Parallele und sequentielle Unabhängigkeit von Graphtransformationen

Jens Kosiol

1. und 3. Juli 2024

Übersicht

- Parallele und sequentielle Unabhängigkeit von Regelanwendungen ohne NACs
- Eigenschaften:
  - *Unabhängige Regelanwendungen können in beliebiger Reihenfolge stattfinden. (Local Church–Rosser Theorem)*
  - *Unabhängige Regelanwendungen können auch echt parallel stattfinden. (Parallelismustheorem)*
- Parallele und sequentielle Unabhängigkeit von Regelanwendungen mit NACs
- Asymmetrische Konflikte und Abhängigkeiten

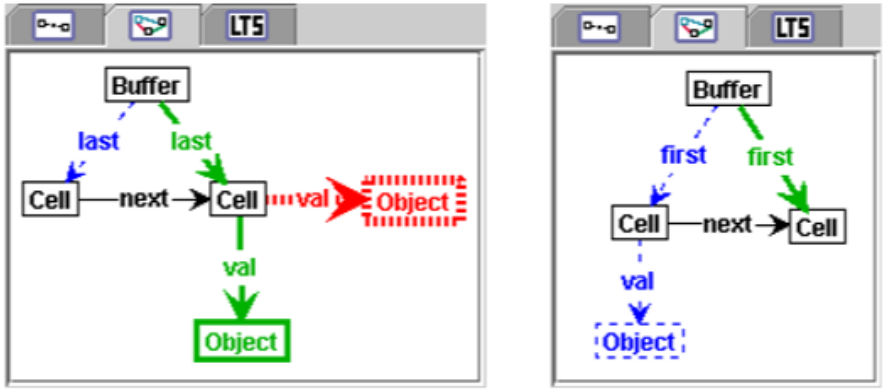
Beispiel: Zirkulärer Puffer

Beispiele für parallele Aktionen:

- Zwei Prozesse wollen gleichzeitig in einen zirkulären Puffer schreiben.
- Ein Prozess möchte reinschreiben, ein anderer auslesen.
- Ein Prozess möchte schreiben und gleichzeitig soll der Puffer um eine Zelle erweitert werden.

Beispiel: Konfliktfreie Transformationen

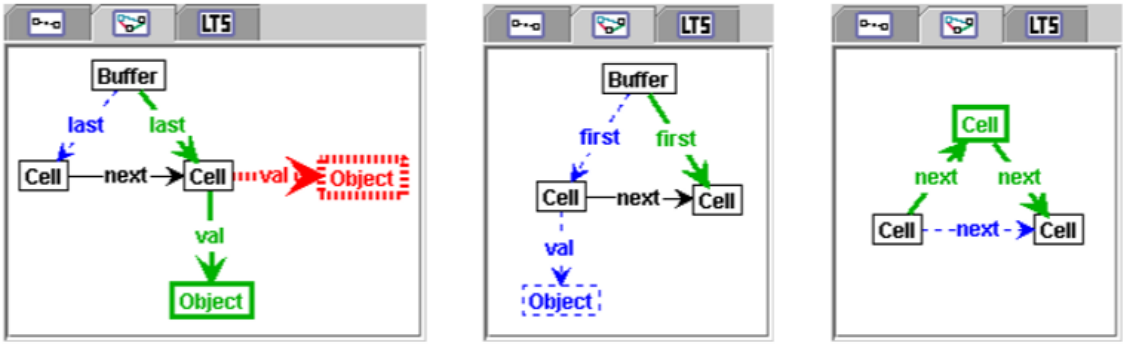
- Anwendungen der Regel *put* sind parallel unabhängig von Anwendungen der Regel *get*.
  - *Wenn beide Regeln anwendbar sind, können sie gleichzeitig oder in einer beliebigen Reihenfolge angewandt werden.*



(a) put rule. (b) get rule.

Beispiel: Konfliktbehaftete Transformationen

- Eine Anwendung der Regel *get* kann in Konflikt mit einer Anwendung der Regel *extend* stehen.
  - *Wenn beide Regeln anwendbar sind, können sie nicht immer gleichzeitig oder in einer beliebigen Reihenfolge angewandt werden.*



(a) put rule. (b) get rule. (c) extend rule.

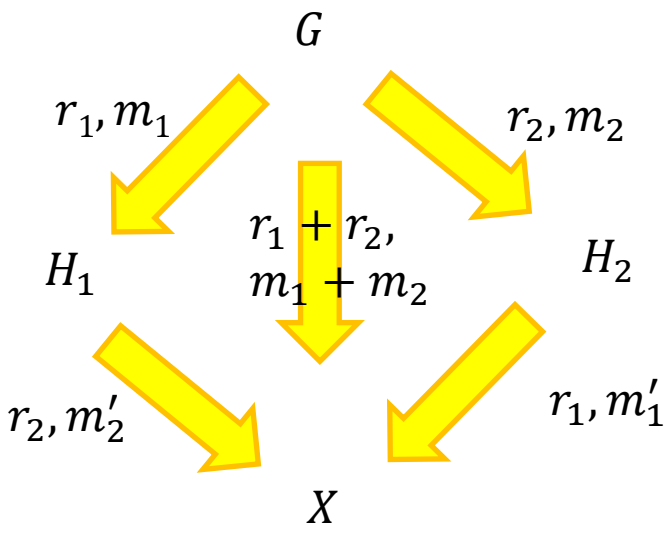
Parallele Transformationen

Parallelität durch Interleaving:

- *Zwei Transformationen werden in verschiedenen Reihenfolgen ausgeführt.*

Echte Parallelität:

- *Zwei Transformationen werden zu einem Schritt zusammengefasst.*

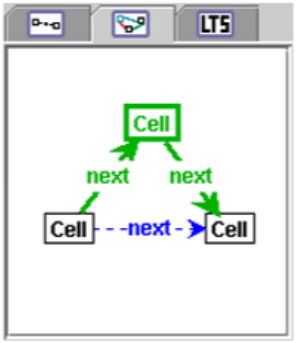


Nachweisverfahren

- Wie können wir nachweisen, dass zwei Regeln immer parallel unabhängig sind?
- Hinreichende Kriterien:
  - *Regel r_1 löscht nur, was Regel r_2 nicht braucht und umgekehrt.*
  - *Regel r_1 erzeugt nur, was Regel r_2 nicht verbietet und umgekehrt.*

Beispiel: Parallel unabhängige Regelanwendungen

- Zwei Anwendungen der Regel *extend* sind unabhängig voneinander.
  - *Voraussetzung: Die Regeln haben verschiedene Ansätze.*
  - *Wenn beide Regeln anwendbar sind, können sie auch gleichzeitig oder in einer beliebigen Reihenfolge angewandt werden.*



(c) extend rule.

Definition: Parallel unabhängige Regelanwendungen

Gegeben zwei Regeln $r_1 = (L_1 \supseteq K_1 \subseteq R_1)$ und $r_2 = (L_2 \supseteq K_2 \subseteq R_2)$, dann sind zwei Transformationsschritte $t_1 : G \Longrightarrow_{r_1,m_1} H_1$ und $t_2 : G \Longrightarrow_{r_2,m_2} H_2$ parallel unabhängig, falls sich ihre Regelansätze nur in zu erhaltenden Graphelementen von r_1 und r_2 überschneiden, also gilt:

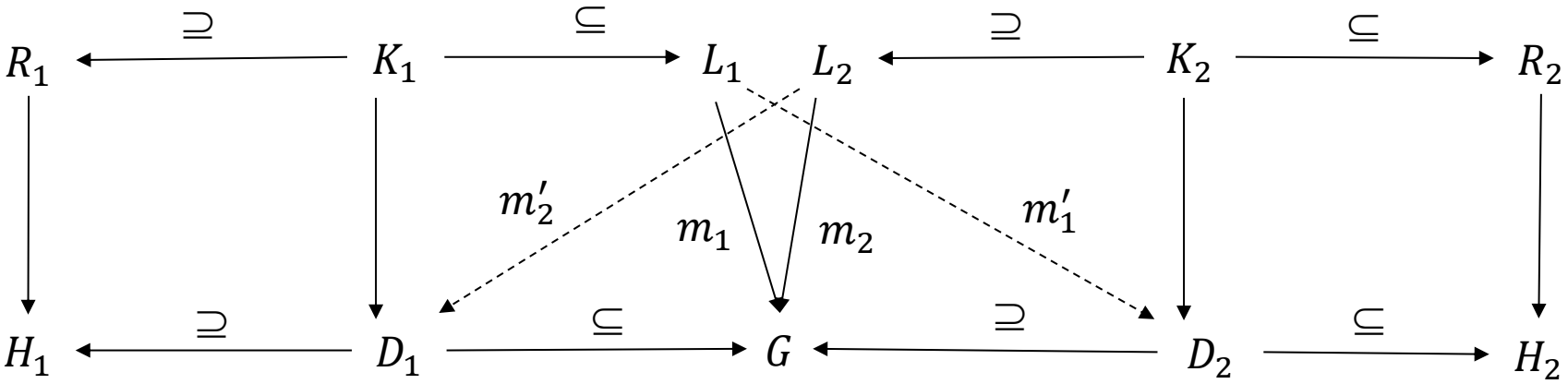
$$m_1(L_1) \cap m_2(L_2) \subseteq m_1(K_1) \cap m_2(K_2).$$

Achtung: Diese Definition deckt keine negativen Anwendungsbedingungen ab!

Charakterisierung: Parallel unabhängige Regelanwendungen

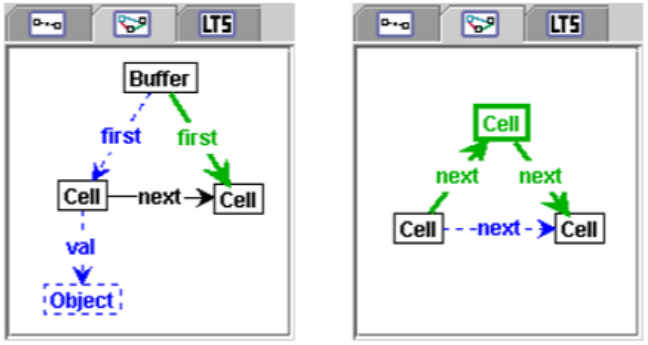
Zwei Transformationsschritte $t_1: G \Longrightarrow_{r_1,m_1} H_1$ und $t_2: G \Longrightarrow_{r_2,m_2} H_2$ sind genau dann parallel unabhängig, wenn es Morphismen

- $m_1': L_1 \to D_2$ mit $m_1'(x) = m_1(x)$ für alle x in L_1 und
- $m_2': L_2 \to D_1$ mit $m_2'(x) = m_2(x)$ für alle x in L_2 gibt.



Beispiel: Parallele Unabhängigkeit

- Wir betrachten alle Paare (r, s) von *get* und *extend*:
  - *(get,get), (get,extend), (extend,get), (extend,extend)*
- Gegeben ein Regelpaar (r, s), gibt es ein Paar von Regelanwendungen $G \Longrightarrow_r H_1$ und $G \Longrightarrow_s H_2$,
  - *das parallel unabhängig ist?*
  - *das parallel abhängig ist?*

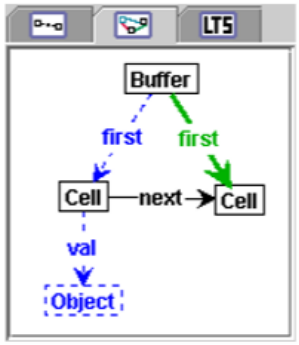


(b) get rule. (c) extend rule.

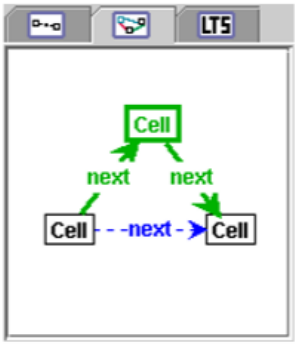
Beispiel: Sequentielle Unabhängigkeit

Eine Anwendung der Regel *get* kann sequentiell abhängig von einer Anwendung der Regel *extend sein*.

- Eine Anwendung von *extend* kann eine next-Kante und Zelle schaffen, die dann in einer Anwendung der Regel *get* verwendet werden.
- Anwendungen von *get* und *extend*, deren Ansätze nicht überlappen, sind sequentiell unabhängig.



(b) get rule.



(c) extend rule.

Definition: Sequentiell unabhängige Regelanwendungen

Gegeben zwei Regeln $r_1 = (L_1 \supseteq K_1 \subseteq R_1)$ und $r_2 = (L_2 \supseteq K_2 \subseteq R_2)$, dann sind Transformationsschritte $t_1 : G \Longrightarrow_{r_1,m_1} H$ mit Ko-Ansatz $n_1 : R_1 \to H$ (mit $n_1(x) = m_1(x)$ für $x \in \mathrm{K}_1$) und $t_2 : H \Longrightarrow_{r_2,m_2} X$ sequentiell unabhängig, falls

- der Regelansatz von t_2 nur zu erhaltene Graphelemente von t_1 nutzt, d.h., dass t_2 nichts braucht, was t_1 erzeugt;
- der Regelansatz von t_2 alles, was t_1 nutzt oder erzeugt, nicht löscht.

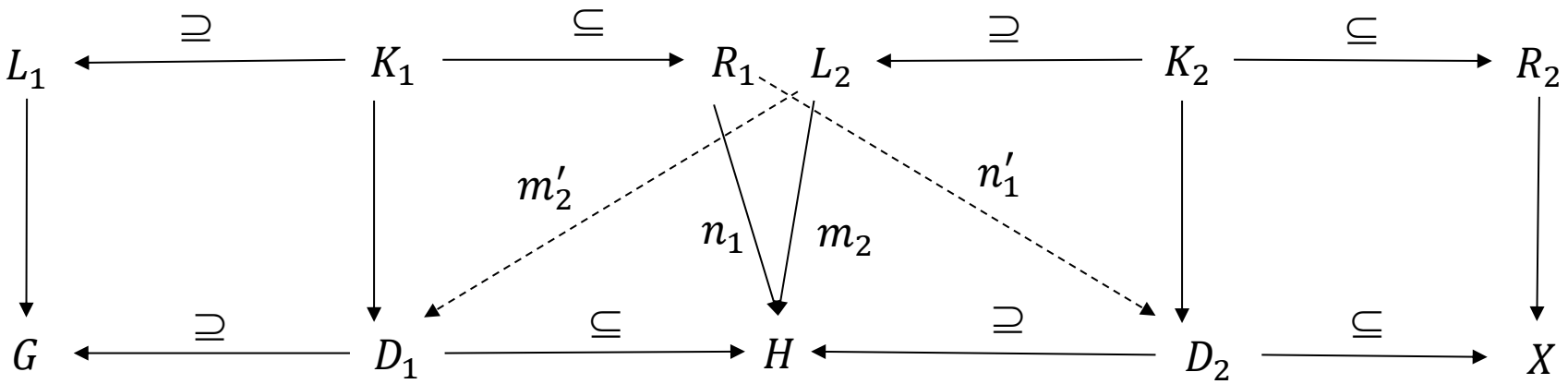
Also, falls gilt:

$$n_1(R_1) \cap m_2(L_2) \subseteq n_1(K_1) \cap m_2(K_2).$$

Charakterisierung: Sequentiell unabhängige Regelanwendungen

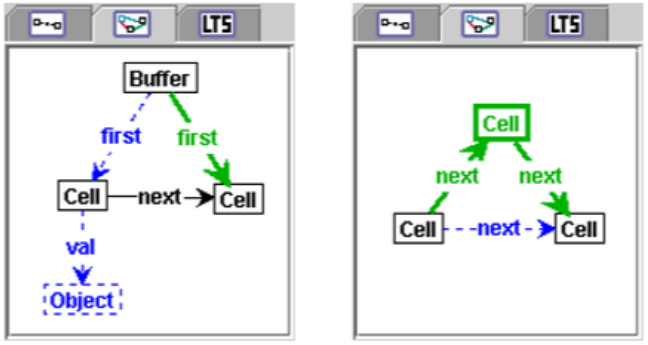
Zwei Transformationsschritte $t_1: G \Longrightarrow_{r_1,m_1} H$ und $t_2: H \Longrightarrow_{r_2,m_2} X$ sind genau dann sequentiell unabhängig, wenn es Morphismen

- $m_2': L_2 \to D_1$ mit $m_2'(x) = m_2(x)$ für alle $x \in L_2$ und
- $n_1': R_1 \to D_2$ mit $n_1'(x) = n_1(x)$ für alle $x \in R_1$ gibt.



Beispiel: Sequentielle Unabhängigkeit

- Wir betrachten alle Paare (r, s) von *get* und *extend*:
  - *(get,get), (get,extend), (extend,get), (extend,extend)*
- Gegeben ein Regelpaar (r, s), gibt es eine Sequenz von Regelanwendungen $G \Rightarrow_r H \Rightarrow_s X$,
  - *die sequentiell unabhängig ist?*
  - *die sequentiell abhängig ist?*



(b) get rule. (c) extend rule.

Übersicht

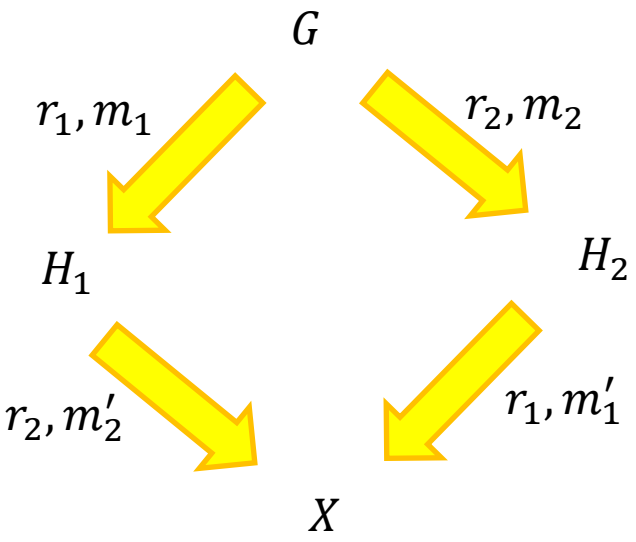
- Parallele und sequentielle Unabhängigkeit von Regelanwendungen ohne NACs
- Eigenschaften:
  - *Unabhängige Regelanwendungen können in beliebiger Reihenfolge stattfinden. (Local Church–Rosser Theorem)*
  - *Unabhängige Regelanwendungen können auch echt parallel stattfinden. (Parallelismustheorem)*
- Parallele und sequentielle Unabhängigkeit von Regelanwendungen mit NACs
- Asymmetrische Konflikte und Abhängigkeiten

Local Church–Rosser Eigenschaft

Gegeben seien zwei Transformationsschritte $t_1 : G \Rightarrow_{r_1,m_1} H_1$ und $t_2 : G \Rightarrow_{r_2,m_2} H_2$. Wenn t_1 und t_2 parallel unabhängig sind, dann gibt es

- einen Graphen X und
- zwei Transformationsschritte $t_1' : H_1 \Rightarrow_{r_2,m_2'} X$ und $t_2' : H_2 \Rightarrow_{r_1,m_1'} X$, sodass $t_1; t_1'$ und $t_2; t_2'$ jeweils sequentiell unabhängig sind.

Gegeben eine sequentiell unabhängige Sequenz $t_1; t_1'$, dann gibt es eine sequentiell unabhängige Sequenz $t_2; t_2'$, sodass t_1 und t_2 parallel unabhängig sind.



Beweis: Th. 5.12 in [EEPT06]

Definition: Parallele Regeln und Anwendungen

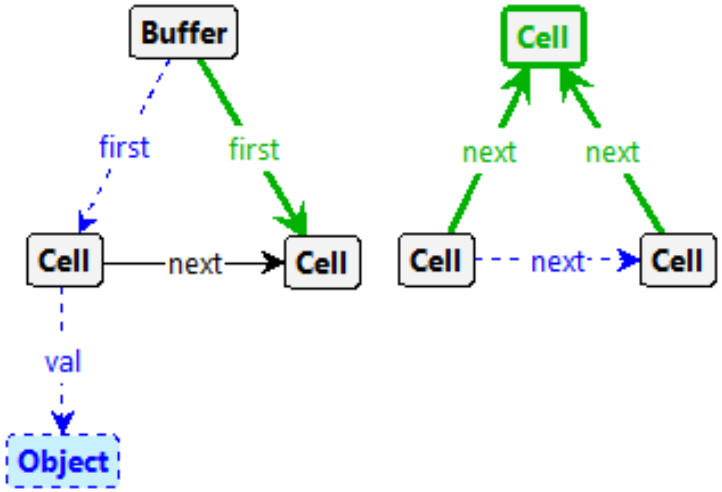
Für zwei Regeln $r_1 = (L_1 \supseteq K_1 \subseteq R_1)$ und $r_2 = (L_2 \supseteq K_2 \subseteq R_2)$ ist die parallele Regel $r_1 + r_2$ durch

$$r_1 + r_2 = (L_1 \uplus L_2 \supseteq K_1 \uplus K_2 \subseteq R_1 \uplus R_2)$$

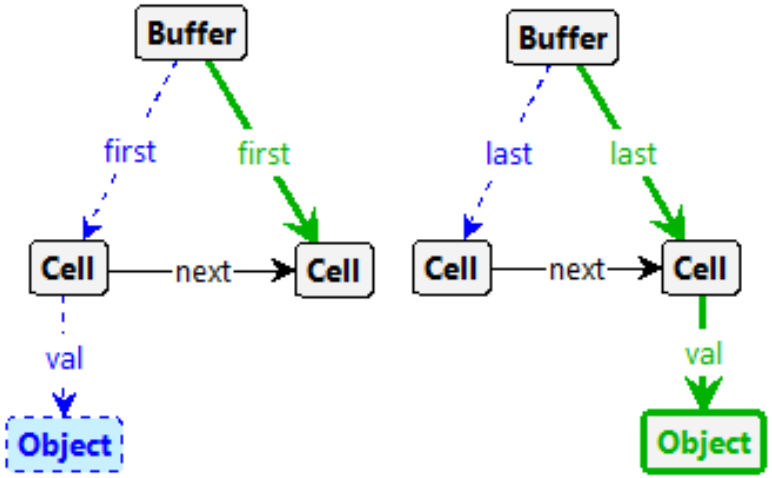
definiert.

Die Anwendung einer parallelen Regel wird parallele Anwendung genannt.

Beispiel parallele Regel



get + *extend*



get + *put* (ohne NAC)

Wiederholung: Anwendung einer Regel an einem injektiven Ansatz

- Gegeben: ein Graph G und eine Regel $r = (L, R)$
  - $K = L \cap R$ *ist ein Graph*
- Die Regel r ist auf G anwendbar, falls
  - *Ansatz (Match): Es gibt einen* **injektiven** *Graphmorphismus* $m: L \to G$
  - *Keine hängenden Kanten:* $D = G \setminus m(L \setminus K)$ *ist ein Graph.*
- Anwendung von Regel r auf Graph G an Ansatz m: $G \Longrightarrow_{r,m} H$
  - $D = G \setminus m(L \setminus K)$
  - $H = D \uplus (R \setminus K)$

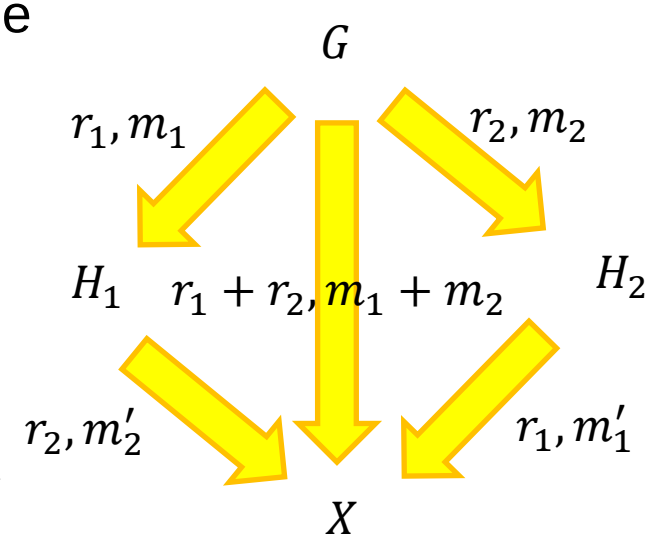
Definition: Anwendung einer Regel an einem allgemeinen Ansatz

- Gegeben: ein Graph G und eine Regel $r = (L, R)$
  - $K = L \cap R$ *ist ein Graph.*
- Die Regel r ist auf G anwendbar, falls
  - *Ansatz (Match): Es gibt einen Graphmorphismus* $m: L \to G$.
  - *Keine hängenden Kanten:* $D = G \setminus m(L \setminus K)$ *ist ein Graph.*
  - *Ein zu löschendes Element und ein weiteres aus* L *werden von* m *nicht auf das gleiche Element in* G *abgebildet.*
- Anwendung von Regel r auf Graph G an Ansatz m: $G \Longrightarrow_{r,m} H$
  - $D = G \setminus m(L \setminus K)$
  - $H = D \uplus (R \setminus K)$

Parallelismustheorem

Nichtinjektive Regelansätze sind erlaubt:

- *Synthese:* Für sequentiell unabhängige Sequenzen $G \Longrightarrow_{r_1} H_1 \Longrightarrow_{r_2} X$ und $G \Longrightarrow_{r_2} H_2 \Longrightarrow_{r_1} X$ gibt es eine parallele Anwendung $G \Longrightarrow_{r_1+r_2} X$.
- *Analyse:* Zu einer parallelen Anwendung $G \Longrightarrow_{r_1+r_2} X$ können zwei sequentiell unabhängige Sequenzen $G \Longrightarrow_{r_1} H_1 \Longrightarrow_{r_2} X$ und $G \Longrightarrow_{r_2} H_2 \Longrightarrow_{r_1} X$ konstruiert werden.
- *Bijektive Korrespondenz:* Die Synthese und die Analyse sind invers zueinander (bis auf Isomorphie).



Beweis: Th. 5.18 in [EEPT06]

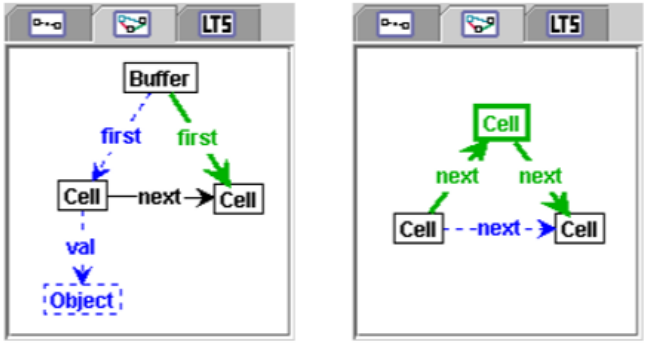
Definition: Parallel und sequentiell unabhängige Regeln

Zwei Regeln r_1 und r_2 sind parallel unabhängig, falls alle Transformationsschritte $G \Longrightarrow_{r_1} H_1$ und $G \Longrightarrow_{r_2} H_2$ parallel unabhängig sind.

Zwei Regeln r_1 und r_2 sind sequentiell unabhängig, falls alle Sequenzen $G \Longrightarrow_{r_1} H_1 \Longrightarrow_{r_2} X$ sequentiell unabhängig sind.

Beispiel: Parallel und sequentiell unabhängige Regeln

- Wir betrachten alle Paarungen (r, s) von *get* und *extend*:
  - *(get,get), (get,extend), (extend,get), (extend,extend)*
- Für jedes Regelpaar (r, s):
  - *Ist (r,s) parallel unabhängig?*
  - *Ist (r,s) sequentiell unabhängig?*



(b) get rule. (c) extend rule.

Parallele und sequentielle Unabhängigkeit: Vergleich

- Parallele Unabhängigkeit beantwortet die Frage ob, gegeben zwei mögliche Transformationen auf einem Ausgangsgraphen, auch beide Transformationen zusammen möglich sind.
  - *Ausgangssituation: Zwei Ansätze m_1, m_2 für Regeln r_1, r_2 auf einem Graphen G*
  - *Intuition: Parallele Unabhängigkeit liegt vor, wenn keine der Regelanwendungen Teile des anderen Ansatzes zerstört.*
- Sequentielle Unabhängigkeit beantwortet die Frage ob, gegeben eine Sequenz von zwei Transformationen, diese Sequenz auch mit getauschter Reihenfolge durchgeführt werden könnte.
  - *Ausgangssituation: Hintereinanderausführung von zwei Transformationen $t_1; t_2$*
  - *Intuition: Sequentielle Unabhängigkeit liegt vor, wenn (i) die erste Transformation keine Teile des zweiten Ansatzes erzeugt und (ii) die zweite Transformation keine Teile des ersten Ansatzes zerstört.*

Konfluenz: Definition und Resultate

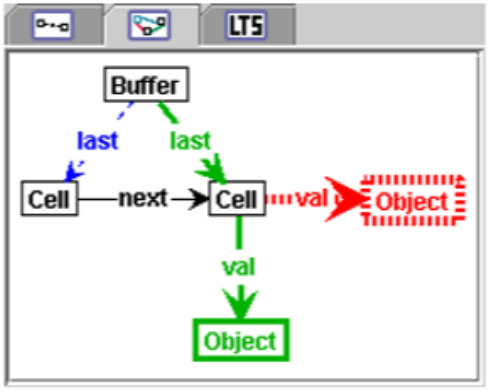
- Definition: Ein Graphtransformationssystem GTS wird konfluent genannt, wenn es für alle Paare von Transformationen $G \Longrightarrow^* H_1$ und $G \Longrightarrow^* H_2$ einen Graphen X und Transformationen $H_1 \Longrightarrow^* X$ und $H_2 \Longrightarrow^* X$ gibt.
- Hinreichendes Kriterium: Wenn alle Regeln eines GTS parallel unabhängig sind, ist das GTS konfluent.
- Ergebnis: Ein konfluentes GTS ist global deterministisch: Jede terminierende Transformationssequenz berechnet (bis auf Isomorphie) das gleiche Resultat [EEPT06, Lemma 3.31]

Übersicht

- Parallele und sequentielle Unabhängigkeit von Regelanwendungen ohne NACs
- Eigenschaften:
  - *Unabhängige Regelanwendungen können in beliebiger Reihenfolge stattfinden. (Local Church–Rosser Theorem)*
  - *Unabhängige Regelanwendungen können auch echt parallel stattfinden. (Parallelismustheorem)*
- Parallele und sequentielle Unabhängigkeit von Regelanwendungen mit NACs
- Asymmetrische Konflikte und Abhängigkeiten

Beispiel: Parallele Abhängigkeit und negative Anwendungsbedingungen

- Zwei Prozesse, die mit der Regel *put* jeweils ein Objekt in den Puffer legen wollen.
- Wenn ein Prozess ein Objekt in den Puffer gelegt hat, hat er den Ansatz der zweiten Regelanwendung zerstört.
- Die Regel *put* kann aber an einem anderen Ansatz noch anwendbar sein.

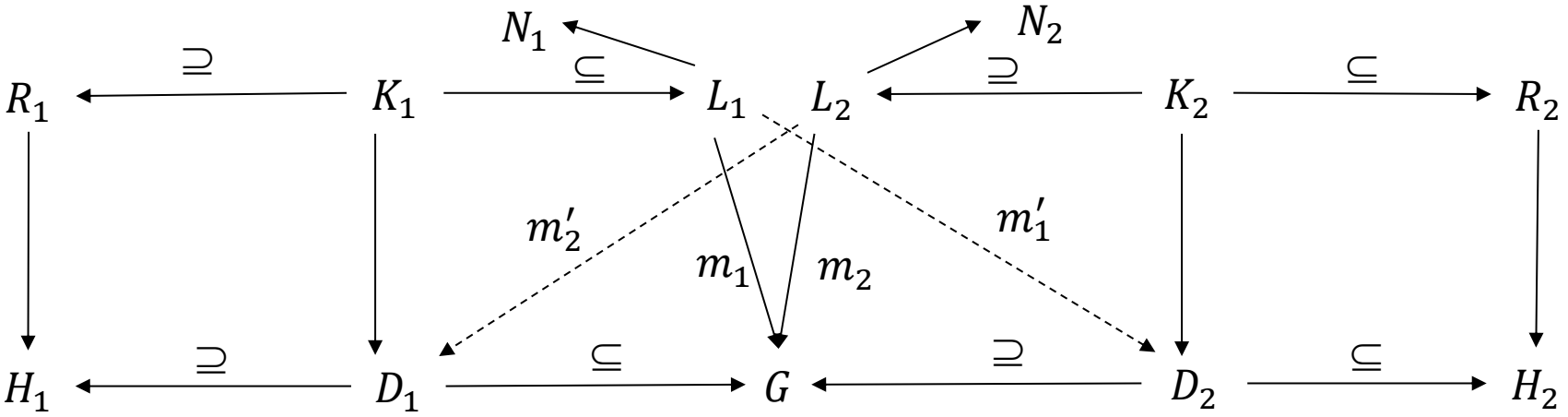


(a) put rule.

Definition: Parallel unabhängige Regelanwendungen mit NACs

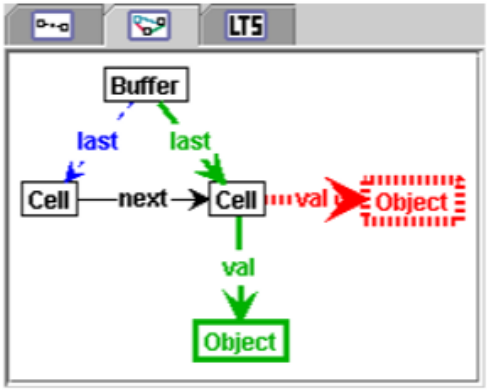
Gegeben zwei Regeln r_1 und r_2 mit NACs, dann sind zwei Transformationsschritte $t_1: G \Longrightarrow_{r_1,m_1} H_1$ und $t_2: G \Longrightarrow_{r_2,m_2} H_2$ parallel unabhängig, wenn es Morphismen $m_1': L_1 \to D_2$ mit $m_1'(x) = m_1(x)$ für alle x in L_1 und $m_2': L_2 \to D_1$ mit $m_2'(y) = m_2(y)$ für alle y in L_2 gibt, sodass

- $\subseteq \circ m_1': L_1 \to H_2$ *erfüllt* N_1 *und* $\subseteq \circ m_2': L_2 \to H_1$ *erfüllt* N_2..

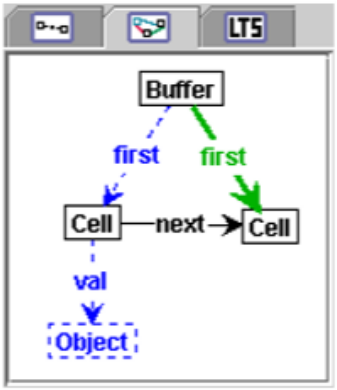


Sequentielle Abhängigkeit und negative Anwendungsbedingungen

Beispiel: Gegeben sei ein voll belegter Puffer. Ein Prozess nimmt ein Objekt mit der Regel *get* aus dem Puffer. Ein zweiter Prozess kann dann mit der Regel *put* ein neues Objekt ablegen. Diese beiden Anwendungen sind sequentiell abhängig, weil durch die Löschung die NAC von *put* erfüllt wird.



(a) put rule.

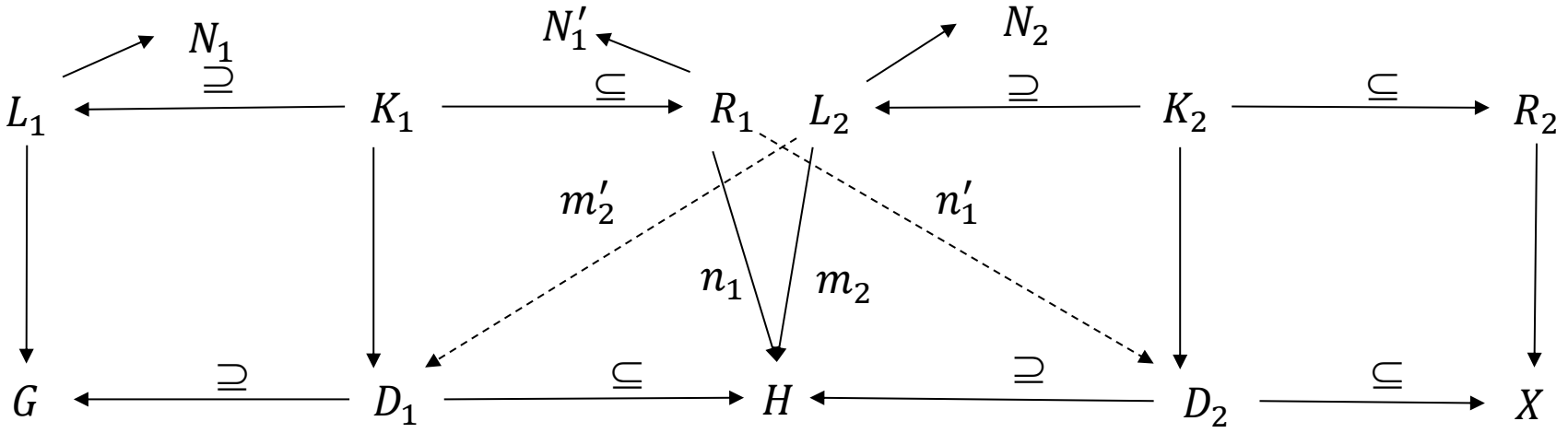


(b) get rule.

Definition: Sequentiell unabhängige Regelanwendungen mit NACs

Gegeben zwei Regeln r_1 und r_2 mit NACs dann sind zwei Transformationsschritte $t_1: G \Longrightarrow_{r_1,m_1} H$ und $t_2: H \Longrightarrow_{r_2,m_2} X$ sequentiell unabhängig, wenn es Morphismen $m_2': L_2 \to D_1$ mit $m_2'(x) = m_2(x)$ für alle x in L_2, $n_1': R_1 \to D_2$ mit $n_1'(y) = n_1(y)$ für alle y in R_1 gibt, sodass:

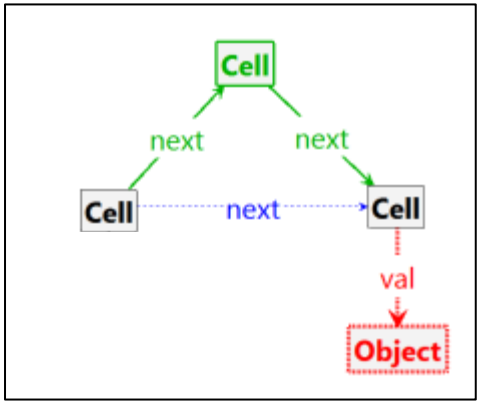
- $\subseteq \circ\, m_2': L_2 \to G$ erfüllt N_2 und $\subseteq \circ\, n_1': R_1 \to X$ erfüllt N_1' (auf R_1 übersetzte NACs).



Beispiel: Sequentiell unabhängige Regelanwendungen mit NACs

- Gibt es Regelanwendungen von (*extend*', *extend*'), die sequentiell unabhängig sind?
- Sind alle Regelanwendungen von (*extend*', *extend*') sequentiell unabhängig?

extend'-Regel:



Unabhängigkeiten von Regelanwendungen mit NACs

- Gegeben: zwei Regeln r_1 und r_2 mit NACs
- Die Local Church–Rosser Eigenschaft gilt.
- Das Parallelismustheorem gilt.
  - *Die NAC der Parallelregel $r_1 + r_2$ ergibt sich durch die disjunkte Vereinigung von* NAC_1 *und* NAC_2*:*
    $\mathrm{NAC}_1 + \mathrm{NAC}_2 = \{ n_1 \uplus id_{L_2} \mid n_1 \in \mathrm{NAC}_1 \} \cup \{ id_{L_1} \uplus n_2 \mid n_2 \in \mathrm{NAC}_2 \}$
- Beweise in [LEPO08]

Konflikte und Abhängigkeiten auf Attributwerten

- Konflikte und Abhängigkeiten auf Attributwerten sind (implizit) abgedeckt: Löschen bzw. Erzeugen von Attributierungskanten.
- Konflikte: Gegeben Transformationen $t_1 : G \Longrightarrow_{r_1,m_1} H_1$ und $t_2 : G \Longrightarrow_{r_2,m_2} H_2$
  - *t_1 löscht einen Attributwert, den t_2 benutzt (oder umgekehrt)*
  - *t_1 setzt einen Attributwert, den t_2 verbietet (oder umgekehrt)*
- Abhängigkeiten: Gegeben Transformationen $t_1; t_2 : G \Longrightarrow_{r_1,m_1} H \Longrightarrow_{r_2,m_2} X$
  - *t_1 setzt einen Attributwert, den t_2 benutzt*
  - *t_1 löscht einen Attributwert, den t_2 verbietet*
  - *t_2 löscht einen Attributwert, den t_1 benutzt*
  - *t_2 setzt einen Attributwert, den t_1 verbietet*

Übersicht

- Parallele und sequentielle Unabhängigkeit von Regelanwendungen ohne NACs
- Eigenschaften:
  - *Unabhängige Regelanwendungen können in beliebiger Reihenfolge stattfinden. (Local Church–Rosser Theorem)*
  - *Unabhängige Regelanwendungen können auch echt parallel stattfinden. (Parallelismustheorem)*
- Parallele und sequentielle Unabhängigkeit von Regelanwendungen mit NACs
- Asymmetrische Konflikte und Abhängigkeiten

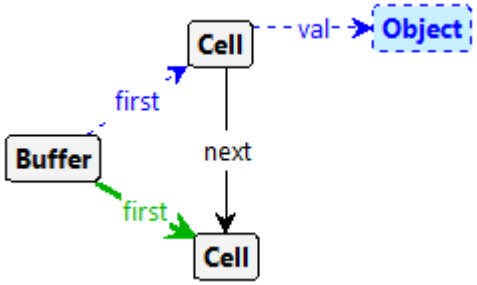
(Asymmetrischer) Konflikt

- Wenn zwei Transformationen $t_1: G \Longrightarrow_{r_1,m_1} H_1$ und $t_2: G \Longrightarrow_{r_2,m_2} H_2$ nicht parallel unabhängig sind, dann verursacht t_1 einen Konflikt mit t_2 oder umgekehrt.
- Die Transformation t_1 verursacht den Konflikt, falls
  - *durch t_1 ein Element gelöscht wird, das t_2 braucht, oder*
  - *t_1 ein Element erzeugt, das eine NAC in t_2 verbietet.*
- Formal:
  - $m_1(L_1) \cap m_2(L_2) \not\subseteq m_1(K_1)$ *(ohne NACs) bzw.*
  - *Es existiert keine Abbildung $m_2': L_2 \to D_1$, sodass N_2 durch $\subseteq \circ\, m_2': L_2 \to H_1$ erfüllt wird.*

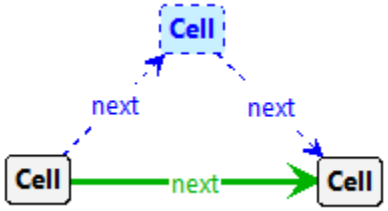
(Asymmetrische) Abhängigkeit

- Wenn eine Sequenz $t_1; t_2: G \Rightarrow_{r_1,m_1} H \Rightarrow_{r_2,m_2} X$ von Transformationen nicht sequentiell unabhängig ist, dann ist t_2 von t_1 abhängig oder umgekehrt.
- Die Transformation t_2 ist von t_1 abhängig, falls
  - *durch t_1 ein Element erzeugt wird, das t_2 braucht, oder*
  - *t_1 ein Element löscht, das eine NAC in t_2 verbietet.*
- Formal:
  - $n_1(R_1) \cap m_2(L_2) \not\subseteq n_1(K_1)$ *(ohne NACs) bzw.*
  - *Es existiert keine Abbildung $m_2': L_2 \to D_1$, sodass N_2 durch $\subseteq \circ\, m_2': L_2 \to G$ erfüllt wird.*
- t_2 verhindert sequentielle Unabhängigkeit, falls
  - *durch t_2 ein Element gelöscht wird, das t_1 braucht, oder*
  - *t_2 ein Element erzeugt, das eine NAC in t_1 verbietet.*
- Formal:
  - $n_1(R_1) \cap m_2(L_2) \not\subseteq m_2(K_2)$ *(ohne NACs) bzw.*
  - *Es existiert keine Abbildung $n_1': R_1 \to D_2$, sodass N_1' durch $\subseteq \circ\, n_1': R_1 \to X$ erfüllt wird.*

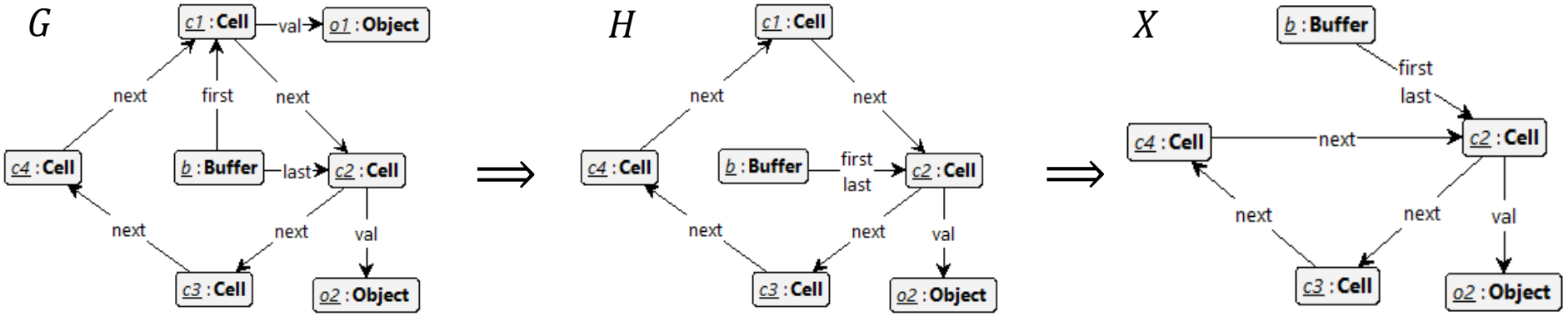
Versteckte Abhängigkeit



Regel *get*



Regel *contract*



Anwendung von get löscht Kanten, die Löschung des Knoten $c1$ überhaupt erst ermöglichen: $n_1(R_1) \cap m_2(L_2) = \{b, c1, c2\}$ (ohne Kanten), aber $c_1 \notin m_2(K_2)$

Zusammenfassung

- Graphtransformationssysteme eignen sich, um sequentielle und parallele Ausführungen zu untersuchen.
- Eine Regelanwendung ist von einer anderen parallel (sequentiell) unabhängig, wenn
  - *die eine Anwendung nur die Graphelemente nutzt, die die andere nicht ändert.*
- Parallel unabhängige Regelanwendungen können
  - *in verschiedenen Reihenfolgen stattfinden. (LCR-Eigenschaft)*
  - *echt parallel ausgeführt werden. (Parallelismustheorem)*

Literatur

- [EEPT06] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, Gabriele Taentzer: Fundamentals of Algebraic Graph Transformation, Springer, 2006
  - *Kapitel 3 und 5*
- [LEPO08] Leen Lambers, Hartmut Ehrig, Ulrike Prange, Fernando Orejas: Parallelism and Concurrency in High-Level Replacement Systems with Negative Application Conditions, Electronic Notes in Theoretical Computer Science 203, 2008
  - *Kapitel 2*
- [Lam08] Leen Lambers: Certifying Rule-Based Models Using Graph Transformation, Doktorarbeit, TU Berlin, Fak. IV, 2009