Konfluenz und Termination von Graphtransformationssystemen

Jens Kosiol

3. Juli 2024

Übersicht

- Funktionales Verhalten:
  - *Ein konfluentes und terminierendes Graphtransformations-system zeigt ein funktionales Verhalten.*
- Welche hinreichenden Kriterien gibt es, die Konfluenz eines Graphtransformationssystems zu entscheiden?
  - *Alle Regelpaare sind parallel unabhängig oder:*
  - *Ein kritisches Paar ist ein Paar von parallel abhängigen minimalen Transformationen. Wenn alle kritischen Paare eines Graphtransformationssystems GTS lokal konfluent sind und GTS ist terminierend, dann ist GTS konfluent.*
- Welche hinreichenden Kriterien gibt es für die Termination eines Graphtransformationssystems?

Konfluenz von Graphtransformationssystemen

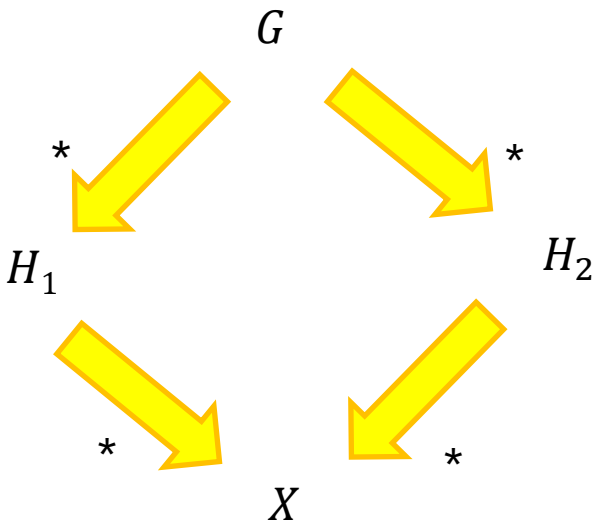
Funktionales Verhalten

- Funktionales Verhalten:
  - *Ein konfluentes Graphtransformationssystem GTS verhält sich, global gesehen, deterministisch.*
  - *Ein konfluentes und terminierendes GTS zeigt ein funktionales Verhalten.*
- Konfluenz:
  - *Zwei Transformationen $G \Longrightarrow^* H_1$ und $G \Longrightarrow^* H_2$ können so fortgesetzt werden, dass beide einen gemeinsamen Graphen X erreichen.*
- Termination:
  - *Ein Graphtransformationssystem ist terminierend, wenn jede mögliche Sequenz von Regelanwendungen endlich ist.*

Konfluenz: Definition und Resultat

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.

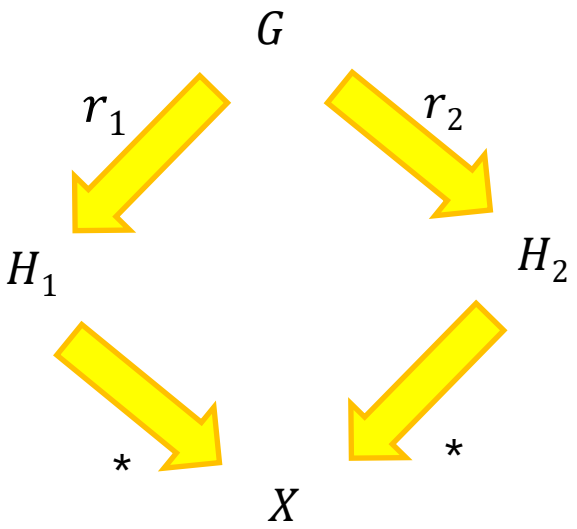
Das GTS ist lokal konfluent, wenn es für alle Transformationsschritte $G \Longrightarrow H_1$ und $G \Longrightarrow H_2$ konfluent ist.



Theorem: Jedes Transformationssystem, das lokal konfluent und terminierend ist, ist konfluent. (Beweis: [DM79])

Konfluenz: Hinreichende Kriterien

Theorem: Wenn alle Regeln eines GTS parallel unabhängig sind, dann ist das GTS konfluent. (Beweis: [EEPT06])

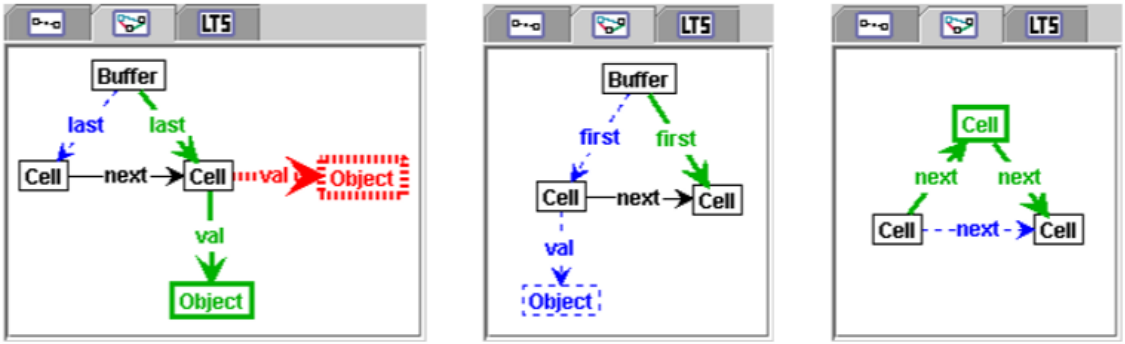


Wenn es auch parallel abhängige Regelpaare im GTS gibt:

- Wenn alle Paare von abhängigen, minimalen Regelanwendungen (strikt) lokal konfluent sind, dann sind alle Regelanwendungen des GTS lokal konfluent.

Beispiel: Zirkulärer Puffer und Konfluenz

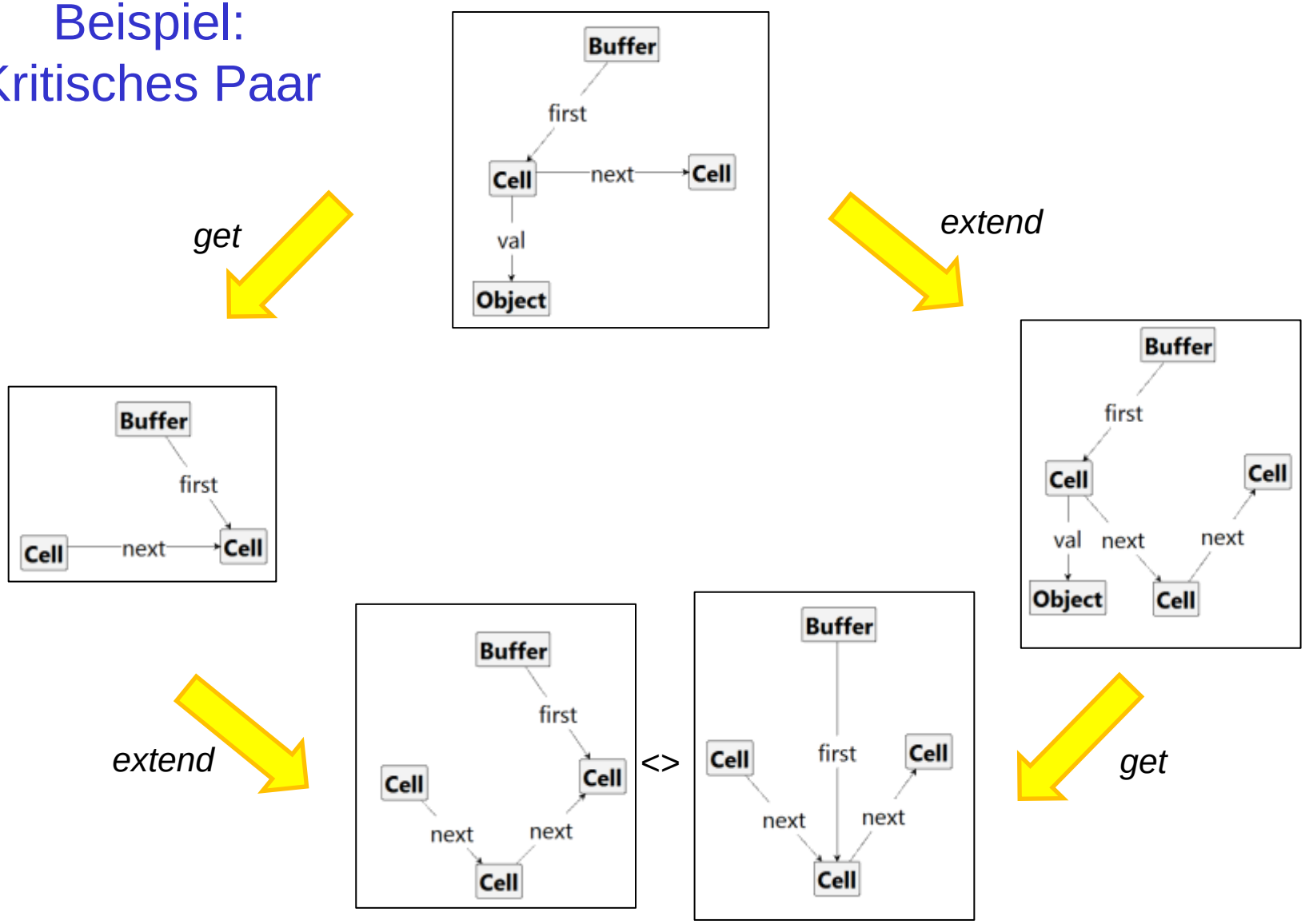
- Gibt es Paare von minimalen, abhängigen Regel-anwendungen?
  - *Minimal: Überlappungen der linken Regelseiten*
  - *(put,extend) und (get,extend) mit überlappender next-Kante.*



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

Für Regeln mit NACs müssen eventuell auch Teile der NACs überlappt werden.

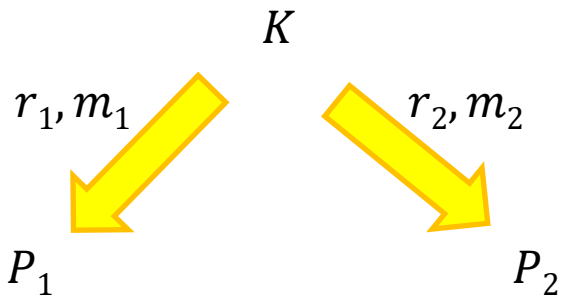
Beispiel: Kritisches Paar



Es gibt keine Regelanwendungen, die das kritische Paar wieder zusammenführen.

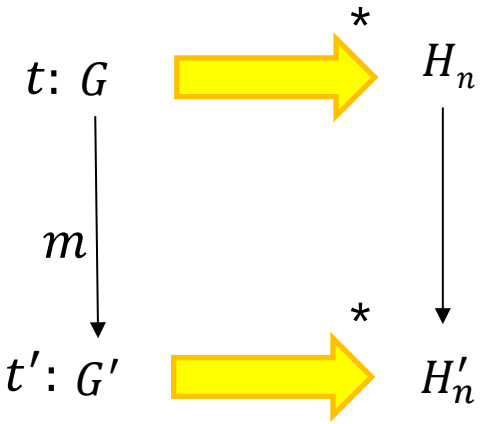
Definition: Kritisches Paar

Gegeben zwei Regeln $r_1 = (L_1, R_1)$ und $r_2 = (L_2, R_2)$, dann ist ein kritisches Paar ein Paar von parallel abhängigen Regelanwendungen $K \Longrightarrow_{r_1,m_1} P_1$ und $K \Longrightarrow_{r_2,m_2} P_2$, sodass $m_1(L_1) \cup m_2(L_2) = K$ gilt.



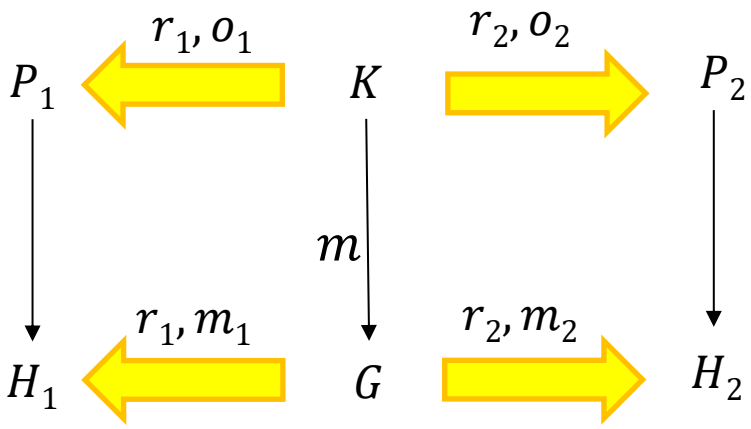
Einbettung von Regelanwendungen

Theorem: Eine Sequenz von Regelanwendungen $t: G \Longrightarrow^* H_n$ kann an einem injektiven Graphmorphismus $m: G \to G'$ in einen größeren Kontext $t': G' \Longrightarrow^* H'_n$ eingebettet werden, wenn m bzgl. aller Regelanwendungen in t keine hängenden Kanten erzeugt. Es gibt dann eine eindeutige Einbettung $H_n \to H'_n$. [EEPT06, Theorem 6.14]



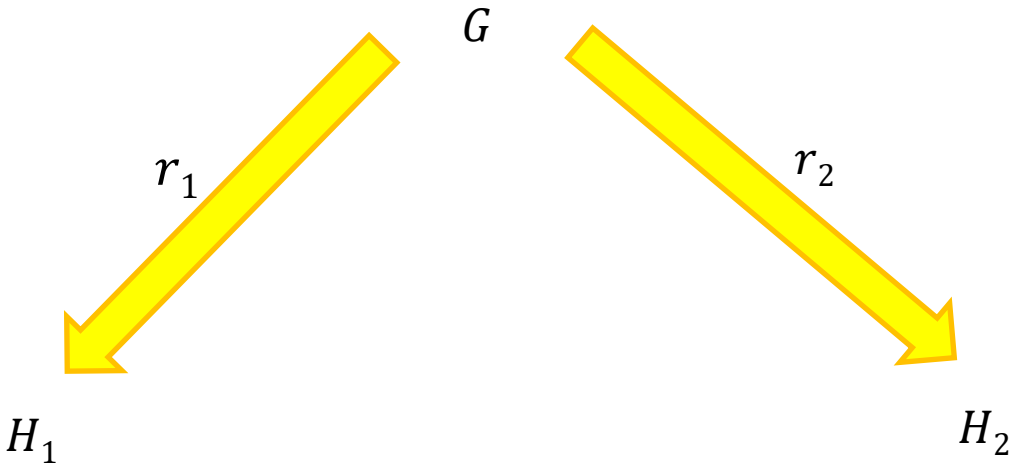
Vollständigkeit der Menge der kritischen Paare

Theorem: Gegeben ein GTS: Für jedes Paar von parallel abhängigen Regelanwendungen $t_1: G \Rightarrow_{r_1,m_1} H_1$ und $t_2: G \Rightarrow_{r_2,m_2} H_2$ gibt es ein kritisches Paar $K \Rightarrow_{r_1,o_1} P_1$ und $K \Rightarrow_{r_2,o_2} P_2$ und einen injektiven Graphmorphismus $m: K \to G$, sodass das kritische Paar in (t_1, t_2) eingebettet werden kann. [EEPT06, Lemma 3.33]



Local Confluence Theorem

Ein Graphtransformationssystem ist lokal konfluent, wenn alle seine kritischen Paare strikt konfluent sind. [EEPT06, Thm. 3.34]

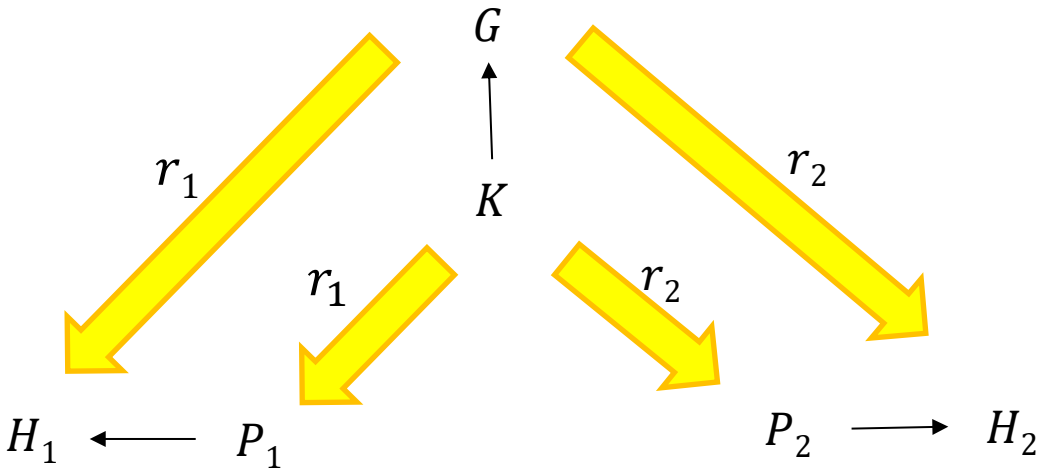


Ein kritisches Paar ist strikt konfluent, wenn es nicht nur zu einem gem. Graphen K' zusammengeführt werden kann, sondern die durch die Transformationen über P_1 und P_2 definierten partiellen Abbildungen von K nach K' gleich sind.

1. Geg: $H_1 \Leftarrow G \Rightarrow H_2$

Local Confluence Theorem

Ein Graphtransformationssystem ist lokal konfluent, wenn alle seine kritischen Paare strikt konfluent sind. [EEPT06, Thm. 3.34]

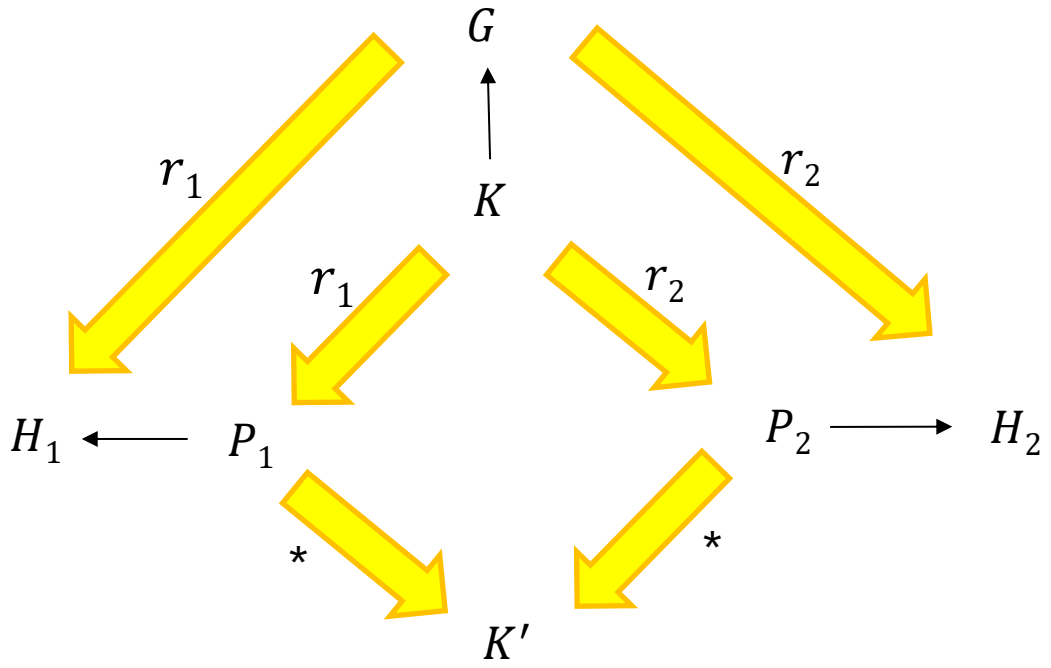


Ein kritisches Paar ist strikt konfluent, wenn es nicht nur zu einem gem. Graphen K' zusammengeführt werden kann, sondern die durch die Transformationen über P_1 und P_2 definierten partiellen Abbildungen von K nach K' gleich sind.

2. Es gibt kritisches Paar wegen Vollständigkeit

Local Confluence Theorem

Ein Graphtransformationssystem ist lokal konfluent, wenn alle seine kritischen Paare strikt konfluent sind. [EEPT06, Thm. 3.34]

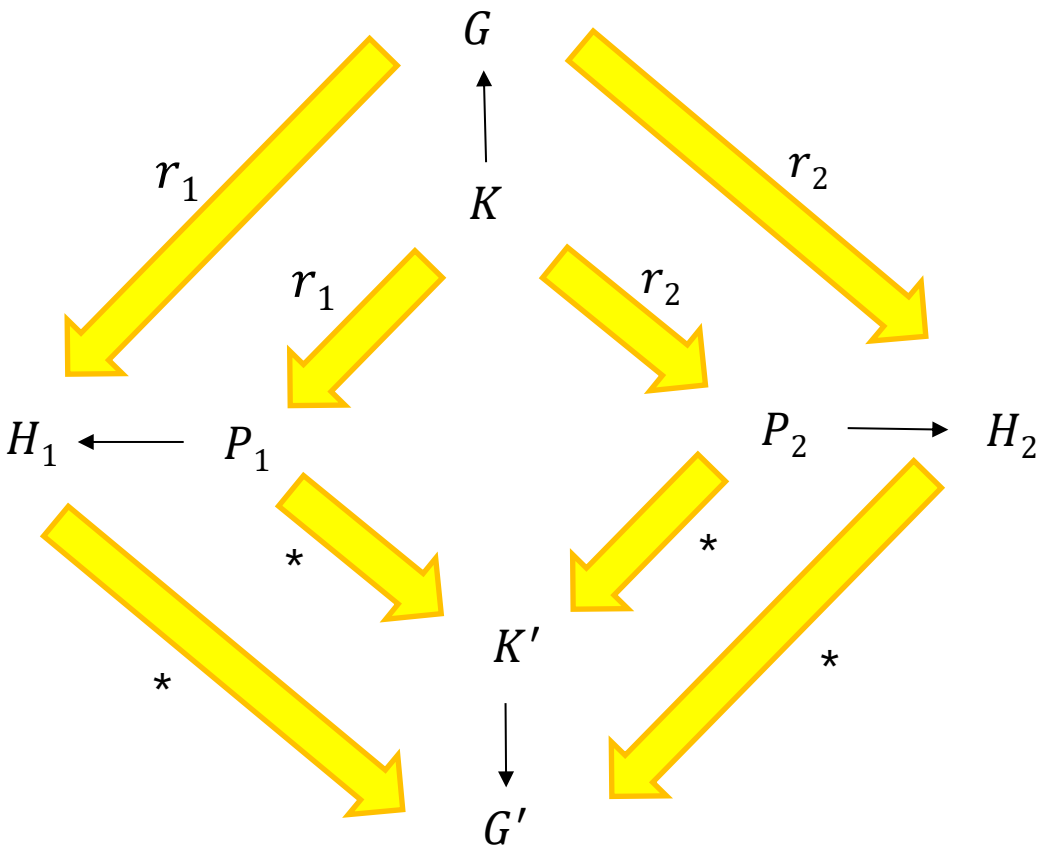


Ein kritisches Paar ist strikt konfluent, wenn es nicht nur zu einem gem. Graphen K' zusammengeführt werden kann, sondern die durch die Transformationen über P_1 und P_2 definierten partiellen Abbildungen von K nach K' gleich sind.

3. Es gibt Transformationen zu K', da das krit. Paar konfluent ist.

Local Confluence Theorem

Ein Graphtransformationssystem ist lokal konfluent, wenn alle seine kritischen Paare strikt konfluent sind. [EEPT06, Thm. 3.34]



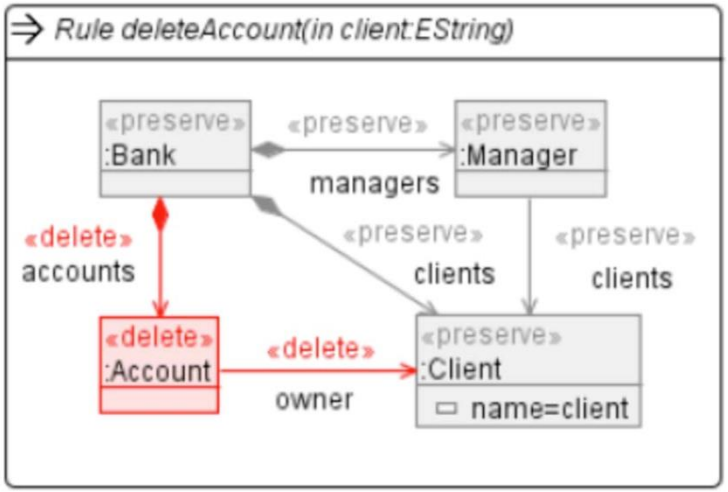
Ein kritisches Paar ist strikt konfluent, wenn es nicht nur zu einem gem. Graphen K' zusammengeführt werden kann, sondern die durch die Transformationen über P_1 und P_2 definierten partiellen Abbildungen von K nach K' gleich sind.

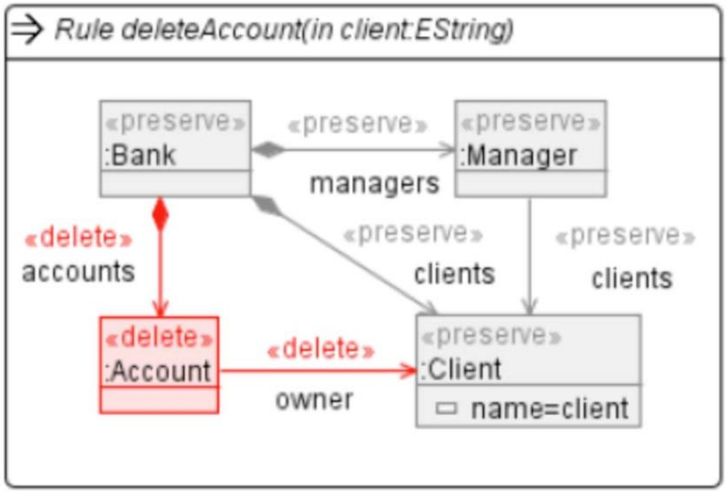
4. Wegen strikter Konfluenz gib es eindeutige Einbettung $K' \rightarrow G'$ auf beiden Seiten.

Beispiel: Kritische Paare

Wie viele kritische Paare hat (*deleteAccount*,*deleteAccount*)?

1. *Keins*
2. *Eins*
3. *Mehrere*

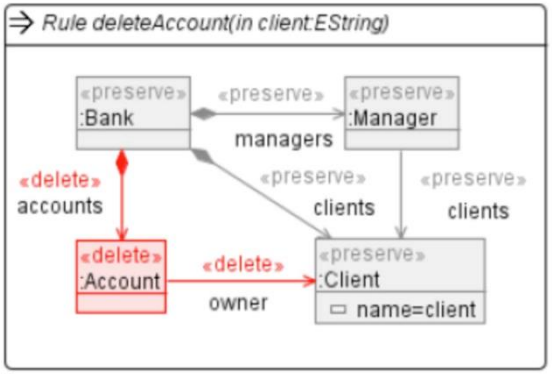


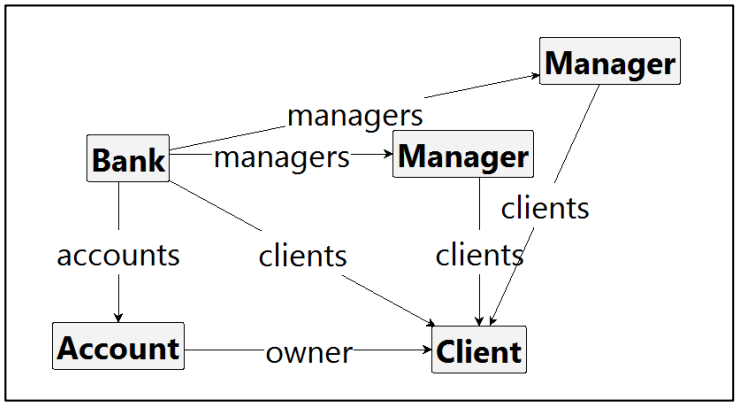


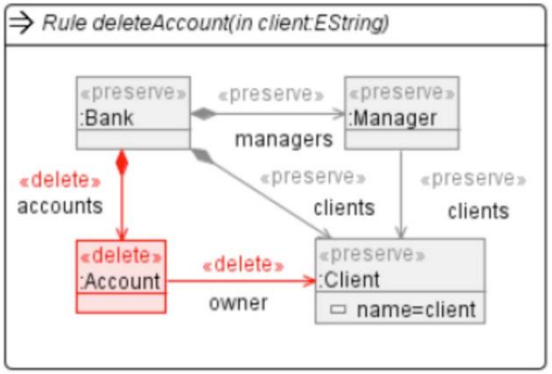
Beispiel: Konfluenz

Ist das gezeigte kritische Paar von (*deleteAccount*,*deleteAccount*) konfluent?

1. *Ja*
2. *Nein*



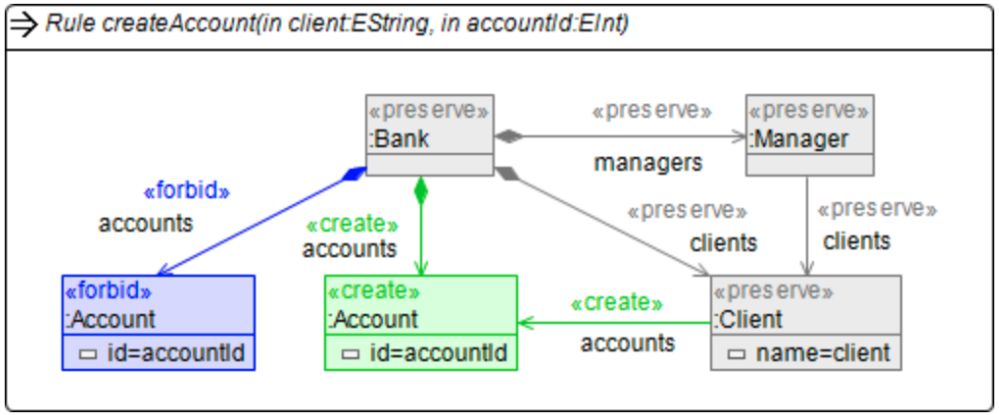


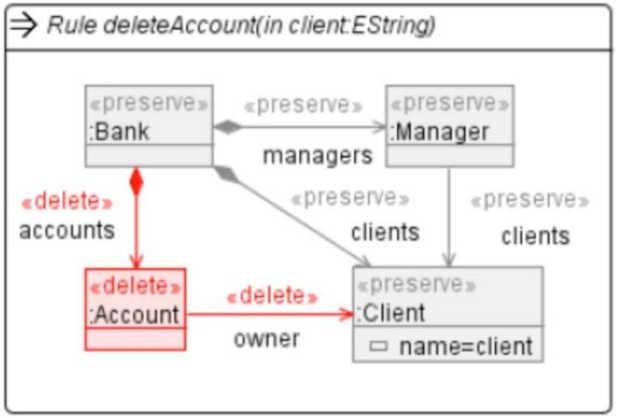


Beispiel: Kritische Paare

Wie viele kritische Paare hat (*createAccount*,*deleteAccount*)?

1. *Keins*
2. *Eins*
3. *Mehrere?*





Überblick Konflikte

- Zwei Regeln $r_1 = (L_1, R_1, NAC_1)$ und $r_2 = (L_2, R_2, NAC_2)$ *stehen* potentiell in Konflikt zueinander, wenn es Ansätze auf einem gemeinsamen Graphen G gibt, sodass
  - *eine Regel ein Element löscht, das die andere benutzt (delete-use bzw. delete-delete Konflikt)*
  - *eine Regel ein Element erzeugt, das die andere verbietet (create-forbid Konflikt)*
- Kritische Paare verschaffen einen Überblick über Konflikte in einem minimalen Kontext.

Da eine Transformationssequenz $t_1; t_2 : G \Rightarrow_{r_1,m_1} H \Rightarrow_{r_2,m_2} X$ genau dann sequentiell unabhängig ist, wenn $G \Leftarrow_{r_1^{-1}, n_1} H \Rightarrow_{r_2,m_2} X$ parallel unabhängig ist, kann die Analyse von Abhängigkeiten auf die von Konflikten zurückgeführt werden.

Termination von Graphtransformationssystemen

Termination

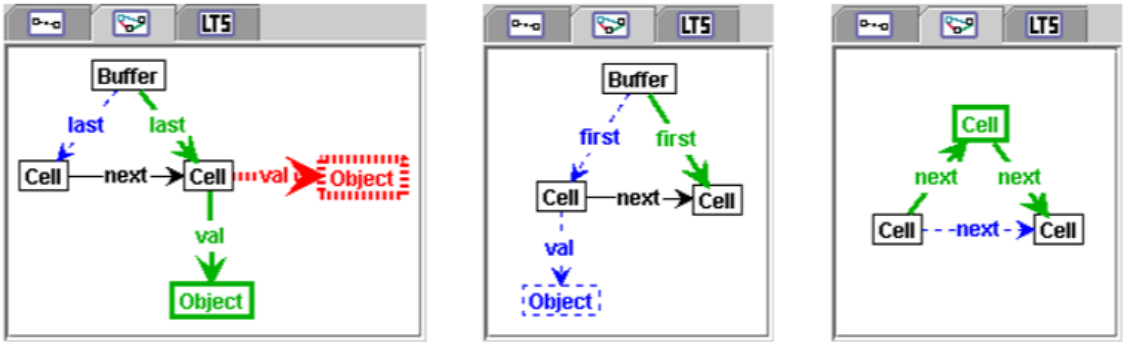
- Ein GTS ist terminierend, wenn jede erlaubte Sequenz von Regelanwendungen endlich ist.
- Im allgemeinen ist die Termination von Graphtrans-formationssystemen unentscheidbar. [Plu98]
- Es gibt mehrere hinreichende Kriterien für Termination:
  - *Aufeinander folgende Regelanwendungen sind absteigend geordnet, z.B. ist die Anzahl der Graphelemente abnehmend.*
  - *Regelanwendungen können in Ebenen eingeteilt werden. Innerhalb einer Ebene werden nur Elemente der nächsten Ebenen erzeugt und*
    - nur Elemente der bestehenden Ebenen gelöscht oder
    - die Anzahl der Regelansätze reduziert.
  - *und weitere [EEL+05]*

Termination: Ordnungsrelationen

- Gegeben: ein GTS
- Gesucht: eine (wohlfundierte) Ordnungsrelation $>$
  - *GTS terminiert, wenn für alle $G \Longrightarrow_r H$ gilt: $G > H$*
  - *Wenn $L > R$ für alle Regeln $r = (L, R)$ aus GTS gilt und Regelanwendung mit $>$ verträglich ist, dann terminiert GTS.*
- Beispiele für Ordnungsrelationen:
  - *Abnehmende Anzahl der Graphelemente in allen Regeln*
  - *Abnehmende Anzahl der Graphelemente eines bestimmten Typs in allen Regeln*
  - *(Abnehmender Attributwert eines bestimmten Knotens in allen Regeln)*
  - *Abnahme von möglichen Regelansätzen*

Beispiel: Termination

- Welche Graphtransformationssysteme terminieren?
  1. $GTS_1 = (T, S, \{get\})$
  2. $GTS_2 = (T, S, \{extend\})$
  3. $GTS_3 = (T, S, \{put\})$
  4. $GTS_4 = (T, S, \{put,get\})$

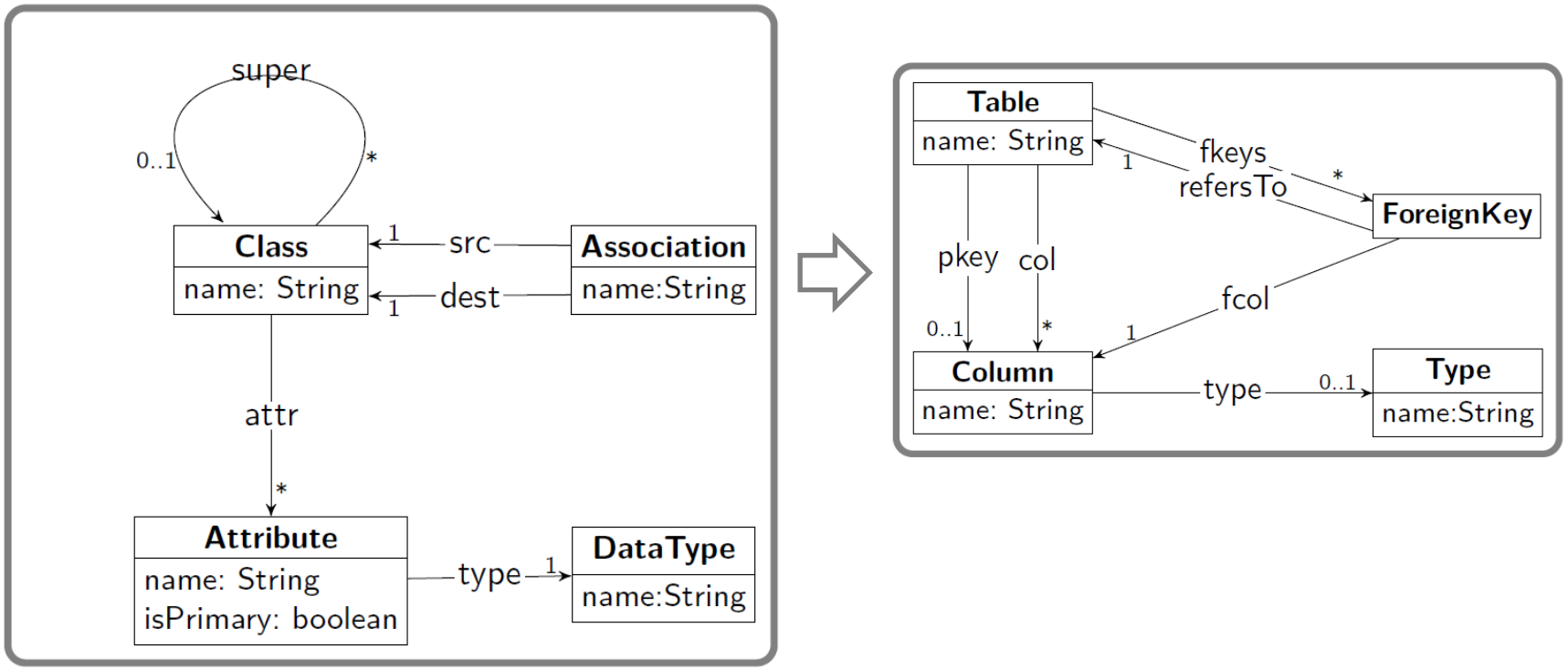


(a) put rule. (b) get rule. (c) extend rule.

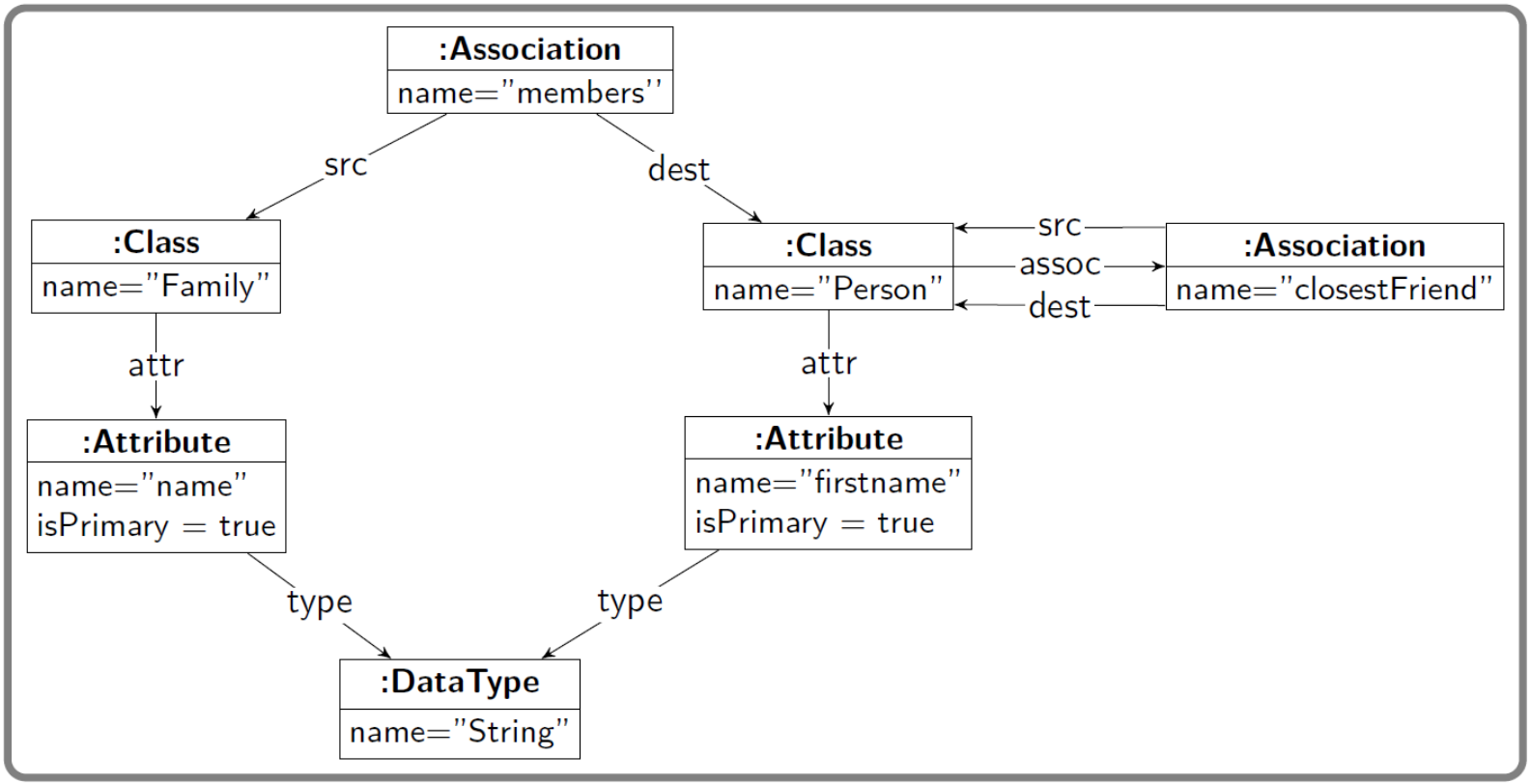
Szenario: Übersetzung von Modellen zwischen Sprachen

[HT20]

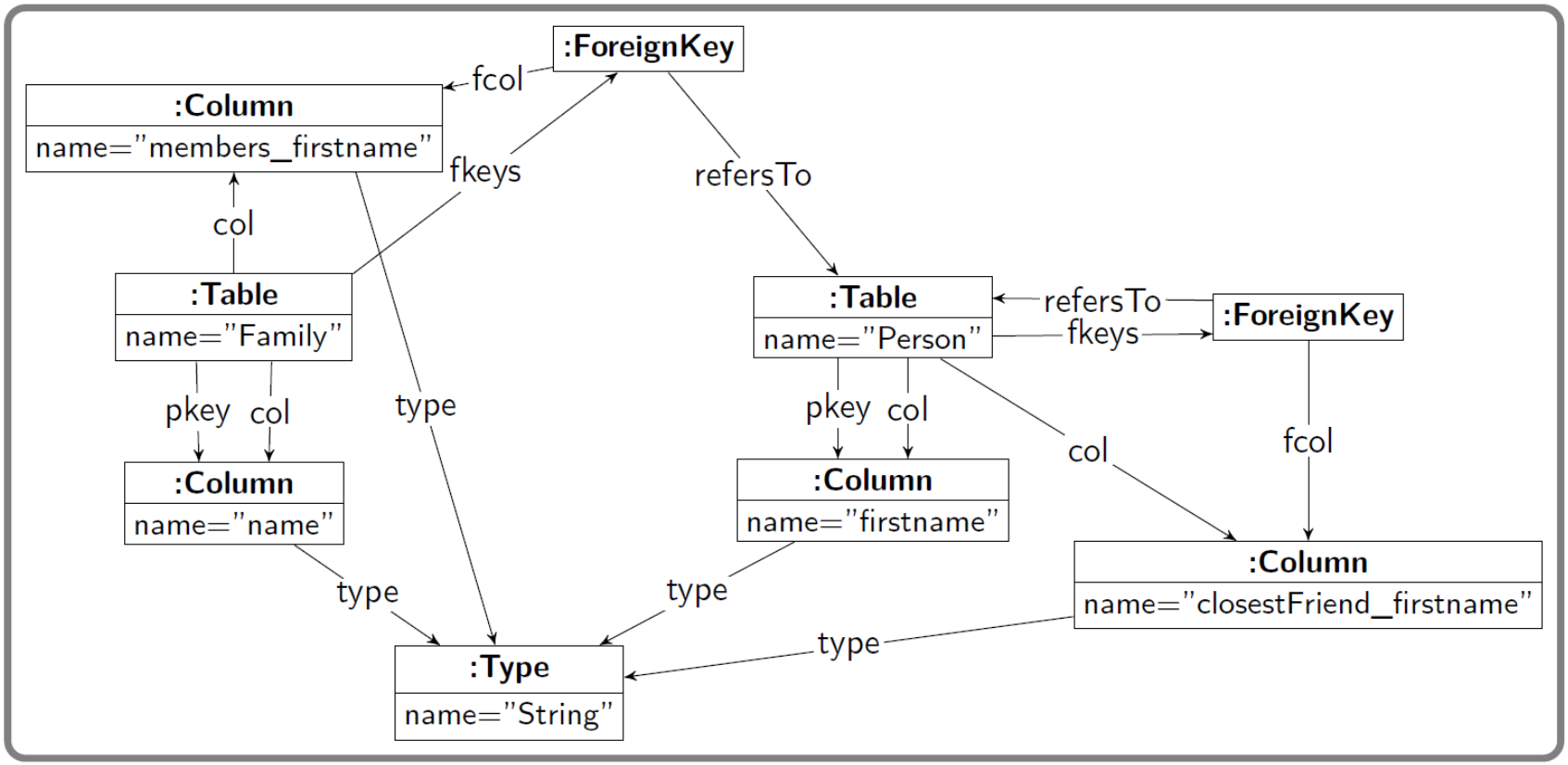
- Beispiel: Übersetzung von objektorientierten Modellen in Datenbankmodelle



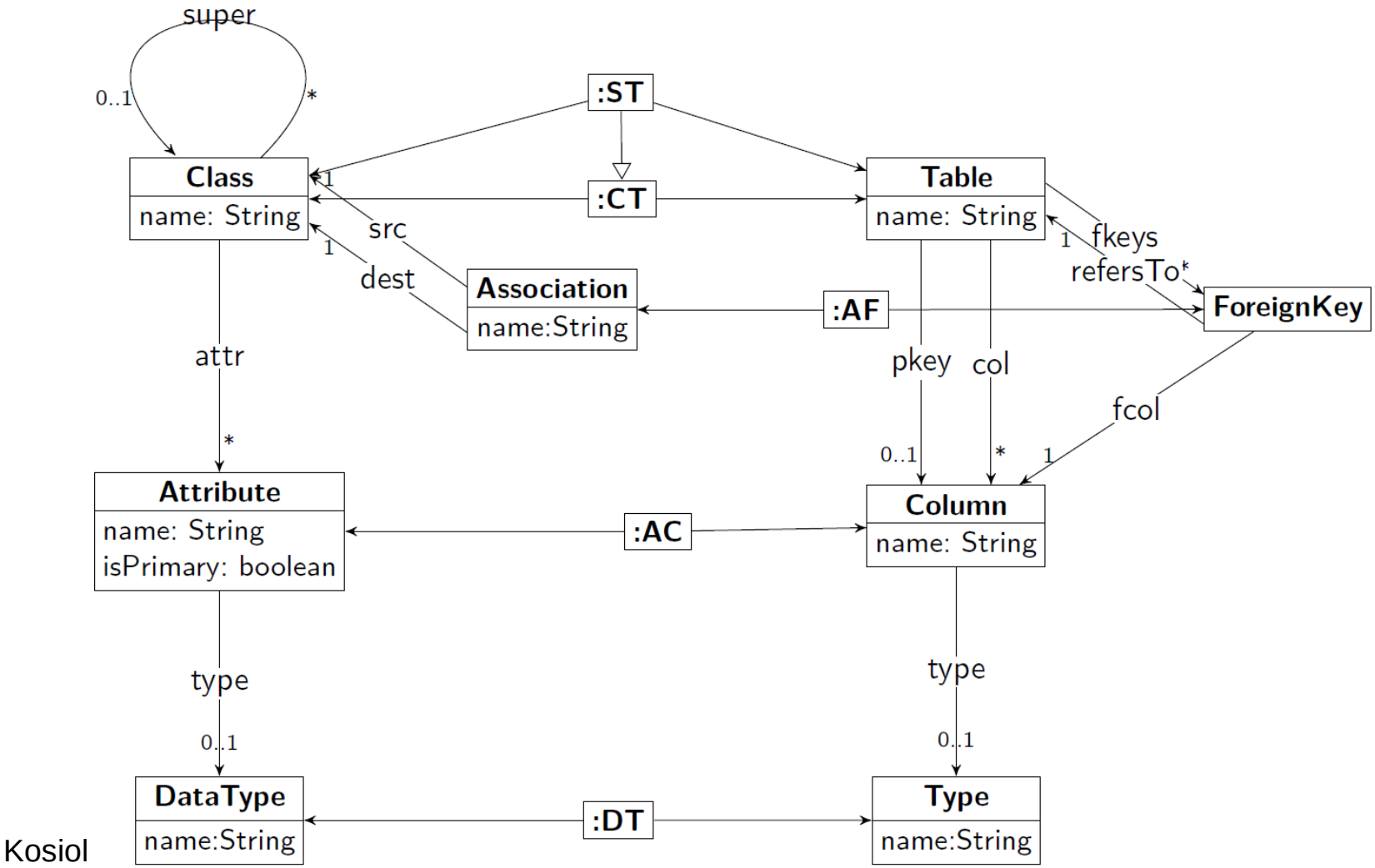
Beispiel: Ein Klassenmodell



Beispiel: Resultierendes Datenbankmodell



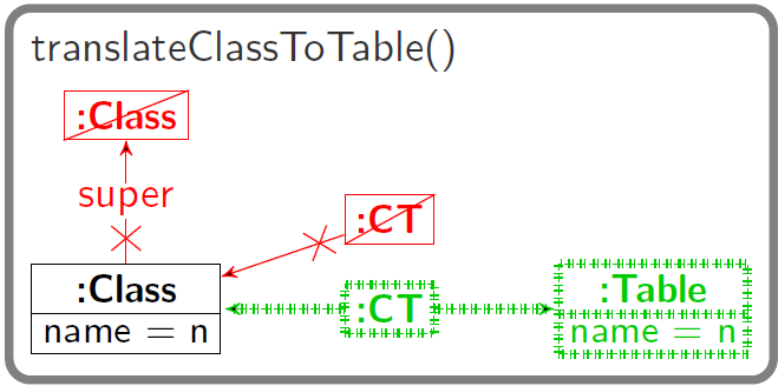
Beispiel: Korrespondierende Typen

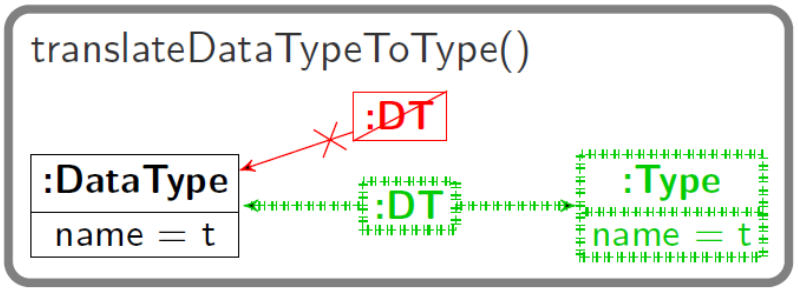


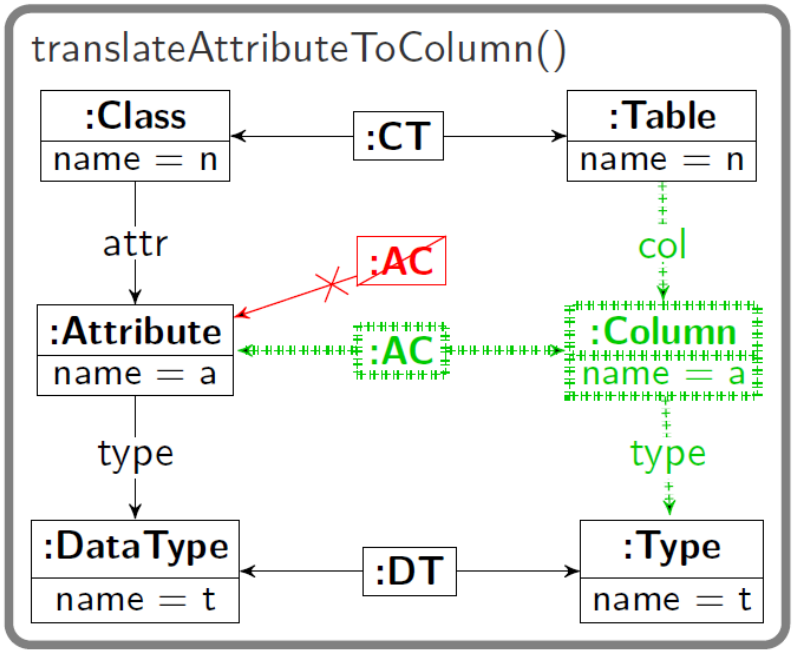
Beispiel: Übersetzung

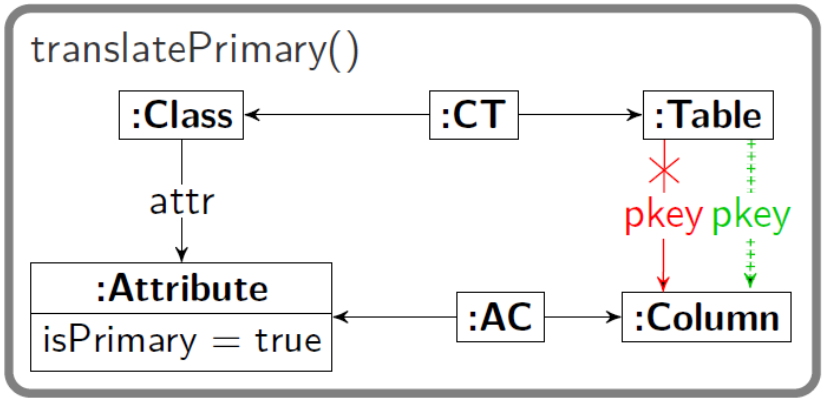
- Für jede Klasse K, die nicht Unterklasse ist, eine Tabelle K. Jede Unterklasse korrespondiert zur Tabelle ihrer Oberklasse.
- Für jeden Datentyp T ein Typ T.
- Für jedes Attribut A einer Klasse eine Spalte A in der korrespondierenden Tabelle. Datentyp des Attributs und Typ der Spalte entsprechen sich. Handelt es sich um ein Schlüsselattribut, wird die korrespondierende Spalte auch ein Schlüssel.
- Für jede Assoziation a von Klasse K nach Klasse L: eine Spalte a_s in der zu Klasse K korrespondierenden Tabelle. Diese Spalte enthält den Schlüssel s der Tabelle L.

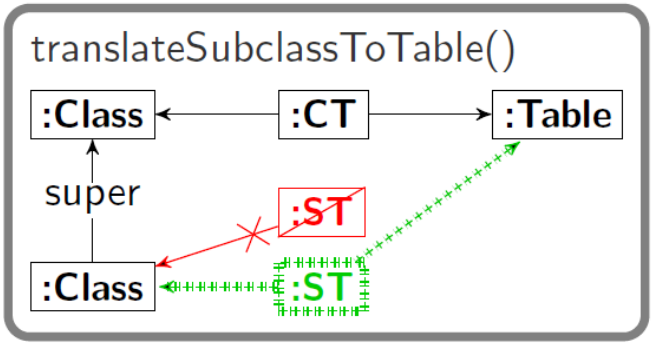
Beispiel: Übersetzungsregeln



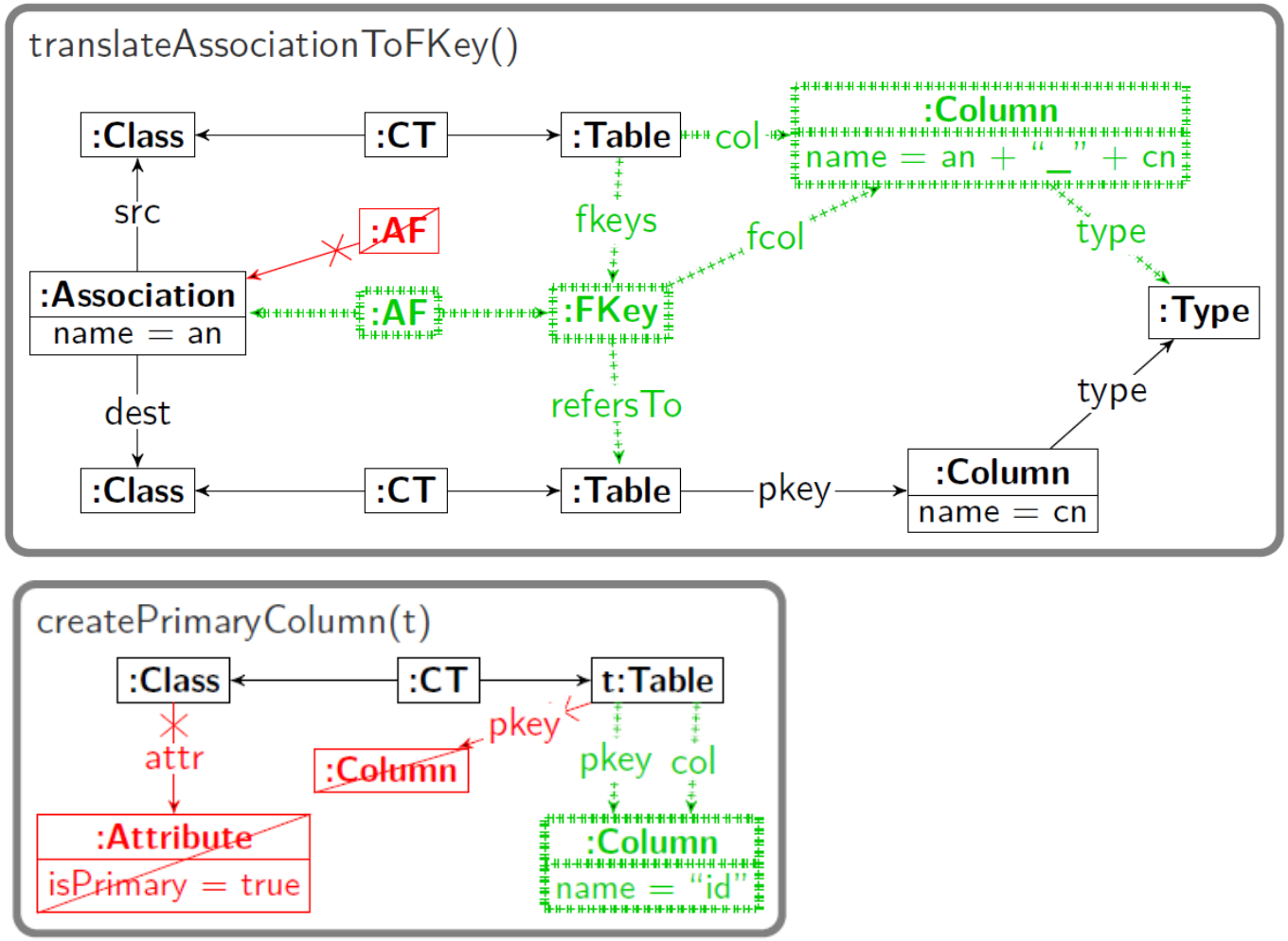








Beispiel: Übersetzungsregeln (2)



Übersetzung in Phasen

Die Übersetzungsregeln werden, wie folgt, angewandt:

Phase 1: translateClassToTable(), translateDataTypeToType()

Phase 2: translateSubclassToTable(),

Phase 3: translateAttributetoColumn(), createPrimaryColumn(), translatePrimary()

Phase 4: translateAssociationToFKey()

Graphelemente werden entlang von Typen erzeugt:

Phase 1: Alle Typen der Klassenmodelle: CT, DT, Table, Type und anhängende Kantentypen

Phase 2: ST und anhängende Kantentypen

Phase 3: AC, Column und anhängende Kantentypen (abgesehen von fcol)

Phase 4: AF, FKey und foreign columns

Die Phasen werden auch Ebenen genannt.

Termination: In Ebenen geordnete Regelanwendungen

- Gegeben: Ein GTS, dessen Regelmenge R in $1 \leq i \leq n$ Untermengen R_i (Ebenen) aufgeteilt ist.
  - *Zu jeder Ebene werden Typen zugeordnet.*
- Für jede Untermenge R_i gilt:
  - *Falls R_i löschende Regeln enthält:*
    - Jede Regel löscht mindestens ein Element. Es gehört zu einer bestehenden Ebene.
    - Jede Regel erzeugt nur Elemente der nächsten Ebene.
  - *Falls R_i keine löschende Regeln enthält:*
    - Jede Regel hat eine NAC, die die Existenz von Elementen prüft, die sie erzeugt.
    - Jede Regel erzeugt nur Elemente der nächsten Ebene
- Dann ist GTS terminierend.

Zusammenfassung

- Funktionales Verhalten:
  - *Ein konfluentes und terminierendes Graphtransformations-system zeigt ein funktionales Verhalten.*
- Hinreichende Kriterien für die Konfluenz eines GTS:
  - *Alle Regelpaare sind parallel unabhängig oder:*
  - *Alle kritischen Paare sind strikt konfluent und das Graphtransformationssystem terminiert.*
- Hinreichende Kriterien für die Termination eines GTS:
  - *Es gibt eine absteigende Ordnung von Regelanwendungen.*
  - *Regelanwendungen können in Ebene eingeteilt werden. Innerhalb einer Ebene werden nur Elemente der nächsten Ebenen erzeugt und nur Elemente der bestehenden Ebenen gelöscht oder die Anzahl der Regelansätze reduziert.*

Literatur

- [HT20] Reiko Heckel, Gabriele Taentzer: Graph Transformation for Software Engineers, Springer, 2020, Kapitel 4 und 12
- [EEPT06] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, Gabriele Taentzer: Fundamentals of Algebraic Graph Transformation, Springer, 2006
  - *Kapitel 3, 5 und 14*
- [EEL+05] Hartmut Ehrig, Karsten Ehrig, Juan de Lara, Gabriele Taentzer, Dániel Varró, Szilvia Varró-Gyapay: Termination Criteria for Model Transformation. Int. Conf. On Fundamental Approaches to Software Engineering 2005, Springer: 49–63
- [Plu98] Detlef Plump. Termination of graph rewriting is undecidable. *Fundam. Inform.* 33(2):201–209, 1998
- [DM79] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, 1979