Visual Modeling and Model Transformation

Hans-Jörg Kreowski

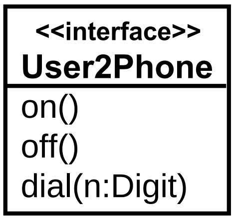
- hope to solve the software problem
  (as software engineering promises for nearly 40 years)
- but not without rigorous semantics and verification
  (as software engineering tends to ignore)

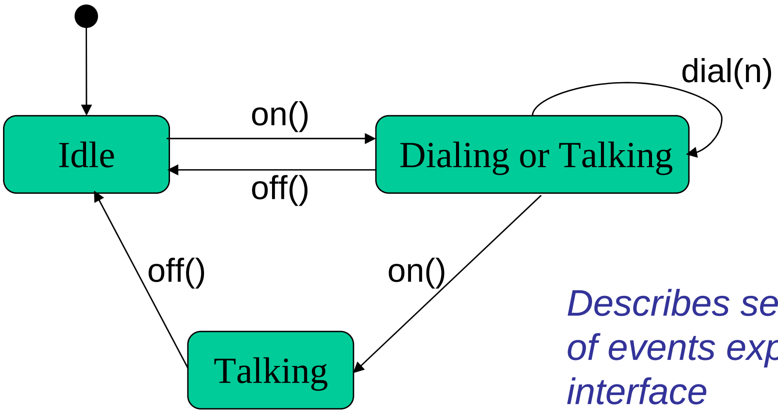
University of Bremen

Visual modeling

- UML (use cases, class and object diagrams, state charts, sequence and collaboration diagrams, etc.)
- many other types of diagrams (ER diagrams, flow diagrams, etc.)
- Petri nets
- graph grammars (as counterpart to Chomsky grammars and contextfree grammars in particular)

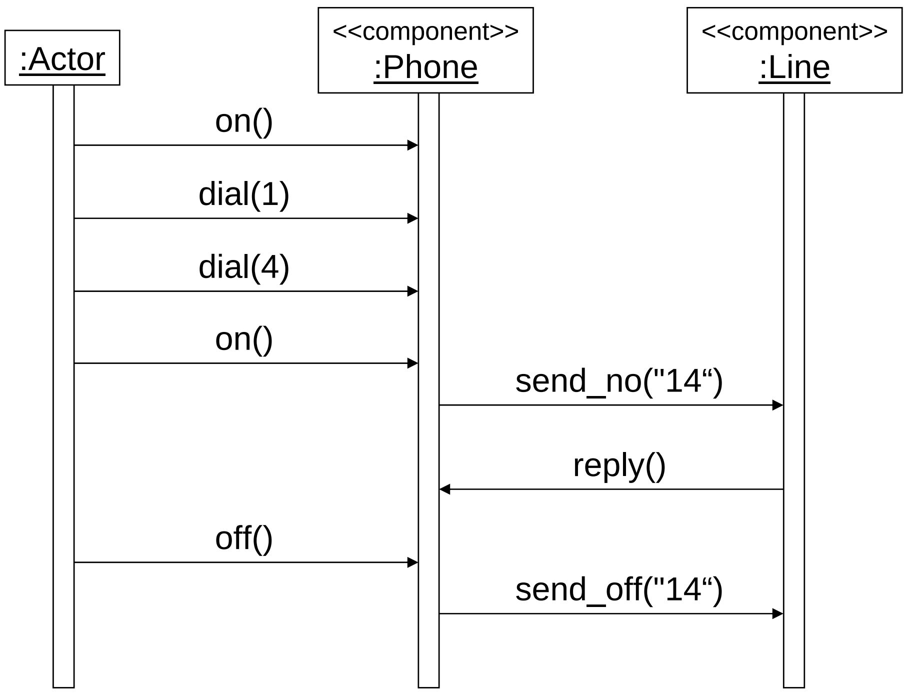
Protocol Statechart
Interface *User2Phone*

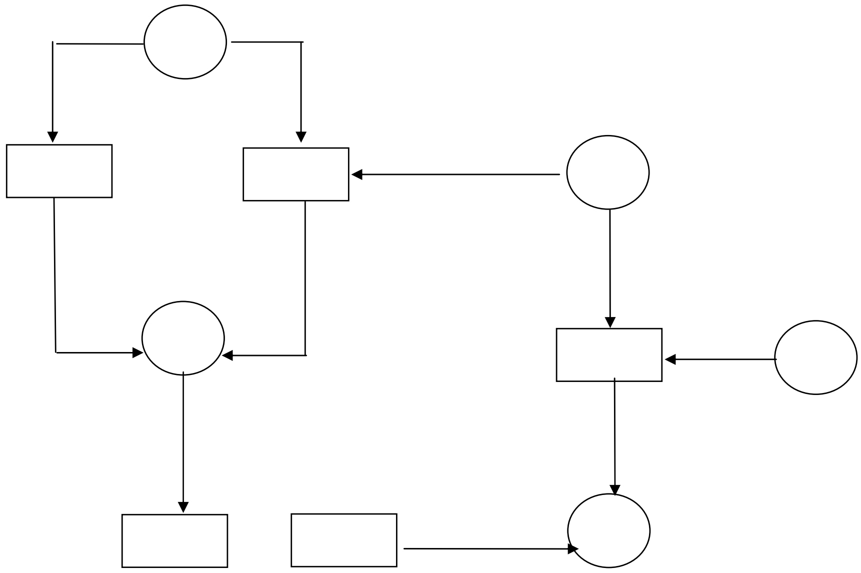


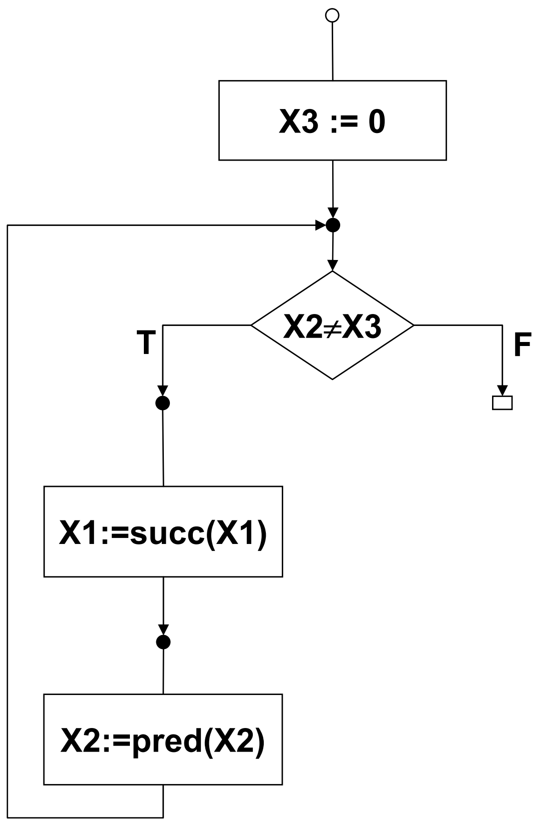


Describes sequences of events expected by interface

Sequence Diagram
Calling Scenario

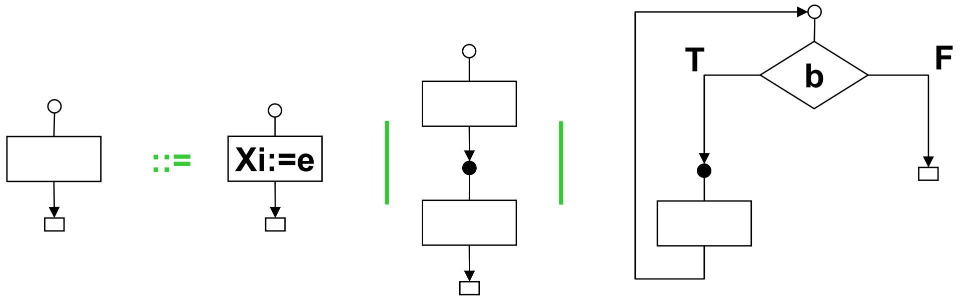






X1:=X1+X2

rules:

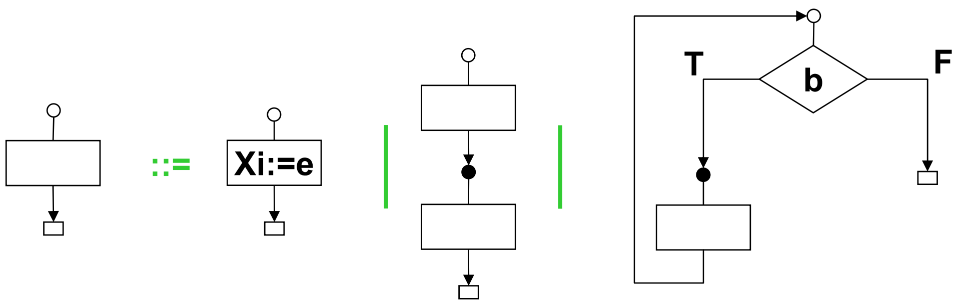


e ::= 0 | succ(Xj) | pred(Xj)

b ::= Xi≠Xj

1≤ i, j ≤ k

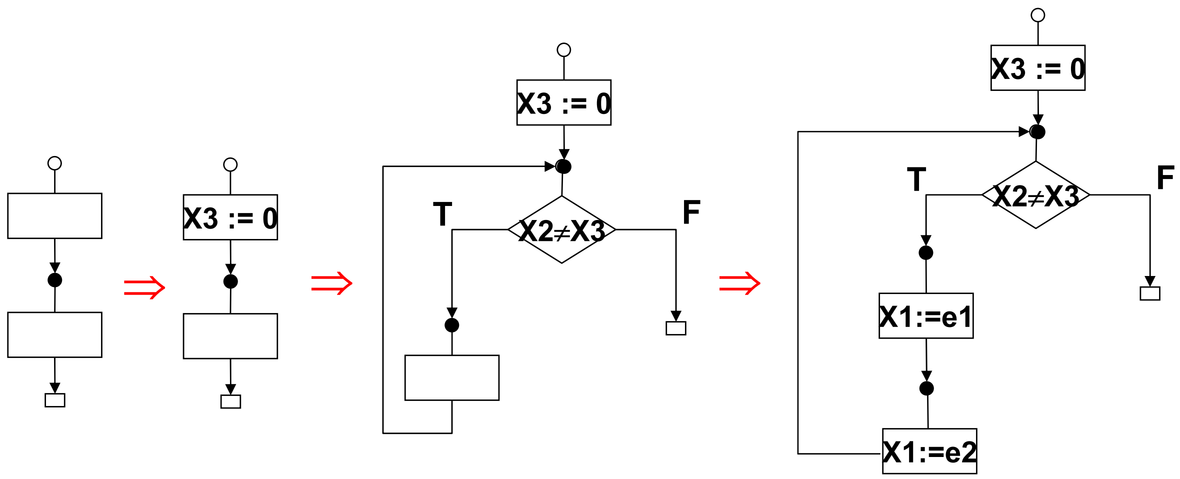
rules:



e ::= 0 | succ(Xj) | pred(Xj)

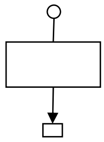
b ::= Xi≠Xj

1≤ i, j ≤ k

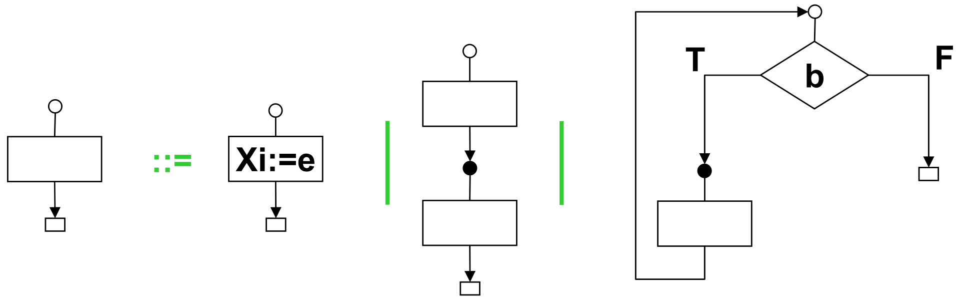


generation of well-structured flow diagrams

initial:



rules:

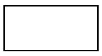


e ::= 0 | succ(Xj) | pred(Xj)

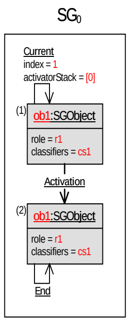
b ::= Xi≠Xj

1≤ i, j ≤ k

terminal: ***no***



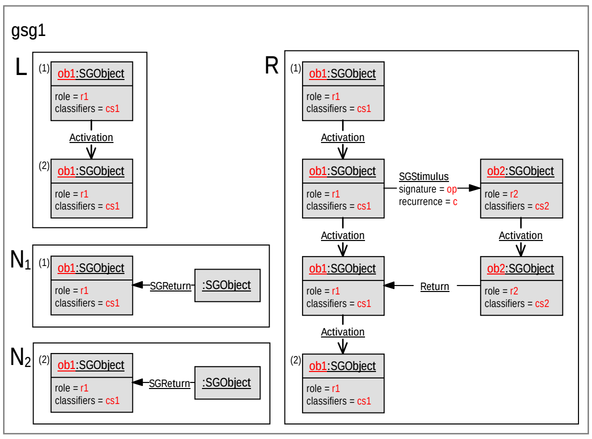
Generation of Sequence Graphs

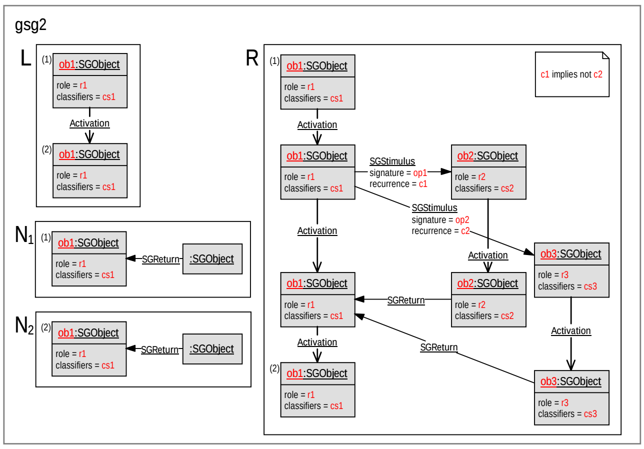


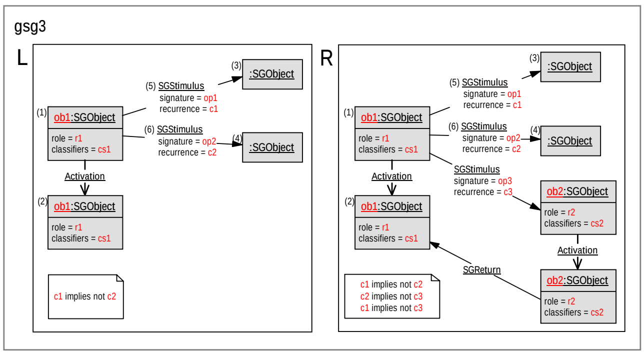
GenerateSG

init: SG_0

rules: gsg1, gsg2, gsg3



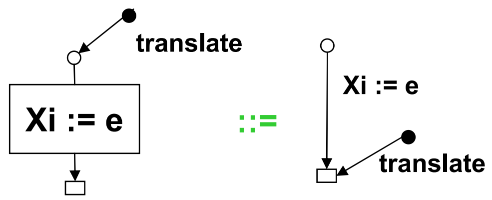


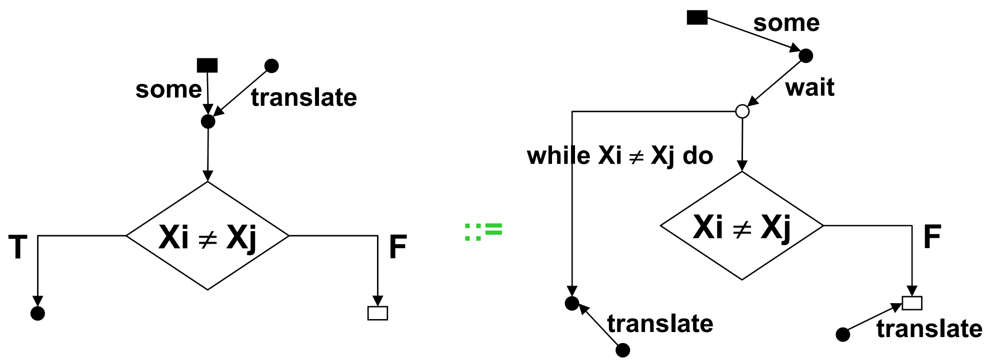


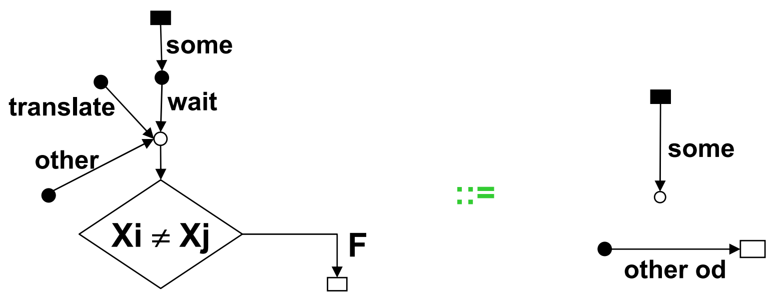
Model transformation

- MDA (model-driven architecture, successor of UML)
- compiler (from source to target programs)
- software development process and program transformation
- graph transformation

translation of well-structured flow diagrams into textual expressions



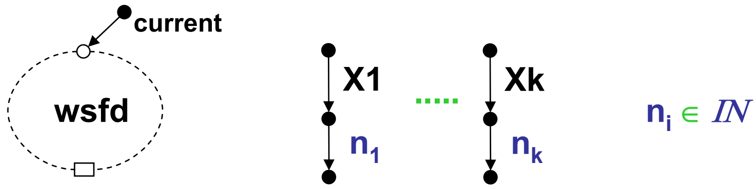




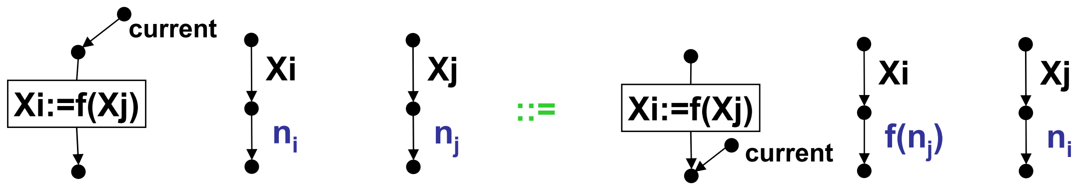
plus some string composition rules

evaluation of well-structured flow diagrams

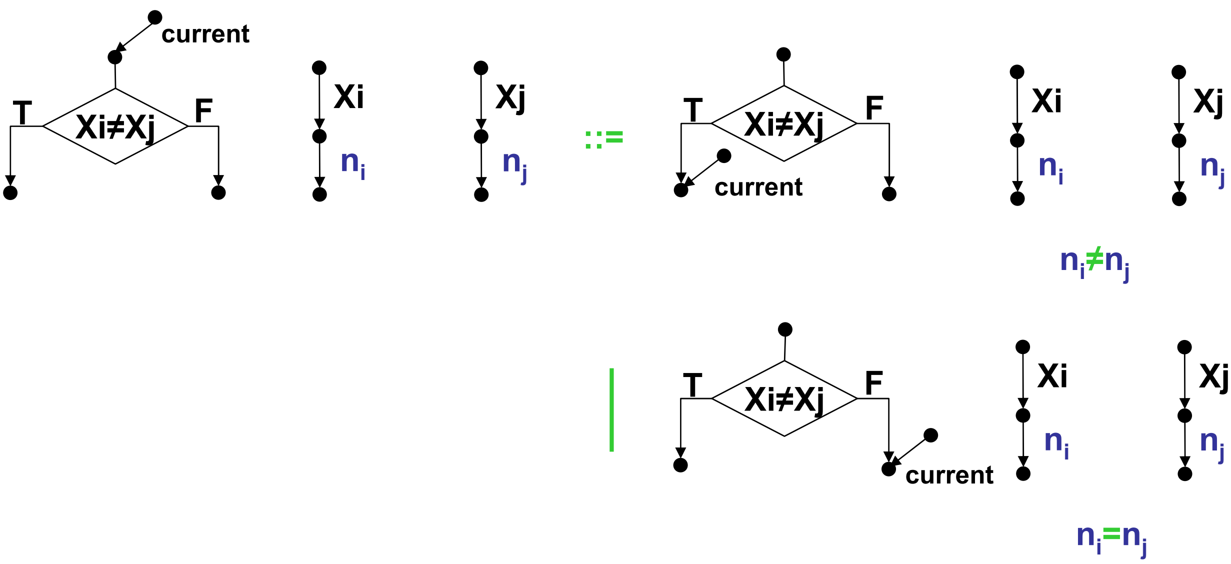
initial:



rules:



$f \in \{succ, pred\}$



terminal: *reduced forms*

Rule base

- basic ingredients of a rule-based setting

- set of configurations $\mathcal{K}$ (e.g. graphs)
- set of rules $\mathcal{R}$ (e.g. graph transformation rules)
- rule application operator $\Rightarrow$ assigning
  $\underset{r}{\Rightarrow} \subseteq \mathcal{K} \times \mathcal{K}$ to each $r \in \mathcal{R}$

- $con \underset{r}{\Rightarrow} con'$ direct derivation, computation step, evaluation step, reduction step, etc.

- $\underset{P}{\Rightarrow}$ union of all $\underset{r}{\Rightarrow}$ for all $r \in P$
- $\overset{*}{\underset{P}{\Rightarrow}}$ reflexive and transitive closure

Rule-based system (1st version)

- syntax $P \subseteq \mathcal{R}$

- operational semantics

  iterated rule application relation $\overset{*}{\underset{P}{\Rightarrow}} \subseteq \mathcal{K} \times \mathcal{K}$

  rule application graph $Graph(P) = (\mathcal{K}, \underset{P}{\Rightarrow})$

examples

- derivation graph (in grammatical context)
- reachability graph (of place/transition nets)
- transition relation

Rule base with control condition

- control the nondeterminism of rule applications
- set of control conditions $\mathcal{C}$
  with $SEM(c) \subseteq \mathcal{K} \times \mathcal{K}$ for $c \in \mathcal{C}$

examples

- priorities, regular expressions, evaluation strategies, etc
- (I, T) with $SEM((I, T)) = \mathcal{K}(I) \times \mathcal{K}(T)$
  ($\mathcal{K}(I) \subseteq \mathcal{K}$ initial config's and $\mathcal{K}(T) \subseteq \mathcal{K}$ terminal ones,
  may be used together with other control conditions)
- (S, all) with $\mathcal{K}(S) = \{S\}$ for $S \in \mathcal{K}$ and $\mathcal{K}(all) = \mathcal{K}$

Rule-based system (2nd version)

- syntax (P, c) with $P \subseteq \mathcal{R}$ and $c \in \mathcal{C}$
- operational **(?!)** semantics
  iterated rule application relation $\overset{*}{\underset{P}{\Rightarrow}} \cap SEM(c)$
  rule application graph
  $Graph(P, (I, T)) = (\mathcal{K}, \underset{P}{\Rightarrow}, \mathcal{K}(I), \mathcal{K}(T))$

examples

- various kinds of grammars, place/transition systems,
  term rewrite systems, graph transformation,
  finite state machines and statecharts with OR states, etc

Model transformation

- $(P, (I, T), c)$ specifies input-output transformation

- model transformation if $\mathcal{K}(I)$ and $\mathcal{K}(T)$) are sets of (visual) models of some kind (cf. compiler semantics)

examples

(Cordes & Hoelscher 2003)

Semantics by model transformation

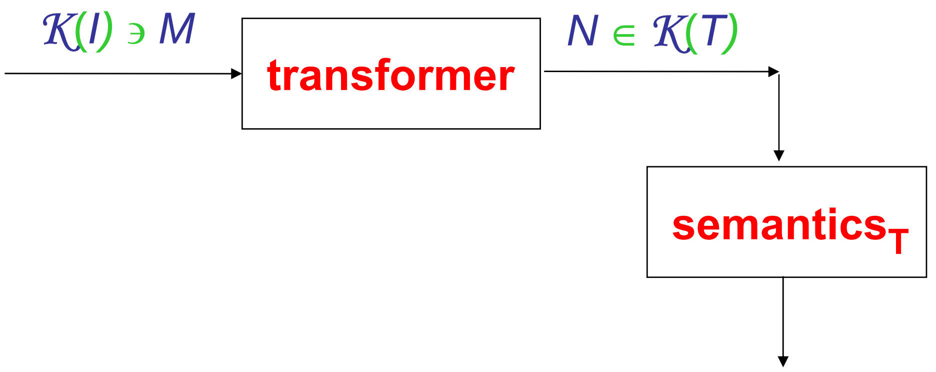
- $(P, (I, T), c)$ specifies input-output transformation

- semantics of models in $\mathcal{K}(I)$ *if* $\mathcal{K}(T)$) is set of semantic models of some kind

example

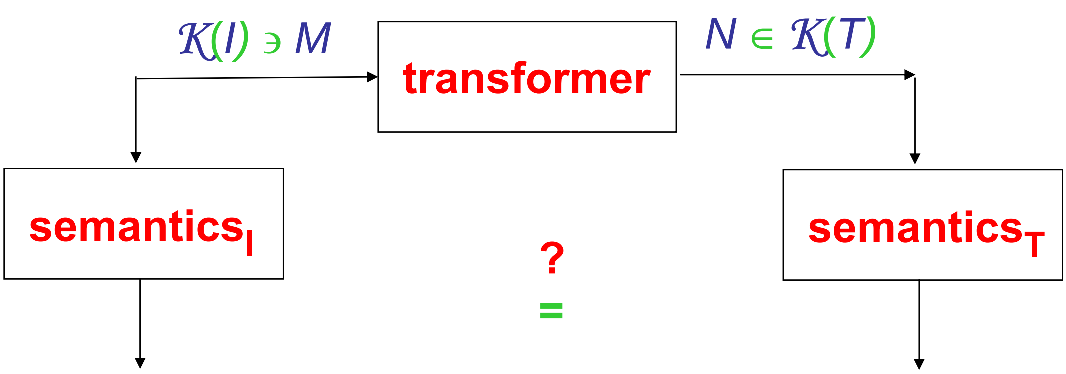
Nice aspect of model transformation (1)

- source models borrow semantics from target models
  (if they have some)



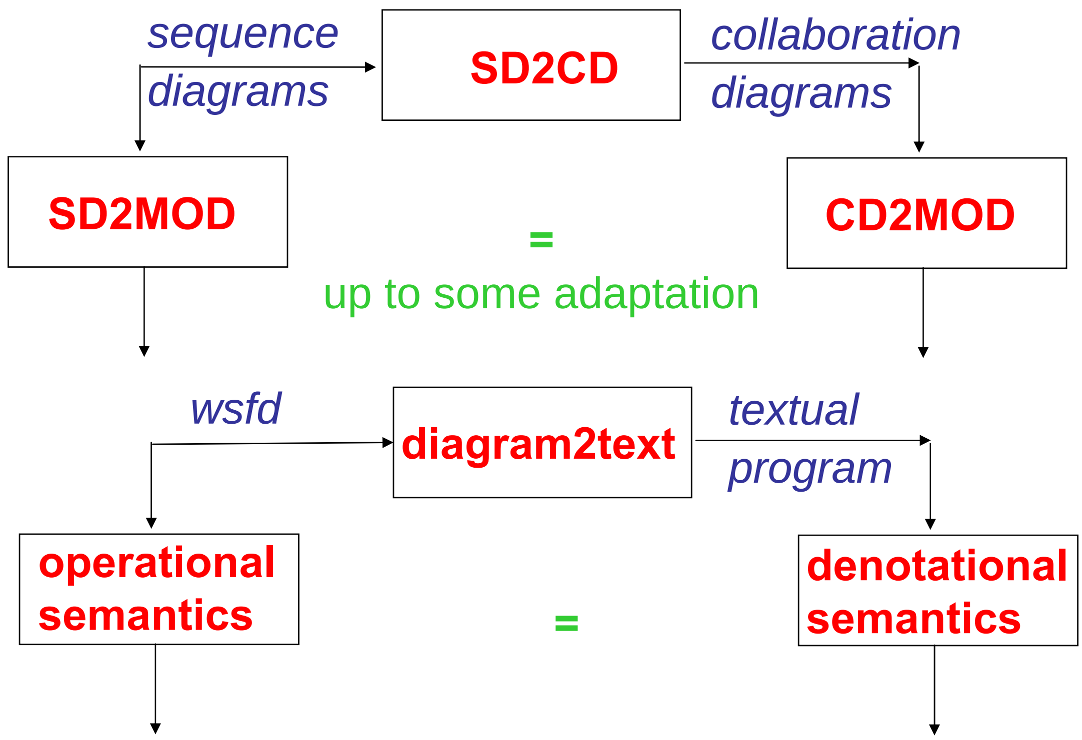
Nice aspect of model transformation (2)

- notion of correctness
  (if source and target models have got semantics)



Nice aspect of model transformation (2)

examples



Denotational semantics

- mapping of (visual) models into semantic domain
- like model transformation if semantic entities are graphs

example

wsfd´s $\xrightarrow{S}$ **state transition functions** $\mathbb{N}^k \longrightarrow \mathbb{N}^k$

defined by

$S(Xi{:=}0 \mid f(Xj))(n_1,\ldots,n_k) = (n_1,\ldots,n_{i-1},0 \mid f(n_j),n_{i+1},\ldots,n_k)$

$S(wfds1;wfds2)(n_1,\ldots,n_k) = S(wfds2)(S(wfds1)(n_1,\ldots,n_k))$

$S(\text{while } Xi \neq Xj \text{ do } wfds \text{ od})(n_1,\ldots,n_k) = \text{if } n_i \neq n_j \text{ then } S(\text{while } Xi \neq Xj \text{ do } wfds \text{ od})(S(wfds)(n_1,\ldots,n_k)) \text{ else } (n_1,\ldots,n_k)$

Conclusion

- rule-based framework suitable for operational semantics and model transformation in visual modeling
- see, e.g., Baresi, Ehrig, Engels, Gogolla, Heckel, Minas, Schürr, Taentzer for more details
- interesting aspects are missing like structuring, composition and non-sequentiality
- future research will shed some more light on the semantic foundation of visual modeling

- Handbook on Graph Grammars and Computing by Graph Transformation