ASN.1 – *Abstract Syntax Notation* : comment communiquer dans un environnement hétérogène

Plan

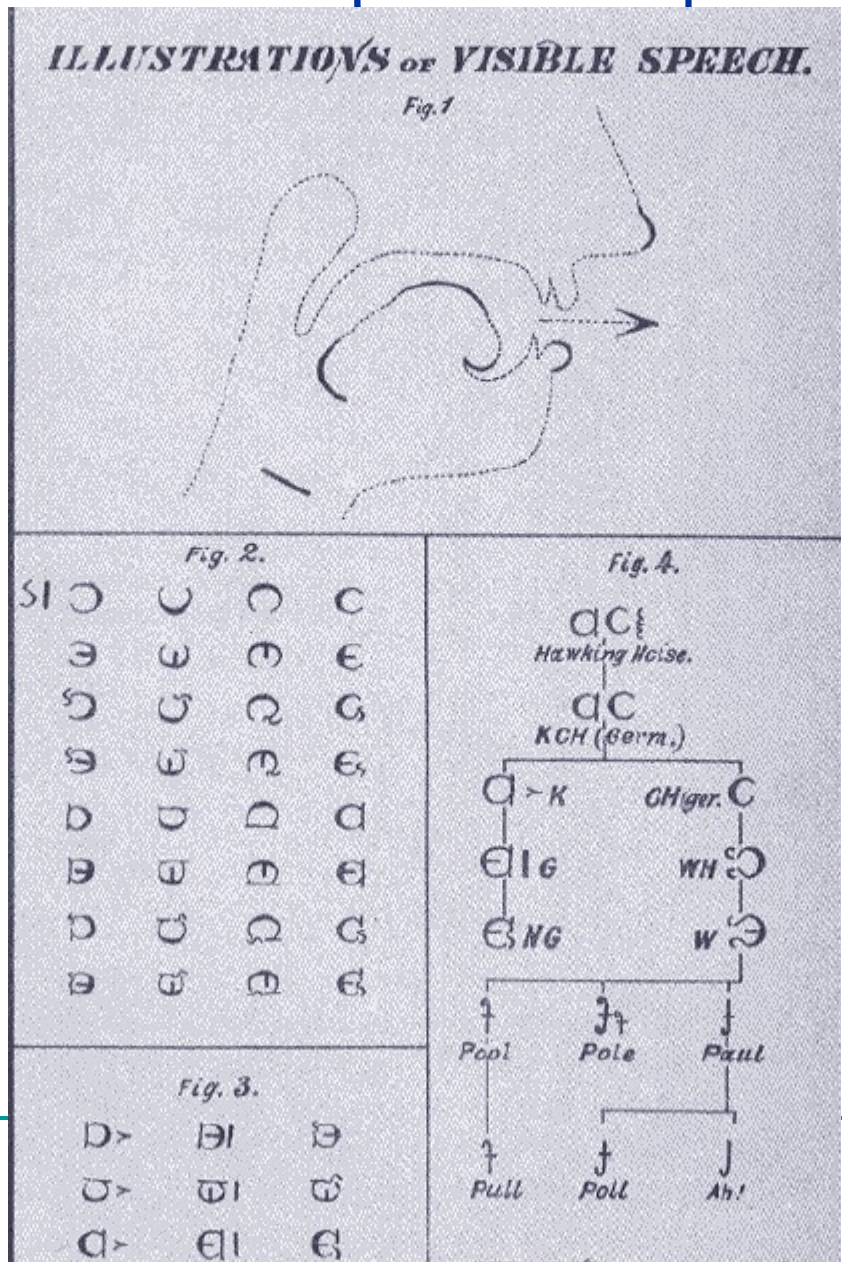
- Introduction et histoire de la notation ASN.1
- Syntaxe ASN.1
- Règles d’encodage
- Outils
- Conclusion

Histoire de ASN.1

- Quelques dates importantes :
  - 1982 : naissance d'un standard
  - 1984 : apparition du premier standard
  - 1994 : évolution majeure des réseaux (débits importants, nouveaux alphabets…) mène à l'évolution de ASN.1
- Organismes menant cette standardisation :
  - International Telecommunications Union-Telecommunications Sector (ITU-T)
  - International Organization for Standardization (ISO)
  - International Electrotechnical Commission (IEC)

ASN.1, c'est un peu comme la phonétique …

- En 1867, Alexander Melville Bell introduit le "Visible speech" :
  - une sorte d'alphabet décrivant la position de la langue, des lèvres, des actions précises à effectuer pour dialoguer
  - Objectif : faciliter l'apprentissage de la langue parlée auprès des malentendants



ASN.1, c'est un peu comme la phonétique …

- Le "visible speech" est
  - Un language physiologique : des symboles visuels représentent la position des organes de la parole
  - Le premier alphabet universel, dans un sens **une notation abstraite**
  - Un langage **indépendant du langage "parlé",** du dialect (cf illustration)

LETTER-VALUE OF THE PRINCIPAL CONSONANT AND GLIDE SYMBOLS.

CONSONANTS.

Am. American, Cock. Cockney, E. English — F. French, Ga. Gaelic, Ge. German — Hu. Hungarian, Ir. Irish, It. Italian — Pec. peculiar, Po. Polish, Sc. Scotch — Sp. Spanish, W. Welsh, Z. Zulu.

he (E.)	(Vowel whisp.)	variety of defective *r*		bŭ'er for *butter* (west of Scot.)	
nach (Ge.) *pech* (Sc.)	*auch* (Ge.) *sough* (Sc.)	hiss of water-fowl.		*c, k, q,* (E.) ⎫ kind (E.) ⎭	sink* (E.)
ich (Ge.)	*s, c,* (E.) ⎫ *ciudad* (Sp) ⎭	variety of defective *s*	*thin* (E.)	variety of *t*	variety of *n**
theatre (F.) *-rh* (W.)	*show* (E.) *chaud* (F.)	*temple* (F.) felt* (E.)	*ll* (W.) *hl* (Z.)	*tie* (E.)	tent* (E.)
variety of *f* or *wh*	*why* (E.)	*fie* (E.)	gutturalized variety of *f*	*pie* (E.)	lamp* (E.) *m/m!* (Sc.)
tage (Ge.) ⎫ *zeige* (Ge.) ⎬ burred *r* ⎭	var^ty of *g* (Ge.) and of defective *r* (E.)	*laogh* (Ga.) barred *l* (Po.)	labialized variety of Gaelic *l*	*go* (E.) ⎫ *guide* (E.) ⎭	*sing* (E.)
yes (E.)	*zeal* (E.) ⎫ *d*, final (Sp.) ⎭	*llano* (Sp.) *gli* (It.)	*then* (E.)	Magyar (Hu.)	Bologne (F.)

ASN.1, c'est un peu comme la phonétique …

- Le visible speech est un langage qui peut être utilisé en tant que lien entre deux dialectes :
  - A chaque "nation" de définir ses règles d'encodage
  - Ici : anglais ➔visible speech, visible speech ➔ anglais
  - ToDo: Français ➔ visible speech, visible speech ➔ français

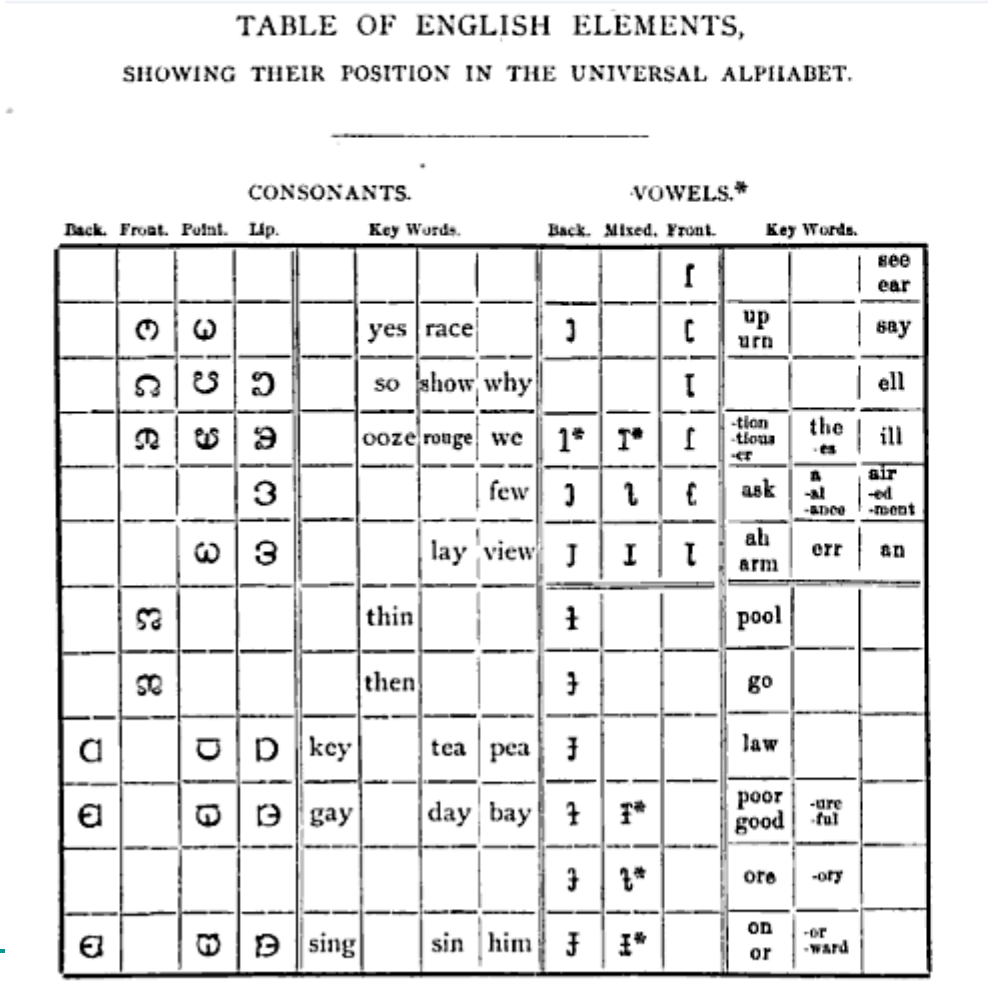


TABLE OF ENGLISH ELEMENTS,

SHOWING THEIR POSITION IN THE UNIVERSAL ALPHABET.

CONSONANTS.								VOWELS.*					
Back.	Front.	Point.	Lip.	Key Words.				Back.	Mixed.	Front.	Key Words.		
													see ear
					yes	race					up urn		say
					so	show	why						ell
					ooze	rouge	we				-tion -tious -cr	the -es	ill
							few				ask	a -al -ance	air -ed -ment
						lay	view				ah arm	err	an
					thin						pool		
					then						go		
				key		tea	pea				law		
				gay		day	bay				poor good	-ure -ful	
											ore	-ory	
				sing		sin	him				on or	-or -ward	

ASN.1 – définition

- Notation utilisée pour décrire les messages échangés entre des programmes
- Fournit une **description haut niveau** des messages
  - “on ne se place pas au niveau des bits, des octets, d’un langage de programmation particulier”
  - Est un sorte de visible speech
- En pratique, ASN.1 est une notation abstraite permettant la spécification d’informations générées par des protocoles hauts-niveaux (comme SNMP), sans perte de généralité au niveau matériel ou logiciel

Règles d'encodage

- Très proches de ASN.1, mais ne faisant pas partie du standard, se trouvent les règles d'encodage
- Les règles d'encodage décrivent les bits et les octets (messages) transitant entre les programmes
- ASN.1 et les règles d'encodage
  - ne sont liés à aucune architecture, système d'exploitation, langage de programmation
  - Sont tous deux publiés par ITU, ISO, IEC

Pourquoi ASN.1?

- La diversité des architectures machines
- La rapidité des échanges et l'explosion de leur nombre ne signifient pas que les structures des informations échangées soient simplifiées
- Les réseau et les terminaux constituent un mode hétérogène

Quelques exemples historiques – encodage au niveau des terminaux

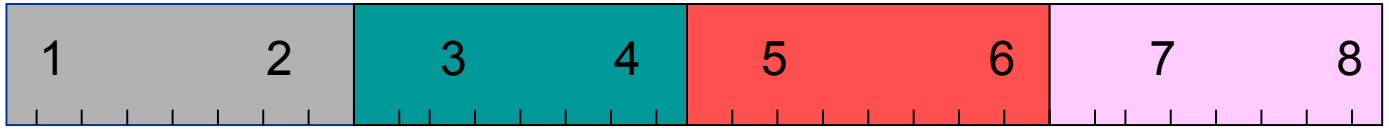
- Deux représentations différentes co-existent dans les années 50- début 60
  - Le code ASCII (*American Standard Code for Information Interchange*) connu sous la dénomination de ANSI X3.4
  - Le code EBCDIC (*Extended Binary Coded Decimal Interchange Code*) proné par IBM dans les années 50-60, abandonné depuis…

Quelques exemples historiques – transfert

- L'ordonnancement des octets dans lequel les données sont stockées dans la mémoire, ou sont échangés, peut être effectué de deux façon :
  - big endian : les octets sont numérotés de gauche à droite
  - little endian : les octets sont numérotés de droite à gauche
- Des manufacturiers « little endian » et des manufacturiers « big endian »

Exemple – transfert d'un registre vers la mémoire via le bus de données

Registre de 32 bits

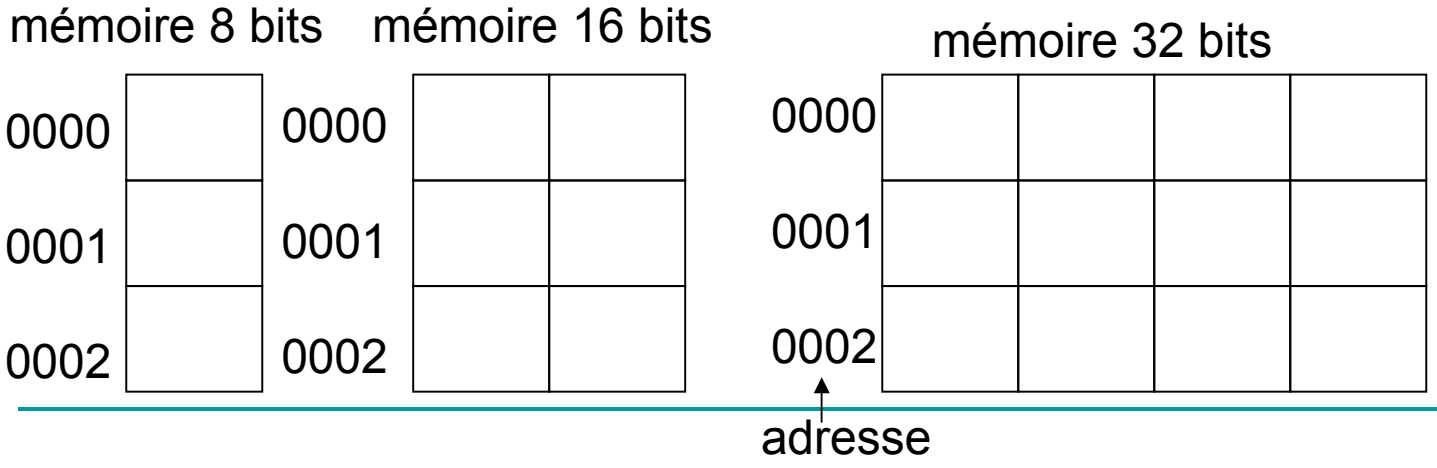


On considère trois transferts entre le registre et la mémoire :

- 1 octet
- 2 octets
- 4 octets

Échanges *via* un bus reliant le registre à la mémoire

On considère trois types de mémoire :



Exemple – transfert de 1 octet

Big endian

mémoire 8 bits

0000	78
0001	
0002	
0003	

mémoire 16 bits

0000	
0001	
0002	
0003	

mémoire 32 bits

0000				78
0001				
0002				
0003				

Little endian

mémoire 8 bits

0000	78
0001	
0002	

mémoire 16 bits

0000	78
0001	
0002	

mémoire 32 bits

0000	78			
0001				
0002				

Exemple – transfert de 2 octets

Big endian

mémoire 8 bits

0000	78
0001	56
0002	

mémoire 16 bits

0000	56	78
0001		
0002		

mémoire 32 bits

0000			56	78
0001				
0002				

Little endian

mémoire 8 bits

0000	78
0001	56
0002	

mémoire 16 bits

0000	78	56
0001		
0002		

mémoire 32 bits

0000	78	56		
0001				
0002				

Exemple – transfert de 4 octets

Big endian

mémoire 8 bits

0000	78
0001	56
0002	34
	12

mémoire 16 bits

0000	56	78
0001	12	34
0002		

mémoire 32 bits

0000	12	34	56	78
0001				
0002				

Little endian

mémoire 8 bits

0000	78
0001	56
0002	34
0003	12

mémoire 16 bits

0000	78	56
0001	34	12
0002		

mémoire 32 bits

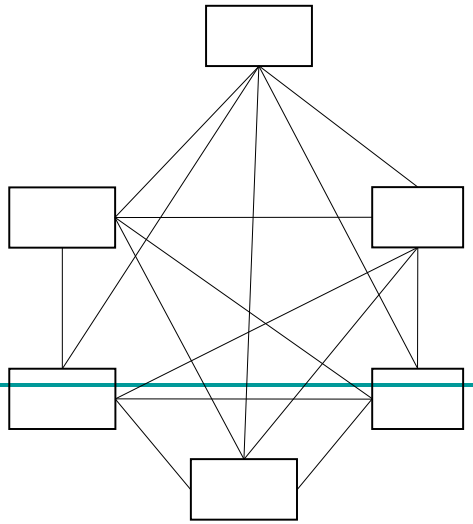
0000	78	56	34	12
0001				
0002				

Programme de conversion - question

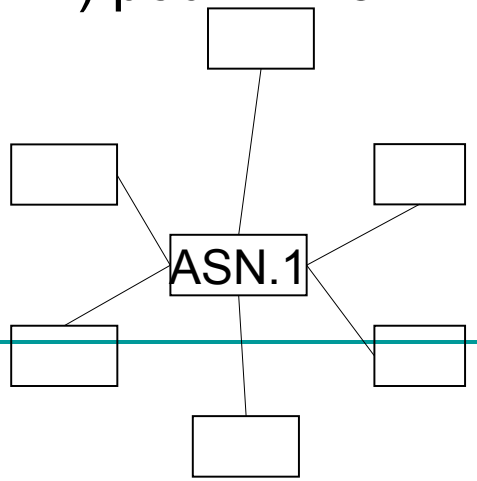
- Question : comment et à quel endroit effectuer la conversion ?
- Hypothèse : n machines
  - Chaque machine connaît la représentation interne de ces n-1 voisins
  - L'émetteur et le récepteur se mettent d'accord pour un format de données
  - Une communication bi-directionnelle

Programme de conversion – réponse

- Solution 1
  - n . (n-1) conversions sont nécessaires
  - Une seule traduction est effectuée pour chaque transfert



- Solution alternative
  - Un format commun de transfert (par ex. ASN.1) est utilisé
  - n encodeurs et n décodeurs, soit 2.n
  - Remarque : 2. n < n . (n-1) pour n > 3



Définitions– syntaxe concrète

- Représentation dans un **langage donné** des structures de données à transférer
  - Il s'agit d'une **syntaxe** car elle respecte les règles lexicales et grammaticales d'un langage (C par ex.)
  - Cette syntaxe est c**oncrète** car elle a à voir avec la machine, ses restrictions

Définition– syntaxe abstraite

- Est **indépendante** du langage de programmation
- Dispose d'une **grammaire** devant être respectée par les données échangées
- Offre un **formalisme** riche au niveau du typage : elle offre la possibilité de définir des données simples/primitives et complexes/construites/structurées (c'est-à-dire construites récursivement)
- Est **formelle** ce qui permet de prévenir les ambigüités
- Définit la structure des données mais **pas la sémantique** : **l'interprétation des données est effectuée par l'application**

Exemple pratique – syntaxe abstraite, syntaxe concrète

Syntaxe abstraite

```
Record SEQUENCE {
name PrintableString(SIZE(1..30));
age INTEGER;
}Record;
```

Machine A

```
Typedef struct Record {
Char name [31];
int age;
}Record;
```

Syntaxe concrète

Machine B

```
Typedef struct Record {
Char name [31];
int age;
}Record;
```

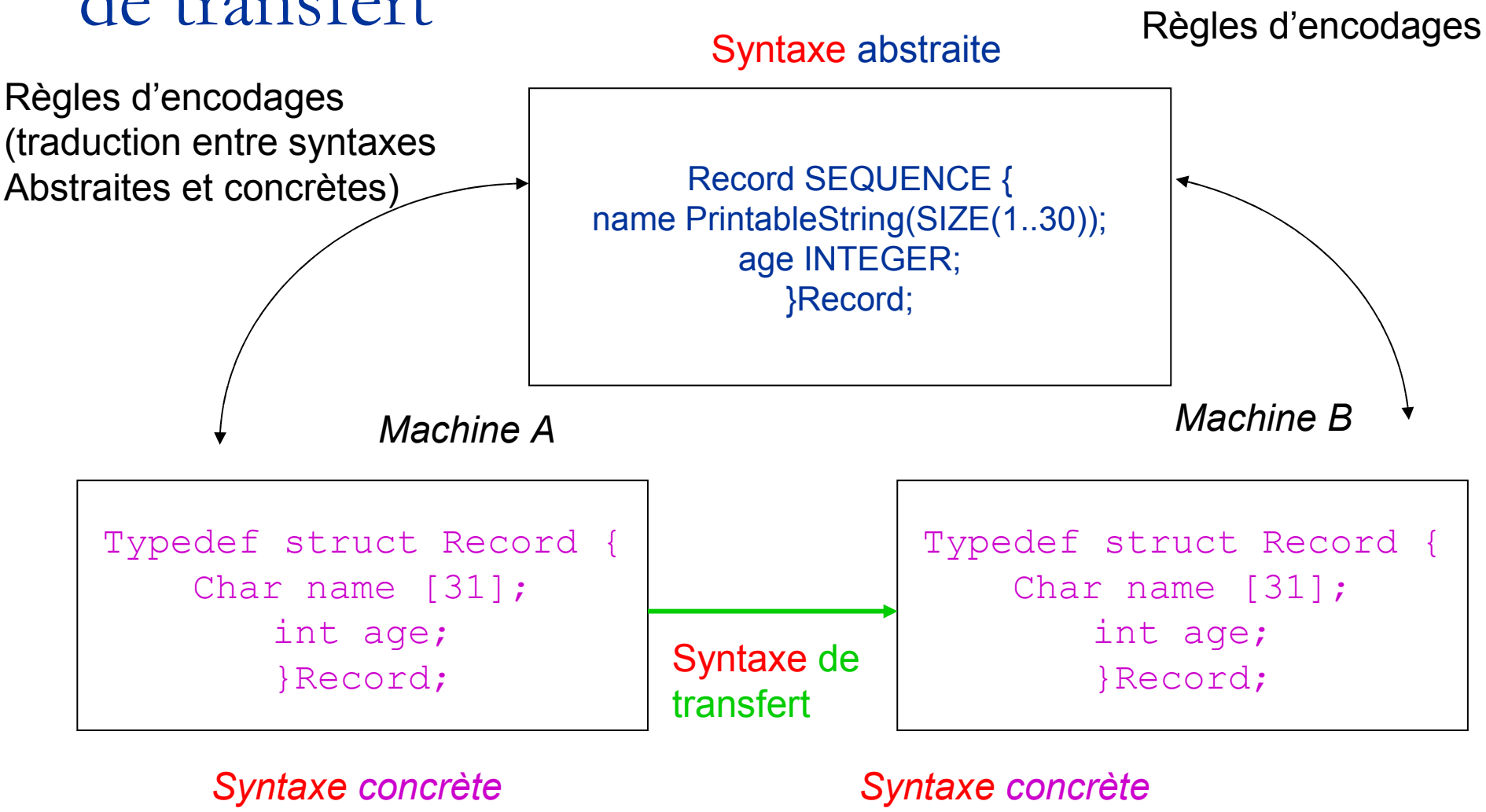
Syntaxe concrète

Question : dans quel ordre transférer ?

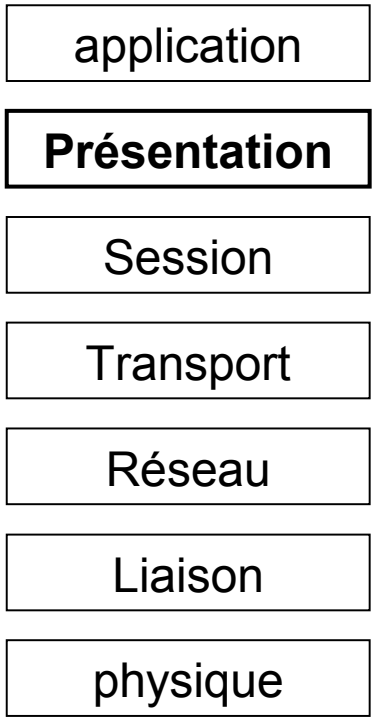
Définition - syntaxe de transfert

- Un flux de bits reçu par une machine est conforme à une syntaxe de transfert
- Cette dernière définit la représentation des données échangées entre deux machines (par exemple l'ordre)
- Elle est fortement liée à la syntaxe abstraite
- Plusieurs syntaxes de transfert peuvent être associées à une syntaxe abstraite

Exemple – syntaxe abstraite, concrète et de transfert

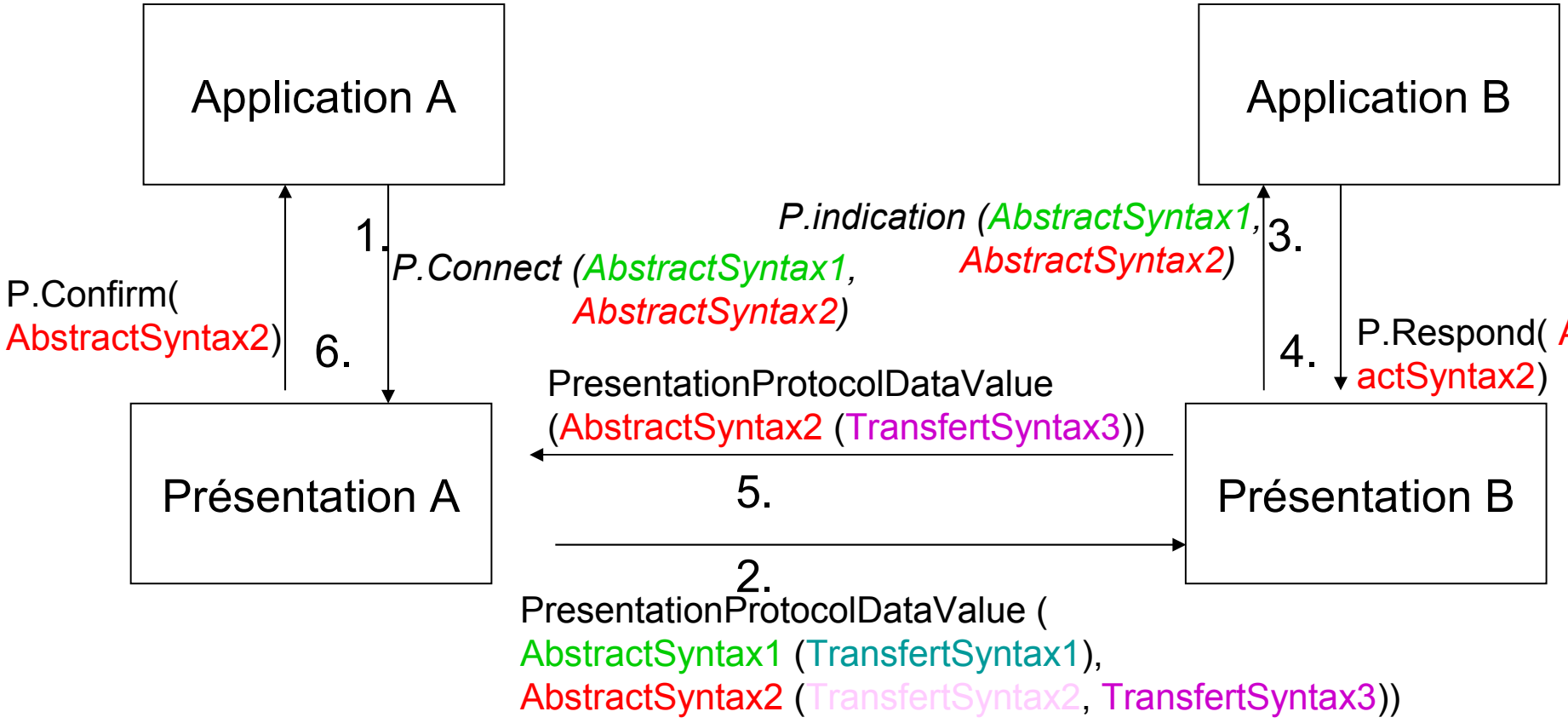


ASN.1 et le modèle OSI



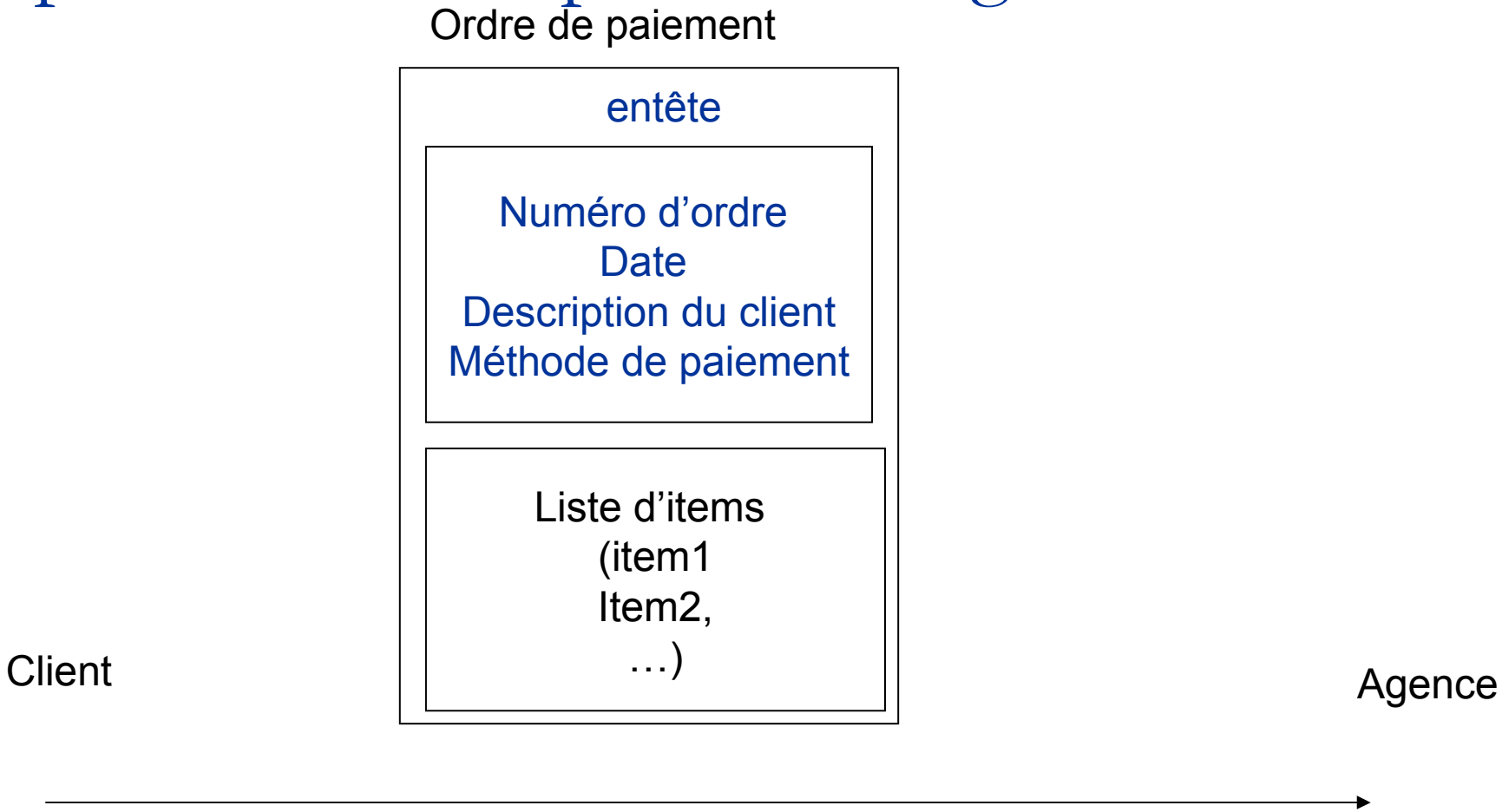
- La couche **présentation**
  - Concerne la syntaxe des données à échanger (syntaxe abstraite) et la syntaxe de transfert (ces données telles qu'elles circulent)
    - Contexte de présentation = Syntaxe abstraite + syntaxe de transfert
  - A pour rôle de
    - Négocier l'encodage (négociation de la syntaxe de transfert) et renégocier (mise à jour)
    - Prendre en charge l'encodage/décodage suivant des règles prédéfinies

Négociation entre couche application et couche présentation



Partie 2 - Syntaxe

Cas d'étude : passage d'un ordre d'achat par un client auprès d'une agence



Etape 1 : définition des types

Lettre majuscule pour les types, minuscules pour le noms

« := » indique une définition

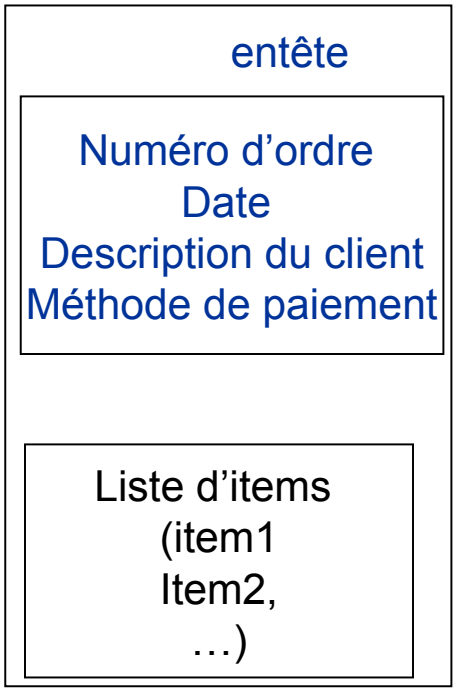
Les mots clés ASN.1
Sont en majuscules

```
Order ::= SEQUENCE {
header Order-header,
items SEQUENCE OF Order-line}
```

Liste de données

```
Order-header ::= SEQUENCE {
number Order-number,
date Date,
client Client,
payment Payment-method }
```

...



Types simples de ASN.1 (1/2)

`BOOLEAN`	Booléen prenant la valeur `TRUE` ou `FALSE`
`NULL`	Valeur NULL (est utile notamment en tant qu'alternative lors d'un choix.
`INTEGER`	Nombre signé
`EXTERNAL, EMBEDDED PDV`	Type comportant une valeur dont le type est défini dans un autre module (type de changement de contexte). PDV : Presentation Data Value
`UTCTime` `GeneralizedTime`	Date (format UTC) ou GMT
`REAL`	Nombre réel

Types simples de ASN.1 (2/2)

`ENUMERATED`	Énumération d'identifiants (par exemple, les états d'une machine). Permet d'associer (implicitement ou explicitement) un nombre à un identifiant
`BIT STRING,` `OCTET STRING,` `CHARACTER STRING,` `String`	Chaîne de bits, d'octets ou de caractères, string
`OBJECT IDENTIFIER,` `RELATIVE OID`	Identification, absolue ou relative, unique d'une entité

Exemple - types simples définis à partir des types ASN.1

- Des types simples peuvent être définis à partir des types ASN.1

Type ASN.1

```
SimpleComputerState ::= BOOLEAN

NoType ::= NULL

NumberOfPeople ::= INTEGER

ComputerStates ::= ENUMERATED { up, down,
  hybernate, frozen } -- nombre fini
  d'états

ArtImage ::= BIT STRING

Data ::= OCTET STRING
```

commentaire

Définition de valeurs simples

assignement

- Des valeurs peuvent être associées aux types ASN.1

```
ok SimpleComputerState ::= TRUE --
  FALSE
down INTEGER ::= -1
bitString BIT STRING ::= '000'B
decimals BIT STRING ::= '243F6A8'H
width REAL ::= 2.57
height REAL ::= {257, 10, -2} -- 2.57
  = 257.10-2
```

binaire

hexadécimal

Exemple – assignation d'une signification

- Autre utilisation : un nom est donné à un nombre pour lui assigner une signification

```
Abort-reason ::= INTEGER {
reason-not-specified(0),
unrecognized-ppdu(1),
unexpected-ppdu(2)}
abort Abort-reason ::= reason-not-
  specified
height REAL ::= {mantisse 257, base 10,
  exposant -2}
```

- Seul le nombre est codé (et est envoyé donc)

ASN.1 et les strings : alphabets basiques, visuels

- ASN.1 offre un support très important aux string
  - `NumericString` : ‘0’-‘9’ et espace
  - `PrintableString` \A"-\Z", \a"-\z", \0"-\9", espace, \'", \ (", \)", \+",\,", \-", \.", \/", \:", \=", \?
  - `VisibleString` **ou** `ISO646String` : ASCII et espace
  - `IA5String` : version nationale de l’ASCII avec par ex. les accents…
  - `TeletexString` ou `T61String` : alphabet roman de 208 caractères, caractères multilingues, graphiques et espace, suppression
  - `VideoString` : orienté audio et visualisation transmis par un réseau téléphonique
  - `GraphicString` : alphabet international et ‘ESC’

ASN.1 et les strings : alphabets génériques

- `GeneralString` : alphabet de `GraphicString` et 'Ctrl', utilisation non recommandée
- `UniversalString` : Unicode
- `BMPString` : utilisation encouragée (forme compacte de `UniversalString`)
- `UTF8String` : syntaxe abstraite équivalente à `UniversalString` mais utilise un encodage spécifique supporté par les servers WEB et navigateur XML

- Pour éviter les problèmes d'interopérabilité (décodage erroné), sous-typer en utilisant
  - La contrainte sur la taille avec `SIZE`
  - Un alphabet contraint avec `FROM`

Exemple – restriction de l'ensemble des valeurs prises

- ASN.1 permet de définir des sous-types en contraignant l'ensemble des valeurs prises
- A cet effet, une variété de syntaxes sont permises

```
Lottery-number ::= INTEGER (1..49)
Lottery-number ::= INTEGER (0<..49)
Lottery-draw ::= SEQUENCE SIZE (6) OF
  Lottery-number
Upper-case-words ::= IA5String (FROM
  ("A".."Z"))
Phone-number ::= NumericString (FROM
  ("0".."9"))(SIZE (10))
O2 ::= OCTET STRING (SIZE(28))
SmallImage ::= BIT STRING (SIZE (1..100))
```

Restriction de L'alphabet

Restriction de la taille

Types construits de ASN.1

`CHOICE`	**Choix** entre des types
`SEQUENCE`	Structure **ordonnée** de valeurs de **types** (généralement) **différents**
`SEQUENCE OF`	Structure **ordonnée** de valeurs de **même type**
`SET`	Structure **non ordonnée** de valeurs de types (généralement) différents
`SET OF`	Structure **non ordonnée** de valeurs de **même type**

Exemple - Séquence

```
--définition des types
IrishStereotype ::= SEQUENCE {
numberOfRainingDays INTEGER,
drinkBeer BOOLEAN OPTIONAL,
}-- assignation des valeurs
irishMan IrishStereotype ::=
  {numberOfRainDeer 365,
drinkBeer TRUE
}
```

Exemple – set et choix

- Set

```
Coordinates ::= SET { x [1] INTEGER,
y [2] INTEGER}
```

- Choix :

```
Afters ::= CHOICE { cheese [0]
  PrintableString,
  dessert[1] PrintableString }
choice Afters ::=
  dessert:"profiterolles"
```

Assignation de la valeur

Exemples – choix : une quantité exprimée suivant trois unités (1/2)

FAUX !!!

```
Quantity ::= CHOICE { units INTEGER,
millimeters INTEGER,
milligrams INTEGER }
```

- Lorsqu'il est encodé , chaque type se voit associé une étiquette (le type `INTEGER` prend l'étiquette 2)
  - Ainsi le récepteur peut interpréter correctement les données arrivant
  - Problème : comme les trois alternatives sont de même type => la même étiquette est attribuée

Exemples – choix : une quantité exprimée suivant trois unités (2/2)

- Solution : associer « manuellement » une étiquette

L'étiquette 0

```
Quantity ::= CHOICE { units [0]
  INTEGER,
millimeters [1] INTEGER,
milligrams [2] INTEGER }
```

- Après 1994 (nouvelle spécification), un tel étiquetage est géré automatiquement par les règles d'encodage mais certains développeurs préfèrent le faire

Autres types structurés nécessitant un étiquetage spécifique

- `SET` est utilisé pour représenter un ensemble non ordonné => une étiquette distincte doit permettre de spécifier de quel type il s'agit
- Même chose pour une séquence ordonnée dans laquelle se trouve un composant optionnel et un composant par défaut

Composants Optionnels et par défaut

- Le mot clé OPTIONAL désigne qu'un composant n'est pas nécessairement transmis (dans l'exemple : drinkBeer)
- Le mot clé DEFAULT définit une valeur par défaut (exemple : la valeur de défaut est 365 de pluie par an)
- Exemple :

```
IrishStereotype ::= SEQUENCE {
numberOfRainingDays INTEGER
DEFAULT default-rainNb,
drinkBeer BOOLEAN OPTIONAL,
}-- assignation des valeurs
irishMan IrishStereotype ::= {numberOfRainDeer 365,
drinkBeer TRUE
}
default-rainNb INTEGER ::= 365
```

Macro

- Permet de définir un nouveau type de notation et une nouvelle valeur
  - Offre un degré de liberté supplémentaire au développeur
  - Fournit une notion plus appropriée pour un domaine d'application spécifique.
- Son but n'est pas d'étendre les types de ASN.1
- Danger: permettre à un développeur de réécrire la grammaire de ASN.1 -> son utilisation n'est pas recommandée et a été rendue obsolète

Macro - définition

```
MACRO-NAME MACRO ::=
BEGIN
TYPE NOTATION ::= -- type syntax
VALUE NOTATION ::= -- value syntax
-- grammatical productions used for
  defining the type syntax and the
  value syntax
END
```

Macro – exemple du macro pour les nombres complexes

Nom de la macro

```
COMPLEX MACRO ::=
BEGIN
TYPE NOTATION ::= "Re" "=" type(ReType) ","
"Im" "=" type(ImType)
VALUE NOTATION ::= value(ReValue ReType) "+"
  value(ImValue ImType) "i"
<VALUE SEQUENCE { real-p ReType, imaginary-p
  ImType} ::=
{ real-p ReValue, imaginary-p ImValue }>
END
```

Exemple – deux instances, c1 et c2 de la macro COMPLEX

- Nombre imaginaire (entiers)

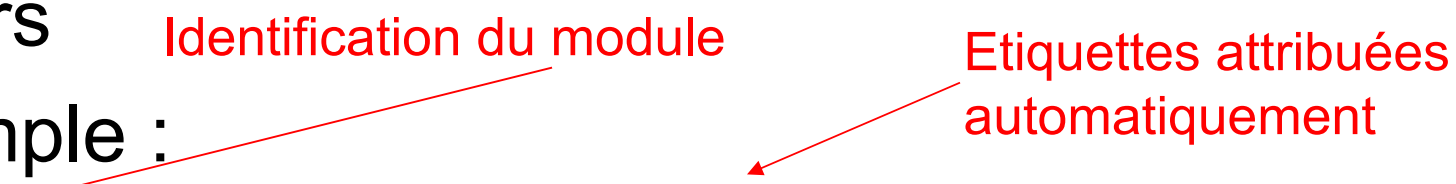
```
c1 COMPLEX
Re = INTEGER,
Im = INTEGER ::= 5 + 3 i
```

- Nombre imaginaire (réels)

```
c2 COMPLEX
Re = REAL,
Im = REAL ::= {56,10,0} + {3561,10,-3}i
```

Module ASN.1

- Constitue la structure de niveau le plus élevé
- Permet de regrouper l'ensemble des types et valeurs
- Exemple :




```
MyModule DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
ProblemType ::= SEQUENCE {
sometimes INTEGER OPTIONAL,
always INTEGER
}
END
```

Contingences pratiques - extensibilité et compatibilité (1/2)

- Les marqueurs '...' (lors de la définition des types comme `CHOICE`, `SEQUENCE`, `ENUMERATED`) ou '`EXTENSIBILITY IMPLIED`' (lors de la définition du module) stipulent que les types seront étendus dans les versions ultérieurs
- Ainsi, le décodeur ne prend pas en compte les champs inconnus

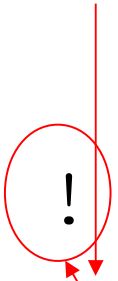
Contingences pratiques - extensibilité et compatibilité (2/2)

```
-- Application
  version 1
Msg ::= SEQUENCE
{
msgID INTEGER,
…
}
```

```
-- Application
  version 2
Msg ::= SEQUENCE
{
msgID INTEGER,
…,
[[
payLoad BIT STRING
]]
}
```

Contingences pratiques - exceptions (2/2)

```
payLoadError INTEGER ::= 1
msgType ::= SEQUENCE {
msgID INTEGER,
payLoad BIT STRING (0.. 1 !
  payLoadError)
}
```



Identifie une exception

La valeur 1 prise par payLoadError est envoyée si le décodage ne peut être correctement effectué

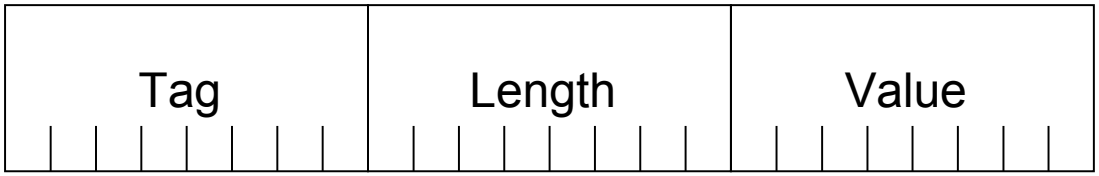
Partie 3 – BER : Basic Encoding Rules

BER : Basic Encoding Rules

- Règles standards utilisées pour encoder (*versus* décoder) en vue d'une transmission
- Règles originalement développées pour ASN.1
  - D'autre règles (comme PER pour Packet Encoding Rules) vont être développées et peuvent être appliquées
- Sont indépendantes de l'architecture, d'un langage de programmation :
  - Les poids prépondérants ont été définis : *big endians* => le bit de poids fort est à gauche
  - Le format d'encodage (TLV) a été choisi de façon à être facilement gérable par les machines

Triplet TLV

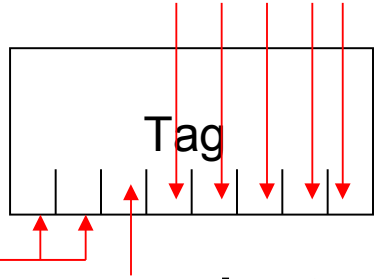
- La syntaxe de transfert est définie sous la forme d'un triplet TLV :



- Chaque champs est une série **d'octets** : la taille d'un champs est un multiple de 8
- Length : longueur de la valeur exprimée en nombre d'octets
- Tag (étiquette): valeur du type
- Value : valeur
- Les champs T, L, V peuvent faire plus de 1 octet

Champs Tag (1/2)

- Bits 1-5 : numéro de tag

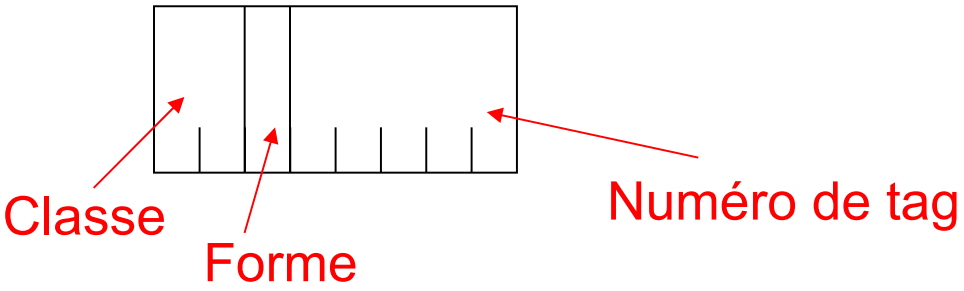


- Bit 6 : forme du type primitif (= 0) ou construit (=1)
- Bits 7 et 8 : classe

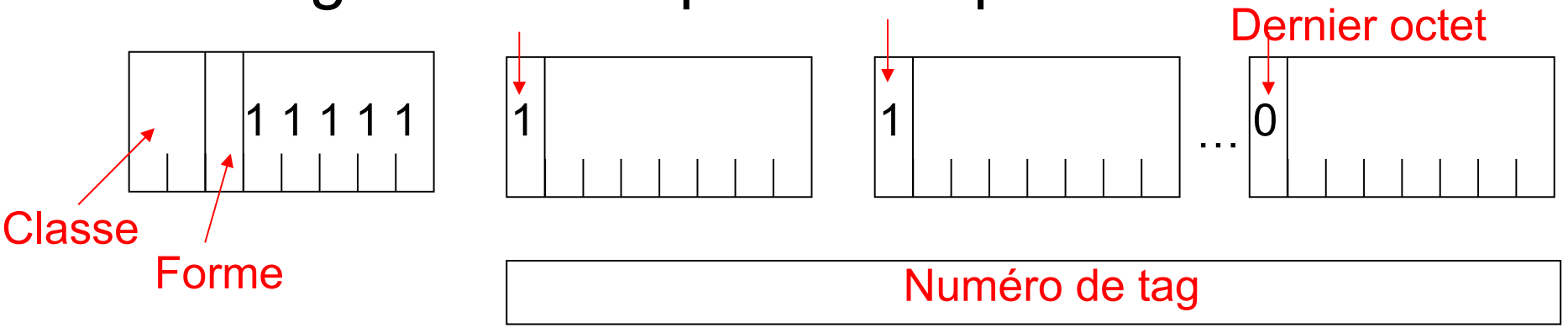
Classe	**Bit 8**	**Bit 7**
UNIVERSAL	0	0
APPLICATION	0	1
Context-specific	1	0
PRIVATE	1	1

Champs Tag (2/2)

- Codage d'une étiquette sur un octet

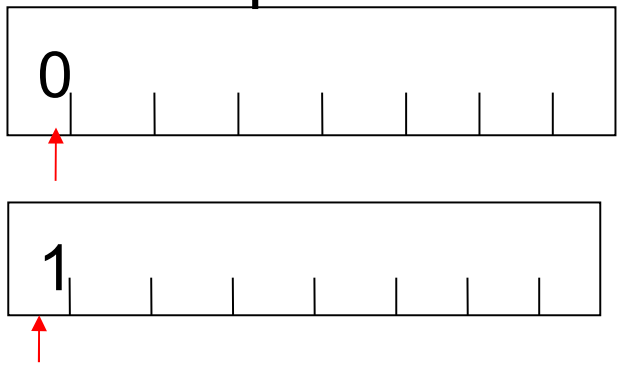


- Codage d'une étiquette sur plusieurs octets

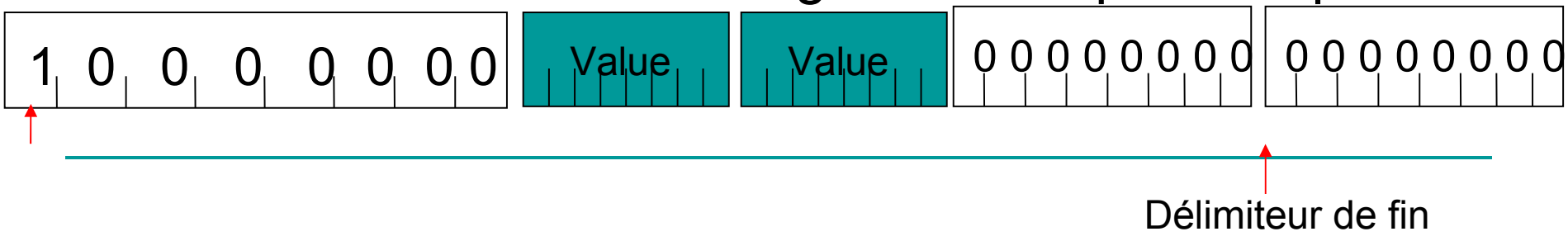


Champs Length

- La taille (length) du champs valeur, prend 2 formes :
  - **Forme primitive** : la longueur est indiquée
    - Si length < = 127
    - Si 127 < length < $256^{126} - 1$

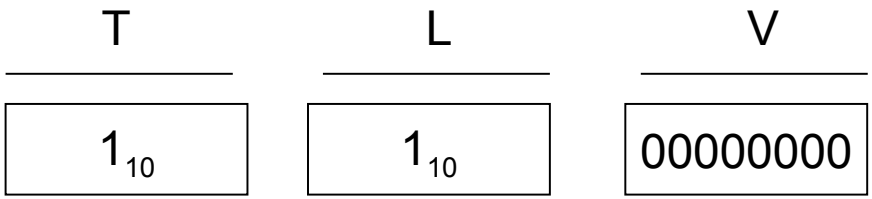


  - **Forme indéfinie** : la longueur n'est pas indiquée



Champs Valeur

- Le codage du champs valeur dépend du type de données
- Exemple : encodage de la valeur booléenne TRUE

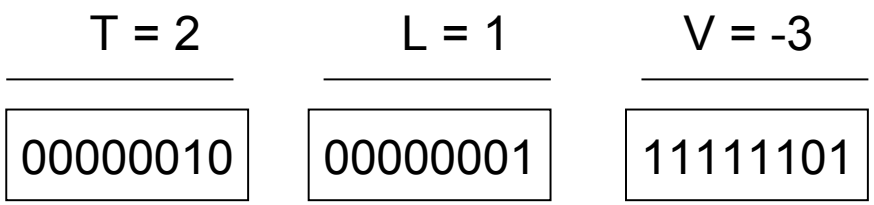


Règles d'encodage – booléen, entier

- `BOOLEAN` :
  - Type primitif, étiquette
  - Codage :
    - `TRUE` est encodé 00
    - `FALSE` prend n'importe quelle valeur à l'exception de 00
- `INTEGER` :
  - Type primitif
  - Codage : nombre binaire codé sur le minimum d'octets

Exemple – encodage de types primitifs

- Encodage entier -3
  - Valeur binaire de -3 ?
    - Valeur binaire de 3 : 00000011
    - Complément à 1 de 3 : 11111100
    - Complément à 2 de 3 : 11111101
  - Encodage

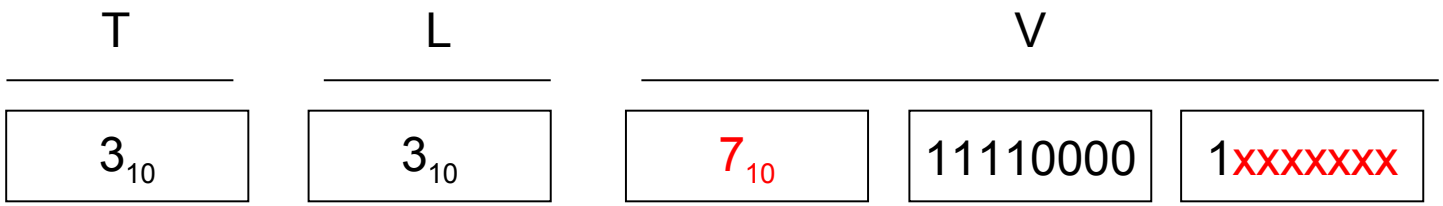


Encodage chaîne binaire

- Deux formes d'encodage peuvent être utilisés pour les chaînes (string) binaires
  - Forme primitive
  - Forme construite (comme avec un type structuré)

Exemple – chaîne binaire, type primitif

- `BIT STRING '111100001'B` dont la longueur n'est pas un multiple de 8

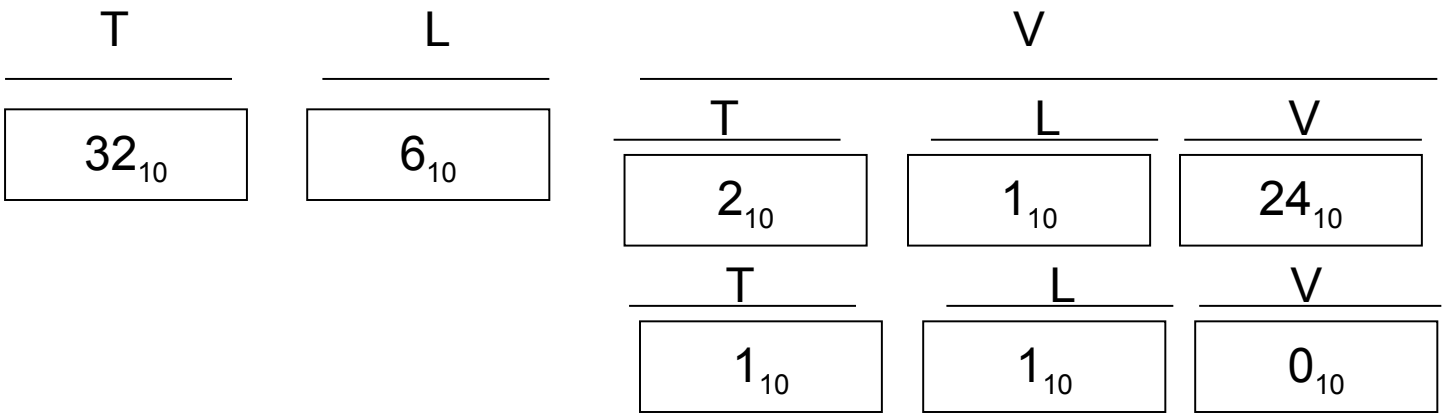


  - Un octet est rajouté pour définir le nombre de bits de bourrage
  - x peut prendre la valeur 0 ou 1

Exemple – type construit

- La forme récursive de TLV permet de représenter des types structurés

```
v SEQUENCE { age INTEGER,
single BOOLEAN } ::= { age 24,
single TRUE }
```



Valeurs prises par les étiquettes

0 réservé
1 BOOLEAN
2 INTEGER
3 BIT STRING
4 OCTET STRING
5 NULL
6 OBJECT IDENTIFIER
7 ObjectDescriptor
8 EXTERNAL, INSTANCE OF
9 REAL
10 ENUMERATED
11 EMBEDDED PDV
12 UTF8String
13 RELATIVE-OID
14 réservé
15 réservé
16 SEQUENCE, SEQUENCE OF
17 SET, SET OF
18 NumericString
19 PrintableString
20 TeletexString, T61String
21 VideotexString
22 IA5String
23 UTCTime
24 GeneralizedTime
25 GraphicString
26 VisibleString, ISO646String
27 GeneralString
28 UniversalString
29 CHARACTER STRING
30 BMPString
31... Réservé

Etiquette (tag)

- Est associée, souvent pas défaut, à chaque module ASN.1
- Est un couple (classe et numéro d'étiquette) :
- `[[class] number] access Type`
  - `class = UNIVERSAL | APPLICATION | PRIVATE`
  - `access = IMPLICIT | EXPLICIT`
- Exemple :

```
Cercle ::= CHOICE {
diameter [PRIVATE 0] IMPLICIT REAL,
beam [PRIVATE 1] IMPLICIT REAL}
```

- Il existe 4 classes d'étiquettes permettant de définir la portée de l'étiquette ➔ Un tag peut être réutilisé hors de cette portée

Classes d'étiquettes (1/2)

- `UNIVERSAL` :
  - Classe réservée au standard ASN.1
  - Ne peut être utilisée par le développeur/spécifieur
- `Context-specific`
  - définit la portée du contexte dans laquelle l'étiquette est unique
    - Si la classe est absente, le tag est spécifique au contexte
    - Ne peut apparaître que dans les types structurés
  - Exemple : 2 étiquettes de portée restreinte au choix

```
Afters ::= CHOICE { cheese [0] IA5String,
dessert [1] IA5String }
```

Pas de nom de classe

Classes d'étiquettes (2/2)

- `APPLICATION`
  - Est spécifique à une application (par ex. service d'annuaire X.500)
  - son utilisation est peu recommandée abolie en 1994

```
Order-number ::=
[APPLICATION 0] NumericString (SIZE (12))
```

- `PRIVATE`
  - Classe utilisée de façon à définir des types de données dans une organisation (compagnie par ex.) ou à un pays
  - son utilisation est peu recommandée abolie en 1994

Etiquette (tag)

- Une étiquette peut aussi être associée manuellement en cas ambigüité. L'attribution « manuelle » d'étiquette est fastidieuse et complexe
- Après 1994 (nouvelle spécification), cet étiquetage manuel est gérée automatiquement aux niveau des règles d'encodage
- Ce changement est une source potentiel d'erreur
  - L'étiquetage est-il géré manuellement? Ou une partie est elle gérée automatiquement?
  - Il est nécessaire de clarifier qui/comment réalise l'encodage

Etiquetage implicite, explicite, automatique

- Solution : stipuler comment l'étiquetage est effectué
  - Étiquetage explicit
  - Etiquetage implicit
  - Etiquetage automatique

Etiquetage implicite

- L'étiquette courante est remplacée par la nouvelle étiquette attribuée
  - Représentation : TLV
  - Syntaxe : `tag IMPLICIT typeDefinition`
  - Exemple :

```
Binfile:= [APPLICATION 3] IMPLICIT OCTET STRING
```

  - Il s'agit de étiquetage le plus commun
  - L'étiquette n'identifie plus le type mais l'implique
    - Le type CHOICE ne doit pas être étiqueté implicitement (car l'étiquetage serait ambiguë)

Etiquetage explicite

- L'étiquette courante s'ajoute à la nouvelle étiquette attribuée
  - Représentation : TL TLV …
  - Désavantage : longueur
  - Syntaxe : tag EXPLICIT typeDefinition
  - Peut être utilisé dans toutes les circonstances
    - Par défaut, l'étiquetage est explicite
  - Exemple :

```
BinEncod := [PRIVATE 8] EXPLICIT BIT STRING
```

Synthèse BER – avantages

- La structure TLV est particulièrement **simple** (tableau) : elle permet de gérer facilement plusieurs variables tout en supportant la récursivité
  - Le champs T permet :
    - De reconnaître la présence/l'absence d'éléments optionnels
    - De pouvoir spécifier des choix
  - Le champs L facilite la gestion de champs de tailles variables (par ex. des chaînes)

Synthèse BER – désavantages

- BER est **verbeux** (comparé à un protocole envoyant des bits)
  - Avec TLV : tout est un multiple de 8 : 3 octets pour l'envoi d'un booléen!
  - Le champs T pourrait souvent se déduire de la syntaxe abstraite
  - Le champs L est certaines fois redondant
  - BER **préserve la syntaxe abstraite** en surchargeant
- Beaucoup d'alternatives apparaissent : ASN.1 n'est pas entravé par un seul style d'encodage

Exemple d'alternative à BER : PER - Packet Encoding Rules

- vise à obtenir un encodage compacte mais simple
- est moins facile à lire
- Structure : [Preamble] [Length] [Value]
- PLV ne sont plus des séries d'octets mais de bits
- Le champs lenght devient optionnel (n'est ajouté que si nécessaire)
- PER n'est plus « self-contain » : plus d'étiquettes : il nécessite que l'analyse la syntaxe abstraite

Exemple d'alternative à BER – XER : XML Encoding Rules

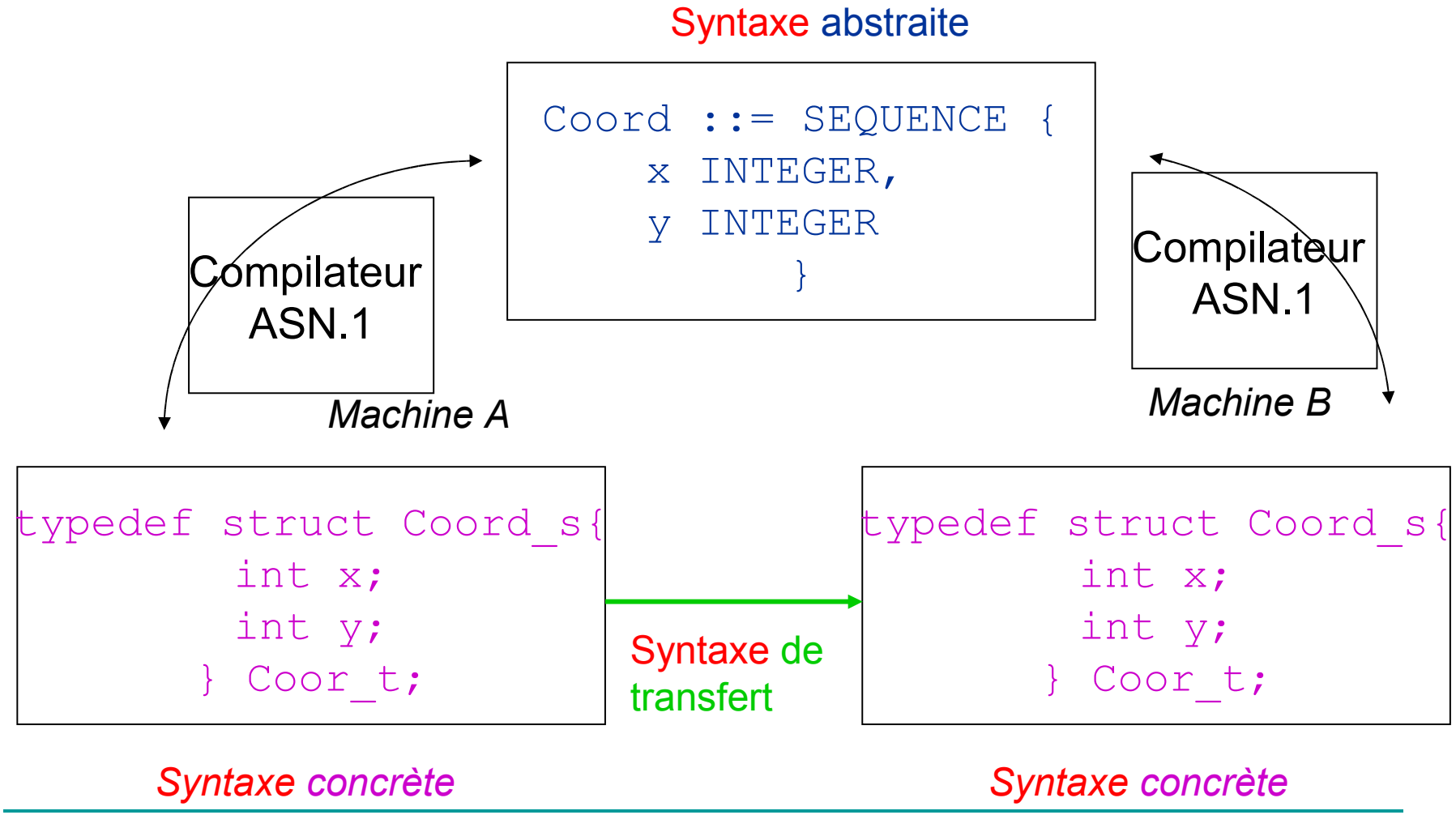
- Pouvoir visualiser avec un navigateur un document ASN.1
- pouvoir d'encoder des documents XML binaires avec PER

```
ChildInformation ::= SEQUENCE {
name AnyName,
dateOfBirth INTEGER (1..MAX) }
   -- yyyymmdd
AnyName ::= SEQUENCE {
givenName VisibleString,
initial VisibleString (SIZE
   (1)) OPTIONAL,
familyName VisibleString }
```

```
<ChildInformation>
<name>
<givenName> Lee </givenName>
<familyName> Owen
   </familyName>
</name>
<dateOfBirth> 19501003
   </dateOfBirth>
</ChildInformation>
```

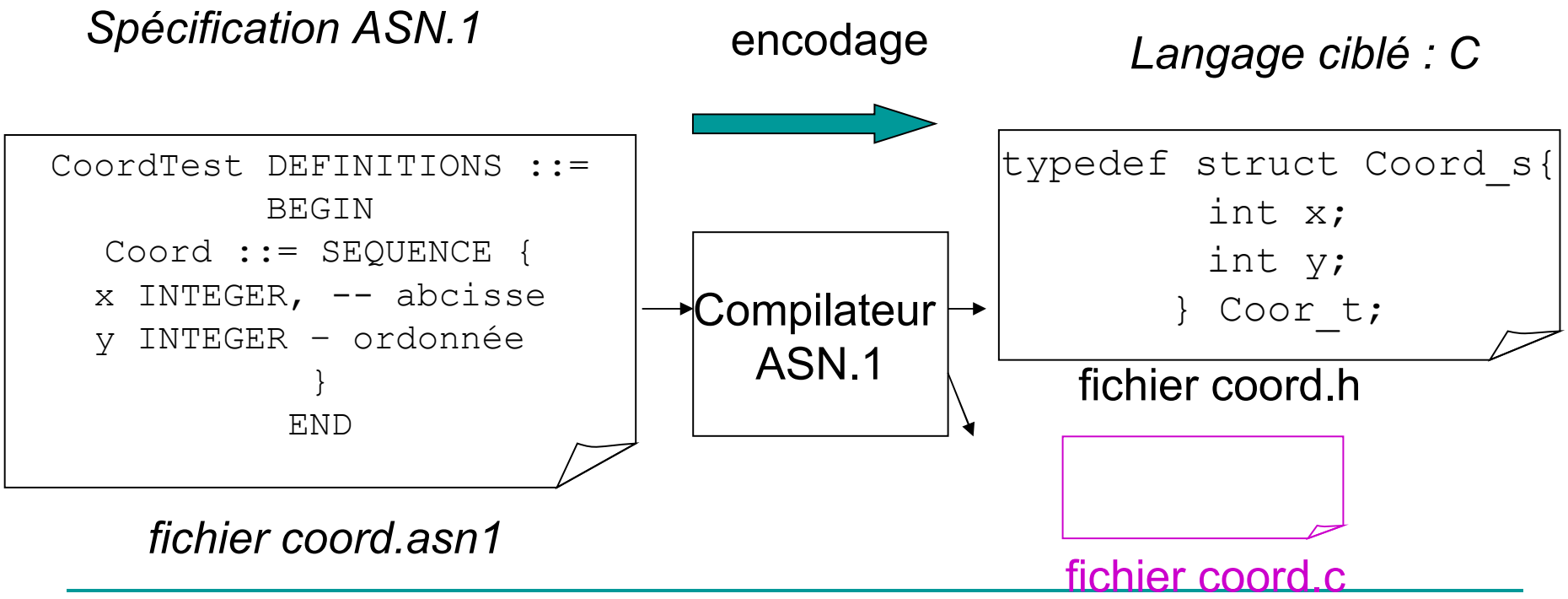
Outils ASN.1

Compilation ASN.1



Compilateur ASN.1 – codage, machine A

- Rôle : convertir la spécification ASN.1 dans un autre langage (exemple : langage C)

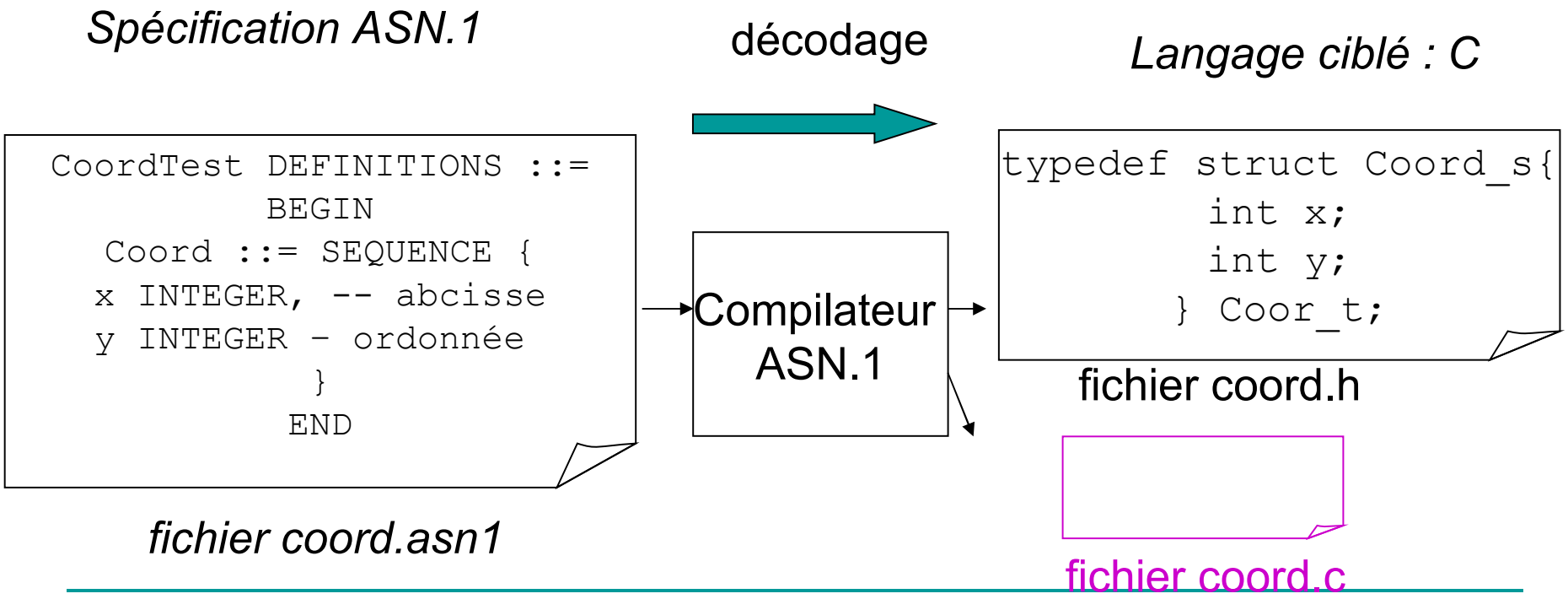


Fichier coord.c, codage, machine A

```
#include <Rectangle.h>
…
int main (){
Coord_t *coordonnees; // Coord_t est défini dans le fichier .h
Coordonnes.x=10;
Coordonnees.y=2;
/* ouverture fichier binaire en écriture pour une sortie BER*/
FILE *fp = fopen(filename, ”wb”);
/* Encodage du type Coord avec BER */
 ber_encode(&asn_DEF_Coord, coordonnees, write_out, fp);
fclose(fp);//et voila !
}
```

Compilateur ASN.1-décodage, machine B

- Rôle : convertir la spécification ASN.1 dans un autre langage (exemple : langage C)



Fichier coord.c, décodage, machine B

```
#include <Rectangle.h>
…
int main (){
Coord_t *coordonnees = 0; // Coord_t est défini dans le fichier .h
char buf[1024]; /* buffer */
/* ouverture fichier binaire en lecture pour une entrée BER*/
FILE *fp = fopen(filename, ”rb”);
size_t size = fread(buf, 1, sizeof(buf), fp;//nb octets reçus
…
/* Decodage du type Coord avec BER */
ber_decode(0, &asn_DEF_Coord,(void **)&coordonnees,
            buf, size);
…
}
```

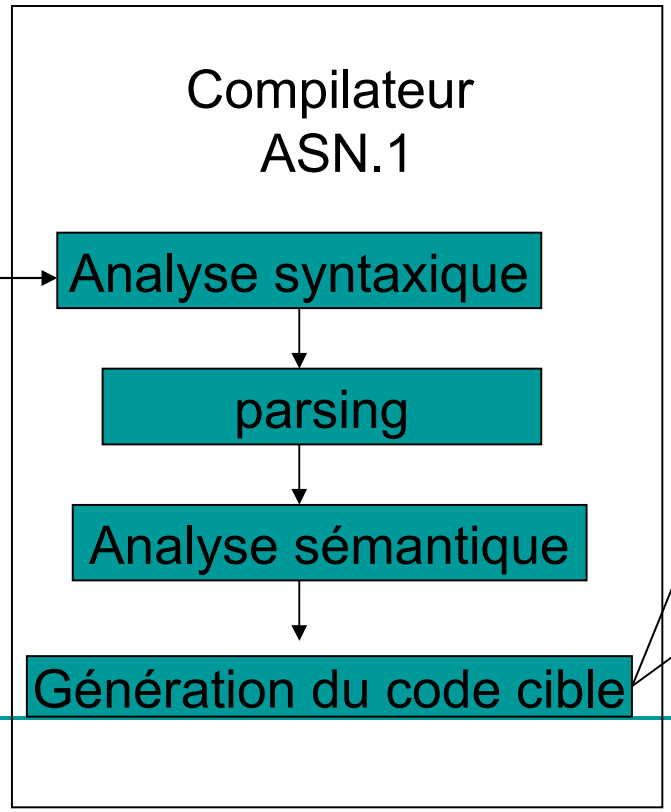
Etape de compilation ASN.1, codage, Machine A

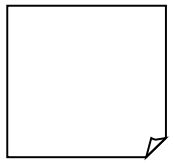
- 4 étapes nécessaires à la compilation

Spécification ASN.1

```
CoordTest DEFINITIONS ::=
          BEGIN
   Coord ::= SEQUENCE {
  x INTEGER, -- abcisse
  y INTEGER – ordonnée
            }
          END
```

fichier coord.asn1





fichier coord.h

fichier coord.c

Approfondissement – réalisation codeur-décodeur

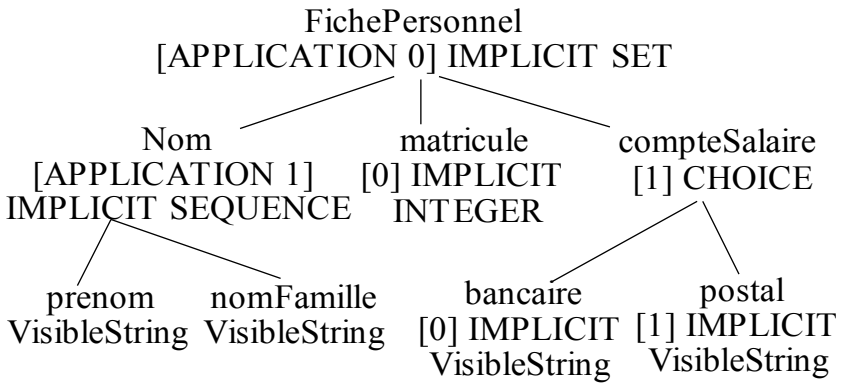
- Le codeur/décodeur doit réaliser la conversion entre la représentation concrète de chaque machine et la syntaxe de transfert.
- Pour faciliter ces conversions de syntaxe, il est commode d'adopter une représentation de la syntaxe abstraite dont la structure reflète l'APDU
- L'analyseur syntaxique d'une définition de données en syntaxe abstraite génère une structure d'arbre (première étape habituelle d'un compilateur) sur lequel travaille ensuite les convertisseurs..

Extrait du cours d'Eric Gressier-Soudan

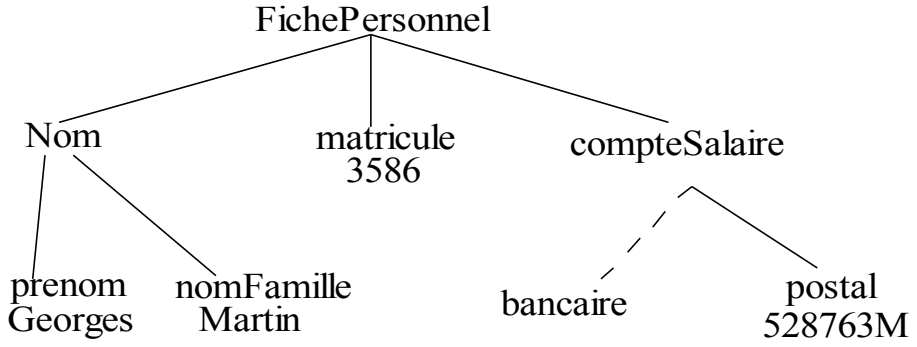
Exemple

- FichePersonnel ::= [APPLICATION 0] IMPLICIT SET {Nom,
- matricule [0] IMPLICIT INTEGER,
- compteSalaire[1] CHOICE {
- bancaire [0] IMPLICIT VisibleString,
- postal [1] IMPLICIT VisibleString}}
- Nom ::= [APPLICATION 1] IMPLICIT SEQUENCE {
- prenom VisibleString,
- nomFamille VisibleString}

Représentation par arbre de type



Arbre des valeurs pour la ficheMartin



Extrait du cours d'Eric Gressier-Soudan

Fonctionnement du codeur

- **Première passe**

```
60   *                                   fiche perso
     61   *                              nom
               1A   07   47656F72476573   prénom
               1A   06   4D417274494E     nomFamille
     80   02        0E02                 matricule
     A1   *                              compte
               81   02   3532383736334D   postal
```

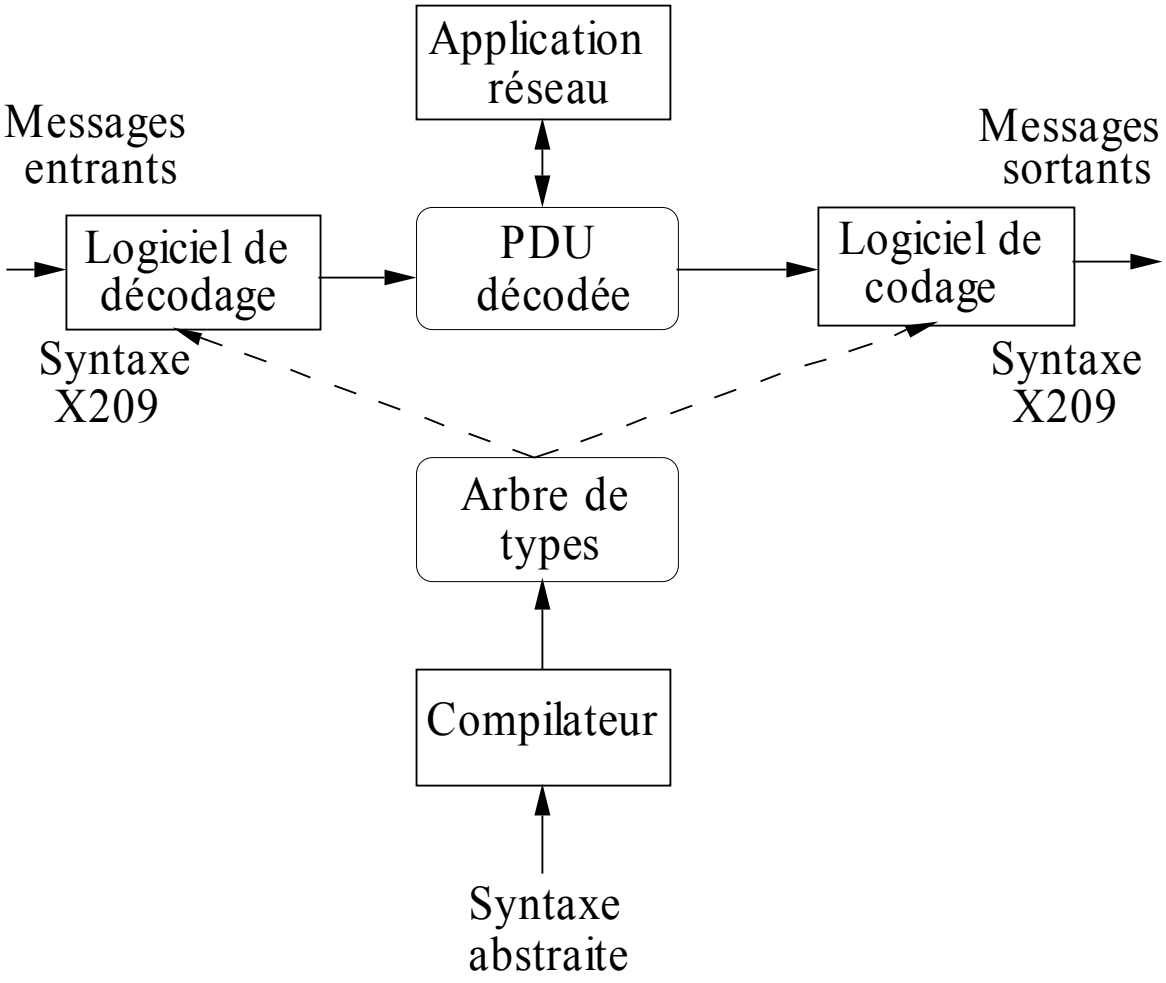
  - Initialisation des zones valeurs qui correspondent aux symboles terminaux avec détermination des longueurs dans ce cas
  - Les zones longueurs des nœuds intermédiaires sont non calculées.

- **Deuxième passe** : détermination complète des longueurs

```
60   22
     61   11                             nom
               1A   07   47656F72476573   prenom
               1A   06   4D417274494E     nomFamille
     80   02        0E02                 matricule
     A1   09                             compte
               81   02   3532383736334D   postal
```

Extrait du cours d'Eric Gressier-Soudan

Outils logiciels – codeur, décodeur



Extrait du cours d'Eric Gressier-Soudan

Usage de ASN.1 (1/2)

- ASN.1 un langage d'excellence pour définir les messages d'un protocole
- La définition ASN.1 peut être combinée avec une ou plusieurs règles de transfert utilisés pour générer les codeurs et décodeurs du protocole
- Beaucoup de normes utilisent ASN.1
  - Messagerie X400
  - Système d'annuaire X500
  - Dans le domaine de la supervision
    - SNMP
    - CMIP

Conclusion

- ASN.1 est un standard international permettant de
  - Décorréler clairement l'information de l'encodage
  - fournir une spécification claire des informations
  - Ajouter/soustraire facilement des messages
  - Faciliter le développement de protocoles (test & validation rapide)
- Plusieurs outils, libres/propriétaires, permettent de « traduire » une spécification ASN.1 en C, C++, java
  - Plus d'informations à :
    - http://www.itu.int/ITU-T/asn1/
    - **http://www.asn1.org**