Objektkatalog für das Straßen- und Verkehrswesen

Ableitung von OKSTRA®-XML aus der EXPRESS-Modellierung

Version: 1.1

Datum: 19.01.2011

Status: akzeptiert

Dateiname: T0006.doc

Pfad: n.a.

Verantwortlich: Jochen Hettwer / Clemens Portele

OKSTRA®-Pflegestelle

interactive instruments GmbH
Trierer Straße 70-72
53115 Bonn

Herr Bernd Weidner
Tel. 0228 91410 74
Fax 0228 91410 90
Email weidner@interactive-instruments.de

Im Auftrag von

Bundesanstalt für Straßenwesen
ZD - OKSTRA
Brüderstraße 53
51427 Bergisch Gladbach

Herr Alfred Stein
Tel. 02204 43 354
Fax 02204 43 673
Email stein@bast.de

0 Allgemeines

0.1 Inhaltsverzeichnis

0.2 Abkürzungen und Definitionen

siehe Glossar in Kapitel 0

0.3 Abbildungsverzeichnis

0.4 Tabellenverzeichnis

0.5 Bezüge

Dokument	Bemerkungen
XML	Recommendation 1.0, Third Edition, W3C, 04 February 2004 Hyperlink: http://www.w3.org/TR/2004/REC-xml-20040204/
XML Schema	Recommendation, Second Edition, W3C, 28 October 2004 Hyperlinks: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/ Informativ: http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/
GeoInfoDok (u.a. AFIS®-ALKIS®-ATKIS®, NAS)	Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok) Hyperlink: http://www.adv-online.de/; dort in der Navigationsleiste auf der linken Seite den Punkt "Veröffentlichungen" wählen, dann "AFIS-ALKIS-ATKIS Projekt" und dann "GeoInfoDok 4.0"
GML 3.00	Open GIS Consortium, 2003 – Implementation Specification Hyperlink: http://www.opengis.org/techno/documents/02-023r4.doc
GML 3.1.1	Open GIS Consortium, 2005 – GML, Version 3.1.1 Hyperlink: http://schemas.opengis.net/gml/3.1.1/

0.6 Änderungen

Name	Datum	Kapitel	Bemerkungen	Bearbeiter
N0028	11.01.2002	alle	Dokument auf Basis von N0023 erstellt	Dietmar König
N0028	23.04.2002	alle	Dokument fortgeführt zum ersten Abstimmungsentwurf	Dietmar König / Clemens Portele
N0028	06.06.2002	alle	Dokument fortgeführt zur Veröffentlichung	Dietmar König
N0028	28.04.2003	alle	Dokument aktualisiert gemäß Fortführung der XML Schemata nach ersten praktischen Erfahrungen und Anpassung an GML3	Dietmar König / Clemens Portele
T0006	23.05.2003	alle	Übernahme als technisches Dokument	Dietmar König
T0006	02.09.2003	alle	Fortführung zur OKSTRA®-Version 1.008	Dietmar König
T0006	25.05.2003	4.7.2 4.5.2 4.8.2	Abbildung mehrfacher Netzbezüge durch Vererbung, Abbildung inverser Relationen Anpassung der Koordinatenreferenzsysteme	Dietmar König
T0006	29.06.2004	4.7.2 4.5.2 4.8.2 4.8.3	Fortführung zur OKSTRA®-Version 1.009	Dietmar König
T0006	03.08.2005		Fortführung zur OKSTRA®-Version 1.010	Dietmar König
T0006	22.06.2006		Fortführung zur OKSTRA®-Version 1.011	Jochen Hettwer
T0006	16.08.2007		Fortführung zur OKSTRA®-Version 1.012	Jochen Hettwer
T0006	23.10.2008	4.9	Fortführung zur OKSTRA®-Version 1.013	Jochen Hettwer
T0006	02.12.2010	4.1 4.6.2 4.7 4.8.7 4.9	Fortführung zur OKSTRA®-Version 1.015	Jochen Hettwer

0.7 Bearbeitungsvermerke

- keine

1 Zielsetzung

XML gewinnt zunehmende Bedeutung als lingua franca bei der plattformunabhängigen Repräsentierung, Bereitstellung und Übertragung von Informationen. Dies gilt für alle Anwendungsbereiche der IT – auch für das Straßen- und Verkehrswesen.

Auch aus den Reihen der OKSTRA®-Nutzer wird der Ruf nach XML lauter. In diesem Konzept werden die Rahmenbedingungen und Designentscheidungen erläutert, die bei der Definition von OKSTRA®-XML zugrunde gelegt wurden.

Anmerkung: Ein kleines Glossar zu den verwendeten Begriffen findet sich am Ende des Dokuments (siehe 0). Wir empfehlen, sich die Begriffe in diesem Glossar anzueignen, um Missverständnisse zu vermeiden.

2 Bedeutung von XML für den OKSTRA®

XML bietet die Möglichkeit, Informationen strukturiert darzustellen. Es bestehen weitreichende Möglichkeiten, XML-Daten untereinander zu verknüpfen. In diesem Sinne wird XML für den OKSTRA® als

Format zur strukturierten, vernetzten Darstellung von OKSTRA®-Daten

verstanden. Die Referenzmodellierung des OKSTRA® bleibt weiterhin das EXPRESS-Schema. Neben dem nativen Austauschformat, OKSTRA®-CTE, wird XML als Alternative hinzugenommen. Besonders für die Bereitstellung von Daten auf Servern und die Vernetzung von Daten bietet XML hier Vorteile.

Wichtig: "OKSTRA®-XML" ist keine konzeptionelle Neu-Modellierung des OKSTRA®, sondern eine XML-basierte Umsetzung der existierenden EXPRESS-Schemata. Dies liefert ein geeignetes Format zur Repräsentierung, Bereitstellung und Übertragung von OKSTRA®-Daten in XML.

Zusätzlich zu den erweiterten Möglichkeiten der Vernetzung von OKSTRA®-Daten, die eine XML-basierte Beschreibung gegenüber den derzeitigen Beschreibungen bietet, ergibt sich auch ein strategischer Vorteil für den OKSTRA®. XML besitzt bereits heute eine hohe Verbreitung, die sich auch in der Verfügbarkeit von Standardsoftware ausdrückt. Aller Voraussicht nach wird diese Verbreitung zukünftig noch steigen. Ganz konkret basiert auch das neue, durch die AdV definierte, Austauschformat NAS der Vermessungsverwaltung auf XML. Ein Datenaustausch mit der Landesvermessung wird durch die Verwendung von XML erleichtert.

XML-basierte Beschreibungen werden heute sowohl zur Repräsentierung von Informationen wie zur Übertragung von Informationen verwendet. Der geplante Einsatz hat maßgeblichen Einfluss auf die Gestaltung der zugehörigen XML Schemata[1].

[1] Die Abbildung erfolgt bewusst nach XML Schema und nicht als DTD, da davon auszugehen ist, dass die Bedeutung von XML Schema weiter steigen wird. Darüber hinaus sind die zur Verfügung stehenden Sprachmittel einer Umsetzung des OKSTRA® deutlich angemessener.

3 Grundsätze

OKSTRA®-XML wird in Form von XML Schemata definiert. Diese XML Schemata liefern die Strukturvorgaben für OKSTRA®-Daten in XML.

Stark vereinfachend kann man sagen, dass die Beziehung von XML Schema zu XML-Daten der Beziehung von EXPRESS zu CTE entspricht. Das eine gibt ein konzeptionelles Schema vor, das andere bezeichnet konkrete Daten in dem vorgegebenen Format. Zu unterscheiden ist daher zwischen der Strukturvorgabe als XML Schema und dem resultierenden `Content Model` von XML-Daten, d.h. der Gestalt der OKSTRA®-XML-Daten, die sich aus den OKSTRA®-XML Schemata ergibt. Das `Content Model` wird in diesem Konzept verbindlich festgelegt. Für die XML Schemata, die XML-Daten mit diesem `Content Model` beschreiben, existieren verschiedene Möglichkeiten.

Es werden in diesem Konzept die Regeln festgelegt, wie sich aus der Referenzmodellierung des OKSTRA® in EXPRESS das `Content Model` von OKSTRA®-XML ableitet und wie ein entsprechendes XML Schema zum OKSTRA® abgeleitet wird.

Wie auch bei der EXPRESS-Modellierung sollten für die Verwendung von XML im OKSTRA® komplett ausformulierte XML Schemata angegeben werden. Dabei ist analog zu EXPRESS eine Gruppierung fachlich zusammengehöriger Objektklassen in verschiedene Schemata zu berücksichtigen.

Um die breite Nutzbarkeit von OKSTRA®-XML zu gewährleisten, sind die Entwicklungen im Umfeld zu berücksichtigen. Von besonders hoher Bedeutung sehen wir hier die NAS der Vermessungsverwaltung (AFIS®, ALKIS®, ATKIS®) und die Standards für im Aufbau befindliche Geodateninfrastrukturen (i.d.R. basierend auf Standards des Open GIS Consortiums). Hierbei werden Objektinformationen in GML codiert. Bei der Geography Markup Language (GML) handelt es sich um eine wegweisende Spezifikation des Open GIS Consortiums zur Codierung von Objekten in XML mit einer besonderen Unterstützung für raumbezogene Eigenschaften und eine verteilte Datenhaltung. Die NAS wird selbst als GML Anwendungsschema modelliert. Dies spielt bei der Abbildung der Geometrie eine wesentliche Rolle (siehe 4.8). GML 3.1.1 ist darüber hinaus identisch mit ISO 19136.

Die Verwendung von GML auch im OKSTRA® hat die folgenden Vorteile:

- Standardmechanismen wie die Beschreibung von Geometrie können übernommen werden und müssen nicht neu und OKSTRA®-spezifisch festgelegt werden
- Erleichterung des Datenaustausches mit der Vermessungsverwaltung
- Einfachere Integration von OKSTRA®-Daten in Geodateninfrastrukturen

Im OKSTRA® wird aus dem Anwendungsschema in EXPRESS das Austauschformat abgeleitet. Bei der NAS arbeitet man ganz ähnlich: Hier wird aus den statischen Klassendiagrammen eines UML-Anwendungsschemas die NAS als XML Schema automatisch abgeleitet. Ebenso kann aus der EXPRESS-Modellierung ein GML-Anwendungsschema erzeugt werden. Für die hierzu erforderlichen Regeln haben wir uns an den „NAS Encoding Rules" orientiert.

Für die Verwendung im OKSTRA® wird ein Profil von GML 3.1.1 definiert, das nur die im Rahmen von OKSTRA®-XML verwendeten Konstrukte von GML enthält.

3.1 Allgemeine Hinweise zum Verständnis des Dokuments

XML-Begriffe werden im Zeichensatz `Courier New` dargestellt.

Die **Beispiele** sind je nach der verwendeten Sprache farblich unterlegt:

. EXPRESS-CTE XML-Daten EXPRESS XML Schema .

In eckige Klammern eingefasste Ausdrücke [Ausdruck] sind Platzhalter. Der Ausdruck wird im konkreten Fall ausgewertet. Also z.B. [ENTITY-Name] ist durch den entsprechenden Namen des ENTITYs zu ersetzen. Optionale Angaben sind in { } eingefasst.

Wichtiger Hinweis: Die verwendeten Beispiele aus der EXPRESS-Modellierung sind an vielen Stellen für die Zwecke des Beispiels reduziert bzw. angepasst. Sie dienen jeweils der Illustration einer bestimmten Idee und nicht der Wiedergabe der exakten OKSTRA®-Modellierung.

4 Regeln für OKSTRA®-XML

4.1 OKSTRA®-Versionen

Zu jeder OKSTRA®-Version werden OKSTRA®-XML Schemata definiert. Diese XML Schemata werden auf den OKSTRA®-Webseiten unter

`http://www.okstra.de/schema/xyyyz/okstra.xsd`

zur Verfügung gestellt. Dabei bezeichnet in dem Ausdruck `xyyyz` das `x` die Hauptversion des OKSTRA® (bisher stets 1) und `yyy` die dreistellige Unterversion des OKSTRA® (z.B. 015). Das `z` dient zur Bezeichnung einer ggf. im Rahmen einer bestimmten OKSTRA®-Version erstellten Entwicklungsversion (z.B. 1). Wird kein Wert für z angegeben, ist die ursprünglich veröffentlichte Unterversion des OKSTRA® gemeint. Die Unterversion 1.015 des OKSTRA® liegt auf den OKSTRA®-Webseiten beispielsweise unter

`http://www.okstra.de/schema/1015/okstra.xsd.`

Falls es eine Arbeitsversion 1.015.1 zur Unterversion 1.015 gäbe, läge sie unter

`http://www.okstra.de/schema/10151/okstra.xsd.`

Sofern an einem XML Schema Fehlerkorrekturen nach der Veröffentlichung des Schemas durchgeführt werden müssen, wird das korrigierte Schema unter

`http://www.okstra.de/schema/xyyyzkorrn/okstra.xsd`

abgelegt, wobei das `n` hinter dem Ausdruck `korr` die Nummer der Korrekturversion zu einem bestimmten Schema bezeichnet. Falls es beispielsweise eine zweite Korrekturversion zur OKSTRA®-Version 1.015 gäbe, läge sie unter

`http://www.okstra.de/schema/1015korr2/okstra.xsd.`

Innerhalb des jeweiligen OKSTRA®-XML Schemas wird der `targetNamespace` stets `okstra` genannt, unabhängig von der Version. Der Beginn eines OKSTRA®-XML Schemas könnte also z.B. folgendermaßen aussehen:

XML Schema:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
    <!--  File: okstra.xsd  -->
    <schema targetNamespace="http://schema.okstra.de/1015/okstra"
```

```
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:okstra="http://schema.okstra.de/1015/okstra"
        xmlns:xlink="http://www.w3.org/1999/xlink"
        xmlns:gml="http://www.opengis.net/gml"
        elementFormDefault="qualified"
        version="1.015">
        <annotation>
            <appinfo>1015/okstra.xsd</appinfo>
            <documentation xml:lang="de">
                zentrales Schema zum Include von Schemata in OKSTRA(R)
                XML 1.015
            </documentation>
        </annotation>
        <include schemaLocation="Strassennetz.xsd" />
        <include schemaLocation="Administration.xsd" />
        [...]
        <include schemaLocation="Entwurf.xsd" />
        [...]
    </schema>
```

Anmerkung: Innerhalb einer OKSTRA®-XML-Datei muss exakt eine Version von OKSTRA®-XML verwendet werden. Eine Vermischung der Versionen ist nicht zulässig.

4.1.1 Organisation in XML Schemata

Über `okstra.xsd` in der jeweiligen Version erhält man Zugriff auf die gesamten XML Schema-Definitionen zu der gewählten Version des OKSTRA®. Benötigt man nur Teile des OKSTRA®, z.B. einzelne Schemata, so ist es aus Performancegründen sinnvoll, nur die entsprechenden Teile zu laden. Zu diesem Zweck werden die OKSTRA®-XML-Definitionen in mehrere Schemata gegliedert:

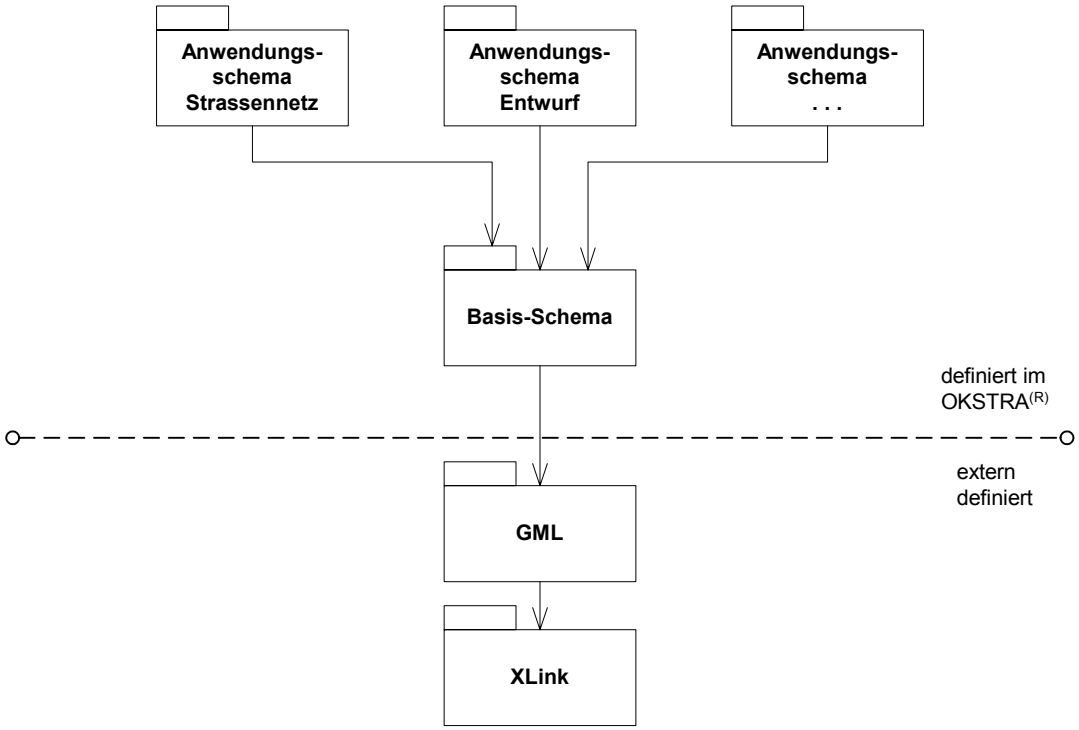

Abbildung 1 – Organisation von OKSTRA®-XML in XML Schemata

Das Basis-Schema `okstra_basis.xsd` definiert allgemein verwendbare Datentypen (zu diesem Zweck werden dort auch die Dateien `okstra_typen.xsd`, `okstra_konz_typen.xsd` und `okstra_schluesseltabellen.xsd` eingebunden). Insbesondere definiert das Basis-Schema den `OKSTRAObjektmengeType`, der dem `root element` jeder OKSTRA®-XML-Datei zugrunde liegt, sowie das `root element` „`OKSTRAObjektmenge`" selbst. Ein Anwendungsschema entspricht vom fachlichen Umfang her i.w. einem EXPRESS-Schema des OKSTRA®. Jedes Anwendungsschema bezieht das Basis-Schema ein[2]. Das Basis-Schema seinerseits importiert die erforderlichen Schemata aus dem XML- und GML-Bereich.

Im Basis-Schema werden beispielsweise für die OKSTRA®-Version 1.015 folgende Definitionen aufgeführt (Auszug):

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!-- File: okstra_basis.xsd -->
<schema targetNamespace="http://schema.okstra.de/1015/okstra"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:okstra="http://schema.okstra.de/1015/okstra"
    elementFormDefault="qualified" version="1.015">
```

[2] Importiert der Benutzer mehr als ein Anwendungsschema, so wird das Basis-Schema zwangsläufig mehrfach importiert. Dieses ist zulässig, jedoch stilistisch unschön. Ein Ausweg daraus wäre ein Customizer, d.h. ein Tool, das nach den Anforderungen des Benutzers die erforderlichen XML Schema-Definitionen in einem individuellen XML Schema zusammenstellt.

```
    <annotation>
        <appinfo source="urn:okstra:okstra-basis:v1.015">
            okstra_basis.xsd v1.015</appinfo>
        <documentation xml:lang="de">
            GML-Anwendungsschema fuer OKSTRA(R) 1.015, Basis-Schema
        </documentation>
    </annotation>
    <!--  Type-Definitionen und externe Schemata einbeziehen  -->
    <include schemaLocation="okstra_typen.xsd"/>
    <include schemaLocation="okstra_schluesseltabellen.xsd"/>
    <include schemaLocation="okstra_konz_typen.xsd"/>

    [...]

    <!-- -->
    <!-- ========================================== -->
    <!-- = zentrale Zeiger- und Mengen-Definitionen = -->
    <!-- ========================================== -->
    <!-- -->
    <!-- Root-Element fuer OKSTRA-FeatureCollection -->
    <element name="OKSTRAObjektmenge" type="okstra:OKSTRAObjektmengeType"
        substitutionGroup="gml:_FeatureCollection"/>
    <!-- definiere OKSTRAObjektmengeType -->
    <complexType name="OKSTRAPropertyType">
        <complexContent>
            <extension base="gml:FeaturePropertyType">
                <sequence/>
                <attribute name="Objektklasse" type="string"
                    use="optional"/>
            </extension>
        </complexContent>
    </complexType>
    <element name="okstraObjekt" type="okstra:OKSTRAPropertyType"
        substitutionGroup="gml:featureMember"/>
    <complexType name="OKSTRAObjektmengeType">
        <complexContent>
                <extension base="gml:AbstractFeatureCollectionType"/>
        </complexContent>
    </complexType>
    <!-- Meta-Daten fuer OKSTRA-Daten -->
    <element name="OKSTRAMetaDaten" type="okstra:OKSTRAMetaDatenType"
```

```
        substitutionGroup="gml:_MetaData"/>
    <complexType name="OKSTRAMetaDatenType" mixed="true">
        <complexContent>
            <extension base="gml:AbstractMetaDataType">
                <sequence>
                    <element name="description" type="string"
                        minOccurs="0" maxOccurs="unbounded"/>
                    <element name="implementation_level" type="string"
                        minOccurs="0"/>
                    <element name="name" type="string" minOccurs="0"/>
                    <element name="time_stamp" type="dateTime"
                        minOccurs="0"/>
                    <element name="author" type="string" minOccurs="0"
                        maxOccurs="unbounded"/>
                    <element name="organization" type="string"
                        minOccurs="0" maxOccurs="unbounded"/>
                    <element name="preprocessor_version" type="string"
                        minOccurs="0"/>
                    <element name="originating_system" type="string"
                        minOccurs="0"/>
                    <element name="authorization" type="string"
                        minOccurs="0"/>
                    <element name="schema_identifiers" type="string"
                        maxOccurs="unbounded"/>
                    <element name="relRep" type="okstra:RelRepType"
                        minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="RelRepType">
        <restriction base="string">
            <enumeration value="einseitig"/>
            <enumeration value="beidseitig"/>
        </restriction>
    </simpleType>
    <!-- Referenzierungs-Typ fuer OKSTRA-Objekte -->
    <complexType name="ObjectRefType">
        <simpleContent>
            <extension base="string">
                <attributeGroup ref="gml:AssociationAttributeGroup"/>
```

```
                <attribute name="Objektklasse" type="string"
                    use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
    <!-- complexType fuer OKSTRA-Schluesseltabellen -->
    <complexType name="KeyValuePropertyType">
        <attributeGroup ref="xlink:simpleLink"/>
        <attribute name="Objektklasse" type="string" use="optional"/>
        <attribute name="Kennung" type="string" use="optional"/>
    </complexType>

    [...]

</schema>
```

4.1.2 OKSTRAObjektmenge

Jede OKSTRA®-XML-Datei enthält als `root element` eine `OKSTRAObjektmenge`. Diese enthält:

- `gml:metaDataProperty`: Diese `property` enthält ein `element OKSTRAMetaDaten`, in dem mindestens die zugrundeliegende OKSTRA®-Version angegeben ist.
- `gml:boundedBy`: Ein einschließendes Rechteck für die gesamte Datei sollte hier angegeben werden.
- Beliebig viele `elements okstraObjekt`: Diese enthalten entweder ein `element` zu einem OKSTRA®-Objekt, z.B. einen Netzknoten, oder einen Verweis in Form eines `xlink:href`[3]. Im Falle des Verweises kann zur Unterstützung im `attribute Objektklasse` der Klassenname des Objekts angegeben werden.
- Beliebig viele `elements okstraKeyValue`: Dies ist die XML-Darstellung für Schlüsseltabellen des OKSTRA®.

Beispiel:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<okstra:OKSTRAObjektmenge
    xmlns:okstra="http://schema.okstra.de/1015/okstra"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

[3] Hier wird bewusst kein `string content` zugelassen wie bei der derzeit möglichen Darstellung von Relationen durch Angabe der fachlichen Kennung als String. Im Hinblick auf eine Vereinheitlichung der Darstellung und im Licht der derzeit geführten Diskussionen zur Objekt-Id sollte zukünftig die Angabe von Relationspartnern ebenfalls einheitlich über den `xlink:href` geregelt werden.

```
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xsi:schemaLocation="http://schema.okstra.de/1015/okstra
                        http://www.okstra.de/schema/1015/okstra.xsd">
    <gml:metaDataProperty>
        <okstra:OKSTRAMetaDaten>
            <okstra:implementation_level>1</okstra:implementation_level>
            <okstra:name>Beispiel.xml</okstra:name>
            <okstra:time_stamp>2010-12-02T10:45:27</okstra:time_stamp>
            <okstra:preprocessor_version>XTRA
            </okstra:preprocessor_version>
            <okstra:schema_identifiers>OKSTRA 1.015
            </okstra:schema_identifiers>
            <okstra:relRep>einseitig</okstra:relRep>
        </okstra:OKSTRAMetaDaten>
    </gml:metaDataProperty>
    <gml:boundedBy>
        <gml:Envelope>
            <gml:coordinates>
                35661201.,562303868. 36435196.,563977511.
            </gml:coordinates>
        </gml:Envelope>
    </gml:boundedBy>
    <okstra:okstraObjekt xlink:href="urn:asb:A555"
                         Objektklasse="Strasse"/>
    <okstra:okstraObjekt>
        <okstra:Nullpunktort gml:id="Nullpunktort.50">
            <okstra:bei_Strassenpunkt>
                <okstra:Strassenpunkt>
                    <okstra:Station>0.</okstra:Station>
                    <okstra:auf_Abschnitt_oder_Ast Objektklasse="Ast"
                                                   xlink:href="#Ast.19"/>
                </okstra:Strassenpunkt>
            </okstra:bei_Strassenpunkt>
            <okstra:bei_Nullpunkt Objektklasse="Nullpunkt"
                                                   xlink:href="#Nullpunkt.19"/>
        </okstra:Nullpunktort>
    </okstra:okstraObjekt>
    […]
</okstra:OKSTRAObjektmenge>
```

4.1.3 Metadaten

Für die `OKSTRAObjektmenge` werden Metadaten analog zu den Festlegungen für den Header einer CTE-Datei definiert (siehe Dokument T0005 der OKSTRA®-Pflege). Diese Metadaten sind im `complexType OKSTRAMetaDatenType` beschrieben.

In den Metadaten muss zwingend die verwendete OKSTRA®-Version als `element schema_identifiers` angegeben werden. Das Format für die Angabe ist

OKSTRA x.yyy

wobei x die Hauptversion des OKSTRA® bezeichnet (bisher immer 1) und yyy die Unternummer der OKSTRA®-Version, z.B. 015. Die OKSTRA®-Version 1.015 würde als

OKSTRA 1.015

codiert.

Ein kleines Beispiel ist oben im Beispiel für die `OKSTRAObjektmenge` enthalten.

4.1.4 Schema-Location

Im `root element` jeder OKSTRA®-XML-Datei muss zwingend ein Ort angegeben werden, an dem die zugehörigen OKSTRA®-XML-Schemata zu finden sind. Dies geschieht im `attribute`

`xsi:schemaLocation`

im `root element`, wobei `xsi` das Präfix für den `namespace`

`http://www.w3.org/2001/XMLSchema-instance`

darstellt. Dieses `attribute` enthält Paare von URIs, wobei alle Einträge durch Leerzeichen voneinander getrennt sind. Der erste Eintrag in jedem Paar gibt einen `namespace` an, der innerhalb der Datei verwendet wird. Der zweite Eintrag gibt eine physische Stelle an, wo dieser `namespace` definiert wird.

Da eine OKSTRA®-XML-Datei dem OKSTRA®-XML-Schema der entsprechenden Version genügen muss, ist hier zum `namespace` der verwendeten OKSTRA®-Version eine Angabe zu machen. Beispielsweise kann hier auf das entsprechende Schema-Dokument auf den OKSTRA®-Webseiten verwiesen werden, z.B. für die OKSTRA®-Version 1.015:

```
xsi:schemaLocation="http://schema.okstra.de/1015/okstra
                    http://www.okstra.de/schema/1015/okstra.xsd"
```

4.2 Grundsätzliche Abbildung eines ENTITYs

4.2.1 Content Model

Als Grundregel gilt: Eine Instanz eines OKSTRA®-ENTITYs, d.h. ein OKSTRA®-Objekt, wird in den XML-Daten durch ein `element` beschrieben, mit einem Start-`tag` und Ende-`tag` zu dem ENTITY-Namen.

EXPRESS-CTE:

```
#[CTE-Id] = [ENTITY-Name] ( [...Eigenschaften des ENTITYs...] );
```

XML-Daten:

```
<[ENTITY-Name] gml:id="[XML-Id]">
    [...Eigenschaften des ENTITYs...]
</[ENTITY-Name]>
```

Über das `attribute gml:id` wird dem ENTITY eine Objekt-Id innerhalb der XML-Daten gegeben. Das Attribut `gml:id` wird im `AbstractGMLType` aus GML definiert.

Als **Beispiel** betrachten wir eine Strasse. Die Beschreibung einer Strasse wird geklammert durch `tags` zum Wort "Strasse".

EXPRESS-CTE:

```
#12 = Strasse ( [...Eigenschaften der Strasse...] );
```

XML-Daten:

```
<Strasse gml:id="str1">
    [...Eigenschaften der Strasse...]
</Strasse>
```

4.2.2 XML Schema

Im XML Schema wird ein ENTITY als `complexType` mit zugehörigem `complexContent` abgebildet. Für jeden derartigen `Type` wird ein entsprechendes globales `element` definiert, das das gemäß Vererbungshierarchie übergeordnete Element ersetzen kann, d.h. es befindet sich in dessen `substitutionGroup`. Als übergeordnetes `element` kommen hier das abstrakte `_OKSTRAObjekt` in Frage, oder `elements`, die Netzbezüge und/oder Historisierung repräsentieren, da diese für den OKSTRA® eine besondere Bedeutung haben.

EXPRESS:

```
ENTITY [ENTITY-Name];
    [...Eigenschaften des ENTITYs...]
END_ENTITY;
```

XML Schema:

```
<complexType name="[ENTITY-Name]Type">
    <complexContent>
        <extension base="[AbstractOKSTRAObjektType bzw. ein
                          entsprechender Type für netzbezogene
                            und/oder historisierende Objekte]">
            <sequence>
                [...]
            </sequence>
        </extension>
```

```
        </complexContent>
    </complexType>

    <element name="[ENTITY-Name]" type="[ENTITY-Name]Type"
            substitutionGroup="[_OKSTRAObjekt bzw. ein
                entsprechender Type für netzbezogene
                und/oder historisierende Objekte]" />
```

Das Konstrukt `sequence` legt fest, dass die Eigenschaften in einer XML-Datei gemäß diesem XML Schema in genau der festgelegten Reihenfolge aufgeführt werden müssen. Innerhalb dieses Konstrukts werden grundsätzlich alle Attribute und Relationen eines ENTITYs festgelegt, ganz analog zu der festgelegten Reihenfolge im EXPRESS-Schema.

Der `complexType` zum ENTITY wird in der Regel als `extension` des `AbstractFeatureType` aus GML definiert[4]. Hierzu wird zur strukturellen Klarheit zunächst ein eigener `AbstractOKSTRAObjektType` aus dem `AbstractFeatureType` abgeleitet, aus dem wiederum die eigentlichen OKSTRA®-`Types` erben. Der `complexType` zum ENTITY erhält dadurch allgemeine `elements`, z.B. eine Beschreibung (`description`), und das oben bereits verwendete `attribute gml:id` zur Festlegung einer XML-Id.

XML Schema:

```
    <complexType name="AbstractOKSTRAObjektType" abstract="true">
        <complexContent>
            <extension base="gml:AbstractFeatureType"/>
        </complexContent>
    </complexType>

    <element name="_OKSTRAObjekt" type="okstra:AbstractOKSTRAObjektType"
        abstract="true" substitutionGroup="gml:_Feature"/>
```

Für Punkt-, Strecken- und Bereichsobjekte werden in OKSTRA®-XML abstrakte `complexTypes` definiert (jeweils in einer historisierenden und einer nicht historisierenden Variante), aus denen die entsprechenden ENTITYs, wie z.B. der Betriebskilometer, ebenfalls per `extension` abgeleitet werden.

Das geerbte `attribute gml:id` ermöglicht die Vergabe einer Objekt-Id vom Typ ID als Eigenschaft für jede Instanz des ENTITYs. Dies dient zur Identifikation, z.B. für Verweise.

Anmerkung: XML Schema erlaubt derzeit nur einfache Vererbung, d.h. ein `complexType` kann höchstens aus einem anderen `Type` abgeleitet werden. Daher ist die `extension` nicht als allgemeines Mittel zur Darstellung von Vererbung in OKSTRA®-XML ausreichend. Die `extension` wird hier für den besonders wichtigen Vererbungszweig von Netzbezug und/oder Historisierung verwendet.

---

[4] Die `extension` in XML arbeitet ähnlich wie die Vererbung (SUBTYPE) in EXPRESS. Der abgeleitete `Type` erhält zu seinen eigenen Eigenschaften die Eigenschaften des Supertypes hinzu.

XML Schema:

```
    <!-- -->
    <!-- ================================================== -->
    <!-- = abstrakte OKSTRA-Typen, jeweils als              = -->
    <!-- = historisierende und nicht historisierende Variante = -->
    <!-- ================================================== -->
    <!-- -->
    <complexType name="AbstractOKSTRAObjektType" abstract="true">
        <complexContent>
            <extension base="gml:AbstractFeatureType"/>
        </complexContent>
    </complexType>
    <group name="HistGroup">
        <sequence>
            <element name="gueltig_von" type="okstra:Datum"
                minOccurs="0"/>
            <element name="gueltig_bis" type="okstra:Datum"
                minOccurs="0"/>
            <element name="erzeugt_von_Ereignis"
                type="okstra:ObjectRefType"  minOccurs="0">
                <annotation>
                    <appinfo>
                        <okstra:Zielobjekttyp>
                            Ereignis
                        </okstra:Zielobjekttyp>
                        <okstra:inverseRelation>
                            erzeugt_historisches_Objekt
                        </okstra:inverseRelation>
                    </appinfo>
                </annotation>
            </element>
            <element name="geloescht_von_Ereignis"
                type="okstra:ObjectRefType" minOccurs="0">
                <annotation>
                    <appinfo>
                        <okstra:Zielobjekttyp>
                            Ereignis
                        </okstra:Zielobjekttyp>
                        <okstra:inverseRelation>
                            loescht_historisches_Objekt
                        </okstra:inverseRelation>
```

```
                </appinfo>
            </annotation>
        </element>
        <element name="hat_Vorgaenger_hist_Objekt"
            type="okstra:ObjectRefType" minOccurs="0">
            <annotation>
                <appinfo>
                    <okstra:Zielobjekttyp>
                        historisches_Objekt
                    </okstra:Zielobjekttyp>
                    <okstra:inverseRelation>
                        hat_Nachfolger_hist_Objekt
                    </okstra:inverseRelation>
                </appinfo>
            </annotation>
        </element>
        <element name="hat_Nachfolger_hist_Objekt"
            type="okstra:ObjectRefType" minOccurs="0">
            <annotation>
                <appinfo>
                    <okstra:Zielobjekttyp>
                        historisches_Objekt
                    </okstra:Zielobjekttyp>
                    <okstra:inverseRelation>
                        hat_Vorgaenger_hist_Objekt
                    </okstra:inverseRelation>
                </appinfo>
            </annotation>
        </element>
    </sequence>
</group>
<complexType name="AbstractOKSTRAHistObjektType" abstract="true">
    <complexContent>
        <extension base="okstra:AbstractOKSTRAObjektType">
            <sequence>
                <group ref="okstra:HistGroup"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- -->
```

```
<!-- ================================================= -->
<!-- = globale complexTypes fuer die Netzbezugs-Klassen = -->
<!-- ================================================= -->
<!-- -->
<complexType name="PunktobjektType">
    <complexContent>
        <extension base="okstra:AbstractOKSTRAObjektType">
            <sequence>
                <element name="bei_Strassenpunkt"
                    type="okstra:StrassenpunktPropertyType"
                    minOccurs = "0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="StreckenobjektType">
    <complexContent>
        <extension base="okstra:AbstractOKSTRAObjektType">
            <sequence>
                <element name="hat_Strecke"
                    type="okstra:ObjectRefType"
                    minOccurs="0" maxOccurs="unbounded">
                    <annotation>
                        <appinfo>
                            <okstra:Zielobjekttyp>
                                verallgemeinerte_Strecke
                            </okstra:Zielobjekttyp>
                            <okstra:inverseRelation>
                                zu_Streckenobjekt
                            </okstra:inverseRelation>
                        </appinfo>
                    </annotation>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="BereichsobjektType">
    <complexContent>
        <extension base="okstra:AbstractOKSTRAObjektType">
            <sequence>
```

```
                <element name="hat_Netzbereich"
                    type="okstra:ObjectRefType"
                    minOccurs="0" maxOccurs="unbounded">
                    <annotation>
                        <appinfo>
                            <okstra:Zielobjekttyp>
                                Netzbereich
                            </okstra:Zielobjekttyp>
                            <okstra:inverseRelation>
                                zu_Bereichsobjekt
                            </okstra:inverseRelation>
                        </appinfo>
                    </annotation>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="PunktobjektHistType">
    <complexContent>
        <extension base="okstra:PunktobjektType">
            <sequence>
                <group ref="okstra:HistGroup"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="StreckenobjektHistType">
    <complexContent>
        <extension base="okstra:StreckenobjektType">
            <sequence>
                <group ref="okstra:HistGroup"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<complexType name="BereichsobjektHistType">
    <complexContent>
        <extension base="okstra:BereichsobjektType">
            <sequence>
                <group ref="okstra:HistGroup"/>
```

```
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <!-- definiere Basis-Elemente fuer OKSTRA-Elemente -->
    <element name="_OKSTRAObjekt" type="okstra:AbstractOKSTRAObjektType"
        abstract="true" substitutionGroup="gml:_Feature"/>
    <element name="_OKSTRAHistObjekt"
        type="okstra:AbstractOKSTRAHistObjektType" abstract="true"
        substitutionGroup="okstra:_OKSTRAObjekt"/>
    <element name="_Punktobjekt" type="okstra:PunktobjektType"
        abstract="true" substitutionGroup="okstra:_OKSTRAObjekt"/>
    <element name="_Streckenobjekt" type="okstra:StreckenobjektType"
        abstract="true" substitutionGroup="okstra:_OKSTRAObjekt"/>
    <element name="_Bereichsobjekt" type="okstra:BereichsobjektType"
        abstract="true" substitutionGroup="okstra:_OKSTRAObjekt"/>
    <element name="_PunktobjektHist" type="okstra:PunktobjektHistType"
        abstract="true" substitutionGroup="okstra:_Punktobjekt"/>
    <element name="_StreckenobjektHist"
        type="okstra:StreckenobjektHistType" abstract="true"
        substitutionGroup="okstra:_Streckenobjekt"/>
    <element name="_BereichsobjektHist"
        type="okstra:BereichsobjektHistType" abstract="true"
        substitutionGroup="okstra:_Bereichsobjekt"/>
```

Im **Beispiel** der *Straße* ergibt sich folgende Modellierung:

EXPRESS:

```
    ENTITY Strasse;
        [...Eigenschaften der Strasse...]
    END_ENTITY;
```

XML Schema:

```
    <complexType name="StrasseType">
        <complexContent>
            <extension base="okstra:AbstractOKSTRAHistObjektType”>
                <sequence>
                    [...]
                </sequence>
```

```
            </extension>
        </complexContent>
    </complexType>

    <element name="Strasse" type="okstra:StrasseType"
             substitutionGroup="okstra:_OKSTRAHistObjekt" />

    <complexType name="BetriebskilometerType">
        <complexContent>
            <extension base="okstra:PunktobjektHistType">
                <sequence>
                    [...]
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <element name="Betriebskilometer"
             type="okstra:BetriebskilometerType"
             substitutionGroup="okstra:_PunktobjektHist" />
```

4.3 Grundsätzliche Abbildung eines TYPEs

4.3.1 Content Model

Im OKSTRA® werden EXPRESS-TYPEs definiert, die letztlich auf einfache Datentypen zurückgehen. In den XML-Daten wird ein Attribut mit einem solchen definierten TYPE als `element` mit dem zugehörigen einfachen Datentyp als `Type` dargestellt.

EXPRESS-CTE:

```
    #11 = BAB_Knotennummer(..., 12, ...);
```

XML-Daten:

```
    <BAB_Knotennummer>
        [...]
        <Knotennummer>12</Knotennummer>
        [...]
    </BAB_Knotennummer>
```

4.3.2 XML Schema

Ein selbstdefinierter TYPE wird im XML Schema

- bei TYPEs ohne Einheit in Form eines `simpleTypes` als `restriction` des finalen Basisdatentyps definiert und
- bei TYPEs mit Einheit in Form eines `simpleTypes` als `extension` des finalen Basisdatentyps oder eines passenden GML-Types mit einem zusätzlichen `attribute uom` für die Einheit definiert bzw. bei weiterer Spezialisierung wieder als `restriction` eines solchen TYPEs.

EXPRESS:

```
TYPE [TYPE-Name] = [Basisdatentyp];
WHERE
    [Bedingungen]
END_TYPE;
```

XML Schema:

```
<simpleType name="[TYPE-Name]Type">
    <restriction base="[finaler Basisdatentyp]"/>
</simpleType>
```

In der EXPRESS-Modellierung des OKSTRA® verwenden wir für TYPE-Definitionen nur einige wenige Basis-Datentypen. Die folgende Tabelle gibt die Korrespondenz zu vordefinierten Datentypen in XML Schema:

EXPRESS	XML Schema
INTEGER	integer
REAL	double
STRING	string
STRING (für Datum)	date

Tabelle 1 – Korrespondenz von Datentypen

Falls bei TYPEs mit Einheit ein passender GML-Type definiert ist, z.B. gml:LengthType für Längen-Angaben, wird dieser Type verwendet. In diesem Fall wird der TYPE als `restriction` des GML Types definiert, wobei das `attribute uom` auf die korrekte Maßeinheit fixiert wird.

Da es sich bei OKSTRA®-XML um ein abgeleitetes Anwendungsschema handelt, werden die Einschränkungen an den Wertebereich eines TYPEs nicht mit in das XML Schema aufgenommen. Die Überprüfung der Einhaltung dieser Bedingungen ist Aufgabe der Anwendungssoftware.

Als **Beispiel** betrachten wir den TYPE *Anzahl*, definiert als die Menge der nicht-negativen ganzen Zahlen.

EXPRESS:

```
TYPE Anzahl = INTEGER;
```

```
    WHERE
        Anzahl_nicht_negativ    : SELF >= 0;
    END_TYPE;
```

XML Schema:

```
    <simpleType name="AnzahlType">
        <restriction base="integer"/>
    </simpleType>
```

Bei mehrstufigen TYPE-Definitionen, z.B. einen TYPE *Anzahl_dreistellig* mit den ganzen Zahlen von 0 – 999, wird die `restriction` direkt mit dem finalen Basisdatentyp gebildet:

EXPRESS:

```
    TYPE Anzahl_dreistellig = Anzahl;
    WHERE
        maximal_dreistellig    : SELF <= 999;
    END_TYPE;
```

XML Schema:

```
    <simpleType name="Anzahl_dreistelligType">
        <restriction base="integer"/>
    </simpleType>
```

Alternativ könnte man die TYPE-Struktur aus EXPRESS übernehmen. In unserem Beispiel würde dann als `restriction base` nicht `integer` sondern `Anzahl` genommen. Für eine konzeptionelle Modellierung des OKSTRA® in XML Schema wäre dies sicher angemessen. Da der Fokus hier jedoch auf der Repräsentierung, Bereitstellung und Übertragung von OKSTRA®-Daten liegt, schlagen wir die weniger komplexe Darstellung mit einer einstufigen TYPE-Definition vor.

4.4 Abbildung von Attributen

Für den Datentyp eines Attributs im OKSTRA® wurden folgende Konstrukte verwendet:

- einfacher Datentyp,
- definierter TYPE, der auf einem einfachen Datentyp oder wieder auf einem definierten TYPE basiert,
- als Sonderfall ein ENTITY, das eine Schlüsseltabelle darstellt.

Der Sonderfall der Schlüsseltabellen wird in 4.9 behandelt. Hier gehen wir also von einem einfachen Datentyp aus bzw. von einem Datentyp der über ggf. mehrere TYPE-Definitionen auf einem einfachen Datentyp basiert.

Wichtige Anmerkung: Der Begriff "Attribut" wird im OKSTRA® für einfache, nicht objekthafte Eigenschaften eines ENTITYs verwendet. Dies ist nicht zu verwechseln mit dem `attribute` in XML Schema.

4.4.1 Content Model

Zur Darstellung des Attributs wird stets ein Wert des zugrundeliegenden einfachen Datentyps verwendet. Dies ist die gleiche Vorgehensweise wie bei CTE.

EXPRESS-CTE:

```
#[CTE-Id] = [ENTITY-Name] (...,[Wert einfacher Datentyp],...);
```

XML-Daten:

```
<[Attribut-Name]>
    [Wert einfacher Datentyp]
</[Attribut-Name]>
```

Als **Beispiel** betrachten wir hier die Anzahl der Fahrzeugbenutzer bei den Angaben zu einem *Unfallbeteiligten* im Schema „Unfall":

EXPRESS-CTE:

```
#34 = Unfallbeteiligter (...,3,...);
```

XML-Daten:

```
<Unfallbeteiligter>
    [...]
    <Anzahl_Fahrzeugbenutzer>3</Anzahl_Fahrzeugbenutzer>
    [...]
</Unfallbeteiligter>
```

4.4.2 XML Schema

Ein Attribut wird als `element` innerhalb der `complexType`-Definition seines zugehörigen ENTITYs definiert, d.h. als lokales Element. Als Datentyp erhält er seinen definierten Datentyp, der gemäß den TYPE-Abbildungen in 4.3 im XML Schema definiert wird:

EXPRESS:

```
[Attribut-Name]    : [Datentyp des Attributs];
```

XML Schema:

```
<element name="[Attribut-Name]"
    type="[Datentyp des Attributs]Type"
    minOccurs="[0 für OPTIONAL Attribut, sonst 1]" />
```

Die Eigenschaft `minOccurs` legt für ein OPTIONAL Attribut die minimale Anzahl auf `0` fest. Für zwingende Attribute muss dieses `attribute` nicht gesetzt werden, da der Default `1` ist.

Die Behandlung multipler Attribute, die im OKSTRA® sehr selten vorkommen, erfolgt analog zur Handhabung multipler Relationen (siehe 4.5). Hier wird der Deutlichkeit halber nur der Standard-Fall eines einfachen Attributs, in optionaler oder zwingender Modalität, dargestellt.

Für die Abbildung von Kardinalitäten siehe die entsprechende Beschreibung in 4.5.2.

Als **Beispiel** betrachten wir wie oben den *Unfallbeteiligten*:

EXPRESS:

```
    Anzahl_Fahrzeugbenutzer     : Anzahl_zweistellig;
```

XML Schema:

```
    <element name="Anzahl_Fahrzeugbenutzer"
                type="okstra:Anzahl_zweistellig" />
```

Der Datentyp *Anzahl_zweistellig* wird dabei wie in 0 erläutert definiert.

4.5 Abbildung von Relationen

Bei der Abbildung von Relationen sind verschiedene Kriterien zu berücksichtigen:

- Ist die Relation direkt oder invers?
- Ist die Relation optional?
- Ist die Relation einfach oder multipel (SET oder LIST)?
- Ist der Relationspartner ein konzeptionelles Objekt, d.h. vollständig abhängig?

Für OKSTRA®-XML-Daten werden zwei Möglichkeiten zur Darstellung von Relationen angeboten, bezogen auf die Modellierung der Relation in EXPRESS:

- erste Variante: Relationen werden grundsätzlich nur in der direkten Richtung aufgeführt. Die inverse Richtung ergibt sich implizit.
- zweite Variante: Relationen werden grundsätzlich beidseitig angegeben. Direkte und inverse Richtung in den XML-Daten müssen miteinander verträglich sein, d.h. A verweist auf B mit einer Relation R genau dann, wenn B auf A mit der zu R inversen Relation R^{-1} verweist.

Zu beachten ist hier, dass mit der ersten Variante für inverse Relationen Verweise auf Objekte außerhalb einer XML-Datei nicht direkt navigierbar sind, da der Verweis nur in dem außerhalb liegenden Objekt verzeichnet ist. Sofern man sich nur für eine Richtung der Relation interessiert, z.B. von einem untergeordneten Objekt auf ein übergeordnetes, kann dies durchaus sinnvoll sein und spart Speicherplatz. Die zweite Variante enthält dafür Redundanzen und ermöglicht Inkonsistenzen in den Relationen. Das OKSTRA®-XML Schema unterstützt beide Varianten.

Die erste Variante eignet sich vor allem für eine kompakte Speicherung von in sich geschlossenen Datenbeständen, d.h. Relationen bestehen nur zwischen Objekten innerhalb einer XML-Datei. Wie gesagt sind Relationen zu Objekten außerhalb einer Datei bei dieser Variante nur entlang direkter Relationen navigierbar, die inversen Relationen sind nicht (direkt) navigierbar.

Die zweite Variante eignet sich in vollem Umfang auch für Verweise auf Objekte außerhalb einer XML-Datei. Ferner erleichtert diese Variante die Navigation und Auswertung von Daten, da alle Relationen eines Objekts komplett in der XML-Darstellung des Objekts vorliegen.

Aus Konsistenzgründen gilt die gewählte Variante für alle in einer XML-Datei enthaltenen Objekte. In den Metadaten der OKSTRA®-Objektmenge wird die verwendete Variante angegeben. Fehlt eine Angabe, so wird Variante 1 angenommen (redundanzfreie Repräsentierung).

XML-Daten:

```
<okstra:OKSTRAObjektmenge relRep="beidseitig">
    <gml:metaDataProperty>
        <okstra:OKSTRAMetaDaten>
            <okstra:schema_identifiers>
                OKSTRA 1.015
            </okstra:schema_identifiers>
        </okstra:OKSTRAMetaDaten>
    </gml:metaDataProperty>
    <gml:description>Beispielmenge</gml:description>
    <gml:boundedBy>
        <gml:Box srsName="DE_DHDN_3GK4">
            <gml:coordinates>
                4373512.25590199,5653729.90479046
                4373533.01647874,5653733.51509939
                4373602.10292724,5653755.70699833
                4373784.10645786,5653820.92257877
                4373952.27785213,5653880.28765859
            </gml:coordinates>
        </gml:Box>
    </gml:boundedBy>
    <okstra:okstraObjekt>
        <okstra:Strasse><!-- ... --></okstra:Strasse>
    </okstra:okstraObjekt >
    <okstra:okstraObjekt >
        <okstra:Abschnitt><!-- ... --></okstra:Abschnitt>
    </okstra:okstraObjekt >
    <!-- ... -->
</okstra:OKSTRAObjektmenge>
```

Beziehungen zu konzeptionellen Objekten, wie z.B. zum Strassenpunkt, werden nicht als Relationen abgebildet. Die Darstellung des konzeptionellen Objekts wird in die Beschreibung des Objekts integriert, das es verwendet. Zur Abbildung konzeptioneller Objekte siehe 4.6.

Wir betrachten hier also nur den Fall von Relationen zwischen eigenständigen Objekten. Dabei verwenden wir die XLink-Sprache, die Teil der Familie der XML-Technologien ist.

4.5.1 Content Model

Für den Verweis verwenden wir einen selbstdefinierten `complexType ObjectRefType`. Dieser beschreibt

- entweder einen URI (Uniform Resource Identifier) – das Partnerobjekt wird in Form einer URL (Uniform Resource Locator) über seinen primären Zugriffspfad identifiziert, z.B. in Form einer http-Adresse über das Internet oder auch in Form eines lokalen XML-Identifiers innerhalb der Datei (siehe 4.2.1) bzw. über einen URN (Uniform Resource Name) wobei die Vergabe von URN Namespaces und die Auflösung von URNs Aufgabe der Anwendungssoftware ist
- oder einen symbolischen Verweis[5] – das Partnerobjekt wird über seinen global eindeutigen Schlüssel adressiert, z.B. ein Netzknoten über seine eindeutige Netzknoten-Nummer. Wichtig: Hier dürfen ausschließlich die festgelegten Kennungen der symbolischen Verweise verwendet werden, wie sie im entsprechenden (NIAM-)Dokument zum zugehörigen Schema definiert sind. Dies stellt sicher, dass die Angabe vom Datenempfänger interpretiert werden kann.

Der URI erhält dabei Vorrang vor dem symbolischen Verweis, falls – unerlaubterweise – beide vorhanden sind.

Wir betrachten als **Beispiel** einen Verweis über einen URL:

EXPRESS-CTE:

```
#[CTE-Id] = [ENTITY-Name] (...,#[CTE-Id des Relationspartners],...);
```

XML-Daten:

```
<[Relations-Name] xlink:href="[URL des Partnerobjekts]" />
```

Als **Beispiel** betrachten wir die Beziehung vom *Nullpunktort* zum *Nullpunkt*:

EXPRESS-CTE:

```
#56 = Nullpunktort (...,#78);
```

XML-Daten:

```
<bei_Nullpunkt xlink:href="#o156" />
```

Hier wird als Verweis ein URL verwendet, der auf einen *Nullpunkt* mit der XML-ID `#o156` innerhalb derselben XML-Datei zeigt. Dieser *Nullpunkt* sei der gleiche, der in der CTE-Datei die ENTITY-Nummer #78 trägt.

Multiple Relationen werden durch Wiederholung dieses Elements für jeden Relationspartner abgebildet. Bei Listen (LIST) wird die Reihenfolge der Relationspartner durch die so gegebene Reihenfolge bestimmt, bei Mengen (SET) ist die Reihenfolge irrelevant.

---

[5] Der im Sinne von XML korrekte Weg wäre hier die Verwendung eines URN (Uniform Resource Name). Im Hinblick auf die derzeit laufenden Diskussionen zur Objekt-Id ist davon auszugehen, dass in näherer Zukunft diese „Notlösung" mit einer String-Darstellung fallengelassen werden kann und zu URNs übergegangen werden kann. Dieser zweite Weg ist daher als „deprecated feature" zu betrachten.

In der folgenden Beschreibung verwenden wir die alternative Darstellung der Relation über symbolische Verweise.

EXPRESS-CTE:

```
#[CTE-Id] = [ENTITY-Name] (...,(#[CTE-Id RP1],#[CTE-Id RP2],
                              #[CTE-Id RP3],...),...);
```

XML-Daten:

```
<[Relations-Name]>[Kennung des Partnerobjekts 1]</Relations-Name>
<[Relations-Name]>[Kennung des Partnerobjekts 2]</Relations-Name>
<[Relations-Name]>[Kennung des Partnerobjekts 3]</Relations-Name>
[... weitere Relationspartner ...]
```

Wir betrachten als **Beispiel** die *Nullpunkte* zu einem *Netzknoten*. Diesmal verwenden wir symbolische Verweise zur Darstellung der Relation zu den *Nullpunkten*.

EXPRESS-CTE:

```
#12 = Netzknoten (...,(#121,#122,#123),...);
#121 = Nullpunkt_Symbol ("5308001B");
#122 = Nullpunkt_Symbol ("5308001C");
#123 = Nullpunkt_Symbol ("5308001O");
```

XML-Daten:

```
<hat_Nullpunkt>5308001B</hat_Nullpunkt>
<hat_Nullpunkt>5308001C</hat_Nullpunkt>
<hat_Nullpunkt>5308001O</hat_Nullpunkt>
```

4.5.2 XML Schema

Der oben genannte `complexType ObjectRefType` für die Darstellung von Relationen wird wie folgt definiert:

XML Schema:

```
<complexType name="ObjectRefType">
    <simpleContent>
        <extension base="string">
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
            <attribute name="Objektklasse" type="string"
                use="optional"/>
        </extension>
```

```
        </simpleContent>
    </complexType>
```

Die referenzierte `attributeGroup gml:AssociationAttributeGroup` liefert die Möglichkeit eines Verweises über einen URI, der `string` aus der `extension` dient zur Darstellung eines symbolischen Verweises.

Das optionale `attribute` "Objektklasse" ermöglicht es, die instanzierte Objektklasse des Relationspartners zu speichern. Dies kann zur Navigation und Auswertung einen erheblichen Performancegewinn liefern, wenn die Klasse des Relationspartners mehr als eine instanzierbare Ausprägung hat, z.B. ein ABSTRACT SUPERTYPE mit mehreren SUBTYPEs in der EXPRESS-Modellierung des OKSTRA®.

Es gibt keine Entsprechung zum genannten `ObjectRefType` im EXPRESS-Schema des OKSTRA®. Dort werden stets die expliziten ENTITY-Namen aufgeführt.

Eine Relation wird im XML Schema als `element` mit `type ObjectRefType` dargestellt.

Zusätzlich wird bei der Relation vermerkt, welches der Zielobjekttyp ist und wie die Gegenrelation heißt (sofern vorhanden).

EXPRESS:

```
    [Relations-Name]     : [Klasse des Relationspartners];
```

Anm.: Die EXPRESS-Darstellung ist hier nur ein Beispiel. Die Relation kann noch weitere Eigenschaften haben wie OPTIONAL, SET oder LIST, mit den entsprechenden Grenzen.

XML Schema:

```
    <element name="[Relations-Name]"
        type="okstra:ObjectRefType"
        minOccurs="[minimale Anzahl]"
        maxOccurs="[maximale Anzahl]">
        <annotation>
            <appinfo>
                <okstra:Zielobjekttyp>
                    [Klassenname des Zielobjekts]
                </okstra:Zielobjekttyp>
                <okstra:inverseRelation>
                    [Name der zugehörigen inversen Relation]
                </okstra:inverseRelation>
            </appinfo>
        </annotation>
    </element>
```

Der innerhalb des `appinfo elements` angegebene `Zielobjekttyp` gibt den ENTITY-Namen des Relationspartners an, als Unterstützung für die Anwendungssoftware.

Warnung: Dieser Name ist für Instanzen dieses `elements` nicht immer identisch mit dem `element`-Namen des zugehörigen Relationspartners oder eines `elements` in dessen `substitutionGroup` der Relationspartner liegt, sondern gibt den ENTITY-Namen aus der Referenzmodellierung des OKSTRA® wieder.

Die angegebene `inverseRelation` gibt die korrespondierende Relation im Relationspartner an, falls diese sinnvoll angegeben werden kann. Diese `inverseRelation` wird auch für Paare von Halbrelationen angegeben, deren Beziehung als Relation und inverse Relation infolge der Einführung symbolischer Verweise aufgelöst worden sind, z.B. zwischen Netzknoten und BAB_Knotennummer.

Die Eigenschaften `minOccurs` bzw. `maxOccurs` geben die minimale bzw. maximale Anzahl von Relationspartnern an. Als Grundregel sind dies die gemäß EXPRESS-Schema vorgeschriebenen Grenzen, jedoch mit folgenden zusätzlichen Regeln:

- Der Default-Wert für `minOccurs` und `maxOccurs` ist in beiden Fällen `1`. Ist eine der beiden Grenzen `1`, so braucht das entsprechende `attribute` nicht angegeben zu werden.
- Der EXPRESS-Ausdruck OPTIONAL bewirkt zwangsweise ein `minOccurs="0"`, auch wenn bei multiplen Relationen eine explizite untere Grenze angegeben ist.
- Eine von beiden Richtungen jeder Relation ist auf jeden Fall mit einem `minOccurs="0"` versehen, unabhängig davon welche konzeptionellen Kardinalitäten im EXPRESS angegeben sind. Dies ist erforderlich, um einen sukzessiven Aufbau von gültigen OKSTRA®-XML-Daten zu ermöglichen. Beidseitig zwingende Relationen würden hier zu einem Deadlock führen. Sofern im EXPRESS eine inverse Relation angegeben ist, wird diese verwendet. Andernfalls wird die aus semantischer Sicht inverse Relation gewählt.
- Die in EXPRESS angegebene minimale Anzahl wird in `minOccurs` eingetragen, es sei denn ein OPTIONAL ist gegeben oder es handelt sich um die inverse Richtung der Relation (s.o.).
- Die in EXPRESS angegebene maximale Anzahl wird in `maxOccurs` eingetragen. Eine unbestimmte obere Grenze ('?') wird durch den Ausdruck `unbounded` beschrieben.

In folgender Tabelle sind diese Korrespondenzen zwischen EXPRESS und XML Schema noch einmal beispielhaft aufgeführt:

EXPRESS	`minOccurs`	`maxOccurs`
[Name Relationspartner]	1 (kann entfallen)	1 (kann entfallen)
OPTIONAL [Name Relationspartner]	0	1 (kann entfallen)
SET [5:10] OF [Name Relationspartner]	5	10
SET [1:?] OF [Name Relationspartner]	1 (kann entfallen)	unbounded
LIST [2:3] OF [Name Relationspartner]	2	3
LIST [1:?] OF [Name Relationspartner]	1 (kann entfallen)	unbounded
OPTIONAL SET [3:?] OF [Name Relationspartner]	0	unbounded
OPTIONAL LIST [1:3] OF [Name Relationspartner]	0	3

Tabelle 2 – Kardinalitäten von Relationen in XML Schema

Anmerkung: Für optionale Mengen und Listen (OPTIONAL SET, OPTIONAL LIST ...) kann diese Festlegung leicht unscharf sein. Der Ausdruck OPTIONAL LIST [2:3] erlaubt beispielsweise 0, 2 oder 3 Einträge. Im gegebenen XML Schema werden nur die extremalen Werte festgelegt.

Wichtig: In den XML-Daten werden Einträge für optionale Attribute oder Relationen, für die keine Werte gegeben sind, ganz weggelassen. Dies ist möglich, da die Elemente explizit benannt sind. In OKSTRA®-CTE musste für fehlende Informationen stets ein "$" angegeben werden, da die Bedeutung eines Elements ausschließlich durch seine Position bestimmt war.

Als **Beispiel** betrachten wir wie oben die *Nullpunkte* zu einem *Netzknoten*.

EXPRESS:

```
    hat_Nullpunkt     : SET [1:?] OF Nullpunkt;
```

XML Schema:

```
    <element name="hat_Nullpunkt" type="okstra:ObjectRefType"
        minOccurs="1"
        maxOccurs="unbounded" />
```

Hierbei müsste das `attribute minOccurs` nicht gesetzt sein, da `1` der Default ist.

4.6 Konzeptionelle Objekte

Wie in 4.5 bereits angedeutet, wird in den XML-Daten die Darstellung eines konzeptionellen Objekts, d.h. vollständig von einem übergeordneten Objekt abhängigen Objekts, in die Darstellung des übergeordneten Objekts integriert. Man kann sagen, dass ein konzeptionelles Objekt attributiven Charakter hat und nur zur konzeptionellen Klarheit als eigene Objektklasse modelliert wird.

Verweist ein konzeptionelles Objekt mit einer Relation auf eine andere Objektart, handelt es sich dabei ebenfalls um eine Angabe von „attributivem" Charakter; in diesem Fall existiert daher keine Rückrelation. Beispielsweise verweist das konzeptionelle Objekt *Straßenpunkt* zwar auf den *Abschnitt_oder_Ast*, auf dem der *Straßenpunkt* liegt; der *Abschnitt_oder_Ast* hat aber keine Rückrelation zum *Straßenpunkt*.

GML legt fest, dass sich in einem GML-Anwendungsschema Object-Types und Property-Types bei Verschachtelung abwechseln müssen. Das bedeutet für die Einbettung der konzeptionellen Objekte, dass in dem `element` zur Relation (Property) zunächst wieder ein Object-Type folgen muss. Erst auf der nächsten Stufe dürfen wieder Eigenschaften (Property) folgen.

4.6.1 Content Model

Die Darstellung des konzeptionellen Relationspartners, wie sie für ein eigenständiges Objekt gegeben wäre, wird in die Darstellung des übergeordneten Objekts integriert. Für CTE konnte ein solcher Übergang von der konzeptionellen Modellierung zu einer geeigneten datentechnischen Repräsentierung leider nicht festgelegt werden.

XML-Daten:

```
[...davor liegende Eigenschaften des übergeordneten Objekts...]
<[Attribut-Name zum konzeptionellen Objekt]>
    <[Object-Type zum konzeptionellen Objekt]>
        [Darstellung des konzeptionellen Objekts]
    </[Object-Type zum konzeptionellen Objekt]>
</[Attribut-Name zum konzeptionellen Objekt]>
[...dahinter liegende Eigenschaften des übergeordneten Objekts...]
```

Als **Beispiel** betrachten wir den *Straßenpunkt* zum *Betriebskilometer*:

XML-Daten:

```
[...davor liegende Eigenschaften des Betriebskilometers...]
<bei_Strassenpunkt>
    <Strassenpunkt>
        <Station>[Station des Strassenpunkts]</Station>
        <auf_Abschnitt_oder_Ast xlink:href="[uri des zugehoerigen
                                            Abschnitt_oder_Ast]"/>
    </Strassenpunkt>
</bei_Strassenpunkt>
[...dahinter liegende Eigenschaften des Betriebskilometers...]
```

4.6.2 XML Schema

Die Beschreibung des konzeptionellen Objekts wird im XML Schema in die Beschreibung des übergeordneten Objekts eingebettet. Der zugehörige `Type` des konzeptionellen Objekts wird jedoch global definiert (in der Datei `okstra_konz_typen.xsd`).

EXPRESS:

```
ENTITY [Name konz. Objekt]
SUBTYPE OF (OKSTRA_konzept_Objekt);
    [Eigenschaften des konzeptionellen Objekts]
END_ENTITY;

ENTITY [Name des übergeordneten ENTITYs];
    [...]
    [Relations-Name]    : [Name konz. Objekt];
    [...]
END_ENTITY;
```

XML Schema:

```
<complexType name="[ENTITY-Name konz. Objekt]PropertyType">
    <sequence>
        <element name="[ENTITY-Name konz. Objekt]"
            type="[ENTITY-Name konz. Objekt]Type"/>
    </sequence>
</complexType>
<complexType name="[ENTITY-Name konz. Objekt]Type">
    <sequence>
        [Beschreibung des konzeptionellen Objekts]
    </sequence>
</complexType>


[... davor liegende Beschreibung des verwendenden Objekts ...]
<element name="[Relations-Name zum konzeptionellen Objekt]"
        type="okstra:[ENTITY-Name konz. Objekt]PropertyType" />
[... dahinter liegende Beschreibung des verwendenden Objekts ...]
```

Als **Beispiel** betrachten wir den *Straßenpunkt* zum *Betriebskilometer*:

EXPRESS:

```
ENTITY Betriebskilometer
SUBTYPE OF (Punktobjekt);
    in_Block                      : SET [1:?] OF Block;
END_ENTITY;

ENTITY Punktobjekt
ABSTRACT SUPERTYPE OF (ONEOF(Betriebskilometer));
    bei_Strassenpunkt             : OPTIONAL Strassenpunkt;
END_ENTITY;

ENTITY Strassenpunkt
SUBTYPE OF (OKSTRA_konzept_Objekt);
    Station                       : Kilometer;
    Abstand_zur_Bestandsachse     : OPTIONAL Meter;
    Abstand_zur_Fahrbahnoberkante : OPTIONAL Meter;
    auf_Abschnitt_oder_Ast        : SET [1:?] OF
                                    Abschnitt_oder_Ast_abstrakt;
END_ENTITY;
```

XML Schema:

```
<complexType name="StrassenpunktPropertyType">
    <sequence>
        <element name="Strassenpunkt"
            type="okstra:StrassenpunktType"/>
    </sequence>
</complexType>
<complexType name="StrassenpunktType">
    <sequence>
        <!-- Eigenschaften Strassenpunkt-->
        <element name="Station" type="okstra:Meter"/>
        <element name="Abstand_zur_Bestandsachse" type="okstra:Meter"
            minOccurs="0"/>
        <element name="Abstand_zur_Fahrbahnoberkante"
            type="okstra:Meter" minOccurs="0"/>
        <element name="auf_Abschnitt_oder_Ast"
            type="okstra:ObjectRefType"
            maxOccurs="unbounded"/>
    </sequence>
</complexType>

[... davor liegende Eigenschaften des Betriebskilometers ...]
<element name="bei_Strassenpunkt" type="StrassenpunktPropertyType"
    minOccurs="0"/>
[... dahinter liegende Eigenschaften des Betriebskilometers ...]
```

4.7 Grundsätzliche Abbildung von SUPERTYPEs

Das XML Schema des OKSTRA® soll, wie bereits eingangs erläutert, vor allem als weiteres Format zur Repräsentierung, Bereitstellung und Übertragung von OKSTRA®-Daten dienen. Für eine performante Darstellung empfiehlt sich daher ein möglichst flaches Vererbungsmodell. Im OKSTRA® sollten daher als Grundregel die Attribute und Relationen der SUPERTYPEs in die Darstellung des abgeleiteten ENTITYs (SUBTYPE) eingebettet werden – ganz analog zum Vorgehen bei OKSTRA®-CTE.

4.7.1 Content Model

Wie beschrieben wird die Darstellung des SUPERTYPEs in die Darstellung des SUBTYPEs eingebettet.

EXPRESS-CTE:

```
#[CTE-Id] = [ENTITY-Name] ( [Eigenschaften erster SUPERTYPE],
```

```
                                [Eigenschaften zweiter SUPERTYPE],
                                [...],
                                [Eigenschaften letzter SUPERTYPE],
                                [eigene Eigenschaften] );
```

XML-Daten:

```
    [...]
    [Eigenschaften erster SUPERTYPE]
    [Eigenschaften zweiter SUPERTYPE]
    [...]
    [Eigenschaften letzter SUPERTYPE]
    [... Darstellung der eigenen Eigenschaften des ENTITYs ...]
```

Wir betrachten als **Beispiel** eine *Route*:

EXPRESS-CTE:

```
    #90 = Route ( ... , '01.01.1980',$,#91,$,$,(#92,#93,#94),$ );
    #91 = Ereignis (...);
    #92 = Strassenelement (...);
    #93 = Strassenelement (...);
    #94 = Strassenelement (...);
```

XML-Daten:

```
    [...]
    <!--  Eigenschaften des SUPERTYPEs ASB_Objekt  -->
    [...]
    <!--  Eigenschaften des SUPERTYPEs Teilnetzkomponente  -->
    [...]
    <!--  Eigenschaften des SUPERTYPEs Routenkomponente  -->
    [...]
    <!--  Eigenschaften des SUPERTYPEs historisches_Objekt  -->
    <gueltig_von>1980-01-01</gueltig_von>
    <erzeugt_von_Ereignis xlink:href="[Verweis auf Ereignis]" />
    <!--  eigene Eigenschaften des ENTITYs Route  -->
    <entlang_Routenkomponente xlink:href="[Verweis auf Routenkomp.]" />
    <entlang_Routenkomponente xlink:href="[Verweis auf Routenkomp.]" />
    <entlang_Routenkomponente xlink:href="[Verweis auf Routenkomp.]" />
    [...]
```

Die *Route* ist noch nicht zeitlich beendet, daher trägt sie kein End-Datum (gueltig_bis) und referenziert kein *Ereignis*, das zu ihrer Löschung geführt hat (geloescht_von_Ereignis). Eine Vorgängerversion ist ebenfalls nicht angegeben (hat_Vorgaenger_hist_Objekt). Diese optionalen, nicht gesetzten Eigenschaften werden in der XML-Darstellung der *Route* daher nicht aufgeführt.

Hier besteht die *Route* aus drei *Routenkomponenten*, die in den XML-Daten aufgezählt werden.

4.7.2 XML Schema

Im XML Schema werden die Beschreibungen der Eigenschaften der SUPERTYPEs in analoger Weise in die Beschreibung des ENTITYs selbst integriert.

EXPRESS:

```
ENTITY [ENTITY-Name]
SUBTYPE OF ([Name des ersten SUPERTYPEs,
               Name des zweiten SUPERTYPESs, [...,]
               Name des letzten SUPERTYPEs]);
    [...eigene Eigenschaften des ENTITYs...]
END_ENTITY;
```

XML Schema:

```
[...]
<!--  Eigenschaften des SUPERTYPEs [Name des ersten SUPERTYPEs]  -->
[Beschreibung der Eigenschaften des ersten SUPERTYPEs]
<!--  Eigenschaften des SUPERTYPEs [Name des zweiten SUPERTYPEs]  -->
[Beschreibung der Eigenschaften des zweiten SUPERTYPEs]
[...]
<!--  Eigenschaften des SUPERTYPEs [Name des letzten SUPERTYPEs]  -->
[Beschreibung der Eigenschaften des letzten SUPERTYPEs]
<!--  eigene Eigenschaften des ENTITYs [Name des ENTITYs]  -->
[Beschreibung der eigenen Eigenschaften des ENTITYs]
[...]
```

Wir betrachten wieder das **Beispiel** der *Route*.

EXPRESS:

```
ENTITY Route
SUBTYPE OF (ASB_Objekt,Teilnetzkomponente,Routenkomponente,
            historisches_Objekt);
    [...eigene Eigenschaften der Route...]
END_ENTITY;
```

XML Schema:

```
[...]
<!--  Eigenschaften des SUPERTYPEs ASB_Objekt  -->
[...]
<!--  Eigenschaften des SUPERTYPEs Teilnetzkomponente  -->
[...]
<!--  Eigenschaften des SUPERTYPEs Routenkomponente  -->
[...]
<!--  Eigenschaften des SUPERTYPEs historisches_Objekt  -->
<element name="gueltig_von" type="date" />
<element name="gueltig_bis" type="date" minOccurs="0" />
<element name="erzeugt_von_Ereignis" type="okstra:ObjectRefType"
    minOccurs="0" />
<element name="geloescht_von_Ereignis" type="okstra:ObjectRefType"
    minOccurs="0" />
<element name="hat_Vorgaenger_hist_Objekt"
    type="okstra:ObjectRefType" minOccurs="0" />
<!--  eigene Eigenschaften des ENTITYs Route  -->
<element name="entlang_Routenkomponente"
    type="okstra:ObjectRefType" maxOccurs="unbounded" />
[...]
```

Eine Ausnahme bilden Supertypes, die einen Netzbezug herstellen und/oder die Historisierung repräsentieren. Diese werden per `extension` vererbt. Dies empfiehlt sich, um die herausragende Rolle des Netzbezugs und der Historisierung angemessen auszudrücken. Definiert werden `Types` zum Punktobjekt, Streckenobjekt und Bereichsobjekt, jeweils in einer historisierenden und in einer statischen (nicht historisierenden) Variante. Die jeweiligen Subtypen werden in die `substitutionGroup` dieses Netzbezugs eingetragen. Sollte mehr als ein Supertype einen Netzbezug herstellen, so wird der zuerst aufgeführte Supertype per `extension` abgebildet, alle weiteren durch Einbettung wie bei gewöhnlichen Supertypes.

4.8 Abbildung der Geometrie

Neben den bereits genannten strategischen Vorteilen der Modellierung von OKSTRA®-XML als GML-Anwendungsschema (→ Normbasierte Austauschschnittstelle NAS der AdV, Integration in Geodateninfrastrukturen) ist ein weiterer wesentlicher Vorteil die dadurch mögliche Übernahme der standardisierten Geometrierepräsentierung.

Es macht wenig Sinn, eine eigene, OKSTRA®-spezifische Geometriemodellierung für die XML-Codierung festzulegen, wo bereits standardisierte Codierungen verfügbar sind. Grundsätzlich gilt auch, dass es zu überlegen wäre, das derzeitige Geometriemodell des OKSTRA® durch ein Profil des inzwischen von ISO, OGC und der AdV übernommenen Spatial Schemas ISO 19107 abzulösen und so die derzeitige Speziallösung im OKSTRA® durch einen Standard zu ersetzen.

4.8.1 Content Model

Zu unterscheiden sind punktförmige, linienförmige, flächenförmige und volumenförmige Geometrie bzw. Topologie[6]. Diese werden über geometrische bzw. topologische Eigenschaften den OKSTRA®-Objekten zugeordnet.

Im OKSTRA® kann einem Objekt entweder eine geometrische oder eine topologische Darstellung zugeordnet werden, nicht beides auf einmal. Die geometrische Darstellung selbst kann jedoch einer topologischen Darstellung zugeordnet sein und umgekehrt.

Zunächst einige einfache **Beispiele**, im Anschluss daran wird eine genaue Spezifikation getroffen, welche Teile der GML-Strukturen für Geometrie und Topologie im OKSTRA® genutzt werden.

XML-Daten:

```
<Netzknoten>
    <!-- ... -->
    <dargestellt_von_Punkt>
        <gml:Point srsName="urn:adv:crs:DE_DHDN_3GK4">
            <gml:coordinates>
                4373512.25590199,5653729.90479046
            </gml:coordinates>
        </gml:Point>
    </dargestellt_von_Punkt>
    <!-- ... -->
</Netzknoten>

<Nullpunkt>
    <!-- ... -->
    <dargestellt_von_Knoten>
        <gml:Node gml:id="node">
            <gml:pointProperty xlink:href="#point"/>
        </gml:Node>
    </dargestellt_von_Knoten>
    <!-- ... -->
</Nullpunkt>

<Abschnitt>
    <!-- ... -->
    <dargestellt_von_Linie>
        <gml:Curve gml:id="line" srsName="urn:adv:crs:DE_DHDN_3GK4">
```

[6] In GML 3.1.1 lauten die für den OKSTRA® geeigneten Types PointPropertyType, CurvePropertyType, SurfacePropertyType und SolidPropertyType bzw. DirectedNodePropertyType, DirectedEdgePropertyType, DirectedFacePropertyType und DirectedTopoSolidPropertyType.

```
            <gml:segments>
                <gml:LineStringSegment>
                    <gml:coordinates>
                        4373512.25590199,5653729.90479046
                        4373533.01647874,5653733.51509939
                        4373602.10292724,5653755.70699833
                        4373784.10645786,5653820.92257877
                        4373952.27785213,5653880.28765859
                    </gml:coordinates>
                </gml:LineStringSegment>
        </gml:Curve>
    </dargestellt_von_Linie>
    <!-- ... -->
</Abschnitt>

<Streifenbegrenzung>
    <!-- ... -->
    <dargestellt_von_Kante>
        <gml:Edge gml:id="edge">
            <gml:curveProperty xlink:href="#line"/>
        </gml:Edge>
    </dargestellt_von_Kante>
    <!-- ... -->
</Streifenbegrenzung>

<Bundesland>
    <!-- ... -->
    <dargestellt_von_Flaeche>
        <gml:Polygon gml:id="surface"
            srsName="urn:adv:crs:DE_DHDN_3GK4">
            <gml:exterior>
                <gml:Ring>
                    <gml:curveMember xlink:href="#line"/>
                </gml:Ring>
            </gml:exterior>
        </gml:Polygon>
    </dargestellt_von_Flaeche>
    <!-- ... -->
</Bundesland>

<Gemeinde>
```

```
        <!-- ... -->
        <dargestellt_von_Masche>
            <gml:Face gml:id="face">
                <gml:surfaceProperty xlink:href="#surface"/>
            </gml:Face>
        </dargestellt_von_Masche>
        <!-- ... -->
    </Gemeinde>

    <Schicht>
        <!-- ... -->
        <dargestellt_von_Volumen>
            <gml:Solid gml:id="solid" srsName="urn:adv:crs:DE_DHDN_3GK4">
                <gml:interior xlink:href="#surface"/>
                <gml:interior xlink:href="#surface2"/>
            </gml:Solid>
        </dargestellt_von_Volumen>
        <!-- ... -->
    </Schicht>

    <Schicht>
        <!-- ... -->
        <dargestellt_von_Koerper>
            <gml:TopoSolid>
                <gml:directedFace orientation="+" xlink:href="#face"/>
                <gml:directedFace orientation="+" xlink:href="#face2"/>
            </gml:TopoSolid>
        </dargestellt_von_Koerper>
        <!-- ... -->
    </Schicht>
```

Wie oben bereits gesagt wird nicht der volle Umfang der Geometrie-/Topologie-Darstellung aus GML 3.1.1 für OKSTRA®-XML verwendet, sondern ein sinnvolles Profil. Dieses Profil wird im Folgenden präzisiert.

4.8.2 Koordinatenreferenzsysteme

In GML 3.1.1 wird das verwendete Koordinatenreferenzsystem im `attribute srsName` angegeben. Dieses `attribute` kann an verschiedenen Stellen und auf verschiedenen

Hierarchieebenen in geometrischen `elements` vorkommen. Für OKSTRA®-XML beziehen wir uns hier auf die Version 3 der GeoInfoDok der AdV[7]. Als Beispiel geben wir hier Darstellungen für Koordinatenreferenzsysteme an:

Beschreibung	Pos. Ref. System	Coord. System	Code
3° Gauß-Krüger (Bessel-Ellipsoid) - 2. Streifen	DHDN	3GK2	DE_DHDN_3GK2
3° Gauß-Krüger (Bessel-Ellipsoid) - 3. Streifen	DHDN	3GK3	DE_DHDN_3GK3
3° Gauß-Krüger (Bessel-Ellipsoid) - 4. Streifen	DHDN 40-83	3GK4	DE_DHDN_3GK4 DE_40-83_3GK4
3° Gauß-Krüger (Bessel-Ellipsoid) - 5. Streifen	DHDN 40-83	3GK5	DE_DHDN_3GK5 DE_40-83_3GK5
3° Gauß-Krüger (Krassowski-Ellipsoid) - 4.Streifen	42-83	3GK4	DE_42-83_3GK4
3° Gauß-Krüger (Krassowski-Ellipsoid) - 5.Streifen	42-83	3GK5	DE_42-83_3GK5
6° Gauß-Krüger (Krassowski-Ellipsoid) - 2.Streifen	42-83	6GK2	DE_42-83_6GK2
6° Gauß-Krüger (Krassowski-Ellipsoid) - 3.Streifen	42-83	6GK3	DE_42-83_6GK3
WGS 84	WGS84	X-Y-Z	WGS84_X-Y-Z
ETRS 89	ETRS89	Lat-Lon-h UTM32 UTM33	ETRS89_Lat-Lon-h ETRS89_UTM32 ETRS89_UTM33

Der angegebene Code wird als URN mit dem URN-Namespace `urn:adv:crs` im `attribute srsName` eingetragen, also z.B. `urn:adv:crs:DE_DHDN_3GK4`.

Weitere Erläuterungen finden sich im NIAM-Dokument zum Geometrieschema (D018).

Für die aktuell gültige offizielle Festlegung dieser Codes verweisen wir auf die jeweils aktuellen Veröffentlichungen der AdV.

4.8.3 Profil für punktförmige Geometrie und Topologie

Punktförmige Geometrieeigenschaften werden als `element` mit Typ `gml:PointPropertyType` abgebildet. Dies entspricht dem *Punkt* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Point` gegeben. Dieses `child element` enthält entweder ein `gml:pos element` oder ein `gml:coordinates element`. Alternativ kann auf einen anderen Punkt innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung eines *Punktes* durch `gml:pos element`:

[7] Die Version 3 der GeoInfoDok wurde am 1. April 2004 veröffentlicht.

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Punkt>
    <gml:Point {gml:id="[XML-Id des Punktes]"}
               {srsName="[Code für Koordinatensystem]"}>
        <gml:pos {srsName="[Code für Koordinatensystem]"}
                 {srsDimension="[Dimension der Koordinaten]"}>
            11.1 12.2
        </gml:pos>
    </gml:Point>
</dargestellt_von_Punkt>
<!-- ... -->
```

Bei dieser Darstellung werden im `gml:pos element` zwei oder drei Gleitkommawerte durch " " getrennt und mit einem "." als Dezimalzeichen angegeben. Das verwendete Koordinatensystem wird im `gml:Point element` angegeben. Zusätzlich kann im `attribute srsDimension` die Dimension der Koordinate als Hinweis für einen Parser angegeben werden. In diesem Fall muss die Anzahl der Gleitkommawerte mit dieser Angabe übereinstimmen.

Dem `gml:Point` kann im `attribute gml:id` eine XML-Id für die Referenzierung von einer anderen Stelle aus gegebenen werden.

An mindestens einer Stelle innerhalb des `gml:Point elements` sollte ein `srsName` angegeben werden.

Darstellung eines *Punktes* durch `gml:coordinates element`:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Punkt>
    <gml:Point {gml:id="[XML-ID des Punktes]"}
               srsName="[Code für Koordinatensystem]">
        <gml:coordinates>1.0,2.0</gml:coordinates>
    </gml:Point>
</dargestellt_von_Punkt>
<!-- ... -->
```

Bei dieser Darstellung werden im `gml:coordinates element` zwei oder drei Gleitkommawerte durch "," getrennt und mit einem "." als Dezimalzeichen angegeben. Das verwendete Koordinatensystem wird im `gml:Point element` angegeben.

Dem `gml:Point` kann im `attribute gml:id` eine XML-Id für die Referenzierung von einer anderen Stelle aus gegebenen werden.

Darstellung eines *Punktes* durch Verweis auf einen anderen *Punkt*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Punkt xlink:href="#[XML-Id eines Punktes]"/>
<!-- ... -->
```

Bei dieser Darstellung wird auf einen `gml:Point` an einer anderen Stelle im selben OKSTRA®-XML Dokument verwiesen.

Durch Anpassung des OKSTRA® an die ASB-Netzdaten, Stand September 2002, können für *Punkte* die Herkunft und die Genauigkeit der Koordinate angegeben werden. Die Standardisierung solcher Angaben in GML/ISO ist noch im Fluss. Bis dort eine Festlegung getroffen wird, werden diese Angaben über allgemeine GML-Metadaten zum `Point` abgebildet. Hierfür wurde das Metadaten `element okstra:KoordinatenMetadaten` definiert und die Schlüsseltabelle `Koordinaten-herkunft` aus dem Geometrieschema des OKSTRA® in OKSTRA®-XML übernommen. Die „Koordinatengenauigkeit" wird in `Meter` angegeben. **Beispiel**:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Punkt>
    <gml:Point {gml:id="[XML-ID des Punktes]"}
               srsName="[Code für Koordinatensystem]">
        <gml:metaDataProperty>
            <okstra:KoordinatenMetadaten>
                <okstra:Herkunft>
                    <okstra:Koordinatenherkunft>
                        <okstra:Kennung>03</okstra:Kennung>
                        <okstra:Langtext>ATKIS</okstra:Langtext>
                    </okstra:Koordinatenherkunft>
                </okstra:Herkunft>
                <okstra:Genauigkeit>0.1</okstra:Genauigkeit>
            </okstra:KoordinatenMetadaten>
        </gml:metaDataProperty>
        <gml:coordinates>1.0,2.0</gml:coordinates>
    </gml:Point>
</dargestellt_von_Punkt>
<!-- ... -->
```

Als Werte für die Herkunft sind die im OKSTRA® im Schema Geometrieschema für die Schlüsseltabelle *Koordinatenherkunft* festgelegten Werte zulässig.

Punktförmige Topologieeigenschaften werden als `element` mit Typ `gml:DirectedNode-PropertyType` abgebildet. Dies entspricht dem *Knoten* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Node` gegeben. Dieses `child element` kann ein `gml:pointProperty element` enthalten mit einem Verweis auf eine Realisierung als *Punkt*.

Alternativ kann auf einen anderen *Knoten* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Ein *isolierter_Knoten* kann zusätzlich einen Verweis auf die *Masche* enthalten, in der er liegt:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Knoten>
    <!-- isolierter Knoten -->
    <gml:Node {gml:id="[XML-Id des Knotens]"}>
        <!-- umgebende Masche -->
        {<gml:container xlink:href="#[XML-Id einer Masche]"/>}
        <!-- Realisierung als Punkt -->
        {<gml:pointProperty xlink:href="#point"/>}
    </gml:Node>
</dargestellt_von_Knoten>
<!-- ... -->
```

Dem `gml:Node` kann im `attribute gml:id` eine XML-Id für die Referenzierung von einer anderen Stelle aus gegebenen werden.

Ein *nicht_isolierter_Knoten* kann zusätzlich Verweise auf die *Kanten* enthalten, die er vorne oder hinten begrenzt:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Knoten>
    <!-- nicht isolierter Knoten -->
    <gml:Node [gml:id="[XML-Id des Knotens]"]>
        <!-- Angabe von begrenzten Kanten -->
        {<gml:directedEdge orientation="-"
            xlink:href="#[XML-Id einer vorne begrenzten Kante"/>}
        {<gml:directedEdge orientation="-"
            xlink:href="#[XML-Id einer vorne begrenzten Kante"/>}
        {<gml:directedEdge orientation="+"
            xlink:href="#[XML-Id einer hinten begrenzten Kante"/>}
        {<gml:directedEdge orientation="+"
            xlink:href="#[XML-Id einer hinten begrenzten Kante"/>}
        <!-- Realisierung als Punkt -->
        {<gml:pointProperty xlink:href="#point"/>}
```

```
        </gml:Node>
    </dargestellt_von_Knoten>
    <!-- ... -->
```

Die vorne begrenzten *Kanten* werden mit `orientation="-"` angegeben, die hinten begrenzten *Kanten* mit `orientation="+"`.

Dem `gml:Node` kann im `attribute gml:id` eine XML-Id für die Referenzierung von einer anderen Stelle aus gegebenen werden.

Darstellung eines *Knotens* durch Verweis auf einen anderen *Knoten*:

XML-Daten:

```
    <!-- ... -->
    <dargestellt_von_Knoten xlink:href="#[XML-Id eines Knotens]"/>
    <!-- ... -->
```

Bei dieser Darstellung wird auf einen `gml:Node` an einer anderen Stelle im selben OKSTRA®-XML Dokument verwiesen.

4.8.4 Profil für linienförmige Geometrie und Topologie

Linienförmige Geometrieeigenschaften werden als `element` mit Typ `gml:CurvePropertyType` abgebildet. Dies entspricht der *Linie* im OKSTRA®.

Eine *Linie* im OKSTRA® ist per Definition zusammenhängend. Mehrere nicht zusammenhängende Linien für ein linienförmiges Objekt müssen daher durch mehrere Objekte der Klasse *Linie* dargestellt werden.

Für OKSTRA®-XML wird diesem `element` im allgemeinen Fall genau ein `child element gml:Curve` gegeben. Dieses `child element` enthält genau ein `gml:segments element`, das die verschiedenen Segmente der Linie enthält. Als Spezialfall kann für Polygonzüge statt der `gml:Curve` auch der einfachere `gml:LineString` verwendet werden (orientierbare Linien und Flächen sind zwar mit dem Geometrieschema des OKSTRA® möglich, werden aber zunächst nicht berücksichtigt).

Alternativ kann auf eine andere *Linie* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Der Begriff des Segments wird hier für einen Teil einer Linie mit unveränderter Interpolationsmethode verwendet. Dies entspricht einem oder mehreren *Linienelement_3D*-Objekten des OKSTRA®. Eine Folge von geraden *Linienelementen* kann beispielsweise in einem einzigen `gml:coordinates element` dargestellt werden.

Die Darstellung der Kontrollpunkte eines Segments geschieht stets auf folgende Weise:

Entweder werden die Kontrollpunkte durch eine Anzahl von `gml:pos/gml:pointRep elements` dargestellt:

XML-Daten:

```
    <!-- ... -->
```

```
<gml:pointRep xlink:href="#[XML-Id eines Punktes]"/>
<gml:pos {srsName="[Code des Koordinatensystems]"}
            {dimension="[Dimension der Koordinate]"}>
    [Wert der Koordinate, Dimensionen durch Leerzeichen getrennt,
        Punkt als Dezimalzeichen]
</gml:pos>
<!-- ... -->
```

Bei dieser Alternative können `gml:pos` und `gml:pointRep elements` beliebig gemischt werden.

Oder die Kontrollpunkte werden kompakt als `gml:coordinates element` dargestellt. In diesem Fall muss sich das Koordinatensystem aus den übergeordneten `elements` ergeben.

XML-Daten:

```
<!-- ... -->
<gml:coordinates>
    [Werte der Koordinaten, durch Leerzeichen getrennt,
        Dimensionen durch Komma getrennt, Punkt als Dezimalzeichen]
</gml:coordinates>
<!-- ... -->
```

Zwischen diesen beiden Alternativen kann für jedes Segment einer Linie gewählt werden.

Im allgemeinen Fall wird eine *Linie* durch das `gml:Curve element` des `gml:CurvePropertyType` dargestellt. Um die Darstellung nicht zu sehr zu zerteilen, werden alle Arten von *Linienelement_3D*-Objekten des OKSTRA® in der nachfolgenden Übersicht berücksichtigt. Genauere Erläuterungen folgen anschließend.

Darstellung einer *Linie* durch `gml:Curve`:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Linie>
    <gml:Curve {gml:id="[XML-Id der Linie]"}
                {srsName="[Code des Koordinatensystems]"}>
        <gml:segments>
            <!-- gerades_Linienelement, ggf. mehrere -->
            <gml:LineStringSegment
                {srsName="[Code für Koordinatensystem]"}>
                [Darstellung von 2 oder mehr Kontrollpunkten, s.o.]
                [jedes aufeinanderfolgende Paar def. ein gerades LE]
            </gml:LineStringSegment>
```

```
                <!-- Kreisbogen -->
                <gml:Arc {srsName="[Code für Koordinatensystem]"}>
                    [Darstellung von 3 Kontrollpunkten, s.o.]
                    [Anfangspunkt – Zwischenpunkt – Endpunkt]
                </gml:Arc>
                <!-- Linienelement_Spline -->
                <gml:CubicSpline
                    {srsName="[Code für Koordinatensystem]"}>
                    [Darstellung von 3 oder mehr Kontrollpunkten, s.o.]
                    [Anfangspunkt – [Stuetzpunkte] – Endpunkt]
                    <gml:vectorAtStart>
                        [normalisierter Tangentialvektor am Anfang]
                    </gml:vectorAtStart>
                    <gml:vectorAtEnd>
                        [normalisierter Tangentialvektor am Ende]
                    </gml:vectorAtEnd>
                </gml:CubicSpline>
            </gml:segments>
        </gml:Curve>
    </dargestellt_von_Linie>
    <!-- ... -->
```

Die Reihenfolge der Darstellung der verschiedenen *Linienelement_3D*-Objekte ist relevant.

Bei der Darstellung von geraden *Linienelementen* können aufeinanderfolgende gerade *Linienelemente* in einem Segment zusammengefasst werden. Als `interpolation` wird grundsätzlich der Default `linear` angenommen. Abweichende Festlegungen sind nicht zulässig.

Der Kreisbogen benötigt wie im OKSTRA® exakt 3 Kontrollpunkte. Die Punkte müssen in der Reihenfolge Anfangspunkt – Zwischenpunkt – Endpunkt gegeben sein. Als `interpolation` wird grundsätzlich der Default `circularArc3Points` angenommen. Abweichende Festlegungen sind nicht zulässig.

Als Spline ist nur der kubische Spline zulässig. Es müssen mindestens 3 Kontrollpunkte vorhanden sein, d.h. mindestens ein innerer Stützpunkt. Als Randbedingungen werden wie in der internationalen Standardisierung üblich die normalisierten Tangentialvektoren am Anfang und am Ende angegeben. *Linienelement_Spline*-Objekte des OKSTRA® mit tangentialem Anschluss werden in OKSTRA®-XML zu einem einzigen kubischen Spline verbunden, geschlossene Splines werden durch evtl. Anfügen der Anfangskoordinate und Übernahme des Tangentialvektors vom Anfang geschlossen. Als `interpolation` wird grundsätzlich der Default `cubicSpline` angenommen. Abweichende Festlegungen sind nicht zulässig.

Anmerkung: Die Angabe von Krümmungen als Randbedingungen bei Splines wird in OKSTRA®-XML nicht unterstützt. Sie stellt ebenso wie die Eigenschaften "tangentialer Anschluss" und "geschlossen" des *Linienelement_Spline* im OKSTRA® eine Redundanz dar. Diese Redundanzen sollten bei einer Überarbeitung des Geometrieschemas behoben werden.

Spezialfall der Darstellung einer *Linie* durch `gml:LineString`:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Linie>
    <gml:LineString {srsName="[Code des Koordinatensystems]"}>
        [Darstellung der Kontrollpunkte des Polygonzugs]
    </gml:LineString>
</dargestellt_von_Linie>
<!-- ... -->
```

Diese Darstellung kann für den Spezialfall eines Polygonzugs verwendet werden, d.h. wenn die *Linie* nur aus geraden *Linienelementen* besteht.

Die Darstellung der Kontrollpunkte erfolgt wie zu Beginn dieses Unterkapitels erläutert.

Darstellung durch Verweis auf eine andere *Linie*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Linie xlink:href="#[XML-Id einer Linie]"/>
<!-- ... -->
```

Bei dieser Darstellung wird auf eine `gml:Curve` oder einen `gml:LineString` an einer anderen Stelle im selben OKSTRA®-XML Dokument verwiesen.

Linienförmige Topologieeigenschaften werden als `element` mit Typ `gml:DirectedEdgePropertyType` abgebildet. Dies entspricht der *Kante* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Edge` gegeben. Dieses `child element` enthält genau zwei `gml:directedNode elements` für die begrenzenden *Knoten*, je eines mit `orientation="-"` und `orientation="+"`. Ferner können die von dieser *Kante* begrenzten *Maschen* angegeben werden. Es kann ein `gml:curveProperty element` angegeben werden mit einem Verweis auf eine Realisierung als *Linie*.

Alternativ kann auf eine andere *Kante* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung durch `gml:Edge`:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Kante>
    <gml:Edge {gml:id="[XML-Id der Kante]"}
                {srsName="[Code des Koordinatensystems]"}>
        <!-- Knoten am Anfang -->
        <gml:directedNode orientation="-"
```

```
                xlink:href="#[XML-Id eines nicht isolierten Knotens]"/>
            <!-- Knoten am Ende -->
            <gml:directedNode orientation="+"
                xlink:href="#[XML-Id eines nicht isolierten Knotens]"/>
            <!-- ggf. begrenzte Maschen -->
            {<gml:directedFace orientation="+"
                xlink:href="#[XML-Id einer Masche]"/>}
            {<gml:directedFace orientation="+"
                xlink:href="#[XML-Id einer Masche]"/>}
            <!-- Realisierung als Linie -->
            {<gml:curveProperty xlink:href="#[XML-Id einer Linie]"/>}
        </gml:LineString>
    </dargestellt_von_Kante>
    <!-- ... -->
```

Orientierung von *Kanten* als Begrenzung von *Maschen* ist im OKSTRA® unbekannt. Daher sind *Kanten* stets als positiv orientierte Begrenzung anzunehmen bzw. in eine solchen umzurechnen. Dieser Mangel ist bei der Überarbeitung des Geometrieschemas zu beheben.

Ein begrenzendes `gml:directedNode element` kann auch ein eingebettetes `gml:Node element` enthalten.

Darstellung durch Verweis auf eine andere *Kante*:

XML-Daten:

```
    <!-- ... -->
    <dargestellt_von_Kante xlink:href="#[XML-Id einer Kante]"/>
    <!-- ... -->
```

4.8.5 Profil für flächenförmige Geometrie und Topologie

Flächenförmige Geometrieeigenschaften werden als `element` mit Typ `gml:SurfacePropertyType` abgebildet. Dies entspricht dem *Flächenelement* im OKSTRA®. *Komplexe_Flächen* im OKSTRA® sind derzeit einfach addierte *Flächenelemente*. Dies kann bis zu einer Überarbeitung des Geometrieschemas im OKSTRA® durch mehrere *Flächenelemente* im *Flächenobjekt_Modell* dargestellt werden.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Polygon` gegeben. Dieses `child element` enthält ein `gml:exterior element` für den äußeren Rand des *Flächenelements* (*Linienfunktion* im OKSTRA® = 0, einschließende Linie) und beliebig viele `gml:interior elements` für die Ränder von Ausschlüssen aus dem *Flächenelement* (*Linienfunktion* im OKSTRA® = 1, ausschließende Linie).

Alternativ kann auf eine andere *Fläche*, genauer *Flächenelement*, innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung durch `gml:Polygon`:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Flaeche>
    <gml:Polygon {gml:id="[XML-Id des Flaechenelements]"}
                 {srsName="[Code des Koordinatensystems]"}>
        <!-- einschliessender Rand -->
        <gml:exterior>
            <gml:Ring {gml:id="[XML-Id des Flaechenelements]"}
                      {srsName="[Code des Koordinatensystems]"}>
                <gml:curveMember>
                    [Darstellung einer Linie]
                </gml:curveMember>
            </gml:Ring>
        </gml:exterior>
        <!-- ausschliessende Raender -->
        <gml:interior>
            <gml:Ring {gml:id="[XML-Id des Flaechenelements]"}
                      {srsName="[Code des Koordinatensystems]"}>
                <gml:curveMember>
                    [Darstellung einer Linie]
                </gml:curveMember>
            </gml:Ring>
        </gml:interior>
        <gml:interior>
            <gml:LinearRing {gml:id="[XML-Id des Flaechenelements]"}
                      {srsName="[Code des Koordinatensystems]"}>
                [Darstellung eines Polygonzugs]
            </gml:LinearRing>
        </gml:interior>
    </gml:Polygon>
</dargestellt_von_Flaeche>
<!-- ... -->
```

Die oben genannte Darstellung einer Linie geschieht analog zur Darstellung von linienförmiger Geometrie (siehe 4.8.4). Das `gml:curveMember element` tritt dabei an die Stelle des `elements dargestellt_von_Linie`. Auf diese Weise kann als `gml:curveMember element` auch ein `gml:Curve` oder `gml:LineString element` an einer anderen Stelle des OKSTRA®-XML Dokuments referenziert werden.

Analog zur Darstellung einer Linie kann auch hier im Spezialfall eines Polygonzugs als Randlinie statt des allgemeinen `gml:Ring elements` ein einfacheres `gml:LinearRing element` verwendet werden, wie in obigem Beispiel bereits verwendet. Hierbei tritt `gml:LinearRing` an die Stelle des `gml:LineString` aus der Darstellung linienförmiger Geometrie. Ein `gml:LinearRing` muss aus mindestens vier Kontrollpunkten bestehen.

Darstellung durch Verweis auf eine andere *Fläche*, genauer *Flächenelement*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Flaeche xlink:href="#[XML-Id einer Flaeche]"/>
<!-- ... -->
```

Flächenförmige Topologieeigenschaften werden als `element` mit Typ `gml:DirectedFacePropertyType` abgebildet. Dies entspricht der *Masche* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Face` gegeben. Dieses `child element` enthält mindestens ein `gml:directedEdge element` für die begrenzenden Kanten. Orientierung von *Kanten* als Begrenzung von *Maschen* ist im OKSTRA® unbekannt. Daher sind *Kanten* stets als positiv orientierte Begrenzung anzunehmen bzw. in solche umzurechnen. Es kann ein `gml:surfaceProperty element` angegeben werden mit einem Verweis auf eine Realisierung als *Fläche*, genauer *Flächenelement*. Ferner können die in dieser *Masche* enthaltenen *isolierten_Knoten* angegeben werden.

Alternativ kann auf eine andere *Masche* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung durch `gml:Face`:

XML-Daten:

```
<!-- ... -->
<okstra:dargestellt_von_Masche>
    <gml:Face {gml:id="[XML-Id der Masche]"}
                {srsName="[Code des Koordinatensystems]"}>
        <!-- ggf. enthaltene isolierte Knoten -->
        {<gml:isolated xlink:href="#[XML-Id eines isol. Knotens]"/>}
        {<gml:isolated xlink:href="#[XML-Id eines isol. Knotens]"/>}
        <!-- begrenzende Kanten -->
        <gml:directedEdge xlink:href="#[XML-Id einer Kante]"/>
        <gml:directedEdge xlink:href="#[XML-Id einer Kante]"/>
        <!-- Realisierung als Flaeche -->
        {<gml:surfaceProperty xlink:href="#[XML-Id einer Flaeche]"/>}
    </gml:Face>
</okstra:dargestellt_von_Masche>
```

```
<!-- ... -->
```

Ein `gml:directedEdge element` kann auch ein eingebettetes `gml:Edge element` enthalten.

Anmerkung: Eine Angabe der begrenzten *Körper* ist derzeit nicht sinnvoll, da die vorliegende Version von GML 3.00 einen eingebetteten `gml:TopoSolid` zwingend vorschreibt.

Darstellung durch Verweis auf eine andere *Masche*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Masche xlink:href="#[XML-Id einer Masche]"/>
<!-- ... -->
```

4.8.6 Profil für volumenförmige Geometrie und Topologie

Volumenförmige Geometrieeigenschaften werden als `element` mit Typ `gml:SolidPropertyType` abgebildet. Dies entspricht dem *Volumen* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:Solid` gegeben. Dieses `child element` enthält beliebig viele `gml:interior elements` für die begrenzenden *Flächen*. Dieses Vorgehen wird gewählt, da das derzeitige Geometrieschema des OKSTRA® keine äußeren und inneren Begrenzungsflächen unterscheidet. Dieser Mangel muss im Zuge einer Überarbeitung des Geometrieschemas behoben werden.

Alternativ kann auf ein anderes *Volumen* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung durch `gml:Solid`:

XML-Daten:

```
<!-- ... -->
<okstra:dargestellt_von_Volumen>
    <gml:Solid {gml:id="[XML-Id des Volumens]"}
               {srsName="[Code des Koordinatensystems]"}>
        <!-- begrenzende Flaechen -->
        <gml:interior xlink:href="#[XML-Id einer Flaeche]"/>
        <gml:interior xlink:href="#[XML-Id einer Flaeche]"/>
    </gml:Solid>
</okstra:dargestellt_von_Volumen>
<!-- ... -->
```

Die oben genannte Darstellung einer *Fläche* geschieht analog zur Darstellung von flächenförmiger Geometrie (siehe 4.8.5). Das `gml:interior element` tritt dabei an die Stelle des `elements`

`dargestellt_von_Flaeche`. Auf diese Weise kann im `gml:interior element` auch ein `gml:Polygon element` direkt eingebettet werden.

Darstellung durch Verweis auf ein anderes *Volumen*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Volumen xlink:href="#[XML-Id eines Volumens]"/>
<!-- ... -->
```

Volumenförmige Topologieeigenschaften werden als `element` mit Typ `gml:DirectedTopo-SolidPropertyType` abgebildet. Dies entspricht dem *Körper* im OKSTRA®.

Für OKSTRA®-XML wird diesem `element` genau ein `child element gml:TopoSolid` gegeben. Dieses `child element` enthält mindestens ein `gml:directedFace element` für die begrenzenden *Maschen*. Orientierung von *Maschen* als Begrenzung von *Körpern* ist im OKSTRA® unbekannt. Daher sind *Maschen* stets als positiv orientierte Begrenzung anzunehmen bzw. in solche umzurechnen.

Alternativ kann auf einen anderen *Körper* innerhalb desselben OKSTRA®-XML Dokuments verwiesen werden.

Darstellung durch `gml:TopoSolid`:

XML-Daten:

```
<!-- ... -->
<okstra:dargestellt_von_Koerper>
    <gml:TopoSolid {gml:id="[XML-Id des Koerpers]"}
                {srsName="[Code des Koordinatensystems]"}>
        <!-- begrenzende Maschen -->
        <gml:directedFace xlink:href="#[XML-Id einer Masche]"/>
        <gml:directedFace xlink:href="#[XML-Id einer Masche]"/>
    </gml:TopoSolid>
</okstra:dargestellt_von_Koerper>
<!-- ... -->
```

Die oben genannte Darstellung einer *Masche* geschieht analog zur Darstellung von flächenförmiger Topologie (siehe 4.8.5). Das `gml:directedFace element` tritt dabei an die Stelle des `elements dargestellt_von_Masche`. Auf diese Weise kann im `gml:directedFace element` auch ein `gml:Face element` eingebettet werden.

Darstellung durch Verweis auf einen anderen *Körper*:

XML-Daten:

```
<!-- ... -->
<dargestellt_von_Koerper xlink:href="#[XML-Id eines Koerpers]"/>
<!-- ... -->
```

4.8.7 XML Schema

In XML Schema drückt sich eine entsprechende geometrische Eigenschaft durch ein `element` mit einem in GML definierten `complexType` aus, hier jeweils für punktförmige, linienförmige, flächenförmige und volumenförmige Geometrie/Topologie:

XML Schema:

```
<complexType name="[ENTITY-Name]Type">
    [ ... ]
    <!-- Darstellung punktfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Punkt"
            type="gml:PointPropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Knoten"
        type="gml:DirectedNodePropertyType" maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>

<complexType name="[ENTITY-Name]Type">
    [ ... ]
    <!-- Darstellung linienfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Linie"
            type="gml:CurvePropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Kante"
        type="gml:DirectedEdgePropertyType" maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>

<complexType name="[ENTITY-Name]Type">
    [ ... ]
    <!-- Darstellung flaechenfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Flaeche"
```

```
                type="gml:SurfacePropertyType" maxOccurs="unbounded"/>
            <element name="dargestellt_von_Masche"
            type="gml:DirectedFacePropertyType" maxOccurs="unbounded"/>
        </choice>
        [ ... ]
    </complexType>

    <complexType name="[ENTITY-Name]Type">
        [ ... ]
        <!-- Darstellung volumenfoermiger Geometrie/Topologie -->
        <choice minOccurs="0">
            <element name="dargestellt_von_Volumen"
                type="gml:SolidPropertyType" maxOccurs="unbounded"/>
            <element name="dargestellt_von_Koerper"
                type="gml:DirectedTopoSolidPropertyType"
                maxOccurs="unbounded"/>
        </choice>
        [ ... ]
    </complexType>
```

Die Metadaten zu Herkunft und Genauigkeit der Koordinate zum *Punkt* sind wie folgt definiert:

XML Schema:

```
    <element name="KoordinatenMetadaten"
        type="okstra:KoordinatenMetadatenType"
        substitutionGroup="gml:_MetaData"/>
    <complexType name="KoordinatenMetadatenType">
        <complexContent>
            <extension base="gml:AbstractMetaDataType">
                <sequence>
                    <element name="Herkunft"
                        type="okstra:KoordinatenherkunftPropertyType"
                        minOccurs="0"/>
                    <element name="Herkunft_Land"
                         type="okstra:Koordinatenherkunft_
                               LandPropertyType" minOccurs="0"/>
                    <element name="Genauigkeit" type="okstra:Meter"
                        minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
```

```
    </complexType>
    <complexType name="KoordinatenherkunftPropertyType">
        <complexContent>
            <extension base="okstra:KeyValuePropertyType">
                <sequence>
                    <element name="Koordinatenherkunft"
                        type="okstra:KoordinatenherkunftType"
                        minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="Koordinatenherkunft"
        type="okstra:KoordinatenherkunftType"
        substitutionGroup="okstra:_KeyValue"/>
    <complexType name="KoordinatenherkunftType">
        <complexContent>
            <extension base="okstra:AbstractKeyValueType">
                <sequence>
                    <element name="Kennung"
                        type="integer"/>
                    <element name="Langtext"
                        type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="Koordinatenherkunft_LandPropertyType">
        <complexContent>
            <extension base="okstra:KeyValuePropertyType">
                <sequence>
                    <element name="Koordinatenherkunft_Land"
                        type="okstra:Koordinatenherkunft_LandType"
                        minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="Koordinatenherkunft_Land"
        type="okstra:Koordinatenherkunft_LandType"
        substitutionGroup="okstra:_OKSTRAObjekt"/>
```

```
<complexType name="Koordinatenherkunft_LandType">
    <complexContent>
        <extension base="okstra:AbstractOKSTRAObjektType">
            <sequence>
                <element name="Kennung"
                    type="string"/>
                <element name="Langtext"
                    type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

Das `element KoordinatenMetadaten` wird in einer `gml:metaDataProperty` in den `gml:Point` eingetragen. Für ein Beispiel siehe 4.8.3.

XML Schema:

```
<complexType name="NetzknotenType">
    [ ... ]
    <!-- Darstellung punktfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Punkt"
            type="gml:PointPropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Knoten"
        type="gml:DirectedNodePropertyType" maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>

<complexType name="AbschnittType">
    [ ... ]
    <!-- Darstellung linienfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Linie"
            type="gml:CurvePropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Kante"
        type="gml:DirectedEdgePropertyType" maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>
```

```
<complexType name="BundeslandType">
    [ ... ]
    <!-- Darstellung flaechenfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Flaeche"
            type="gml:SurfacePropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Masche"
        type="gml:DirectedFacePropertyType" maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>

<complexType name="AufbauschichtType">
    [ ... ]
    <!-- Darstellung volumenfoermiger Geometrie/Topologie -->
    <choice minOccurs="0">
        <element name="dargestellt_von_Volumen"
            type="gml:SolidPropertyType" maxOccurs="unbounded"/>
        <element name="dargestellt_von_Koerper"
            type="gml:DirectedTopoSolidPropertyType"
            maxOccurs="unbounded"/>
    </choice>
    [ ... ]
</complexType>
```

4.9 Abbildung von Schlüsseltabellen

Schlüsseltabellen im OKSTRA® sind konzeptionell Kombinationen aus einer eindeutigen Kennung und einem oder mehreren (Text-)Attributen. In EXPRESS wird eine Schlüsseltabelle als ENTITY modelliert, hat aber für den OKSTRA® normalerweise Attributcharakter. Die möglichen Kennungen für ein Element einer Schlüsseltabelle sind im OKSTRA® festgelegt.

In OKSTRA®-XML kann der attributive Charakter von Schlüsseltabellen stärker betont werden. So ist es sinnvoll, Einträge aus Schlüsseltabellen analog zu den konzeptionellen Objekten in das verwendende Objekt einzubetten statt zu referenzieren. Ebenso muss aber die Möglichkeit gegeben sein, einen Eintrag aus einer Schlüsseltabelle einmal zentral zu definieren und auf diese Instanz zu referenzieren. Darüber hinaus ist es zur Abfrage oder Fortschreibung von Schlüsseltabellen-Einträgen mit Webservices aus technischen Gründen nötig, Schlüsseltabellen-Einträge (wie „richtige" OKSTRA®-Objekte) als GML-Features zu definieren.

4.9.1 Content Model

In den XML-Daten wird die Kennung eines Schlüsseltabellen-Eintrags in die XML-ID integriert. Die ID wird zusammengesetzt aus dem Namen der Schlüsseltabelle und der Kennung.

Da verschiedene OKSTRA®-Schlüsseltabellen über eine unterschiedliche Anzahl von Attributen und unterschiedliche Attributnamen verfügen können, besitzt das XML Schema für jede Schlüsseltabelle einen eigenen Type. Der Verweis auf einen Schlüsseltabellen-Eintrag erfolgt ähnlich zur Darstellung von Relationen. Alternativ kann ein Schlüsseltabellen-Eintrag direkt in das verwendende Objekt eingebettet werden. Dabei kann entweder der gesamte Schlüsseltabellen-Eintrag (mit allen seinen Attributen) eingebettet oder eine Kompaktdarstellung verwendet werden, bei der nur die Kennung eines Schlüsseltabellen-Eintrags (und, falls gewünscht, der Name der Schlüsseltabelle) angegeben wird. Bei beiden Einbettungsformen ist keine Navigation mehr erforderlich. Die verschiedenen Varianten sind in der folgenden Beschreibung für eine typische OKSTRA®-Schlüsseltabelle mit den Pflichtattributen „Kennung" und „Langtext" aufgeführt.

EXPRESS-CTE:

```
#[CTE-Id] = [Name der Schlüsseltabelle] ( [Kennung des Eintrags],
                          [Langtext des Schlüsseltabelleneintrags] );
```

XML-Daten:

```
<okstraObjekt>
    <[Name der Schlüsseltabelle]
        {gml:id="[Name Schlüsseltabelle].[Kennung des Eintrags]"}>
        <Kennung>[Kennung des Eintrags]</Kennung>
        <Langtext>[Langtext des Schl.tabelleneintrags]</Langtext>
    </[Name der Schlüsseltabelle]>
</okstraObjekt>

<[Objekt mit Schlüsseltabellen-Attribut]>
    [...]
    <[Name Schlüsseltabellen-Attribut]
        xlink:href="[uri des Schlüsseltabellen-Eintrags]"/>
    [...]
</[Objekt mit Schlüsseltabellen-Attribut]>

<[Objekt mit Schlüsseltabellen-Attribut]>
    [...]
    <[Name Schlüsseltabellen-Attribut]>
        <[Name der Schlüsseltabelle]>
            <Kennung>[Kennung des Eintrags]</Kennung>
            <Langtext>[Langtext Schl.tabelleneintrag]</Langtext>
        </[Name der Schlüsseltabelle]>
    </[Name Schlüsseltabellen-Attribut]
    [...]
</[Objekt mit Schlüsseltabellen-Attribut]>

<[Objekt mit Schlüsseltabellen-Attribut]>
```

```
        [...]
        <[Name Schlüsseltabellen-Attribut]
            {Objektklasse=[Name der Schlüsseltabelle]}
            Kennung=[Kennung des Eintrags]/>
        [...]
    </[Objekt mit Schlüsseltabellen-Attribut]>
```

Als **Beispiel** betrachten wir den *Achselementtyp* im OKSTRA®. Wieder sind die verschiedenen Alternativen (Verweis, Einbettung und Kompaktdarstellung) aufgeführt:

EXPRESS-CTE:

```
    #200 = Achselementtyp ('1', 'Gerade' );
```

XML-Daten:

```
    <okstraObjekt>
        <Achselementtyp gml:id="Achselementtyp.1">
            <Kennung>1</Kennung>
            <Langtext>Gerade</Langtext>
        </Achselementtyp>
    </okstraObjekt>

    <Achselement>
        [...]
        <Elementtyp xlink:href="Achselementtyp.1"/>
        [...]
    </Achselement>

    <Achselement>
        [...]
        <Elementtyp>
            <Achselementtyp>
                <Kennung>1</Kennung>
                <Langtext>Gerade</Langtext>
            </Achselementtyp>
        </Elementtyp>
        [...]
    </Achselement>

    <Achselement>
        [...]
        <Elementtyp Objektklasse="Achselementtyp" Kennung="1"/>
```

```
        [...]
    </Achselement>
```

4.9.2 XML Schema

Die Darstellung der Schlüsseltabellen im XML Schema besteht aus zwei Ebenen:

Auf der ersten (abstrakten) Ebene wird mit dem `KeyValuePropertyType` ein allgemeiner `PropertyType` definiert, der die auf alle Schlüsseltabellen anwendbaren Zugriffsmechanismen enthält. Darunter fallen die Möglichkeiten zum Verweis auf einen Schlüsseltabellen-Eintrag sowie die Nutzung der Kompaktdarstellung über die Angabe einer Kennung (und ggf. noch des Namens der entsprechenden Schlüsseltabelle im `attribute` „Objektklasse").

XML Schema:

```
    <complexType name="KeyValuePropertyType">
        <attributeGroup ref="xlink:simpleLink"/>
        <attribute name="Objektklasse" type="string" use="optional"/>
        <attribute name="Kennung" type="string" use="optional"/>
    </complexType
```

Auf der zweiten (konkreten) Ebene werden für jede Schlüsseltabelle zwei `complexTypes` definiert: Der erste mit dem Namen `[Name der Schlüsseltabelle]Type` wird aus dem `AbstractOKSTRAObjektType` abgeleitet und dient zur Darstellung der jeweiligen Schlüsseltabelle mit ihren Attributen. Der zweite mit dem Namen `[Name der Schlüsseltabelle]PropertyType` wird aus dem `KeyValuePropertyType` abgeleitet. Dies ist der `Type` für Schlüsseltabellen-Attribute, die auf die jeweilige Schlüsseltabelle verweisen. Er ermöglicht die vollständige Einbettung der jeweiligen Schlüsseltabelle über ein optionales `element` sowie, da er aus dem `KeyValuePropertyType` abgeleitet wird, auch die Referenzierung der Schlüsseltabelle über einen `xlink`-Verweis und die Nutzung der Kompaktdarstellung. Der folgende Ausschnitt aus dem XML Schema zeigt als **Beispiel** die beiden `complexTypes` für die Schlüsseltabelle *Knotenart:*

XML Schema:

```
    <complexType name="KnotenartType">
        <complexContent>
            <extension base="okstra:AbstractOKSTRAObjektType">
                <sequence>
                    <!-- Eigenschaften Knotenart -->
                    <element name="Kennung"
                        type="string"/>
                    <element name="Langtext"
                        type="string"/>
                </sequence>
```

```
            </extension>
        </complexContent>
    </complexType>

    <complexType name="KnotenartPropertyType">
        <complexContent>
            <extension base="okstra:KeyValuePropertyType">
                <sequence>
                    <element name="Knotenart" type="okstra:KnotenartType"
                    minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

Für den Fall, dass eine Schlüsseltabelle als `okstraObjekt` (und damit als GML-Feature) angegeben werden soll, wird für jede Schlüsseltabelle noch ein zusätzliches globales `element` mit dem Namen der Schlüsseltabelle und dem zugehörigen `Type` definiert – für die Schlüsseltabelle *Knotenart* beispielsweise das im Folgenden aufgeführte `element Knotenart`:

XML Schema:

```
    <element name="Knotenart" type="okstra:KnotenartType"
        substitutionGroup="okstra:_OKSTRAObjekt"/>
```

Wie bereits erwähnt, wird ein Schlüsseltabellen-Attribut im XML Schema als `element` mit dem `Type [Name der Schlüsseltabelle]PropertyType` dargestellt:

EXPRESS:

```
    [Attribut-Name]     : [Name der Schlüsseltabelle];
```

Anm.: Die EXPRESS-Darstellung ist hier nur ein Beispiel. Die Relation kann noch weitere Eigenschaften haben wie OPTIONAL, SET oder LIST, mit den entsprechenden Grenzen.

XML Schema:

```
    <element name="[Attribut-Name]"
        type="okstra:[Name der Schlüsseltabelle]PropertyType"
        {minOccurs="[minimale Anzahl]"}
        {maxOccurs="[maximale Anzahl]"} />
```

Die Kardinalitäten sind hier analog zu Relationen zu setzen (siehe 4.5.2).

Als **Beispiel** betrachten wir die *Knotenart* zum *Netzknoten*:

EXPRESS:

```
ENTITY Netzknoten;
    [...]
    Knotenart    : OPTIONAL Knotenart;
    [...]
END_ENTITY;
```

XML Schema:

```
<complexType name="NetzknotenType">
    [...]
    <element name="Knotenart" type="okstra:KnotenartPropertyType"
        minOccurs="0"/>
    [...]
</complexType>
```

Die `Types` und `elements` zur Darstellung von Schlüsseltabellen befinden sich im OKSTRA®-XML-Schema in der Datei `okstra_schluesseltabellen.xsd`.

Glossar

Begriff	Erläuterung
AdV	Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland
AFIS®	Amtliches Festpunkt-Informationssystem
ALKIS®	Amtliches Liegenschaftskataster-Informationssystem
ATKIS®	Amtliches Topographisch-Kartographisches Informationssystem
`Content Model`	hier: tatsächliches Format von OKSTRA®-XML-Daten OKSTRA®-XML-Daten sind das Bild von OKSTRA®-Daten, die gemäß den Festlegungen im XML Schema zum OKSTRA® formatiert worden sind. Verschiedene XML Schema-Definitionen können die gleiche Struktur in den resultierenden XML-Daten erzeugen. Diese Struktur in der resultierenden XML-Datei nennt man auch `Content Model` des XML Schemas.
CTE	Clear Text Encoding, aus EXPRESS abgeleitetes, textbasiertes Austauschformat, Austauschformat des OKSTRA®, ISO 10303-21
`Element`	Teil einer XML-Datei, der durch korrespondierendes Start-`tag` und End-`tag` begrenzt wird
EXPRESS	Modellierungssprache aus dem CAD-Bereich, wird für die Referenzmodellierung des statischen OKSTRA® verwendet, ISO 10303-11
GML	Geography Markup Language, Anwendung von XML zur einheitlichen Repräsentierung geographischer Informationen auf der Basis von XML Schema, entwickelt von der OGC
NAS	Normbasierte Austauschschnittstelle, in der Evaluierung befindliche Vorschrift zum Datenaustausch des amtlichen Vermessungswesens (AFIS-ALKIS®-ATKIS®)
OGC	Open GIS Consortium, Zusammenschluss führender GIS-Hersteller und -Anwender, de facto Standardisierungsgremium
OKSTRA®	Objektkatalog für das Straßen- und Verkehrswesen
`Tag`	Schlüsselwort in einer XML-Datei, eingeklammert durch '<' und '>'
UML	Unified Modeling Language, Modellierungssprache zur Darstellung statischer und dynamischer Objektmodelle
XML	Extensible Markup Language, Sprache zur Repräsentierung strukturierter Daten in Textdateien
XML Schema	Strukturvorgabe für XML-Dokumente