On-the-Fly Construction, Correctness and Completeness of Model Transformations based on Triple Graph Grammars

Hartmut Ehrig, Claudia Ermel,
Frank Hermann, and Ulrike Prange

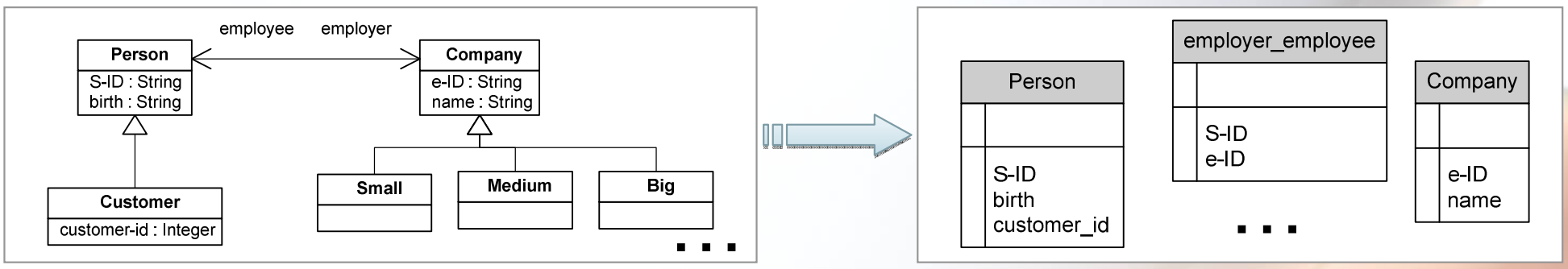
Technische Universität Berlin
Germany

Outline

1. Motivation
2. Triple Graph Grammars with NACs
3. Main Results:
   - On-the-fly construction
   - Termination
   - Correctness and Completeness
   - Partial Order Reduction
4. Conclusion and Future Work

Motivation

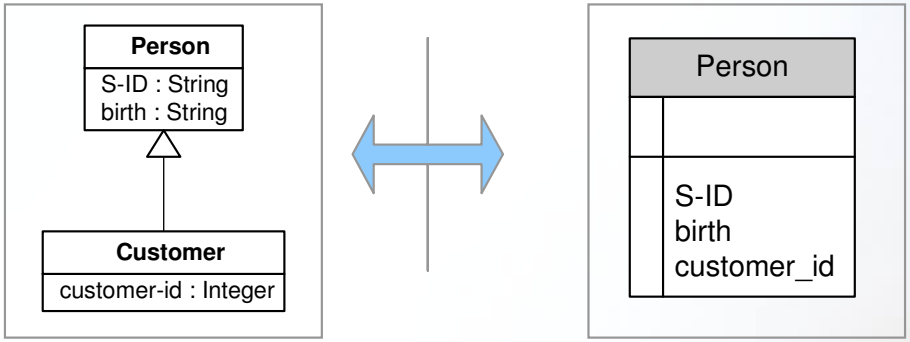
- Correct and Efficient Model Transformations



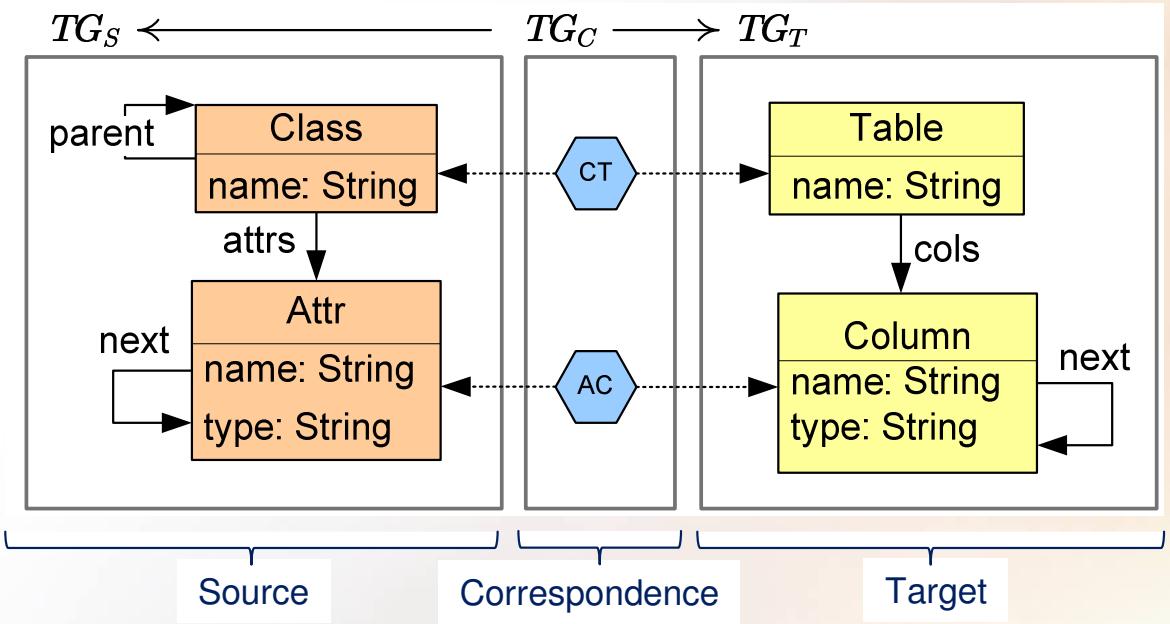
- **Triple Graph Grammars:**
  - Bidirectional Model transformations [Schürr94, SK08]
  - Derived Forward & Backward Transformation Rules [Schürr94]
  - Formal results for Correctness and Completeness [EEH08,EHS09]
  - Control Mechanisms:
    - Negative Application Conditions (NACs) [EHS09]
    - Source Consistency [EEE+07]

Example: CD2RDBM

- Relationship of instances in concrete syntax:



- Type graph in abstract syntax:



Triple Graph Grammars with NACs

Triple graphs and triple rules

- Graph $G = (V, E, s, t)$
- Category **TripleGraphs**
  - triple graph: $G = (G^S \xleftarrow{s_G} G^C \xrightarrow{t_G} G^T)$
  - triple graph morphism:

$$tr = (tr^S, tr^C, tr^T): L \to R$$

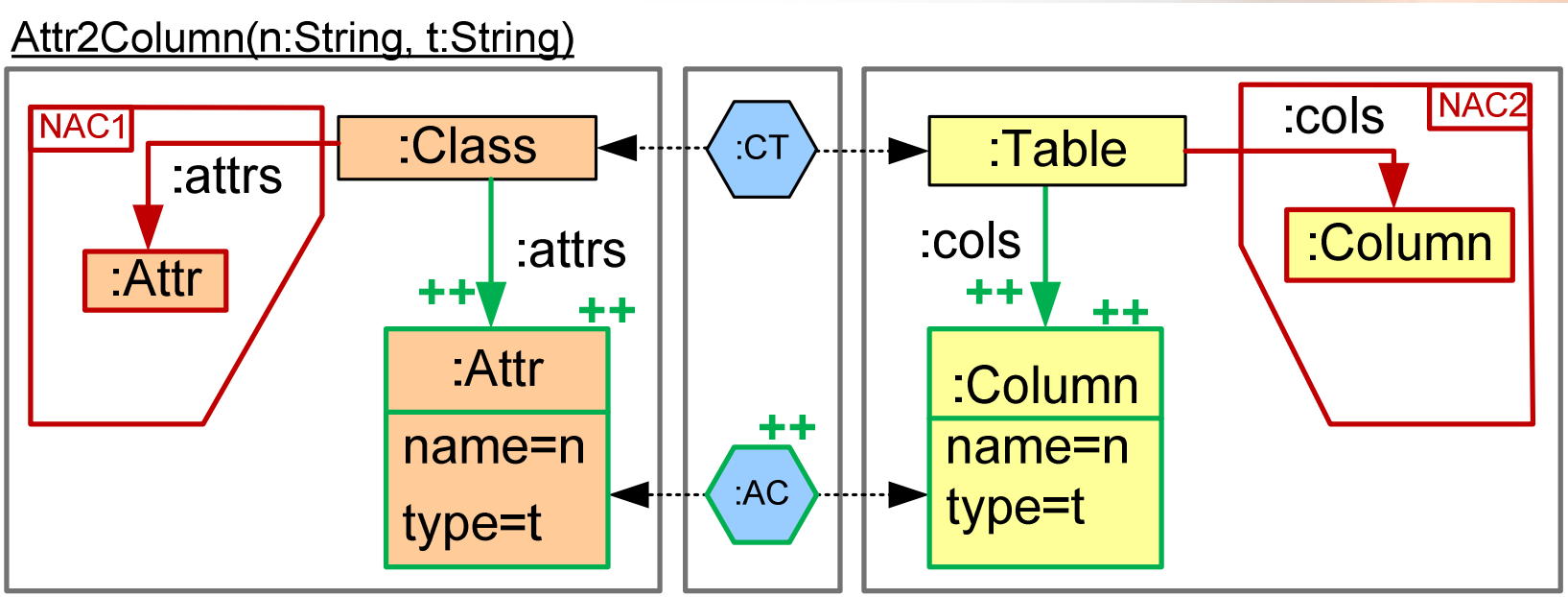
$$\begin{array}{ccccc} L = (L^S & \xleftarrow{s_L} & L^C & \xrightarrow{t_L} & L^T) \\ tr\downarrow \quad tr^S\downarrow & & tr^C\downarrow & & tr^T\downarrow \\ R = (R^S & \xleftarrow[s_R]{} & R^C & \xrightarrow[t_R]{} & R^T) \end{array}$$

- Triple rule is injective triple graph morphism: $tr: L \to R$

  $\Rightarrow$ Rules are nondeleting

  specified for synchronous creation of model elements in source and target language

Triple Rules of *CD2RDBM*

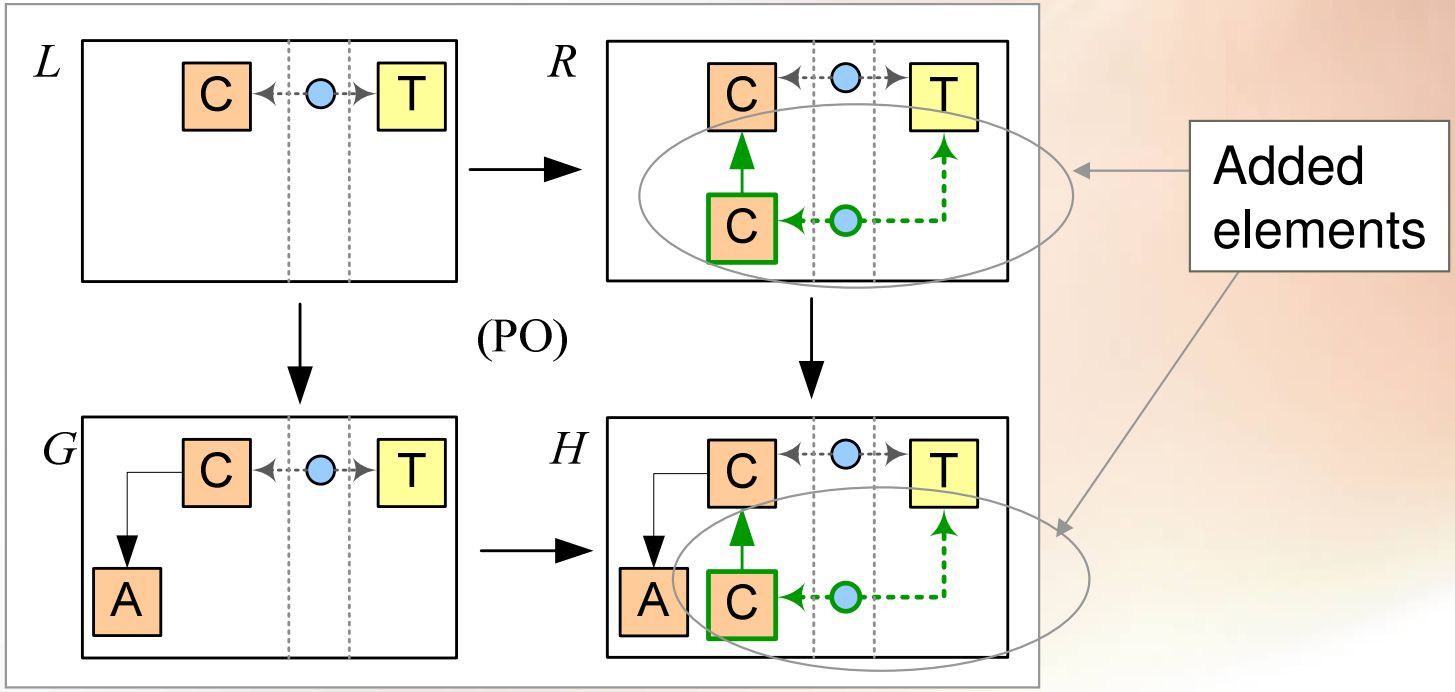


Triple Graph Transformation

- Transformation: $G \overset{tr,m,n}{\Longrightarrow} H$
  - injective match m
  - PO in **TripleGraphs**
  - H is gluing of R and G along L

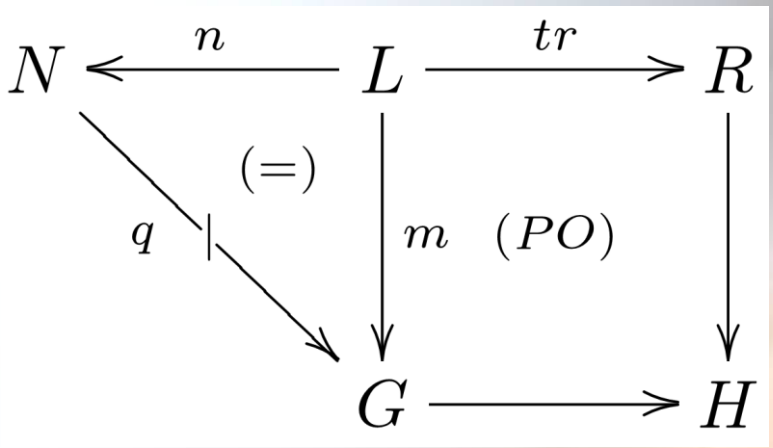
$$\begin{array}{ccc} L & \xrightarrow{tr} & R \\ {\scriptstyle m}\downarrow & (PO) & \downarrow{\scriptstyle n} \\ G & \xrightarrow[t]{} & H \end{array}$$

- Example:



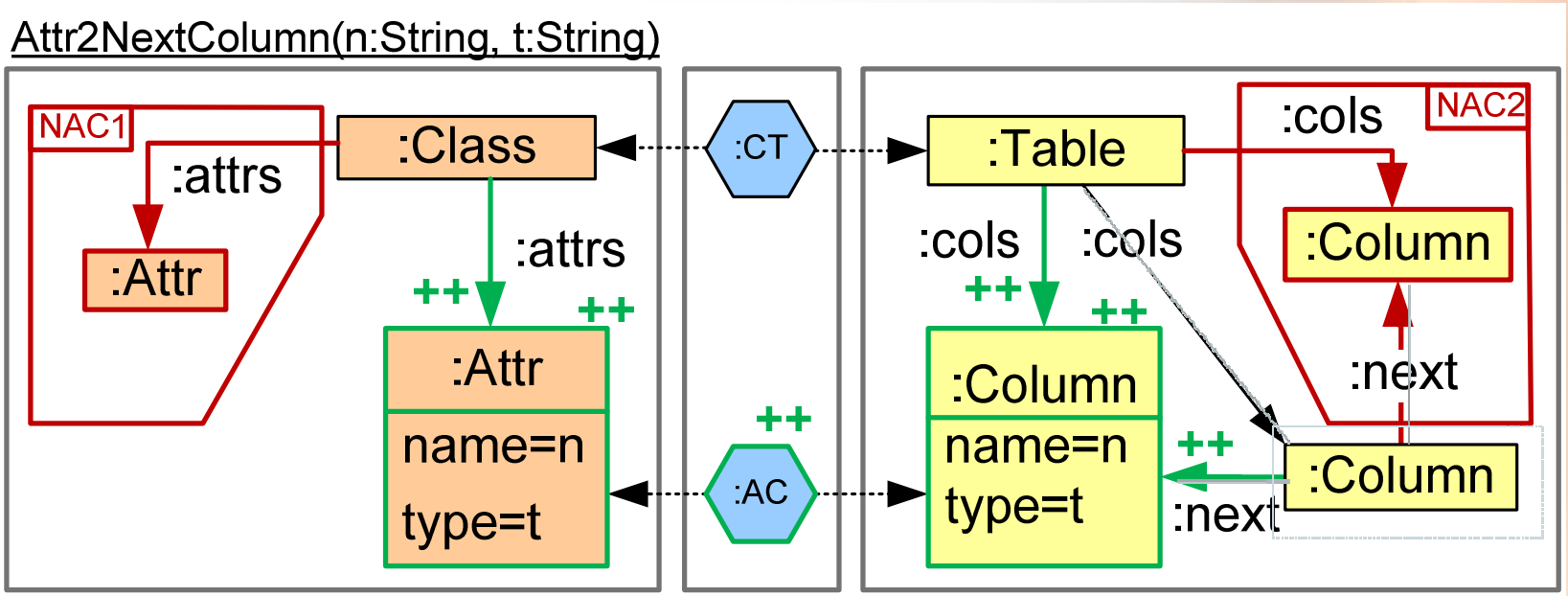
Negative Application Conditions

- **General NAC** (n,N):
  n: $L \to N$ is injective
  triple graph morphism

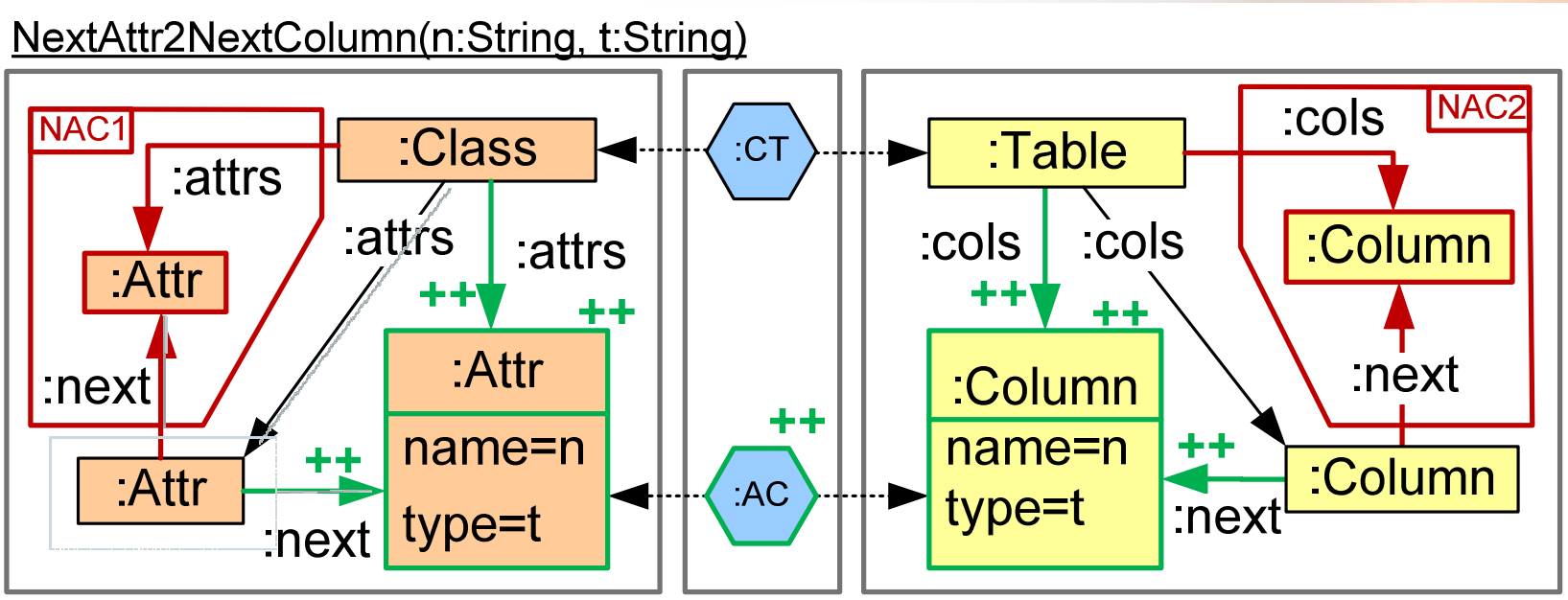


- Match m: $L \to G$ is **NAC consistent:** $\nexists$ injective q: $N \to G$ s. t. $q \circ n = m$ for all NACs of tr
- **Source NAC** (n,N): $n=(n^S, id_{L^C}, id_{L^T})$, i.e. NAC on source component only
- **Target NAC** (n,N): $n=(id_{L^S}, id_{L^C}, n^T)$, i.e. NAC on target component only
- In most cases: Source-Target NACs sufficient

Triple Rules of *CD2RDBM*



Triple Rules of *CD2RDBM*



Derived Source Rule

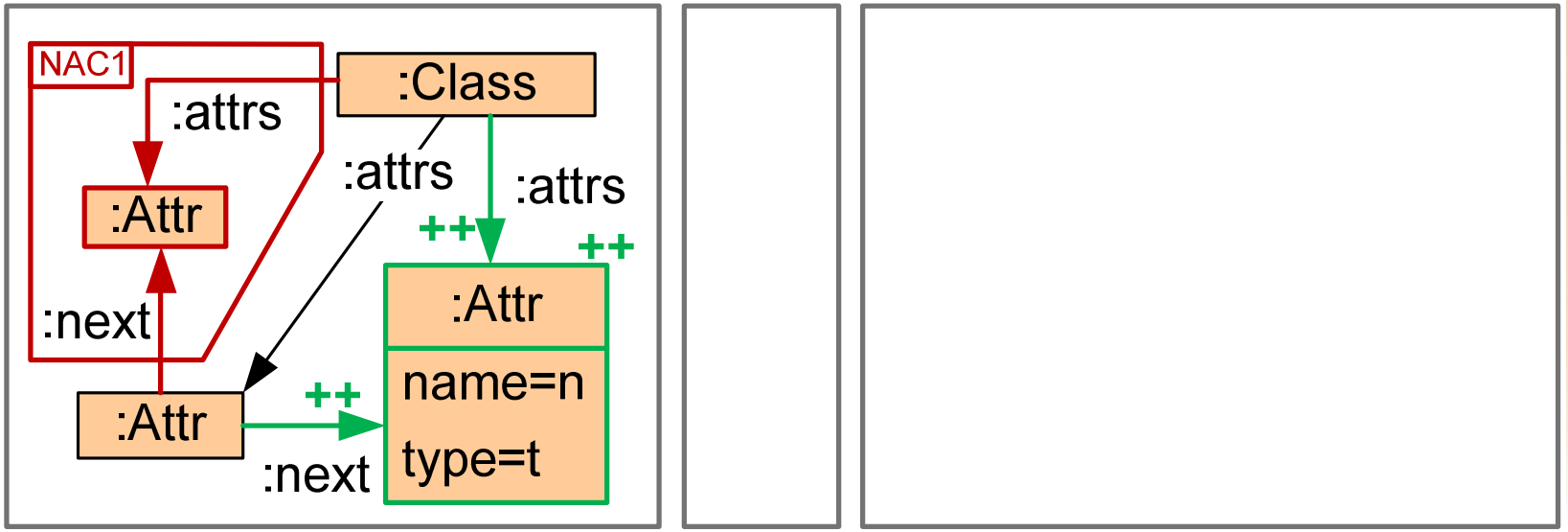
$$\begin{array}{ccccc} L & = & (L^S & \xleftarrow{s_L} & L^C \xrightarrow{t_L} L^T) \\ {\scriptstyle tr}\downarrow & & {\scriptstyle tr^S}\downarrow & & {\scriptstyle tr^C}\downarrow \quad {\scriptstyle tr^T}\downarrow \\ R & = & (R^S & \xleftarrow[s_R]{} & R^C \xrightarrow[t_R]{} R^T) \end{array}$$

triple rule tr
with source and target NACs

$$\begin{array}{ccccc} L_S & = & (L^S & \longleftarrow & \varnothing \longrightarrow \varnothing) \\ & & {\scriptstyle tr^S}\downarrow & & \downarrow \quad \downarrow \\ R_S & = & (R^S & \longleftarrow & \varnothing \longrightarrow \varnothing) \end{array}$$

source rule tr_S
with source NACs

NextAttr2NextColumn$_S$(n:String, t:String)



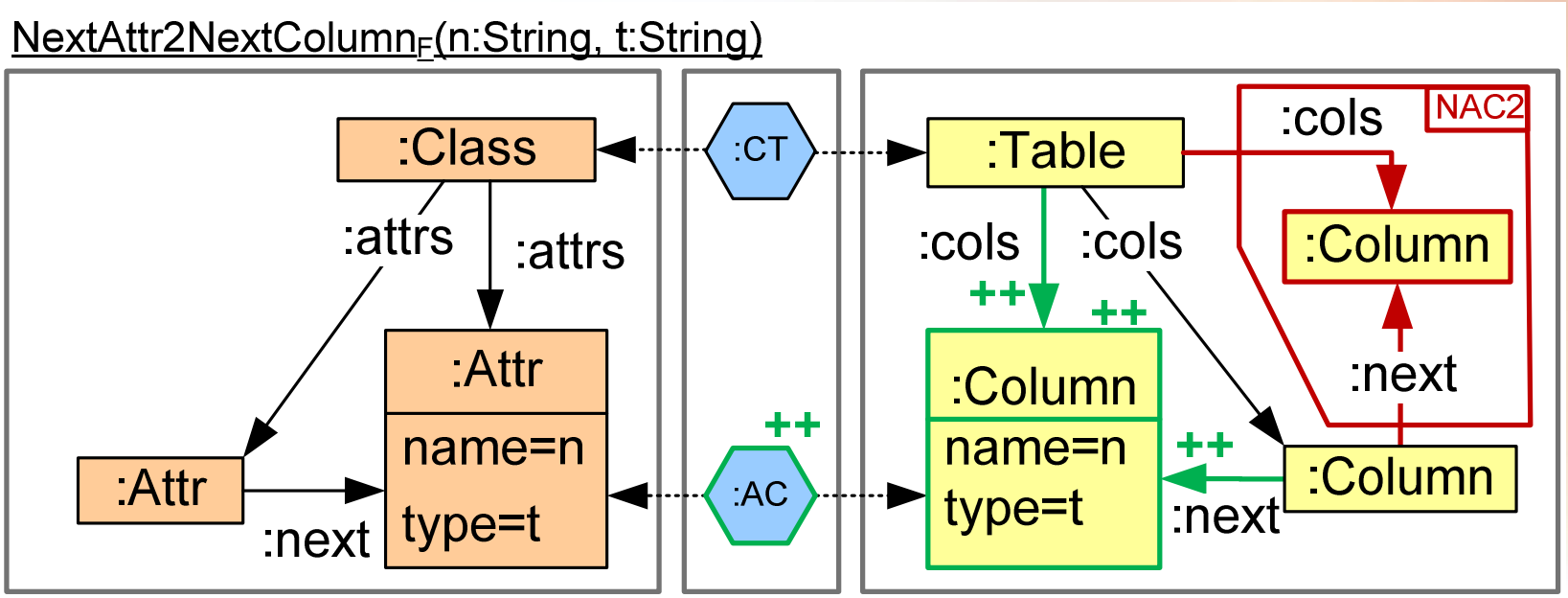
Derived Forward Rule

$$\begin{array}{ccccc} L & = & (L^S & \xleftarrow{s_L} & L^C & \xrightarrow{t_L} & L^T) \\ {\scriptstyle tr}\downarrow & & {\scriptstyle tr^S}\downarrow & & {\scriptstyle tr^C}\downarrow & & {\scriptstyle tr^T}\downarrow \\ R & = & (R^S & \xleftarrow[s_R]{} & R^C & \xrightarrow[t_R]{} & R^T) \end{array}$$

triple rule tr
with source and target NACs

$$\begin{array}{ccccc} L_F & = & (R^S & \xleftarrow{tr^S \circ s_L} & L^C & \xrightarrow{t_L} & L^T) \\ & & {\scriptstyle id}\downarrow & & {\scriptstyle tr^C}\downarrow & & \downarrow{\scriptstyle tr^T} \\ R_F & = & (R^S & \xleftarrow{s_R} & R^C & \xrightarrow{t_R} & R^T) \end{array}$$

forward rule tr_F
with target NACs



Model transformation based on Fw-rules

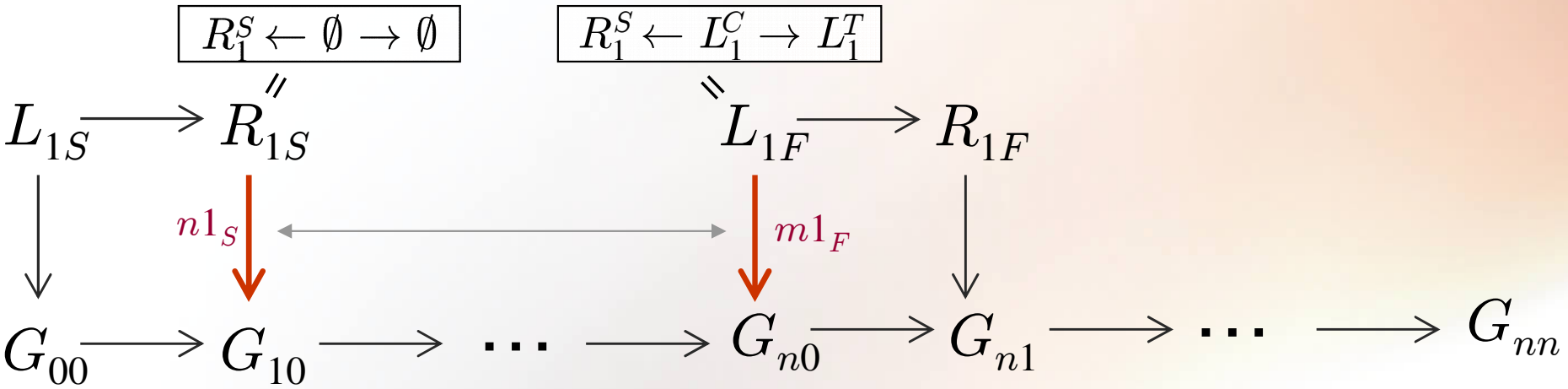
- $MT : VL_S \Rrightarrow VL_T$

  defined by forward rules: $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$
  NAC and source consistent

- Source consistency ensures:
  - Source model can be parsed according to source rules
  - All elements of source model are translated exactly once, thus model transformation stops when everything is translated

Model Transformation Sequence

- Model transformation sequence $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$:
  - $G_S = proj_S(G_0)$, $G_T = proj_T(G_n)$ and $G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is NAC and source consistent
- **Source consistency**: There is a NAC consistent sequence $\emptyset \overset{tr_S^*}{\Longrightarrow} G_0$, s.t. $\emptyset \overset{tr_S^*}{\Longrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_n$ is match consistent
- Match consistency: each comatch ni_S in source sequence determines S-component of match mi_F in forward sequence



Main Results

On-The-Fly Construction

- **Start:** $G_0 = (G_S \leftarrow \varnothing \rightarrow \varnothing)$
- **Running:**

$$\varnothing = G_{00} \overset{tr_S^*}{\Longrightarrow} G_{n-1,0} \overset{g_n}{\hookrightarrow} G_0 \overset{tr_F^*}{\Longrightarrow} G_{n-1}$$

Source sequence

Forward sequence

Match consistency holds for i=0 .. n-1

1. Find : new forward match $m_{n,F}$ for $tr_{n,F}$
2. Derive : source match $m_{n,S}$ for $tr_{n,S}$
3. Check : - NACs of $tr_{n,F}$ and $tr_{n,S}$
   - Effective elements $R_{i,S} \setminus L_{i,S}$ are matched for the first time: (1) is PB
4. Extend : source and forward sequences (breath first search or depth first search)

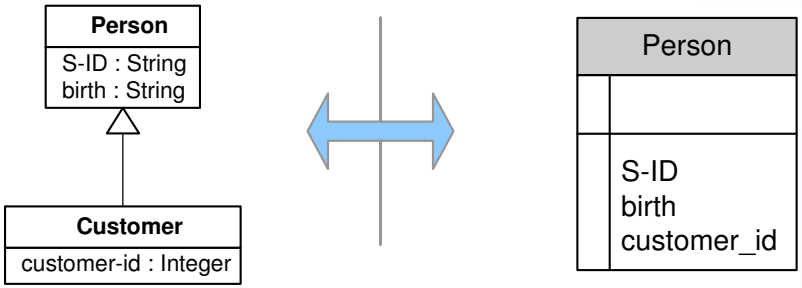
$$\begin{array}{ccccc} L_{n,S} & \hookrightarrow & R_{n,S} & \hookrightarrow & L_{n,F} \\ {\scriptstyle m_{n,S}}\downarrow & & (1) & & \downarrow{\scriptstyle m_{n,F}} \\ G_{n-1,0} & \underset{g_{n-1}}{\hookrightarrow} & G_0 & \hookrightarrow & G_{n-1} \end{array}$$

- **Finish:** if $g_n = id$, i.e. G_0 is constructed by the source sequence, result: G_n

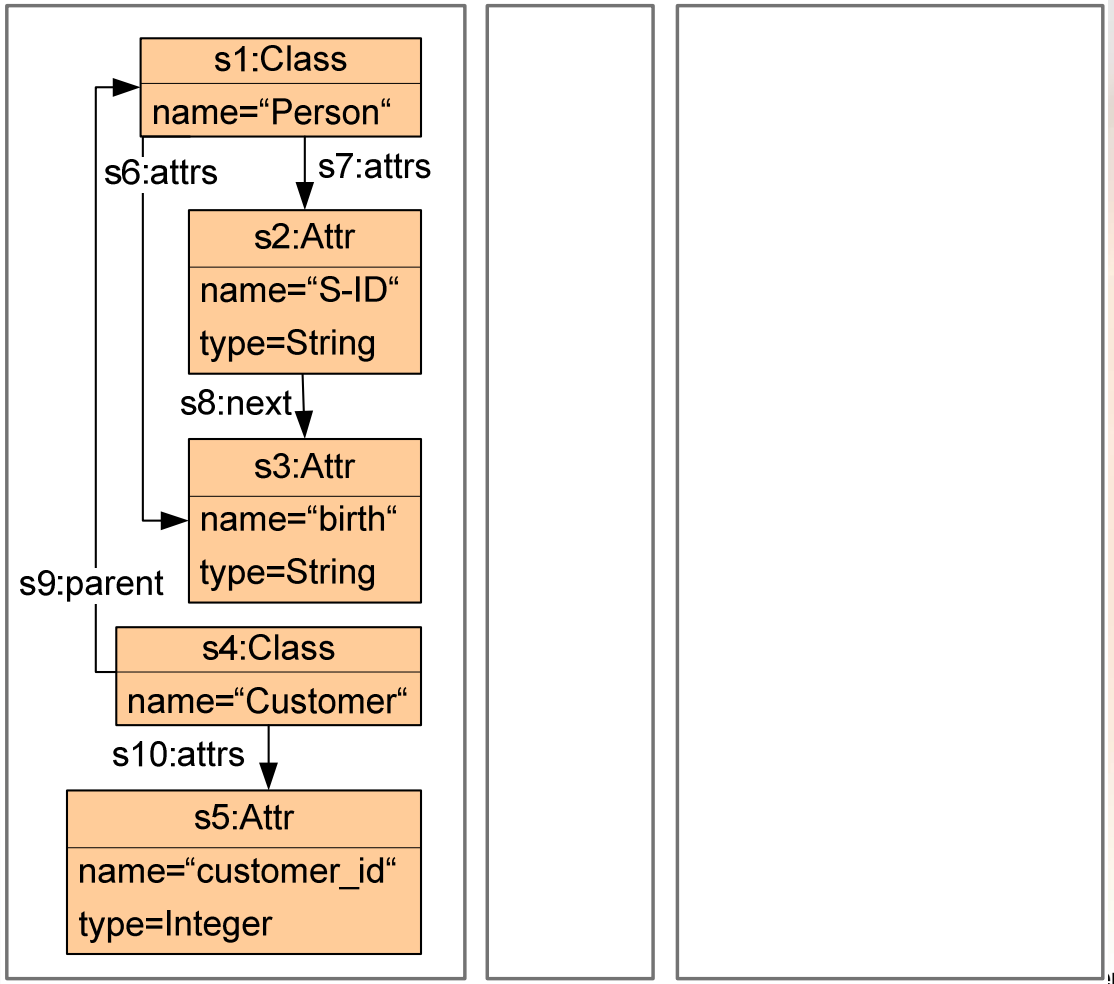
Main Results: On-The-Fly Construction

- Triple language $VL = \{G \mid \emptyset \Rightarrow^* G \; via \; TR\}$
- **Termination:** guaranteed, if all derived source rules are creating: $R_{i,S} \setminus L_{i,S} \neq \emptyset$
- **Correctness:** For each constructed MT-sequence: $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ there is a $G \in VL$, s.t. $G = (G_S \leftarrow G_C \rightarrow G_T)$
- **Completeness:** For each $G_S \in VL^S$ there is an MT-seuqence: $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$

Example: Model Trafo

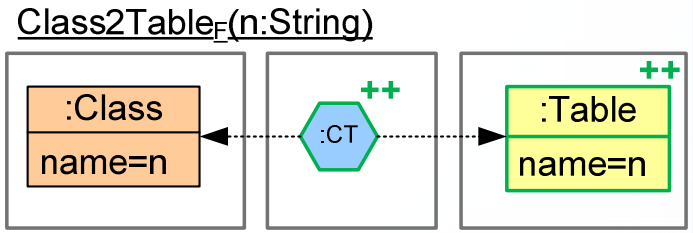


Instance in concrete syntax

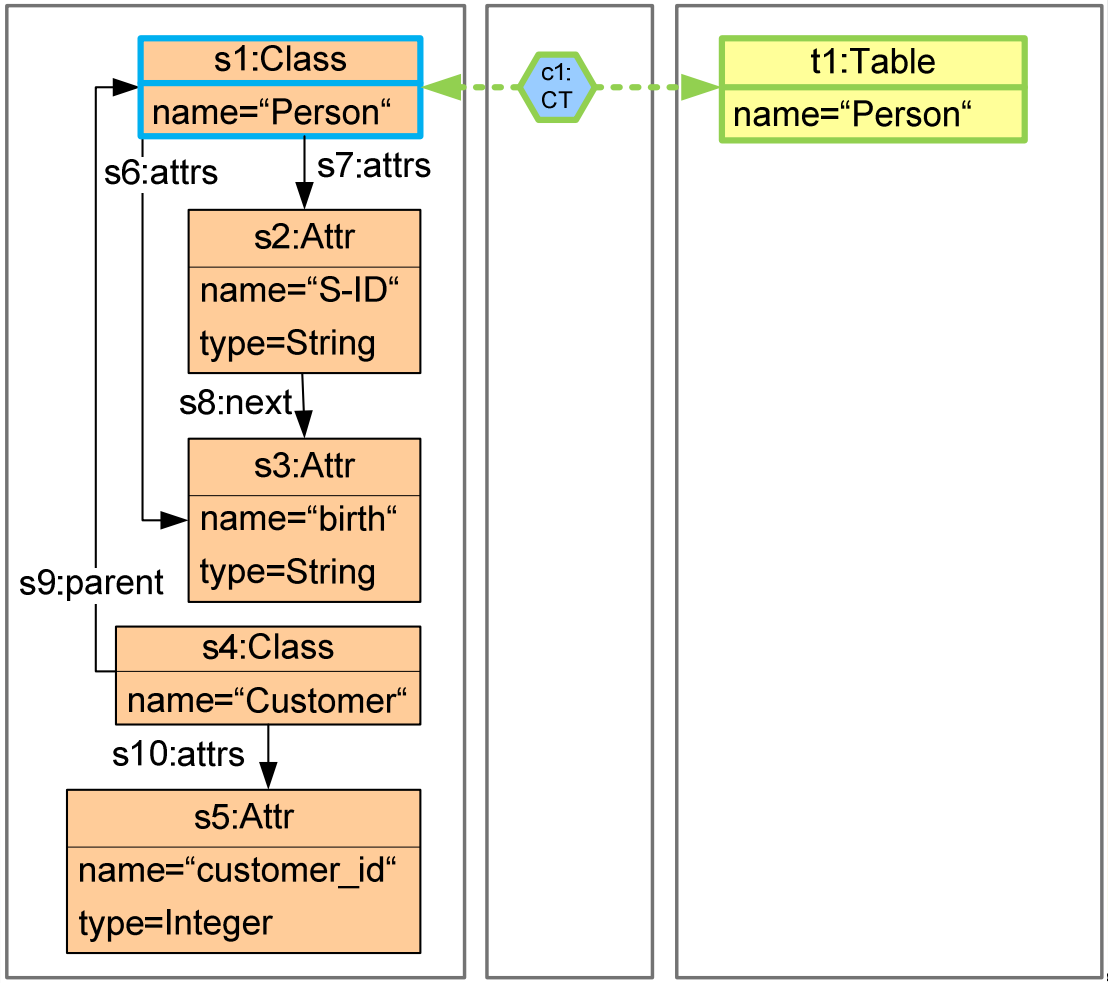


Triple Graph (abstract syntax)

Example: Model Trafo

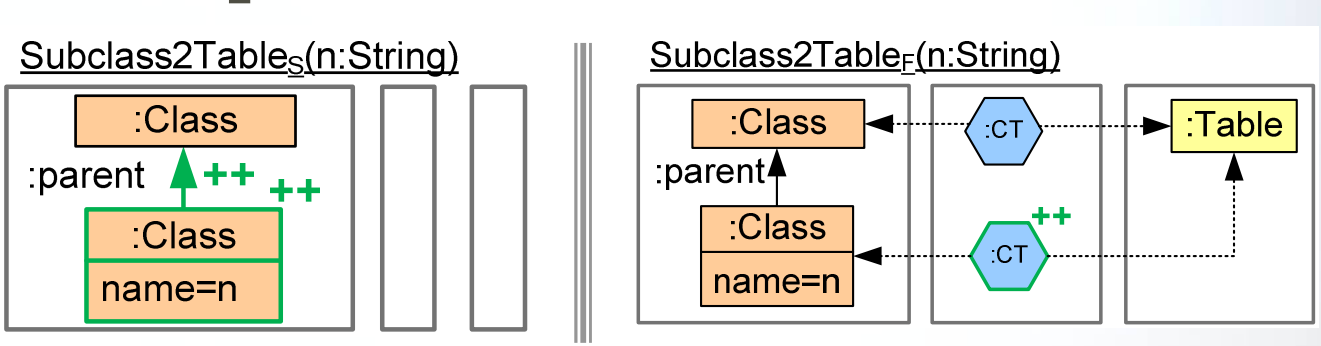


FW-Rule
(abstract syntax)



Triple Graph
(abstract syntax)

Example: Model Trafo



Source rule and FW-Rule (abstract syntax)

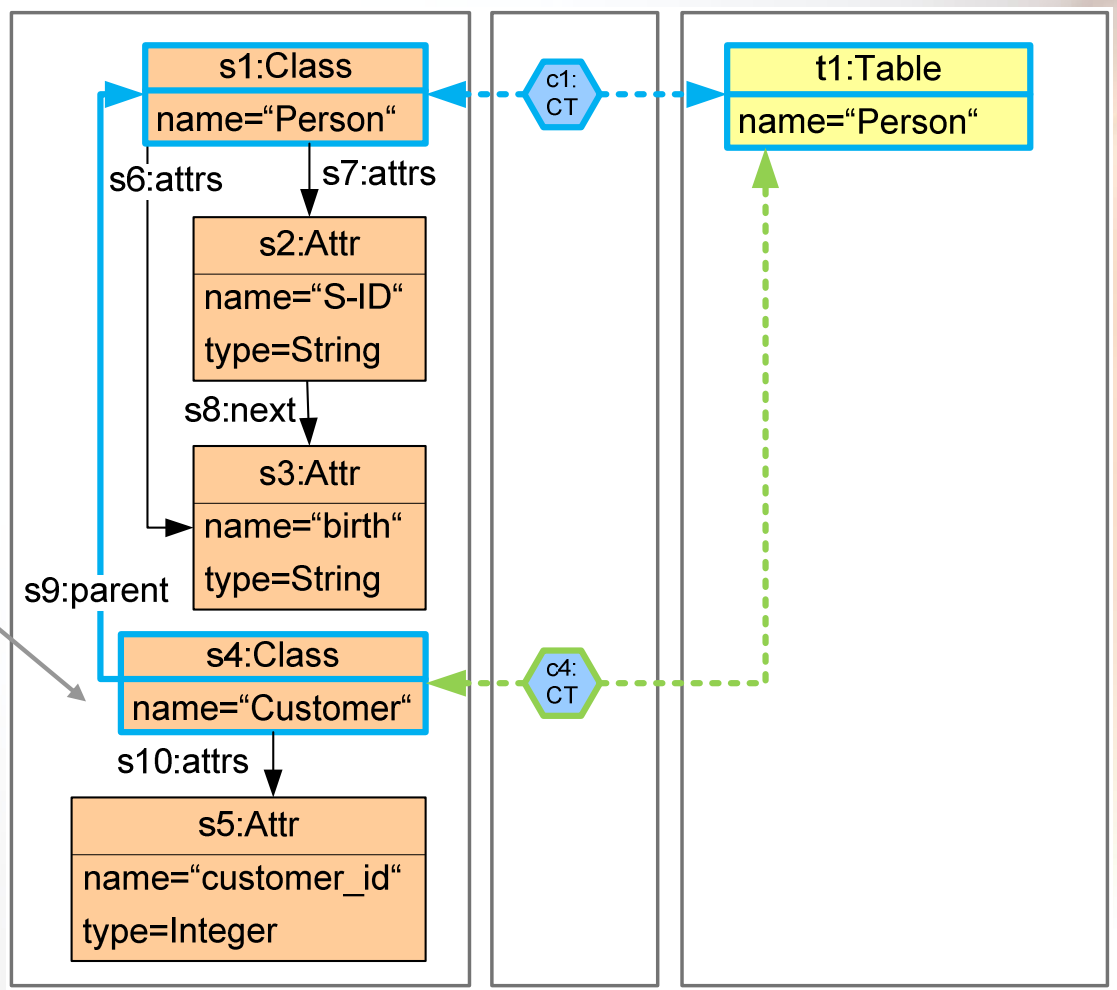
On-the-fly Construction

1. Forward match
2. Source match
3. Check:

a) NACs

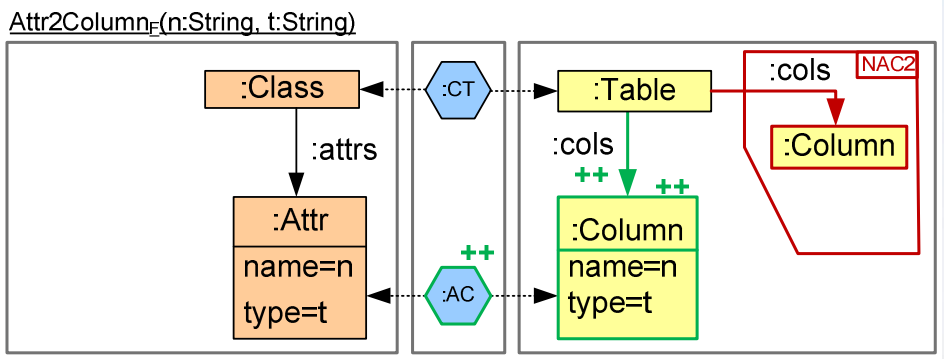
b) Effective Elements not matched before

4. Extend

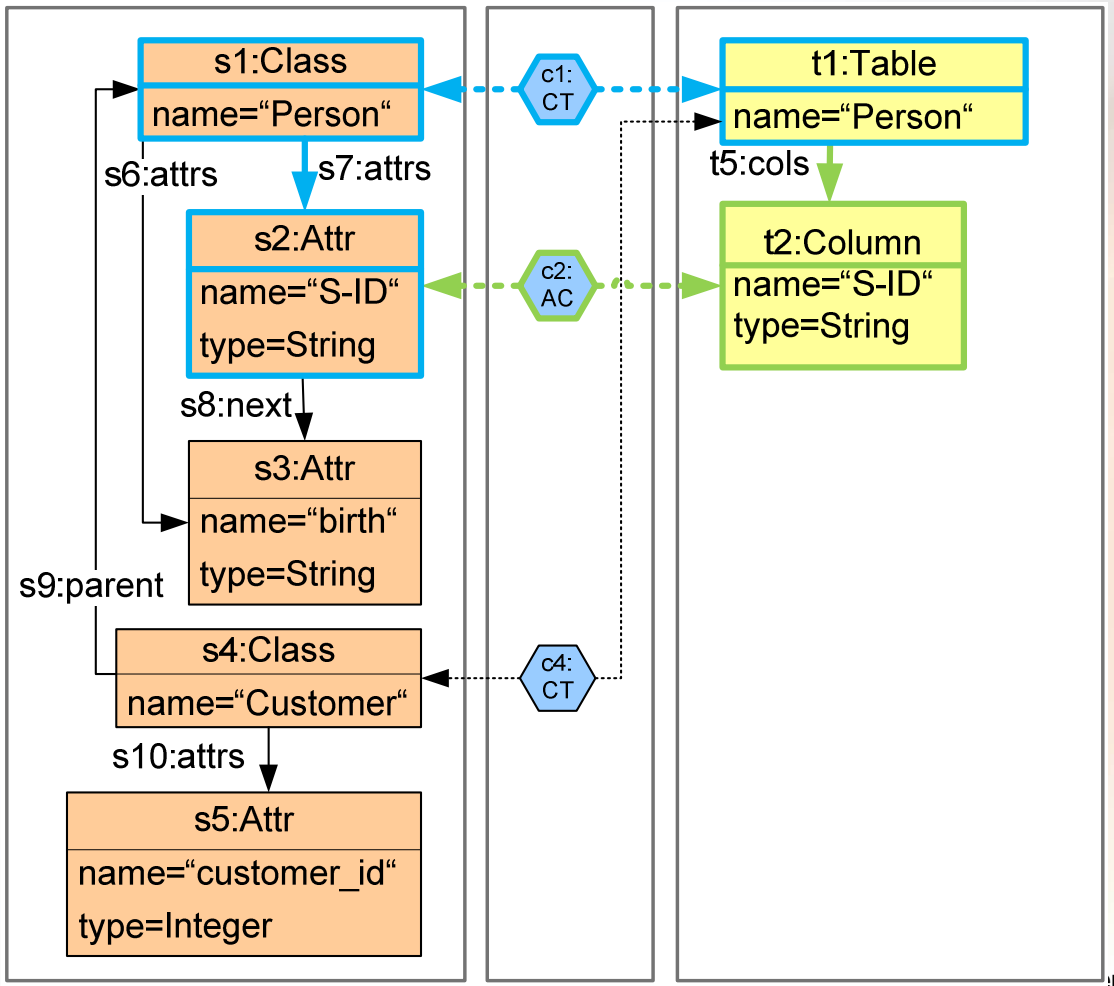


Triple Graph (abstract syntax)

Example: Model Trafo

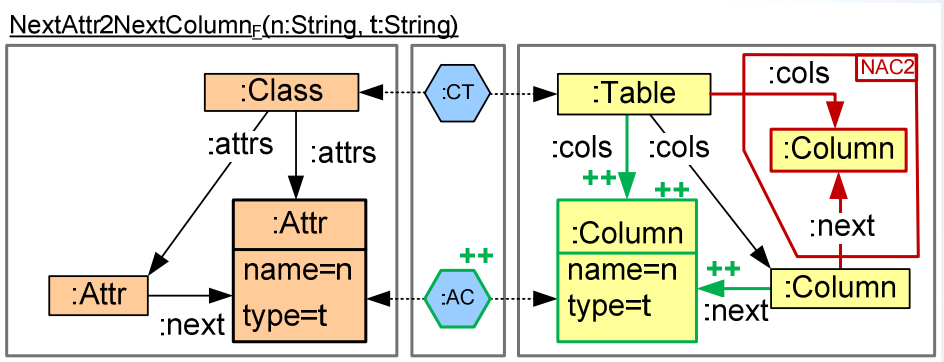


FW-Rule
(abstract syntax)

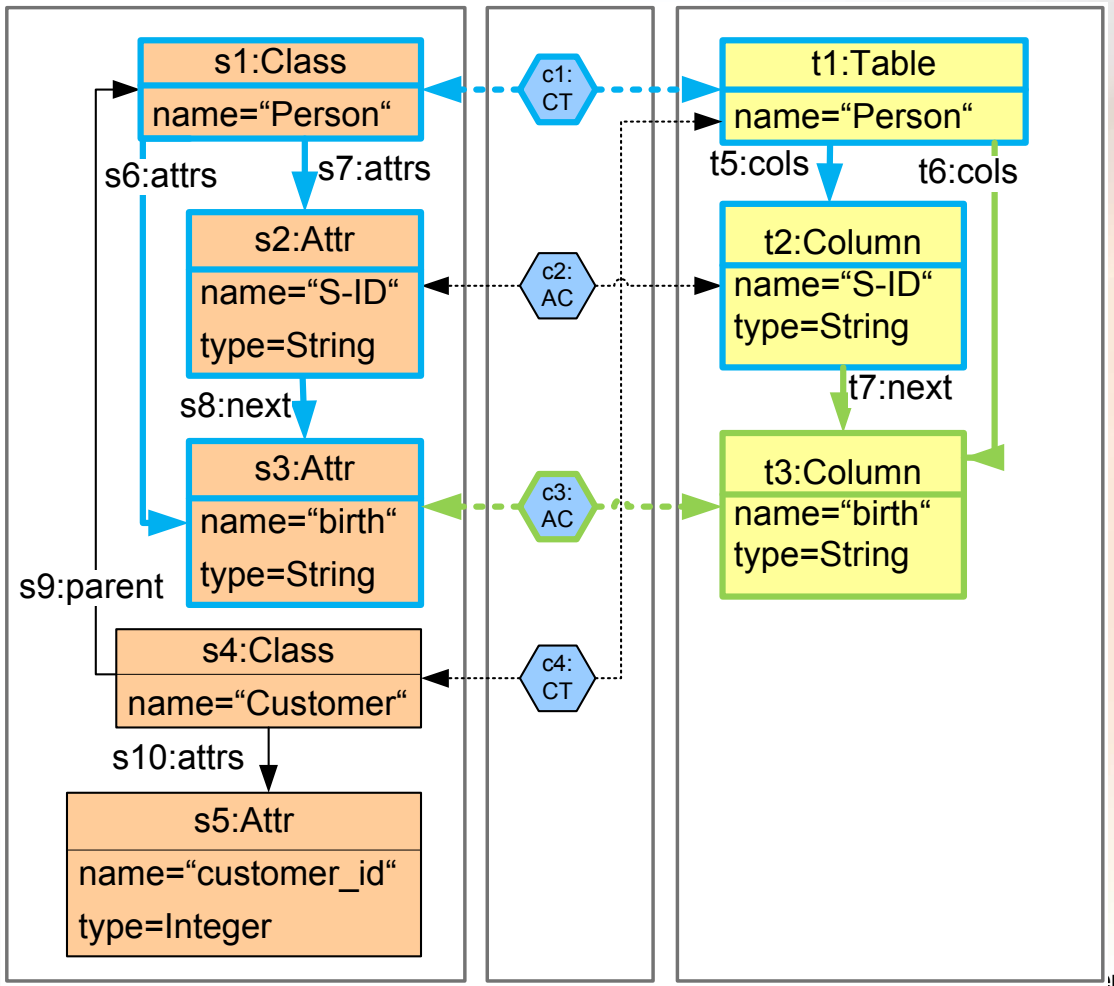


Triple Graph
(abstract syntax)

Example: Model Trafo

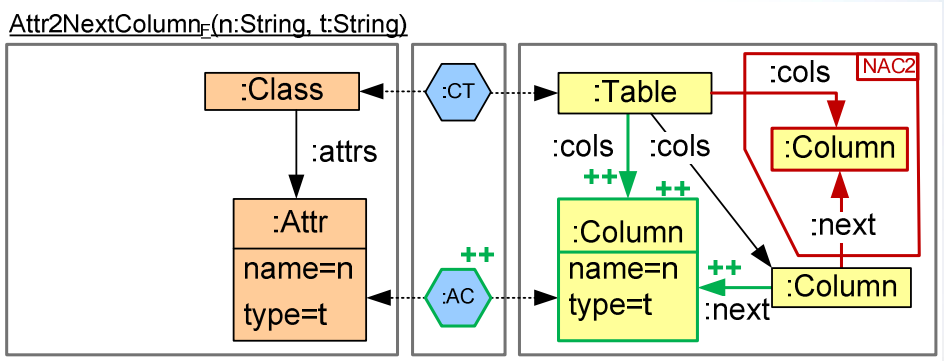


FW-Rule
(abstract syntax)

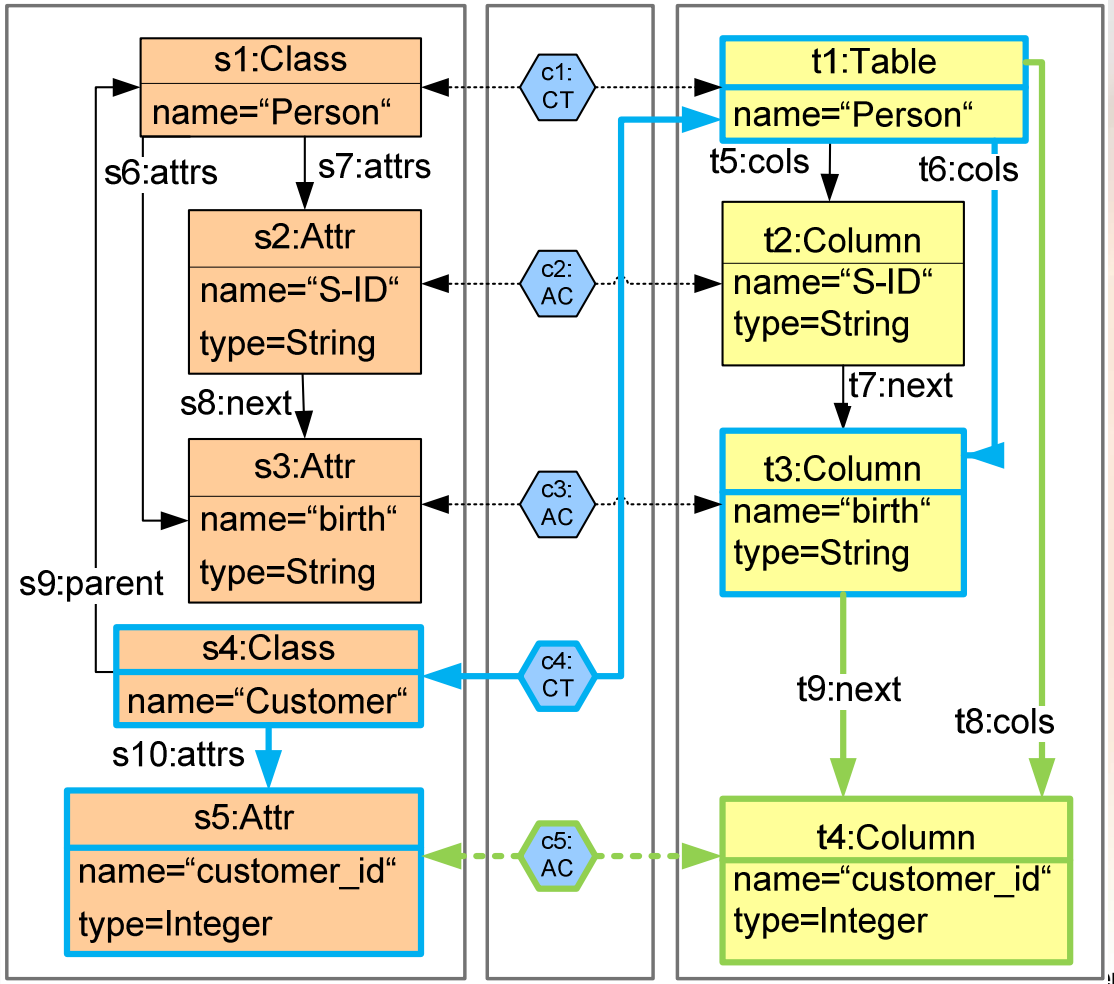


Triple Graph
(abstract syntax)

Example: Model Trafo

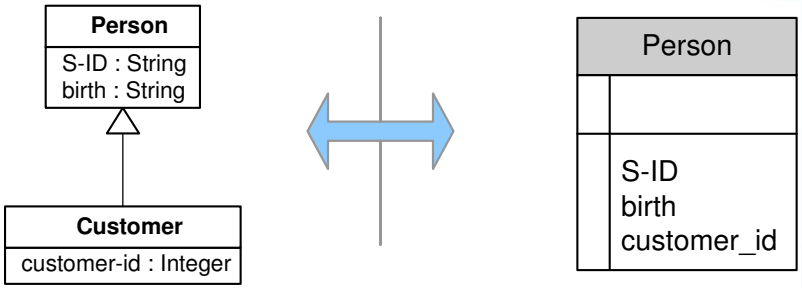


FW-Rule
(abstract syntax)

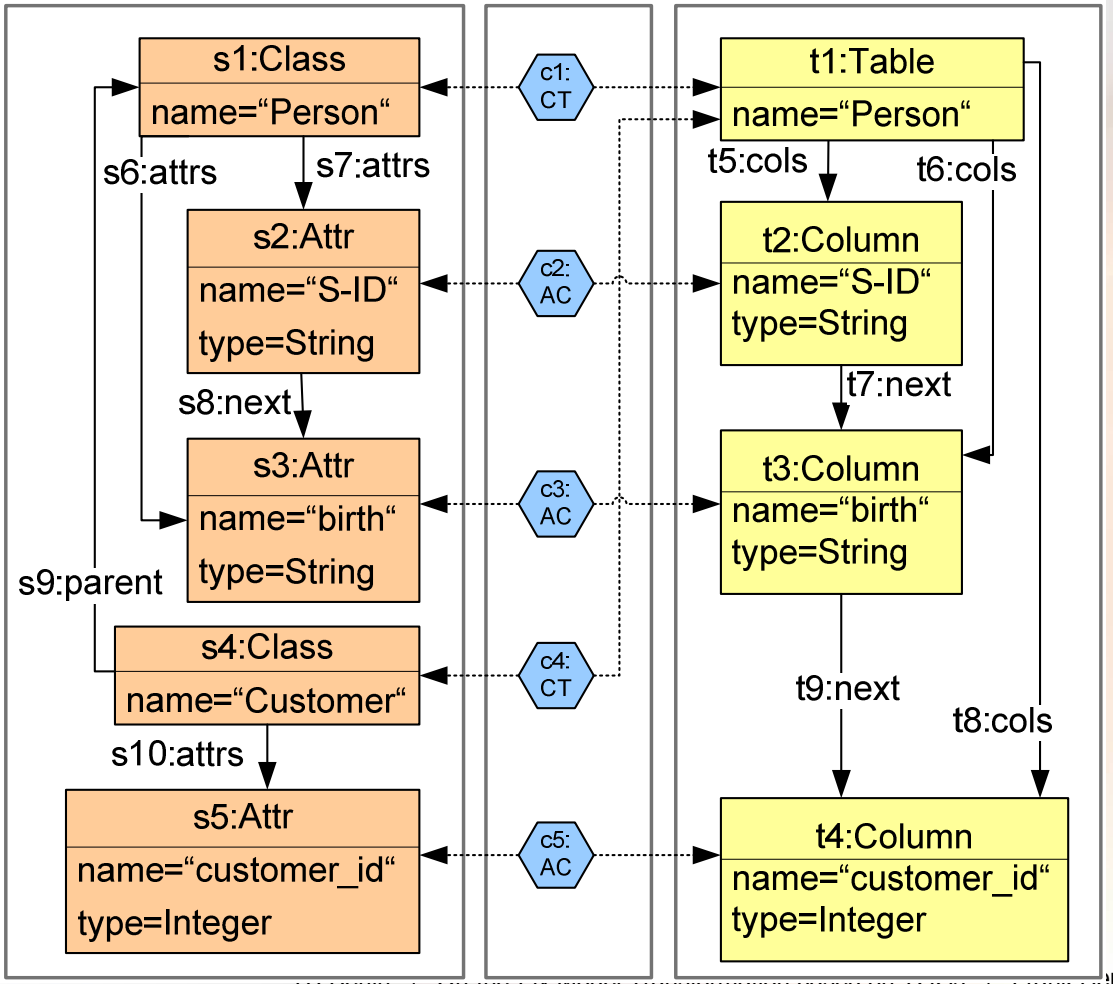


Triple Graph
(abstract syntax)

Example: Model Trafo



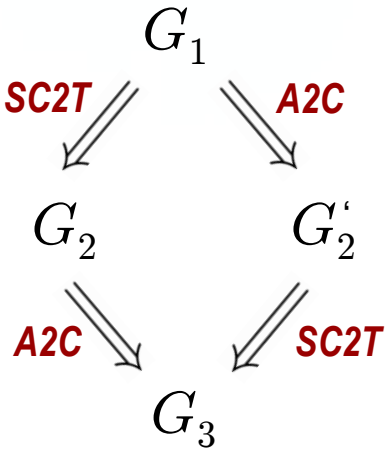
Instance in concrete syntax

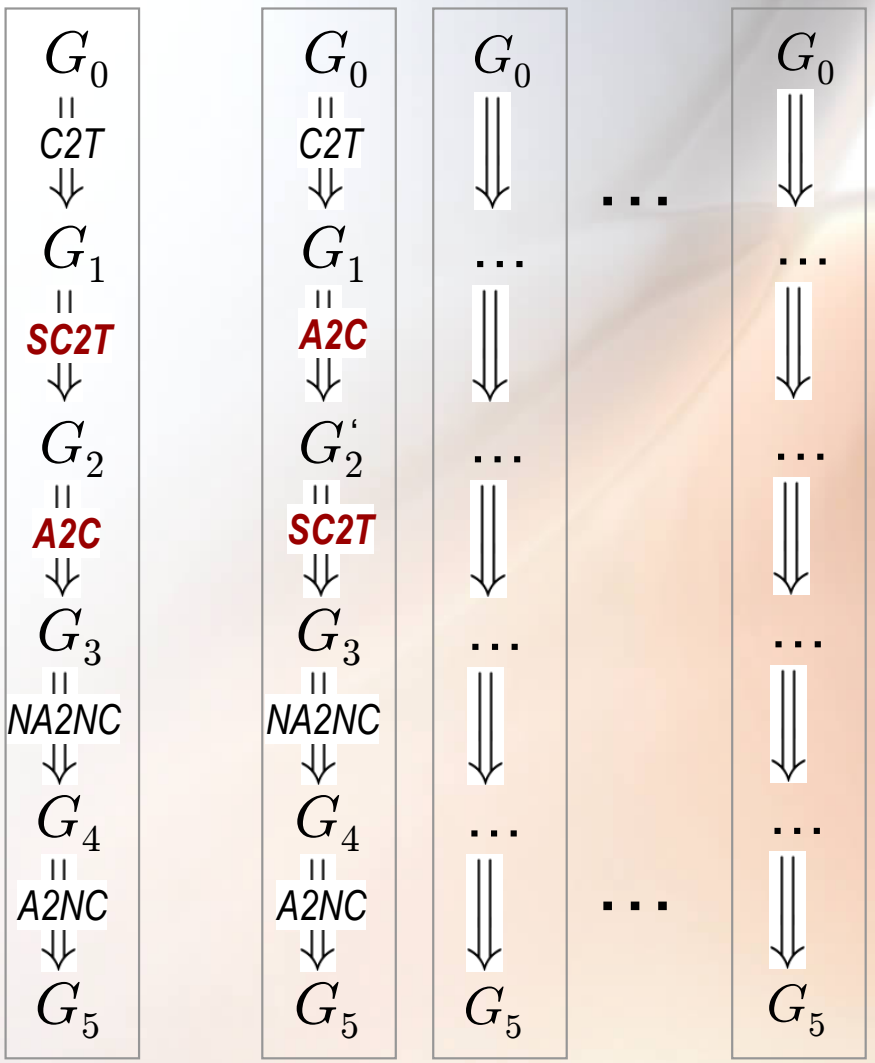


Triple Graph (abstract syntax)

Partial Order Reduction

- Based on parallel independence and switch equivalence





- One sequence is sufficient
- Double Diamond Condition

Main Results

- **Termination:** guaranteed, if all derived source rules are creating: $R_{i,S} \setminus L_{i,S} \neq \emptyset$
- **Correctness:** For each constructed MT-sequence: $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$ there is a $G \in VL$, s.t. $G = (G_S \leftarrow G_C \rightarrow G_T)$
- **Completeness:** For each $G_S \in VL^S$ there is an MT-seuqence: $(G_S, G_0 \overset{tr_F^*}{\Longrightarrow} G_n, G_T)$
- **Parallel Independence and Partial Order Reduction**

Conclusion

- **Important** new results for TGGs with **NACs**
  - Termination, Correctness and Completeness of MTs
- Efficiency of standard construction improved by:
  - On-the-fly construction
  - Partial order reduction

Future work:

- Implementation: based on Mathematica and Eclipse
- Benchmarks: effect of partial order reduction and the on-the-fly construction

References

[EEE+07]	Ehrig, H. and Ehrig, K. and Ermel, C. and Hermann, F. and Taentzer, G.: **Information Preserving Bidirectional Model Transformations**. *Proc. FASE'07*. LNCS, vol. 4422, pp. 72-86. Springer (2007).
[EEH08]	Ehrig, H. and Ermel, C. and Hermann, F.: **On the Relationship of Model Transformations Based on Triple and Plain Graph Grammars**. *Proc. GraMoT'08*. ACM (2008).
[EHS09]	Ehrig, H. and Hermann, F. and Sartorius, C.: **Completeness and Correctness of Model Transformations based on Triple Graph Grammars with Negative Application Conditions**. *Proc. GT-VMT'09*. EC-EASST (2009).
[SK08]	Schürr, A., Klar, F.: **15 Years of Triple Graph Grammars**. *Proc. of ICGT'08*. LNCS, vol. 5214, pp. 411-425. Springer (2008).
[Schürr94]	Schürr, A.: **Specication of Graph Translators with Triple Graph Grammars**. *Proc. of WG 1994*. LNCS, vol. 903, pp. 151–163. Springer (1995).