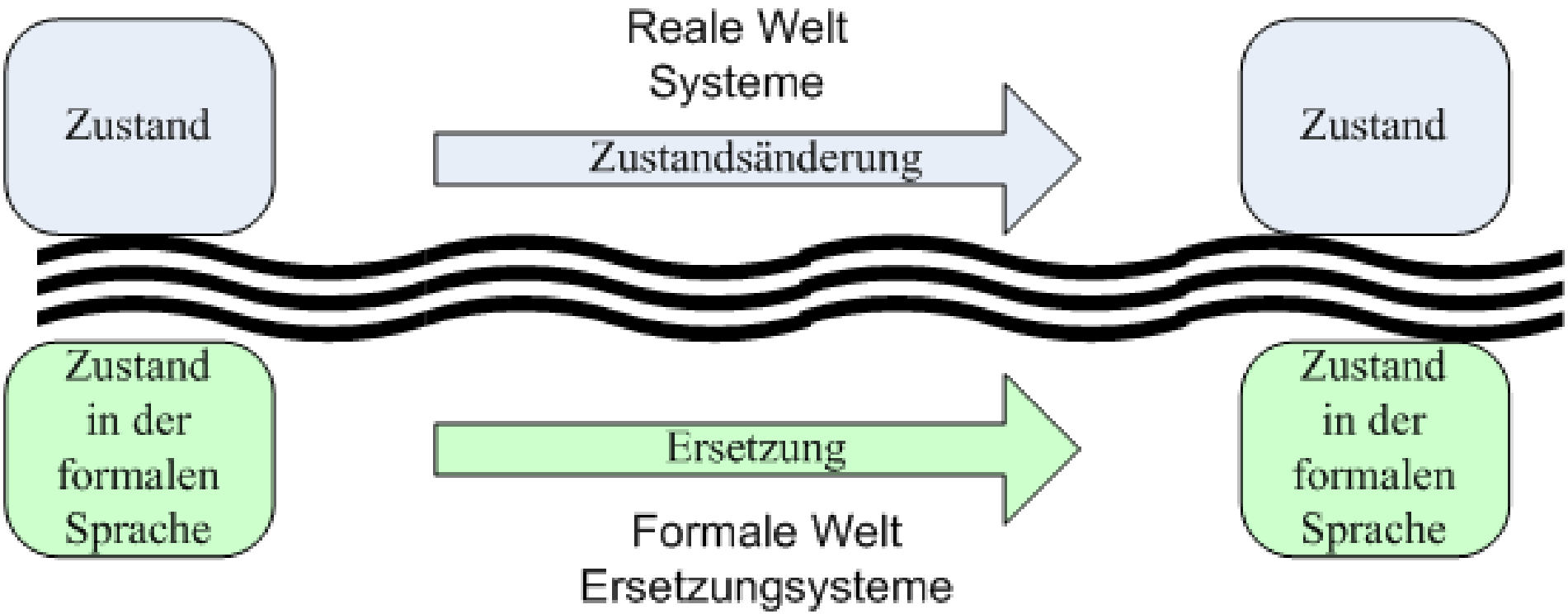
Julia Padberg

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Graphen in der Informatik

- **Graphentheorie:** Welche Typen von Graphen gibt es? Welche Strukturen treten in Graphen auf? Welche bekannten Sätze über Graphen gibt es? (z.B. Vierfarbentheorem)
- **Graphalgorithmen:** Wie kann man Graphen am besten algorithmisch verarbeiten? Welche Verfahren gibt es, um Graphen zu untersuchen? (z.B. Chinese Postman Problem, Max Flow)
- **Graph Drawing:** Wie lassen sich Graphen am besten visualisieren? Welche Algorithmen gibt es, um Graphen zu zeichnen?
- **Graphtransformation:** Wie kann man Graphen mit Hilfe von Regeln transformieren? Wie funktionieren Graphgrammatiken? Wie kann man nebenläufige Systeme mit Hilfe von Graphtransformationsregeln modellieren?

Einführung in Graphersetzungssysteme



- Beschreibung von System-Zuständen
  → Petri-Netze, Graphen, Hypergraphen, attributierte Graphen, ...
- Beschreibung von Zustands-Änderungen
  → Netz-, Graph-, Hypergraph-, attributierte Graph-Ersetzung, ...

Graphersetzungssysteme

Graphen

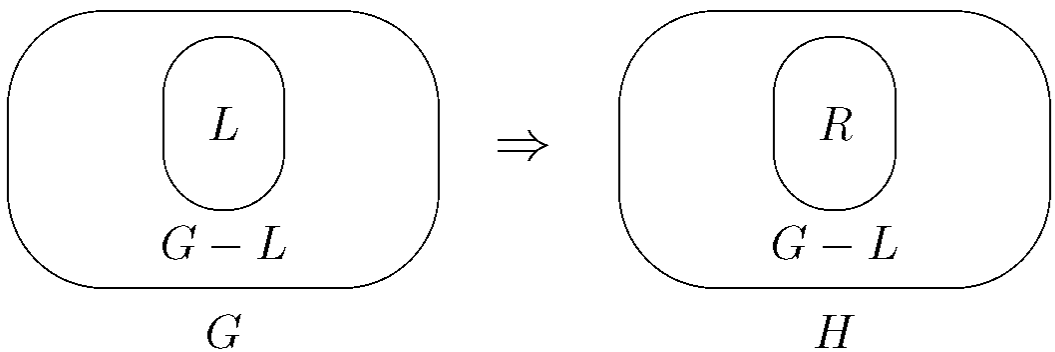
- komplexe Datenobjekte, die Informationen und Beziehungen repräsentieren
- anschaulich
- mathematische Gebilde

Graphmanipulation

- durch Ersetzungsregeln $L \Longrightarrow R$ (bewirken lokale Änderung)
- zum Erzeugen von Graphen
- zum Ändern von Zuständen
- als Berechnungsprozess
- $\vdots$

Idee der regelbasierten Graphersetzung

- Ziel: Intuitive und formale Beschreibung von Veränderungen.
- Idee: Regeln $L \Longrightarrow R$ beschreiben lokale Veränderungen.
- Graphmanipulation durch Anwendung von $L \Longrightarrow R$ in beliebiger Umgebung. Intuition: Gegeben ein “Vorkommen” von L in G, dann verändern wir L zu R.



- In welcher Form darf L in G vorkommen? Teilgraph? Teilgraph bis auf Isomorphie? . . .
- Was passiert mit Kanten in $G - L$, die Knoten in L berühren? Wie wird R mit $G - L$ verbunden?

Anwendungsbereiche

- Auswertung von funktionalen Ausdrücken
- Logische Programmierung
- Semantik von objektorientierten Sprachen
- **Semantik von visuellen Sprachen**
- modellgetriebene Softwareentwicklung
- Modellierung von nebenläufigen oder verteilten Systemen
- **Rollenbasierte Zugriffskontrolle**
- **Migration von Software**

Semantik von visuellen Sprachen

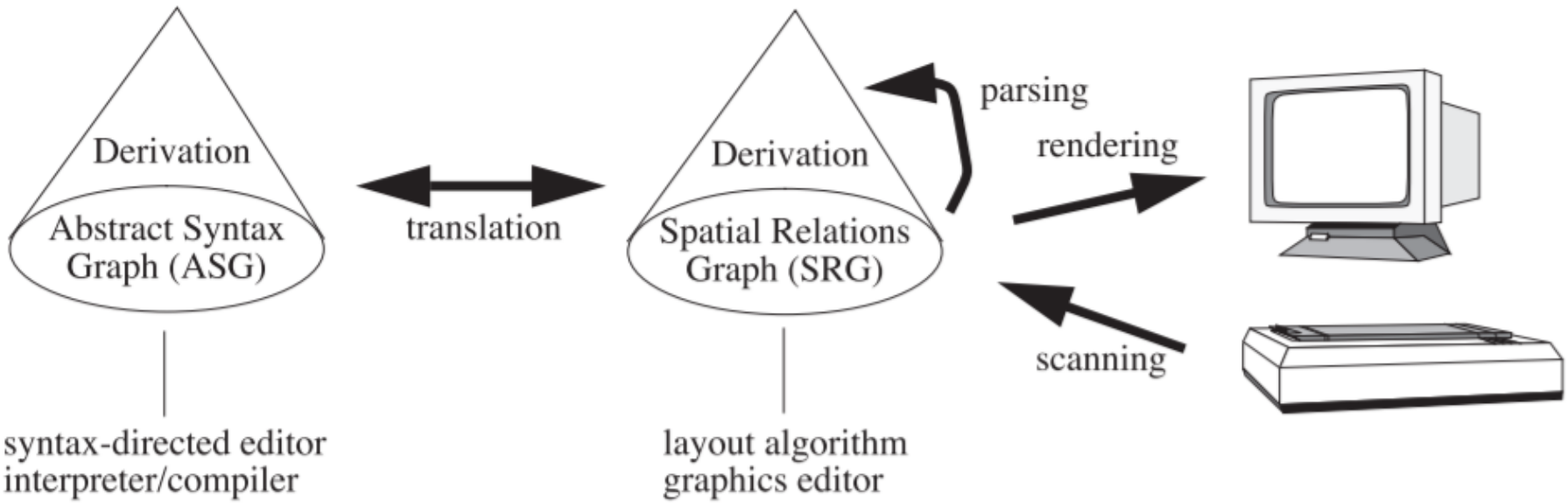
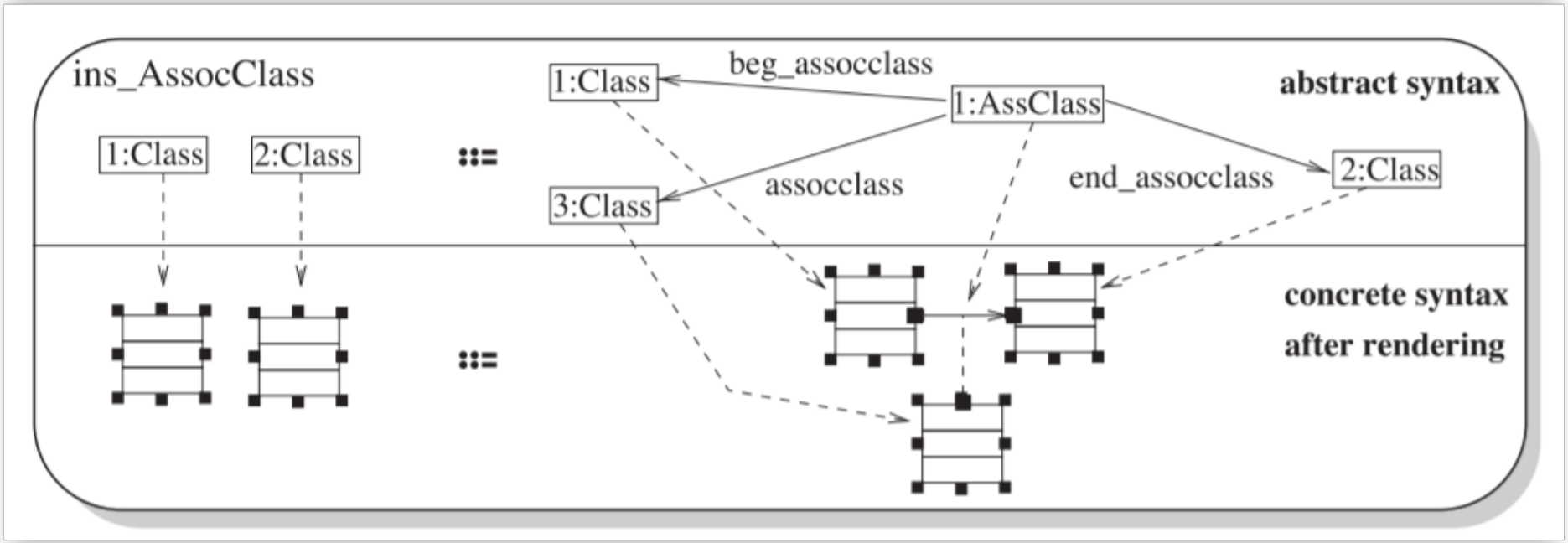
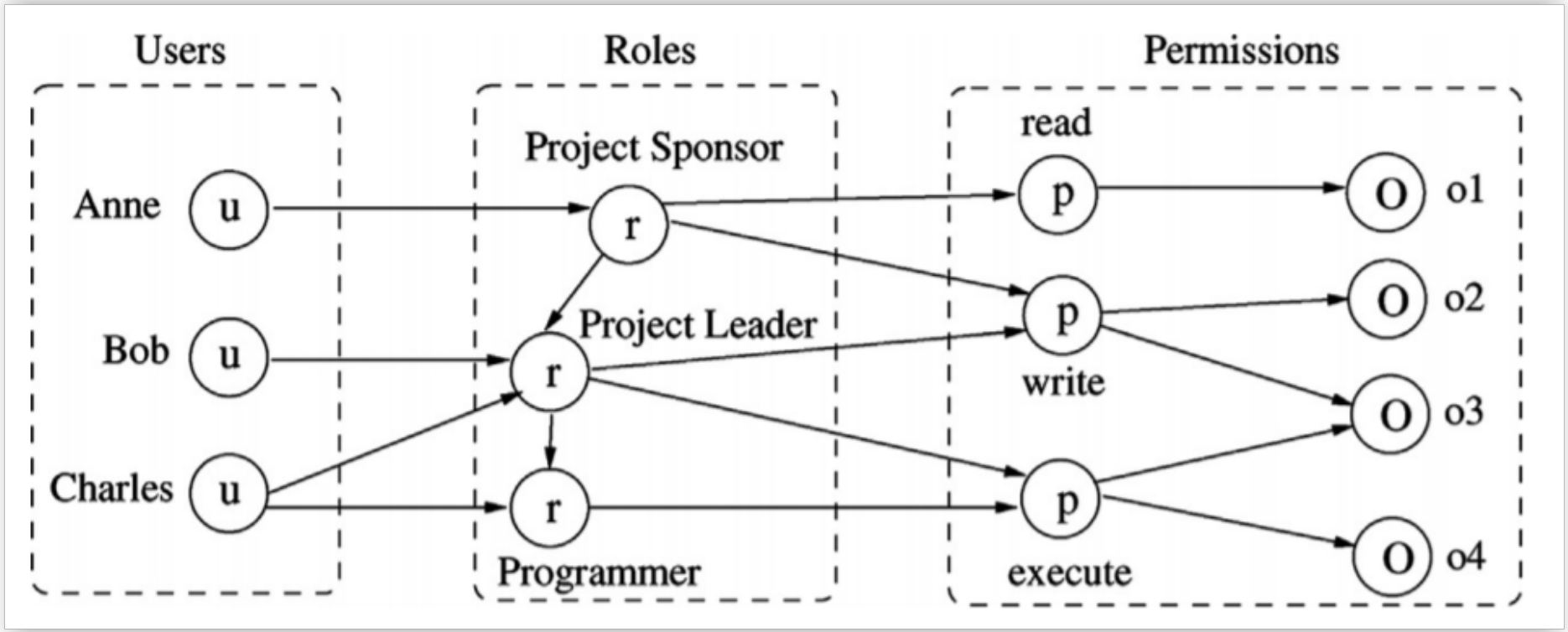


Fig. 5. Abstract and concrete syntax of visual languages from [18].

Einfügen einer Assoziation in Klassendiagramm



Rollenbasierte Zugriffskontrolle



Software Migration

Translating Satellite Procedures: PIL2SPELL

- Uni Luxembourg
  & industrieller Partner SES (Sociètè Europèenne des Satellites)
- Hersteller nutzen propritäre Sprachen
- SES betreibt 56 Satelliten
- automatisierte Migration
  open source satellite language SPELL

```
1  SELECT
2    CASE ($BATT = "HIGH")
3      CHECKTM(TEMP_C1)
4      CHECKTM(VOLT_D2 = 4)
5    ENDCASE
6    CASE ($BATT = "LOW")
7      SEND SWITCH_B1_B2
8        CHECKTM(VOLT3 = 5)
9      ENDSEND
10   ENDCASE
11 ENDSELECT
```

```
1 if (BATT == 'HIGH'):
2   GetTM('T TEMP_C1')
3   Verify([['T VOLT_D2', eq, 4]])
4 elif (BATT == 'LOW'):
5   Send(command = 'C SWITCH_B1_B2',
6        verify  = [['T VOLT3', eq, 5]])
7 #ENDIF
```

Fig. 1. Procedure written in *PIL* (left) and translated procedure in *SPELL* (right)

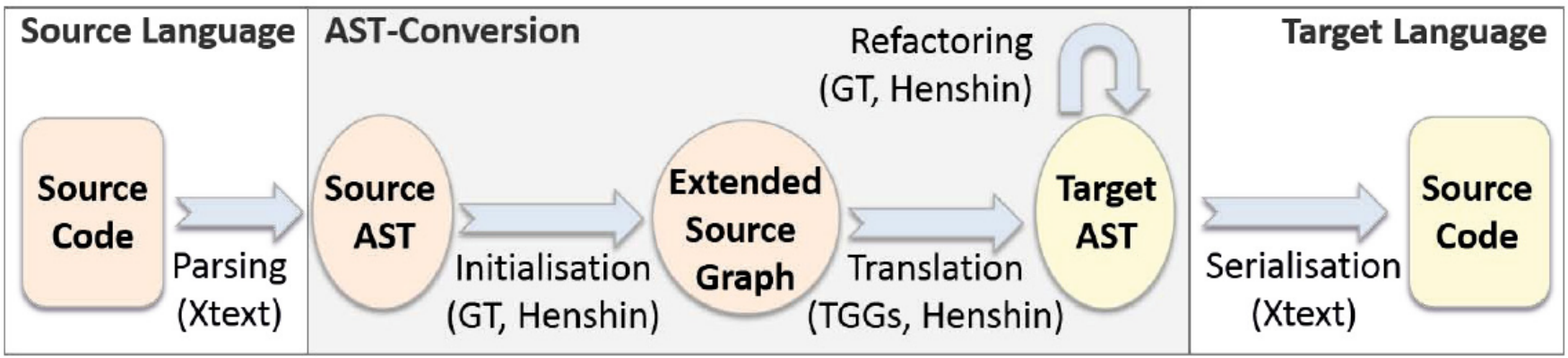


Fig. 2. Concept for software translation

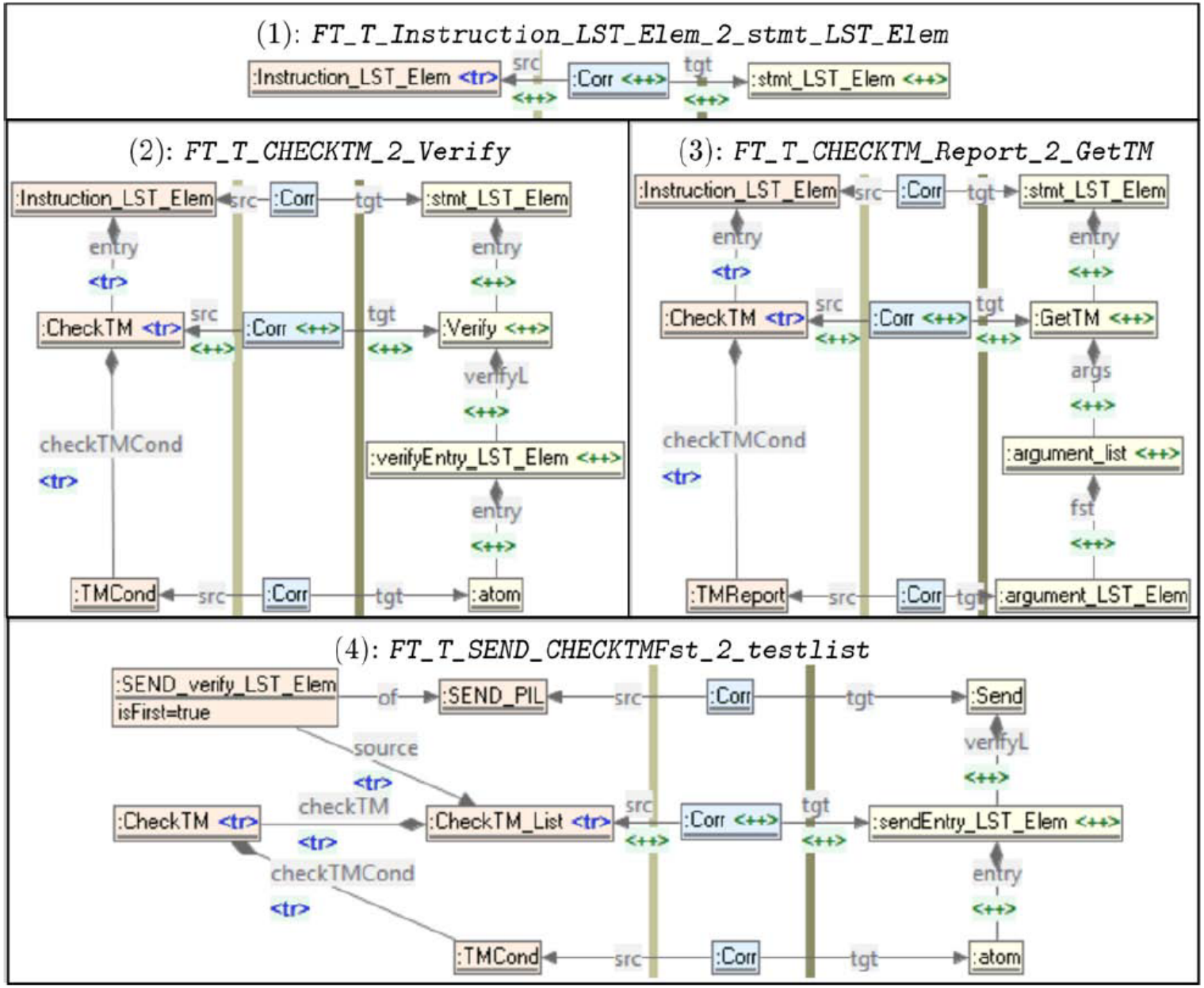


Fig. 5. Forward translation rules (generated by Henshin)

Übersicht

Intro GraTras

Basisdefinitionen

Formalisierung

Modellierungskonzepte

Graphersetzungssystem

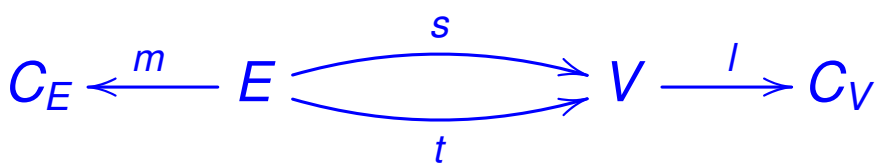
NACs

Graphen

Definition

Sei $C = (C_V, C_E)$ ein Paar von **Markierungsalphabeten**.
Ein gerichteter, markierter **Graph** über C ist durch $G = (V, E, s, t, l, m)$ gegeben. Dabei sind

- V, E endliche Mengen von **Knoten** und **Kanten**,
- $s, t \colon E \to V$ Abbildungen, die jeder Kante eine **Quelle** und ein **Ziel** zuordnen,
- $l \colon V \to C_V$ und $m \colon E \to C_E$ Abbildungen, die jedem Knoten eine **Knotenmarkierung** und jeder Kante eine **Kantenmarkierung** zuordnen.

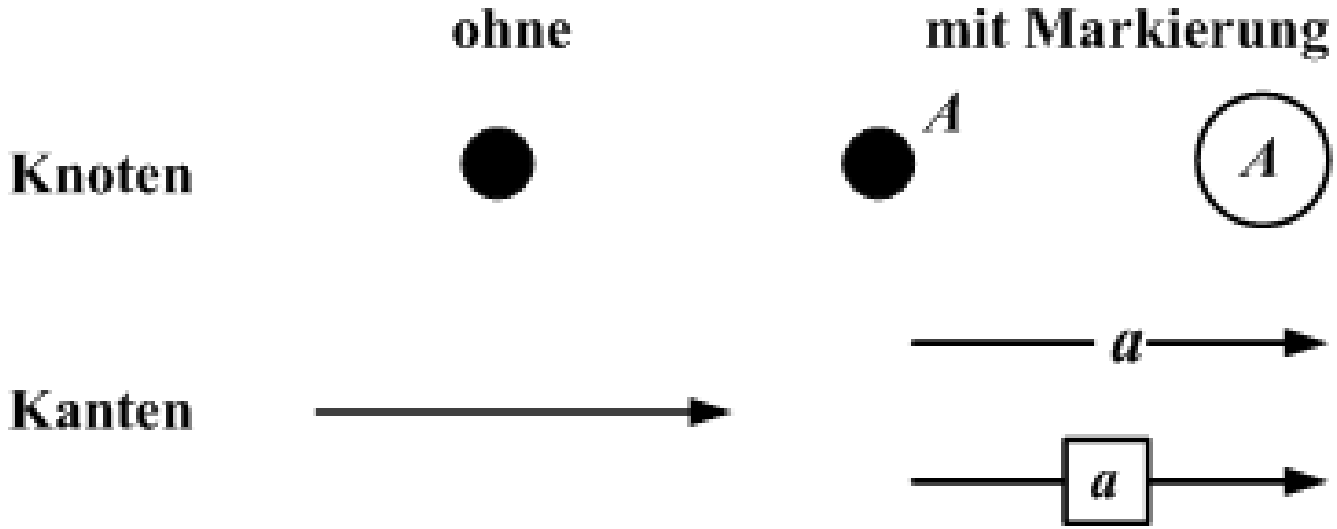


Die Komponenten von G werden auch mit V_G, E_G, s_G, t_G, l_G und m_G bezeichnet.

!!! Ein Graph mit leerer Knotenmenge heißt **leerer** Graph und wird mit $\emptyset$ bezeichnet.

Notation

. . . . . und welche Graphen wir ab jetzt benutzen



Wir benutzen also

- gerichtete,
- kantenmarkierte,
- knotenmarkierte
- Multigraphen (also mit Schlingen & Mehrfachkanten)

Konventionen

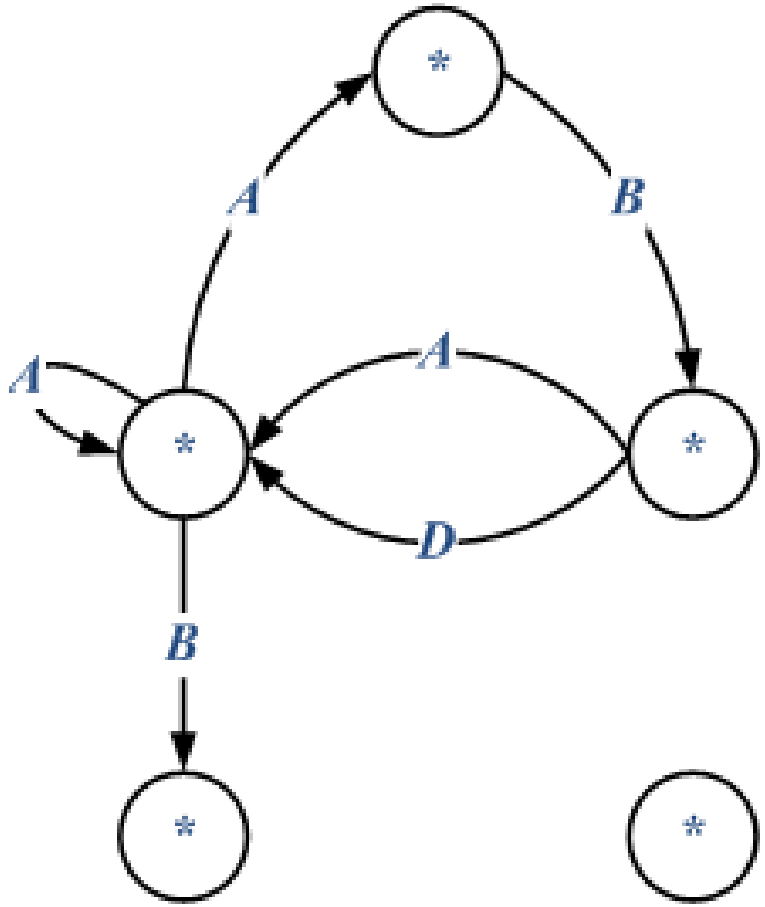
C	coloring alphabet	Markierungsalphabet
V	vertex set	Knotenmenge
E	edge set	Kantenmenge
s	source	Quelle
t	target	Ziel
l	labelling	Knotenmarkierung
m	marking	Kantenmarkierung
$v_1, v_2, v_3, \ldots$	vertices	Knoten
$e_1, e_2, e_3, \ldots$	edges	Kanten

BSP

$G = (V_G, E_G, s_G, t_G, l_G, m_G)$ mit
$V_G = \{v_1, \ldots, v_5\}$
$E_G = \{e_1, \ldots, e_6\}$

	e_1	e_2	e_3	e_4	e_5	e_6
s_G	v_2	v_2	v_2	v_3	v_4	v_4
t_G	v_1	v_2	v_3	v_4	v_2	v_2
m_G	B	A	A	B	D	A

$l_G(v_i) = *$ für $i = 1, \ldots, 5$
$* \in C_V$ und $\{A, B, D\} \subseteq C_E$



Idee der regelbasierten Graphmanipulation

- Regeln haben eine linke und eine rechte Seite. Die Seiten stehen über einen gemeinsamen Klebegraphen in Beziehung:

$$r = \langle L \supseteq K \subseteq R \rangle.$$

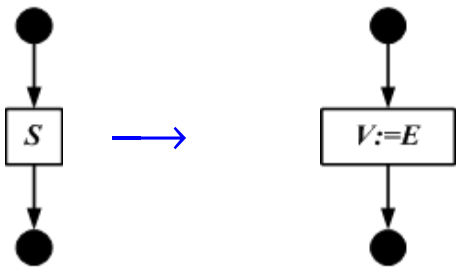
  - $L{-}K$ beschreibt die zu löschende Elemente.
  - $R{-}K$ beschreibt die hinzuzufügenden Elemente.
  - K beschreibt den zu erhaltenden Teil.
- Das Vorkommen der linken Seite L in einem Graphen G wird durch einen Graphmorphismus $g\colon L \to G$ beschrieben.
- Nebenbedingungen garantieren, dass das Ergebnis der Regelanwendung wieder ein Graph ist.

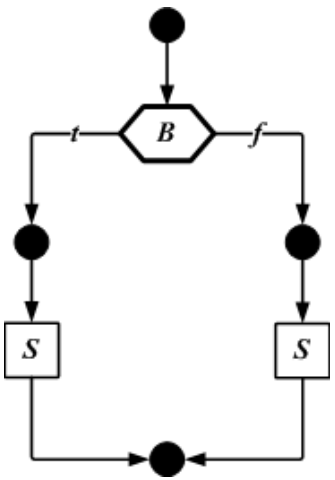
Beispiel

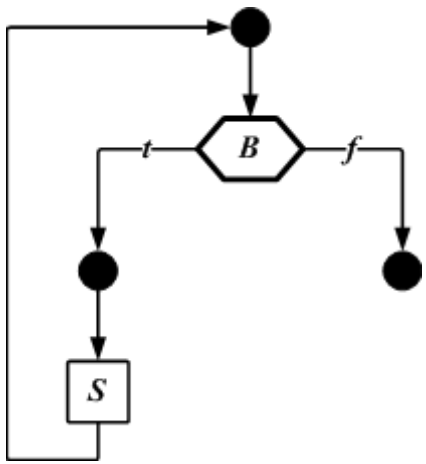
Erzeugung von Flussdiagrammen

Syntax der Anweisungen (BNF)

	Zuweisung	Sequenz	Alternative	Schleife
$S ::=$	$V := E$	$S\,;\,S$	*if* B *then* S *else* S	*while* B *do* S



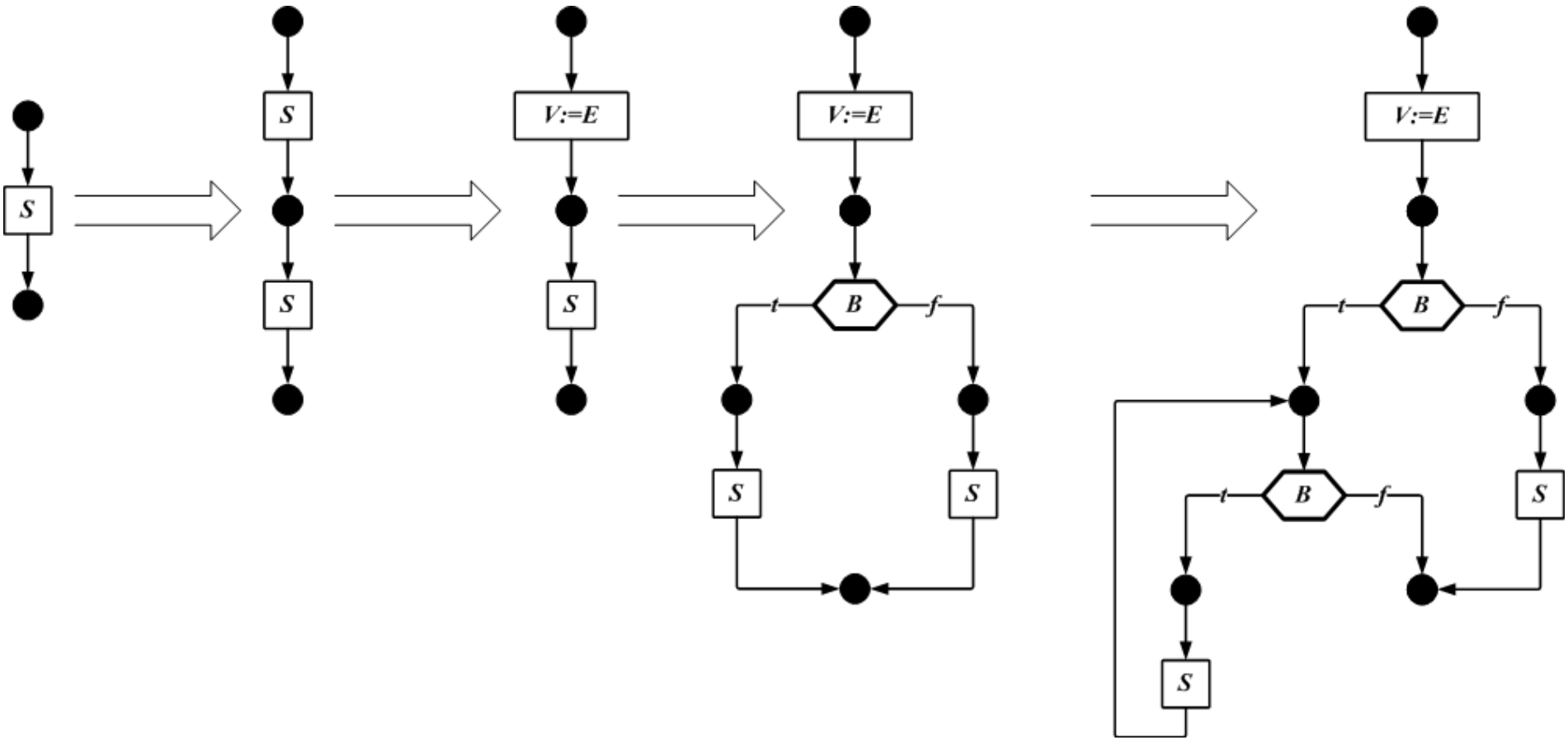




$C_V = \{\bullet, S, V := E, B\}$ und $C_E = \{$„unmarkiert“, $t, f\}$

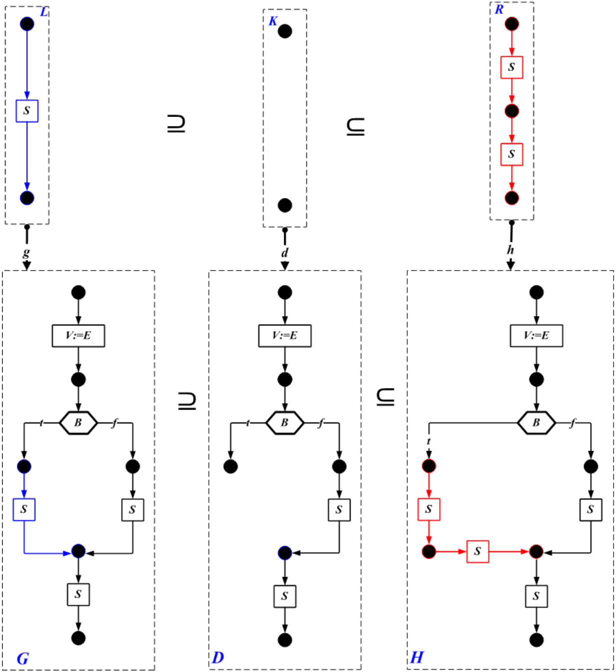
Beispiel

Ableitung



BSP

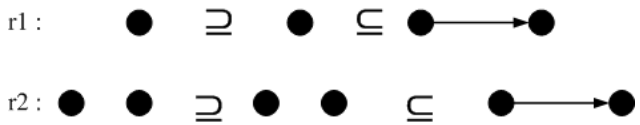
Direkte Ableitung



Aufgabe 1:

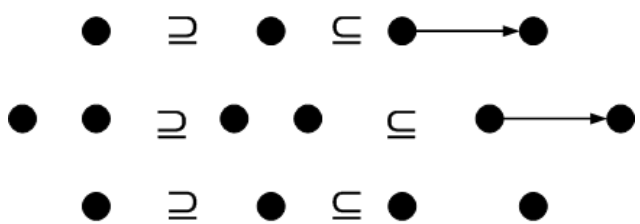
1. Geben Sie bitte Graphregeln an, die von dem Graphen der aus einem Knoten besteht ausgehend, nur zusammenhängende Graphen erzeugt.

Lösung



2. Wie müssten Sie Ihre Regelmenge verändern, um auch unzusammenhängende Graphen zu erzeugen?

Lösung



Formalisierung

- Morphismen
- Regeln
- Vorkommen
- Ableitung

Graphmorphismen

Graphmorphismen bestehen aus

struktur- und markierungerhaltenden Abbildungen

zwischen Graphen:

- bilden Knoten auf Knoten und Kanten auf Kanten ab,
- bewahren Quelle und Ziel von Kanten,
- bewahren Markierungen.

Hinweis : Graphisomorphie

Ausflug: Abbildungen

Definition

Unter einer Abbildung f von einer Menge A in eine Menge B versteht man eine Vorschrift, die jedem $a \in A$ eindeutig ein bestimmtes $b = f(a) \in B$ zuordnet: $f : A \longrightarrow B$.
Für die Elementzuordnung verwendet man die Schreibweise $a \mapsto b = f(a)$ und bezeichnet b als das Bild von a, bzw. a als ein Urbild von b.

- Eigenschaften: injektiv, surjektiv, bijektiv
- Komposition
- Identität, Umkehrabbildung
- Bild, Urbild, Kern

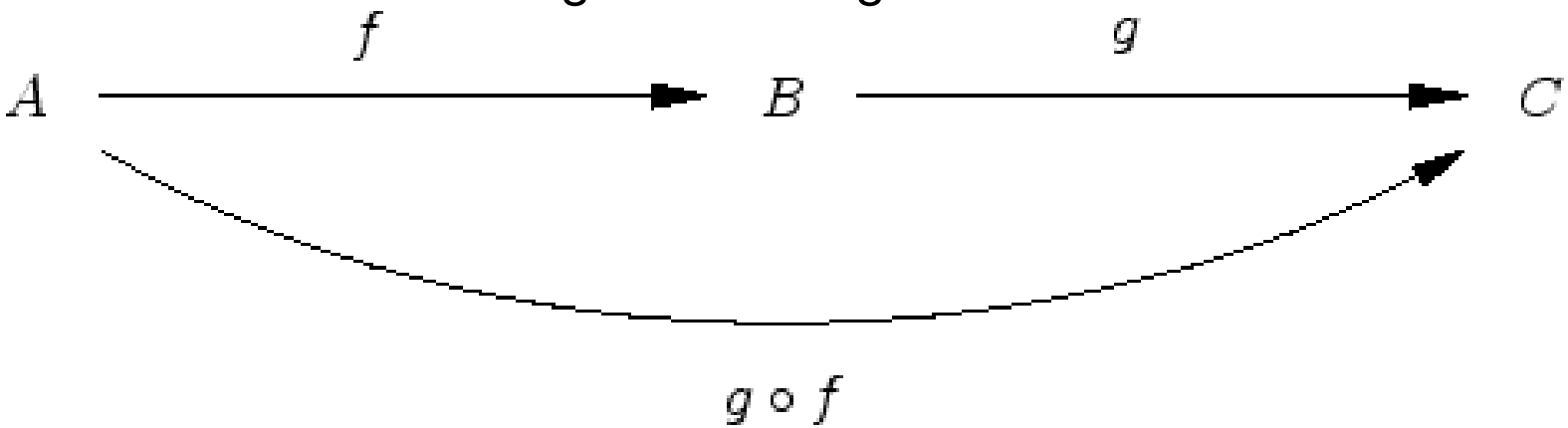
Ausflug: Abbildungen

Komposition

Die **Komposition** (oder Verknüpfung) zweier Abbildungen $f : A \to B$ und $g : B \to C$ ist durch

$$a \mapsto (g \circ f)(a) = g(f(a)), \quad a \in A$$

definiert und in dem folgendem Diagramm veranschaulicht:



Die Verknüpfung $\circ$ ist assoziativ, d.h.

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

aber offensichtlich nicht kommutativ.

Ausflug: Abbildungen

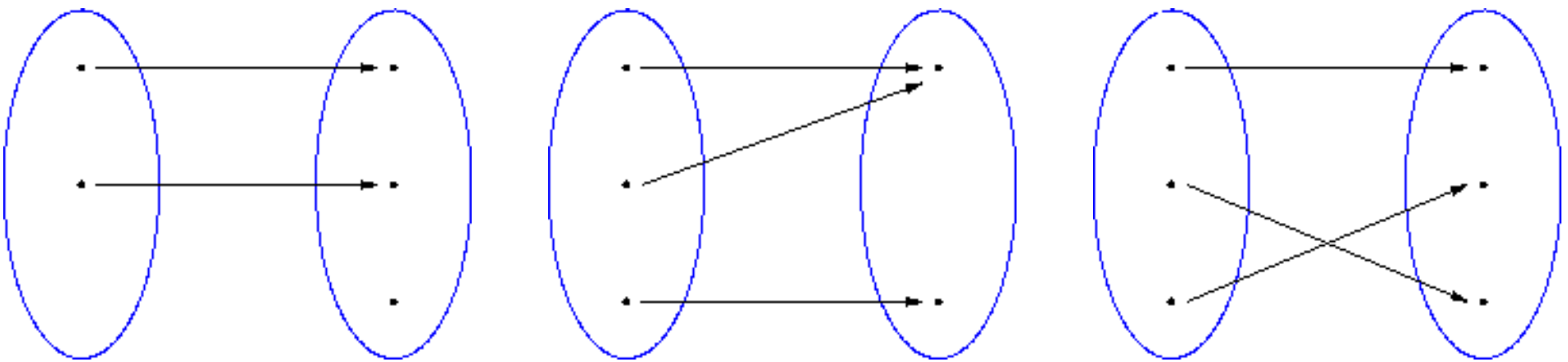
Eigenschaften

Definition

Eine Abbildung $f : A \longrightarrow B$ zwischen zwei Mengen A und B heißt

- **injektiv**, falls $f(a) \neq f(a')$ für alle $a, a' \in A$ mit $a \neq a'$
- **surjektiv**, falls es für jedes $b \in B$ ein $a \in A$ gibt mit $f(a) = b$
- **bijektiv**, falls f sowohl injektiv als auch surjektiv ist.

BSP:



Ausflug: Abbildungen

Aufgabe 2:

Gegeben sei $f : \mathbb{N}^+ \to \mathbb{N}^+$ mit $f(1) = 2$ und $f(n) = f(n-1) + 2$.

1. f ist injektiv. [X] wahr oder [ ] falsch
2. f ist surjektiv. [ ] wahr oder [X] falsch
3. $f \circ f$ ist injektiv. [X] wahr oder [ ] falsch

Graphmorphismen

Definition

Seien G und H Graphen über C. Ein **Graphmorphismus** $f : G \to H$ von G nach H ist ein Paar von Abbildungen $f = \langle f_V : V_G \to V_H, f_E : E_G \to E_H \rangle$, so dass für alle $e \in E_G$ und alle $v \in V_G$ gilt:

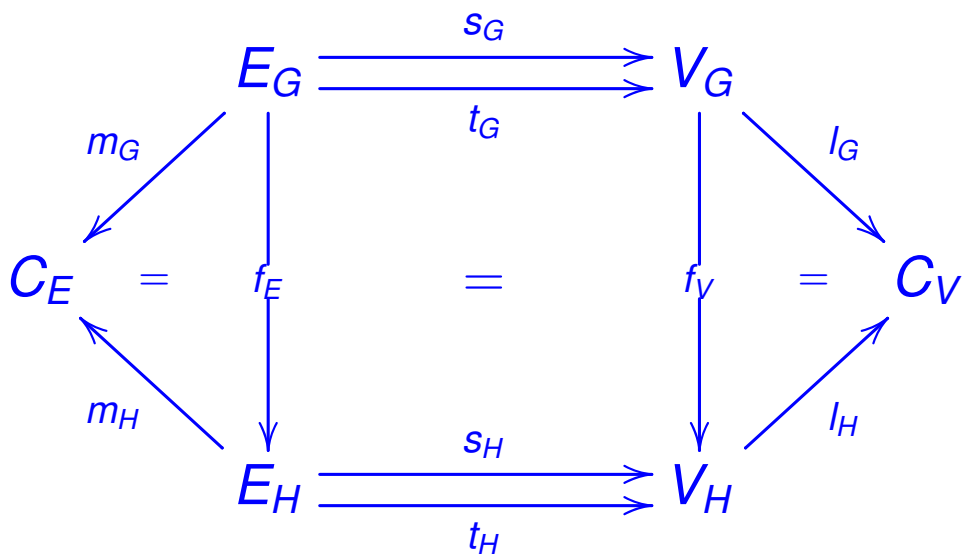
- $f_V(s_G(e)) = s_H(f_E(e))$ und $f_V(t_G(e)) = t_H(f_E(e))$
  (Bewahrung von Quelle und Ziel)
- $l_G(v) = l_H(f_V(v))$ und $m_G(e) = m_H(f_E(e))$
  (Bewahrung von Markierungen)

Diagrammatische Darstellung

Graphmorphismen

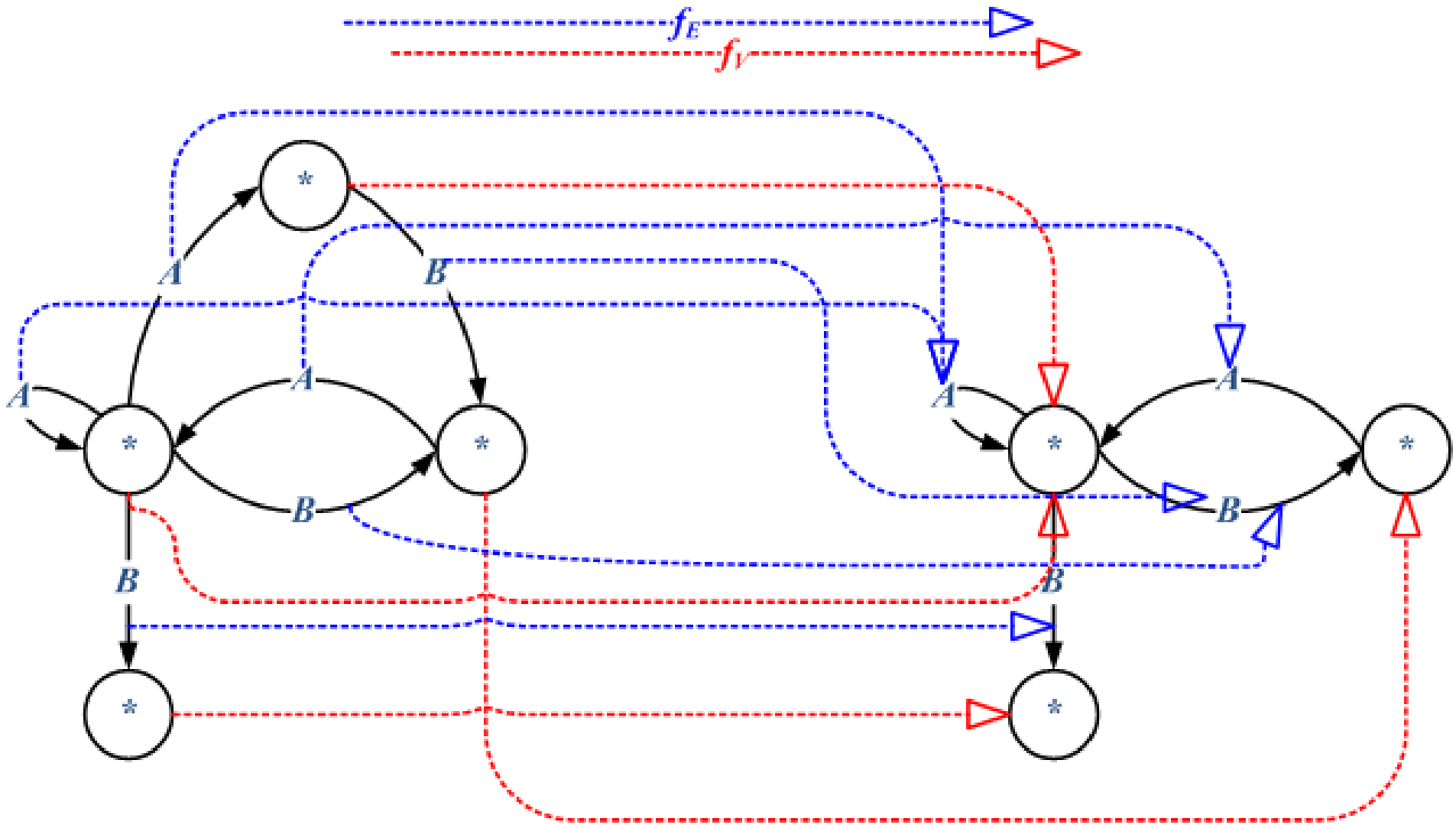
Definition

Seien G und H Graphen über C. Ein **Graphmorphismus** $f : G \to H$ von G nach H ist ein Paar von Abbildungen, so dass



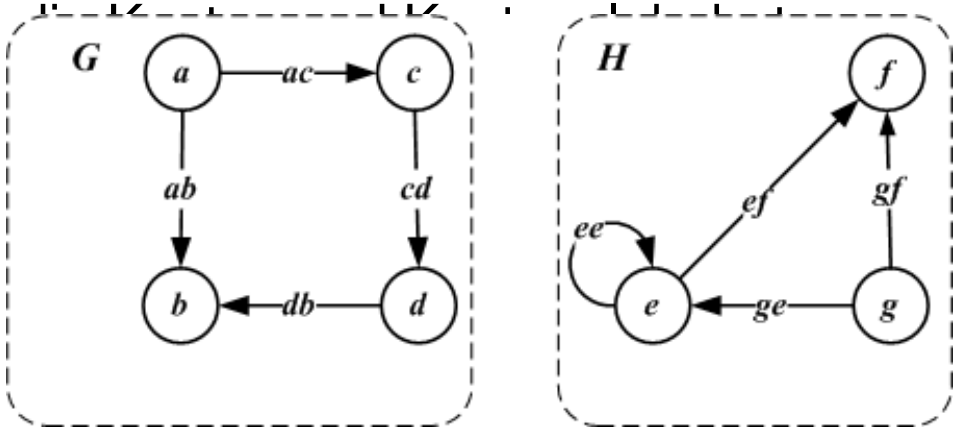
kommutiert.

Beispiel



Aufgabe 3:

Gegeben die Graphen G und H, wobei $C_V = C_E = \{*\}$ seien:



Es gibt einen Morphismus zwischen den folgenden Graphen. Welchen?
Hinweis: nicht injektive und nicht surjektive Abbildungen.

$f =< f_V, f_E >: G \to H$ mit

$f_V : V_G \to V_H$ mit

$$f_V : \begin{array}{lcl} a & \to & e \\ b & \to & e \\ c & \to & e \\ d & \to & e \end{array}$$

und

$f_E : E_G \to E_H$ mit

$$f_E : \begin{array}{lcl} ac & \to & ee \\ ab & \to & ee \\ cd & \to & ee \\ db & \to & ee \end{array}$$

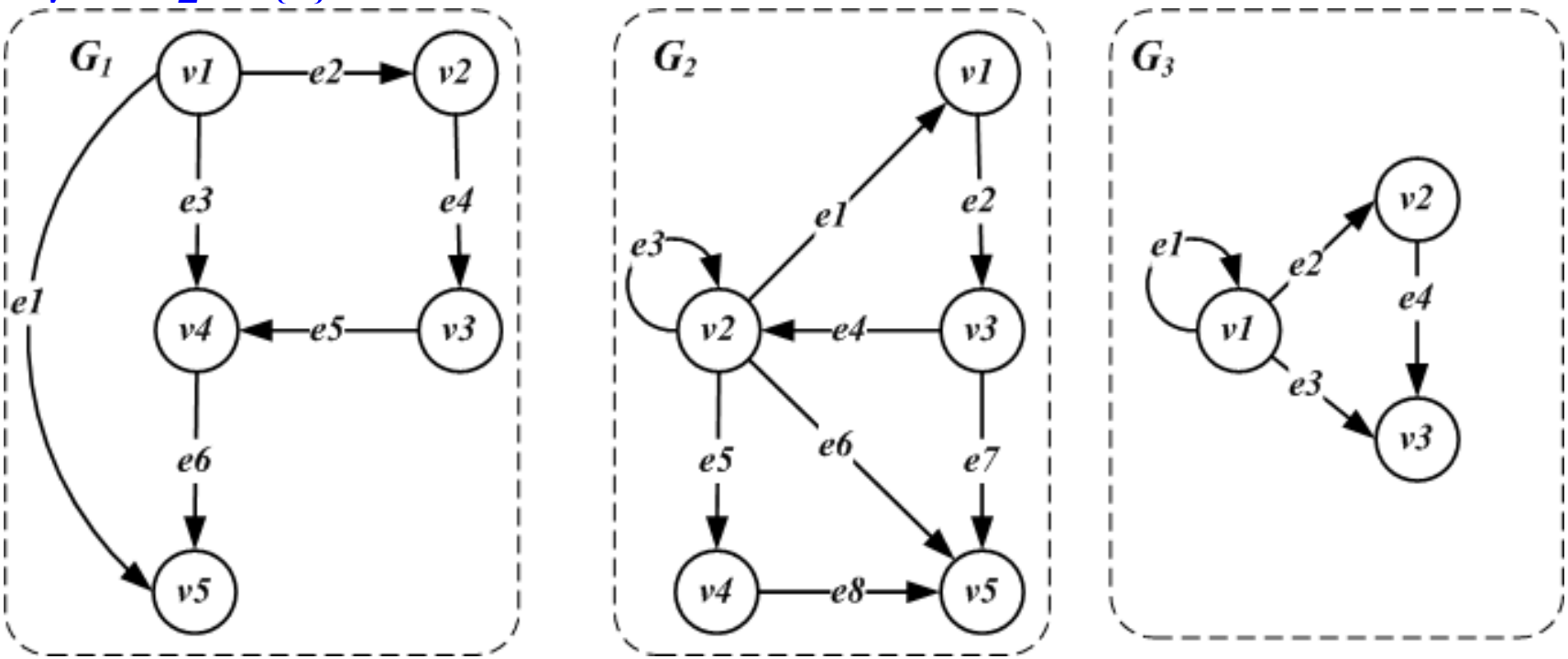
Definition

Ein Graphmorphismus $f = \langle f_V, f_E \rangle$ heißt **injektiv (surjektiv, bijektiv)**, wenn f_V und f_E injektiv (surjektiv, bijektiv) sind.
Zwei Graphmorphismen $f, g \colon G \to H$ sind **gleich**, in Zeichen $f = g$, wenn $f_V = g_V$ und $f_E = g_E$ gilt, d.h $f_V(v) = g_V(v)$ für alle $v \in V_G$ und $f_E(e) = g_E(e)$ für alle $e \in E_G$ gilt.
Ein bijektiver Graphmorphismus $f \colon G \to H$ heißt *Isomorphismus*. In diesem Fall heißen G und H **isomorph**, in Zeichen $G \cong H$.
Ein **abstrakter Graph** $[G]$ ist die Isomorphieklasse eines Graphen G:

$$[G] = \{G' \mid G \cong G'\}.$$

Aufgabe 4:

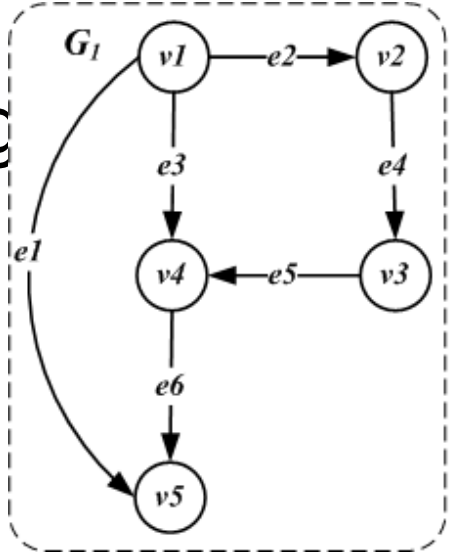
Gegeben die folgenden Graphen, wobei die Knoten und Kantenalphabete $C_V = C_E = \{*\}$ seien:

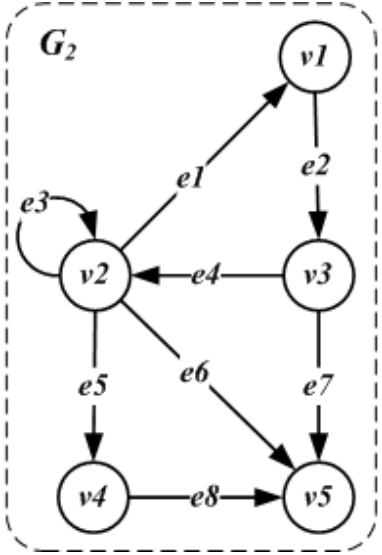


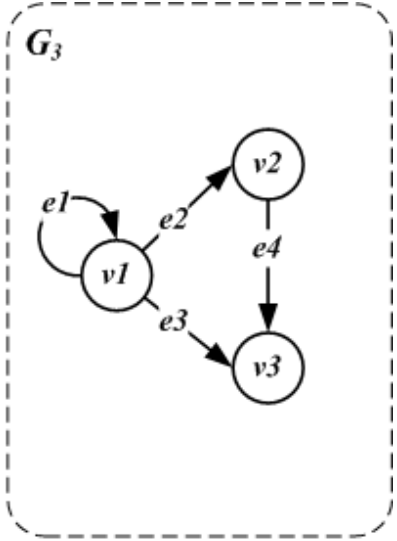
Gibt es einen Morphismus zwischen den folgenden Graphen?
Wenn ja, ist er injektiv oder surjektiv?

1. $f : G_1 \to G_2$
2. $f : G_3 \to G_1$
3. $f : G_2 \to G_3$
4. $f : G_2 \to G_1$

Lösung von Aufg







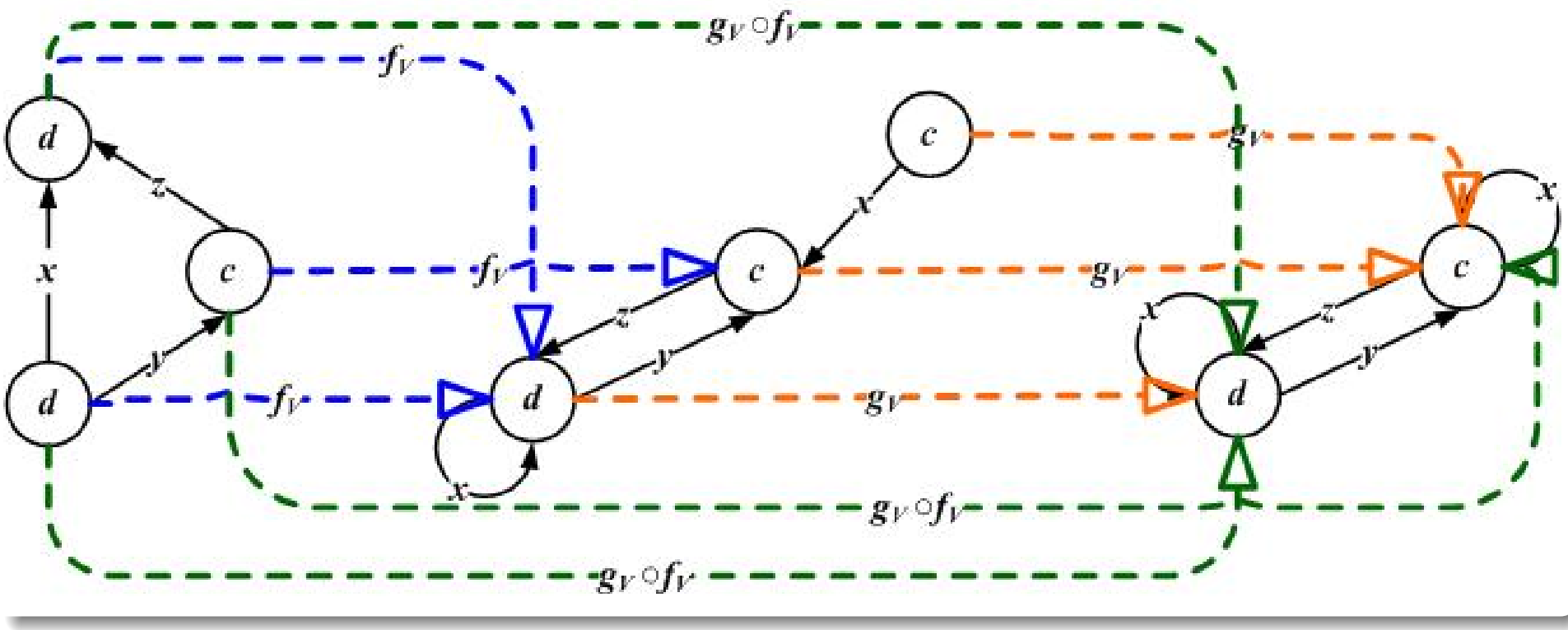
1. $f : G_1 \to G_2$
   Ja, mit
   f_V : $v1 \to v2$
   $v2 \to v1$
   $v3 \to v3$
   $v4 \to v2$
   $v5 \to v4$
   weder injektiv noch surjektiv
2. $f : G_3 \to G_1$
   Nein, den die Schlinge $e1$ kann nicht abgebildet werden.
3. $f : G_2 \to G_3$
   Ja, mit
   f_V : $v1 \to v1$
   $v2 \to v1$
   $v3 \to v1$
   $v4 \to v2$
   $v5 \to v3$
   weder injektiv noch surjektiv
4. $f : G_2 \to G_1$
   Nein, denn die Schlinge $e3$ kann nicht abgebildet werden.

Komposition von Graphmorphismen

Definition

Seien $f : G \to H$ und $g : H \to I$ Graphmorphismen.
Dann ist die **Komposition** $g \circ f : G \to I$ von f und g definiert durch
$g \circ f := (g_V \circ f_V, g_E \circ f_E)$ die Komposition der Kanten- und Knotenabbildungen.

BSP:



Graphersetzungsregel

Definition

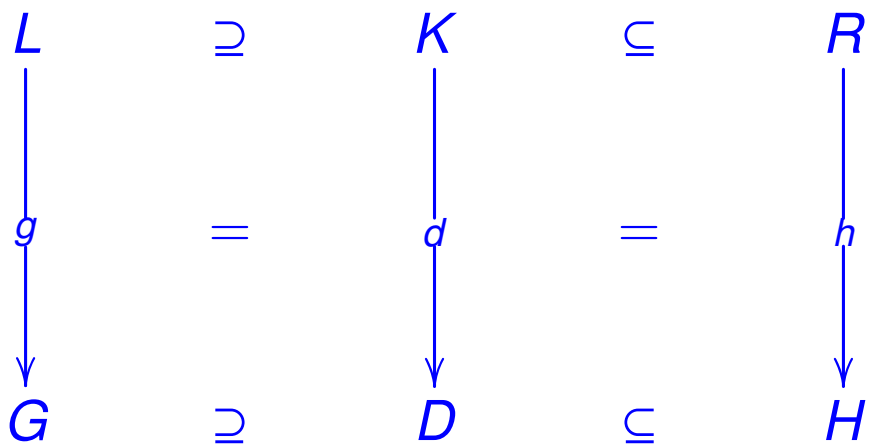
Eine **Graphersetzungsregel** (kurz **Regel**) über C hat die Form

$$r = \langle L \supseteq K \subseteq R \rangle$$

wobei L, K, und R Graphen über C sind.
L heißt **linke Seite**, R **rechte Seite** und K **Klebegraph** von r.

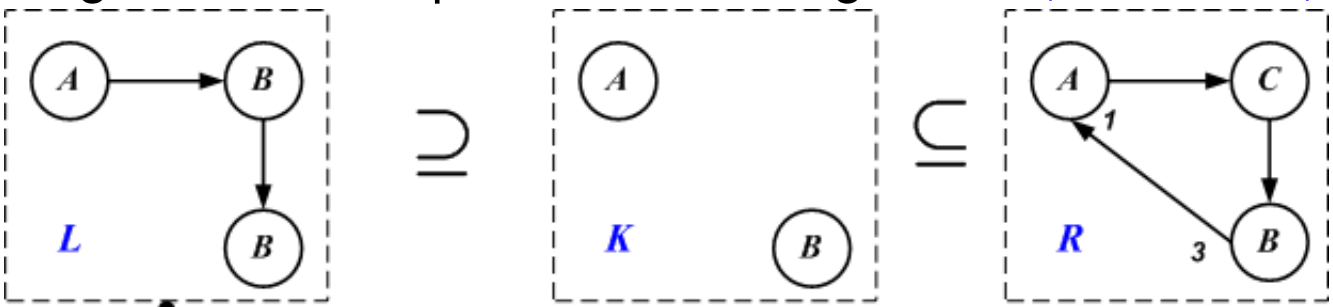
Anwendung von $r = \langle L \supseteq K \subseteq R \rangle$ auf G (skizziert):

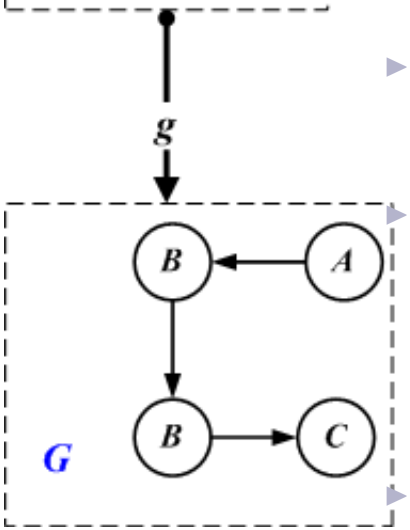
1. Wähle ein Vorkommen von L in G,
   d.h. einen Graphmorphismus $g\colon L \to G$.
2. Überprüfe die Kontakt- und Identifikationsbedingung.
3. Lösche $g(L–K)$,
   d.h. alle Kanten in $g_E(E_L–E_K)$ und alle Knoten in $g_V(V_L–V_K)$.
   Zwischenergebnis: $D = G – g(L–K)$.
4. Füge $R–K$ hinzu,
   d.h. alle Knoten in $V_R – V_K$ und alle Kanten in $E_R–E_K$.
   Ergebnis: $H = D+(R–K)$.



Aufgabe 5:

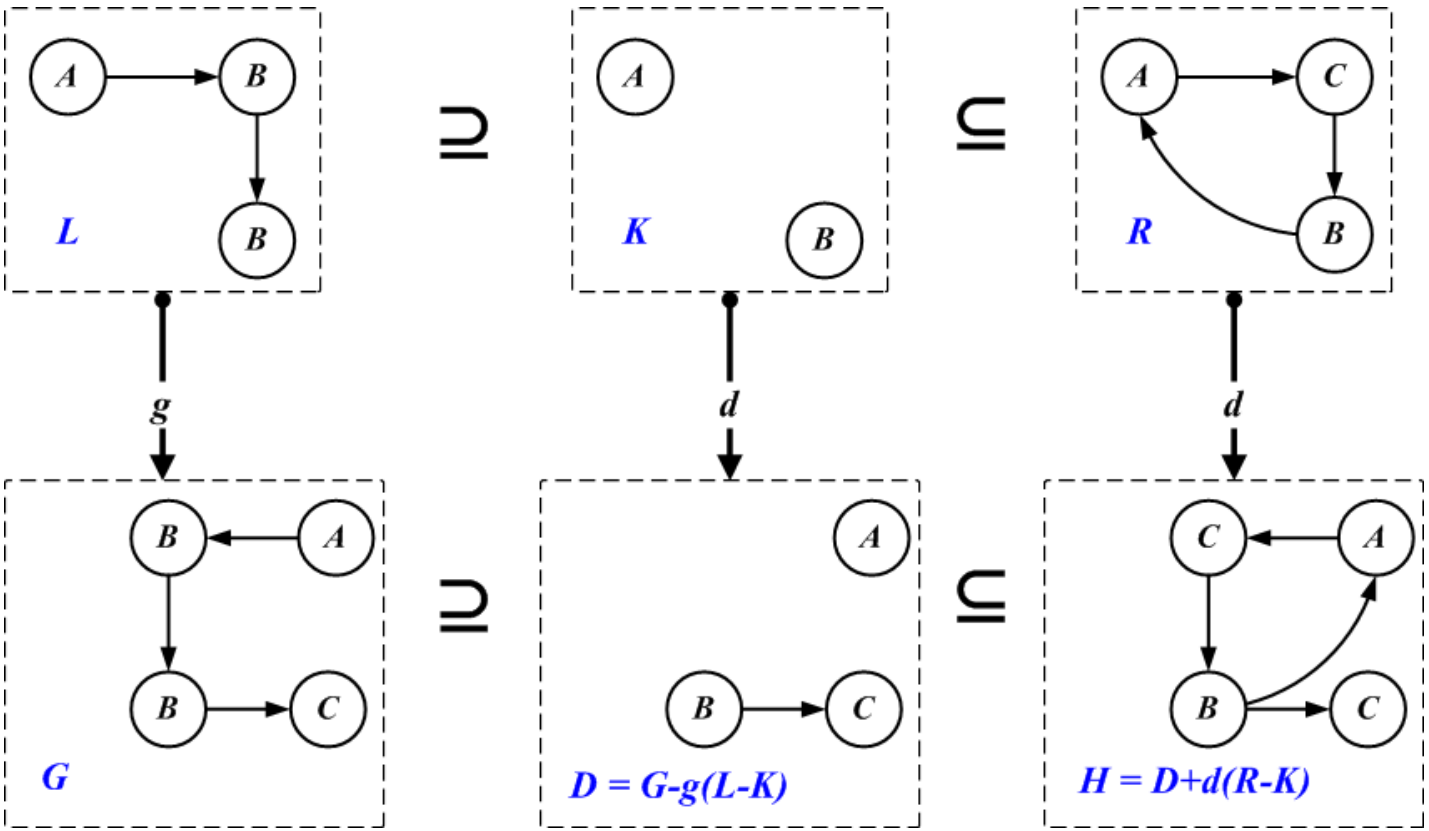
Gegeben der Graph G und die Regel: $r = \langle L \supseteq K \subseteq R \rangle$





- Wählen Sie ein Vorkommen von L in G,
  d.h. einen Graphmorphismus $g\colon L \to G$.
- Lösche $g(L–K)$,
  d.h. alle Kanten in $g_E(E_L–E_K)$ und alle Knoten in $g_V(V_L–V_K)$.
  Zwischenergebnis: $D = G - g(L–K)$.
- Füge $R–K$ hinzu,
  d.h. alle Knoten in $V_R - V_K$ und alle Kanten in $E_R–E_K$.
  Ergebnis: $H = D+(R–K)$.
- Geht das immer?

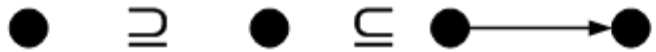
Lösung von Aufgabe 5



Nein, geht nicht immer.
Sie haben jetzt eine direkte Ableitung $G \Longrightarrow H$

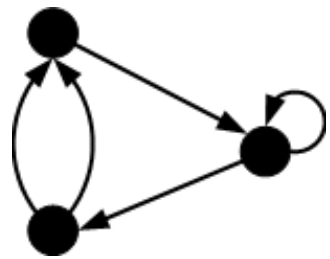
Aufgabe 6:

Gegeben die Graphregeln, die von dem Graphen der aus einem Knoten besteht ausgehend, nur zusammenhängende Graphen erzeugten:

r1 : 

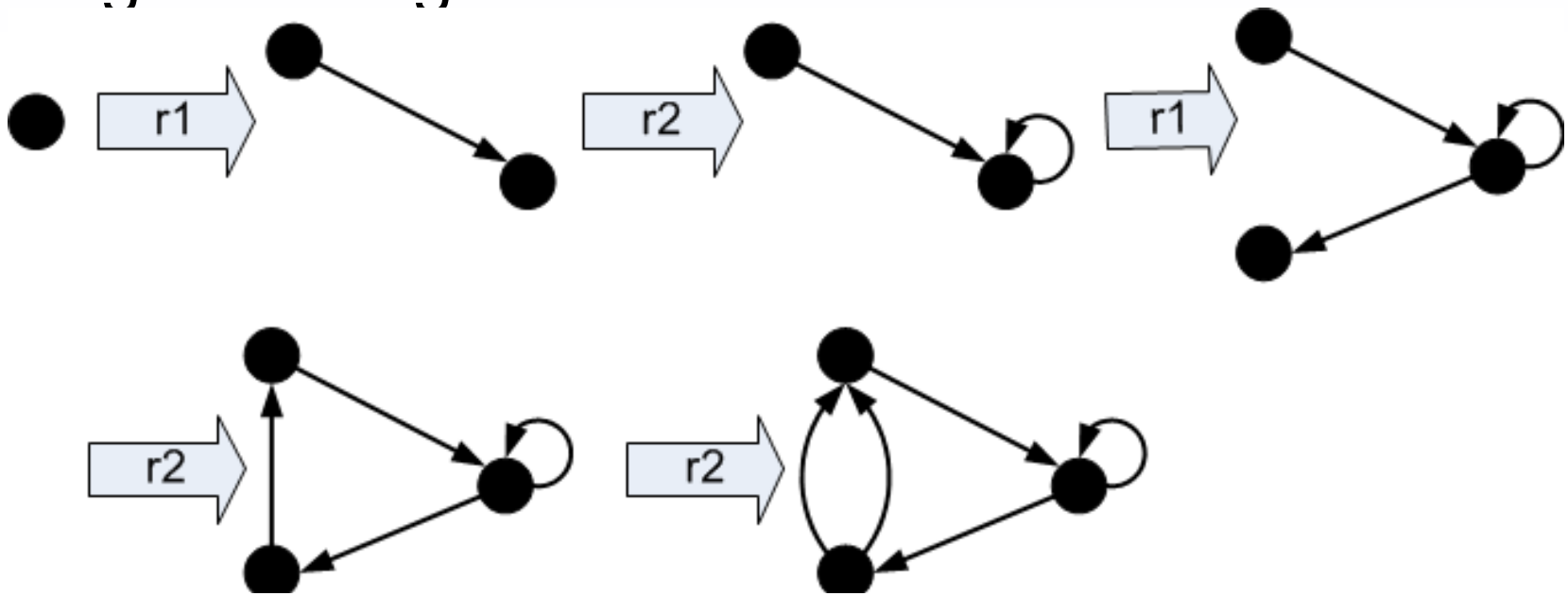
r2 : 

1. Geben Sie bitte die Ableitungen an, um diesen Multigraphen zu erzeugen.
2. Geben Sie bitte den Ableitungsschritt, der die Schlinge erzeugt, explizit an.

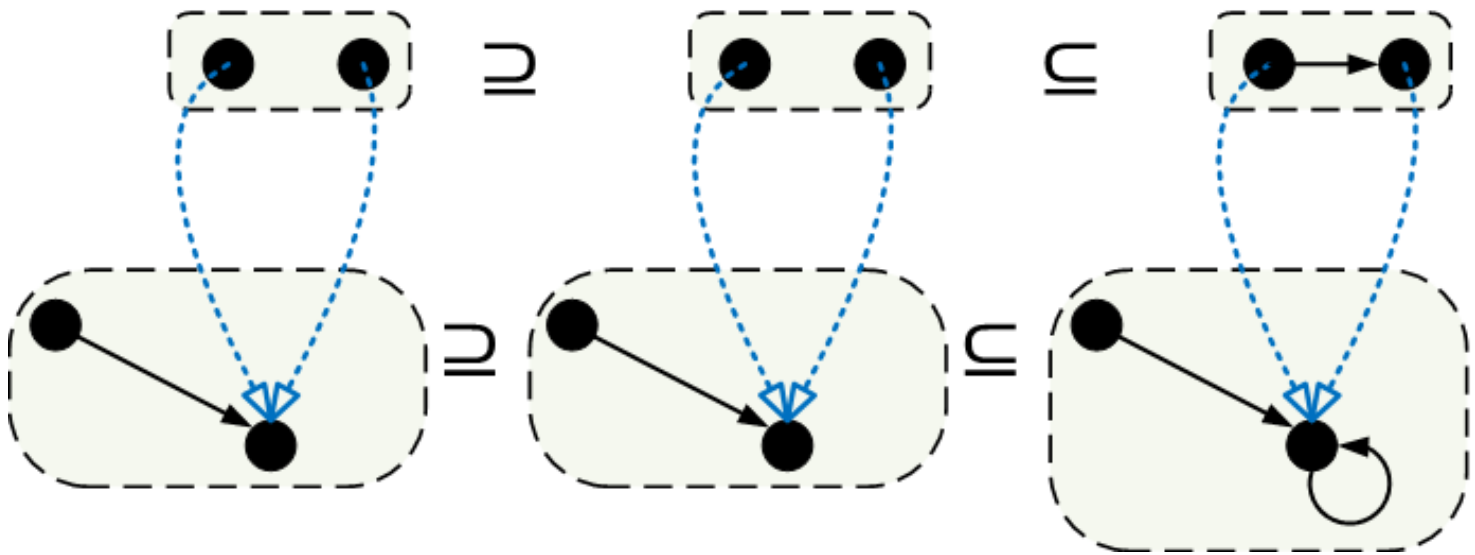


Lösung von Aufgabe 6

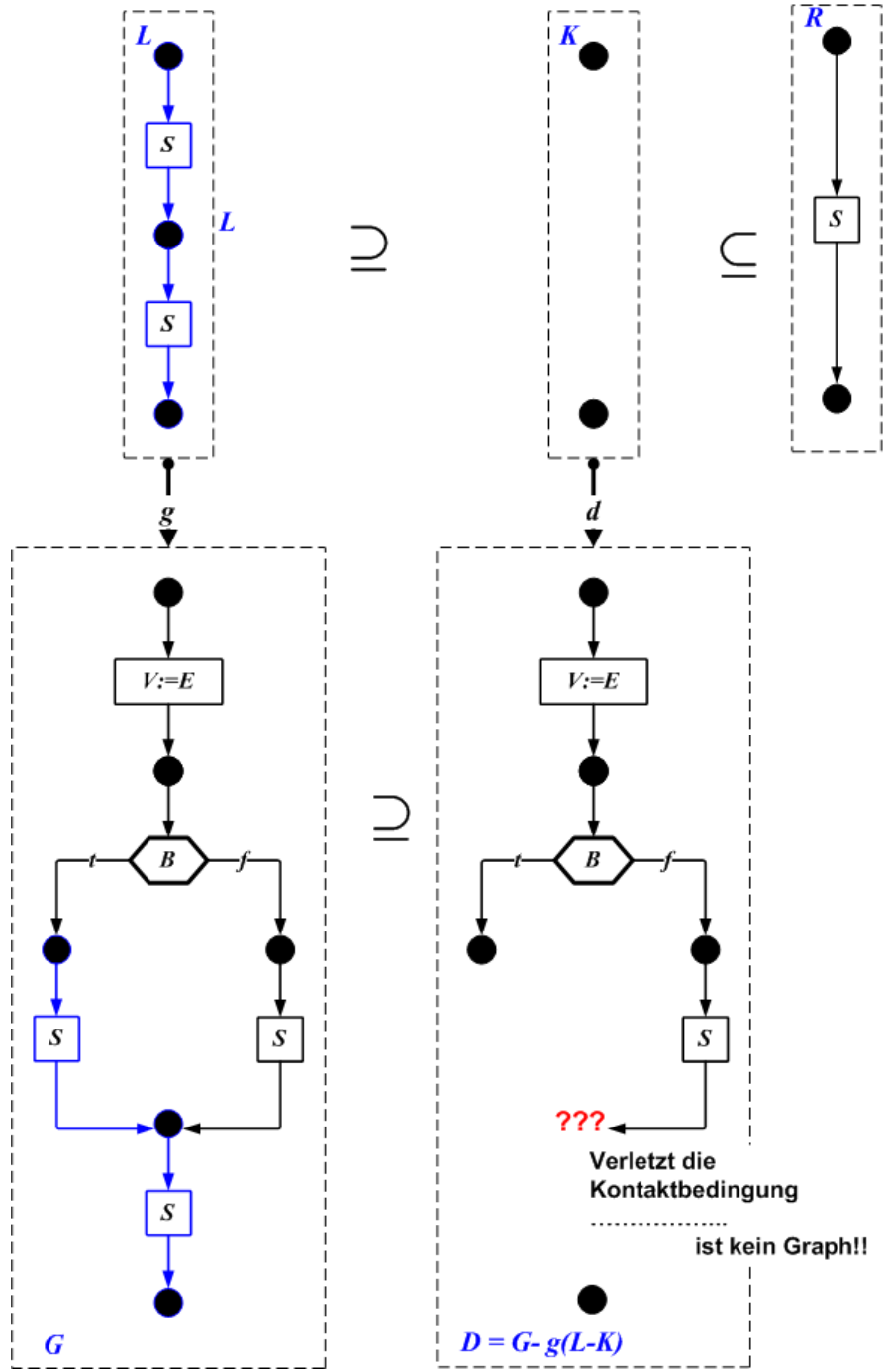
1.



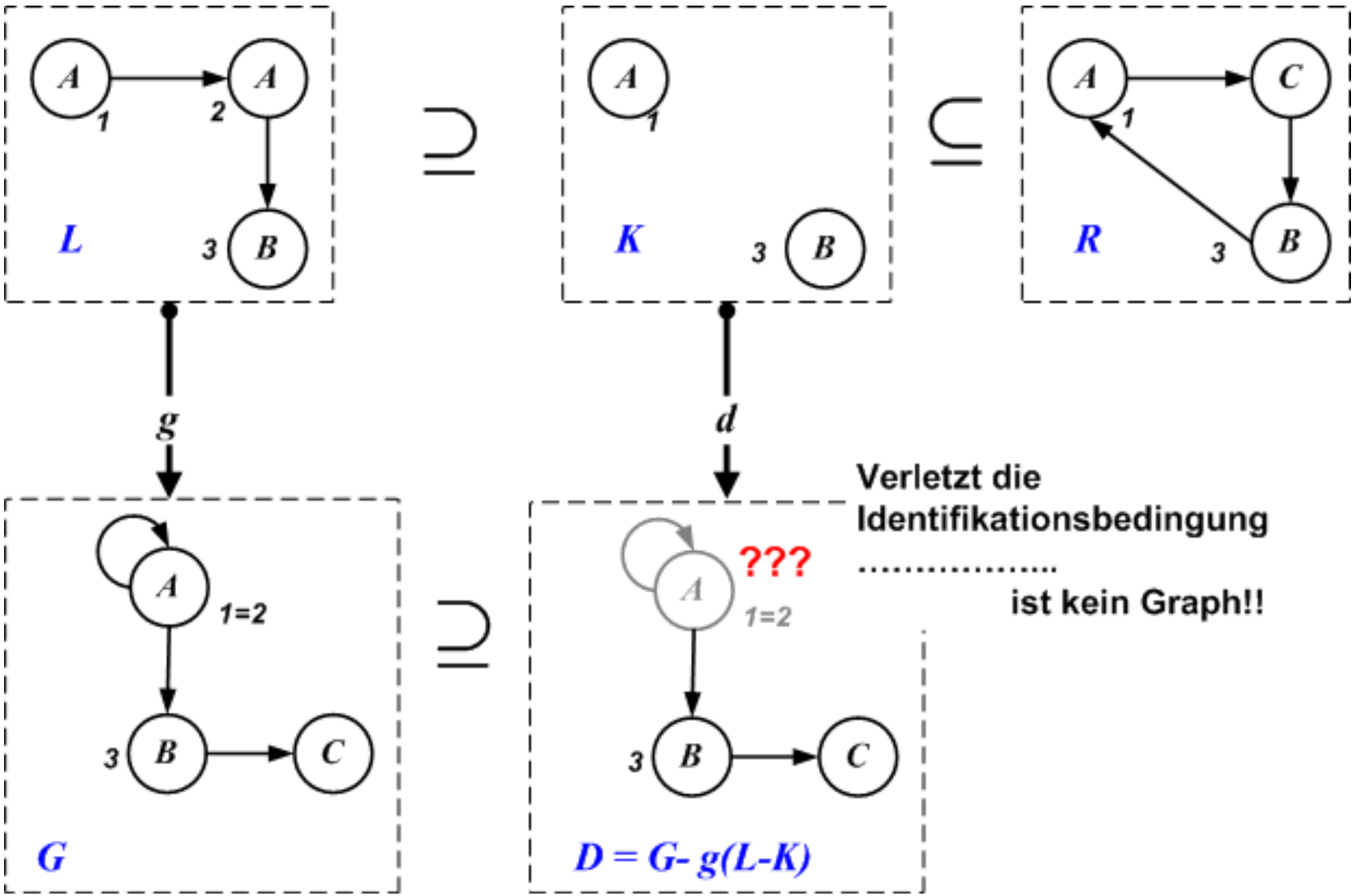
2.



Löschen ohne Kontaktbedingung



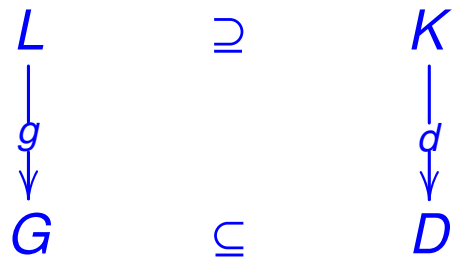
Löschen ohne Identifikationsbedingung



Löschen

Satz

Seien L und K Graphen mit $K \subseteq L$ und $g\colon L \to G$ ein Graphmorphismus, der die folgenden Bedingungen erfüllt:



- **Kontaktbedingung**:
  Für alle $e \in E_G - g_E(E_L)$: $s_G(e), t_G(e) \in V_G - g_V(V_L - V_K)$.
- **Identifikationsbed.**:
  Für alle $x, y \in L$: $g(x) = g(y)$ impl. $x = y$ oder $x, y \in K$.
  ($x \in L$ steht für $x \in V_L \cup E_L$)

Dann ist $D = (V_D, E_D, s_D, t_D, l_D, m_D)$ ein Teilgraph von G mit:

$$V_D = V_G - g_V(V_L - V_K) \text{ und } E_D = E_G - g_E(E_L - E_K)$$
$$s_D = s_G|_{E_D} \text{ und } t_D = t_G|_{E_D}$$
$$l_D = l_G|_{V_D} \text{ und } m_D = m_G|_{E_D}$$

Beweisidee

- Gegeben die Konstruktion von D
- Nachweis, dass D ein Graph ist,
  also für alle $e \in E_D$: $s_G(e), t_G(e) \in V_D$.
  1. Kante nur in G
  2. Kante auch in K
- Markierungen sind trivialerweise Abbildungen, da die Markierungsalphabete nicht verändert werden
- $s_D, t_D : E_D \to V_D$ sind Abbildungen.

Hinzufügen/Verkleben

Satz

Seien K und R Graphen mit $K \subseteq R$ und $d\colon K \to D$ ein Graphmorphismus.
Dann ist $H = (V_H, E_H, s_H, t_H, l_H, m_H)$ mit

$$\begin{array}{ccc} K & \subseteq & R \\ \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle h} \\ D & \subseteq & H \end{array}$$

$$V_H = V_D + (V_R - V_K)$$
$$E_H = E_D + (E_R - E_K)$$
$$s_H\colon E_H \to V_H \text{ mit } s_H(e) = \begin{cases} s_D(e) & \text{für } e \in E_D \\ d_V(s_R(e)) & \text{für } e \in E_R - E_K \text{ mit } s_R(e) \in V_K \\ s_R(e) & \text{sonst} \end{cases}$$
$$t_H\colon E_H \to V_H \text{ analog zu } s_H$$
$$l_H\colon V_H \to C_V \text{ mit } l_H(v) = \begin{cases} l_D(v) & \text{für } v \in V_D \\ l_R(v) & \text{sonst} \end{cases}$$
$$m_H\colon E_H \to C_E \text{ analog zu } l_H$$

ein Graph, die **Verklebung** von D und R gemäß d.

Eigenschaften der Verklebung

Satz

Seien K und R Graphen mit $K \subseteq R$ und $d\colon K \to D$ ein Graphmorphismus.

$$\begin{array}{ccc} K & \subseteq & R \\ \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle h} \\ D & \subseteq & H \end{array}$$

Dann hat die Verklebung H von D und R gemäß d folgende Eigenschaften:

1. $D \subseteq H$
2. $h\colon R \to H$ mit $h(x) = \begin{cases} x & \text{für } x \in R\text{–}K \\ d(x) & \text{sonst} \end{cases}$ ist ein Graphmorphismus.
3. $d\colon K \to D$ ist die Einschränkung von $h\colon R \to H$ auf K und D.

Direkte Ableitung

Definition

Sei G ein Graph, $r = \langle L \supseteq K \subseteq R \rangle$ eine Regel, $g : L \to G$ ein Graphmorphismus, der die Kontakt- und Identifikationsbedingung erfüllt, und M isomorph zu dem Ergebnis der Anwendung von r auf G:

$$\begin{array}{ccccc} L & \supseteq & K & \subseteq & R \\ \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle h} \\ G & \supseteq & D & \subseteq & H \xrightarrow{\cong} M \end{array}$$

$G \overset{(r,g)}{\Longrightarrow} M$ heisst **direkte Ableitung** von G nach M bezüglich r und g. Hierfür schreiben wir auch $G \overset{r}{\Longrightarrow} M$ oder kurz $G \Longrightarrow M$.

Ableitung

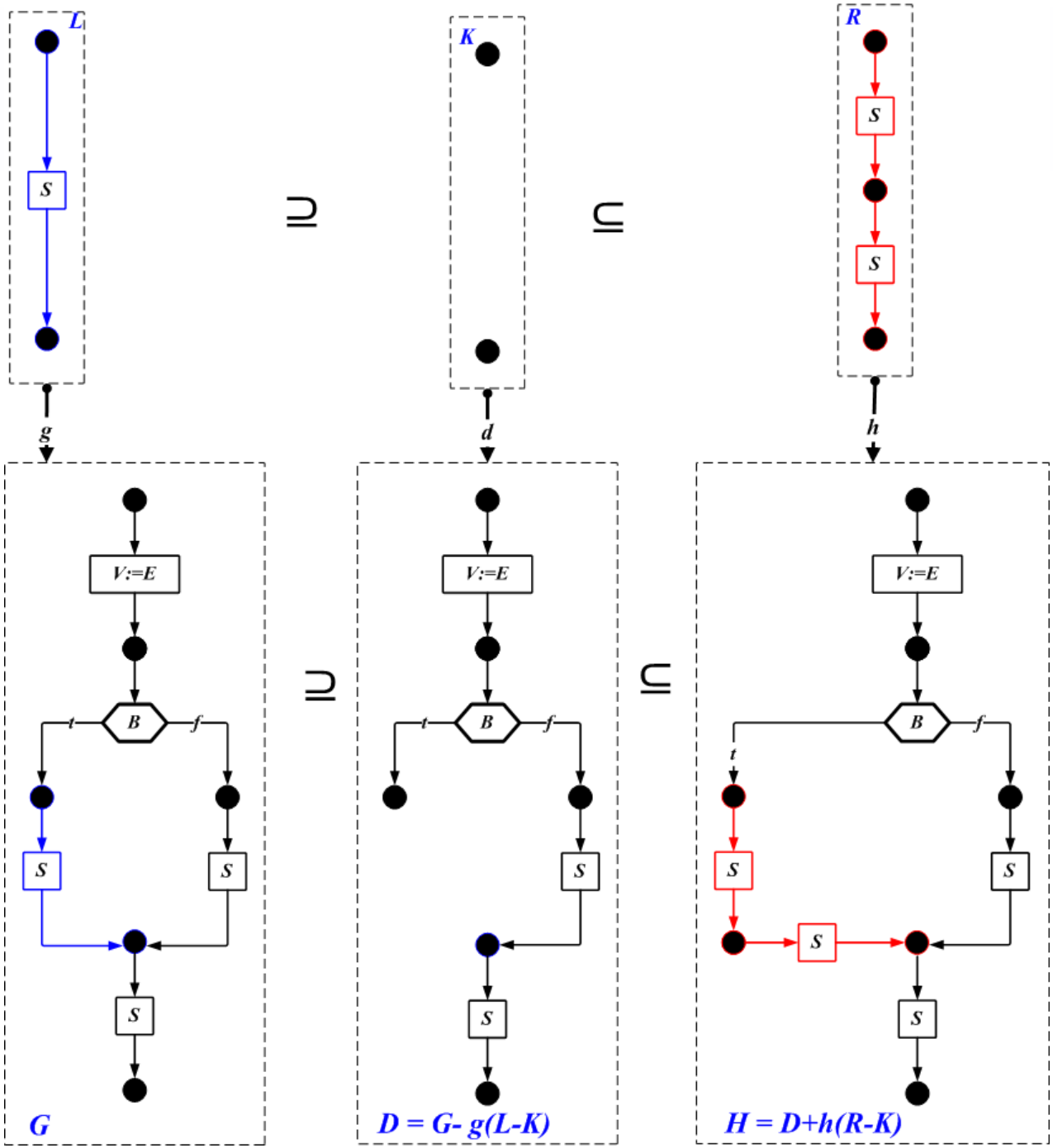
Definition

$G \Longrightarrow M$ heisst **Ableitung** von G nach M, wenn

- $G \cong M$ oder
- wenn es eine Folge direkter Ableitungen der Form $G = G_0 \overset{r_1,g_1}{\Longrightarrow} \ldots \overset{r_n,g_n}{\Longrightarrow} G_n \cong M$ gibt. Wir schreiben für eine Regelmenge $\mathcal{R}$ auch $G \overset{\mathcal{R}}{\Longrightarrow} M$ falls $r_1, \ldots, r_n \in \mathcal{R}$.

BSP

Direkte Ableitung



Konzepte

	Fokus	typische Ergänzungen
Graph-Ersetzungs-Systeme[1]	Modellierung von Systemen durch direkte, sequentielle oder pararalle Ableitungen	Kontrollstrukturen, wie **negative Anwendungsbedingungen**[3] Typgraphen Transformationseinheiten
Graph-Grammatiken[2]	Beschreibung aller ableitbaren Graphen, also **Graphsprachen Mächigkeit**	**Terminal- und Nonterminalsymbole**

[1] auch Graphtransformationssysteme (engl graph transformation (rewriting) systems)
[2] (engl graph grammars)
[3] (engl Negative Application Conditions (NACs))

Graphersetzungssystem

Definition

Ein **Graphersetzungssystem** ist ein System $\langle C, \mathcal{R} \rangle$, wobei C ein Paar von Markierungsalphabeten und $\mathcal{R}$ eine Menge von Graphersetzungsregeln über C ist.

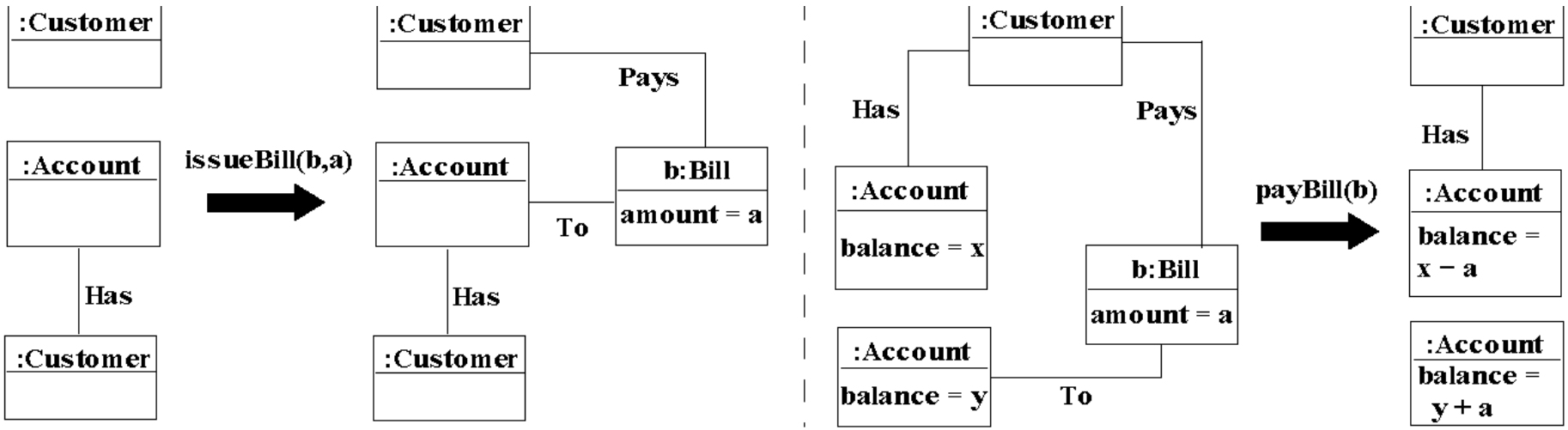
Graphersetzungssystem

Definition

Ein **Graphersetzungssystem** ist ein System $\langle C, \mathcal{R} \rangle$, wobei C ein Paar von Markierungsalphabeten und $\mathcal{R}$ eine Menge von Graphersetzungsregeln über C ist.

- Modellierung von Systemen mit
  - hoher Nebenläufigkeit
  - unendlichem Zustandsraum
  - dynamischem Erzeugen/Löschen von Objekten
  - variabler Topologie
    (Systemstruktur verändert sich während der Laufzeit)
  - Mobilität
    (mobiler Code oder mobile Prozesse)
- Transformation von
  - Modellen
  - Sprachen, Syntaxbäumen
  - Programmcode
- Semantik von
  - Diagrammen
  - visuellen Sprachen

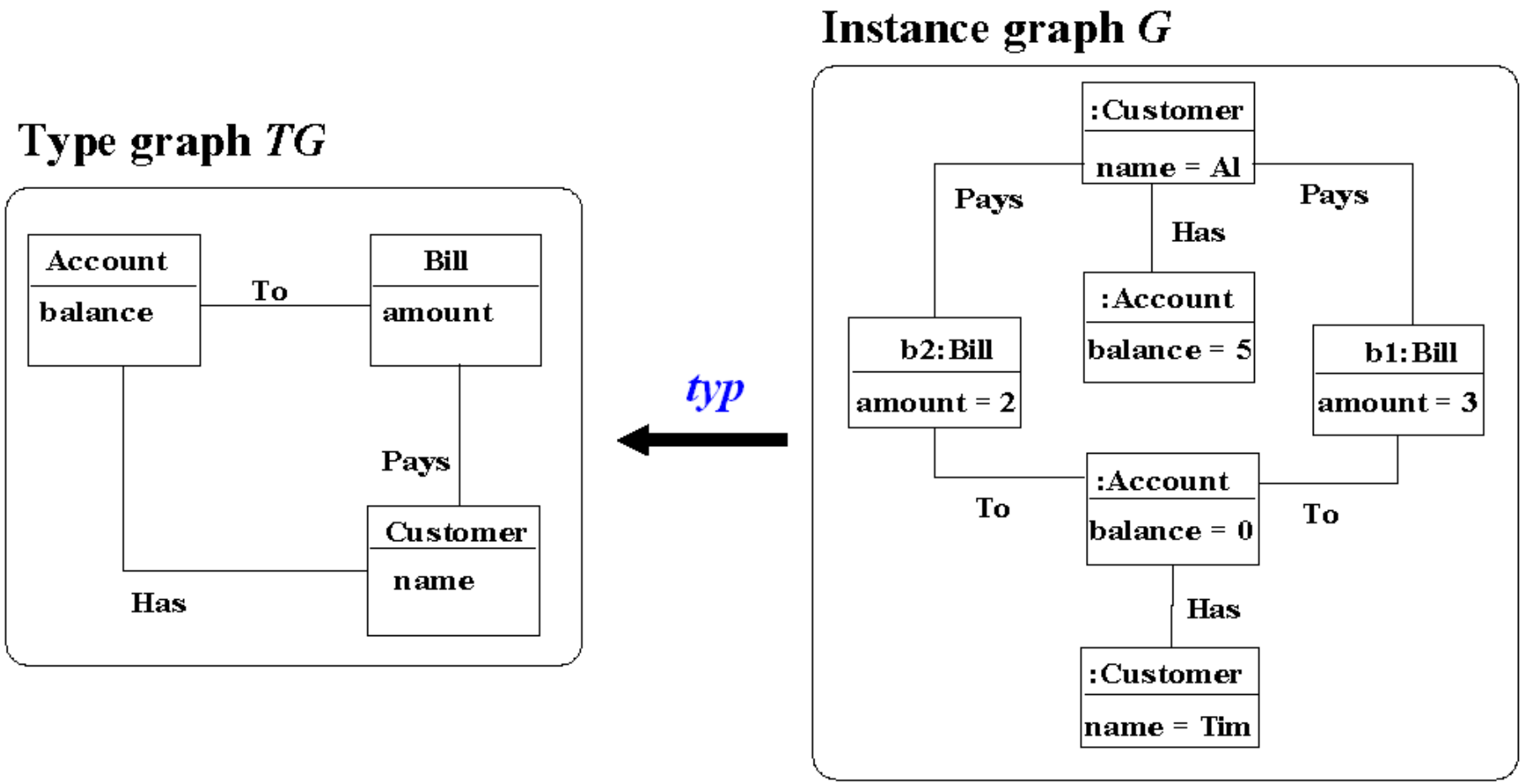
BSP: Rechnungswesen



Typgraph

Definition

Gegeben ein Typgraph *TG*, dann ist *G* ein mit *TG* getypter graph, wenn es eine Graphmorphismus $typ : G \rightarrow TG$ gibt.



Negative Anwendungsbedingungen

Definition

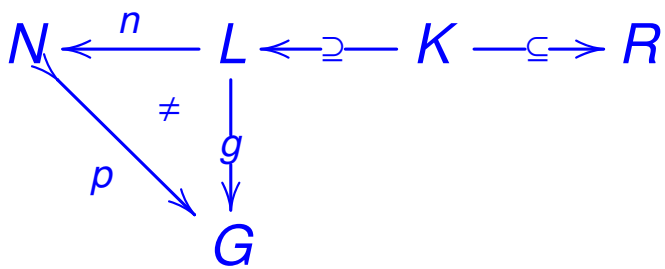
Sei $r = \langle L \supseteq K \subseteq R \rangle$ eine Regel. Eine **negative Anwendungsbedingung** (NAC) für r ist ein Graph N mit einem Morphismus $n : L \to N$.

Intuition: Es wird verlangt, dass N nicht in der Umgebung von L vorkommt, wenn die Regel auf einen Graph G angewandt wird.

Vorkommen erfüllt NAC

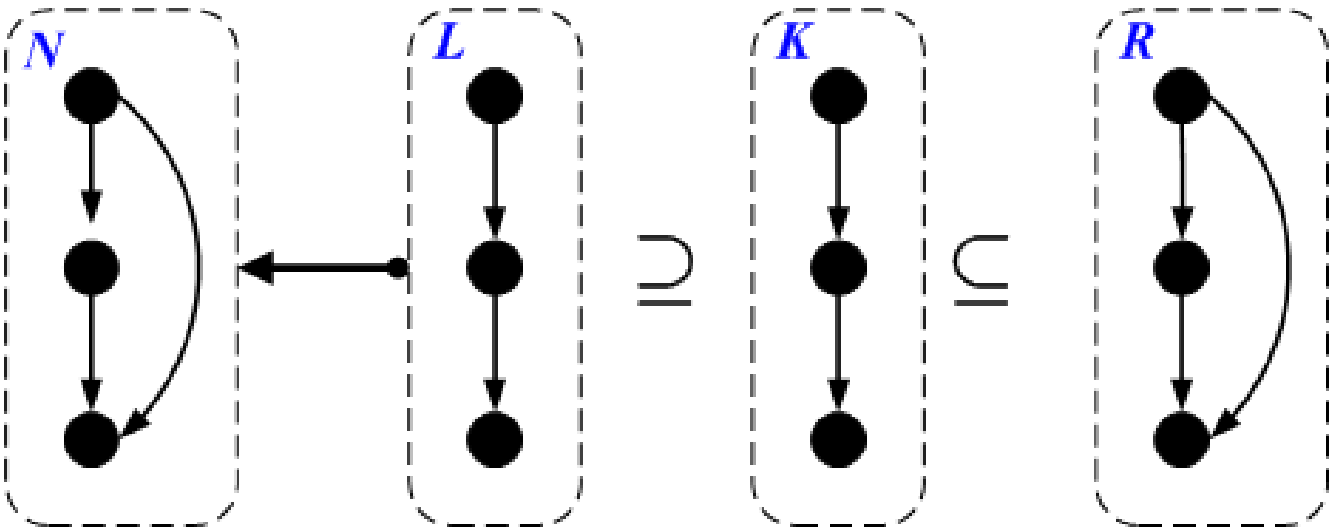
Definition

Sei $r = \langle L \supseteq K \subseteq R \rangle$ eine Regel und $n : L \to N$ eine negative Anwendungsbedingung und $g : L \to G$ ein Vorkommen von L in G. g erfüllt die negative Anwendungsbedingung, wenn es **keinen injektiven Morphismus** $p : N \to G$ gibt mit $p \circ n = g$.
Also für alle $p : N \to G$:

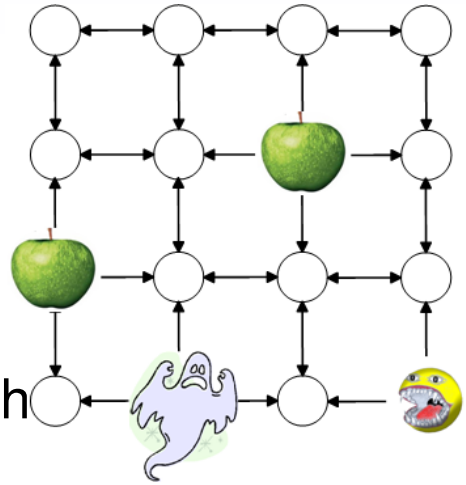


BSP: Transitiven Hülle

Beispiel: Bilden der transitiven Hülle eines (gerichteten) Graphen
Idee: Zwei Knoten, die indirekt über einen dritten Knoten verbunden sind, direkt mit einer Kante verbinden.
Dies soll aber nur dann geschehen, wenn diese Verbindung nicht bereits existiert.
Diese Regel soll so lange wie möglich angewandt werden.



Aufgabe 7:



Modellieren Sie bitte PAC-Man
als getyptes Graphersyetzungssystem mit NACs:

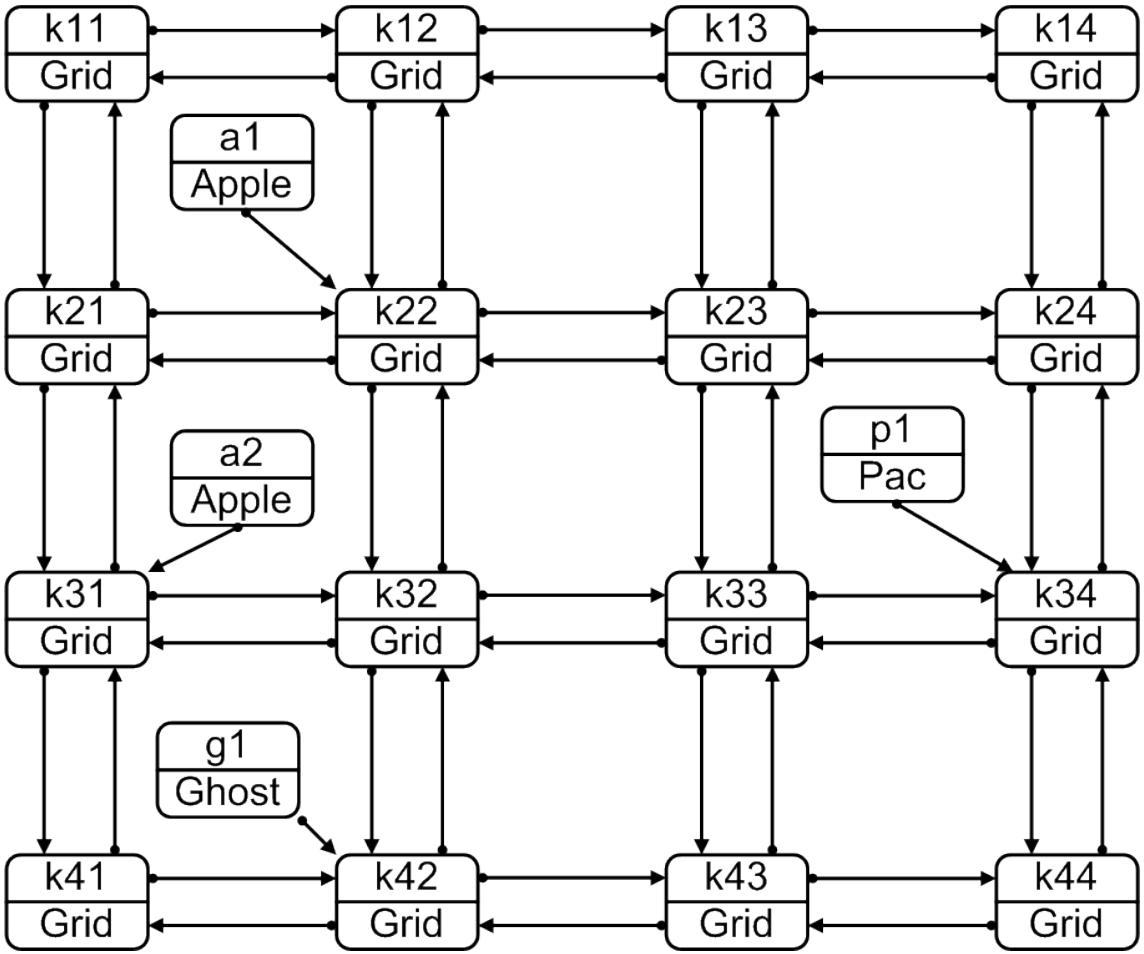
1. Gitter mit Pacman, Äpfeln und Gespenst als Startgraph
2. Typgraph mit Positionsknoten,
3. Knoten des Gitters sind mögliche Felder
4. Knoten für Pacman, die Äpfel und das Gespenst
5. Regeln für
   - Bewegen des Pacman und des Gespensts
   - Fressen des Apfels (durch Pacman)
   - Fressen Pacmans (durch Gespenst)

   wobei
   - Pacman nicht auf ein Feld mit dem Gespenst darf und
   - das Gespenst nicht auf ein Feld mit dem Apfel darf.

Lösung von Aufgabe 7

Lösung

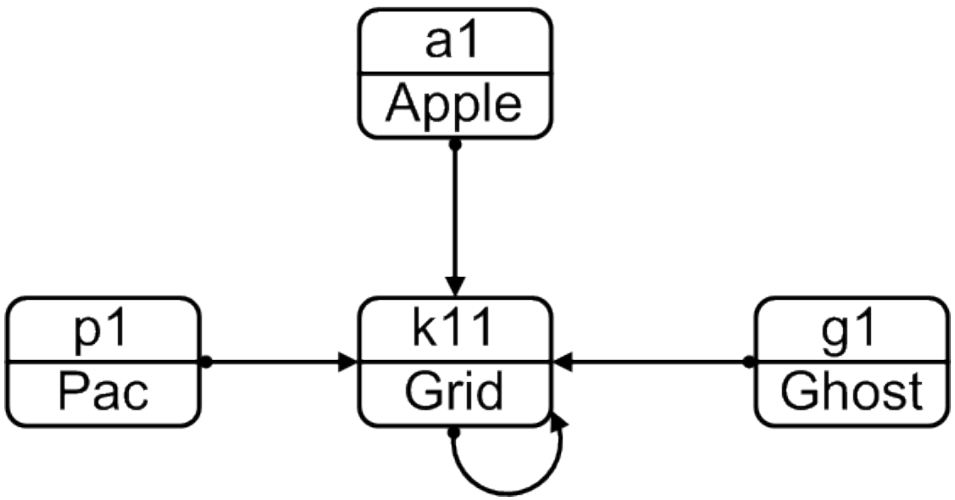
Startgraph:



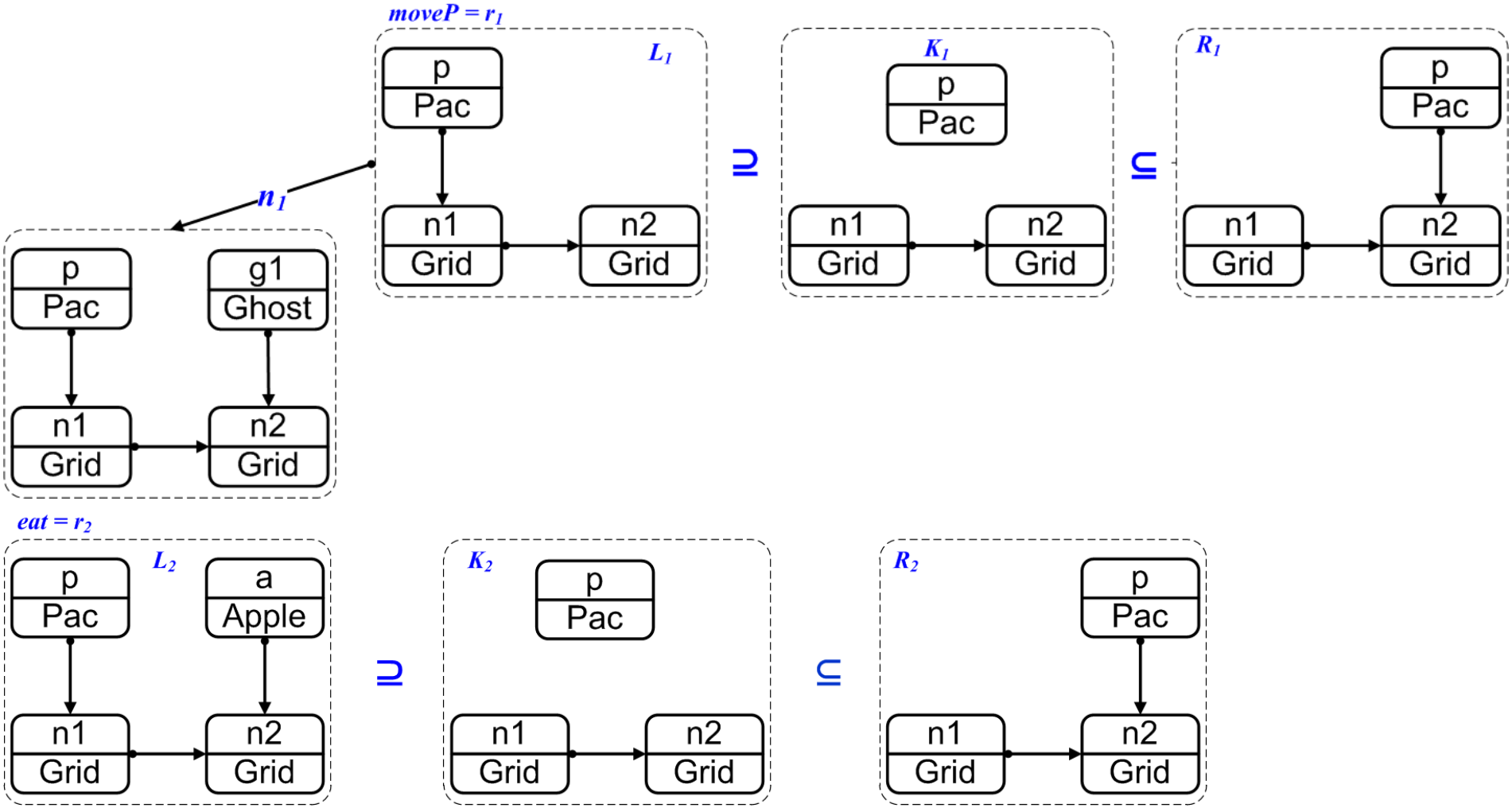
Lösung von Aufgabe 7

Lösung

Typgraph:



Lösung von Aufgabe 7



Lösung von Aufgabe 7