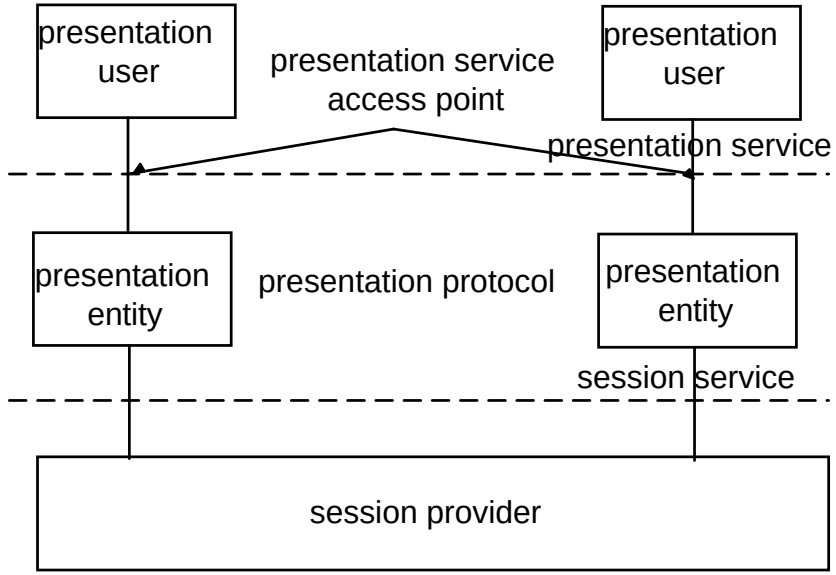
8 Die Darstellungsschicht

8.1 Aufgaben und Funktionsweise



Aufgabe: Repräsentation von Daten zur Übertragung zwischen kommunizierenden Endsystemen

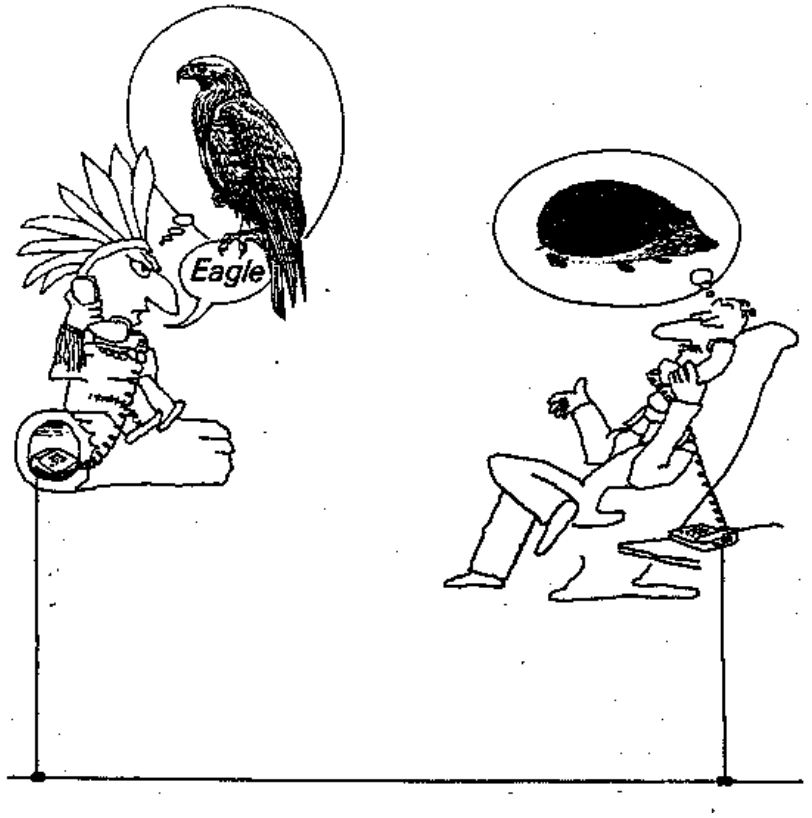
Funktionen:

- Mittel für die Spezifikation komplexer Datenstrukturen
- Aushandlung der aktuell benötigten Syntaxen
- Konvertierung von einer lokalen in eine globale Darstellung

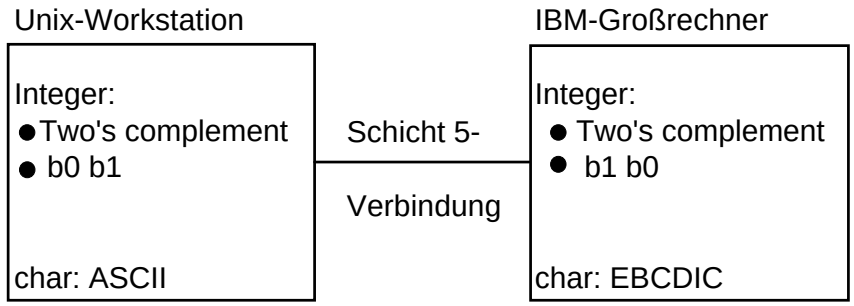
Normen für die Darstellungsschicht

ISO	IS	8822	Darstellungsdienst
ISO	IS	8823	Darstellungsprotokoll
ISO	IS	8824	Abstract Syntax Notation One (ASN.1)
ISO	IS	8825	Basic Encoding Rules für ASN.1

Verbindung ist nicht Kommunikation



Heterogene Rechnerarchitekturen



Trotz fehlerfreier Verbindung in den tieferen Schichten ist keine Kommunikation möglich. Denn die Semantik der übertragenen Nachricht geht verloren.

Heterogene Software

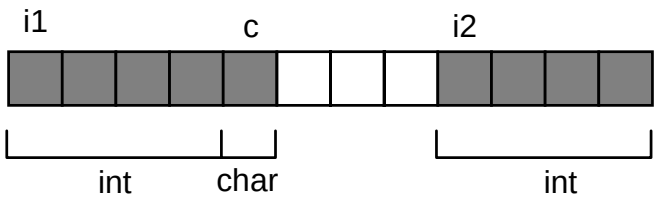
Die lokale Datendarstellung (die lokale "Kodierregel") ist häufig sogar vom Übersetzer abhängig!

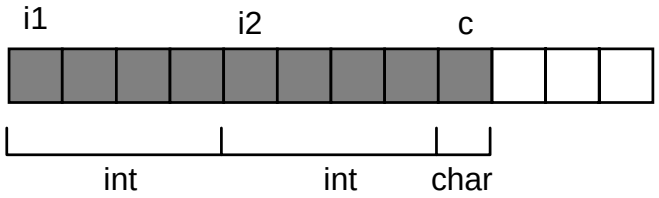
Beispiel

```
struct {
    int  i1;
    char c;
    int  i2;
}
```

char:	ein Byte, ohne Ausrichtungsbedingung
int:	4 Bytes, Ausrichtung auf eine durch 4 teilbare Adresse

Übersetzer mit und ohne Vertauschungsstrategie:



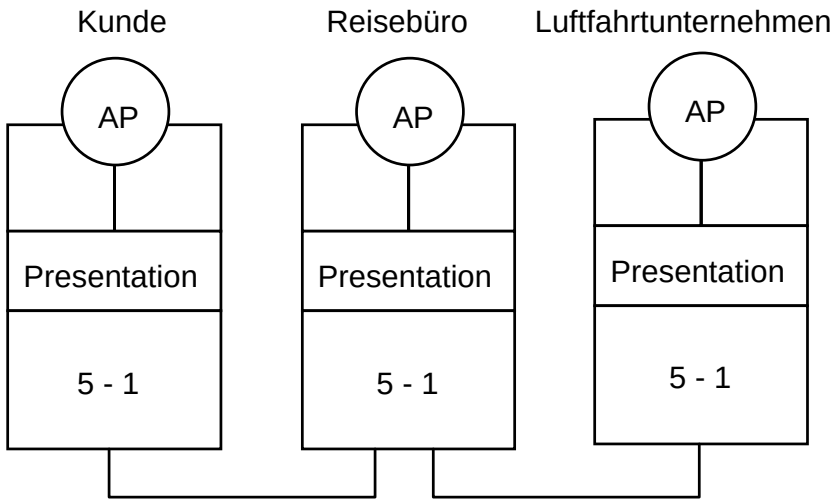


Kodierregeln

Typ (Pascal)	Kodierregeln
INTEGER	Länge Kodierungsart Anordnung Ausrichtung
REAL	Länge der Mantisse Länge des Exponenten Basis des Exponenten Kodierungsart Anordnung Ausrichtung
CHARACTER	Kodierungsart
BOOLEAN	Kodierung für TRUE Kodierung für FALSE
Pointer	Länge Kodierungsart Anordnung Ausrichtung
RECORD	Anordnung Ausrichtung
ARRAY	Anordnung Ausrichtung

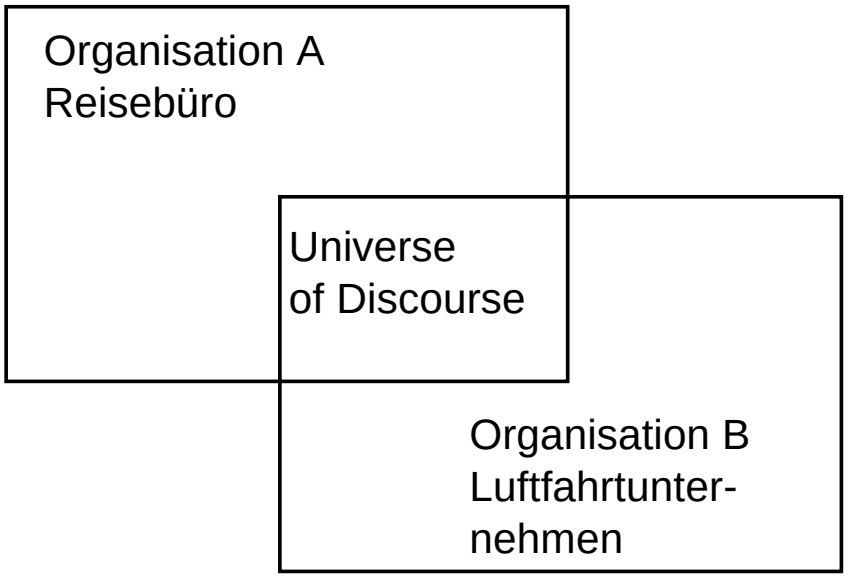
Datenrepräsentation: Anforderungen (1)

Eine verteilte Anwendung in mehreren Organisationen



Datenrepräsentation: Anforderungen (2)

Diskursuniversum = der Ausschnitt aus der realen Welt, der im Rechnersystem behandelt werden soll

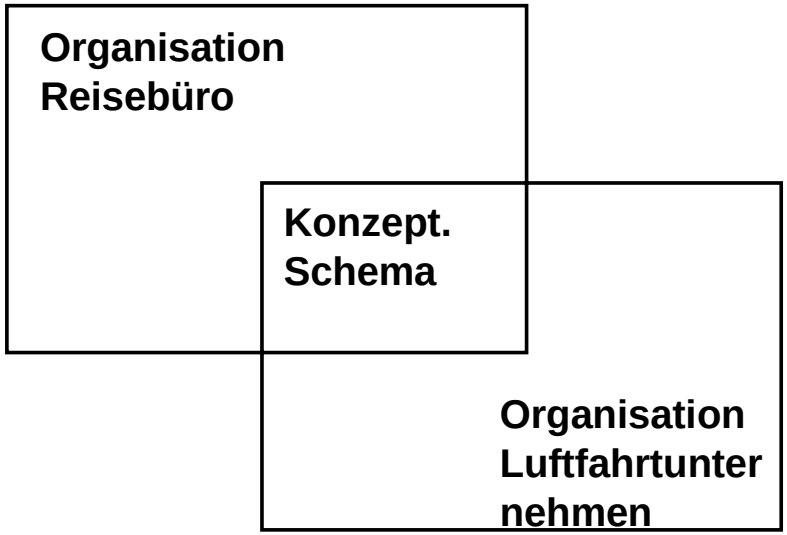


Erste Anforderung:
Alle Partner müssen sich auf dasselbe Diskursuniversum beziehen

Zweite Anforderung:
Benutzung eines gemeinsamen konzeptionellen Schemas

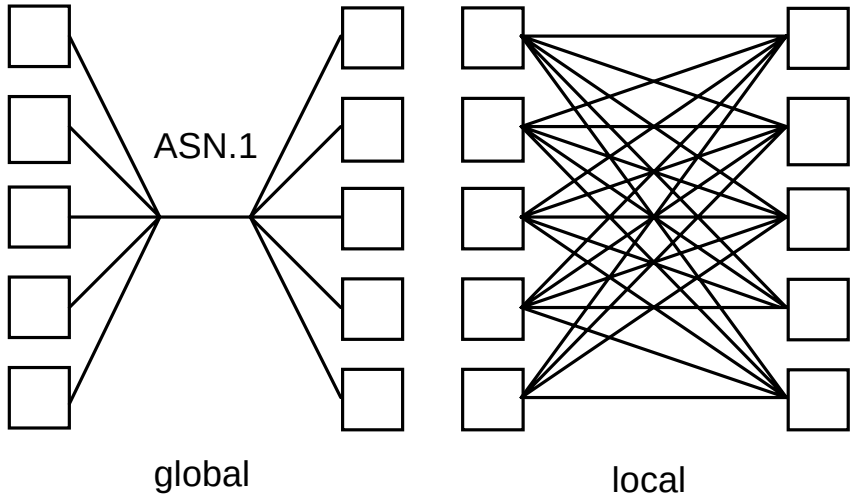
Datenrepräsentation: Anforderungen (3)

Konzeptionelles Schema = formale Beschreibung des Diskursuniversums



Dritte Anforderung:
Für beide Kommunikationspartner verständliche Repräsentation der Objekte des konzeptionellen Schemas
⇒ Kodierregeln, Datenkonvertierung

Datenrepräsentation: Ansätze



Globale Datenrepräsentation

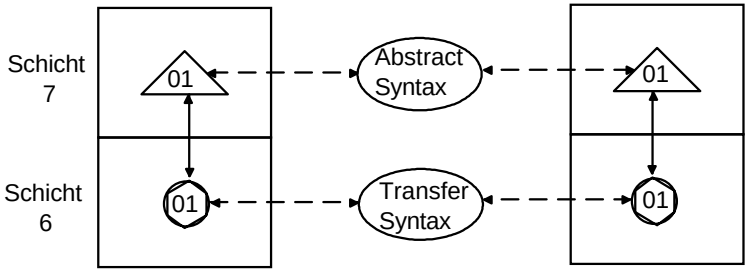
- 2 × n Konvertierungsroutinen
- 2 Konvertierungen pro Datum
  (lokal 1 ⇒ global ⇒ lokal 2)

Lokale Repräsentation eines Kommunikationspartners

- n(n-1) Konvertierungsroutinen erforderlich
- maximal eine Konvertierung pro Datum
  (lokal 1 ⇒ lokal 2)

8.2 Die Darstellungsschicht nach ISO/OSI

Abstrakte Syntax und Transfersyntax



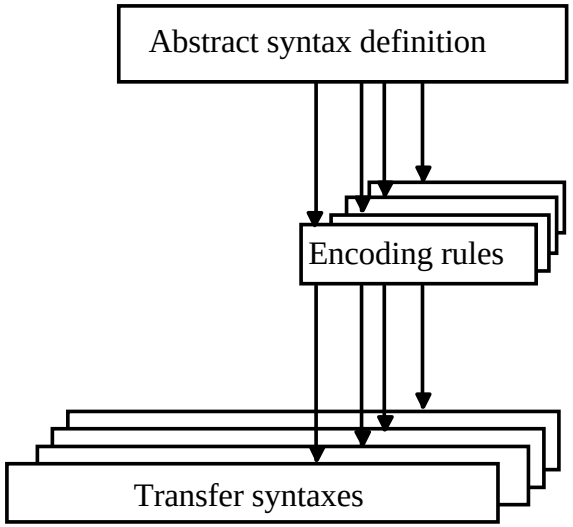
Abstrakte Syntax
Eine Menge von Typdefinitionen für Datenobjekte, die in einem Anwendungsprotokoll verwendet werden

ASN.1 (Abstract Syntax Notation One)
eine Notation zur Definition einer abstrakten Syntax

Transfer Syntax
Eine konkrete Repräsentation der durch eine abstrakte Syntax beschriebenen Daten.

Die Transfersyntax wird durch eine Menge von Kodierregeln definiert.

Abstrakte Syntax und Transfersyntax

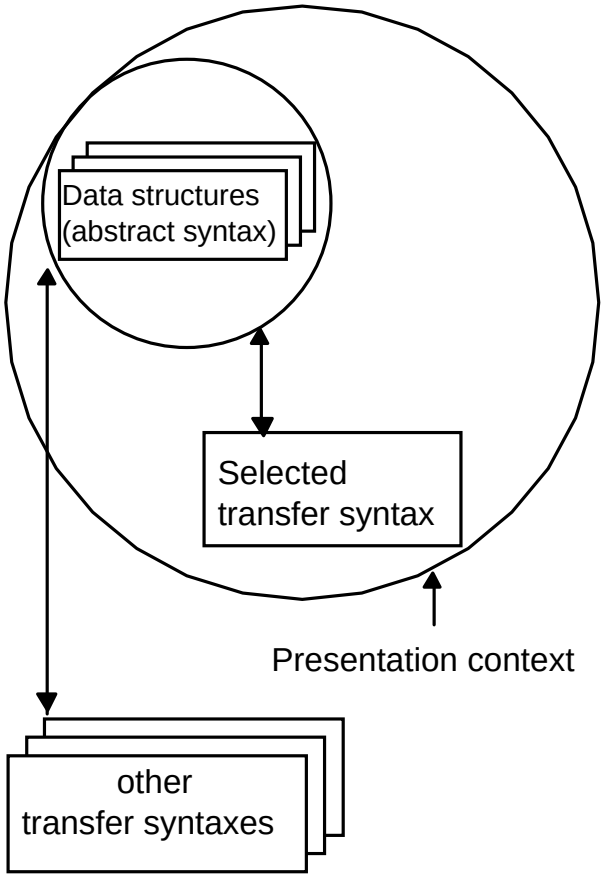


Für eine abstrakte Syntax können mehrere Transfersyntaxen existieren, zum Beispiel

- normale Kodierung
- verschlüsselte Kodierung
- komprimierte Kodierung
- etc.

Bisher sind von der ISO nur die **Basic Encoding Rules** standardisiert worden.

Darstellungskontext



Darstellungskontext

- wird während der Verbindungsaufbauphase ausgehandelt
- ist ein Paar (Abstrakte Syntax, Transfer Syntax)

Abstract Syntax Notation 1 (ASN.1)

Vier Klassen von Tags

1. UNIVERSAL - Boolean, Integer, Bitstring, etc.
2. Application - definiert durch einen internationalen Standard
3. PRIVATE - definiert durch ein Unternehmen
4. Kontext-spezifisch - interpretiert gemäß dem Kontext

Prinzipien von ASN.1

ASN.1 (ISO)

- Eine Sprache zur Spezifikation der Parameter von Benutzerdaten
- Erlaubt die Spezifikation von Typen und Werten, ohne die Werterepäsentation zu bestimmen.
- Ein ergänzender Standard zum Protokollstandard der Anwendungsschicht (Schicht 6)

Kodierregeln

- Spezifizieren die Abbildung der Benutzerdaten der Darstellungsschicht zu den Benutzerdaten der Kommunikationssteuerungsschicht und umgekehrt.
- Werden zwischen kommunizierenden Darstellungsschichten ausgehandelt

Basic Encoding Rules für ASN.1

- Eine spezielle Menge von Kodierregeln für den Transfer, durch die ISO standardisiert.

ASN.1 – Typen

Typdefinition

- NameOfType ::= <<type>>

Initialisieren einer Instanz

- name Of Value Name Of Type ::= <<value>>

Basistypen:

- BOOLEAN
- INTEGER
- BIT STRING
- OCTET STRING
- REAL
- ANY

Konstruktortypen:

- SEQUENCE
- SEQUENCE OF
- SET
- SET OF
- CHOICE

ASN.1 - Basistypen (Beispiele)

BOOLEAN

Werte: TRUE und FALSE

```
Multiple-defined-contexts ::=
   BOOLEAN
simple Multiple-defined-contexts ::=
   TRUE     -- or FALSE
```

INTEGER

Ganze Zahl, keine Begrenzung!

```
ContentLength ::=
   INTEGER
length ContentLength ::=
   100     -- or ´64´H
```

Alternative Form: Aufzählung der möglichen Werte

```
RegistrationMailType ::=
   INTEGER {
      non-registered-mail (0),
      registered-mail (1),
      registered-mail-to-address-
        in-person (2)   }
Mymail RegistrationMailType ::=
   registered-mail     -- or 1
```


ASN.1 - Basistypen

OCTET STRING

0 oder mehrere Bytes

```
UserName ::= OCTET STRING
initiator UserName ::=
   "anon"     -- or ´616E6F6E´H
```

Verfeinerungen

- IA5String (CCITT Internat. Alphabet Number 5)
- NumericString
- VisibleString
- PrintableString

ASN.1 - Konstruktortypen

SEQUENCE

Geordnete Liste von 0 oder mehreren Elementen (entspricht einem Record in Pascal)

```
VarBind ::=
   SEQUENCE {
      ObjectName,
      ObjectSyntax
   }
```

Bessere Lesbarkeit durch Text-Labels

```
VarBind ::=
   SEQUENCE {
      name
         ObjectName,
      value
         ObjectSyntax
   }
```

Möglichkeit von OPTIONAL- und DEFAULT-Werten

```
Interrupt-Request ::=
   SEQUENCE {
      fatal-error
         BOOLEAN
         DEFAULT TRUE,
      message
         PrintableString
         OPTIONAL
   }
```


ASN.1 - Konstruktortypen

SEQUENCE OF

Geordnete Liste von Elementen desselben Typs (dynamisches Array)

```
RoutingTable ::=
   SEQUENCE OF
      RoutingEntry
```

SET

Menge von 0 oder mehr Elementen möglicherweise unterschiedlichen Typs. Unterscheidung der Elemente erfolgt durch ihren Typ.

```
IMPIdentifier ::=
   SET {
      user
         ORAddress
         OPTIONAL,

      user-relative-identifier
         LocalIMPIdentifier
   }
```

ASN.1 - Konstruktortypen

SET OF

Menge von 0 oder mehr Elementen desselben Typs

```
MulticastGroup ::=
    SET OF
        Peer
```

CHOICE

Vereinigung von einem oder mehreren Datentypen. Die Instanz nimmt den Wert genau eines dieser Typen an ("variant record" in Pascal)

ASN.1 - Tags

ASN.1 assoziiert mit jedem Datentyp einen Identifizierer (Tag)

Es gibt vier Klassen :

- Universal tag:
  Global eindeutige Identifikation. Werden im ASN.1-Standard definiert (BOOLEAN, INTEGER, SEQUENCE, ...)
- Application-wide tag:
  Eindeutig innerhalb eines ASN.1-Moduls
- Context-specific tag:
  Eindeutig "innerhalb" eines Konstruktortyps.
- Private-use tag:
  Definiert durch ein Unternehmen. Eindeutig nur im Bereich des Unternehmens.

ASN.1 Universal Tags

Universal Tag	ASN.1 Type
1	BOOLEAN
2	INTEGER
3	BIT STRING
4	OCTET STRING
5	NULL
6	OBJECT IDENTIFIER
7	Object Descriptor
8	EXTERNAL
9	REAL
10	ENUMERATED
12-15	Reserved for addenda
16	SEQUENCE, SEQUENCE OF
17	SET, SET OF
18	Numeric String
19	Printable String
20	Teletex String
21	Videotex String
22	IA5 String
23	UTC Time
24	Generalized Type
25	Graphics String
26	Visible String
27	General String
28	Character String
29- ...	Reserved for addenda

ASN.1 Tags

Das Tag eines Datentyps ist integraler Teil der Struktur dieses Typs

```
IA5String ::=
   [UNIVERSAL 22]
      OCTET STRING

   [UNIVERSAL 22] [UNIVERSAL 4]
   <OctetStringStructure>
```

IMPLICIT Type unterdrückt den Tag des Typs bei der Kodierung

```
IA5String ::=
   [UNIVERSAL 22]
      IMPLICIT OCTET STRING

   [UNIVERSAL 22]
   <OctetStringStructure>
```

ASN.1 - Beispiel (1)

```
Personal-Stammsatz
Personal-Stammsatz ::= [APPLICATION 0] IM-
PLICIT SEQUENCE
{
   kennung              PersonalNummer,
   ps-name              Name,
   stellung         [0] VisibleString,
   adresse              Adresse,
   einstellung      [1] Datum,
   familienStand    [2] IMPLICIT
                        FamilienStand,
   ehegatte         [3] Name OPTIONAL,
   kinder           [4] IMPLICIT SEQUENCE
                        OF Kinder-Info
   DEFAULT{}
}
Kinder-Info ::= SET
{
   kind-name          Name,
   geboren       [0] Datum
}
```


ASN.1 - Beispiel (2)

```
Name ::=[APPLICATION 1] IMPLICIT SEQUENCE
{
   titel             [0] IMPLICIT
                         VisibleString
                         OPTIONAL,
   rufname           [1] IMPLICIT
                         VisibleString,
   weitereVorname    [2] IMPLICIT SET OF
                         VisibleString,
   FamilienName      [3] IMPLICIT
                         VisibleString,
}
Adresse ::=[APPLICATION 2] IMPLICIT SE-
QUENCE
{
   strasse           VisibleString,
   hausnummer   [0]  VisibleString,
                        --z.B. 3a
   plz               INTEGER,
   ort          [1]  VisibleString,
   land         [2]  VisibleString DEFAULT
                        "D",
   telnr             IA5String OPTIONAL
}
```

ASN.1 - Beispiel (3)

```
PersonalNummer ::= [APPLICATION 3]
                   IMPLICIT   INTEGER
Datum          ::= [APPLICATION 4]
                   IMPLICIT   IA5String
FamilienStand ::= INTEGER
{
   ledig        (0),
   verheiratet  (1),
   geschieden   (2),
   verwitwet    (3)
}
```


ASN.1 - Beispiel (4)

Spezielle Ausprägung des Personal-Stammsatzes

```
hans-Meier Personal-Stammsatz ::=
{
   kennung      4711,
   ps-name{     rufname          "Hans",
                weitereVornamen {"Georg"},
                familienName     "Meier"
   },
   stellung      "SachbearbeiterTZ",
   adresse{      strasse "Hauptstraße",
                 hausnummer "16",
                 plz 8520,
                 ort "Erlangen",
                 telnr "0 91 31/95 95 95"
   },
   einstellung   "1.10.1985",
   familienStand 1,
```

ASN.1 - Beispiel (5)

```
   ehegatte{     rufname      "Marie-
                               Luise",
                 weitereVornamen
                              {"Gertrud"},
                 familienName "Meier"
   }
   kinder{
      { kind-name {rufname        "Otto"
                   weitereVornamen
                              {"Wilhelm"},
                   familienName    "Meier"
        },
        geboren  "25.6.1969"
      },
      { kind-name { rufname      "Sabine",
                    weitereVornamen
                        {"Helga","Maria"},
        familienName    "Schmidt"
        },
        geboren     "6.11.1979"
      }
   }
}
```


Basic Encoding Rules

Jede Kodierung eines Datenwertes hat gemäß den "Basic Encoding Rules" die folgende Form:

Identifizierer	Länge	Inhalt	Ende-Zeichen

Identifizierer (tag):

Ein oder mehrere Oktaden, welche eine Kodierung für Datentyp-Klasse und -Nummer enthalten.

Länge :

Gibt die Länge des Inhalts in Oktaden an oder kennzeichnet, daß ein spezielles Ende-Zeichen vorhanden ist.

Ende-Zeichen:

Ein Ende-Zeichen, bestehend aus zwei 0-Oktaden, kann vorhanden sein. Die Kodierung mit Ende-Zeichen statt mit Längenfeld wird insbesondere dann gewählt, wenn bei Beginn des Kodiervorgangs die Länge des Datenfeldes noch nicht bekannt ist (z. B. bei sehr großen Dateien)

Basic Encoding Rules

Inhalt:
Hier wird der eigentliche Datenwert kodiert. Der Inhalt eines zusammengesetzten Typs wird in naheliegender Weise gebildet, z.B.

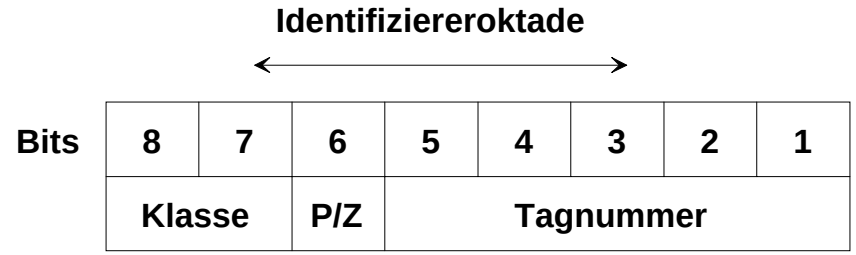
SequenceType
Der Dateninhalt ist eine Folge von Datenwerten, entsprechend den in der Definition aufgeführten Typen.

SequenceOfType
Der Dateninhalt besteht aus null oder mehreren Datenwerten vom angegebenenTyp.

ChoiceType:
Der Dateninhalt ist vom ausgewählten Typ aus der Liste der möglichen Typen.

Basic Encoding Rules: Tag - Felder

Identifizieroktade (niedrige tag-Nummer)



P/Z = 0: Primitiv
1: Zusammengesetzt

Kodierung der tag-Klasse

Klasse	Bit 8	Bit 7
UNIVERSAL	0	0
APPLICATION	0	1
kontext-spezifisch	1	0
PRIVATE	1	1

Basic Encoding Rules: Längenfeld

a) bekannte Länge (definite length encoding)

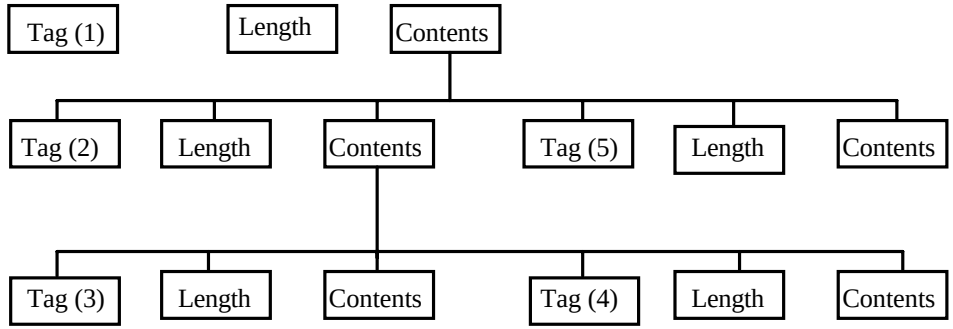
- kurze Form: 1 Byte
  - — erstes Bit 0
  - — die anderer 7 Bits kodieren die Länge als eine binäre Ganzzahl
- lange Form: n Bytes, n > 1
  - — erstes Bit 1
  - — alle anderen Bits des ersten Bytes kodieren die Anzahl der Längenoktaden
  - — alle anderen Bits aller folgenden Längenoktaden kodieren die Länge als eine binäre Ganzzahl

b) unbekannte Länge (indefinite length encoding)

- Kennung: 1 Byte
  - — erstes Bit 1
  - — alle anderen Bits 0

Wird bei der Kodierung zusammengesetzer Datentypen verwendet, bei denen die Länge nicht sofort verfügbar ist.

BER: Konstruktortypen




```
X-Record ::=
   SEQUENCE {
      xy-Record
         SEQUENCE {
            y-elem1 INTEGER,
            y-elem2 OCTET STRING
         }
      x-elem BOOLEAN
   }
```

Kodierung gemäß Basic Encoding Rules

```
Personalstammsatz     Länge        Inhalt
<60>                  <81><FA>
Personalnummer        Länge        Inhalt
<43> <02>             <12><67>
Name                  Länge        Inhalt
<61>                  <16>
[1]                   Länge        Inhalt = "Hans"
<81>                  <04>          <48><61><6E><73>
[2]                   Länge        Inhalt
<A2>                  <07>
VisebleString         Länge        Inhalt = "Georg"
<1A>                  <05>         <47><65><6F><72><67>
[3]                   Länge        Inhalt = "Meier"
<83>                  <05>         <4D><65><69><65><72>
[0]                   Länge        Inhalt
<A0>                  <13>
VisibleString         Länge        Inhalt =
                                   "Sachbearbeiter TZ"
<1A>                  <11>
<53><61><63><68><62><65><61><72><62>

<65><69><74><65><72><20><54><5A>
Adresse               Länge        Inhalt
<62>                  <37>
VisibleString         Länge        Inhalt =
                                   "Hauptstraße"
```



```
<1A>                  <0C>         <48><61><75><70><74>
<73><74><72><62>73><73><65>
[0]                   Länge        Inhalt
<A0>                  <04>
VisibleString         Länge        Inhalt = "16"
<1A>                  <02>         <31><36>
INTEGER               Länge        Inhalt
<02>                  <08>         <21><48>
[1]                   Länge        Inhalt
<A1>                  <0A>
VisibleString         Länge        Inhalt = "Erlangen"
<1A>                  <08>
<45><72><6C><61><6E><67><65><6E>
[2]                   Länge        Inhalt
<A2>                  <03>
VisibleString         Länge        Inhalt = "D"
<1A>                  <01>         <44>
LA5String             Länge        Inhalt =
                                   "09131/959595"
<16>                  <0C>
<30><39><31><33><31><2F>

<39><35><39><35><39><35>
[1]                   Länge        Inhalt
<A1>                  <0B>
Datum                 Länge        Inhalt = "1.10.1985"
```

```
<44>                    <09>
<31><2E><31><30><2E><31><39><38><35>
[2]                     Länge       Inhalt
<82>                    <01>        <01>
[3]                     Länge       Inhalt
<A3>                    <21>
Name                    Länge       Inhalt
<61>                    <1F>
[1]                     Länge       Inhalt =
                                    "Marie-Luise"
<81>                    <0B>
<4D><61><72><69><65><2D>
                                    <4C><75><69><73><65>
[2]                     Länge       Inhalt
<A2>                    <09>
VisibleString           Länge       Inhalt = "Gertrud"
<1A>                    <07>
<47><65><72><74><72><75><64>
[3]                      Länge      Inhalt = "Meier"
<83>                     <05>       <4D><65><69><65><72>
[4]                      Länge      Inhalt
<A4>                     <5B>
Kinder-Info              Länge      Inhalt
<31>                     <27>
```



```
Name                    Länge       Inhalt
<61>                    <18>
[1]                     Länge       Inhalt = "Otto"
<81>                    <04>        <4F><74><74><6F>
[2]                     Länge       Inhalt
<A2>                    <09>
VisibleString           Länge       Inhalt = "Wilhelm"
<1A>                    <07>
<57><69><6C><68><65><6C><6D>
[3]                     Länge       Inhalt = "Meier"
<83>                    <05>        <4D><65><69><65><72>
[0]                     Länge       Inhalt
<A0>                    <0B>
Datum                   Länge       Inhalt = "25.6.1969"
<44>                    <09>        <32><35><2e><36><2e>
                                    <31><39><36><39>
Kinder-Info             Länge       Inhalt
<31>                    <30>
Name                    Länge       Inhalt
<61>                    <21>
[1]                     Länge       Inhalt = "Sabine"
<81>                    <06>
<53><61><62><69><6E><65>
[2]                     Länge       Inhalt
<A2>                    <0E>
```

```
VisibleString          Länge       Inhalt = "Helga"
<1A>                   <05>        <48><65><6C><67><61>
VisibleString          Länge       Inhalt = "Maria"
<1A>                   <05>        <4D><61><72><69><61>
[3]                    Länge       Inhalt = "Schmidt"
<83>                   <07>
<53><63><68><6D><69><64><74>
[0]                    Länge       Inhalt
<A0>                   <0B>
Datum                  Länge       Inhalt = "6.11.1979"
<44>                   <09>
<36><2E><31><31><2E><31><39><37><39>
```


Die OSI-Darstellungsschicht (Presentation Layer)

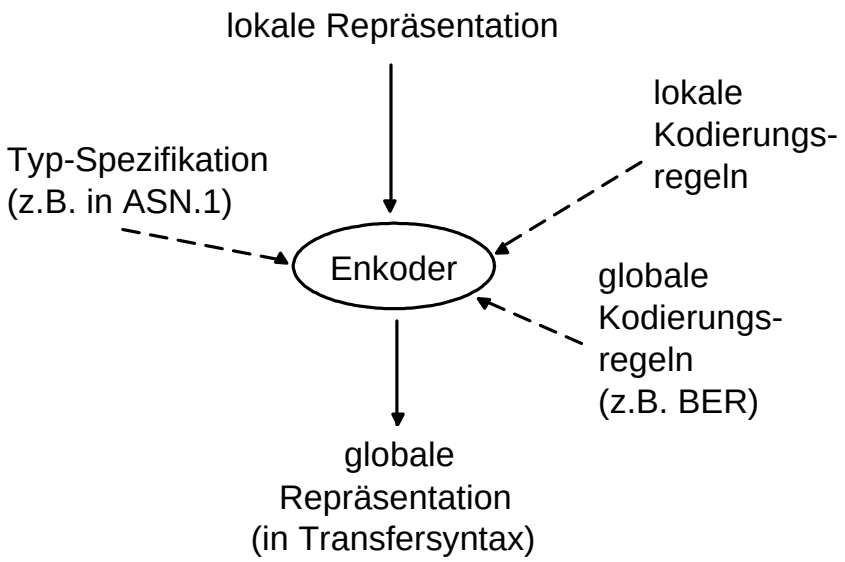
Aufgabe

Semantikerhaltende Kodierung bei der Übertragung zwischen kommunizierenden Systemen

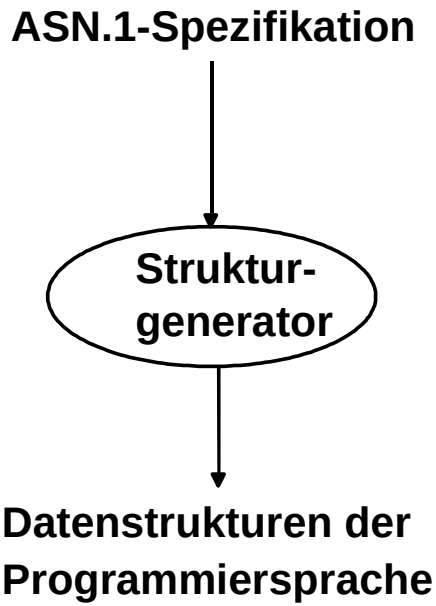
Funktionen

- Verhandlung ders Darstellungskontexts (Paar aus abstrakter Syntax und Transfersyntax)
- Transformation in die und aus der Transfersyntax
- Durchreichung von Kommunikationssteuerungsdiensten (Diensten der Schicht 5)

Kodierer / Dekodierer



Tools: Strukturgenerator



Der Programmierer erhält die fertigen Datenstrukturen zum Einbinden in sein Programm. Er schreibt nur noch den prozeduralen Teil des Moduls. Die Konvertierung in die und aus der Transfersyntax wird von der Darstellungsschicht automatisch durchgeführt.

Beispiel

```
PDU ::=
   SEQUENCE {
     ...
      error-status
        INTEGER {
          noError(0),
          tooBig(1),
          noSuchName(2),
          badValue(3),
          readOnly(4),
        },
       ...
   }

struct type_SNMP_PDU{
  ...
   int     error__status;
  #define int_SNMP_error__status_noError 0
  #define int_SNMP_error__status_tooBig 1
  #define
     int_SNMP_error__status_noSuchName 2
  #define int_SNMP_error__status_badValue 3
  #define int_SNMP_error__status_readOnly 4
  ...
};
```


8.3 XDR, die Darstellungs"schicht" im Internet

XDR = External Data Representation

- Eine Darstellungs"schicht" mit minimaler Funktionalität
- Beispiel für ein Darstellungsproblem: Ganzzahlen (integers)

Prozessoren der Familien Motorola 680x0, IBM 370

höherwertig			niederwertig
Byte 1	Byte 2	Byte 3	Byte 4

Prozessoren der Familie Intel 80x86

niederwertig			höherwertig
Byte 1	Byte 2	Byte 3	Byte 4

"big-endian" versus "little-endian"

XDR

Alle Daten werden auf eine **fest vereinbarte** Transfersyntax abgebildet (keine Verhandlungsmöglichkeit):

- Alle Ganzzahlen (integers) als 4-Byte big-endians
- Gleitkommazahlen im IEEE-Format:

```
Mantisse   23 Bits
Exponent   8 Bits
Vorzeichen 1 Bit
```

- Texte im ASCII - Code
- alle Datenelemente (Felder) auf 4-Byte-Grenze ausgerichtet.

Nachteil: Zwei identische Maschinen, die andere interne Darstellungsformate als die oben genannten verwenden, müssen zweimal umwandeln.

Es gibt einen XDR-Compiler, der zu XDR-Definitionen passende C-Datenstrukturen und Programmstücke zum Kodieren/Dekodieren erzeugt.

Zusammenfassendes Beispiel

Ein Datenpaket des Protokolls "Remote Procedure Call" (RPC) im Internet-Protokollstack

Paketanfang

Ethernet-Header
IP-Header
UDP-Header
RPC-Header
Benutzer-Daten im XDR-Format
Ethernet-Checksumme

Paketende

Wie man sieht, benötigt XDR **keinen eigenen Header**; die Kodierung folgt implizit den Konventionen von XDR.