A generalization of termination conditions for partial model completion

Fazle Rabbi[1,2] Yngve Lamo[1] Lars Michael Kristensen[1]
Ingrid Chieh Yu[2]

[1]Bergen University College, Bergen, Norway
[2]University of Oslo, Oslo, Norway
Fazle.Rabbi@hib.no, Yngve.Lamo@hib.no, ingridcy@ifi.uio.no, Lars.Michael.Kristensen@hib.no

October 22, 2015

Introduction: Why do we need Domain-specific languages (DSLs)?

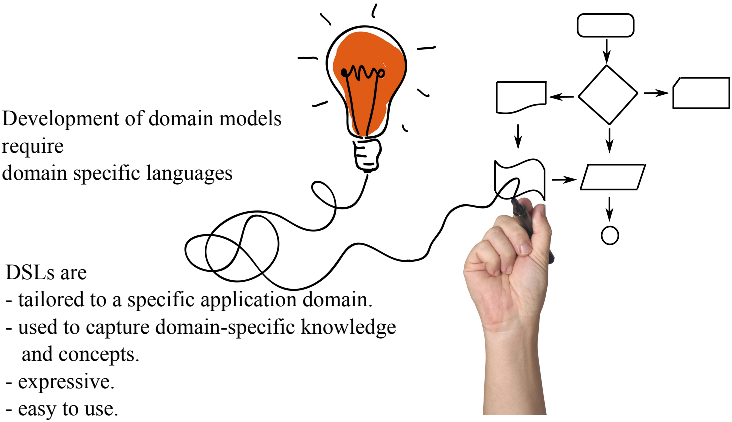


Figure : MDE focuses on exploiting domain models

Introduction: How to develop DSLs?

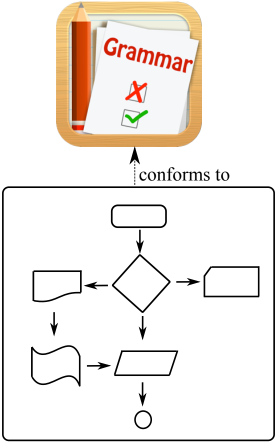


Figure : How to develop domain specific modelling languages

Introduction: Complexity of developing DSLs

DSL
development is hard, requiring
- language development skill
- domain knowledge

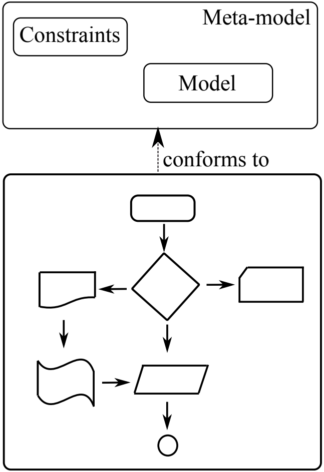


Figure : Metamodelling for DSL development

Introduction: Consistency management

Attached OCL constraint

```
context Ward
inv rule:
    self.caregivers ->
      includesAll(self.department.caregivers)
```

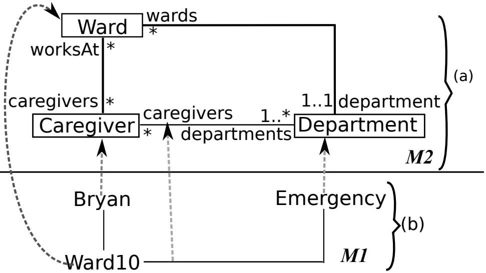


Figure : (a) Model M2, (b) a partial model M1 (not conforming to M2)

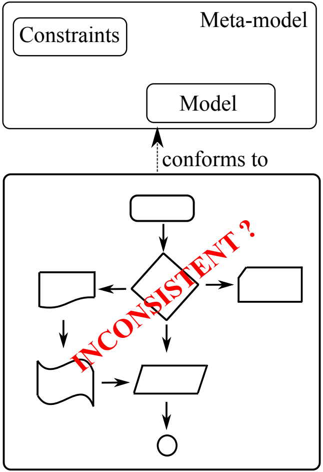
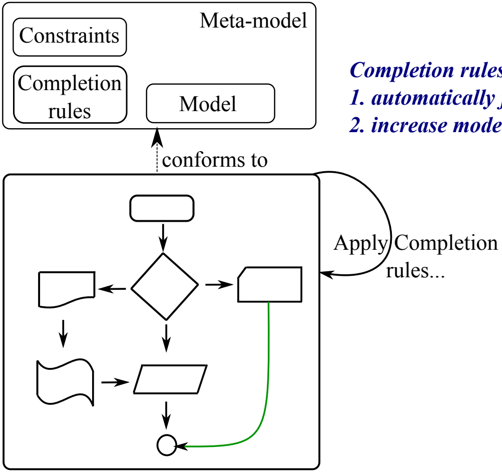


Figure : Inconsistent model

Introduction: Consistency management



Completion rules may
1. automatically fix an inconsistent model
2. increase modelling efficiency

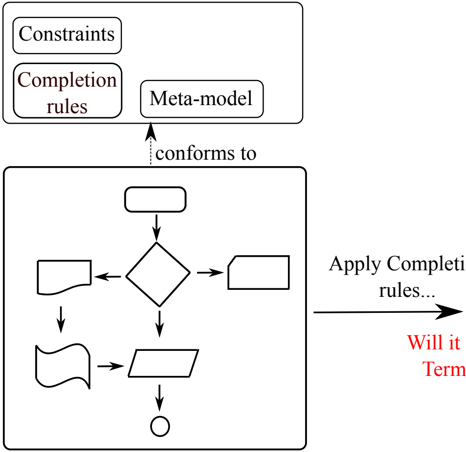
Figure : Consistency management by fixing inconsistencies

Challenges

Can we reuse completion rules ?

Can we automatically construct completion rules by processing constraints ?

Will it always find a consistent model ?



Apply Completion rules...

Will it Terminate ?

Multilevel metamodelling

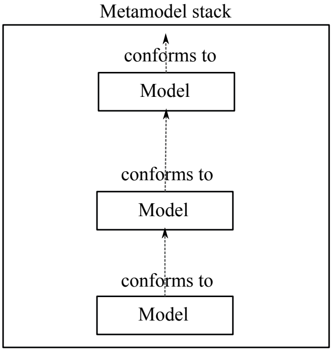
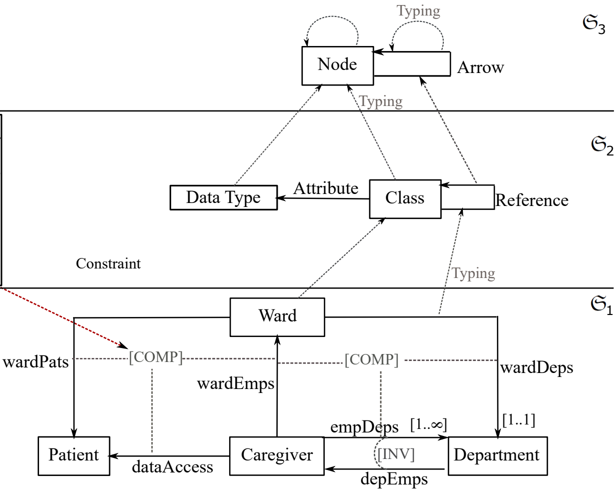


Figure : Multilevel metamodelling

- Multilevel metamodelling offers a clean, simple and coherent semantics for metamodelling [Atkinson and Kühne, 2001]
- It is an essential requirement for the development of domain-specific modelling languages

Diagram Predicate Framework (DPF) [Rutle, 2010]

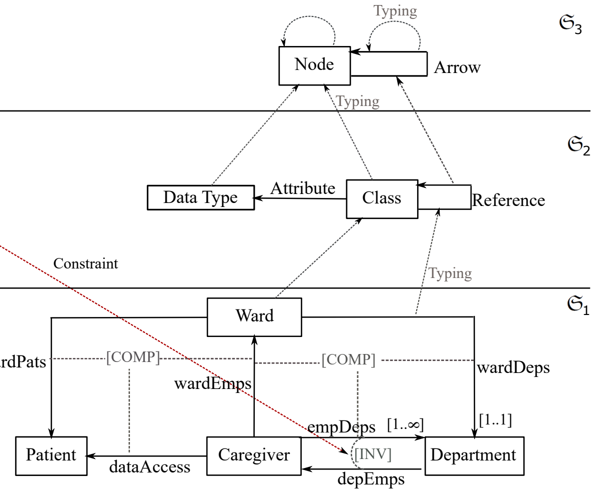
Predicate, p	$\alpha^{\Sigma}(p)$
[mult(n,m)]	$X \xrightarrow{f} Y$
[inverse]	$\begin{tikzcd} X \arrow[r, bend left, "f"] & Y \arrow[l, bend left, "g"] \end{tikzcd}$
[composite]	$\begin{tikzcd} Y \arrow[dr, "g"] & \\ X \arrow[u, "f"] \arrow[r, "h"'] & Z \end{tikzcd}$



Semantic interpretation of [composite]:
For each instance of *(f; g)*, there exists an instance of *h*.

Diagram Predicate Framework (DPF) [Rutle, 2010]

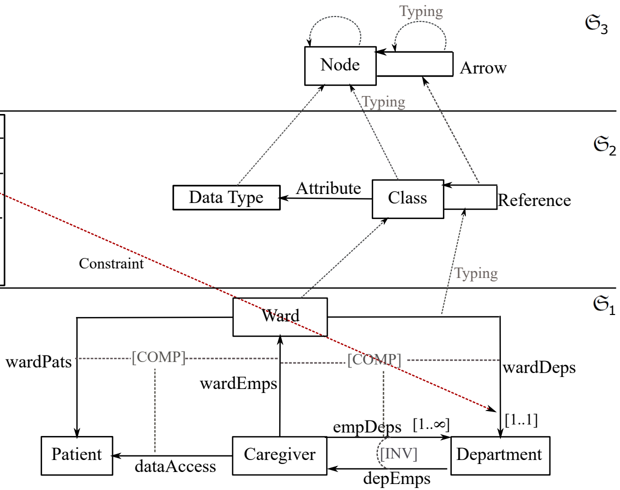
Predicate, p	$\alpha^{\Sigma}(p)$
[mult(n,m)]	$X \xrightarrow{f} Y$
[inverse]	$X \underset{g}{\overset{f}{\rightleftarrows}} Y$
[composite]	$X \xrightarrow{f} Y \xrightarrow{g} Z$, $X \xrightarrow{h} Z$



Semantic interpretation of [inverse]:
For each instance of *f* there exists an instance of *g* or vice

Diagram Predicate Framework (DPF) [Rutle, 2010]

Predicate, p	$\alpha^{\Sigma}(p)$
[mult(n,m)]	$X \xrightarrow{f} Y$
[inverse]	$X \underset{g}{\overset{f}{\rightleftarrows}} Y$
[composite]	$X \xrightarrow{f} Y$, $Y \xrightarrow{g} Z$, $X \xrightarrow{h} Z$



Semantic interpretation of [mult(n,m)]:
f must have at least *n* and at most *m* instances

Diagram Predicate Framework (DPF) [Rutle, 2010]

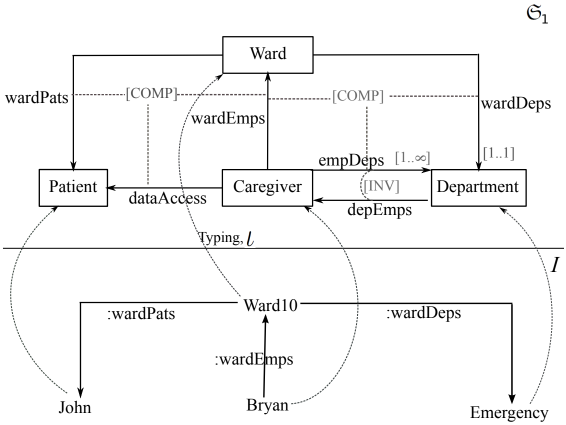


Figure : Conformance checking

Diagram Predicate Framework (DPF) [Rutle, 2010]

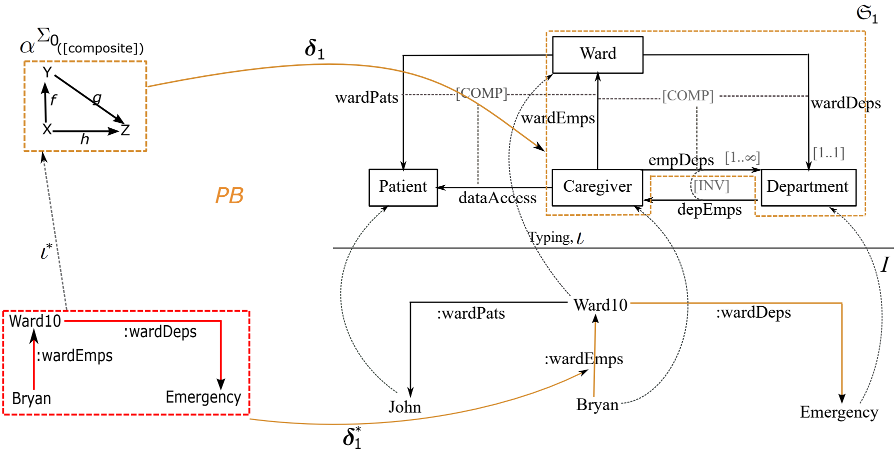


Figure : Pullback $\alpha^{\Sigma_0}([composition]) \xleftarrow{\iota^*} O^* \xrightarrow{\delta_1^*} I$ of $\alpha^{\Sigma_0}([composition]) \xrightarrow{\delta_1} S \xleftarrow{\iota} I$

An inconsistent instance

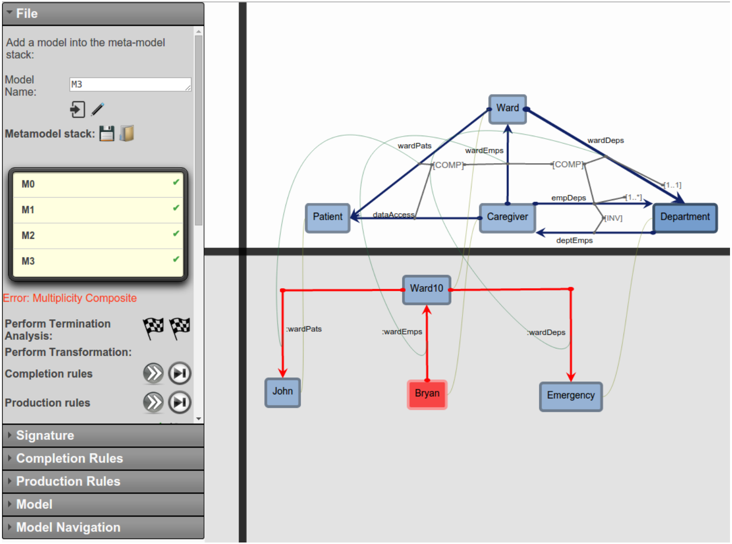


Figure : An inconsistent instance

Diagrammatic model completion

- Diagrammatic model completion is based on completion rules
- Completion rules are typed coupled transformation rules
- Type graphs of completion rules are not changed by the transformation
- Completion rules are linked to predicates
- Completion rules are applied to a partial model to correct inconsistencies
- We use the standard double-pushout (DPO) approach [Ehrig, 2006] for defining completion rules.

Diagrammatic model completion

- Diagrammatic model completion is based on completion rules
- Completion rules are typed coupled transformation rules
- Type graphs of completion rules are not changed by the transformation
- Completion rules are linked to predicates
- Completion rules are applied to a partial model to correct inconsistencies
- We use the standard double-pushout (DPO) approach [Ehrig, 2006] for defining completion rules.

$$
\begin{tikzcd}
\alpha^{\Sigma}(p) & \alpha^{\Sigma}(p) \arrow[l, "id"'] & \alpha^{\Sigma}(p) \arrow[l, "id"'] \arrow[r, "id"] & \alpha^{\Sigma}(p) \\
N \arrow[u, dotted, "\iota_N"] & L \arrow[l, "n"] \arrow[u, dotted, "\iota_L"'] & K \arrow[l, "k"] \arrow[r, "l"'] \arrow[u, dotted, "\iota_K"'] & R \arrow[u, dotted, "\iota_R"']
\end{tikzcd}
$$

Figure : A completion rule

Diagrammatic model completion

- Diagrammatic model completion is based on completion rules
- Completion rules are typed coupled transformation rules
- Type graphs of completion rules are not changed by the transformation
- Completion rules are linked to predicates
- Completion rules are applied to a partial model to correct inconsistencies
- We use the standard double-pushout (DPO) approach [Ehrig, 2006] for defining completion rules.

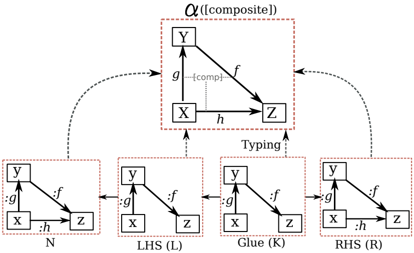


Figure : A transformation rule is linked to the [*composite*] predicate

Diagrammatic model completion

- Diagrammatic model completion is based on completion rules
- Completion rules are typed coupled transformation rules
- Type graphs of completion rules are not changed by the transformation
- Completion rules are linked to predicates
- Completion rules are applied to a partial model to correct inconsistencies
- We use the standard double-pushout (DPO) approach [Ehrig, 2006] for defining completion rules.

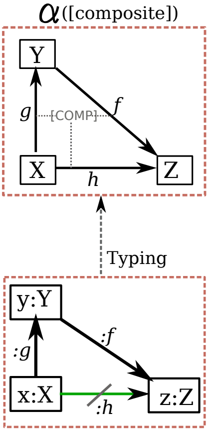
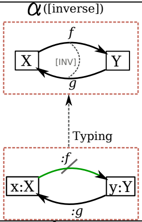
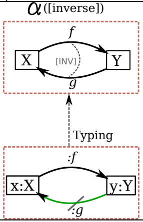
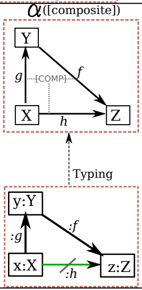


Figure : A completion rule

Diagrammatic model completion

Table : Completion scheme for predicates and completion rules of $\mathfrak{S}_1$

C_p	$\zeta(C_p)$		Interpretation
$[\text{inv-com}\rangle_{[inv]}$			derive an edge $y \xrightarrow{:g} x$ (if it does not exist) from the existence of an edge $x \xrightarrow{:f} y$ or vice versa.
$[\text{comp-com}\rangle_{[comp]}$			derive an edge $x \xrightarrow{:h} z$ (if it does not exist) from the existence of edges $x \xrightarrow{:g} y$ and $y \xrightarrow{:f} z$.

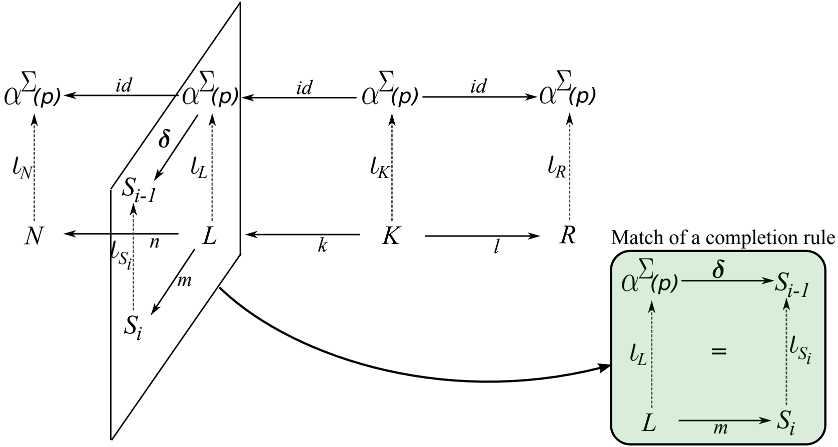
Application of a completion rule

$$
\begin{tikzcd}
\alpha^{\Sigma}(p) & \alpha^{\Sigma}(p) \arrow[l, "id"'] & \alpha^{\Sigma}(p) \arrow[l, "id"'] \arrow[r, "id"] & \alpha^{\Sigma}(p) \\
N \arrow[u, dashed, "\iota_N"] & L \arrow[l, "n"] \arrow[u, dashed, "\iota_L"] & K \arrow[l, "k"] \arrow[r, "l"'] \arrow[u, dashed, "\iota_K"] & R \arrow[u, dashed, "\iota_R"]
\end{tikzcd}
$$

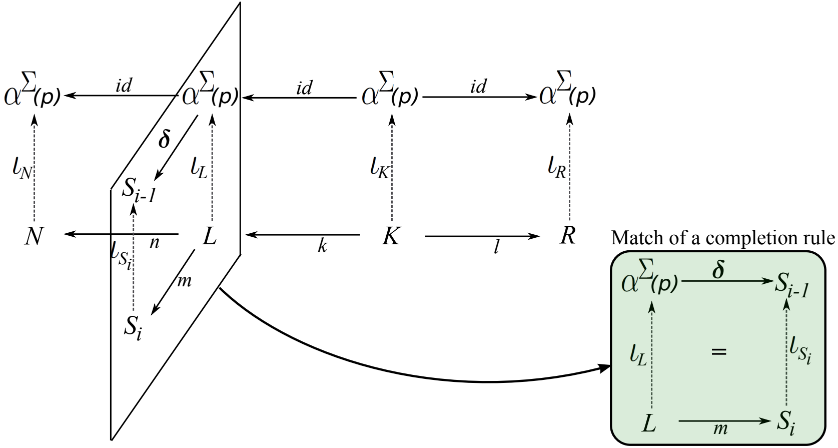
Application of a completion rule

$$
\begin{tikzcd}
\alpha^{\Sigma}(p) & & \alpha^{\Sigma}(p) \arrow[ll, "id"'] & \alpha^{\Sigma}(p) \arrow[l, "id"'] \arrow[r, "id"] & \alpha^{\Sigma}(p) \\
& S_{i\text{-}1} & & & \\
N \arrow[uu, dotted, "\iota_N"] & & L \arrow[ll, "n"] \arrow[uu, dotted, "\iota_L"'] & K \arrow[l, "k"] \arrow[uu, dotted, "\iota_K"'] \arrow[r, "l"'] & R \arrow[uu, dotted, "\iota_R"'] \\
& S_i \arrow[uu, dotted, "\iota_{S_i}"] & & &
\end{tikzcd}
$$

Application of a completion rule



Application of a completion rule



A match (δ, m) is given by an atomic constraint $\delta : \alpha^{\Sigma}(p) \to S_{i-1}$ and a match $m : L \to S_i$ such that the constraint δ and match m together with typing morphisms $\iota_L : L \to \alpha^{\Sigma}(p)$ and $\iota_{S_i} : S_i \to S_{i-1}$ constitute a commuting square: $\iota_L ; \delta = m ; \iota_{S_i}$

Application of a completion rule

$$
\begin{tikzcd}
\alpha^{\Sigma}(p) \arrow[dr] & & \alpha^{\Sigma}(p) \arrow[ll, "id"'] \arrow[dl, "\delta"] & \alpha^{\Sigma}(p) \arrow[l, "id"'] \arrow[r, "id"] & \alpha^{\Sigma}(p) \\
& S_{i\text{-}1} & & & \\
N \arrow[uu, dashed, "\iota_N"] \arrow[dr, "q"'] \arrow[dr, phantom, "\neq", shift left=3] & & L \arrow[ll, "n"] \arrow[uu, dashed, "\iota_L"'] \arrow[dl, "m"] & K \arrow[l] \arrow[uu, dashed, "\iota_K"'] \arrow[r] & R \arrow[uu, dashed, "\iota_R"'] \\
& S_i \arrow[uu, dashed, "\iota_{S_i}"] & & &
\end{tikzcd}
$$

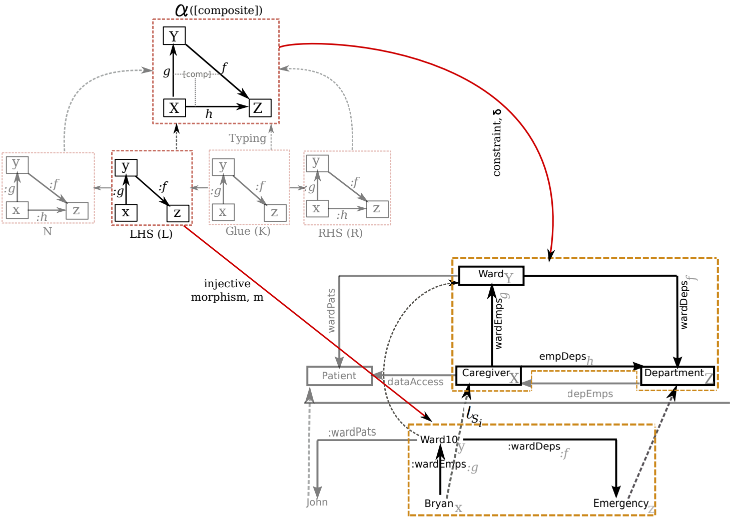
$(\delta, m) \models NAC$ if there does not exists an injective morphism $q : N \to S_i$ with $n; q = m$ such that the typing morphisms

$\iota_N : N \to \alpha^{\Sigma}(p)$ and $\iota_{S_i} : S_i \to S_{i-1}$ constitute a commuting square $\iota_N; \delta = q; \iota_{S_i}$.

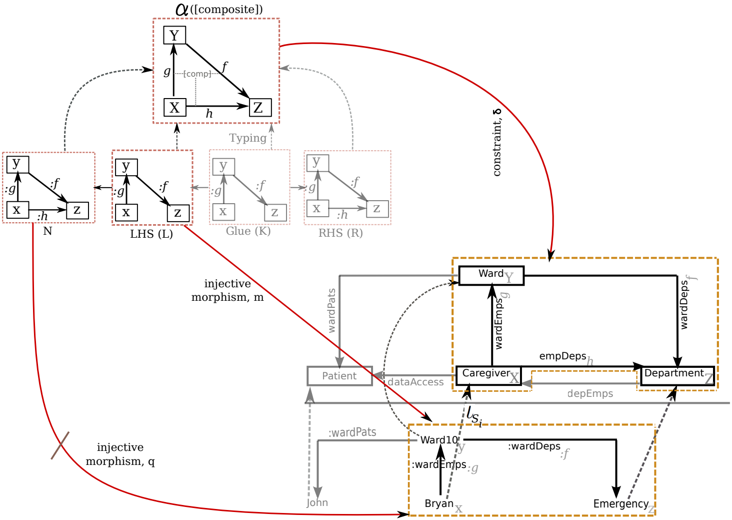
Application of a completion rule

$$
\begin{tikzcd}
\alpha^{\Sigma}(p) \arrow[dr] & & \alpha^{\Sigma}(p) \arrow[ll, "id"'] \arrow[dl, "\delta"'] & & \alpha^{\Sigma}(p) \arrow[ll, "id"'] \arrow[rr, "id"] \arrow[dl, "\delta"'] & & \alpha^{\Sigma}(p) \arrow[dl, "\delta"'] \\
& S_{i\text{-}1} & & S_{i\text{-}1} \arrow[ll] \arrow[rr] & & S_{i\text{-}1} & \\
N \arrow[uu, dashed, "\iota_N"] \arrow[dr, "q"'] \arrow[r, phantom, "\neq"] & & L \arrow[ll, "n"] \arrow[uu, dashed, "\iota_L"'] \arrow[dl, "m"] \arrow[dr, phantom, "(PO1)"] & & K \arrow[ll] \arrow[rr] \arrow[uu, dashed, "\iota_K"'] \arrow[dl] \arrow[dr, phantom, "(PO2)"] & & R \arrow[uu, dashed, "\iota_R"'] \arrow[dl, "m*"] \\
& S_i \arrow[uu, dashed, "\iota_{S_i}"] & & S'_i \arrow[ll] \arrow[rr] \arrow[uu, dashed] & & S^*_i \arrow[uu, dashed] &
\end{tikzcd}
$$

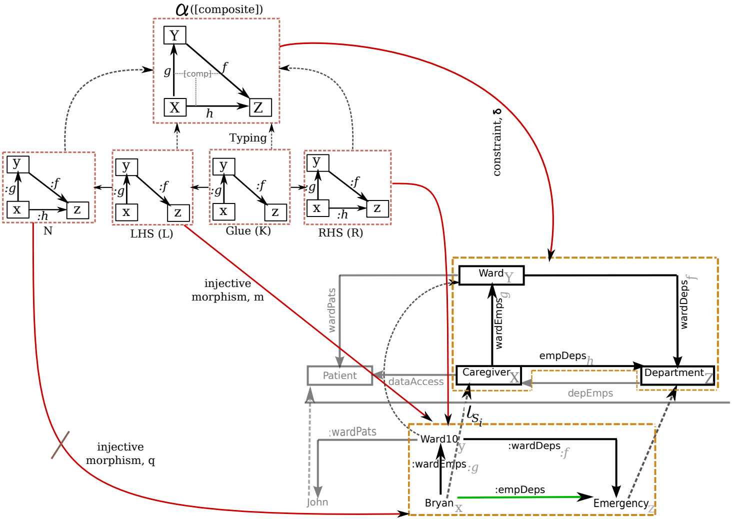
Example: Application of a completion rule



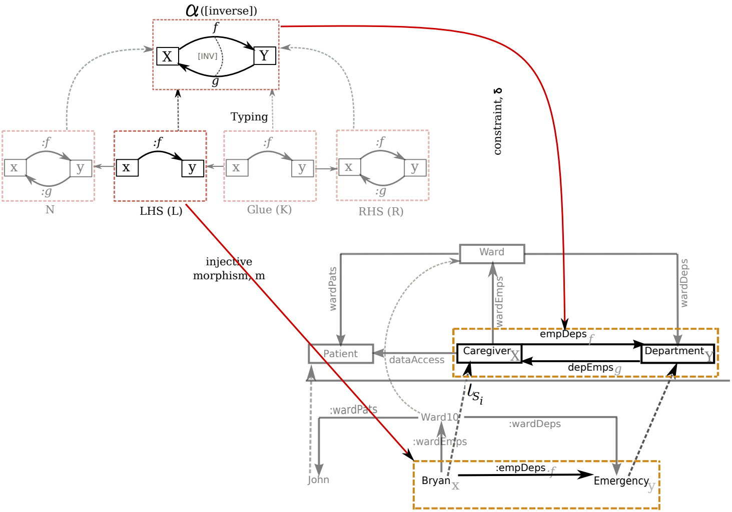
Example: Application of a completion rule



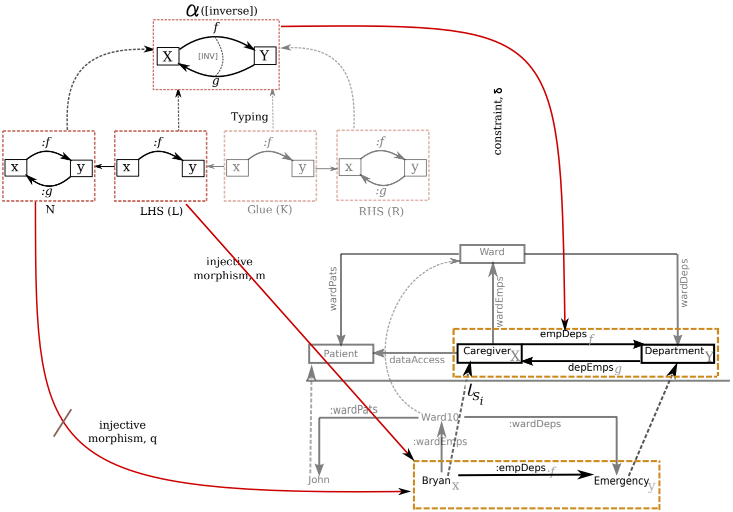
Example: Application of a completion rule



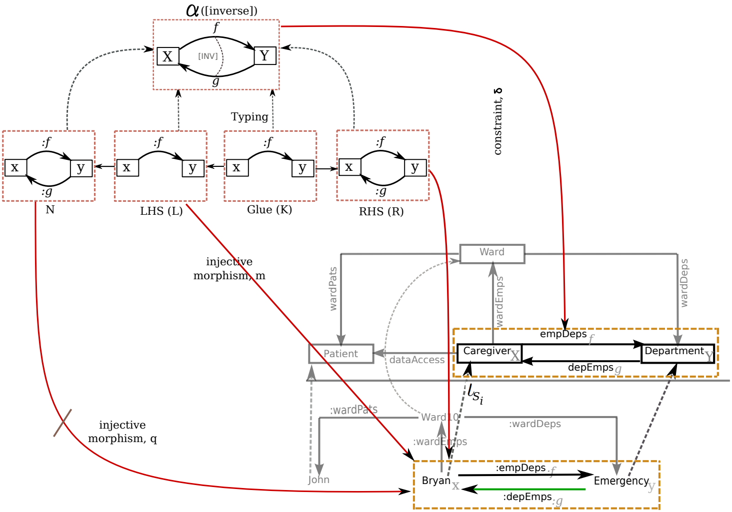
Example: Application of a completion rule



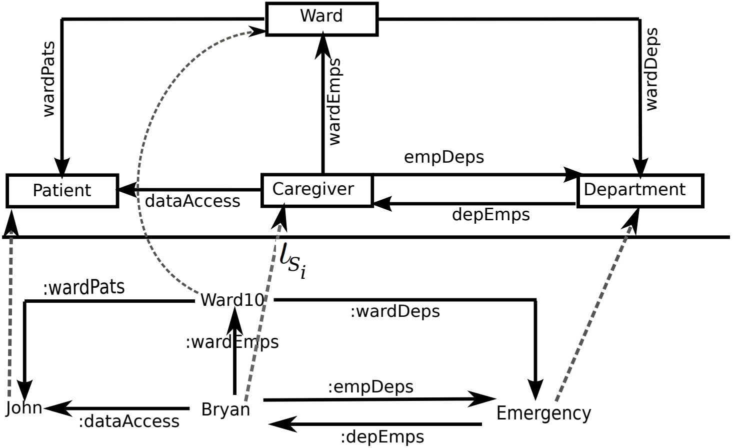
Example: Application of a completion rule



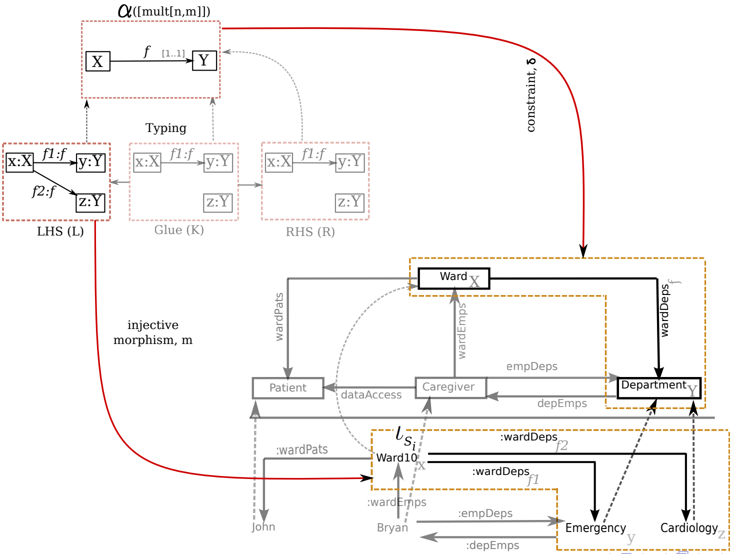
Example: Application of a completion rule



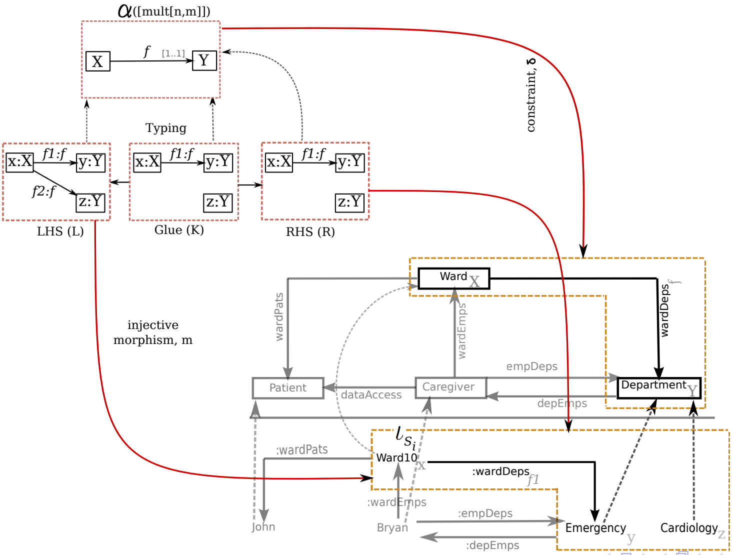
Example: Application of a completion rule



Example: Application of a completion rule that deletes elements



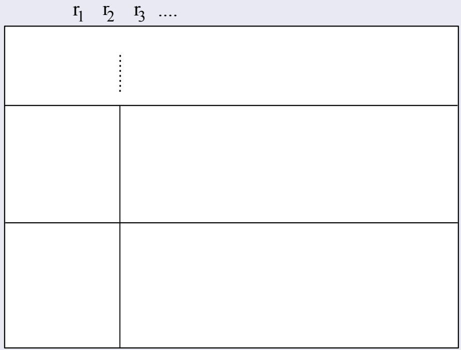
Example: Application of a completion rule that deletes elements



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

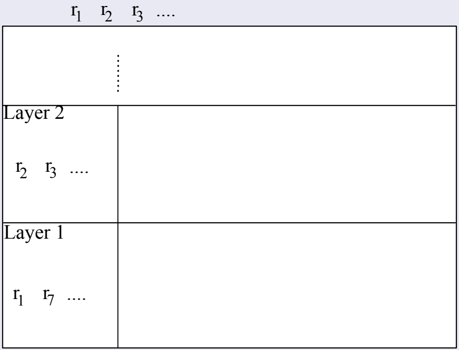
Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

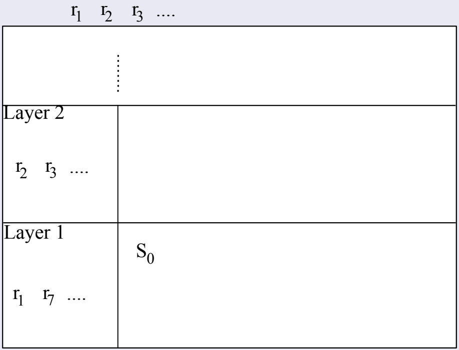
Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

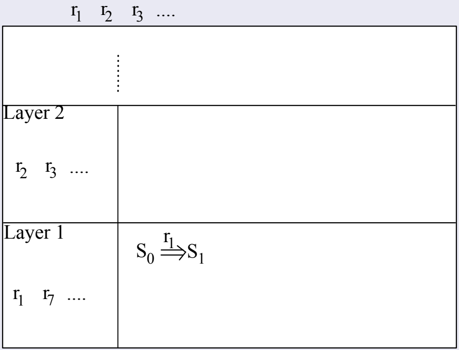
Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

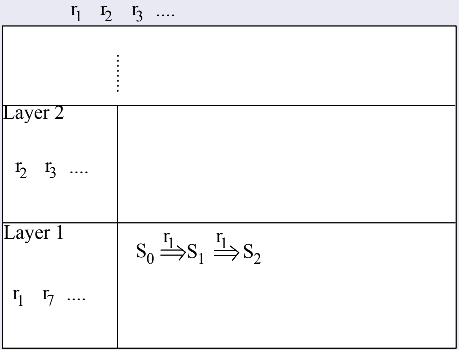
Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

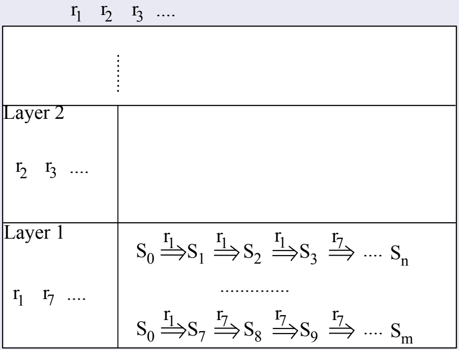
Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

Generalized layered approach



Termination criteria

- Based on the principles adapted from layered graph grammars [Ehrig, 2006].
- Completion rules are distributed across different layers.
- Rules of a layer are applied as long as possible before going to the next layer.
- We generalize the layer conditions from [Ehrig, 2006] allowing deleting and non-deleting rules to reside in the same layer as long as the rules are loop-free.

Generalized layered approach

$r_1 \quad r_2 \quad r_3 \quad ....$

	⋮
Layer 2 $r_2 \quad r_3 \quad ....$	$S_n \xRightarrow{r_2} S_{21} \xRightarrow{r_2} S_{22} \xRightarrow{r_3} S_{23} \xRightarrow{r_3} .... S_t$ .... $S_m \xRightarrow{r_2} S_{33} \xRightarrow{r_3} S_{34} \xRightarrow{r_3} S_{35} \xRightarrow{r_3} .... S_v$
Layer 1 $r_1 \quad r_7 \quad ....$	$S_0 \xRightarrow{r_1} S_1 \xRightarrow{r_1} S_2 \xRightarrow{r_1} S_3 \xRightarrow{r_7} .... S_n$ .... $S_0 \xRightarrow{r_1} S_7 \xRightarrow{r_7} S_8 \xRightarrow{r_7} S_9 \xRightarrow{r_7} .... S_m$

Termination criteria

Our generalized layered approach is based on a necessary condition ($C1 \vee C2 \vee C3$) for looping, where:

- C1: A rule r_i that creates an element x of type t does not have a *NAC* that forbids the existence of element of type t.
- C2: If a rule r_i creates an element x of type t and has a *NAC* that forbids the existence of element of type t, then there exists a rule r_j that deletes an element of type t.
- C3: If a rule r_i deletes an element x of type t, then there exists a rule r_j that creates an element of type t.

Termination criteria

Lemma 1. $(C1 \vee C2 \vee C3)$ is a necessary condition for looping of a set of rules at layer k

Proof: Let $G_0 = S_i$ be an initial graph typed by S_{i-1} where S_i, S_{i-1} are finite graphs. Let R_k be a finite set of rules at layer k. A rule $r \in R_k$ can either

1. creates an element x of type t, where
   - a r does not have a *NAC* that forbids the existence of element of type t.
     or
   - b r has a *NAC* that forbids the existence of element of type t.

   and/or
2. deletes an element x' of type t'

Termination criteria

Lemma 1. $(C1 \vee C2 \vee C3)$ is a necessary condition for looping of a set of rules at layer k

Proof:
A rule $r \in R_k$ can either

1. creates an element x of type t, where
   a. r does not have a *NAC* that forbids the existence of element of type t.
      or
   b. r has a *NAC* that forbids the existence of element of type t.

   and/or
2. deletes an element x' of type t'

Consider case 1.(a):
The rule r has finite number of injective matches $c_r = \{(\delta, m) \mid (\delta, m)\ \textit{is a match for}\ G_0 \rhd S_{i-1}\}$.
For each injective match of $L \rightarrow G_0$, application of r creates an element x of type t.
The rule can be applied indefinitely in a loop during the derivation process of layer k since the application of rule r does not decreases the number of matches.
Therefore $C1$ is a necessary condition for looping.

Termination criteria

Lemma 1. $(C1 \vee C2 \vee C3)$ is a necessary condition for looping of a set of rules at layer k

Proof:
A rule $r \in R_k$ can either

1. creates an element x of type t, where
   a. r does not have a *NAC* that forbids the existence of element of type t.
   or
   b. r has a *NAC* that forbids the existence of element of type t.

   and/or
2. deletes an element x' of type t'

Consider case 1.(b):
The rule r has finite number of injective matches $c_r = \{(\delta, m) \mid (\delta, m)$ *is a match for* $G_0 \triangleright S_{i-1}$ *and* $(\delta, m) \models NAC\}$. For each injective match of $L \rightarrow G_0$, application of r creates an element x of type t.
Therefore, the application of rule r decreases the number of matches.
In order to apply r indefinitely in a loop during the derivation process of layer k, elements of type t must be deleted.
Therefore $C2$ is a necessary condition for looping.

Termination criteria

Lemma 1. $(C1 \vee C2 \vee C3)$ is a necessary condition for looping of a set of rules at layer k

Proof:
A rule $r \in R_k$ can either

1. creates an element x of type t, where
   a. r does not have a *NAC* that forbids the existence of element of type t.
      or
   b. r has a *NAC* that forbids the existence of element of type t.

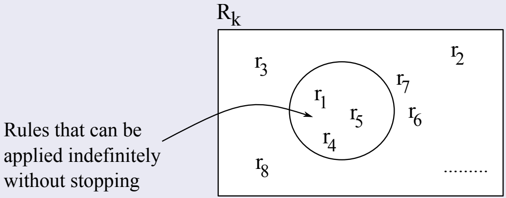
   and/or
2. deletes an element x' of type t'

Consider the first case 2:
The rule r has finite number of injective matches $c_r = \{(\delta, m) \mid (\delta, m)$ *is a match for* $G_0 \triangleright S_{i-1}$ *and* $(\delta, m) \models NAC\}$.
In order to apply r indefinitely in a loop during the derivation process, new elements of type t' must be created.
Therefore $C3$ is a necessary condition for looping.

Termination criteria

Corollary 1. A sufficient condition for loop-free rules in layer k is the negation of $(C1 \vee C2 \vee C3)$

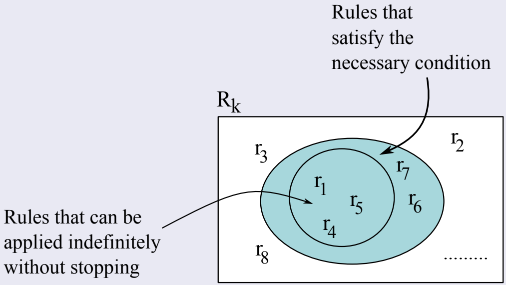
Generalized layered approach



Termination criteria

Corollary 1. A sufficient condition for loop-free rules in layer k is the negation of $(C1 \vee C2 \vee C3)$

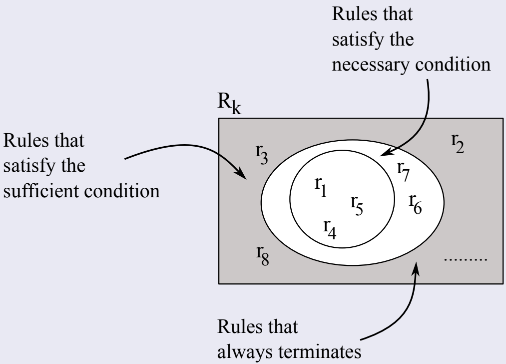
Generalized layered approach



Termination criteria

Corollary 1. A sufficient condition for loop-free rules in layer k is the negation of $(C1 \vee C2 \vee C3)$

Generalized layered approach

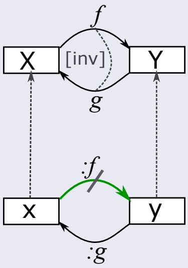


Termination criteria

We propose a loop detection algorithm that is based on the following sufficient conditions for loop freeness. Let R_k be the set of rules of a layer k.

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its NAC,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach



Termination criteria

We propose a loop detection algorithm that is based on the following sufficient conditions for loop freeness. Let R_k be the set of rules of a layer k.

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its *NAC*,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach

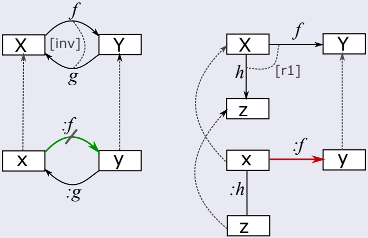
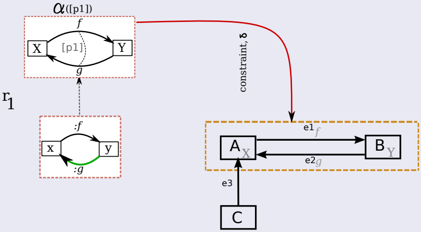


Figure : These rules may produce a non-terminating situation if they are executed in the same layer

Termination criteria

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its *NAC*,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach

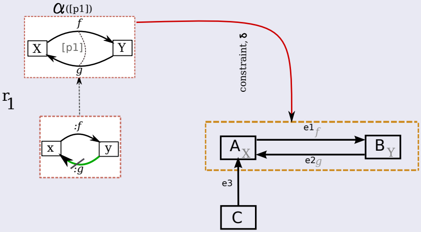


	NAC(e2)	R(e2)	NAC(e3)	R(e3)	NAC(e1)	R(e1)	NAC(A)	R(A)	NAC(B)	R(B)	NAC(C)	R(C)
r_1		✓										

Termination criteria

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its *NAC*,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach

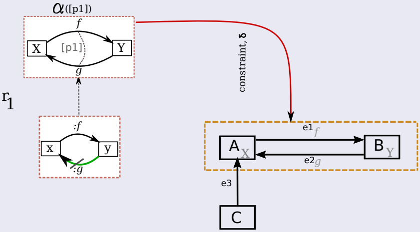


	NAC(e2)	R(e2)	NAC(e3)	R(e3)	NAC(e1)	R(e1)	NAC(A)	R(A)	NAC(B)	R(B)	NAC(C)	R(C)
r_1	✓	✓										

Termination criteria

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its *NAC*,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach

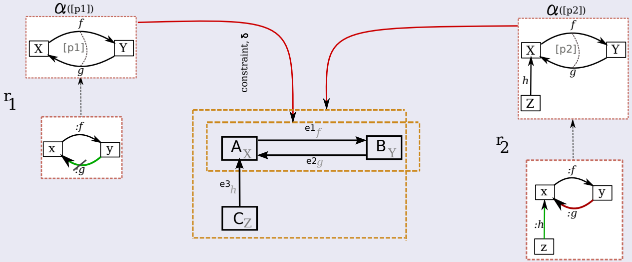


	NAC(e2)	R(e2)	NAC(e3)	R(e3)	NAC(e1)	R(e1)	NAC(A)	R(A)	NAC(B)	R(B)	NAC(C)	R(C)

Termination criteria

- If a rule $r_i \in R_k$ creates an element x of type t, then r_i must have an element of type t in its NAC,
- If a rule $r_i \in R_k$ creates an element x of type t, then there is no rule in $r_j \in R_k$ that deletes an element of type t,
- If a rule $r_i \in R_k$ deletes an element of type t, then there is no rule in $r_j \in R_k$ that creates an element of type t

Generalized layered approach



	NAC(e2)	R(e2)	NAC(e3)	R(e3)	NAC(e1)	R(e1)	NAC(A)	R(A)	NAC(B)	R(B)	NAC(C)	R(C)
r_1	✓	✓										
r_2		X		✓								

Termination criteria

Theorem 1. (termination of loop-free rules). An empty table obtained by loop free rule detection analysis for a set of rules E implies that the execution of E will terminate for any finite size initial graph.

Conclusion and Future work

Summary

- Completion rules are defined as coupled graph transformation rules
- Completion rules are reusable
- Generalized termination analysis is based on layered approach
- We have Implemented a proof-of-concept of the proposed approach

Future Work

- Improve performance of the transformation system
- Automatically construct completion rules by processing constraints
- Develop concrete graphical syntax
- Support collaborative development

References

Atkinson, C. and Kühne, T. (2001).

The essence of multilevel metamodeling.

In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, UML; '01, pages 19–33, London, UK, UK. Springer-Verlag.

Zinovy Diskin, Uwe Wolter:

A Diagrammatic Logic for Object-Oriented Visual Modeling. *Electr. Notes Theor. Comput. Sci. 203(6): 19-41 (2008)*

Adrian Rutle:

Diagram Predicate Framework: A Formal Approach to MDE. *PhD thesis, Department of Informatics, University of Bergen, Norway, November 2010.*

Gabriele Taentzer, Florian Mantz, Thorsten Arendt, Yngve Lamo:

Customizable Model Migration Schemes for Meta-model Evolutions with Multiplicity Changes. *MoDELS 2013: 254-270*

Hartmut Ehrig, Karsten Ehrig, Ulrike Prange, Gabriele Taentzer:

Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. *An EATCS Series, Springer 2006,* ISBN 978-3-540-31187-4.

Annegret Habel, Reiko Heckel, Gabriele Taentzer:

Graph Grammars with Negative Application Conditions. *Fundam. Inf., 1996* vol. 26, pp. 287–313.

The End