Definition von domänenspezifischen Modellierungssprachen

Jens Kosiol

24. und 26. Juni 2024

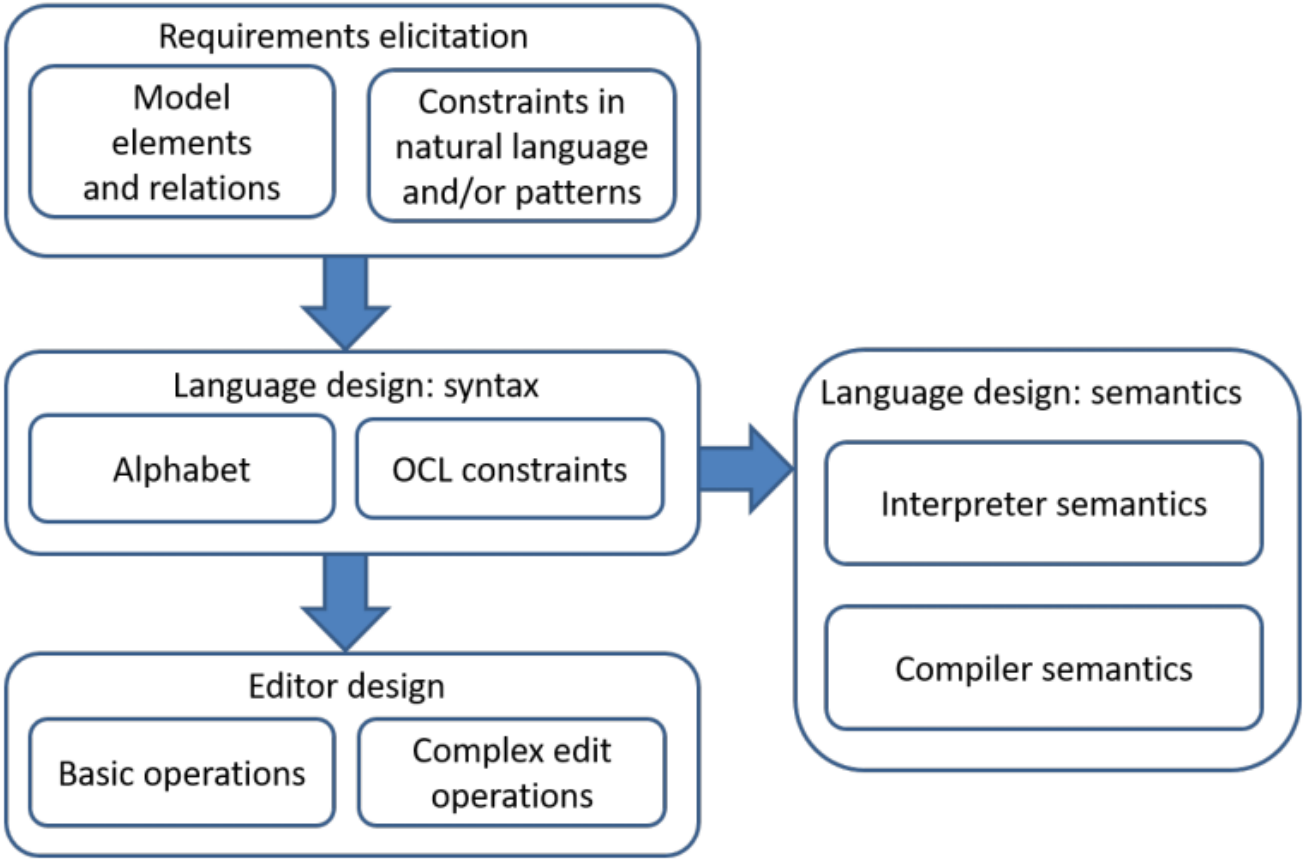
Überblick

- Domänenspezifische Modellierungssprachen (DSMLs) helfen, Probleme in einer Anwendungsdomäne leichter zu verstehen und Lösungen zu entwickeln.
  - Z.B. als Testmodell in modellbasiertem Testen
  - Z.B. als Designmodell für modellbasierte Softwareentwicklung
- Wie kann die Definition von DSMLs systematisiert und automatisiert werden?
  - *Wie werden DSMLs üblicherweise definiert?*
  - *Wie kann Graphtransformation helfen?*

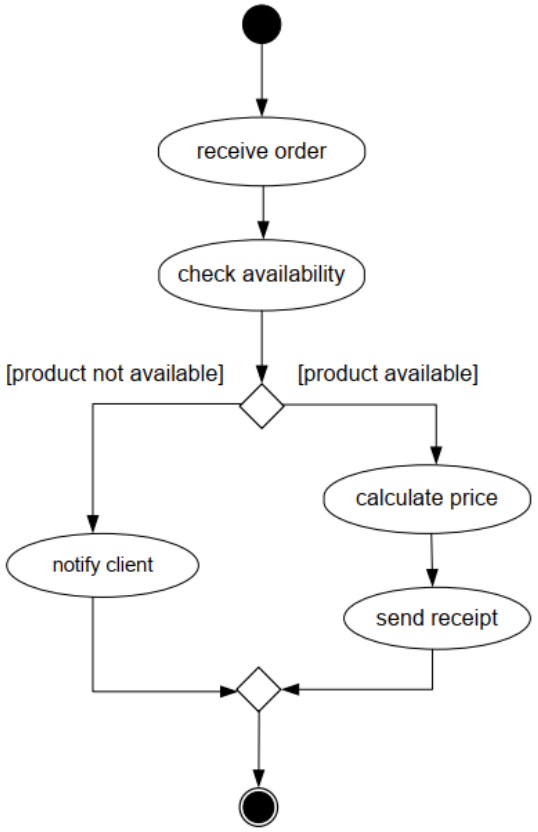
Definition von DSMLs

- Die Definition einer DSML kann als spezielles Softwareprojekt aufgefasst werden.
- Übertragung des Softwareentwicklungsprozesses:
  - *Anforderungsbeschreibung für die Sprache*
  - *Sprach- und Werkzeugdesign*
  - *Werkzeuggenerierung (Implementierung)*
- Unterstützende Techniken:
  - *Metamodellierung (Meta Object Facilities (MOF))*
  - *Grammatiken, speziell Graphgrammatiken*

Hauptaufgaben des Sprach-Designs



Beispiel: Einfache Aktivitätsdiagramme



Modellelemente:

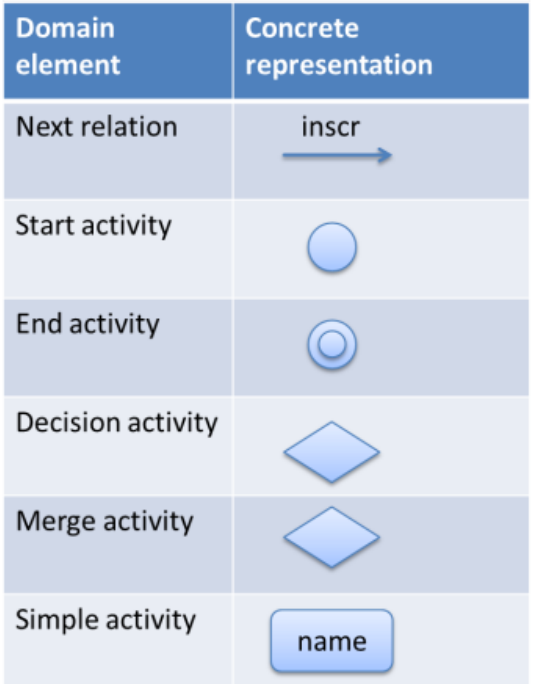
- Verschiedene Typen von Aktivitäten:
  - *Startaktivitäten*
  - *einfache Aktivitäten*
  - *Entscheidungsaktivitäten*
  - *Endaktivitäten*
- Pfeile beschreiben Kontrollfluß
  - *können Bedingungen als Annotationen haben*

Bedingungen (Constraints):

- genau eine Start- und eine Endaktivität
- wohldefinierte Entscheidungs-strukturen

Definition von DSMLs: Konkrete Syntax

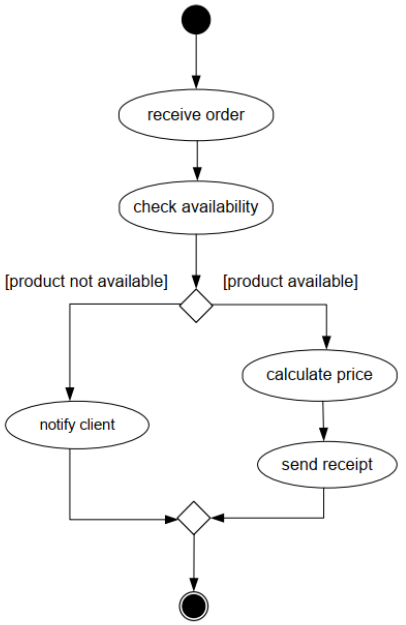
- Alphabetdefinition: *Wie sieht das domänenspezifische Alphabet aus?*
  - *Welche Sprachelemente?*
  - *Welche Attribute haben diese?*
  - *Welche Relationen zwischen diesen?*
- Sprachdefinition: *Welche Wörter/Modelle sind erlaubt?*
  - *nur Wörter/Modelle über dem Alphabet*
  - *Welche zusätzlichen Eigenschaften müssen erfüllt sein?*



Sprache der Aktivitätsdiagramme

Spracheigenschaften:

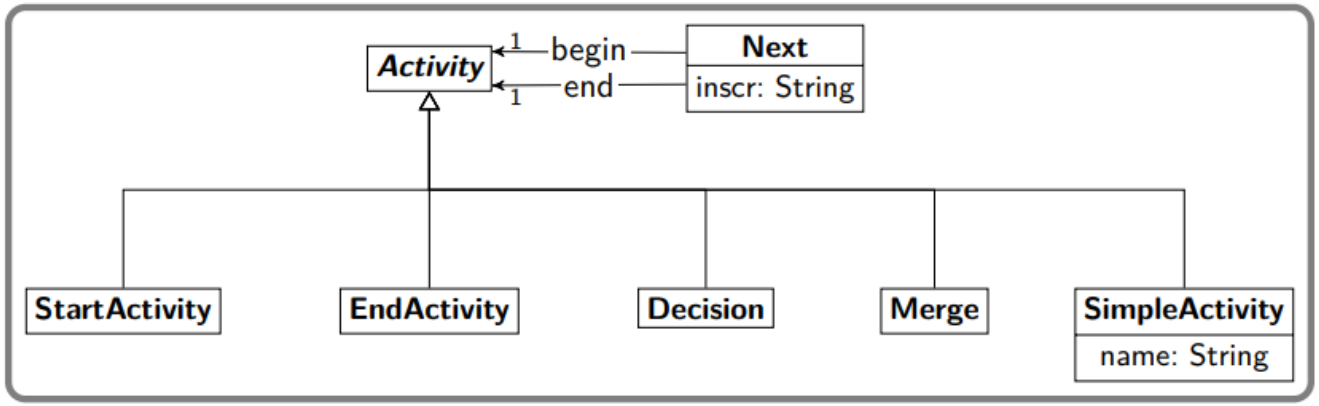
- Es gibt genau eine Start- und eine Endaktivität.
- Jede einfache Aktivität hat genau eine eingehende und eine ausgehende Transition.
- Die Startaktivität hat keine eingehende, die Endaktivität keine ausgehende Transition.
- Jede Entscheidung hat zwei Alternativen, die später wieder vereinigt werden.
- Nur die ausgehenden Transitionen einer Entscheidung haben Beschriftungen.



- Entscheidungen haben eine eingehende und zwei ausgehende, oder zwei eingehende und eine ausgehende Transition.

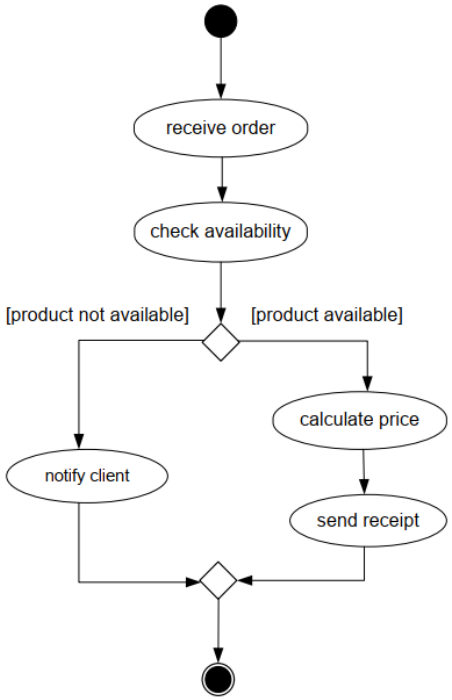
Abstrakte Syntaxdefinition von einfachen Aktivitätsdiagrammen

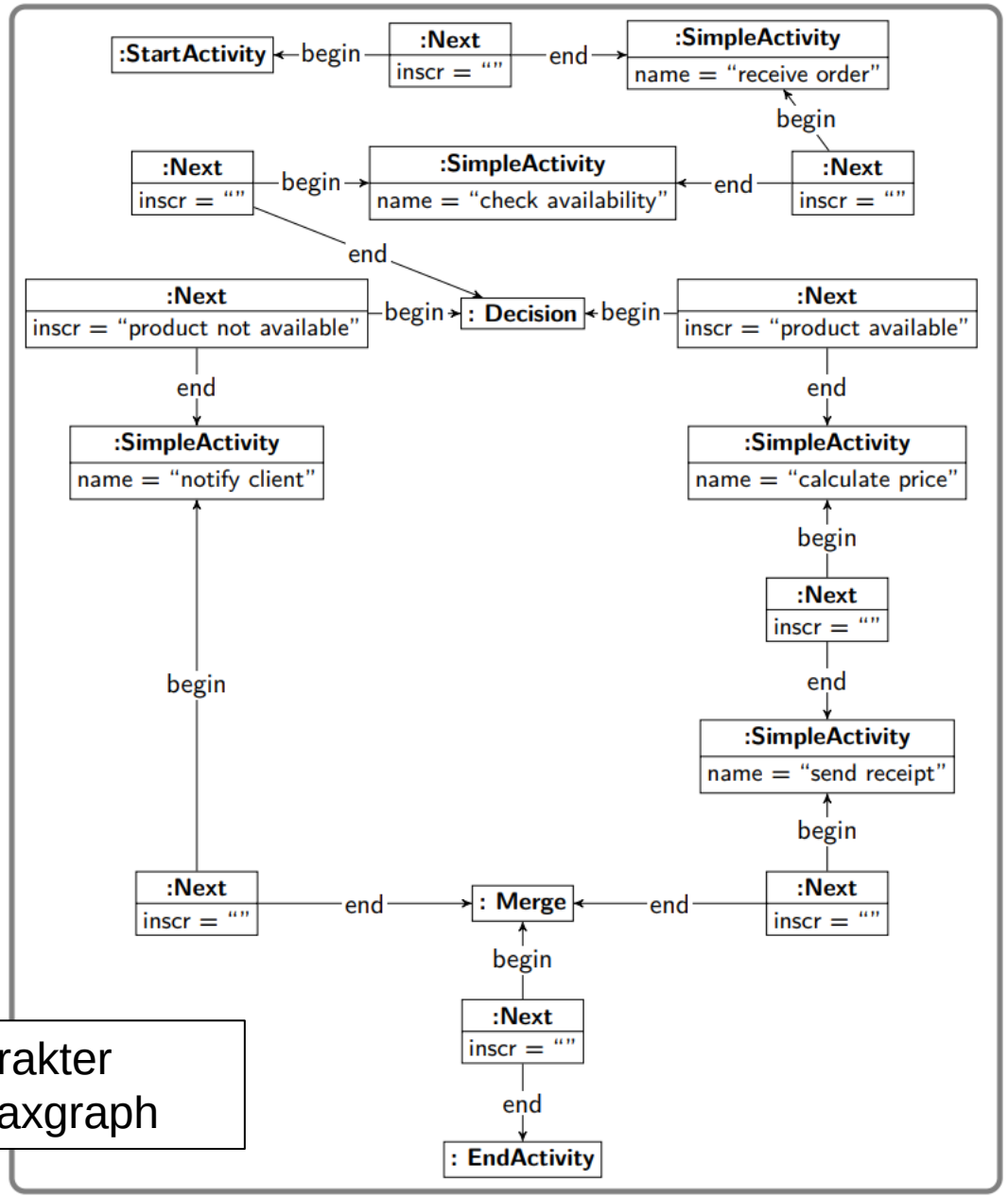
Metamodell für einfache Aktivitätsdiagramme (Typgraph)



Abstrakter Syntaxgraph

Konkretes Modell

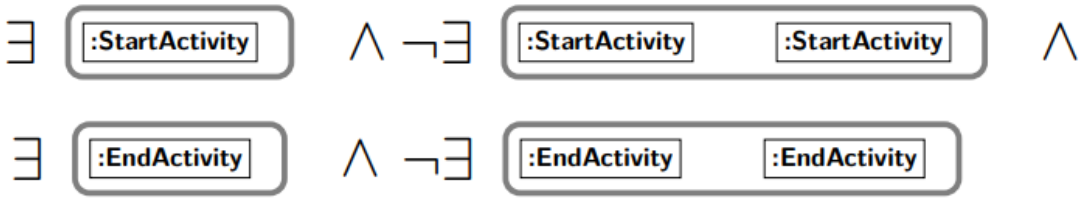




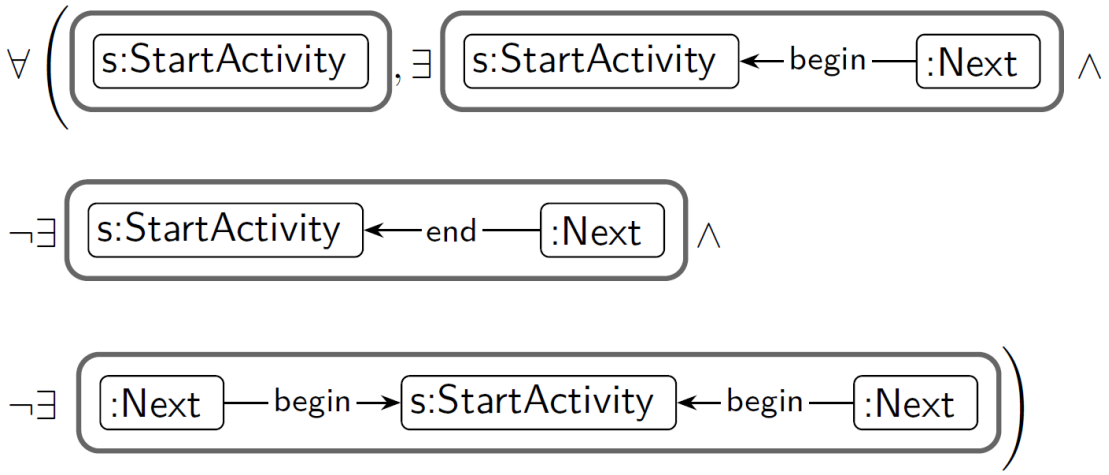
Abstrakter Syntaxgraph

Weitere Sprach-Constraints

Es gibt genau eine Start- und eine Endaktivität.

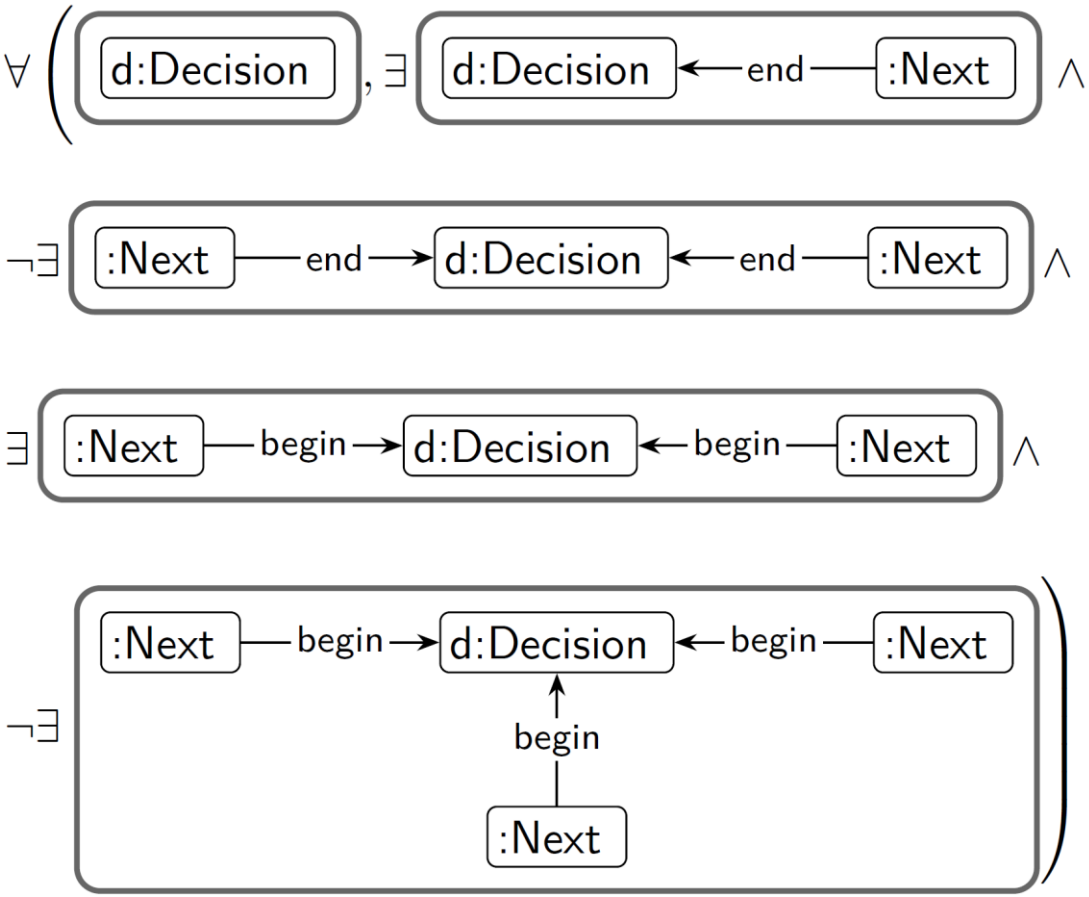


Die Startaktivität hat keine eingehende, aber eine ausgehende Transition.



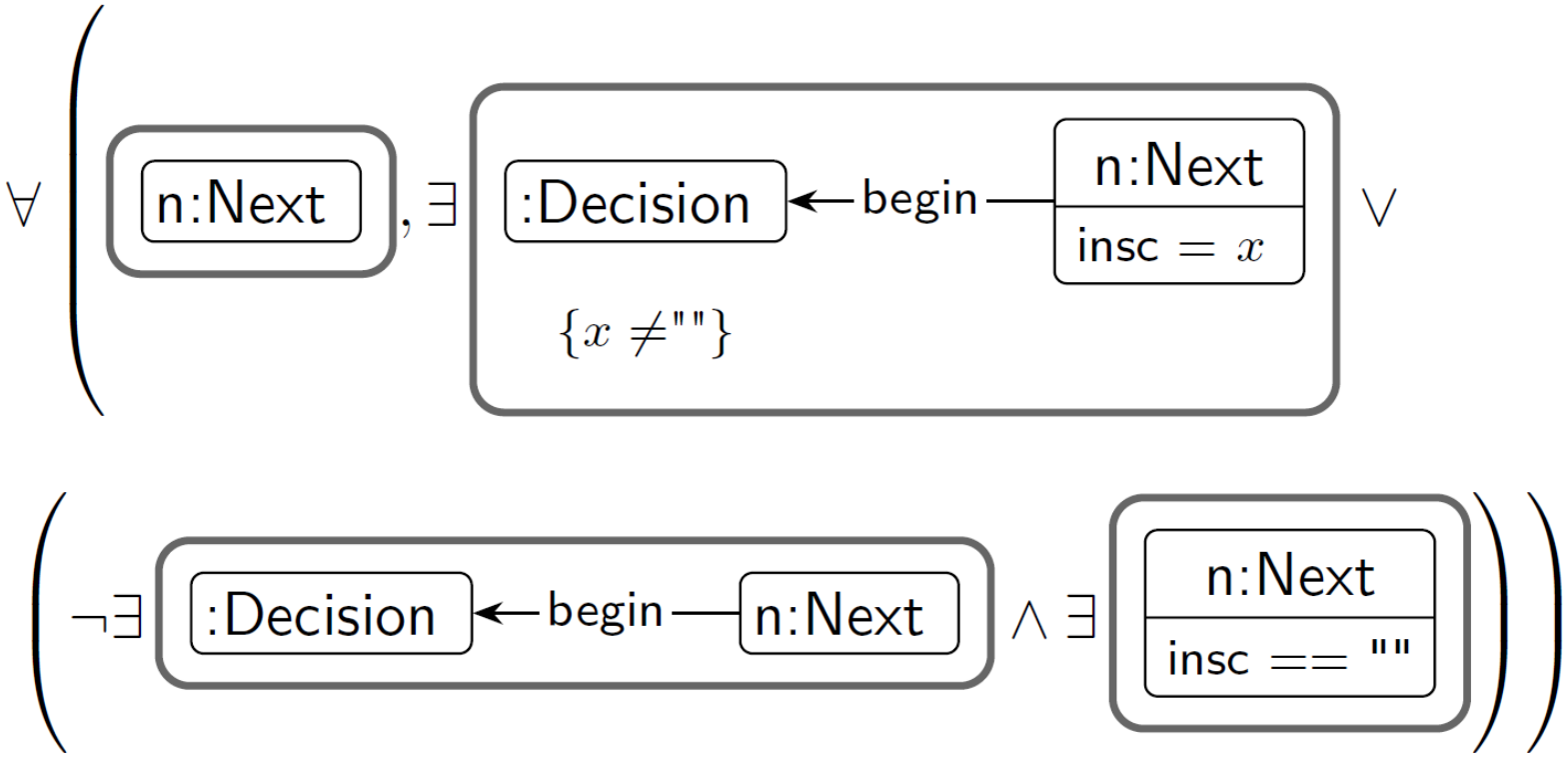
Weitere Sprach-Constraints

Jede Entscheidung hat eine eingehende und zwei ausgehende Transitionen.

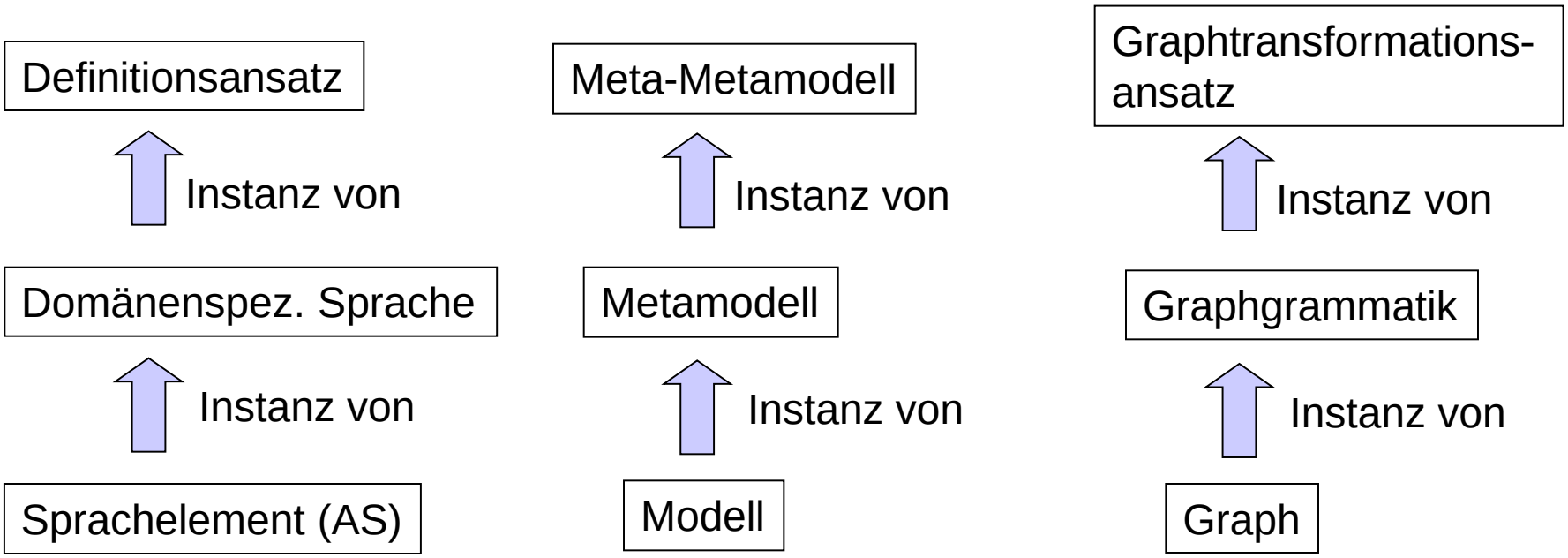


Weitere Sprach-Constraints

Nur die ausgehenden Transitionen einer Entscheidung haben Beschriftungen.



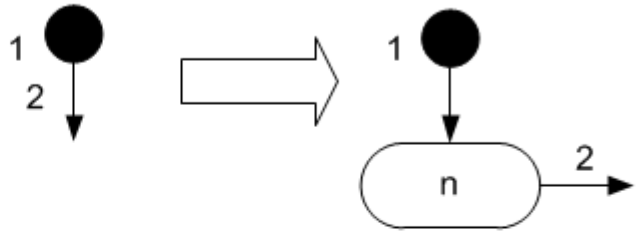
Syntaxdefinition domänenspezifischer Sprachen



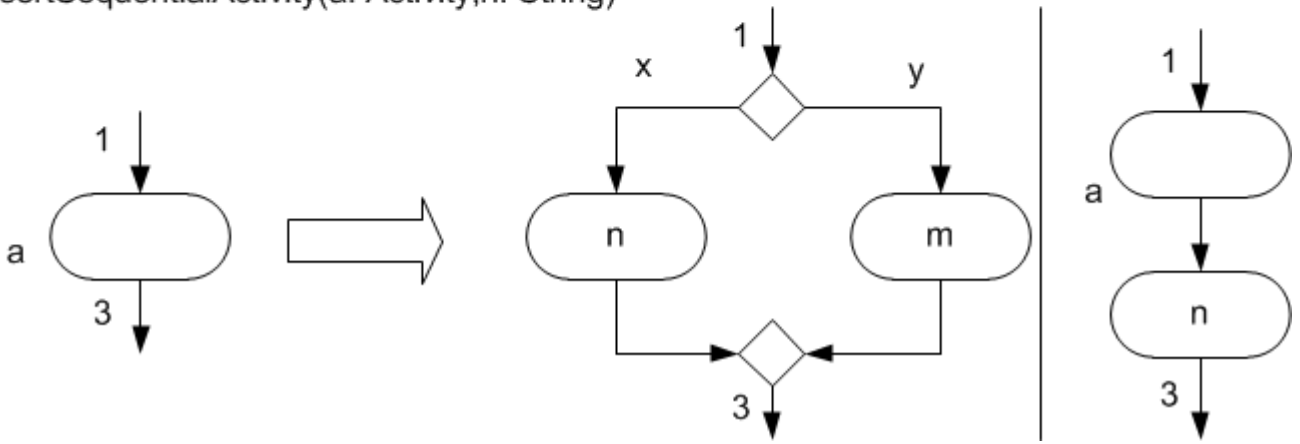
Beispiel: Grammatik für einfache Aktivitätsdiagramme

start graph

insertSimpleActivityAfterStart(n:String)

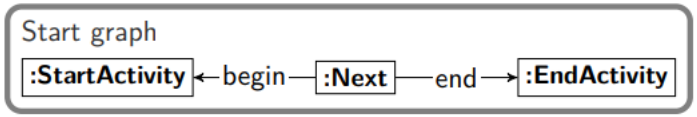


insertDecision(a: Activity, x,y,n,m: String)
insertSequentialActivity(a: Activity,n: String)

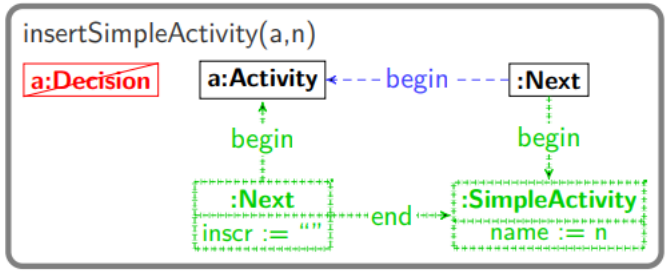


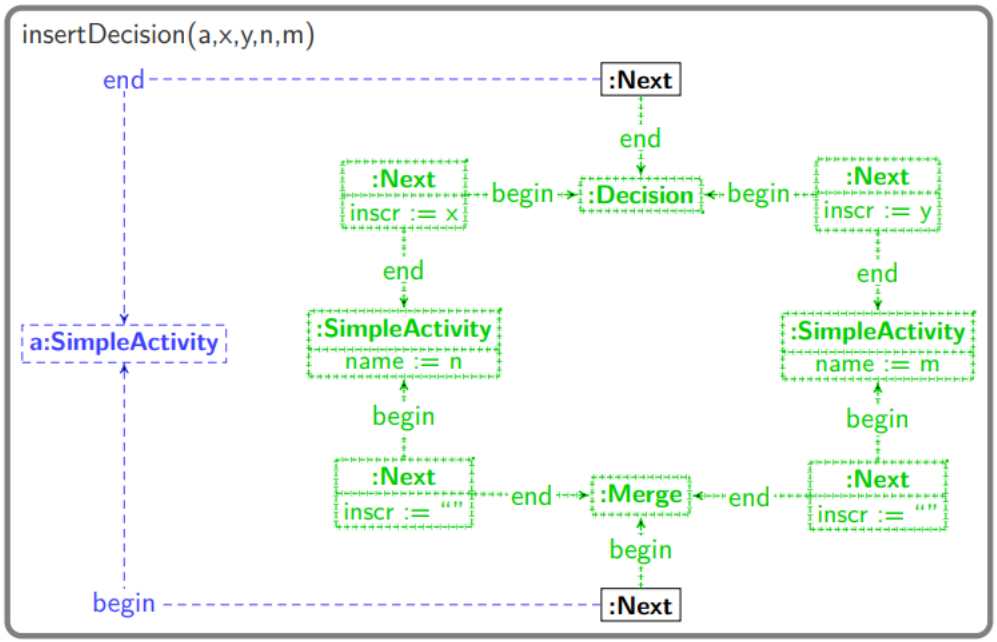
Beispiel: Abstrakte Syntaxgrammatik

Startgraph:



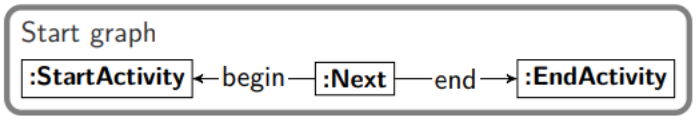
Regeln:

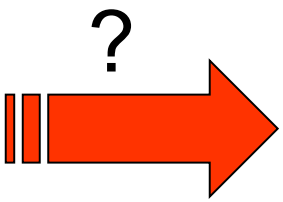


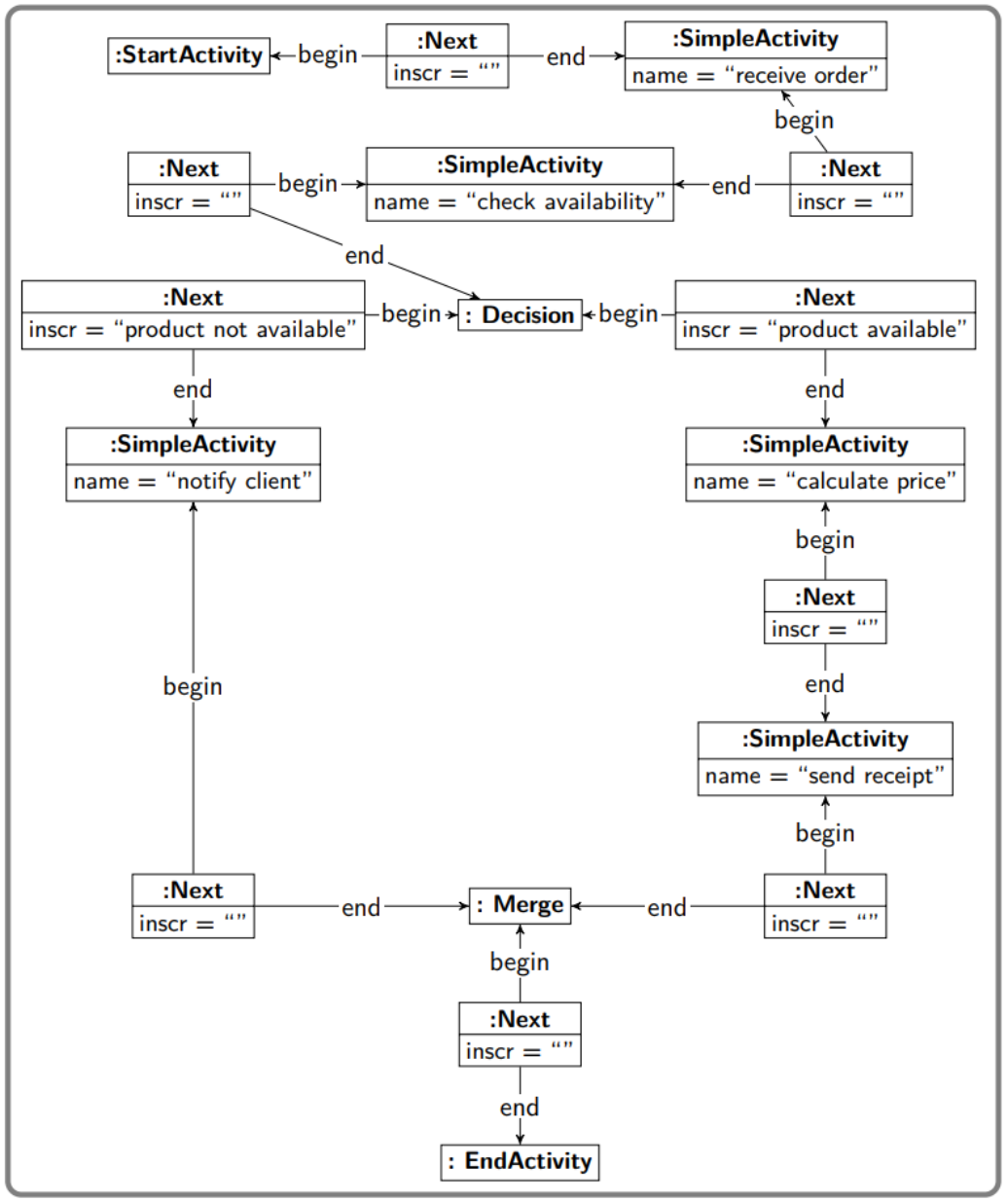


Beispiel: Eine Ableitungssequenz

Startgraph:







Generieren eines abstrakten Aktivitätsdiagramms

Ableitungssequenz:

- insertSimpleActivity(start,"receive Order");
- insertSimpleActivity (receiveOrder, "check availability");
- insertSimpleActivity (checkAvailability, "Decision");
- insertDecision(decision, "product not available", "product available", "notify client", "calculate price");
- insertSimpleActivity (calculatePrice, "send receipt");

receiveOrder, checkAvailibility, decision und calculatePrice sind die Identifier von Activity-Knoten.

Parsing von Modellen

- Graph-Parsing wendet inverse Transformations-regeln an, um einen abstrakten Syntaxgraph in den Startgraph zu reduzieren.
  - *Der Parsingprozess ist i.a. nichtdeterministisch.*
- Beispiel:
  - *insertSimpleActivity $^{-1}$(start,"receive Order");*
  - *insertSimpleActivity $^{-1}$(calculatePrice, "send receipt");*
  - *insertDecision $^{-1}$(decision, "product not available", "product available", "notify client", "calculate price");*
  - *insertSimpleActivity^{-1}(start,"check availability");*
  - *insertSimpleActivity^{-1}(start,"Decision");*

Freies Editieren von Modellen

- feste Sprachsymbole, frei verwendbar:
  - *festes Alphabet*
  - *Zusätzliche Wohlgeformt-heitsregeln müssen überprüft werden.*
- feste graphische Symbole, frei verwendbar:
  - *kein Alphabet definiert*
  - *Abstrakte Syntax muss erkannt werden.*
- freie Zeicheneditoren:
  - *Mit einem Stift können beliebige Formen eingegeben werden.*
  - *Graphische Symbole müssen erst erkannt werden.*
- Von welcher Art sind Modell-editoren typischerweise?

Syntaxgesteuertes Editieren von Modellen

- sprachabhängige Editieroperationen
- Einfügen und Löschen ganzer Sprachkonstrukte
- weitgehend automatisches Layout
- Vor- und Nachteile:
  - *Das Modell ist immer korrekt.*
  - *Ein Syntaxcheck des Modells ist nicht nötig.*
  - *Die Editieroperationen geben meist eine bestimmte Arbeitsweise vor.*
  - *Das Layout kann nur begrenzt individuell festgelegt werden.*

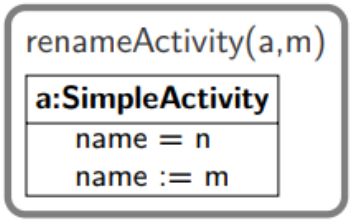
Komplexe Editieroperationen

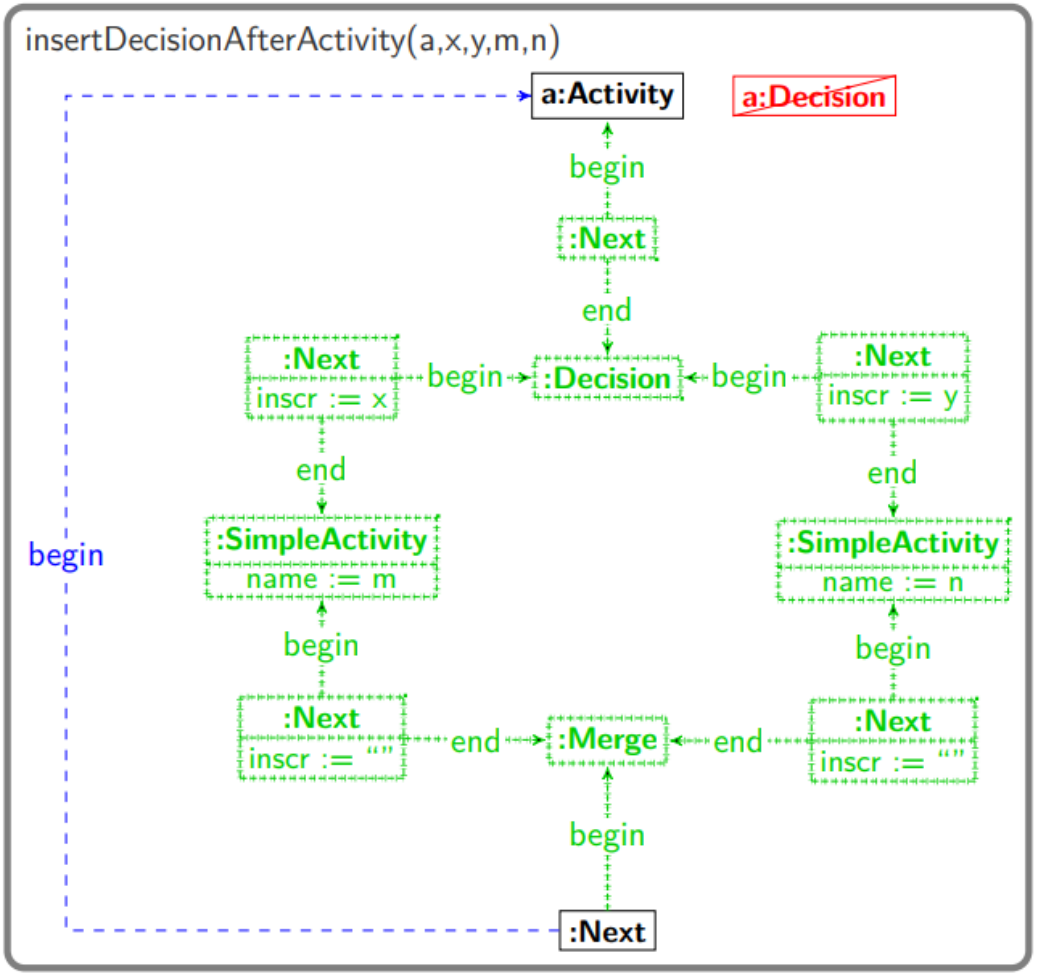
- Editierschritte können durch Graphtransformationsregeln spezifiziert werden.
- Graphgrammatikregeln geben Hinweise auf sinnvolle Editieroperationen.
  - *Einfachere Editieroperationen lassen flexibles Editieren zu, erlauben aber auch inkonsistente Modelle. QuickFix-Operationen wären sinnvoll, um wieder zu konsistenten Diagrammen zu gelangen.*
  - *Komplexere Editieroperationen können sinnvoll sein, um größere Änderungen (z.B. für Refactorings) in einem Schritt durchzuführen.*

Beispiel: Syntaxgesteuertes Editieren von Aktivitätsdiagrammen

- Einfügen:
  - *Einfügen von einfachen Aktivitäten nach der Startaktivität*
  - *Einfügen von einfachen Aktivitäten nach einfachen*
  - *Einfügen von Entscheidungsstrukturen*
- Verändern:
  - *Umbenennen von Aktivitäten und Bedingungen*
- Löschen:
  - *inverse Einfügeregeln mit weniger Parametern*

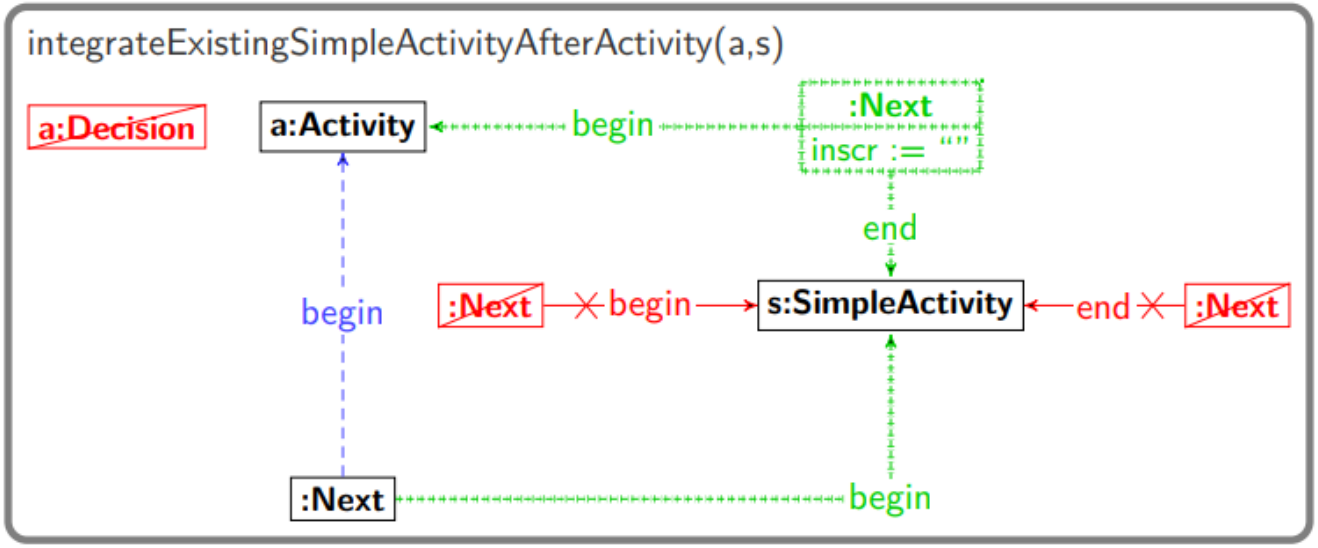
Beispiel: Simple und komplexe Editieroperationen





Freies Editieren und Quickfixing

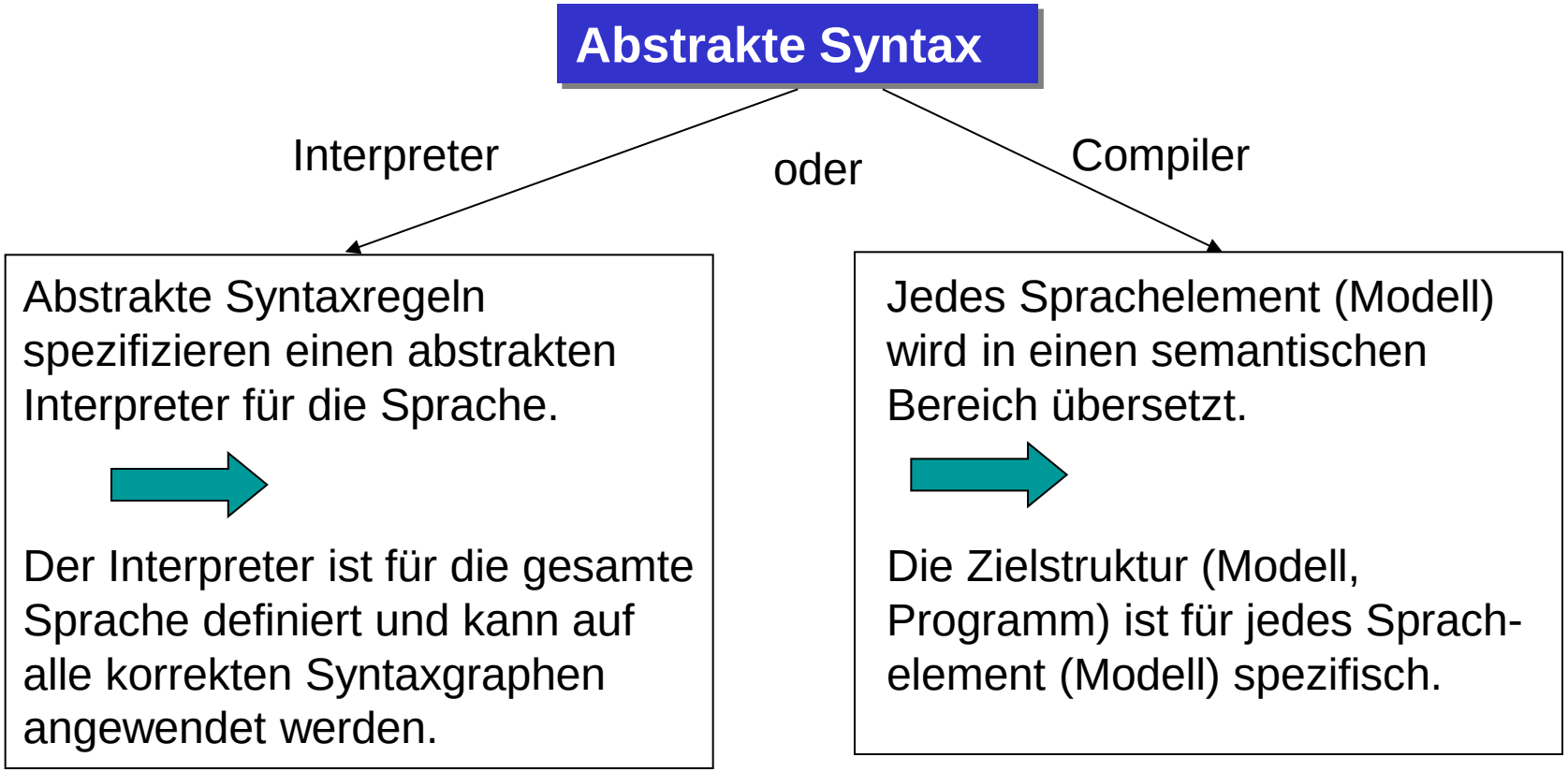
- Freies Einfügen von
  - *SimpleActivity, Next, Decision,…*
  - *Und nachfolgendes Quickfixing*
  - *Beispiel: Quickfixing nach freiem Einfügen einer SimpleActivity*



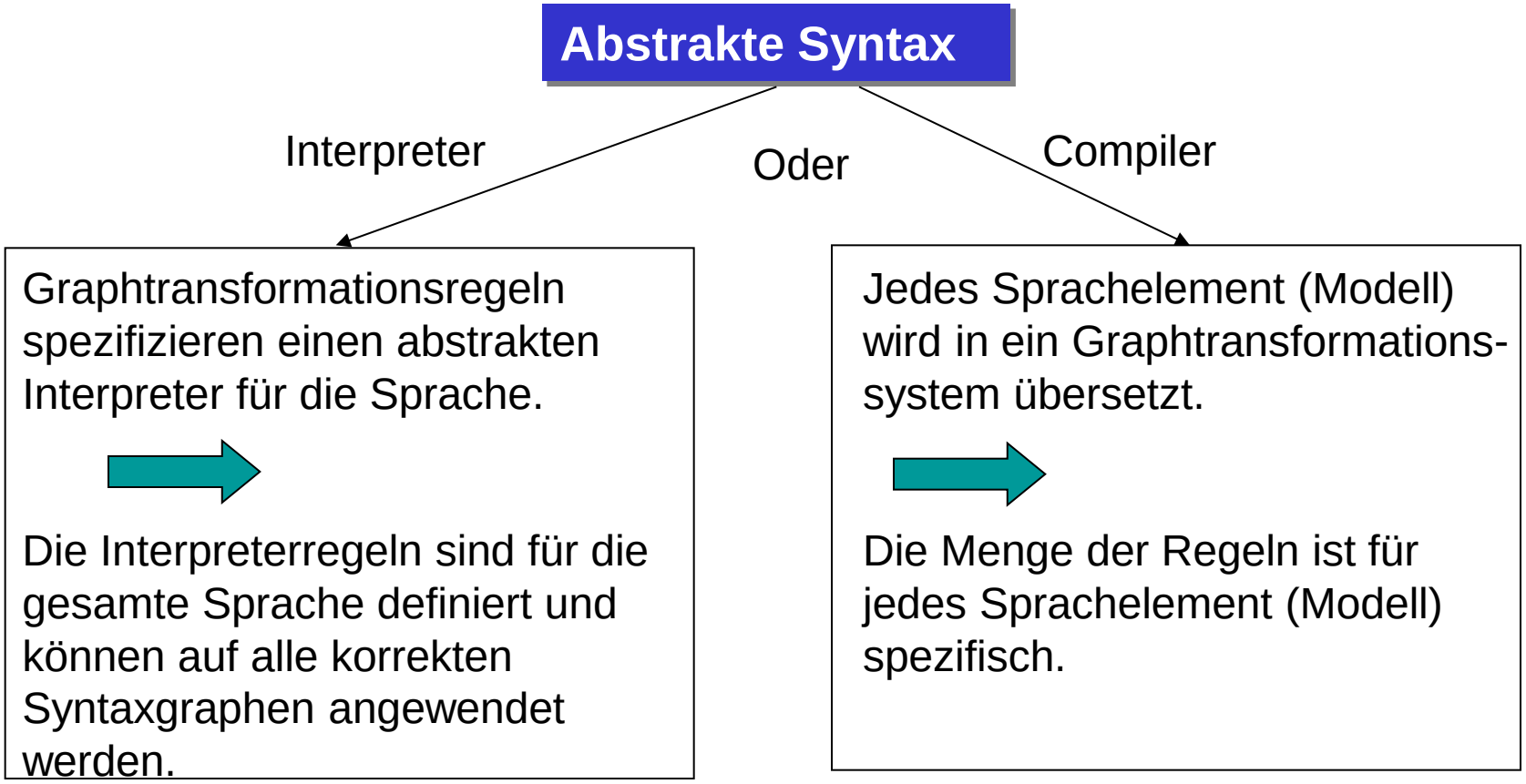
Editorgenerierung

- Das Metamodell, die Grammatikregeln und die zusätzlichen Editierregeln bilden ein Sprachmodell, aus dem ein domänenspezifischer Modelleditor generiert werden kann.
- Nur basierend auf dem Metamodell:
  - *Basiseditieroperationen zum Einfügen, Löschen und Verändern von Modellelementen (aus dem Metamodell abgeleitet)*
  - *Komplexe Editieroperationen und Quickfixing (manuell erzeugt)*
- Zusätzlich: Grammatikregeln
  - *Basis für Graphparsing*
  - *Semiautomatische Erzeugung von Quickfix-Operationen*

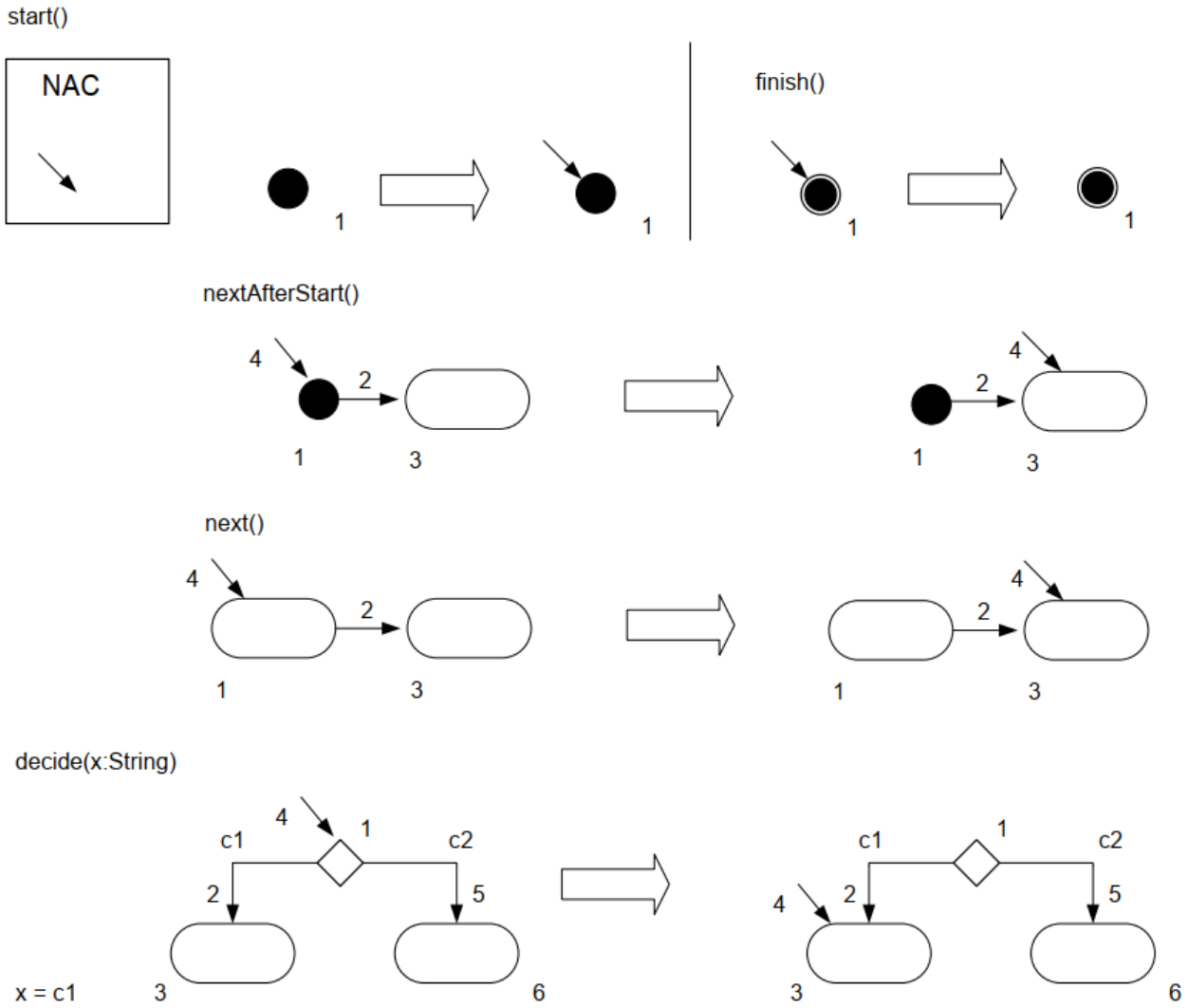
Semantik: 2 Wege zur Definition



Semantik durch Graphtransformation



Interpretersemantik für A.diagramme

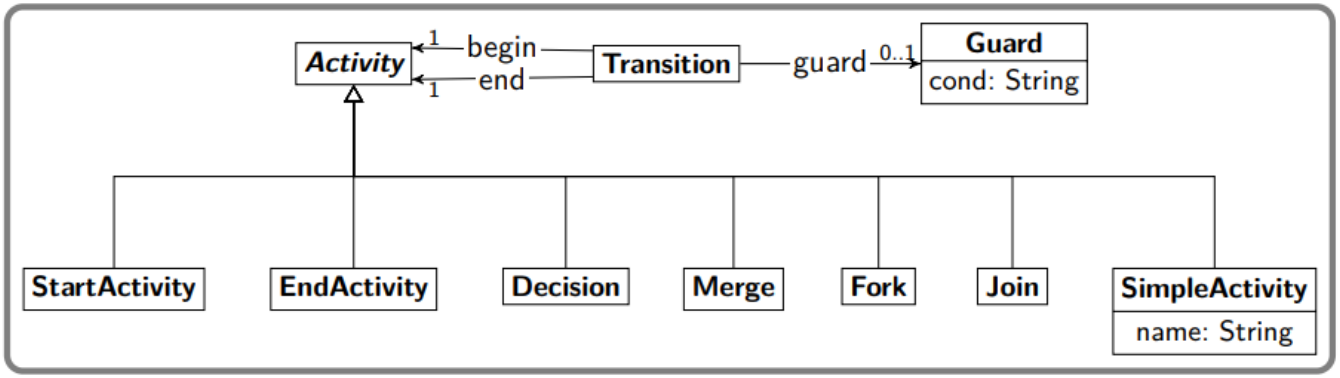


Sprachevolution

Sprachevolution bedeutet die Anpassung der Sprachspezifikation und -werkzeuge an geänderte Anforderungen.

Beispiel:

- *Eine Next-Relation heißt jetzt Transition.*
- *Beschriftungen werden durch Guards ersetzt.*
- *Fork und Join-Aktivitäten kommen hinzu.*
- *Neues Constraint: Jede Fork-Aktivität hat eine eingehende und zwei ausgehende Transitionen (analog für Join).*
- *Die Menge der Editierregeln muss angepasst werden.*



Zusammenfassung

- DSMLs können durch Metamodelle und/oder Graphgrammatiken spezifiziert werden.
  - *Metamodelle bieten eine deklarative Sicht auf eine Sprache, Graphgrammatiken eine konstruktive.*
- Grammatikregeln helfen bei der Definition von Editieroperationen für Modelleditoren.
- Mit dem Graphtransformationsansatz kann für eine DSML eine Interpretersemantik definiert werden.