

# $XG^{ep}$ - XML basiertes Editieren und Prozessieren formaler Grammatiken

Jan Behmann, Dirk Dörschlag, Jörg Schmittwilken  
Institut für Geodäsie und Geoinformation, Universität Bonn

12. September 2010

## Zusammenfassung

In den vergangenen Jahre wurde mehrfach gezeigt, dass sich formale Grammatiken sehr gut zur Modellierung von 3D Stadt- und Gebäudemodellen eignen. Grundsätzlich lassen sich zwei Anwendungsbereiche dieser Modelle unterscheiden: (1) Das prozedurale Generieren von synthetischen Modellen und (2) die Rekonstruktion von Gebäuden oder Gebäudeteilen aus Bildern oder 3D Punktwolken bzw. die Klassifikation dieser Daten.

In diesem Beitrag wird eine Konzept zur Definition und Verarbeitung formaler Grammatiken und dessen Implementierung vorgestellt:  $XG^{ep}$  - *XML based grammar editing and processing*.

Der Fokus der Arbeit liegt auf Grammatiken für geometrische Objekte, insbesondere Gebäude.  $XG^{ep}$  zeichnet sich durch die Verwendung offener Standards und durch seine Modularität und Erweiterbarkeit aus. Es ermöglicht den Umgang mit allen Grammatik-Typen der Chomsky Hierarchie und unterstützt attributierte Grammatiken, die Symbole mit Attributen ergänzen, ebenso wie probabilistische Grammatiken, die Produktionsregeln um Wahrscheinlichkeiten erweitern.



Abbildung 1: Synthetisch generierter Straßenzug mit Gebäuden im Jugendstil

## 1 Motivation

Abstrakte Modelle von Realweltobjekten spielen in vielen GIS-Anwendungen eine große Rolle. Durch die Generalisierung, die diese Modelle darstellen, werden Prozesse vereinfacht und lassen sich somit leichter auf Programme, Datenbankschemata o.ä. abbilden. Eine vollständige Betrachtung und Repräsentation der Realität wäre wegen der Diversität der vorkommenden Objekte zu aufwendig. Aus diesem Grunde wird die Realität abstrahiert. Im Bereich der 3D Stadt- und Gebäudemodelle ist zwischen Modellen zur Beschreibung der Semantik und solchen zur Beschreibung der Geometrie zu unterscheiden. CityGML (Gröger et al., 2008) ist ein prominentes Beispiel der semantischen Abstraktion. Der Standard beschreibt die Aggregations- und Generalisierungsbeziehungen von Stadtobjekten, lässt deren geometrische Ausprägung aber offen. In Arbeiten zur Rekonstruktion von Gebäude(bestandteilen) aus Bildern oder 3D Punktwolken werden die jeweiligen Objekte durch geometrische Primitive abstrahiert. Häufig werden beispielsweise Fenster durch Rechtecke oder Quader beschrieben (Pu und Vosselman, 2006), wenngleich die konkrete Form häufig nicht einem Quader entspricht (Fensterbögen o.ä.).

Auch formale Grammatiken eignen sich als Formalismus zum Modellieren von Realweltobjekten. Insbesondere eignen sich diese regelbasierten Ersetzungssysteme zur Modellierung von Generalisierung und Aggregation. Außerdem kann die Beschreibung der geometrischen Ausprägung der modellierten Objekte in die Grammatik integriert werden. Gerade wegen der regelmäßigen Struktur von Gebäuden und ihrer Bestandteile können selbst Objekte mit hohem Detaillierungsgrad mit wenigen Regeln modelliert werden.

Wegen ihrer regelbasierten Struktur können Grammatiken zufallsbasiert abgeleitet werden, um so synthetische Instanzen der beschriebenen Objekte zu erhalten. Krückhans und Schmittwilken (2009) (vgl. Abbildung 1) und Müller et al. (2006) zeigen dies für Gebäude. Die so erhaltenen Instanzen entsprechen grundsätzlich der durch die Grammatiken definierten Struktur, ihre konkrete Ausprägung kann jedoch durch einen zufallsbasierten Prozess und die Berücksichtigung der Variation der Realität variieren.

Ebenso können Grammatiken den Prozess der Rekonstruktion von Objekten aus Beobachtungen wie Bildern oder Punktwolken steuern, da sie Modellwissen in den Rekonstruktionsprozess einfließen lassen können und die Rekonstruktion somit beschleunigen oder robuster machen (Becker, 2009; Ripperda, 2008).

Das in diesem Beitrag präsentierte Konzept  $XG^{ep}$  wurde entwickelt, um eine hoch konfigurierbare und erweiterbare Plattform zum Umgang mit formalen Grammatiken zu schaffen. Im Folgenden werden die wesentlichen Konzepte von  $XG^{ep}$  vorgestellt und einige Anwendungen beschrieben. Der Rest des Aufsatzes ist wie folgt gegliedert: Im nächsten Kapitel (2) werden die theoretischen Grundlagen formaler Grammatik behandelt. Im Anschluss werden das zugrunde liegende XML-Schema (Kapitel 3) und die Hauptkomponenten der Java-Implementierung, nämlich der Prozessierer (Kapitel 4) und der Instanzierer (Kapitel 5), beschrieben. Abschließend werden in Kapitel 6 Ergebnisse vorgestellt und eine abschließende Betrachtung präsentiert.

Alle in diesem Beitrag gezeigten Beispiele und Deoms sind im Internet unter der Adresse <http://www.ikg.uni-bonn.de/forschung/xgep.html> verfügbar.

## 2 Grammatiken

Grundsätzlich ist allen Arten formaler Grammatiken gemein, dass es sich um regelbasierte Ersetzungssysteme handelt. Eine Grammatik  $G$  ist definiert als Quadrupel  $\{N, T, S, P\}$  aus Nichtterminalsymbolen  $N$ , Terminalsymbolen  $T$ , einem Startsymbol  $S \in N$  und einer Menge  $P$  von Produktionsregeln, welche die Art der Ersetzung beschreiben:  $(N \cup T)^+ \rightarrow (N \cup T)^*$ . Grundsätzlich wird das Symbol der linken Seite einer Produktionsregel durch die Symbole der rechten Seite ersetzt. Die Ersetzung, auch Ableitung genannt, beginnt beim Startsymbol und endet wenn ausschließlich Terminalsymbole existieren. Die Ableitung einer Grammatik lässt sich im Ableitungsbaum darstellen. Die Blätter des Ableitungsbaum enthalten die Terminalsymbole, die ein Wort der durch die Grammatik definierten Sprache bilden.

### 2.1 Grundlagen

Chomsky (1956) definiert vier Typen von Grammatiken, die sich in der Form ihrer Produktionsregeln unterscheiden. Die vier Typen sind als *Chomsky-Hierarchie* bekannt und werden als *reguläre* (Typ 3), *kontext-freie* (Typ 2), *kontext-sensitive* (Typ 1) und *rekursiv aufzählbare* (Typ 0) Grammatiken bezeichnet.

Die von Chomsky definierte Art formaler Grammatiken wurde von Knuth (1968) durch *Attribute* erweitert (attributierte Grammatiken). Die Symbole solcher Grammatiken werden durch Attribute ergänzt, deren Propagierung im Ableitungsbaum in *semantischen Regeln* beschrieben wird, die wiederum den Produktionsregeln zugeordnet werden. Es wird zwischen *vererbten* und *synthetisierten* Attributen unterschieden. Erstere sind einem Symbol der rechten Seite einer Produktionsregel zugeordnet und berechnen sich aus Attributwerten ihrer Eltern im Ableitungsbaum. Wohingegen letztere einem beliebigen Symbol der Produktionsregel zugeordnet sind und sich aus Attributwerten ihrer Geschwister oder Kinder berechnen.

Semantische Regeln können aber nicht nur zur Berechnung von Attributwerten verwendet werden. Hier können auch Ungleichungen für die Attribute definiert werden, die in  $XG^{ep}$  als Pre- oder Post-Constraints über die Anwendbarkeit einer Produktionsregel entscheiden. *Pre-Constraints* werden vor der Regelausführung geprüft, *Post-Constraints* erst nach Ausführung der Produktionsregel und aller weiteren zur Produktionsregel gehörigen semantischen Regeln.

Probabilistische Grammatiken (Geman und Johnson, 2002) erweitern die bisher vorgestellten Grammatiken um Wahrscheinlichkeiten, die Produktionsregeln zugeordnet werden. Gibt es mehrere Produktionsregeln mit dem gleichen Symbol auf der linken Seite, so kann die Wahrscheinlichkeit der jeweiligen Regelanwendung spezifiziert werden. Gerade bei der Verwendung von Grammatiken zur Modellierung der Realwelt können durch diese Wahrscheinlichkeiten relative Häufigkeiten bestimmter Phänomene wie beispielsweise die Breite von Fenstern oder die Wahrscheinlichkeit des Vorhandenseins einer Treppe vor der Eingangstür beschrieben werden.

## 2.2 Geometrische Grammatiken

Schmittwilken et al. (2007) zeigen das Abbilden von UML-Klassendiagrammen (Unified Modelling Language) und OCL-Regeln (Object Constraint Language) auf kontext-freie, attributierte Grammatiken. Insbesondere Generalisierung und Aggregation können in Produktionsregeln ausgedrückt werden. Aufgrund der Semantik eines jeden Symbols repräsentiert der Ableitungsbaum eines Wortes immer auch ein semantisches Modell des Objekts.

Die Repräsentation geometrischer Objekte mit attributierten, kontext-freien Grammatiken zeigen Schmittwilken et al. (2009a). Sie erläutern die Repräsentation von Geometrie und insbesondere Symmetrie und Topologie von 3D-Stadtobjekten wie Gebäuden oder Gebäudeteilen durch Grammatiken. Grundlage des dort vorgestellten Konzept ist bereits ein Prototyp von  $XG^{ep}$ . Auch die von Krückhans und Schmittwilken (2009) beschriebenen synthetischen Gebäudemodelle wurden mit  $XG^{ep}$  erzeugt.

Vor allem zur Modellierung von Pflanzenwachstum werden L-Systeme (Prusinkiewicz und Lindenmayer, 1991) verwendet. Durch die Interpretation der erzeugten Worte als Anweisungen für "Turtle-Programme" lassen sich L-Systeme sehr einfach grafisch darstellen. Formal unterscheiden sich L-System von den oben beschriebenen formalen Grammatiken der Chomsky-Hierarchie durch die Art der Ableitung. Zur realitätsnahen Simulation von Pflanzenwachstum werden in einem Ableitungsschritt alle Symbole abgeleitet. Terminalsymbole werden stets auf sich selbst abgebildet.  $XG^{ep}$  kann zur Modellierung und Prozessierung von L-Systemen modifiziert und erweitert werden (vgl. Kapitel 4).

Anders als die bisher beschriebenen Grammatiken sind Shape-Grammatiken (Stiny und Gips, 1972; Knight, 1999) nicht auf Basis von Symbolen definiert. Die Primitive dieser Grammatiken sind geometrische Formen und die Produktionsregeln beschreiben deren affine Transformation. Da  $XG^{ep}$  auf symbolisch definierten Grammatiken operiert, sind zur Verarbeitung von Shape-Grammatiken ebenfalls Modifikationen notwendig.

## 2.3 Straßenzug-Grammatik

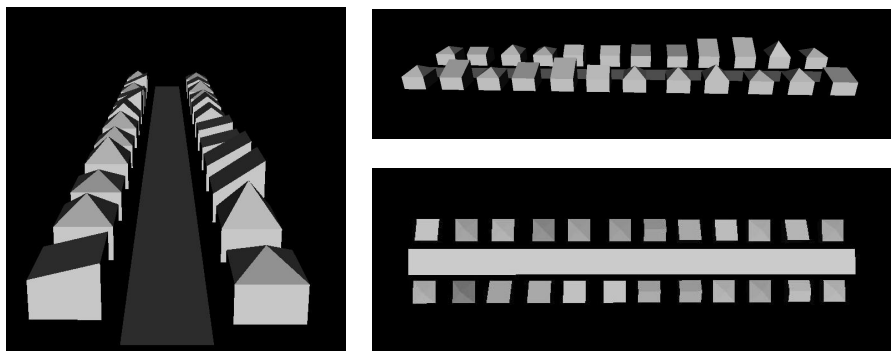


Abbildung 2: Mit einer attributierten, probabilistischen Grammatik erstellter Straßenzug

Listing 1 zeigt eine Beispielgrammatik. Terminalsymbole beginnen wie üblich mit einem Kleinbuchstaben, Nichtterminale mit einem Großbuchstaben. Die hochgestellten Indizes dienen lediglich der Unterscheidbarkeit der Symbole. Die Produktionsregeln sind zur besseren Lesbarkeit in der üblichen „Pfeilnotation“ dargestellt.

Abbildung 2 zeigt ein mit  $XG^{ep}$  erzeugtes Modell der Straßen-Szene.

---

```

P1: Strasse → Haeuserreihe1 Strasse Haeuserreihe2
P2: Haeuserreihe1 → Gebaeude Haeuserreihe2
P3: Gebaeude → gebaeudekoerper Dach
P4: Dach → walmdach
P5: Dach → satteldach
P6: Dach → pultdach

```

---

Listing 1: Beispielgrammatik für eine Straße mit beidseitiger Bebauung und unterschiedlichen Dachformen

Die semantischen Regeln der Grammatik sind auszugsweise in Listing 2 dargestellt. Attribute sind hier durch Punkte von Symbolen getrennt. Der Großteil der semantischen Regeln wird typischerweise für die Vererbung der Lage- und Formparameter benötigt. Hier seien einige der Regeln exemplarisch erläutert: Zeile 2 zeigt einen Pre-Constraint der die wiederholte Ausführung der rekursiven Produktionsregel steuert. Die Regel zum Erzeugen eines weiteren Symbols **Gebaeude** darf nur dann ausgeführt werden, wenn die **Haeuserreihe** genügend Platz für ein weiteres Objekt bietet. In Zeile 5 wird das Attribut der Breite eines Gebäudes mit einem zufälligen Wert belegt. Der Wert wird randomisiert auf Basis einer Normalverteilung mit Mittelwert 8,00m und einer Streuung von 2,00m erzeugt. So können Beobachtungen der Realwelt (Gebäudebreite) in die Grammatik eingepflegt werden und sinnvolle Instanzen erzeugt werden. In Zeilen 6 und 13 wird die vordefinierte Funktion `transform()` aufgerufen, um den Lagereferenzpunkt der neu erzeugten Symbole zu bestimmen.

---

```

1 P2: Haeuserreihe1 → Gebaeude Haeuserreihe2
2   pre: Haeuserreihe1.laenge >= Gebaeude.breite + Gebaeude.abstand
3   Haeuserreihe2.laenge = Haeuserreihe1.laenge - Gebaeude.breite - Gebaeude.abstand
4   Gebaeude.ref = Haeuserreihe1.ref
5   Gebaeude.breite = random(gaussian, 8.00, 2.00)
6   Haeuserreihe2.ref = translate(Haeuserreihe1.ref, Gebaeude.breite + Gebaeude.abstand, 0,0)
7
8 P3: Gebaeude → gebaeudekoerper Dach
9   gebaeudekoerper.ref = Gebaeude.ref
10  gebaeudekoerper.breite = Gebaeude.breite
11  gebaeudekoerper.tiefe = Gebaeude.tiefe
12  gebaeudekoerper.hoehe = Gebaeude.hoehe
13  Dach.ref = translate(Gebaeude.ref, 0,0,Gebaeude.hoehe)

```

---

Listing 2: Auszug aus den semantischen Regeln für zwei Produktionsregeln

### 3 Nutzung von XML Technologie

Die Speicherung der Grammatiken und der entsprechenden Ableitungsbäumen erfolgt in einem Dateiformat, welches auf der Auszeichnungssprache XML (Extensible Markup Language, Bray et al., 2008) basiert. Die Verwendung von XML als Speicherformat bringt alle XML-typischen Vorteile, wie beispielsweise Menschen- und Maschinenlesbarkeit oder eine Vielzahl von Einlese- und Ausgabeimplementationen mit sich. Im Rahmen des  $XG^{ep}$ -Projekts wird XML zur strukturierten Speicherung von Informationen eingesetzt. Weiterhin wird XSLT verwendet, um die Grammatik-Definition zu vereinfachen. Beides wird im Folgende beschrieben.

#### 3.1 XML - Schema

XML definiert bestimmte Eigenschaften einer Datei, wie die Strukturierung durch Elemente, das genaue Dateiformat wird aber erst durch ein XML-Schema (Fallside und Walmsley, 2004) spezifiziert. Durch die Verwendung eines auf diese Weise festgelegten Dateiformats, wird eine problemlose Austauschbarkeit von Grammatiken und eine Validierung gegebener Dateien ermöglicht.

Das hierfür verwendete Schema ist eine XML-Datei, welche den Aufbau des Dokuments und die verwendbaren Strukturelement-Typen beschreibt. In dem von  $XG^{ep}$  verwendeten Grammatik-Schema wurden folgende Bestandteile implementiert: Symbole, Produktionsregeln, Attribute, semantische Regeln, Wahrscheinlichkeiten und die Kontextsymbole kontextsensitiver Grammatiken. Abbildung 3 zeigt in einem UML-Klassendiagramm die Abbildung des XML-Schemas für Grammatiken auf Klassen. Das entwickelte XML-Schema ist zur Einführung neuer Programmeigenschaften oder zur Limitierung auf spezielle Grammatiktypen erweiterbar. Zur offline-Validierung ist das Schema in  $XG^{ep}$  enthalten, zur externen Validierung jedoch auch im Internet frei zugänglich.

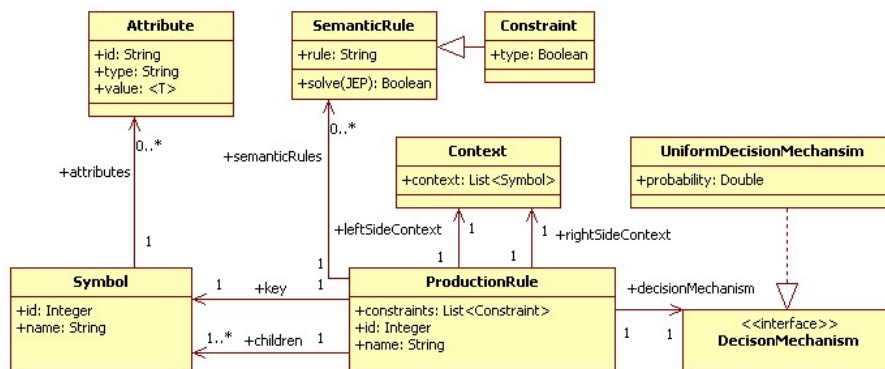


Abbildung 3: Aufbau der verwendeten Grammatikdatei

### 3.2 XSL-Transformationen

XSL-Transformationen sind „Recommendations“ des W3C-Konsortiums (Clark, 1999), werden in der XML-Sprache formuliert und wurden zur Überführung eines XML-Dokuments in ein neues Dokument von beliebigem Typ entwickelt. Im Rahmen von  $XG^{ep}$  wird jedoch ausschließlich die Fähigkeit zur Überführung in andere XML-Dokumente genutzt.

Beispielsweise wird die Berechnung der Lage-Parameter von Geometrie-Objekten unter Nutzen von XSLT realisiert. Die räumliche Lage und Orientierung eines Objekts wird dabei durch eine homogene 3D-Transformationsmatrix beschrieben. Die Transformationsmatrix eines Symbols ergibt aus der räumlichen Lage seines Vater-Symbols und einer zusätzlichen Transformation relativ zu diesem. Die 16 Parameter der Transformationsmatrix eines jeden Symbols werden im Attribut *ref*, einem s.g. *komplexen Attribut*, zusammengefasst. Anstatt nun die Berechnungen für alle 16 Elemente der Transformationsmatrix explizit in Form semantischer Regeln anzugeben, genügt es in  $XG^{ep}$  eine semantische Regel für das Attribut *ref* zu definieren, welche eine vorgegebene Transformationsfunktionen verwendet. Listing 2 zeigt in Zeile 13 beispielhaft eine derartige semantische Regel. Die verwendeten Transformationsfunktionen müssen einmalig in einer XSL-Transformation (hier `translate.xslt`) definiert werden und können dann in beliebigen Grammatiken verwendet werden. Bei der Anwendung der XSLT wird die semantische Regel, die das Attribut *ref* und die vordefinierte Funktionen (hier `translate`) verwendet, wieder in 16 semantische Regeln zur Berechnung der einzelnen Elemente der Transformationsmatrix überführt.

Listing 3 zeigt eine rekursive XSLT-Funktion, die eine Zeichenkette mit mehreren Attributbezeichnern, die durch einen Bindestrich getrennt werden, in einzelne Attribut-Elemente mit den entsprechenden Namen überführt. Dieser Ausschnitt ist Teil einer Funktion, die für die Aufteilung eines komplexen, gebündelten Attributs in die Einzelattribute verwendet wird.

Alle verfügbaren XSL-Transformationen werden hintereinander ausgeführt und erzeugen zusammen eine valide und interpretierbare Grammatikdatei. Da die Größe einer Grammatik-XML-Datei durch die Transformation stark anwachsen kann, werden die Transformationen nur für den Ableitungsprozess durchgeführt. Gespeichert werden die Grammatik-Dateien in ihrer kompakten Form unter Verwendung

komplexer Attribute und vordefinierter Funktionen.

---

```

<xsl:template name="generateAttributes">
  <xsl:param name="list"/>
  <xsl:variable name="newlist" select="normalize-space($list)"/>
  <xsl:variable name="id" select="substring-before($newlist,'-')"/>
  <xsl:variable name="remaining" select="substring-after($newlist,'-')"/>
  <grammar:Attribute grammar:key="{ $id }" grammar:complex="false"/>
  <xsl:if test="$remaining">
    <xsl:call-template name="generateAttributes">
      <xsl:with-param name="$list" select="$remaining"/>
    </xsl:call-template>
  </xsl:if>
</xsl:template>

```

---

Listing 3: XSLT-Template zum Überführen eines komplexen Attributs in Einzelattribute

## 4 Prozessierer

Der Prozessierer bildet die Brücke zwischen Grammatiken und der durch sie beschriebenen Sprachen. Er erlaubt sowohl die Generierung von Wörtern einer Sprach auf der Grundlage ihrer Grammatik, als auch das Identifizieren von zur Grammatik passenden Wörtern (parsen). Beide Funktionen werden durch  $XG^{ep}$  unterstützt. Für die beiden Module *Generieren* und *Parsen* steht ein gemeinsamer Programmkern bereit. Die unterschiedlichen Funktionalitäten der beiden Module werden im Folgenden vorgestellt.

Das Generierungsmodul dient zur Ableitung von Wörtern einer Grammatik. Die jeweiligen Definitionen von Grammatiken müssen den in Tabelle 1 gezeigten Typen von Regeln entsprechen.

Typ	Regelpattern	Symbole
regulär	$P_x :: A \rightarrow aA Aa a \epsilon$	$A \in N, a \in T$
kontext-frei	$P_x :: A \rightarrow \gamma$	$A \in N, \gamma \in (N \cup T)^*$
kontext-sensitiv	$P_x :: \alpha A \beta \rightarrow \alpha \gamma \beta$	$A \in N, \gamma \in (N \cup T)^+, \alpha, \beta \in (N \cup T)^*$
attribuiert	$R_{P_x}^j :: v = f(W)$ (Übertragungsfunktion) $R_{P_x}^i :: true == f(V)$ (Bedingung)	$v \in V, W \subset (V \setminus \{v\})$

Tabelle 1: Von  $XG^{ep}$  verarbeitbare Regeltypen.  $N$  ist die Menge der Nichtterminale,  $T$  die Menge der Terminale,  $V$  die Menge der Attribute aller mit einer einzelnen Produktionsregel verknüpften Symbole

Die Umsetzung des Generierungsmoduls in Java ist in Abbildung 4 in einem UML-Klassendiagramm dargestellt. Ein zentrales Merkmal der Implementierung ist die Unterstützung attributierter Grammatiken, welches insbesondere bei der Umsetzung räumlicher Grammatiken benötigt wird. Bei attributierten Grammatiken verfügen die Symbole der Grammatik über Attribute, deren Werte über den Produktionsregeln  $P_x$  angegliederte Übertragungsfunktionen  $R_{P_x}^j$  im Ableitungsbaum propagiert werden.  $XG^{ep}$  unterstützt das Synthetisieren und Vererben von Werten. Semantische Regeln mit synthetisierten Werten werden in dem Moment gelöst, in welchem diese während des Ableitungsvorgang erstmals mit Werten belegt werden.

Die semantischen Regeln können ebenfalls für die Umsetzung von Bedingungen  $R_{P_x}^i$  für die Anwendung von Produktionsregeln verwendet werden. In  $XG^{ep}$  sind sowohl Vor- als auch Nachbedingungen für die Ausführung einer Produktion realisierbar. Bevor eine Produktionsregel angewendet wird müssen alle Vorbedingungen wahr sein und nach Anwendung einer Produktionsregel müssen alle Nachbedingungen zutreffen, damit das Ergebnis in den Ableitungsbaum übertragen wird.

In den semantischen Regeln können alle lokalen Attribute des Elternsymbols und der neu produzierten Kindersymbole über ihre lokalen Bezeichner eindeutig adressiert werden. Attribute von nicht an der Produktion beteiligter Symbole können nicht adressiert werden.

Ein wesentlicher Aspekt des Generierens ist die Auswahl einer Produktionsregel aus der Menge aller zu einem Symbol möglichen Produktionsregeln. Für die Umsetzung dieses Entscheidungsprozesses besitzt der  $XG^{ep}$ -Prozessierer Schnittstellen, die das Implementieren beliebiger Entscheidungs-

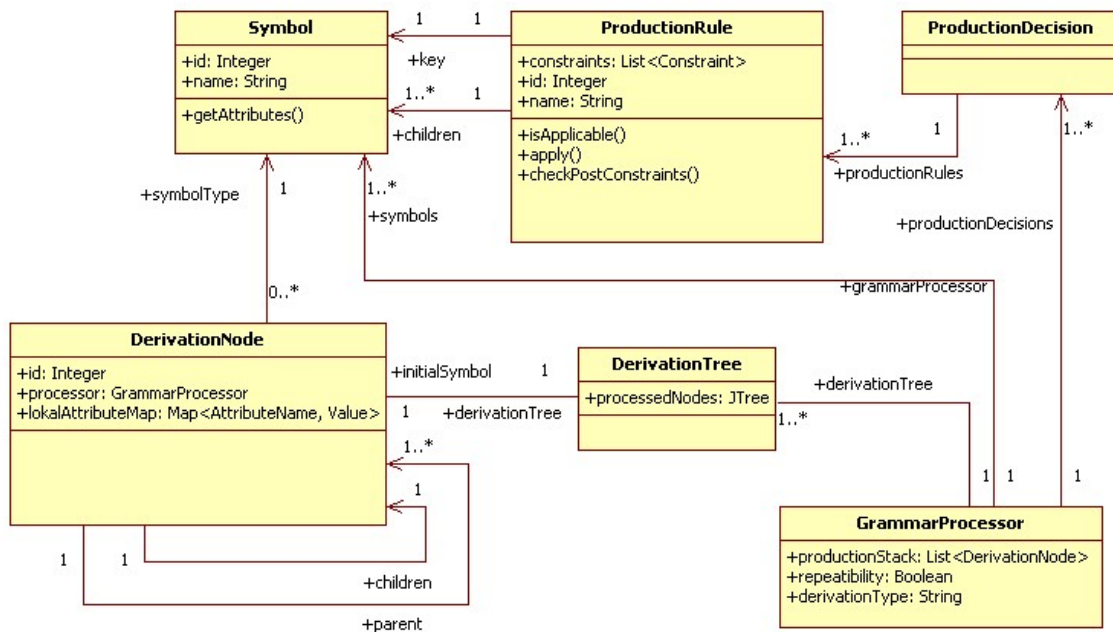


Abbildung 4: Klassen die am Ableitungsvorgang beteiligt sind

funktionen ermöglichen. Die Umsetzung der Schnittstellen in Java ist in Abbildung 5 als UML-Klassendiagramm veranschaulicht. Hierbei bildet der `DecisionMechanism` die Möglichkeit eine Bewertungsfunktion für die Ableitung einer Bewertung einer Produktionsregel zu implementieren was z.B. ein Wahrscheinlichkeitswert sein kann wie z.B. bei der Klasse `UniformDecisionMechanism`. Der `DecisionMaker` implementiert dann die Auswahl einer Produktionsregel anhand ihrer Bewertungen aus der `DecisionMechanism`-Schnittstelle. So ist beispielsweise ein `DecisionMaker` enthalten, welcher die Entscheidung probabilistisch auf Basis einer uniformen Verteilung realisiert.

Die in Abbildung 5 ebenfalls zu sehende `ProductionDecision` schließlich registriert, welche Produktionen bereits ausgeführt wurden und erlaubt somit die Anwendung alternativer Produktionen.

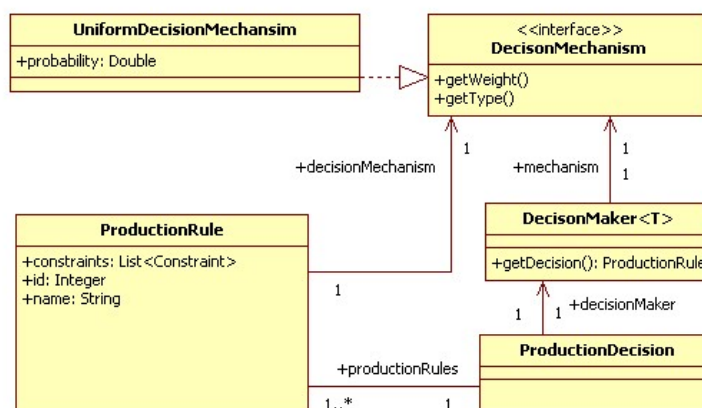


Abbildung 5: Aufbau des verwendeten Entscheidungsprozesses

Um alternative Entwicklungen im Ableitungsprozess zu testen, ist es möglich im Ableitungsbaum ausgeführte Produktionen zurückzunehmen und ab dem dann erreichten Stand neu in den Produkti-

onsprozess einzutreten.

Ergebnis eines Generierungsprozesses sind der Ableitungsbaum, welcher den Prozess der Generierung vollständig abbildet und das generierte Wort der Sprache. Beide können als XML-Datei exportiert werden. Die Definition dieser Exportformate ist ebenfalls als XML-Schema verfügbar.

Wie einführend erläutert, eignen sich Modelle, die durch Grammatiken beschrieben sind, auch für die semantische und geometrische Rekonstruktion. Schließlich beschreiben grammatikbasierte Modelle die Realwelt ebenso wie Messungen (Bilder oder Laserscans). Der  $XG^{ep}$ -Prozessor kann wie von Schmittwilken et al. (2009b) beschrieben zur Rekonstruktion verwendet werden. Unter Einbeziehen einer Grammatik ist Ziel der semantischen und geometrischen Rekonstruktion das Finden des Ableitungsbaums, der die Beobachtungen am besten erklärt. Demnach müssen Teile der Beobachtungen als Symbole der Grammatik erkannt und die entsprechenden Produktionsregeln und der zugehörige Ableitungsbaum gefunden werden.

## 5 Geometrischer Instanzierer

Ein Wort der durch die Grammatik definierten Sprache, kann mit Hilfe des geometrischen Instanzierers in visualisierbare 3D-Modelle überführt werden. Bei einer entsprechend entworfenen Grammatik verfügen auch die inneren Knoten des Ableitungsbaums über eine geometrische Repräsentation. Beispielsweise Split-Grammatiken verfeinern sämtliche Strukturen innerhalb des von ihnen definierten räumlichen Bereichs. Somit können mehrere hierarchische Detaillierungsgrade (Level of Detail) für das gleiche Objekt mit nur einem Ableitungsbaum erzeugt werden.

Die Instanzierung erfolgt durch die Zuordnung prototypischer, parametrisierter Geometrie-Objekte in Boundary Representation (B-Rep) zu den Symbolen der Grammatik. Die jeweilige Ausprägung ist durch die entsprechenden Attribute für Lage und Form festgelegt. Die Erzeugung einer geometrischen Instanz eines Wortes erfolgt zweistufig. In einem ersten Schritt wird die innere Geometrie des Objekts durch die Form-Parameter bestimmt und in einem zweiten Schritt das Objekt durch Anwendung der Transformationsmatrix an die Position im Raum transformiert. Nach Durchführung des Verfahrens für alle zu instanzierenden Symbole des Ableitungsbaums, ergibt sich die 3D-Szene durch Aggregation der Einzelgeometrien.

Dafür benötigte Geometrie-Prototypen für Standard-2D- und 3D-Geometrien, wie beispielsweise Rechtecke oder Quader, sind vorgegeben und in einer s.g. *geom-Datei* gespeichert. Neue Geometrien können aufbauend auf diesen Standardprototypen erzeugt und an die aktuelle Grammatik angepasst werden.

### 5.1 geom-Datei

Zur flexiblen Gestaltung und Bearbeitung von Geometrien wurde die geom-Datei konzipiert (Abb. 6). Sie besteht aus einem ZIP-Container, der die einzelnen Prototyp-Dateien (z.B. *PentRoof.xml*) und eine Zuordnungsdatei (*master.xml*) enthält.

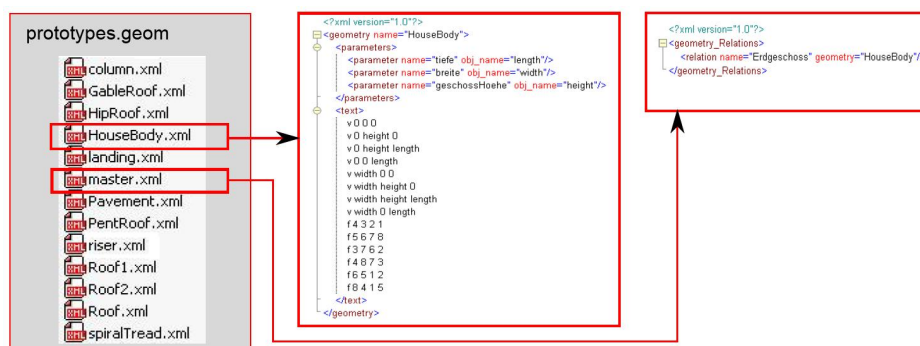


Abbildung 6: geom-File: Speicherformat für Geometrien

Die Zuordnung einer Geometrie zu einem Symbol geschieht in zwei Stufen. (1) In der Zuordnungsdatei (`master.xml`) werden mindestens allen Terminalsymbolen der Grammatik Geometrie-Prototypen zugeordnet - wenn möglich auch den Nichtterminalsymbolen. (2) In der Prototyp-Datei werden die Parameter des Geometrie-Prototypen den jeweiligen Symbolattributen zugeordnet. Verwenden mehrere Symbole den gleichen Geometrie-Prototypen, so müssen alle Attribute, die einem Prototypen-Parameter zugeordnet werden, den gleichen Bezeichner haben. Die Prototyp-Dateien enthalten neben der Attribut-Parameter-Zuordnung auch die Beschreibung der Geometrie. Hierzu wird eine Spezifikation ähnlich dem OBJ-Format verwendet. Zusätzlich können in den Prototyp-Dateien auch Prototyp-spezifische Labels definiert werden, die Informationen wie Farben oder Zugehörigkeiten zu Geometrie-gruppen tragen.

Für diese Zuordnung der Symbole und Attribute zu den Geometrie-Prototypen gibt es einen eigenen Reiter in der grafischen Benutzeroberfläche von  $XG^{ep}$ . Dort ist auch die Erstellung neuer oder Abwandlung alter Geometrien möglich. Des Weiteren kann in der grafischen Benutzeroberfläche auch die Kombination verschiedener geom-Dateien oder Bildung von Untermengen (z.B. nur von der aktiven Grammatik verwendete Geometrien) und Abspeicherung in einer neuen geom-Datei vorgenommen werden. Da es sich um Standard-Komponenten (ZIP-Container und XML-Dateien) handelt, ist natürlich auch eine manuelle Bearbeitung der geom-Dateien in externen Programmen möglich.

## 5.2 Exportformate

Durch die Erstellung eines Format-unabhängigen Zwischenergebnisses der Visualisierung, wird die Implementation des Exports für ein beliebiges 3D-Format erleichtert. Wie in Abbildung 7 gezeigt, wird aus dem geom-File und einem Ableitungsbaum in einem ersten Schritt eine **Instance** erzeugt. Diese enthält alle ebenen Polygone (**Ring**) mit den angefügten Labeln. In einem zweiten Schritt wird aus dieser **Instance** ein 3D-Modell der gewünschten Modellierungssprache generiert. Aktuell sind die Exportformate OBJ, KML, VRML und X3D realisiert; CityGML mit einer aus dem Ableitungsbaum entnommenen Semantik ist geplant bzw. prototypisch implementiert.

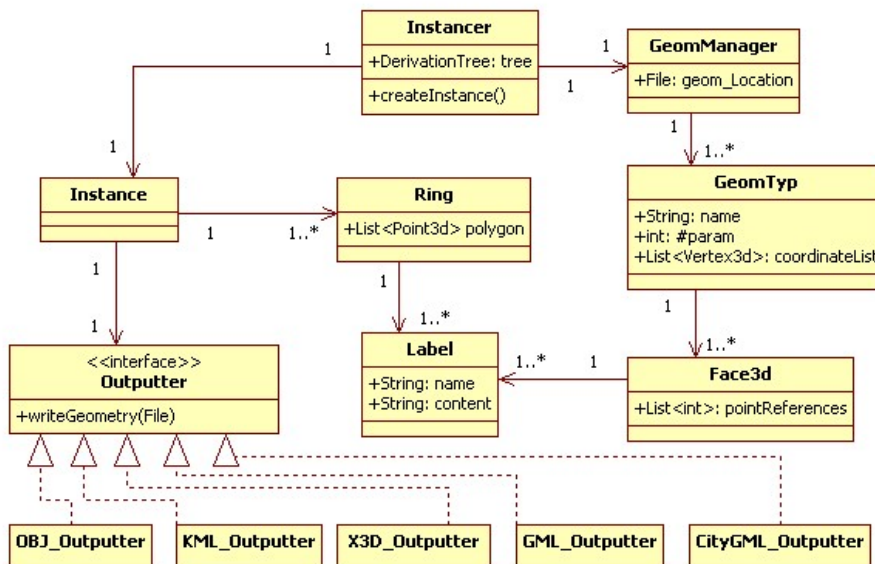


Abbildung 7: Geometrischer Instanzierer

## 6 Resumee

Die Implementierung von  $XG^{ep}$  in Java ist fast abgeschlossen und die Veröffentlichung des Quellcodes noch in diesem Jahr geplant. Das Projekt umfasst die Module Prozessierer (Kapitel 4) und Instanzierer (Kapitel 5) sowie ein Editier-Modul zur Erstellung, Änderung und Prüfung von Grammatiken im  $XG^{ep}$  eigenen XML-Format. Zu den einzelnen Modulen existieren graphische Benutzeroberflächen.

Mit  $XG^{ep}$  steht ein stark modularisiertes, plattformunabhängiges und frei erweiterbares Werkzeug zum Umgang mit räumlichen Grammatiken bereit. Die Erweiterbarkeit erstreckt sich sowohl auf Aspekte der Prozessierung wie z.B. der Entscheidungsmechanismus, als auch auf die Erzeugung geometrischer Instanzen von Wörtern und Ableitungsbäumen.

Die Leistungsfähigkeit von Grammatiken zur Modellierung von Gebäuden und ihrer Bestandteile, aber auch von  $XG^{ep}$  zum Umgang mit diesen Grammatiken soll durch Abbildung 8 unterstrichen werden. Die Abbildung zeigt die geometrische Instanz eines mit einer Grammatik für komplexe Gebäude generierten Wortes in unterschiedlichen Ansichten. Diese Grammatik generiert I-, L-, U- oder O-förmige Gebäudegrundrisse mit unterschiedlich hohen Gebäudeteilen und platziert Fenster und Türen entsprechend der Stockwerke auf der Fassade. Bei genauer Betrachtung der Szene fällt auf, dass im Schnittbereich der Gebäudeteile keine Fenster generiert wurden. Auch die Form der Fenster variiert zwischen den einzelnen Fassaden.

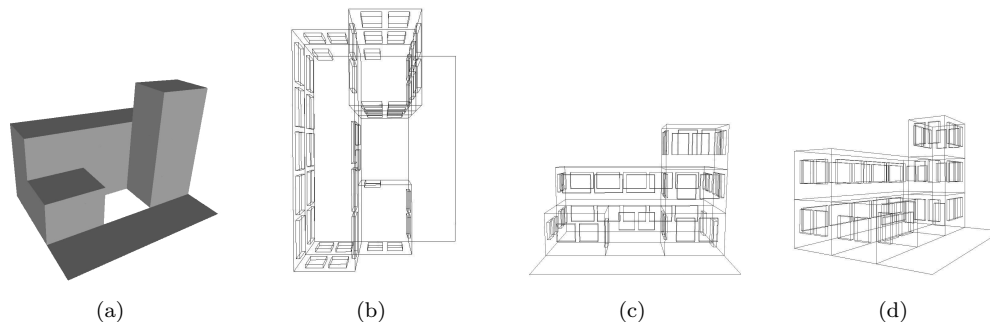


Abbildung 8: Geometrieinstanz eines Wortes der Gebäudegrammatik

Die aktuellen Arbeiten an  $XG^{ep}$  befassen sich vor allem mit der Umsetzung von Parsing-Funktionalität und der Integration von weiteren Attributdatentypen und flexibleren semantischen Regeln um z.B. auch Graphen o.ä. in Attributen zu verwalten.

### Hinweise

Weitere Informationen zu  $XG^{ep}$  sind im Internet unter <http://www.ikg.uni-bonn.de/forschung/xgep.html> verfügbar. Dort sind auch die hier verwendeten Beispiele im Grammatik-XML- und X3D Format zu finden.

Die Autoren danken Martin Krückhans für seine Mitarbeit am  $XG^{ep}$ -Projekt.

E-Mail-Adressen der Autoren: {behmann, doerschlag, schmittwilken}@igg.uni-bonn.de

### Literatur

Becker, Susanne: Generation and application of rules for quality dependent facade reconstruction. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Band 64(6):S. 640 – 653, 2009. ISSN 0924-2716.

Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve und Yergeau, Francois (Hg.): *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C, 2008.

- Chomsky, Noam: Three models for the description of language. In: *Information Theory, IEEE Transactions*, Band 2(3):S. 113–124, 1956.
- Clark, James (Hg.): *XSL Transformations (XSLT) Version 1.0*. W3C, 1999.
- Fallside, David C. und Walmsley, Priscilla (Hg.): *XML Schema 1.0*. W3C, 2004.
- Geman, S. und Johnson, M.: Probabilistic grammars and their applications. In: *International Encyclopedia of the Social & Behavioral Sciences*, Band 2002:S. 12075–12082, 2002.
- Gröger, Gerhard, Kolbe, Thomas H., Czerwinski, Angela und Nagel, Claus (Hg.): *OpenGIS City Geography Markup Language (CityGML)*. Open Geospatial Consortium, 2008.
- Knight, Terry W.: Shape grammars: six types. In: *Environment and Planning B: Planning and Design*, Band 26:S. 15–31, 1999.
- Knuth, Donald E.: Semantics of context-free languages. In: *Theory of Computing Systems*, Band 2(2):S. 127–145, 1968.
- Krückhans, Martin und Schmittwilken, Jörg: Synthetische Verfeinerung von 3d-Stadtmodellen. In: *Proceedings of DGPF-Jahrestagung. Jena 2009*. 2009.
- Müller, Pascal, Wonka, Peter, Haegler, Simon, Ulmer, Andreas und Gool, Luc Van: Procedural modeling of buildings. In: *ACM Transactions on Graphics*, Band 25(3):S. 614–623, 2006.
- Prusinkiewicz, Przemyslaw und Lindenmayer, Aristid: *The algorithmic beauty of plants*. Springer-Verlag New York, Inc., New York, NY, USA, 1991. ISBN 0-387-94676-4.
- Pu, S. und Vosselman, G.: Automatic extraction of building features from terrestrial laserscanning. In: Maas, H.-G. und Schneider, D. (Hg.) *International Archives Of Photogrammetry, Remote Sensing And Spatial Information Sciences, Vol. 36, part 5, Dresden, Germany, September 25-27*. 2006.
- Ripperda, Nora: Grammar based facade reconstruction using rjmcmm. In: *Photogrammetrie, Fernerkundung, Geoinformation*, Band 2:S. 83–92, 2008.
- Schmittwilken, Jörg, Dörschlag, Dirk und Plümer, Lutz: Attribute grammar for 3d city models. In: Krek, A., Rumor, M., Zlatanova, S. und Fendel, E. M. (Hg.) *Proceedings of the Urban and Regional Data Management Society Symposium 2009, Ljubljana, Slovenia, 24-26 June 2009*. UDMS, CRC Press, 2009a, S. 49–58.
- Schmittwilken, Jörg, Saatkamp, Jens, Förstner, Wolfgang, Kolbe, Thomas H. und Plümer, Lutz: A semantic model of stairs in building collars. In: *Photogrammetrie, Fernerkundung, Geoinformation*, Band 2007(6):S. 415–427, 2007.
- Schmittwilken, Jörg, Yang, Michael Ying, Förstner, Wolfgang und Plümer, Lutz: Integration of conditional random fields and attribute grammars for range data interpretation of man-made objects. In: *Annals of GIS*, Band 15:S. 117–126, 2009b.
- Stiny, G. und Gips, J.: Shape grammars and the generative specification of painting and sculpture. In: Freiman, C.V. und Griffith, John E. (Hg.) *Proceedings of IFIP Congress 71, Volume 2 - Applications, Ljubljana, Yugoslavia, August 23-28, 1971*. 1972, S. 1460–1465.