

ASN.1 Tamamlandı

ile

Profesör John Larmouth

özveri

Bu kitap, Withington Girls' School'da kızım Sarah-Jayne ile birlikte bulunan kızlara ve bir gün işe yarayabileceğini umuduyla, oğlum James ile birlikte Manchester Grammar School'da bulunan erkeklerle ithaf edilmiş tir. bazıları!

İçindekiler

İçindekiler	3
Önsöz	13
giriş	15
1 Küresel iletişim im altyapısı 2 ASN.1 tam olarak nedir?	15
16	
3 ASN.1 ile geliş tirme süreci 4 Metnin yapısı.	18
	18
BÖLÜM I ASN.1 GENEL BAKIŞ	21
Bölüm 1 Protokollerin özellikleri 1 Protokol nedir?	22
	22
2 Protokol belirtimi - bazı temel kavramlar 2.1 Katman oluş turma ve protokol "delikleri"	24
	25
2.2 Katmanlanmanın ilk geliş meleri 2.3	26
Katmanlanmanın dezavantajları - basit tutun!	28
2.4 Geniş letilebilirlik	28
2.5 Soyutlama ve aktarım sözdizimi 2.6	30
Komut satırı veya deyim tabanlı yaklaşım 2.7 Arayüz Tanımlama Dilinin Kullanımı	32
	32
3 Soyut ve aktarım söz dizimleri hakkında daha fazla bilgi 3.1 Soyut değil erler ve türleri 3.2	32
	32
Soyut değil erlerin kodlanması 4	33
Değ erlendirmeli tartış ma	35
4.1 Bir kedinin derisini yüzmenin birçok yolu vardır - fark eder mi?	35
4.2 Çoklu aktarım söz dizimleriyle erken çalış ma 4.3	35
Faydalalar	36
Yerel temsillerin verimli kullanımı	36
Zaman içinde geliş tırılmış temsiller	36
Kodlama ş emalarının yeniden kullanımı	36
kodun yapılandırılması	37
Kodun ve ortak araçların yeniden kullanımı	38
Test ve hat izleme araçları	38
Birden çok belge "yapış tırıcı" gerektirir	38
"Araçlar" iş i	39
5 Protokol belirtimi ve uygulaması - bir dizi vaka çalışması 5.1 Sekizli diziler ve sekizli içindeki alanlar 5.2 TLV yaklaşımı 5.3 EDIFACT grafik sözdizimi 5.4 Karakter tabanlı bir sözdizimi belirtmek için BNF kullanımı 5.5 ASN.1 kullanılarak belirtim ve uygulama - 1980'lerin başı 5.6 ASN.1 - 1990'ları kullanarak ş artname ve uygulama	39
	39
	40
	41
	42
	43
	44
Bölüm 2 ASN.1'e Giriş	47
© OS, 31 Mayıs 1999	3

1 Giriş	47
2 Örnek 2.1	
Üst düzey yazı tipi	48
2.2 Önemli olan kalın yazıdır!	49
	49
2.3 İ talik isimler bir ş eyleri birbirine bağ lamak için kullanılır	49
2.4 Normal yazı tipindeki isimler alan/ş e/ş e isimleridir	50
2.5 Örneğ e geri dən!	50
	50
2.6 BranchIdentification türü	52
2.7 Bu etiketler	
3 Farklı yazı tiplerinden kurtulmak	54
4 Bazı	
eksik uçları birleş tirmek	55
4.1 Tür ve deñ er	
atamalarının çəzeti	56
4.2 Adların biçimi	56
4.3 Düzen	
ve yorum	56
5 Peki baş ka ne bilmeniz gerekiyor?	57
	57
	58
	58
Bölüm 3 Bir ASN.1 spesifikasiyonunun yapılandırılması	60
1 Örnek 2 ASN.1	61
spesifikasiyonları için yayın stili	62
2.1 Satır numaralarının kullanımı.	63
2.2 ASN.1 metnini çoğ altma	64
2.3 Makine tarafından okunabilir kopya sağ lama	64
3 Modül başlığ ina dönüyoruz!	65
3.1 Sözdizimsel tartışma	65
3.2 Etiketleme ortamı	67
3.2.1 Açık etiketleme	
ortamı	68
3.2.2 Örtülü etiketleme ortamı	68
3.2.3 Otomatik etiketleme ortamı	68
3.3 Geniş letilebilirlik ortamı	68
	69
4 İ hracat/ithalat bildirimleri	71
5 Yapımızı iyileş tirmek	73
6 Tam özellikler	76
7. Karar	77
Bölüm 4 Temel veri türleri ve oluş turma mekanizmaları - kapatma	78
1 Örnek çizim 2 Yerleşik türlerin tartışılması	79
2.1 BOOLEAN tipi	80
2.2 INTEGER	
tipi	80
2.3 ENUMERATED tipi	80
2.4 REAL tipi	
2.5 BIT STRING tipi	80
2.6 OCTET STRING	
tipi	82
2.7 NULL tipi	82
2.8 Bazı karakter dizisi	
tipleri	83
2.9 OBJECT TANIMLAYICI tipi	83
2.10 ObjectDescriptor tipi	84
2.11 İ ki ASN.1	
tarih/saat tipi	87
	88
	88
	89
	89
	90
	91
3 Ek notasyon yapıları	93
3.1 Seçim tipi	
notasyon	93
3.2 Notasyonun Bİ LEŞ ENLERİ	
3.3 Dİ Zİ veya KÜME?	94
	95
3.4 Dİ Zİ , AYAR ve SEÇİ M (vb) deñ er gösterimi	96
4 X.680/ISO 8824-1'de baş ka neler var?	97

Bölüm 5 Daha karmaşık alanlara referans 1 Nesne tanımlayıcıları 2 Karakter dizisi türleri 3 Alt tipleme 4 Etiketleme 5 Geniş letilebilirlik, istisnalar ve sürüm parantezleri 6 Delik türleri 7 Makrolar 8 Bilgi nesnesi sınıfları ve nesneleri ve nesne kümeleri 9 Diğer er kısıtlama türleri 10 Parametrelendirme 12 ASN.1 semantik model 13 Sonuç	99 100 100 102 103 104 105 106 107 108 108 109 109
Bölüm 6 ASN.1 derleyicisi kullanma 1 Uygulamaya giden yol 2 ASN.1 derleyicisi nedir? 3 Bir ASN.1 derleyici aracının genel özellikleri 4 Basit bir kodlama/kod çözme yordamları kitaplığıının kullanımı 4.1 Kodlama 4.2 Kod çözme 5 Bir ASN.1 derleyici aracı kullanma 5.1 Temel hususlar 5.2 Araç tasarımcılarının neye karar vermesi gereklidir? 5.3 Bir programlama dili veri yapısına eş leme 5.4 Çalışma zamanında bellek ve CPU takasları 5.5 Bir aracın kontrolü	110 110 111 113 113 114 115 116 116 116 117 118 119
6 "OSS ASN.1 Tools" ürününün kullanımı 7 Bir ASN.1 derleyici aracını diğer erinden daha iyi yapan nedir? 8 Sonuç	120 121 122
Bölüm 7 ASN.1 belirtimi ve uygulaması için yönetim ve tasarım sorunları 1 Yönetim kararları için küresel sorunlar 1.1 Belirtim 1.1.1 ASN.1'i kullanmak ya da kullanmamak! 1.1.2 Kopyalamak mı yoksa kopyalamamak mı? 1.2 Uygulama - bütçeyi belirleme 1.2.1 Spesifikasyonları alma 1.2.2 Eğitim kursları, öğrenici ve danışmanlar 1.3 Uygulama platformu ve araçları 2 Belirleyiciler için sorunlar 2.1 Kılavuz ilkeler 2.2 Tarza ilişkilerin kararları 2.3 En üst düzey türünüz 2.4 Tamsayı boyutları ve sınırları 2.5 Geniş letilebilirlik sorunları 2.6 1. istisna iş leme 2.6.1 Gereksinim 2.6.2 Yaygın istisna iş leme 2.6.2.1 DİZİ ve KÜME 2.6.2.2 SEÇİMLER 2.6.2.3 INTEGER ve SAYILANDIRILMIŞ 2.6.2.4 Geniş letilebilir dizeler 2.6.2.5 SET OF ve SEQUENCE OF üzerinde geniş letilebilir sınırlar	123 124 124 124 125 125 126 126 127 127 128 128 129 130 131 131 131 131 132 132 132

2.6.2.6 Kısıtlamalarda geniş letilebilir nesne kümelerinin kullanımı	133
2.6.2.7 Özet 2.6.3 ASN.1'de belirtilen varsayılan istisna iş leme 2.6.4 Resmi istisna belirtim göstergesinin kullanımı	133
2.7 Parametreleme sorunları 2.8	134
Kısıtlanmamış açık türler 2.9	135
Etiketleme sorunları 2.10 Basit tutma	136
3 Uygulayıcılar için sorunlar 3.1 Yol gösterici ilkeler 3.2 Aracınızı öğren 3.3	136
Tam sayıların boyutları 3.4	137
Spesifikasyonlardaki belirsizlikler ve uygulama bağlılığı 3.5 Düzeltme	138
3.6 Geniş letilebilirlik ve istisna iş leme 3.7 El kodlamalarına dikkat 3.8 Posta listeleri	139
3.9 İyi mühendislik - sürüm 2 **gelecek**!	139
	139
	140
	140
	140
4. Sonuç	141
BÖLÜM II DİĞER DETAYLAR	142
Bölüm 1 Nesne tanımlayıcı türü	143
1. Giriş	143
2 Nesne tanımlayıcı ağacı	145
3 Bilgi nesneleri	146
4 Değer gösterimi	147
5 Nesne tanımlayıcı tipinin kullanımları	148
Bölüm 2 Karakter dizisi türleri 1 Giriş 2 NumericString 3 PrintableString 4	149
VisibleString (ISO646String)	150
	150
	151
	151
5 IA5Dizesi 6	152
TeleteksDizesi (T61Dizesi)	152
7 VideotexString 8	152
GraphicString 9	153
GeneralString 10	153
UniversalString 11	153
BMPString 12 UTF8String	153
13 Önerilen karakter dizisi türleri 14 Karakter dizisi türleri için değer gösterimi 15 ASN.1-CHARACTER-MODULE 16 Sonuç	154
	154
	155
	157
	158
Bölüm 3 Alt Tipleme	159
1. Giriş	159
2 Temel kavramlar ve küme aritmetiği	160
3 Tek değer alt tipi	162
4 Değer aralığı alt tipi	162
5 İzin verilen alfabe kısıtlamaları	163

6 Boyut kısıtlamaları 7	164
İ çeren alt tip kısıtlamaları 8 İ ç Alt Tip	166
8.1 Giriş 8.2 Wineco Protokolü Alt	166
Kümesi 8.3 Bir dizinin iç alt tipi 9 Sonuç	166
	168
	170
	171
 Bölüm 4 Etiketleme	172
1 Önceki tartış maların gözden geçirilmesi	172
2 etiket ad alanı	173
3 Soyut bir etiketleme modeli	176
4 Etiketlerin ne zaman farklı olması gerekiğine ilişkin kurallar	179
5 Otomatik etiketleme	180
6. Sonuç	180
 Bölüm 5 Geniş letilebilirlik, İstisnalar ve Sürüm Parantezler 1	181
Geniş letilebilirlik kavramı 2 Uzanti işaretçisi 3 İstisna belirtimi 4 Üç nokta nereye yerleş tirilebilir?	181
	182
	183
	183
5 Sürüm parantezleri 6	184
{...} gösterimi 7	185
Geniş letilebilirlik ve etiketleme arasındaki etkileşim 8	185
Son açıklamalar	187
 Bölüm 6 Bilgi Nesnesi Sınıfları, Kısıtlamalar ve Parametrelendirme	188
1 "deliklere" duyulan ihtiyaç ve onlar için notasyon desteği	189
1.1 OSI Katmanı 1.2	189
ASN.1'de delik desteği 2 ROSE	190
çağırma modeli 2.1 Giriş 2.2 INVOKE	190
mesajına yanıt verme 3 Kullanıcı	190
belirtimini tamamlamak için tabloların kullanımı	192
	193
3.1 Özelden genele 4 Tablolardan	195
Bilgi Nesnesi Sınıflarına 5 ROSE İŞ LEMİ ve HATA	196
Nesnesi Sınıfı tanımları 6 Bilgi Nesnelerini Tanımlama 7 Bir Bilgi Nesnesi Kümesi	198
Tanımlama 8 ROSE protokolünü tamamlamak için bilgiyi kullanma 9	199
Parametreleme ihtiyacı 10 Olmayanlar henüz söylendi mi?	201
	203
	205
	208
 Bölüm 7 Sınıflar, kısıtlamalar ve parametrelendirme hakkında daha fazlası	209
1 Bilgi Nesnesi Sınıf Alanları 1.1 Tip	209
alanları 1.2 Sabit tip değil alanları 1.3	210
Değişken tip değil alanları 1.4 Sabit	211
tip değil er seti alanları 1.5 Değişken	212
tipi değil er seti alanları 1.6 Nesne	213
alanları 1.7 Nesne seti alanları 1.8	213
Geniş letilmiş alan adları	214
	214
	215
2 Bilgi Nesnesi tanımı için değil işken sözdizimi © OS, 31 Mayıs	216

3 Kısıtlamalar yeniden ziyaret edildi - kullanıcı tanımlı kısıtlama	220
4 Parametrelendirmeyle ilgili tüm hikaye 4.1 Neler	221
parametrelendirilebilir ve parametre olabilir?	222
4.2 Soyut səzdiziminin parametreleri 4.3	224
Gereksinimlerinizi açık hale getirme 4.3.1 TYPE-	225
TANIMLAYICI sınıfı 4.3.2 Bir örnek - X.400	225
üstbilgileri 4.3.3 Basit bir Dİ Zİ kullanımı	225
4.3.4 Geniş letilebilir bir Dİ Zİ kullanımı	226
4.3.5 bir bilgi nesnesi seti tanımı 4.3.6 "Baş lıklar"	227
nesne seti	227
	228
4.4 (Boş) geniş letilebilir bilgi nesnesi seti 5 "delikler" için	229
diğ er hükmü	230
5.1	230
HERHANGİ Bİ R 5.2	231
HERHANGİ TARAFINDAN	231
TANIMLANANLAR 5.3 HARİ Cİ	232
5.4 GÖMÜLÜ PDV 5.5 KARAKTER	233
Dİ Zİ Sİ 5.6 OKTET Dİ Zİ Sİ ve Bİ T Dİ Zİ Sİ	233
6 Bölm II'yi sonuçlandırmak için açıklamalar	234
BÖLÜM III KODLAMALAR	235
Bölm 1 Kodlama kurallarına giriş 1 Kodlama	236
kuralları nelerdir ve neden bölm alt başlığdır?	236
2 Kodlama kuralları yaklaşımıın avantajları nelerdir?	238
3 Kodlamaları tanımlama - TLV yaklaşımı 4	239
Geniş letilebilirlik veya "gelecekte prova"	240
5 İlk PER denemeleri - BER ile başlayın ve gereksiz sekizlileri kaldırın 6 PER'nin bazı	241
ilkeleri 6.1 BER deli gömleği ini kırmak 6.2 Bir "T"nin özdüğü ümidi er problemlerle	243
nasıl başa çıkarılır?	243
	244
6.3 SEQUENCE ve SET baş lıkları için hala T ve L'ye ihtiyacımız var mı?	245
6.4 Hizalanmış ve Hizalanmamış PER	246
7 Geniş letilebilirlik - buna sahip olmalısınız!	246
8 PER hakkında bilmeniz gereken başka ne var?	247
9 PER 10 Ayırt Edici ve Kanonik	248
Kodlama Kuralları Deneyimi 11 Sonuç	250
	251
Bölm 2 Temel Kodlama Kuralları 1 Giriş 2	252
Genel sorunlar 2.1 Bit sayıları ve diyagramlar	252
İçin notasyon 2.2 Tanımlayıcı sekizlileri 2.3	253
Uzunluk sekizlileri	253
	254
	256
2.3.1 Kısa biçim 2.3.2 Uzun	256
biçim 2.3.3 Belirsiz biçim	257
2.3.4 Uzunluk değil işkenlerinin	258
tartışılması 3 Ana türlerin V bölmünün	259
kodlamaları 3.1 NULL değilini kodlama 3.2 BOOLEAN	260
değeri kodlama	260
	261

3.3 INTEGER değerini kodlama 3.4	261
SAYILANDIRILMIŞ değerini kodlama 3.5 REAL	262
değerini kodlama	262
3.5.1 Temel 10 değerini kodlama	262
3.5.2 Temel 2 değerini kodlama	263
3.5.3 Özel gerçek değerini kodlama 3.6 Bir	265
OCTET STRING değerini kodlama 3.7 Bir BIT	266
STRING değerini kodlama 3.8 Etiketli tiplerin	266
kodlama değerleri 3.9 CHOICE tiplerinin kodlama	267
değerleri 3.10 Sequence Of Kodlama değerleri	268
3.11 Kodlama SET OF değerleri 3.12 Kodlama	268
SEQUENCE ve SET değerleri 3.13 SEÇENEK ve	269
VAR SAYILAN ögelerin sıra ve küme halinde işlenmesi	269
3.14 OBJECT TANIMLAYICI değerleri kodlama 3.15 Karakter dizisi değerleri kodlama 3.16	270
Zaman türlerinin kodlama değerleri 4 Daha karmaşık ik yapıları için kodlamalar 4.1 Açık	270
tipler 4.2 Görülü pdv tipi ve harici tip 4.3 Tip ÖRNEĞİ 4.4 KARAKTER DİZİSİ tipi 5 Sonuç	273
	275
	275
	275
	276
	276
	276
	277
 Bölüm 3 Paketlenmiş Kodlama Kuralları 1 Giriş	278
2 PER kodlamasının yapısı 2.1 Genel biçim	279
2.2 Kısıtlı hizalama ve PER varyantları	279
2.3 Kanonik kodlamalar 2.4 Dış düzey tam	279
kodlama	280
	281
	281
3 Geniş letilebilir tiplerin kodlama değerleri	282
4 PER-visible kısıtlamaları 4.1 Kavram 4.2	284
Değer işken parametrelerin etkisi 4.3	284
Değer işken uzunluklu kodlamalara sahip	285
karakter dizileri 4.4 Şimdi karmaşık iklas alım!	286
	286
5 INTEGER'leri kodlama - hazırlık tartışması 6 Etkili boyut	288
ve alfabe kısıtlamaları.	289
6.1 Sorunun ifadesi 6.2 Etkili boyut	289
kısıtlaması 6.3 Etkili alfabe kısıtlaması	290
7 Etiketlerin kanonik sırası 8 Sınırsız bir	290
sayısı kodlama 8.1 Üç uzunluk kodlama	291
biçimi 8.2 "Normalde küçük" değerleri	291
kodlama 8.3 Sınırsız sayıların kodlamaları	292
hakkında yorumlar 9 İSTEĞE BAĞLI bit-harita	295
ve SEÇİ M dizini.	296
	296
9.1 İSTEĞE BAĞLI bit haritası 9.2	296
SEÇİ M dizini	297
10 NULL ve BOOLEAN değerlerinin kodlanması.	297
11 INTEGER değerlerinin kodlanması.	297
11.1 Kısıtlanmamış tamsayı türleri 11.2	298
Kısıtlanmış tamsayı türleri	298

11.3 Kısıtlı tamsayı türleri 11.4	299
Tamsayı üzerindeki kısıtlama geniş letilebilir mi?	300
12 Kodlama SAYILANDIRILMIŞ değil erler.	301
13 Dizilerin kodlama uzunluk belirleyicileri vb. 14	302
Kodlama karakteri dizi değil erleri.	304
14.1 Karakter başı ina bit	304
sayısı 14.2 Dolgu bitleri	305
14.3 Geniş letilebilir karakter dizisi türleri	306
15 Kodlama SEQUENCE ve SET değil erleri.	306
15.1 Kodlama VARSAYILAN değil erleri	307
15.2 Kodlama uzantısı eklemeleri	307
16 Kodlama SEÇİ M değil erleri.	310
17 SEQUENCE OF ve SET OF değil erlerinin kodlanması.	311
18 GERÇEK ve OBJECT TANIMLAYICI değil erlerinin kodlanması.	312
19 Açık Tip Kodlama 20 Kalan	312
Tip Kodlama 21 Sonuç	313
	313
 Bölüm 4 ASN.1 ile ilgili diğer kodlama kuralları 1	315
İnsanlar neden yeni kodlama kuralları önerir?	315
2 LWER - Hafif Ağ ıraklı Kodlama Kuralları 2.1	316
LWER yaklaşımı 2.2 İlerlemenin yolu	317
kararlaştı 2.3 Sorunlar, sorunlar, sorunlar	317
2.4 LWER'in düğümü	317
	319
3 MBER - Minimum Bit Kodlama Kuralları	319
4 OER - Sekizli Kodlama Kuralları	320
5 XER - XML (Geniş letilmiş Biçimlendirme Dili) Kodlama Kuralları	321
6 BACnetER - BAC (Bina Otomasyon Komitesi) net Kodlama Kuralları	321
7 Kodlama Kontrol Spesifikasiyonları	322
 BÖLÜM IV TARİH VE UYGULAMALAR	323
 Bölüm 1 ASN.1'in gelişimi 1 İnsanlar 2	324
Daireler çiziyor musunuz?	325
	326
3 Standartları kim üretir?	327
4 Sayı oyunu 5 İlk yıllar	328
- X.409 ve tüm bunlar	329
5.1 Taslaqlar değil iş tirilir ve ASN.1 adı verilir 5.2 Notasyondan	329
BER'in ayrılması 5.3 Değil iş iklikler ne zaman teknik değil iş ikliklerdir?	330
	331
5.4 ASN.1'in neredeyse sona ermesi - İŞLEM ve HATA	331
6 Organizasyon ve yeniden organizasyon!	333
7 Araç satıcıları 8	334
Nesne tanımlayıcıları	334
8.1 Uzun veya kısa, insan veya bilgisayar dostu, soru bu 8.2 Nesne tanımlayıcı	334
ağacı nerede tanımlanmalıdır?	336
8.3 En üst düzey yayalar için savaş ve BAĞL OID'lerin tanıtılması	336
9 GERÇEK tip 10	337
Karakter dizi tipi - kısa tutmaya çalış alım!	338
10.1 Baş tan ASCII'ye 10.2 Karakter	338
setlerinin uluslararası kaydının ortaya çıkışı	338

10.3 ISO 8859'un geliş tirilmesi 340 10.4 ISO 10646 ve Unicode 340'ın ortaya çıkış 1 10.4.1 Dört boyutlu mimari 340
 10.4.2 Unicode 342'ye giriş 10.4.3 Nihai uzlaşma 343 10.5 Ve tüm bunların ASN.1 üzerindeki etkisi? 343 11
 HERHANGİ BİR makrolar ve Bilgi Nesneleri - bu kadar kısa tutmak zor (başlık bile iki satırda düşmüş)! 3452
 ASN.1(1990) tartışması 348 13 PER'nin ortaya çıkış 1 349 13.1 İLK deneme - PER-2 349 13.2 İKİNCİ deneme -
 PER-1 352 14 DER ve CER 353 15 Semantik modeller ve hepsi - ASN.1 1990'ların sonunda 354 16 Ne kaçtı? 355

Bölüm 2 ASN.1 Uygulamaları 1 Giriş 2	357
X.400'ün kökenleri 3 Açık Sistemler	357
Ara Bağlantısı (OSI) ve ISO'ya	358
geçiş 4 Protokol test topluluğu içinde kullanım 5 Entegre Hizmetler	359
Dijital Ağ 1 (ISDN) içinde kullanım	360
6 ITU-T ve multimedya standartlarında kullanım	361
7 Avrupa ve Amerika standartizasyon gruplarında kullanım 8	362
Bilgisayar kontrollü sistemleri yönetmek için kullanım 9 PKCS ve	363
PKIX ve SET ve diğer güvenlikle ilgili protokollerde kullanım 10 Diğer internet	364
spesifikasyonlarında kullanım 11 Büyük kurumsal kullanım iş letmeler ve ajanslar	365
12 Sonuç	366
 EKLER	367
1 Wineco protokol senaryosu	368
2 Wineco için tam protokol	369
Wineco protokolü için C desteği 3 için 3 derleyici çıktıları	372
4 Wineco protokolü için Java desteği 3 için derleyici çıktıları	374
Web üzerinden 5 ASN.1 kaynağı 1	384
 DİZİN	385

Bu sayfa, genel sayfa düzeni için kasıtlı olarak boş bırakılmıştır.

Önsöz

Bu metin, öncelikle protokol spesifikasyonuna veya ASN.1 tabanlı protokollerin uygulanmasına dahil olanlar için yazılmıştır. Bununla birlikte, yöneticiler, öğretmenler ve sadece entelektüel olarak meraklı olanlar dahil olmak üzere daha geniş bir kitlenin ilgisini çekmesi ve kullanması beklenmektedir.

Aşağıdaki Giriş bölümünden azından tüm okuyucular tarafından taranmalıdır ve metnin yapısının tartışılmasıyla sona erer. Bundan sonra, okuyucular genellikle böümleri ve böümleri istedikleri sırayla alma ve bazılarını (veya birçoğu unu) çıkışma konusunda makul bir özgürlük e sahiptir, ancak ASN.1 hakkında çok az bilgisi olanlar için tamamını okumak mantıklı olacaktır. İlk önce Bölüm I'in sunduğu sırayla.

İşte farklı okuyucu türlerinin neleri ele almak isteyebileceğine dair kabaca bir kılavuz:

Yöneticiler: Protokol belirtimi için bir gösterim olarak ASN.1'in olası kullanımına ilişkin kararları almaktan sorumlu olanlar veya ASN.1 kullanılarak tanımlanan protokollerin uygulayan ekipleri yönetmekten sorumlu olanlar, Bölüm I'ı ("ASN.1 Genel Bakış") okumayı ve ihtiyaç duymalıdır. Bölüm IV ("Tarih ve Uygulamalar") da ilgi çekici olabilir, ancak daha fazla okumaya gerek yok.

Bu, ASN.1'i merak edenler ve ASN.1'e kısa ve oldukça okunabilir bir giriş isteyenler için de geçerli olacaktır.

Protokol belirleyicileri: Protokollerin tasarlayan ve belirleyenler için, ASN.1'in bir belirtim olarak kullanılabilir kullanılmayacağıını belirlemek için Bölüm I'ın ("ASN.1'e Genel Bakış") ve Bölüm IV'ün ("Geçmiş ve Uygulamalar") büyük bir kısmı taranmalıdır. ancak Bölüm II ("Diğer ayrıntılar") bu grup için çok önemlidir.

Bir ASN.1 aracı kullanan uygulayıcılar: Bu grup için, Bölüm I ("Kısaca ASN.1") ve Bölüm II ("Diğer Ayrıntılar") yeterli olacaktır.

Elle kodlama yapan uygulayıcılar: (veya ASN.1 araçlarını geliş tiriyor olabilecek kişilere), yukarıdaki böümleri Bölüm III'ü ("Kodlamalar") ve gerçekte de ASN için gerçek ITU-T Tavsiyelerini/ISO Standartlarını dikkatli bir şekilde okuyarak tamamlamalıdır. 1.

Protokol belirtimi tekniklerini kapsayan derslerdeki öğretmenler: Öğrencilerine protokol belirtimine (ve belki de ASN.1'in kendisine) yönelik soyut sözdizimi yaklaşımı anlayışını vermemeyi amaçlayan lisans ve lisansüstü dersler, Bölüm I'ın ("ASN.1 Genel Bakış") ilk kısımlarını yerleşirmelidir. ") ve Bölüm IV'ün bir kısmı ("Tarih ve Uygulamalar") dersin okuma listesinde yer almaktadır.

Entelektüel meraklı: Belki bu grup tüm metni baştan sona okuyacak ve ilginç ve ilham verici bulacaktır! Metni hafif ve okunabilir tutmak için mümkün olan her yerde giriş imlerde bulunuldu - devam edin!

Ek 5'te verilen URL'de bu metnin elektronik bir versiyonu ve ASN.1 ile ilgili diğer kaynakların bir listesi bulunmaktadır. Ve daha da önemlisi, bu sitede indirilmek üzere yazım hatası sayfaları sağlanacaktır.

Örneklerin tümü, (1986'dan beri) ASN kullanılarak tanımlanan protokollerin uygulanmasına yardımcı olacak araçlar gelişti ve pazarlayan bir ABD şirketi olan Open Systems Solutions (OSS) tarafından üretilen ve pazarlanan "OSS ASN.1 Araçları" paketi kullanılarak doğrulanmıştır. .1. © OS için OSS'ye minnettaram, 31 Mayıs 1999

Bu kitabın hazırlanmasında ve bu amaca yönelik araçlarının sağlanmasıında büyük destek.
OSS pek çok şekilde destek ve cesaret vermiş ve ilk taslaklar hakkında çok değil yorumlar yapan çok
sayıda metin eleş tirmeni sunmuş olsa da, bu metinde ifade edilen görüş ler yalnızca yazara aittir.

John Larmouth (j.larmouth@iti.salford.ac.uk)

Mayıs 1999

giriş

Özet:

Bu giriş :

sorunlu ASN.1 adreslerini açıklar,

kısaca ASN.1'in ne olduğunu söyler,

neden yararlı olduğunu açıklar.

1 Küresel iletişim im altyapısı

Her zamankinden daha geniş bir faaliyet yelpazesini gerçekleştirmek için bilgisayar sistemlerinin iş birliği içinde hızlı bir ilerleme dönemimizdeyiz. İnsan gündemi uzaktan oturum açmayı, e-postayı, dosya aktarımını ve son olarak World-Wide Web'i (WWW) destekleyen geleneksel bilgisayar iletişim imleri, bilgisayar sistemleri arasında ve yerleşik bilgisayarlı cihazlar arasında giderek daha karmaşık bilgi alış verişini gerektiren yeni uygulamalarla destekleniyor. cips.

Müzayedelerde teklif verme, para cüzdanı transferleri, elektronik iş imler, oylama desteği veya etkileşimi video gibi bu bilgi alış verişlerinden bazıları insan kaynaklı olmaya devam ediyor.

Diğerleri, cep telefonları (ve diğer telefon uygulamaları), sayaç okuma, kirlilik kaydı, hava trafik kontrolü, güç dağıtımının kontrolü ve evdeki uygulamalar gibi çeşitli faaliyetleri desteklemek üzere otomatik ve otonom bilgisayardan bilgisayara iletişim im için tasarlanmış tır cihazlarının kontrolü.

Her durumda, bilgisayarların gerçekleştireceği iş tokuslarının ayrıntılı spesifikasyonu ve bu iş tokusları destekleyecek yazılımın uygulanması için bir gereklilik vardır.

Bugün bu iş tokuslarının çoğunluğu için en temel destek, TCP/IP ve İnternet kullanımıyla sağlanmaktadır, ancak diğer taş iş makinaları, özellikle telekomünikasyon alanında hala kullanılmaktadır.

Bununla birlikte, TCP (veya diğer taş iş makinaları) kullanılarak iletişim imlerinin veri formatlarının belirtilmesi, uygulama protokollerinin tasarımını ve açık bir şekilde belirtmesini ve ardından bu protokollerin uygulanmasını (veya bunu paralel olarak) gerektirir.

Uygulamalar ve farklı sağlayıcılar tarafından üretilen cihazlar arasında iletişim imlerinin mümkün olabilmesi için bu uygulama protokollerine ilişkin standartlara ihtiyaç vardır. Standartlar, Uluslararası Telekomünikasyon Birliği (ITU-T) ve Uluslararası Standartlar Sekktörü (ISO) veya İnternet Mühendisliği (IETF) gibi tanınmış uluslararası kuruluşları veya endüstriyel birlilikler veya iş birliği grupları tarafından üretilenlerdir ve uluslararası Sivil Havacılık Örgütü (ICAO), Açık Yönetim Grubu (OMG) veya Güvenli Elektronik İş imler (SET) konsorsiyumu gibi konsorsiyumlar veya Reuters veya IBM gibi uluslararası kuruluşları tarafından.

Bu farklı grupların iletişim im standartlarını belirleme görevine yönelik çeşitli iletişim imleri vardır, ancak çoğu durumda ASN.1 veya ağırlıkları sağlayarak kilit bir rol oynar:

Bir standardizasyon kuruluşu tarafından bilgisayar değil iş imlerinin hızlı ve kesin olarak belirtilmesi.

Uygulamayı desteklemek için ürün üretenler tarafından ortaya çıkan standardın kolay ve hatasız uygulanması.

Bazı endüstriyel sektörlerde, özellikle telekomünikasyon sektöründe, güvenlikle ilgili alış verişlerde ve multimedya alış verişlerinde, ASN.1, uygulama protokollerini belirlemenin baskın yoludur. (Diğer tek önemli rakip, IETF tarafından sıkılıkla kullanılan, ancak karmaşık yapılar için daha az uygun olan ve genellikle çok daha az kompakt bir kodlama seti üreten karakter tabanlı yaklaşımındır). Belirtim dili olarak ASN.1'in kullanıldığı bazı uygulamaların açıklaması Bölüm IV'te verilmektedir.

2 ASN.1 tam olarak nedir?

"TCP/IP" terimi, iki protokol özelliğini (İletim Kontrol Protokolü - TCP ve İnternet Protokolü - IP) tanımlamak için veya daha geniş anlamda, TCP/IP'ye dayalı protokollerin ve destekleyici yazılımların tümünü tanımlamak için kullanılabilir. Benzer şekilde, "ASN.1" terimi, "Soyut Sözdizimi Gösterimi Bir" adı verilen bir gösterimi veya dili tanımlamak için dar anlamda kullanılabilir veya daha geniş anlamda gösterimi, ilişkili kodlama kurallarını ve yardımcı olan yazılım araçlarını tanımlamak için kullanılabilir. kullanımı.

ASN.1'i önemli ve benzersiz yapan şeyleler şunlardır:

Veri yapılarını yüksek bir soyutlama seviyesinde belirtmek için uluslararası standartlaşmış, satıcıdan bağımsız, platformdan ve dilden bağımsız bir notasyondur.
(Notasyon, Bölüm I ve II'de açıklanmıştır).

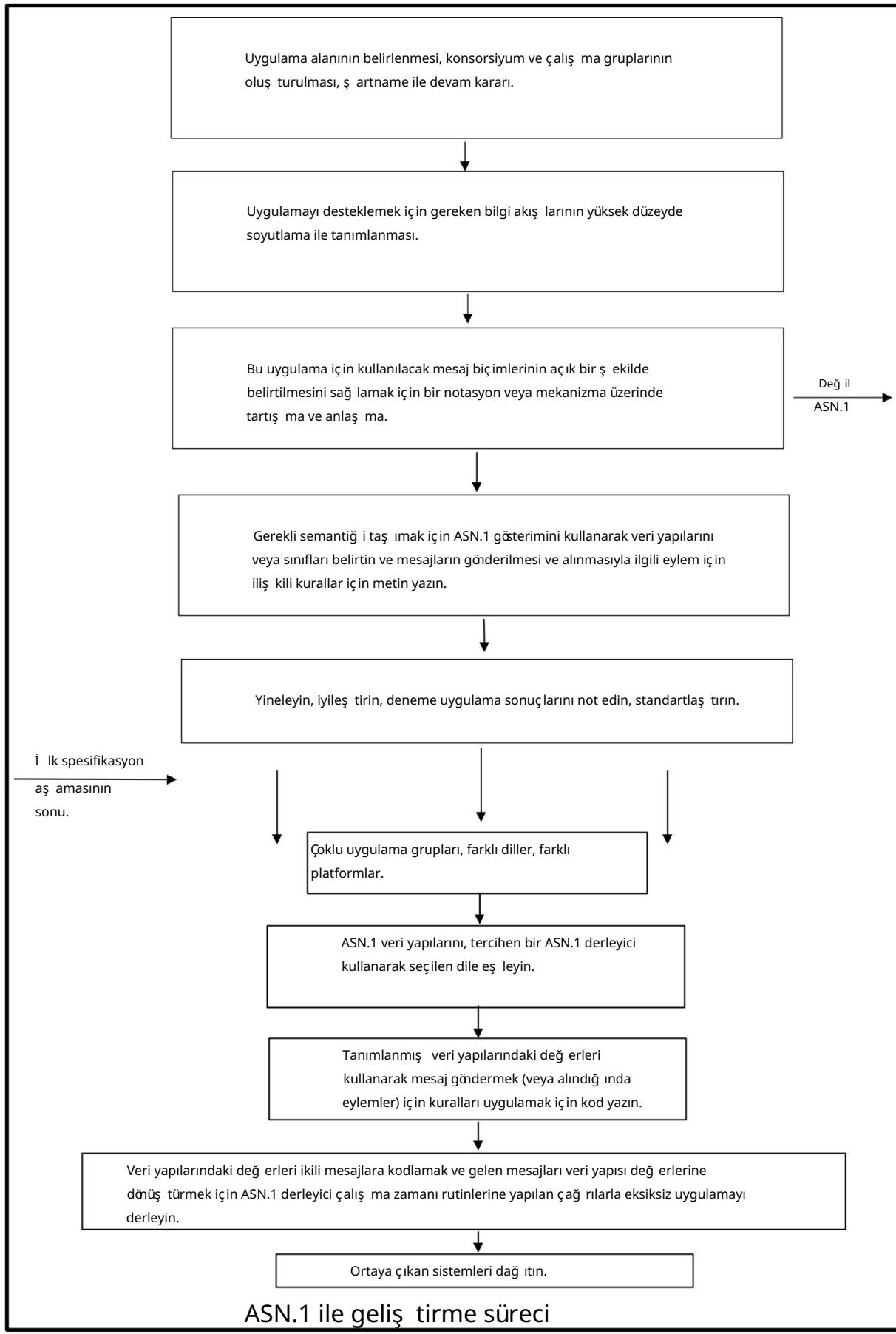
Gereksiz yere ayrıntılı olmayan kodlamalar kullanılarak bir bilgisayar ağı üzerinde aktarılmalı gerektiğiinde bu veri yapılarının değil erlerini temsil edecek kesin bit modellerini (yine platformdan bağımsız ve dilden bağımsız) belirleyen kurallar tarafından desteklenir. (Kodlama kuralları Bölüm III'te açıklanmıştır).

ASN.1 notasyonunu tercih edilen bir bilgisayar programlama dilindeki veri yapısı tanımlarına eşittelenen ve bu veri yapılarının bellekteki değil erleri ile bellekteki veri yapılarının değil erleri arasında otomatik dönüşütmeyi destekleyen birçok platform ve çeşitli programlama dilleri için mevcut araçlar tarafından desteklenir. bir iletişim hattı üzerinden transfer için tanımlı bit modelleri. (Araçlar, Bölüm I, Bölüm 6'da açıklanmaktadır).

ASN.1'in önemli olan ve bu metinde daha sonra tartışılan bir dizi başlığı ince özelliğine vardır. Bunlardan bazıları:

Uzun yıllar arayla tasarlanan ve devreye alınan konuşlandırmalı "sürüm 1" sistemleri ile "sürüm 2" sistemleri arasındaki birlikte çalışma sorununu ele alır ve bu sistemler için destek sağlar. (Buna "geniş letilebilirlik" denir).

Belirli spesifikasyonları (belki çok farklı şeillerde) gelişiren diğer standart gruplarıyla birlikte, bir standart grubu tarafından kısmi veya jenerik spesifikasyona olanak sağlayan mekanizmalar sağlar.



Uzun dizileri, büyük tamsayı değer erlerini, büyük yinelemeli yapıları ve daha az kapasiteye sahip olabilecek küçük sistemleri işleyebilen büyük sistemler arasındaki birlikte çalışma sorunlarının potansiyelini tanır.

Tamsayıların boyutu, adlandırma yapıları ve karakter dizisi türleri gibi genellikle normal programlama dillerinden çok daha zengin olan bir dizi veri yapısı sağlar. Bu, aktarılması gereken değer aralığıının belirlenmesinde kesinliği ve dolayısıyla daha uygun kodlamaların üretilmesini sağlar.

3 ASN.1 ile geliş tirme süreci

Akış şeması, başlangıçtan ilk sistemlerin konuşlandırılmasına kadar olan geliş tirme sürecini göstermektedir.

(Fakat bu sürecin, hem "doğru yapmak" için standardizasyon grubu tarafından yapılan erken revizyonlarla hem de birkaç yıl sonra bir "sürüm 2" standartı üretildiğiinde daha önemli revizyonlarla sık sık tekrarlanan bir süreç olduğu unutulmamalıdır.)

Diyagramdan dikkat edilmesi gereken bazı önemli noktalar:

Bir standartı tanımlama gösterimi olarak ASN.1'i kullanma kararı kilit bir karardır. ASN.1 gösteriminin standardizasyon grubu tarafından iyi anlaşılması gerektir, ancak net bir belirtim için zengin bir dizi kolaylık sağlar. Protokol spesifikasyonunun alternatif yolları, Bölüm I, Bölüm 1'de tartışılmaktadır.

Standardizasyon grubunun (veya uygulayıcılarının) istenen semantik iletişim için kullanılacak ayrıntılı bit modelleriyle ilgilenmesine gerek yoktur: kodlamanın ayrıntıları ASN.1 kodlama kuralları belirtimlerinde ve çalışma türmada "gizlidir". ASN.1 araçları tarafından sağlanan zaman desteği.

Uygulama görevi basittir: Yazılması (ve hatalarının ayıklanması ve test edilmesi) gereken tek kod, uygulama için gerekli anlamsal eylemleri gerçekleştirmek için kullanılan koddur. Karmaşık ikinci anlaşmazlık veya kodlama kodu yazmaya ve hata ayıklamaya gerek yoktur.

4 Metnin yapısı.

Bölüm I, ASN.1 gösteriminin en sık karşılaşılan özelliklerini kapsar. Ayrıca, Bölüm II'de tüm kapsamıyla birlikte notasyonun diğer tüm yönlerini kısaca tanıtmaktadır. ASN.1'i kullanarak spesifikasyonları yazmaktan veya uygulamaları kodlamaktan birincil derecede sorumlu olmayan, ancak (standartların veya uygulamaların) gelişmesine yardımcı olmak veya bunları yönetmek için temel bir anlayış ve ihtiyaç duyanların, ihtiyaç duydukları herşeyi Bölüm'den almaları amaçlanmaktadır. I. Birincil yazma veya kodlama sorumluluğu olanlar Bölüm II'ye de ihtiyaç duyacaktır.

Bölüm III, ASN.1 kodlama kurallarının arkasındaki ilkeleri ve ayrıntılarının çoğuunu açıklamaktadır. Ancak, bu metin gerçekten sadece meraklılar için! Standart yazarlarının veya kodlayıcılarının bu kodlamaları bilmesine gerek yoktur (uygulama için bir araç kullanılması şartıyla).

Bdüm IV, metni (çeşitli destekleyici eklerin dışında), ASN.1'in geçmiş ve onu kullanan belirtilen uygulamalar hakkında bazı ayrıntılar vererek tamamlamaktadır.

ASN.1'in ayrıntılı bir incelemesi oldukça "ağır" bir konudur, ancak mümkün olduğunca biraz hafiflik ve mizah katmaya çalıştım. Dilediğimi atla, ilgini çekeni oku ama lütfen, tadını çıkar!

Bu sayfa, genel sayfa düzeni için kasıtlı olarak boş bırakılmıştır.

BÖLÜM I

ASN.1 Genel Bakış

Bölüm 1 protokollerin özellikleri

(Veya: Basitçe söylemenmesi gerekeni söylemek!)

Özet:

Bu bölüm:

"protokol" kavramını ve özelliklerini tanıtır,

kavramlarına erken bir giriş sağlar.

– katmanlama,

– geniş letilebilirlik,

– özet ve aktarım söz dizimleri,

protokol spesifikasyonu araçlarını tartışır,

Belirtim mekanizmalarının ve notasyonların tasarımda ortaya çıkan yaygın sorunları açıklar.

(Protokol spesifikasyonuna dahil olan okuyucular, bu Bölümdeki ilk "kavramlar" materyalinin çoğu una aşına olmalıdır, ancak yapmaya çalışıkları bazı şeyleler hakkında yeni ve belki de aydınlatıcı bir bakış açısından sağladığını görebilirler.)

1 Protokol nedir?

Bir bilgisayar protokolü şun şeklinde tanımlanabilir:

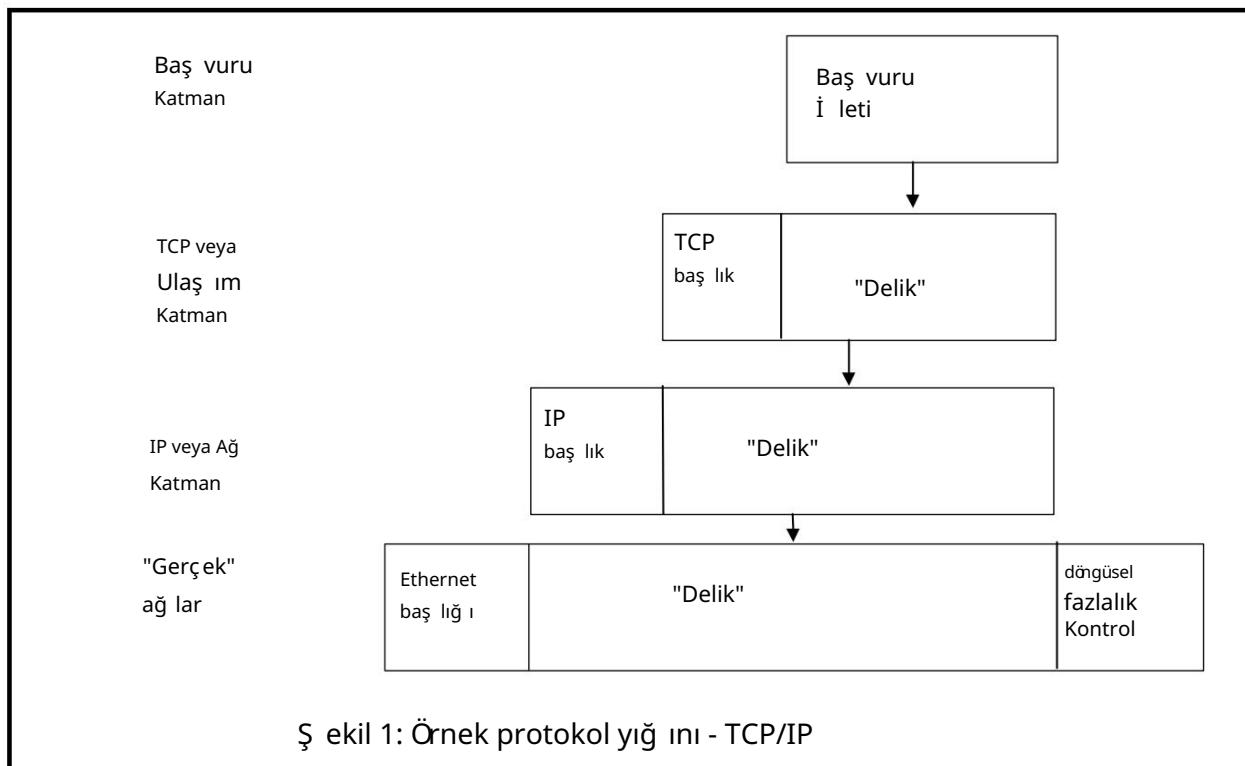
Belirli bir mesajın ne zaman gönderilebileceğiini yöneten kurallarla birlikte, her biri tanımlanmış bir anlam (anlambilim) taşıyan, iyi tanımlanmış bir dizi mesaj (bit kalıpları veya - günümüzde giderek artan şekilde - sekizli diziler).

Ancak, bir protokol nadiren tek başına durur. Bunun yerine, genellikle, bir gönderici tarafından yayınlanan mesajın tamamını belirlemek için birkaç ayrı spesifikasyonun birlikte çalıştığı, bu mesajın bazı böümlerinin ara (anahtarlama) düğümler tarafından eyleme yönlendirildiği ve bazı böümlerinin amaçlandığı bir "protokol yığını"ının parçasıdır. Uzak uç sistem için.

Bu "katmanlı" protokol teknikinde:

Bir belirtim, ortasında bir "delik" bulunan mesajın dış kısmının biçimini ve anlamını belirler. Bu "deliğ e" yerleş tirilen herhangi bir malzemeyi taş ımak için bir "taş iyıcı hizmeti" (veya sadece "hizmeti") sağ lar.

İ kinci bir spesifikasyon, "deliğ in" iç eriğ ini tanımlar, belki de baş ka bir spesifikasyon katmanı iç in baş ka bir delik bırakır ve bu böyle devam eder.



Şekil 1, sağ ladiqları "delikte" taş inan IP protokolü ile gerçek ağların temel taş iyıcı mekanizmasını sağ ladiğ i ve IP'nin TCP (veya daha az bilinen Kullanıcı Datagramı) için bir taş iyıcı görevi gördüğ ü bir TCP/IP yığıını göstermektedir. Protokol - UDP), baş ka bir protokol katmanı oluş turur ve (tipik olarak TCP/IP iç in) yekpare bir uygulama katmanıyla - son "deliğ i" tamamlayan tek bir belirtim

Bir alt katman tarafından sağlanan "hizmet" in (kayıplı, güvenli, güvenilir) ve bu hizmeti kontrol eden herhangi bir parametrenin kesin doğ asının, bir sonraki katmanın bu hizmeti uygun ş ekilde kullanabilmesi iç in bilinmesi gerekir.

Bu bireysel özellik katmanlarının her birini genellikle "bir protokol" olarak adlandırırız ve bu nedenle tanımımızı geliş tirebiliriz:

protokol nedir?

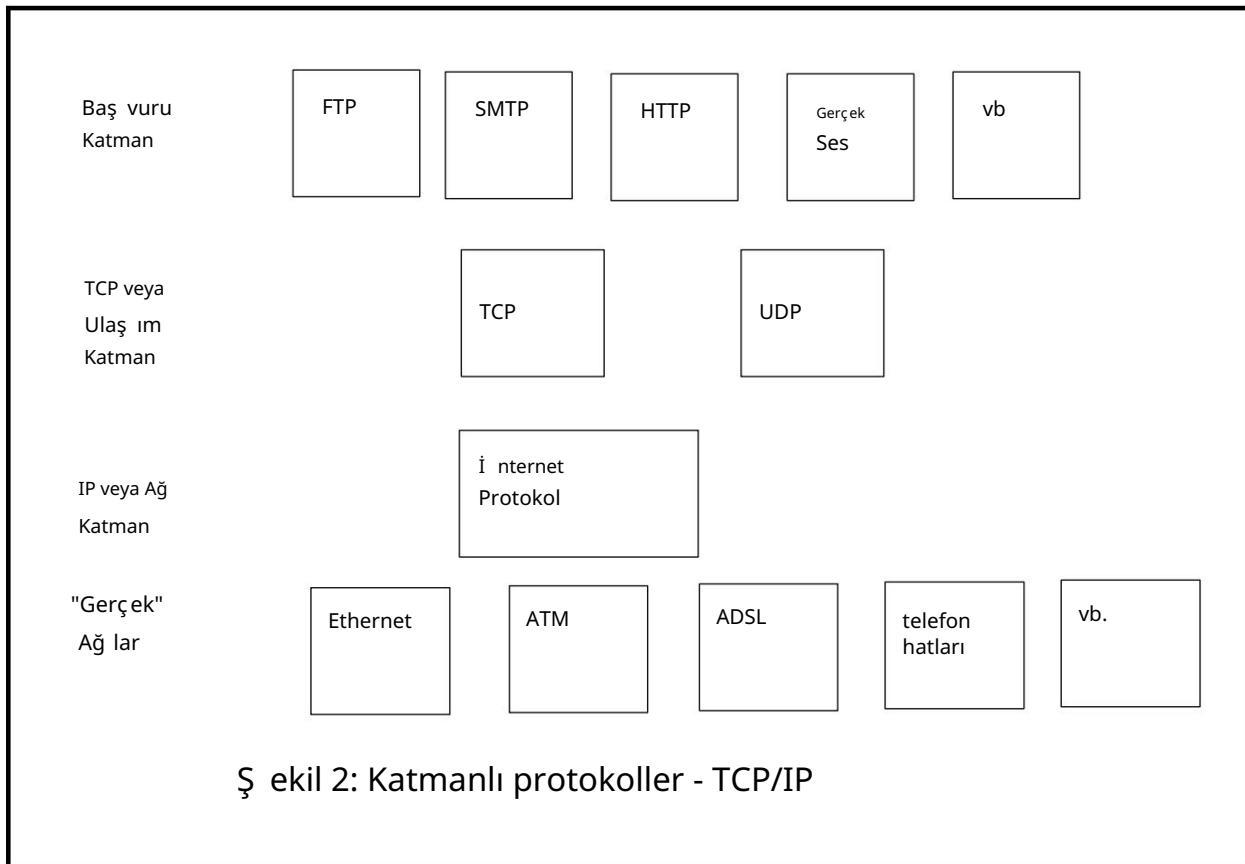
Her biri belirli bir anlam taş ıyan, iyi tanımlanmış bir dizi mesaj ve

Belirli bir mesajın ne zaman gönderilebileceğ ini yöneten kurallar ve mesajları

aktarmak iç in kullanılan hizmetin doğ ası hakkında, ya tek bir son uygulamayı destekleyen ya da daha zengin bir taş iyıcı hizmeti sağ layan Açık Varsayımlar.

Şekil 1'de, IP taşıyıcı tarafından sağlanan "deliğin" bir TCP mesajı veya bir UDP mesajı içerebileceğiini unutmayın - farklı özelliklere sahip (ve kendileri başka bir taşıyıcı hizmeti sağlayan) çok farklı iki protokol. Bu nedenle, "katmanlanmanın" avantajlarından biri, taşıyıcı hizmetinin, alt katman protokollerini geliş tirildiğiinde belki de çok ucu hiç düş ünülümemiş olan çok çeşitli yüksek seviyeli protokollerini desteklemek için yeniden kullanılabilir olmasıdır.

Birden çok farklı protokol aşağında katmanda bir boş luk kaplayabildiğiinde (veya yukarıdaki katman için taşıyıcı hizmetler sağlayabildiğiinde), bu genellikle Şekil 2'de gösterilen katmanlama diyagramı ile gösterilir.



2 Protokol belirtimi - bazı temel kavramlar

Protokoller birçok şekilde belirtilbilir (ve tarihsel olarak belirtilmişdir). Temel bir ayrim, karakter tabanlı belirtim ile ikili tabanlı belirtim arasındadır.

Karakter tabanlı belirtim

"Protokol", ASCII kodlu metnin bir dizi satırı olarak tanımlanır.

İkili tabanlı belirtim

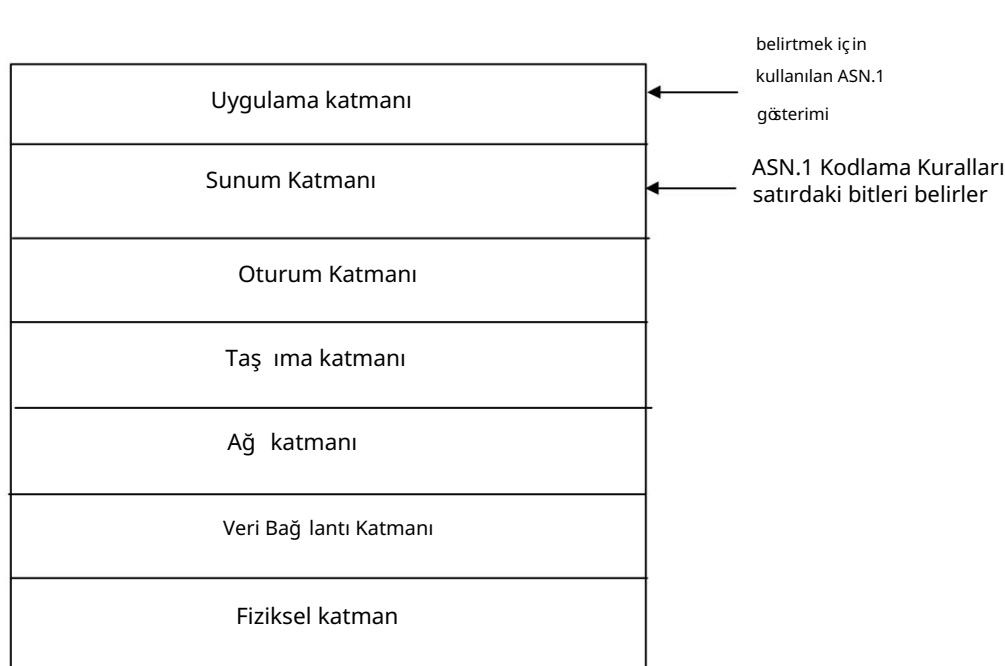
"Protokol", bir sekizli veya bit dizisi olarak tanımlanır.

İkili tabanlı belirtim için yaklaşımalar, çeşitli resim tabanlı yöntemlerden ilişkili uygulamadan bağımsız kodlama kuralları ile ayrı olarak tanımlanmış bir notasyonun kullanımına kadar değişir.

İkincisi, "soyut sözdizimi" yaklaşımı olarak adlandırılır. ASN.1 ile alınan yaklaşım budur. Tasarımcıların kodlama sorunlarıyla gereğinden fazla ilgilenmeden spesifikasyonlar üretmesine olanak sağlama ve ayrıca bu şekilde belirtilen protokollerin kolay uygulanmasını desteklemek için uygulamadan bağımsız araçların sağlanması izin vermesi avantajına sahiptir. Ayrıca, uygulamaya özel uygulama kodu, kodlama kodundan bağımsız olduğu için, geliş tirildikçe iyileş tirilmiş kodlamalara geçiş kolaylaşır.

2.1 Katmanlama ve protokol "delikleri"

Katmanlama konsepti, Şekil 3'te gösterilen Açık Sistemler Ara Bağlantısı (OSI) için belki de en yaygın olarak Uluslararası Standartlar Örgütü (ISO) ve Uluslararası Telekomünikasyon Birliği (ITU) "mimarisi" veya "7 katmanlı model" ile ilişkilendirilir.



Şekil 3: OSI katmanları ve ASN.1

Bu çerçevede gelişen protokollerin çoğu günümüzde pek kullanılmasa da, protokol spesifikasyonuna yaklaşımalar için ilginç bir akademik çalışma olmaya devam ediyor. 1970'lerin sonundaki orijinal OSI konseptinde, her spesifikasyonun "delik" olmaksızın tek bir son uygulamayı desteklediği son bir "uygulama katmanı" ile (giderek daha zengin) taşıyıcı hizmetleri sağlayan yalnızca 6 katman olacaktı.

Bununla birlikte, sonraki on yıl içinde, insanların "uygulama katmanında" bile, daha sonraki uzantılar için spesifikasyonlarında "delikler" bırakmak veya protokollerini belirli ihtiyaçlara göre uyarlamak için bir araç sağlama istedikleri ortaya çıktı. Örneğin, daha yeni ve önemli protokollerden biri - Güvenli Elektronik İşlemeler (SET) - tam olarak tanımlanmış çok sayıda mesaj semantikini içerir, ancak aynı zamanda "tüccar ayrıntılarını" aktarabilen bir dizi "delik" sağlar.

SET spesifikasyonunun kendisinde belirtilmiş tir. Dolayısıyla, satın alma istekleri ve yanıtları, sorgulama istekleri ve yanıtları, yetkilendirme istekleri ve yanıtları vb. için temel mesajlarımız var, ancak bu mesajların içinde "mesaj uzantıları" için "delikler" var - belirli bir satıcıya özgü ek bilgiler.

Bu nedenle, bir protokolü belirtmek için herhangi bir mekanizma veya gösterimin "deliklerin" dahil edilmesini iyi bir şekilde karşı lâyabilmesi önemlidir. Bu, son on yılda ASN.1'deki en önemli geliş melerden biri olmuş tur ve bu kitapta çok daha ayrıntılı bir tartışma konusu olacaktır.

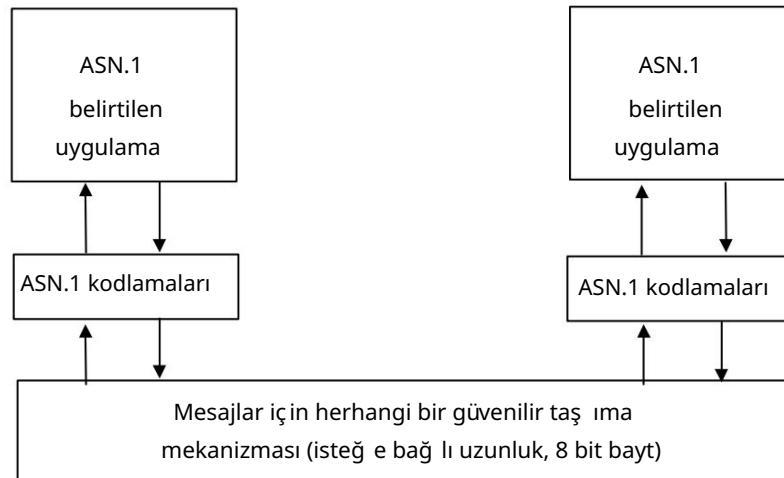
Delik

Baş kaları tarafından tanımlanan malzemeyi taşı mak için tanimsız bırakılan bir spesifikasyonun parçası.

"Deliklerin" dahil edilmesi için "yemek kuyusu", iletişim zamanında bir deliğin içeriğini tanımlamak için gösterimin tanımlı mekanizmalara sahip olması gerektiği anlamına gelir (tercihen bu notasyon kullanılarak yazılan tüm spesifikasyonlara tek tip olarak uygulanır). (Alt katmanlarda buna bazen "protokol kimliği" sorunu denir). Bununla birlikte, aynı derecede önemli olan, bir tanımlamanın eksik olduğuunu (bir delik içerdigini) açıkcaya tanımlamaya yönelik notasyon araçları ve deliğin içeriğinin (belki daha sonra) tanımlamasını deliklerin konumıyla ilişkilendirmek için iyi tanımlanmış mekanizmalardır..

2.2 Katmanlamanın ilk gelişmeleri

En eski protokoller tek bir bağ lantı üzerinden çalışıyordu (şartlı bir şekilde "LINK" protokollerini olarak adlandırılıyordu!) ve uygulamayla ilgili belirli olayları işarete etmek için farklı fiziksel sinyallerin (genellikle voltaj veya akım) kullanıldığı tek bir yekpare belirtimde belirtilmiş tir. (Bir örnek, eski telefon sistemlerindeki "açık" sinyalidir). Farklı bir uygulama çalışma tırmak istiyorsanız, elektronik inizi yeniden tanımladınız ve yeniden oluşturunuz!



Şekil 4: ASN.1 ile uygulama iletişimimi

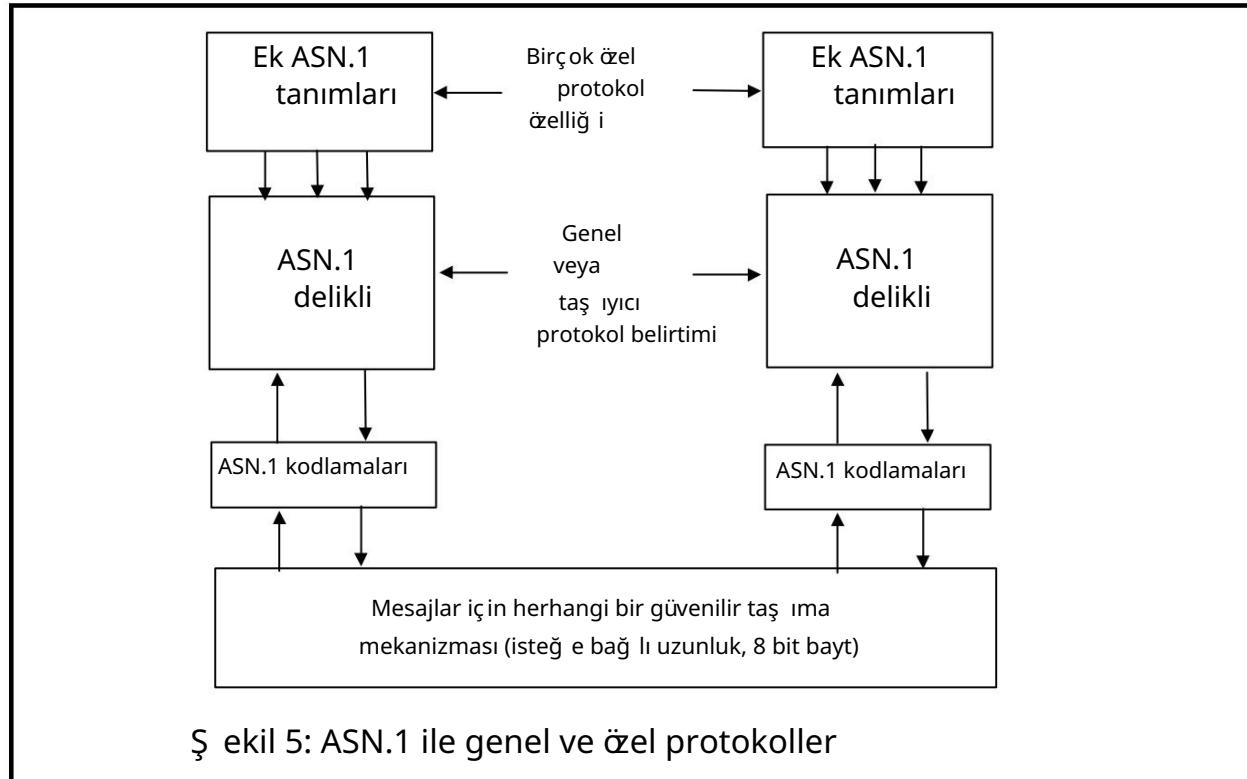
Bu, "katmanlamanın" en büyük avantajını göstermektedir: Şekil 2'de gösterildiği gibi, bir dizi farklı üst düzey protokol veya uygulamayı desteklemek için taşınıcı mekanizmaların yeniden kullanılabilirliğini sağlar.

Bugün hiç kimse eski "LINK" protokollerine benzer tek bir yekpare özellik sağlama hayal bile edemezdi: belki de bilgisayar iletişim teknolojisindeki en önemli adım, akım, voltaj, ses, iş, sinyalizasyon sistemlerinin bir aktarmaktan başlayarak bir şeyle yapmayacağıını kabul etmekti. İki öğeli alfabe - sıfır veya bir - ve bu uygulamalar bunun üzerine inşaat edilecek. Diğer bir önemli adım, bu sürekli bit akışıını sınırlamış veya "çerçevelenmiş" mesajlara dönüşüm türmek için başlayarak protokol "katmanı" sağlama ve daha yüksek katman protokollerinin "bir mesaj göndermek" hakkında konuşmayı sağlama (kaybolabilir, kaybolabilir, aracılıkıyla, ancak tartışma birimi mesajdır).

Ancak bu, ASN.1 üzerinde bir kitap için çok düşük bir tartışma düzeyidir! Bu elektrik seviyeleri ile ASN.1'in birlikte çalıştığı normal taşınıcılar arasında, İnternet veya bir telekom ağı aracılığıyla adresleme ve yönlendirme ile ilgili ve kayıp mesajların kurtarılması ile ilgili protokol katmanlarımız vardır.

ASN.1 düzeyinde, bir makinedeki bir uygulamanın, kendi aralarında güvenilir bir şekilde sekizli diziler göndererek başlayarak makinedeki bir uygulamaya "konuşabileceğini" varsayıyoruz. (ASN.1 tanımlı tüm mesajların 8 bitin tam katı olduğuunu unutmamak - bir sekizli dizi, genel bir bit dizisi değil). Bu, Şekil 4'te gösterilmektedir.

Bununla birlikte, birçok ASN.1 tanımlı uygulama, boş lukları doldurmak için ek özelliklerle (belki farklı gruplar tarafından farklı şekilde sağlanır) ilk önce temel bir "taşınıcı" hizmeti belirtilebilir. Bu, Şekil 5'te gösterilmektedir. Daha sonra göreceğimiz gibi, ASN.1'de "deliklerin" veya "katmanların" kullanımını destekleyen birçok mekanizma vardır.



Şekil 5: ASN.1 ile genel ve özel protokoller

İnsanlar bazen OSI 7 katmanlı modelini "katmanlama çığına döndü" olarak tanımladılar. Katmanlama, spesifikasyonların (ve kodun) yeniden kullanılabilirliğini teşvik etmede ve toplam spesifikasyonun parçalarının (düşük veya yüksek bir katman) diğer kısımları etkilemeden daha sonra iyileş tirilmesi, genişletilmesi (veya sadece onarılması!) için önemli bir araç olabilir. Toplam spesifikasyonun farklı kısımlarını birbirine bağlama araçları yeterince zenginse elde edilecektir.

2.3 Katmanlamanın dezavantajları - basit tutun!

Katmanlama, yeniden kullanılabilirlikte açıkça önemli avantajlar sağlar, ancak aynı zamanda, belirli bir uygulamayı tamamen uygulamak için birçok farklı belgeye başvurmak gerekebilir ve bunları birbirine bağlamak için "yapıştırıcı" her zaman yeterince kesin olmayabilir. Farklı satıcıların uygulamalarının birlikte çalışmasını sağlayın.

Bu nedenle, protokollerini tasarlarken, genellikle uzun ömrü arzusunun, tüm spesifikasyonu basit tutmaya yönelik eşit bir arzu ile yumuşatılması önemlidir. Bu, daha sonra tekrar döneceğimiz bir konudur - ASN.1, çok basit ve net özellikleri çok kolay ve hızlı bir şekilde yazmayı mümkün kılar. Ancak katmanlamayı ve "genişletilebilirliği" desteklemek için güçlü özellikleri içerir (aşağıya bakın). Bu tür özellikleri kullanma veya kullanmama kararı tasarımcıya aittir. İyi bir uzun ömrülü spesifikasyon için kullanımlarının gerekli olduğu durumlar vardır. Ek karmaşıklık (ve bazen uygulama boyutunun) gelişmiş özelliklerin kullanımını haklı çıkarmadığı başka durumlar da vardır.

2.4 Genişletilebilirlik

Katmanlamanın, üstteki ve alttaki katmanların özelliklerini etkilemeden katmanlardan birinin "daha sonra iyileş tirilmesini" sağladığını dair daha önce bir açıklama yapıldı. Bu "sonradan iyileş tirme" kavramı anahtar bir ifadedir ve herhangi bir katmanlama tartışmasının ötesinde bir öne sahiptir. 1980'lerde farklı edilen protokol spesifikasyonunun önemli yönlerinden biri, bir protokol spesifikasyonunun xyz tarihinde nadiren (muhtemelen asla!) tamamlanmaması, uygulanması, dağıtılmaması ve değişmeden bırakılmasıdır.

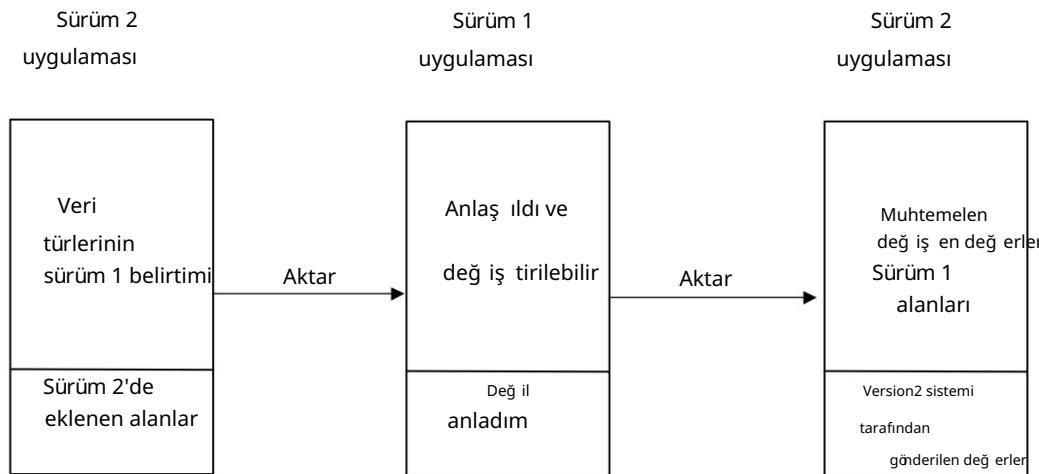
Genişletilebilirlik hükmü

1 sürümünün bir parçası
gelecekteki sürüm 2 (genişletilmiş)
sistemlerin dağıtılan sürüm 1 sistemleriyle
birlikte çalışmasını kolaylaşdırma için
tasarlanmış spesifikasyon

Her zaman bir "sürüm 2" vardır. Ve sürüm 2 uygulamalarının, tercihen sürüm 2 sistemlerine hem sürüm 1 hem de sürüm 2'nin (bazen "dual-yağınlar"). Sürüm 1 ve sürüm 2 de eşit toksunu sağlayan mekanizmalar, bazen yeni ve önceki sürümler arasında "geçiş" veya "birlikte çalışma stratejisi" olarak adlandırılır. IPv4'ten IPv6'ya ("TCP/IP"nin "IP" kismı) geçişte, geçiş sorunlarını çözmek için belki de IPv6'nın kendisini tasarlamak kadar uğraşılmıştır! (Elbette bir abartı, ancak önemli bir nokta var - konuşlanılmış sürüm 1 sistemleriyle birlikte çalışmak önemlidir.)

Görünüşe göre, sürüm 1 belirtiminizi yazarken sürüm 2 için planlar yaparsanız, "geçiş" veya "birlikte çalışma stratejisi" tanımlama görevini çok daha kolay hale getirebilirsiniz.

Genişletilebilirlik hükmünü şunşak şekilde tanımlayabiliriz:



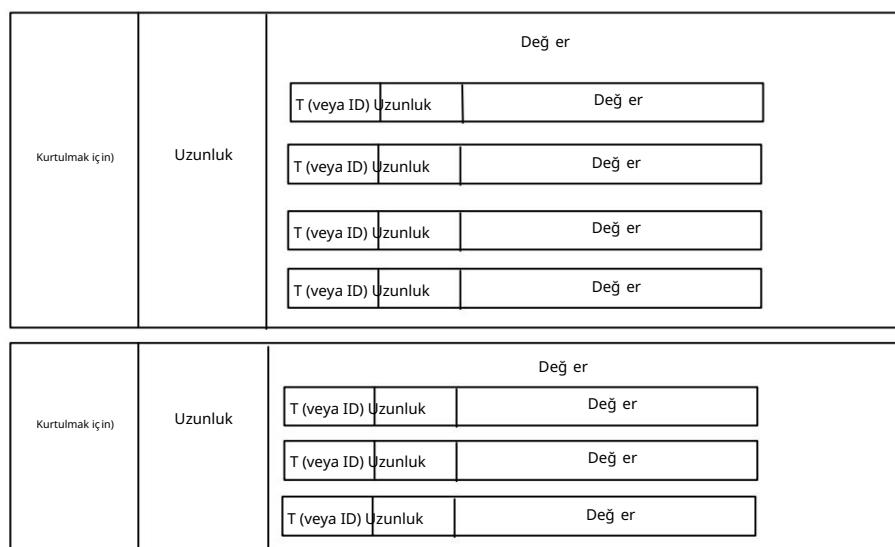
Şekil 6: Versiyon 1 ve Versiyon 2 birlikte çalışması

sürüm 1 mesajlarının belirli noktalarında bilinmeyen materyalin kapsüllenmesine izin veren sürüm 1 belirtiminin ögeleri ve

Bir mesajda bu tür bir materyal varsa, sürüm 1 sistemi tarafından yapılacak eylemlerin belirtilmesi.

ASN.1'deki geniş letilebilirlik hükmü, bu kitapta daha sonra tartışılacak olan önemli bir husustur ve Şekil 6'da gösterilmektedir.

Geniş letilebilirlik, ITU-T ve ISO'daki ilk çalışmalarda, mesajlardaki parametrelerin aktarılması için çok resmî bir tırılmış bir araç kullanılarak, "TLV" - Tip, Uzunluk, Değер olarak adlandırılan ve bir mesajdaki tüm bilgi parçalarının kodlandığı bir kavram kullanılarak mevcuttu. o parçanın doğasını tanımlayan bir tip alanı



Şekil 7: Parametreler ve gruplar için "TLV" yaklaşımı

bilgi, dē eri sınırlandıran bir uzunluk alanı ve ardından gönderilen bilgiyi belirleyen bir kodlama olan dē erin kendisi. Bu, parametreler ve parametre grupları için § ekil 7'de gösterilmektedir. Yaklaş ım, ASN.1 Temel Kodlama Kurallarında (BER) grup gruplarını ve benzerlerini her derinliği e kadar kapsayacak ş ekilde genelleş tirilmiş tir.

Dē er için kullanılan kodlamanın, yalnızca tür alanı tarafından tanımlanan parametre bağlamında uygulama bilgilerini açık bir ş ekilde tanımlaması gerektiği ini unutmayın. Bazı açık "sınıf" veya "tip" tanımlayıcı bağlamındaki bilgileri tanımlayan bu farklı sekiz dizi kavramı, daha sonra geri dönecek olan önemli bir kavramdır.

Sürüm 1 spesifikasyonunda, sürüm 2'de eklenen "tanınmayan" parametrelerin sessizce göz ardı edilmesini gerektirerek, sürüm 2 tasarımcıları, dā ıtlan sürüm 1 sistemleriyle birlikte çalışmak için öngörebilir bir temele sahip olur. Tabii ki, iyi tanımlanmış baş ka herhangi bir davranış kullanılabılır, ancak "sessizce yok sayma" yaygın bir belirtimdir. ASN.1, notasyon kullanılarak tanımlanabilecek herhangi bir mesaj için satırındaki gerçek bitleri birleştiren "kodlama kuralları" ile birlikte mesajların biçimini tanımlamak için bir notasyon sağlar. Yukarıda açıklanan "TLV", en eski ASN.1 kodlama kurallarına (Temel Kodlama Kuralları veya BER) dahil edilmiş tir ve "T" ve "L"nin her öge esinde bulunması nedeniyle geniş letilebilirlik için çok iyi destek sağlar. "yabancı" (sürüm 2) malzeme, kolayca tanımlanacak ve atlanacak (veya iletilecek). Bununla birlikte, geniş letilebilirlik i artırmada kullanımları dışında genellikle gereksiz olan kodlama tanımlaması ve uzunluk alanlarından muzdariptir. Uzun bir süre bu ayrıntıların geniş letilebilirliği in temel bir özellig i olduğu düşündü ve ASN.1 Paketlenmiş Kodlama Kuralları (PER), çok az ek yük ile geniş letilebilirlik için iyi bir destek sağladığında, kodlama kuralı tasarımda büyük bir baş arı oldu. astar.

2.5 Özet ve aktarım sözdizimi

Özet ve aktarım sözdizimi terimleri öncelikle OSI çalışması içinde geliş tirilmiş tir ve dī er ilgili bilgisayar disiplinlerinde çeşit itli ş ekillerde kullanılmaktadır. Bu terimlerin ASN.1'de (ve bu kitapta) kullanımı, OSI'deki kullanımlarıyla hemen hemen aynıdır, ancak ASN.1'i hiç bir ş ekilde OSI'ye bağ ımlı yapmaz.

Bir protokol oluş turan mesajları belirlerken aşagıdaki adımlar gereklidir (bkz. § ekil 8):

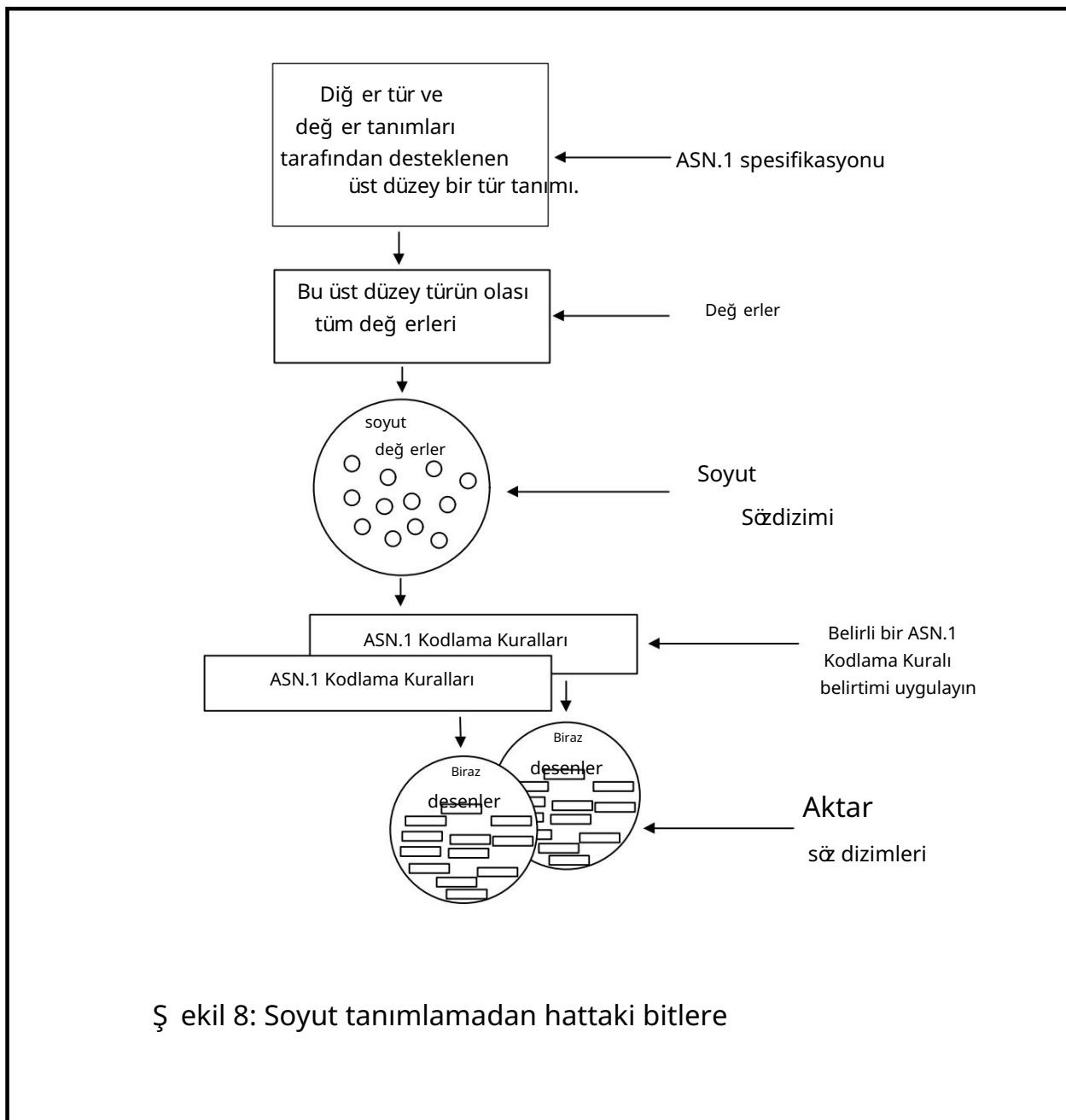
Her mesajda iletilemesi gereken bilgilerin belirlenmesi; bu "iş düzeyinde" bir karardır. Biz burada buna ilgili semantik olarak atıfta bulunuyoruz.

İ leti.

Gerekli anlambilimi taş ıyabilen bir tür veri yapısının tasarımını (üst düzey bir programlama dilinin genellik düzeyinde ve tanımlanmış bir notasyon kullanılarak). Bu veri yapısının dē er kümese mesajların veya uygulamanın soyut sözdizimi denir. Bu veri yapısını veya dē erler kümese tanımlamak için kullandığımız gösterime, mesajlarımız için soyut sözdizimi gösterimi diyoruz. ASN.1, birçok olası soyut sözdizimi gösteriminden yalnızca biridir, ancak muhtemelen en yaygın kullanılanıdır.

Soyut sözdizimi gösterimi kullanılarak tanımlanan herhangi bir mesaj verildiğiinde, bu mesajın anlamını taş ıyan satırındaki gerçek bitlerin yalnızca bir kez ve bir kez belirtilen bir algoritma tarafından belirlendiği (bağ ımsız olarak) mesajları kodlamak için bir kurallar dizisinin hazırlanması. baş vuru. Bu tür kurallara kodlama kuralları diyoruz ve bunları belirli bir uygulama için (soyut sözdizimi) mesaj grubuna uygulamanın sonucunun, o uygulama için bir transfer sözdizimini tanımladığıını söyleyoruz. Bir aktarım sözdizimi, soyut söz dizimindeki soyut dē erleri temsil etmek için kullanılacak bit kalıpları kümeseidir ve her bir bit kalıbı sadece temsil eder.

bir soyut değil er. (ASN.1'de, bir aktarım sözdizimindeki bit modelleri, çok çeş itli taş ıyıcı protokollerinde kolay taş ıma için her zaman 8 bitin katıdır).



Şekil 8: Soyut tanımlamadan hattaki bitlere

İkinci LINK protokollerinin elektrik sinyalini uygulama semantikinden net bir şekilde ayırmadığını gördük ve benzer şekilde bugün bazı protokol spesifikasyonları, soyut bir sözdiziminin spesifikasyonunu hattaki bitlerin spesifikasyonundan (transfer sözdizimi) net bir şekilde ayırmamaktadır. Kullanılacak bit modellerini (aktarım sözdizimi) ve her bir bit modeliyle ilişkili semantik doğrudan belirtmek hala yaygındır. Ancak daha sonra anlaşılıcağ gibi başı arızsızlık

Özetin transfer sözdiziminden açıkça ayrılması, yeniden kullanılabilirlik ve ortak araçların kullanımı için önemli çıkarımlara sahiptir. ASN.1 ile ayırma tamamlanmıştır.

2.6 Komut satırı veya ifadeye dayalı yaklaşımalar

Protokol tasarımlına yönelik bir diğeri önemli yaklaşım (ASN.1'de benimsenen yaklaşım değil), aktarılacak bilgileri tutmak için genel amaçlı bir veri yapısına odaklanmak yerine, her biri her komut veya ifade içinde metinsel parametreler (sıklıkla virgülle ayrılmış) bulunan bir komut veya ifade olarak düşünenmelidir. Bu yaklaşım, ASN.1'in kullanımından önceye dayanmaktadır, ancak bugün hala sıkılıkla, ITU'dan daha çok Internet tanımlı protokollerde (örneğin, World-Wide Web'i destekleyen Internet Hiper-Metin Aktarım Protokolü - HTTP-) kullanılmaktadır. -T/ISO tanımlı protokoller. Bu yaklaşımın daha ayrıntılı bir tartışması aşağıdaki 5.4'te verilmiştir.

2.7 Bir Arayüz Tanımlama Dilinin Kullanımı

Bir Arayüz Tanımlama Dilinin (IDL) kullanımı, ASN.1'in soyut sözdizimi yaklaşımına çok benzer. Ancak burada model, bir nesnenin işlevlerinin veya yöntemlerinin çağırılmasını ve sonuçlarının döndürülmesini sağlayan tanımlanmış arayüzler aracılığıyla bir ağ üzerinden etkileşimlere giren nesnelerdir. Model, her arabirimden geçen veri yapılarının yüksek düzeyde soyutlamada belirtilmesini sağlayan bir Arabirim Tanımlama Dili tarafından desteklenir.

Bugün muhtemelen en önemli IDL, Ortak Nesne İstek Aracı Mimarisi (CORBA) IDL'dir. CORBA'da IDL, IDL için kodlama kuralları ve ağlar arasında erişim arabirimlerine mesaj aktarma araçları dahil olmak üzere çok sayıda belirtim ve araçla desteklenir.

ASN.1 ve CORBA'nın ayrıntılı bir karşılık tırması, bu metnin ötesine geçer ve burada yapılan açıklamalar, bu yazarın 1999 ortalarındaki algısı olarak alınmalıdır. Özünde, CORBA, IDL'nin ve karşılık gelen kodlamaların yalnızca tuğlu eksiksiz bir mimari ve mesaj iletme özelliğidir. nispeten küçük (ama önemli) bir kısım. CORBA IDL, ASN.1 göstergesinden daha basit ve daha az güçlündür ve sonuç olarak kodlamalar genellikle ASN.1'in Paketlenmiş Kodlama Kuralı (PER) kodlamalarından çok daha ayrıntılıdır. ASN.1 genellikle, iletişim im kuran ortaklar arasında çok genel ve esnek mesaj alış-verişinin gerekli olduğu protokol spesifikasyonlarında kullanılırken, CORBA çok daha stilize edilmiş bir "çağırma ve yanıt" yaklaşımını teşvik eder ve genellikle çok daha sağlam bir destekleyici altyapıya ihtiyaç duyar.

3 Özet ve aktarım sözdizimleri hakkında daha fazla bilgi

3.1 Soyut değil erler ve türler

Çoğu programlama dili, tür veya sınıf kavramını (ve yerleşik türlere ve "yapı mekanizmalarına" atıfta bulunarak daha karmaşık bir türü tanımlamak için gösterimi), bir tür veya sınıfın değil eri kavramını (ve değil erleri belirtmek için notasyonu) içermektedir. ASN.1 farklı değil.

Örneğin, C'de yeni bir "Türüm" türünü şunşak şekilde tanımlayabiliriz:

```
typedef struct My-type { short first-item;
    boolean ikinci ög e} Türüm;
```

ASN.1'deki eş değer er tanım aşağıda görülmektedir.

ASN.1'de ayrıca temel türlerin veya daha karmaşık yapıların değer erleri kavramına sahibiz. Bunlara genellikle soyut değer erler denir (tekrar Şekil 8'e bakın), onları bir bilgisayarda veya bir iletişim hattında nasıl temsil edilebilecekleri konusunda herhangi bir endişe duymadan ele aldığıımızı vurgulamak için.

Kolaylık sağlamak için, bu soyut değer erler türler halinde gruplandırılmıştır. Örneğin, ASN.1 tip notasyonumuz var.

TAM SAYI

(az ya da çok) eksi sonsuzdan artı sonsuza kadar soyut değer erlerle tamsayı türüne başı vuran ASN.1 tip göstirimine de sahibiz

BOOLE

yalnızca iki soyut değer er olan "TRUE" ve "FALSE" ile boole türüne başı vuruda bulunur.

Kendimize ait bir tür tanımlayabiliriz:

```
Türüm ::= Dİ Zİ {birinci ög e TAM
                    SAYI, ikinci ög e BOOLEAN}
```

soyut değer erlerinin her biri bir "tamsayı" ve bir "boolean" değer eri çiftidir. Bununla birlikte, önemli olan nokta, birçok amaç için, "Türüm"deki değer erlerin herhangi bir iç yapısını umursamıyor (veya tartışmıyoruz). Tıpkı "integer" ve "boolean" gibi, bir dizi soyut değerinin atıfta bulunmanın uygun bir yoludur.

3.2 Soyut değer erlerin kodlanması

Dolayısıyla (yukarıdaki tartışmaya özetlemek için) ASN.1 kullanılarak tanımlanabilen herhangi bir tür için, bunun bir dizi soyut değer içeriğiini (temsil ettiğini) söyleyoruz. (Tekrar Şekil 8'e bakın).

Ama şimdiden önemli kısım için:

Belirli bir ASN.1 türündeki soyut değer erlere herhangi bir (doğru!) kodlama kuralı seti uygulandığında, her değer eri için bit modelleri (aslında sekizli diziler) üreteceklərdir, öyle ki verilen herhangi bir sekizli dizge tam olarak bir soyuta karşılık gelir değer eri.

Tersinin mutlaka doğru olmadığını unutmayın - belirli bir soyut değer eri için birden fazla sekizli dize olabilir. Bu, kodlama kurallarında seçenekler olabileceğini söylemenin başı bir yoludur.

(ASN.1, tüm uyumlu kod çözümcülerin, uyumlu bir kodlayıcının kullanmasına izin verilen tüm kodlamaları işlemesini gerektirir).

Kodlayıcı seçeneklerini, türdeki herhangi bir soyut değer için tam olarak bir kodlama olacak şekilde kısıtlarsak, kodlama kurallarının kanonik olduğunu söyleziz. Kanonik kodlama kurallarına ilişkin daha fazla tartışma, Bölüm III'te yer almaktadır.

Şimdi ASN.1 kullanarak bir protokolün mesajlarını belirlemek isteyen bir tasarımcıya ele alalım. Bir dizi ASN.1 türü (her farklı mesaj türü için bir tane) tanımlamak ve protokol değer iş tokuslarında iletilecek (ve dolayısıyla kodlamaya ihtiyaç duyan) soyut değer erler kümesinin tümünün kümesi olduğunu söylemek mümkün olacaktır. Tüm bu ASN.1 türlerinin soyut değer erleri. Gözlemci okuyucu (bazı insanlar bunu söylememeden hoşlanmayacak!), doğrudu bir kodlama kuralları kümesine ilişkin yukarıdaki gereklilikin soyut değer erlerin açık bir şekilde iletilmesi için yeterli olmadığını, çünkü iki soyut değer erin ayrı fakat benzer ASN'de olduğunu fark edecektir. 1 tür, aynı sekizli dizi temsiline sahip olabilir. (Her iki tür de iki tam sayıdan oluşan an bir dizi olabilir, ancak çok farklı anlamlar taşıyabilirler).

Soyut sözdizimi

soyut değer erler kümesi

uygulama için en üst düzey tür

Bu nedenle, ASN.1 kullanan protokollerin tasarılanmasında, bir uygulamada kullanılacak toplam soyut değer erler setinin tek bir ASN.1 tipinin soyut değer erler seti olarak belirtilmesi öremli bir gereklilikdir. Bu soyut değer erler kümesine genellikle uygulamanın soyut sözdizimi denir ve bazı kodlama kuralları uygulandıktan sonra karşılık gelen sekizli diziler kümesine o uygulama için olası bir transfer sözdizimi denir. Böylece, ASN.1 Temel Kodlama Kurallarının (Şekil 8'de olduğunu gibi) bir ASN.1 tip tanımına uygulanması, bu soyutları açık bir şekilde temsil etmek için kullanılabilen bir dizi bit modeli olan bir transfer sözdizimi (soyut sözdizimi için) üretir. Aktarım sırasında değer erler.

Aktarım sözdizimi

Aktarım sırasında soyut bir sözdiziminden bir değer eri temsil etmek için kullanılan bir dizi kesin sekizli dize

Verilerin bir ağ üzerinden aktarımından ziyade depolanmasına vurgu yapılan diğer bazı alanlarda, soyut sözdizimi kavramının hala soyut değer erler kümesini temsil etmek için kullanıldığıını, ancak somut sözdizimi teriminin bazen belirli bir değer için kullanıldığıunu unutmayın. malzemenin bir disk üzerindeki bit-desen gösterimi. Bu nedenle, bazı yazarlar sadece "aktarım sözdizimi" yerine "somut aktarım sözdizimi" hakkında konuşacaklar, ancak bu terim bu kitapta kullanılmıyor.

Farklı mesaj türleri için farklı ASN.1 türlerimiz varsa, soyut sözdizimimizi (ve dolayısıyla transfer sözdizimimizi) tanımlamak için bunları tek bir ASN.1 türünde nasıl birleşirebileceğimizi daha sonra göreceğiz. ASN.1'in 1994 sonrası sürümünde, bu "üst düzey" türü açıkça tanımlamak için özel gösterim vardır. Spesifikasyondaki diğer tüm ASN.1 tipi tanımları, yalnızca bu üst düzey tipi desteklemek için vardır ve eğeri onlar tarafından (doğrudan veya dolaylı olarak) referans alınmıyorsa, tanımları gereksizdir ve dikkat dağıtıcı bir ilgisizliktir! Çok ucuza, yayılanmış spesifikasyonlarda gereksiz tip tanımlarını tutmaz, ancak bazen tarihsel nedenlerle (veya baştan savma düzenleme veya her ikisi nedeniyle!) bu tür malzemelerle karşılaşabilirsiniz.

Özetle o zaman: ASN.1 kodlama kuralları, herhangi bir ASN.1 tipindeki soyut değer erleri temsil etmek için açık sekizli diziler sağlar; bir uygulama için en üst düzey türdeki soyut değer erler kümesine, o uygulama için soyut sözdizimi denir; bu soyut değer erleri açık bir şekilde temsil eden karşılık gelen sekizli dizilere (verilen herhangi bir kodlama kuralı kümesinin kullanımıyla) o uygulama için bir transfer sözdizimi denir.

Birkaç farklı kodlama kuralı belirtiminin mevcut olduğunu durumlarda (ASN.1 için olduğunu gibi), Şekil'de gösterildiği gibi, belirli bir uygulama için genel olarak birkaç farklı transfer sözdizimi (farklı ayrıntı ve geniş letilebilirlik - vb - özelliklerle) mevcut olabilir. 8.

OSI dünyasında, hangi transfer sözdiziminin kullanılacağı ina iliş kin çalış ma zamanı müzakeresine izin verilmesi uygun görülmüş tür. Bugün, genellikle uygulama tasarımcısının uygulamanın genel doğ asına ve gereksinimlerine göre bir seçim yapmasını bekleriz.

4 Değ erlendirmeli tartış ma

4.1 Bir kedinin derisini yüzmenin birçok yolu vardır - fark eder mi?

Soyut sözdizimi belirtiminin (iliş kili anlambilimle birlikte) bir aktarım sözdiziminin belirtiminden açık bir ş ekilde ayrılması, saf bir ş ekilde açıkça "temiz" olsa da, fark eder mi? Belirli bir uygulama için birden fazla aktarım söz dizimine sahip olmanın bir değil eri var mı? Protokol tasarımasına yönelik ASN.1 yaklaşımı, herhangi bir sayıda farklı uygulamanın soyut sözdizimini tanımlamak için ortak bir belirtim metni ve bundan transfer sözdizimini türetmek için ortak uygulama kodu ile ortak bir notasyon sağlar. Bu, daha önce tartış ilan karakter çizgisi yaklaşımına göre gerçekten avantajlar sağlıyor mu? Her iki yaklaşım da kesinlikle başarılı bir ş ekilde kullanılmıştır.

Farklı uzmanlar bu konuda farklı görüşlere sahiptir ve protokol tasarıının çoğuunda olduğu gibi, tercih ettiğiniz yaklaşımın mantıklı argümanlardan çok içinde çalışlığınız kültürle bağlı olması daha olasıdır. Aslında, her iki yaklaşımın da ş üphesiz avantajları ve dezavantajları vardır, bu nedenle bir karar, herhangi bir mutlak yargından ziyade, en önemli olduğuunu düşündüğünüz kriterlerden biri haline gelir. Yani burada (bu kitabın bazı kısımlarında olduğu gibi) Ş ekil 999: Okuyucular uyarısı alıyor (değil, "Sigara içmek" yerine "Bu tartış ma" geldi - Birleşik Krallık'taki tüm sigara paketlerinde görünen metinden) geçerlidir. (Bu kitapta biraz tartış malı olabilecek bir açıklama göründüğünde Ş ekil 999'a geri döneceğim). tartış ma sağlığı inize zarar verebilir!

Ş ekil 999: Okuyucular uyarısı alıyor

4.2 Birden çok aktarım sözdizimi ile erken çalış ma

Soyut ve aktarım sözdizimi kavramları açıklanmadan ve terimler tanımlanmadan önce bile, protokol belirleyiciler kavramları tanıdı ve belirtimlerinde birden fazla aktarım sözdizimi sağladı.

Bu nedenle, Bilgisayar Grafikleri Meta Dosyası (CGM) standardında, standardın gödesi, bir "ikili kodlama", bir "karakter kodlaması" ve "açık metin kodlaması". "İkili kodlama" en az ayrıntılı olandır, bir insan için okuması (veya hata ayıklaması) zordu, basit bir programla üretilmesi kolay değil idi ve 8 bit şeffaf bir depolama veya aktarım ortamı gerektiriyordu. "Karakter kodlaması", "komutlar" ve parametreler için iki karakterlik anımsatıcılar kullanıyordu ve prensipte bir metin düzenleyici tarafından üretilebiliyordu. İnsanlar tarafından daha okunabildi, ancak ASCII karakterlerini yazdırarak sekizlilerle eşlenmesi öneği idi ve bu nedenle kullanabileceğim depolama ve aktarım ortamlarında daha sağlılamış (ancak daha ayrıntılıydı). "Açık metin" kodlaması da ASCII tabanlıydı, ancak insanlar tarafından çok okunabilir olacak ş ekilde tasarlandı ve uygun bir metin düzenleyici kullanan bir insan tarafından üretilmeye veya hata ayıklama amacıyla bir insan tarafından görüntülenmeye çok uygun olacak ş ekilde tasarlandı. CGM için herhangi bir grafik arayüz aracı kullanıma sunulmadan önce kullanılabilirdi, ancak bundan sonra alakasızdı.

Bu alternatif kodlamalar, teknoloji olgunlaştıkça ve araçlar geliştiirildikçe ona pazar avantajı sağlayan "ikili kodlamaların" kompaktlığı ile farklı koşullar içinden uygundur.

4.3 Faydalar

Soyut sözdizimi tanımı için bir notasyon kullanıldığında ortaya çıkan faydalardan bazıları, uygun olan yerlerde karşılık argümanlarla birlikte aşağıdaki gibi tanımlanmışdır.

Yerel temsillerin verimli kullanımı

A tipi makinede makineye özgü bir biçimde depolanan büyük miktarlarda malzeme kullanan bir uygulamanız olduğumu varsayıyalım - örneğin her 16 bitlik tamsayıının en önemli sekizlisi alt adres baytında olacak şekilde. Bununla birlikte, B tipi makinede, farklı donanım nedeniyle, aynı soyut değerler temsil edilir ve her 16 bitlik tamsayıının en önemli sekizlisi daha yüksek adres baytında saklanır. (Makine-A/makine-B temsillerinde genellikle başka farklılıklar vardır, ancak tamsayıların bu sözcük "big-endian/little-endian" temsili genellikle en ciddi sorundur.)

A tipi makine ve B tipi makine arasında aktarım yaparken, taraflardan birinin veya her ikisinin (ve tarafsız olacaksa, her ikisinin de olması gereklidir!) CPU döngülerini bazlarına dönüş türmek ve bunlardan çökmek için harcaması açıkça gereklidir. kabul edilen makineden başka ımsız aktarım sözdizimi. Ancak, her ikisi de makine tipi A olan iki ayrı makine arasında aktarım yapıyorsak, bu makinelerdeki depolama biçimimle yakından ilgili bir aktarım sözdizimi kullanmak açıkça daha mantıklıdır.

Bu sorun, daha sonra toplu aktarım için parametreleri müzakere eden küçük başlıklar yerine, büyük miktarlarda yüksek düzeyde yapılandırılmış bilginin aktarımını içeren uygulamalar için genellikle daha önemlidir. İlgili olabileceğim bir örnek, Office Belge Mimarisi (ODA) belirtimidir. Bu, (örneğin) bir Boeing uçağı için eksiksiz bir servis kılavuzunu temsil edebilen büyük bir yapı için bir ISO Standardı ve ITU-T Tavsiyesidir, bu nedenle uygulama verileri son derece büyük olabilir.

Zaman içinde geliş tirilmiş temsiller

Bir protokol için üretilen ilk kodlamaların, kısmen "koruyucu" olma arzusundan veya uygulamanın konuşlanılmamasının ilk aşamasında, kısmen de basit bir şekilde, hata ayıklaması kolay kodlamalara sahip olma isteğiinden dolayı, genellikle verimsiz olduğum bir durumdur. zaman baskıları. Bunun nedeni, bu uygulama için "iyi" bir "bit-on-the-line" seti belirleme "sıkıcı" görevine yetersiz çaba gösterilmesi olabilir.

Bir kez daha, protokolün büyük kısmı, aktardığı bazı "toplulu veriler" ile karşılaştırıldığında küçük, durum - çoklu mesaj için - İnternet'in Hiper Metin Aktarım Protokolü (HTTP) veya Dosya Aktarım Protokolü (FTP) olduğu gibi, o zaman ana protokolün etkinliği nispeten öneksiz hale gelir.

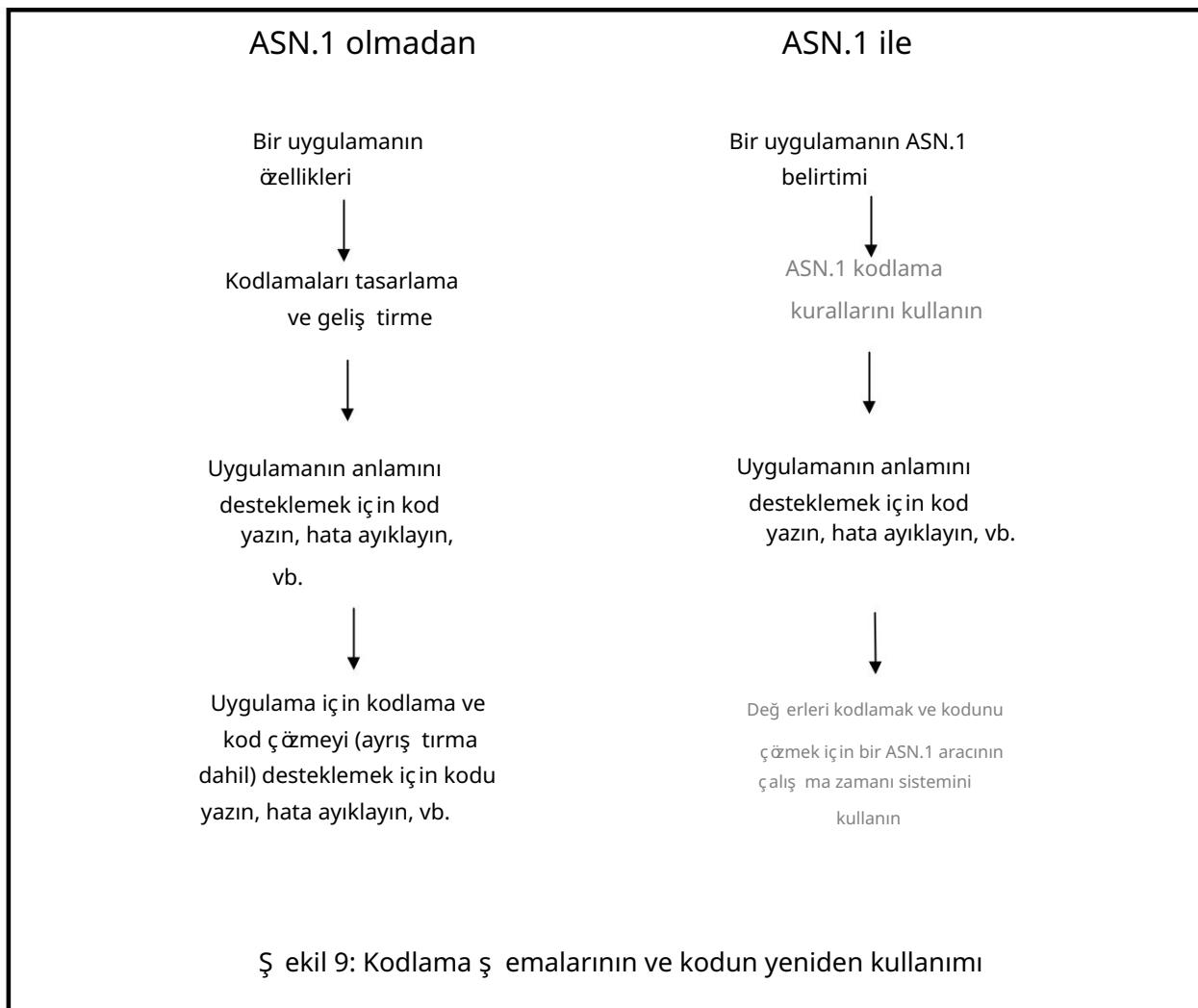
Kodlama biçimlerinin yeniden kullanımı

Soyut sözdizimi tanımı kavramını transfer sözdizimi tanımından net bir şekilde ayırsak ve soyut sözdizimi tanımı için herhangi bir uygulamadan başka ımsız bir gösterime (ASN.1 gibi) sahipsek, o zaman belirtim ve uygulama faydalari hemen artar.

Bu gösterim için "iyi" kodlama kuralları oluşturmak görevi bir kez yapılabilir ve bu kurallara, soyut sözdizimini tanımlamak için bu gösterimi kullanan herhangi bir uygulama tarafından başa kurulabilir. Bu

yeni bir uygulama belirleneceğse yalnızca büyük bir çaba tasarrufu sağ ılamakla kalmaz, aynı zamanda üzerinde tartışılmış, üzerinde anlaşmaya varılmış ve hata ayıklanmış bir aktarım sözdiziminin belirtimini de sağlar!

Bu yaklaşım ayrıca, uygulayıcılar için iyi anlaşılmış özellikler ve anlaşılık ile bir dizi farklı uygulama üzerinden ortaya çıkan transfer sözdizimlerine ortak bir "bak ve hisset" sağlar. Ayrıca, ağ ıda tartışılıan araçların ortaya çıkışmasını da mümkün kılar.



Avantaj, uygulamaya kadar uzanır. Uygulamadan bağımsız, net bir gösterim ve iyi tanımlanmış kodlama kurallarının olduğu yerlerde, herhangi bir uygulama tarafından kullanılabilen bir dizi jenerik kodlama/kod çözme yordamı sağ ılamak mümkün hale gelir. Bu, uygulama çabasını ve artık hataları önemli ölçüde azaltır. Şekil 9, gri metnin mevcut malzemenin yeniden kullanımı nedeniyle gerekli olmayan çabayı tanımladığı bu durumu göstermektedir.

kodun yapılandırılması

Kodlamaların belirtimi, soyut sözdizimi belirtiminden açıkça ayrı tutulursa ve ikincisi, uygulama dilinde veri yapılarına kolayca (bir araçla veya başka bir şekilde) eşlenebiliyorsa, bu teşvik eder (ancak elbette gerektirmez) © OS'ye modüler bir yaklaşım, 31 Mayıs 1999

Verilerin kodlanması sorumlu kodun, uygulamanın semantiğinden sorumlu koddan açıkça ayrı tutulduğ u uygulama tasarımlı.

Kodun ve ortak araçların yeniden kullanımı

Bu belki de ASN.1'in özellig i olan soyut ve aktarım sözdizimi belirtiminin ayrılmadan elde edilebilecek en büyük avantajdır.

Söze ASN.1 "derleyicileri" (bu bölümün 7. Bölümünde daha ayrıntılı olarak ele alınan ve uygulamadan bağımsız olan) kullanılarak, ASN.1'deki herhangi bir soyut sözdizimi tanımı (soyut) verilere eşlenebilir. o dildeki veri türlerinin metinsel gösterimi yoluyla herhangi bir programlama dilinin yapı modeli. Uygulayıcılar daha sonra asa ina oldukları (soyut) veri yapısı modelini kullanarak uygulamayı desteklemek için kod sağlayabilir ve aktarım için bu veri yapısının değil erlerinin kodlamalarını üretmek için uygulamadan bağımsız bir kod parçasını çağırabilir (ve benzer şekilde alımda şifre çözme).

Bu noktada okuyucunun yukarıdaki metinde "(özet)" ifadesinin neden yer aldığıını anlaması çok önemlidir. Tüm programlama dilleri (C'den Java'ya) kullanıcılarına yapıları tanımladıkları, erişiklikleri ve değil iş tirdikleri bir "bellek modeli" sunar. Bu tür modeller platformdan bağımsızdır ve genellikle ilişkili herhangi bir kodun bir düzeyde taşınabilirliğiini sağlar. Bununla birlikte, derleyiciler ve çalışma zamanı kitaplıklarını aracılığıyla gerçek bilgisayar belleğine (soyut veri yapılarının somut temsili) eşlemede, farklı platformların belirli özellikleri devreye girer ve bellekteki kesin temsil, makine türünden makine türüne farklılık gösterir (bkz. Bölüm III, Bölüm 4'teki "big-endian/little endian" tartışması).

Bir araç satıcısı, uygulayıcı tarafından kullanılan soyut veri yapılarının değil erlerini kodlamak/kodunu çözmek için (muhtemelen platforma özgü, ancak kesinlikle uygulamadan bağımsız) çalışma zamanı rutinleri sağlayabilir ve uygulayıcı, mutlu bir şekilde ayrıntılı doğrudan habersiz olmaya devam edebilir. ancak yine de uygulama dilinin değil işkenlerinde depolanan değil erlerden makineden bağımsız aktarım sözdizimlerini verimli bir şekilde üretebilir.

Kod yapısı, yeniden kullanılabilirlik ve araçlarla ilgili herhangi bir tartışmada olduğum gibi, gerçek faydalara yalnızca uygulanacak birden fazla uygulama olduğunda ortaya çıkar. Bazen tek bir uygulamayı desteklemek için genel amaçlı bir araç oluşturmak istenir, ancak çok uzaq zaman bu olmaz. Araçlar, aynı uygulayıcılar veya bir dizi uygulayıcı tarafından birden fazla uygulama için kullanılabilirler faydalıdır.

ASN.1'in araçlar gerçekten ortaya çıktı ve olgunlaşlığı tı çünkü ASN.1 çok çeşitli uygulamalar için tercih edilen spesifikasyon dili haline geldi.

Test ve hat izleme araçları

Mesajların sözdizimini tanımlamak için ortak bir gösterimin kullanılması, bir protokolün basit uygulanmasının ötesine geçen toplam protokol desteğiinin birçok yönünü otomatikleştirmeyi mümkün kılar. Örneğin, otomatik olarak test dizileri oluşturmak ve jenerik hat monitörleri veya "koklayıcılar" sağlama mümkün hale gelir.

Birden çok belge "yapıştırıcı" gerektirir

Soyut ve aktarım sözdizimi belirtiminin ayrılmaması, katmanlamadan farklı olsa da bazı ortak yönlere sahiptir. Spesifikasyonların ve kodun yeniden kullanılabilirliğini teşvik etmek, ancak uygulamanın uygulanmasının mümkün olabilmesi için birden fazla belgenin alınması ve okunması gerektiği anlamına gelir. Ayrıca, toplam spesifikasyonun iki bölüm arasındaki "tutkal" iyi tanımlanmadıkça, hatalara yer olduğu anlamına gelir.

ASN.1 söz konusu olduğunda, "tutkal" ASN.1 göstergesinin kendisidir ve normal kullanım için "tutkalın" "sıyrıldığı" neredeyse hiç bir durum olmamıştır. Bununla birlikte, kanonik kodlama kuralları sorusuna geldiğimizde - burada her bir soyut değer için ayrı bir bit modeli olması gereklidir, ancak yalnızca bir tane, "yapıştırıcı", soyut olanın tam olarak ne olduğunu çok net bir tanımını içermelidir. Herhangi bir ASN.1 tipindeki değerler. Bu, bazı teorik yapılar için kullanımlarının ilk on yılında ASN.1 spesifikasiyonlarıyla ilgili bazı sorunlara ve birçok tartışmaya neden oldu! (Ancak tüm gerçek dünya uygulamaları için hiçbir zaman bir sorun olmadı).

Diğer bir dezavantaj, spesifikasiyon belgelerinin, özellikle "tutkal" - ASN.1 göstergesinin, isteyen herhangi bir tarafından ücretsiz olarak (ücretsiz olarak) mevcut olmaması durumunda ortaya çıkar. Bu, teorik olarak son on buçuk yılda ASN.1 ile ilgili bir sorun olmuştur, ancak ASN.1 belgeleri için ITU-T/ISO fiyatlarını ödemeyen neredeyse herkesin ASN.1 belgelerini almayı başardığından şüpheleniyorum. Öyle ya da böyle onları!

"Araçlar" işi

ASN.1 gibi yüksek düzeyde uygulamadan başımsız bir notasyonda soyut bir sözdizimini ifade etmek, araçların kullanımını sağlar, ancak kendisi gerektirmez ve ASN.1'i kullanan ilk spesifikasiyonun üretilmesinden yaklaşık beş yıl sonra "ASN.1 araçlar" piyasaya çıkmaya başlıyor.

Bugün, uygulama görevlerine yardımcı olacak bir araç edinmenin maliyetini haklı çkarabilenler için ticari bir avantajla, göstergeler ve kodlama kuralları için yeni bir "ASN.1 araçları" iş alanı oluştu.

5 Protokol spesifikasiyonu ve uygulaması - bir dizi vaka çalışması

Bu bölüm, ASN.1 kullanıldığından benimsenen yaklaşımın basit bir sunumuyla sona eren protokol spesifikasiyonuna ve uygulamasına yönelik bir dizi yaklaşımın tartışılmasıyla bu bölüm tamamlanır.

5.1 Sekizli diziler ve sekizli içindeki alanlar

Bilgilerin tamamının veya çoğu unun, hepsinin mevcut olması gereken sabit uzunluklu alanlar olarak ifade edilebileceği protokoller, geleneksel olarak Şekil 10: Geleneksel yaklaşım'da gösterilenler gibi diyagramlar çizilerek belirtilir.

Şekil 10, İnternet Protokol Başlığıının bir parçasıdır (İnternet Protokolü, Şekil 2'de gösterilen TCP/IP yığının IP protokolüdür. Başlık alanlarını tanımlamak için X.25 seviye 2'de benzer bir resim kullanılır.

sekizli sayı
protokol kimliği
Uzunluk
Sürüm
Ömür
S P M S / E R Tip
segment uzunluğu
sağlama toplamı
vb

Şekil 10: Geleneksel yaklaşım

Bu yaklaşım, uygulamaların birleştiirici dil veya BCPL veya sonraki C gibi diller kullanılarak gerçekleştirildiği ve uygulayıcının bir bilgisayar belleğinin ham bayt dizisiyle yakın temas kurmasına izin verdiği ilk günlerde çok popülerdi.

Uygulayıcı içinden iletişim im hattından bellekte belirli bir yere sekizli olarak okumak ve ardından gerektiğiinde farklı alanlara (şematik olarak gösterildiği gibi) uygulama kodu erişimi kablolamak nispeten kolaydı. Benzer şekilde iletişim için. Bu yaklaşımında "kodlama" ve "kod çözme" terimleri genellikle kullanılmamıştır.

Yaklaşım yetmişli yılların ortalarında işe yaradı, tek gözle çarpan başarılılıklar (bir durumda), sekizli bir oktet olarak yorumlarken sekizlilerin (şematik olarak verilen) hangi ucunun en önemli olduğu unun belirtilmesindeki netlik eksikliğiinden kaynaklanıyordu. sayısal değerler ve (şematik olarak verilen) sekizlilerin hangi ucunun bir seri hatta ilk olarak iletilmesi i. Kili tabanlı protokol belirtiminde bu bit sıralarının çok açık bir belirtimine duyulan ihtiyaç bugün iyi anlaşılmıştır ve özellikle ASN.1 belirtimi içinde ele alınmaktadır ve bir ASN.1 tasarımcısı veya uygulayıcısı tarafından göz ardı edilebilir. tabanlı belirtim.

5.2 TLV yaklaşımı

En basit protokoller bile mesajların değerlerini uzunluklu "parametrelerine" ve isteği doğa bağlı olarak çıkarılabilir parametrelerine ihtiyaç duyuyordu. Bu, daha önce bölüm 2.4'te kısaca açıklanmışdır (bkz. Şekil 7).

Bu durumda, belirtim normalde bazı sabit uzunluklu zorunlu başlık alanlarını ve ardından bir "parametre alanı" (genellikle bir uzunluk sayımı ile sonlandırılır) tanımlar. "Parametre alanı", her biri bir tanımlama alanı, bir uzunluk alanı ve ardından parametre değerleri ile kodlanmış bir veya fazla parametre dizisi olacaktır. Sabit uzunluklu bir parametre için bile uzunluk alanı ve zorunlu bir parametre için bile tanımlama alanı her zaman mevcuttu. Bu, temel "TLV" yapısının korunmasını sağladı ve sürüm 1 sistemleri için tanımadıkları parametreleri atlamak için "geniş letilebilirlik" metninin yazılmasını sağladı.

Bir uygulayıcı, giriş akışını taramak ve parametreleri bellekteki bağlı lantılı bir arabelik listesine yerleştirmek için oldukça genel amaçlı bazı kodlar yazar, uygulamaya özel kod daha sonra bağlı lantılı arabelikleri işler. Bununla birlikte, bu yaklaşımın birçok spesifikasyonda oldukça yaygınmasına rağmen, uzunluk kodlamasının kesin ayrıntılarının (örneğin, 255'lik bir sayıyla sınırlı veya sınırsız), spesifikasyondan spesifikasyona değerlerinin, bu nedenle bu parametreleri işlemek için herhangi bir kodun değerlerini mevcut olmadığıunu unutmamak. uygulamaya özel olmalı ve diğer uygulamalar için kolayca yeniden kullanılmalıdır.

Protokoller daha karmaşık hale geldikçe, tasarımcılar, belirli bir gruptaki tüm parametrelerin parametre alanında bir araya toplandığı, var olan veya atlanan eksiksiz parametre gruplarına sahip olma ihtiyacını keşfettiler. Bu, Teletex (ve daha sonra OSI Oturum Katmanı) spesifikasyonlarında benimsenen yaklaşım ve bir parametre grubu için bir dıştan tanımlayıcı, o grubun sonunu gösteren bir uzunluk alanı ve ardından ikinci bir TLV düzeyine yol açtı. gruptaki her parametre için TLV (Şekil 7'yi tekrar ziyaret edin).

Bu yaklaşım, belirli bir parametre değerinin değerini ve sayıda tekrarını gerektiren bilgiler için de çok uygundu.

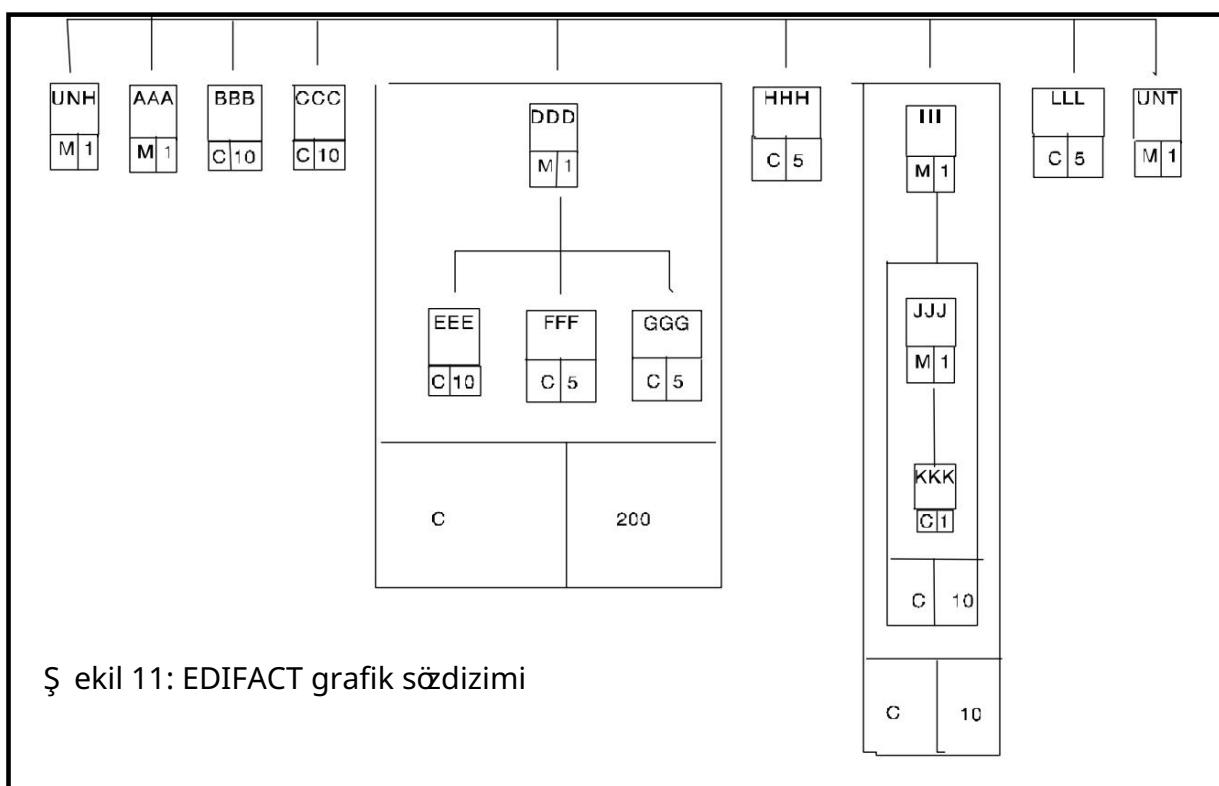
Uygulama düzeyinde, bir giriş sekizli dizesini "ayrıştırın" kodu artık biraz daha karmaşıklaştırır ve uygulamaya özel koda iletilen sonuçta ortaya çıkan veri yapısı, basit bir bağlı lantılı liste yerine iki seviyeli bir ağ yapısı haline gelir. , seviye 1 düğümleri parametre gruplarıdır ve seviye 2 düğümleri parametreleridir.

Bu yaklaşım yukarıda çok "saf" bir biçimde sunuldu, ancak gerçekte nadiren bu kadar saftı! Teleteks ve Oturum Protokolleri aslında en üst düzey parametre grubu TLV'leri ve parametre TLV'lerinde birbirine karışmış tir!

ASN.1 Temel Kodlama Kuralları - BER - (daha sonra çok daha ayrıntılı olarak açıklanacaktır) hakkında halihazırda bilgi sahibi olanlar, bu TLV yaklaşımının BER tarafından kullanılan temel (uygulamadan bağımsız) kodlamayı oluş turmak için genelleş tirildiği ini fark edeceklerdir. BER için mesajın tamamı, bir tanımlayıcı (onu aynı soyut sözdizimindeki diğer er mesaj türlerinden ayıran) ve mesajın sonunu gösteren bir uzunluk alanı ile sarılır. Göerde, genel olarak, her bir üçluğun "V" kısmı ya baş ka TLV üçlüleri (herhangi bir derinlik e kadar) veya bir tamsayı veya bir karakter gibi "ilkel" bir alan olmak üzere, baş ka TLV üçlülerinin bir dizisidir. sicim.

Bu, normal programlama dili veri yapısı tanımlarının, tür gruplarını ve türlerin tekrarlarını herhangi bir derinlik e kadar tanımlama gücü için tam destek sağlar ve ayrıca isteğe bağlılığı eler ve geniş letilebilirlik için her düzeyde destek sağlar.

5.3 EDIFACT grafik sözdizimi



Şekil 11: EDIFACT grafik sözdizimi

Bu yaklaşım, soyut sözdizimi belirtimi için net (grafiksel) bir notasyon ve ayrı bir kodlama kuralı belirtimi ile ASN.1'e en yakın olanıdır. Yönetim, Ticaret ve Taş İma için Elektronik Veri Alış Veriş (EDIFACT) grafik sözdiziminin bir örneği Şekil 11'de verilmiştir. EDIFACT grafik sözdizimi, ASN.1'de olduğu gibi, toplam mesajın tanımı, parçalar için referans adları kullanılarak uygun boyuttaki parçalar halinde yapılabilir, ardından bu parçalar, tam mesajı tanımlamak için birleş tirilir. Dolayısıyla, Şekil 11'de (daha önce veya daha sonra tanımlanmış) mesaj parçasına sahibiz: zorunlu olarak bir kez mevcut olan "UNH", benzer şekilde "AAA", ardından koşullu olarak ve sıfır ile on kez mevcut olan "BBB" ve ardından benzer şekilde "CCC". daha sonra bir "DDD" ve ardından on adede kadar "EEE" vb.'den oluşan an bileşik bir yapının 200'e kadar tekrarı.

Gerçek kodlama kuralları, ASN.1'de olduğu gibi, ayrı olarak belirtilmişti, ancak tüm alanların karakter kodlamasına dayanıyordu. Grafik notasyonu, ASN.1 notasyonundan daha az güçlündür ve ilkel tiplerin aralığı çok daha küçüktür. Kodlama kuralları, yinelenen bir diziyi izleyen bir türün, bu yinelenen dizideki türden farklı olmasını sağlamak için uygulama tasarımcısına da güvenir, aksi takdirde belirsizlik oluşur. Bu, ASN.1'in herhangi bir yasal parçasının kesin kodlamalar ürettiğinin ASN.1'de kaçınılan bir sorundur.

Uygulama düzeyinde, EDIFACT tanımını uygulama dili için bir veri yapısına eşlemek mümkün olacaktır, ancak şunu anda bunu yapan herhangi bir araçtan haberdar değilim.

5.4 Karakter tabanlı bir sözdizimi belirtmek için BNF kullanımı

Bu yaklaşım daha önce kısaca açıklanmışdır ve birçok Internet protokolünde yaygındır.

Bu karakter tabanlı yaklaşımın kullanıldığı durumlarda, her mesaj için izin verilen metin satırlarının kesin olarak belirtilmesi gereklidir. Bu spesifikasyon, soyut bir sözdiziminin tanımına benzer, ancak satırındaki bilgilerin temsiline, soyut bir sözdiziminin ASN.1 tanımında olduğu undan daha fazla odaklanır.

Bu sözdiziminin tanımlamak için kullanılan gösterim, genellikle programlama dillerinin sözdizimini tanımlamak için sıkılıkla kullanılan (ve aslında ASN.1'in kendi sözdizimini tanımlamak için kullanılan) bir gösterimin bir varyasyonudur; orijinal mucitleri.

Örneğin, ASN.1'de BNF ifadeleri:

```
EnumeratedType ::= SAYILANDIRILMIŞ { Numaralandırma }
Numaralandırma ::= NamedNumber |
                    numaralandırma , İ simli Numara
NamedNumber ::= tanımlayıcı(SignedNumber)
SignedNumber ::= sayı | - sayı
```

dilin yapılarından birinin "SAYILANDIRILMIŞ" kelimesinden oluşan tuşunu belirtmek için kullanılır, ardından kırımlı parantez içinde, her öğenin bir tanımlayıcı olduğu virgülle ayrılmış bir liste ve ardından bir sayı (muhtemelen önünde bir eksi işareti bulunur) yuvarlak parantez içinde.

Ne yazık ki, günümüzde kullanımında olan birçok BNF varyasyonu vardır ve onu kullanan uygulamaların çoğu, kendi özel BNF notasyonlarını tanımlamayı gerekli bulmaktadır. Bu, BNF tabanlı spesifikasyonları desteklemek için ortak araçları kullanmayı olması gerekenden daha zor hale getirir.

BNF nispeten düşük seviyeli bir gösterim destek aracıdır. Keyfi sözdizimsel yapıları tanımlamak için çok güçlündür, ancak değil işken uzunluklu öğelerin nasıl sınırlanacağıını veya yineleme sayılarının nasıl belirleneceğiini kendi başına belirlemez. Aynı BNF notasyonunun kullanıldığı yerlerde bile, bu şekilde tanımlanan iki protokolün "bak ve hisset"i, dizeleri (tırnak işareti, ayrılmış karakterler, çıkışlı ayrılmış karakterler) veya öğelerin değil işken uzunluk tekrarları, bu tanım için BNF notasyonu kullanılarak özel uygulamaya yazılmalıdır.

Elbette her araçta olduğu gibi tasarım iyi olursa iyi bir sonuç çıkabilir. Internet protokolü tasarımlarının çoğu bu yaklaşımı benimsenir ve en iyi tasarımcılar, uzunluk ve yineleme sonlandırmalarının elde edilmesi eklinin, diğer ilgili spesifikasyonlarda benimsenen yaklaşımı mümkün olduğunda yakından takip etmesini ve o uygulama içindeki farklı alanlar ve komutlar için tutarlı olmasını sağlar.

BNF tabanlı spesifikasyonları destekleyen yazılım araçları, genellikle gelen bir dizinin sözcüksel analiziyle sınırlıdır ve genellikle uygulamaya özel kod ve kodlama konularının, bir ASN.1 aracı kullanıldığıında normalde olacağından daha yakından iç içe geçmesiyle sonuçlanır.

İletilerdeki satırlar için tanımlama alanları nispeten uzun adlar olma eğilimindedir ve "sıralamalar" da uzun ad listeleri kullanma eğilimindedir, bu nedenle ortaya çıkan protokol oldukça ayrıntılı olabilir. Bu yaklaşımarda, uzunluk alanları normalde ayrılmış karakter sınırlayıcılar veya satır sonuyla değil iş tırılır, genellikle birkaç satırda devam etmeye izin vermek için bir tür kaçış veya uzatma mekanizmasıyla (yine bu mekanizmalar farklı alanlar için her zaman aynı değil) veya farklı uygulamalar için).

Son yıllarda, birkaç İnternet protokolü için sözdizimini tanımlamak için tam olarak aynı BNF gösterimini kullanma girişimi olmuş tur, ancak varyasyonlar hala devam etmektedir.

Uygulama zamanında, bir gönderen uygulama, karakter bilgisini doğrudan satır veya arabelleğe yazdırma için kodlamayı bir dizi "YAZDIR" ifadesi olarak fiziksel olarak bağlayacaktır. Alım sırasında, normalde BNF belirtimi ile sunulabilen ve giriş dizesini ana sözlük öğelerine ayırtırın genel amaçlı bir araç kullanılır. Bu tür araçlar, Unix sistemleri için ücretsiz olarak mevcuttur ve bu şekilde tanımlanan protokollerin uygulamalarının Bilgisayar Bilimleri öğrenicileri için görevler olarak ayarlanması kolaylaşır (özellikle protokol belirtimleri de ücretsiz olarak mevcut olma eğiliminde olduğunu!).

Özet olarak, aktarılacak bilgi doğrudan olarak iki seviyeli bir yapıya uyuyorsa (metin satırları, bir tanımlayıcı ve her satırda virgülle ayrılmış metin parametreleri listesi) bu yaklaşım işe yarayabilir, ancak ne zaman karmaşık olabileceğine? yineleme? ögelerin değil işken sayılarının iç içe geçmesi için daha derin bir derinlik gereklidir ve bir parametrenin parçası olarak virgülere izin vermek için kaçış karakterlerine ihtiyaç duyulduğunda. Yaklaşım aynı zamanda ASN.1 BER'in ikili yaklaşımından çok daha ayrıntılı bir kodlama ve ASN.1 Paket Kodlama Kurallarından (PER) çok daha ayrıntılı bir kodlama üretme eğilimindedir.

5.5 ASN.1 kullanarak spesifikasyon ve uygulama - 1980'lerin başı

ASN.1 ilk olarak X.400 Mesaj İşleme Sistemleri CCITT (Uluslararası Telgraf ve Telefon Danışma Komitesi, daha sonra adı ITU-T olarak değil iş tırilecek) Önerileri setinin tanımını desteklemek için gelişti, ancak temel fikirler Xerox Courier'den alındı. Şartname.

X.400, elektronik mesajlaşma için bilgi akışlarının semantikini doğrudan bir şekilde elde etme konusunda güçlü bir uygulama ilgisi olan, ancak mesajların bit düzeyinde kodlaması hakkında endişelenmeye nispeten az ilgi duyan kişiler tarafından gelişti. Spesifikasyon çalışma malarını desteklemek için üst düzey bir programlama dilinde aşırı yukarı veri yapısı tanımlama gücüne ihtiyaç duydukları açıklandı ve ASN.1 bunu sağlamak için tasarlandı.

Tabii ki, gerçek bir programlama diline daha yakın notasyon kullanılabildi, ancak bu, uygulamayı (platform nedeniyle) farklı bir dil kullanmaya zorlanabilecek olanlar için uygulamayı kolaylaştırmazdı. Ayrıca, mevcut bir dilin kullanılması notasyon problemini çözebilirken, aynı platformdaki aynı dilden veri yapılarının bellek içi temsilleri derleyiciden farklı (ve bugün hala farklı) olduğunu undan, kodlamaları tanımlamak için hala çalışma yapılması gerekecektir. yazardan derleyici-yazara.

Böylece ASN.1 üretildi ve X.400 ve diğer birçok ITU-T ve ISO spesifikasyonu tarafından yoğun bir şekilde kullanıldı; burada, tasarımcılara önemli olan şeyle - uygulama semantikine - konsantre olmaları için verdiği güç ve özgürlük çok takdir edildi. Daha sonra ASN.1 birçok alanda kullanılmaya başlandı.

telekomünikasyon uygulamaları ve belirli iş sektörlerindeki uygulamalar (ve son zamanlarda SET - Güvenli Elektronik İşlemeler).

1980'lerin başında, etraftaki tek ASN.1 araçları, tasarımcıların spesifikasyonu doğrudan bir şekilde anlamalarına yardımcı olan basit söz dizimi denetleyicileriydi. Kodlama kuralları, daha önce açıklanan TLV tabanlı BER idi ve uygulama mimarileri, daha önce açıklanan karakter komut satırı yaklaşımı için kullanılanlara benzer olma eğilimindeydi. Yani bir kodlamanın "T" ve "L" kısmını (ve integer ve boolean gibi ilkel tipler için "V" kısmını) oluşturmak için bazı rutinler üretildi ve mesajın yapısı zordu. -hat boyunca iletim için T ve L parçaları oluşturmak üzere bu alt programlara tekrarlanan çağrılarla uygulamaya kabul edilmişdir. Alım sırasında, oldukça basit (ve uygulamadan bağımsız) ayrıştırma kodu, iç içe TLV kodlamalarının giriş akışını almak ve tamsayılar, boolean'lar gibi ilkel çağrıelerin kodlamalarını içeren açın yaprakları ile bellekte bir ağ yapısı oluşturmak için yazılabilir. karakter dizileri vb. Uygulama kodu daha sonra girdi değil erlerini elde etmek için bu yapıyı "ağ aç" yürütüşür yapacaktır.

Böylece bu ilk günlerde, ASN.1 gösterimi:

Mesajların bilgi içeriğini belirlemenin güçlü, açık ve kullanımı kolay bir yolu sağladı.

Uygulama tasarımcılarını kodlamayı ilgili endişelerden kurtardı.

Sağlanan uygulamadan bağımsız kodlama, yeniden kullanılabilir kodun ve gelişmiş araçların geliş tirilmesini mümkün kılar, ancak anında gerçekleş tirilemez.

Uygulayıcılara, BNF tabanlı yaklaşımı kadar ayrıntılı olmayan ve uygulaması daha zor olmayan (ama daha kolay da olmayan) bir dizi kodlama kuralı verdi.

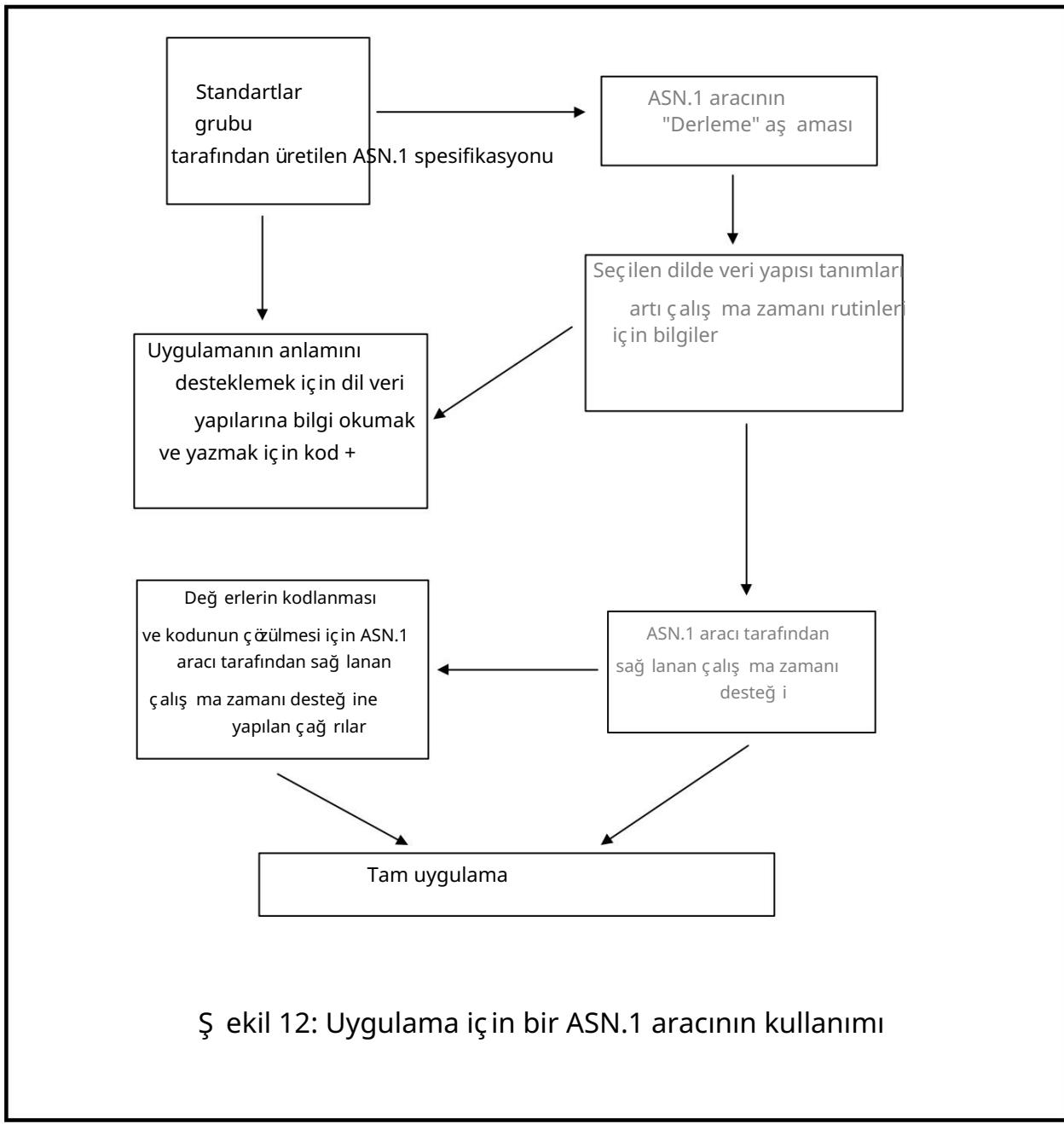
5.6 ASN.1 - 1990'ları kullanan spesifikasyon ve uygulama

ASN.1 tabanlı bir protokolün bir uygulamasını araçlar olmadan üretmek elbette ki hala mümkün değildir. 1980'lerde yapılanlar bugün hala yapılabilir. Ancak, bugün uygulamalar için "pazara çıkış süresini" kısaltmak ve artık hataların minimumda olmasını sağlamak için büyük bir baskı var. Araçların kullanımı bu açıdan çok önemli olabilir.

Bugün ASN.1 kodlama kurallarının iki ana ailesi vardır, orijinal (değil iş memiş) BER ve daha yakın tarihli (1994'te standartize edilmiş) PER (Paketlenmiş Kodlama Kuralları). PER kodlama kuralları belirtimi, BER'inkinden daha karmaşıkktır, ancak çok daha kompakt kodlamalar üretir.

(Örneğin, PER'de bir boole değerin kodlanması yalnızca tek bir bit kullanır, ancak BER'in TLV yapısı en az 24 bit üretir!)

BER için bir araç olmadan kodlama/kod çözme, ayıracak zamanınız varsa yapılacak kabul edilebilir bir şeys olsa da, PER kullanılıyorsa bunun uygulama hatalarına yol açabileceğine dair ortaya çıkan "geleneksel bir bilgelik" var gibi görünüyor. Okuyucu tekrar Şekil 999'a baş vurmalıdır: Okuyucular uyarı alırlar. Bu yazar, araçlar olmadan PER kodlama/kod çözmemi çok uygun bir önerme haline getiren uygulama stratejileri olduğunu iddia edecktir. Çeşitli epleri kodlamak için kullanılacak alan genişliklerini ve dolgu bitlerinin ne zaman ekleneceğini doğrudan bir şekilde belirlemek için tasarım aşamasında kesinlikle çok daha fazla özen gösterilmesi gereklidir (bu yorum, PER ile ilgili bölüm okunduktan sonra daha iyi anlaşılıcaktır), ancak bir kez bu yapıldıktan sonra, bir PER kodlamasını/kodunu uygulama koduna çözmenin hala mümkün olduğunu (bu yazar iddia ederdi).



Şekil 12: Uygulama için bir ASN.1 aracının kullanımı

Bununla birlikte, bugün "ASN.1 derleyicileri" olarak adlandırılan iyi araçlar mevcuttur ve herhangi bir ticari gelişim için paranın karşılığının fazlasıyla alırlar ve yaygın olarak kullanılırlar. Nasıl bir uygulama yapardınız?

Bir araç kullanarak ASN.1 spesifikasyonu? Bu, bu bölümün son bölümünde ("OSS ASN.1 Tools" paketine dayanan örneklerle) daha ayrıntılı olarak ele alınmıştır. Bununla birlikte, temel çerçeveye aşağıdaki idaki gibidir (bkz. Şekil 12).

Uygulama tasarımcısı tarafından üretilen ASN.1, aracın "derleme aşamasına" beslenir. Bu, ASN.1'i C, C++ ve Java da dahil olmak üzere çok çeşitli desteklenen dillerin (ve platformların) herhangi birinde bir dil veri yapısı tanımına eşler. Uygulama kodu daha sonra, yalnızca uygulamanın gerekli semantikine odaklanarak bu veri yapılarından değerlerini okumak ve yazmak için yazılr.

Bir kodlama gerektiğiinde, ASN.1 tanımının belirli yönleri hakkında derleme aşaması tarafından sağlanan bilgileri kullanan ve bu platformda bilgilerin bellekte temsil edilmesi eklini "anlayan" bir çalışma zamanı rutini çağırır. Çalışma zamanı yordamı tüm iletiyi kodlar ve sonuçtaki sekizli diziyi döndürür. Kod çözme için benzer bir iş işlemi kullanılır. Big-endian veya little-endian bayt düzeniyle ilgili herhangi bir sorun (bkz.).

Elbette, bir araç kullanmadan, ASN.1'i bir dil veri yapısına eşleme ve bu veri yapısını kodlamak ve kodunu çözmemek için ayrı bir koda sahip olmak gibi benzer bir yaklaşım mümkündür, ancak muhtemelen daha fazla iş (ve daha fazla hata eğilimli) olacaktır. Yukarıda özetlenen daha "kablolu" yaklaşım. Ancak eşleme ve kodlama/kod çözme rutinlerini sağlayan bir araçla, bu, ASN.1 tabanlı bir uygulamanın uygulanmasını üretmenin son derece basit ve hızlı bir yoludur.

Sonuç olarak, bugün ASN.1 aracını kullanarak:

Protokol tasarımcılarının mesajların bilgi içeriğiini belirlemeleri için güçlü, açık ve kullanımı kolay bir yol sağlar.

Uygulama tasarımcılarını kodlama, isteği bağlı olarak eklelerin tanımlanması, listelerin sonlandırılması vb. endişelerinden kurtarır.

ASN.1 yapılarını bugün kullanılan ana bilgisayar dillerinin yapılarıyla eşleşen araçlar tarafından desteklenir.

Uygulayıcıların, tüm ASN.1 kodlama kuralları için hatalı kodlamalar üretken uygulamadan bağımsız çalışma zamanı kodlama/kod çözme yordamlarını kullanarak, kodlama/kod çözme ile herhangi bir endişe duymadan yalnızca uygulama semantikine konsantre olmalarını sağlar.

ASN.1 izin verir

Tasarımcılar uygulama semantikine odaklanacak

İlgili hataları kodlamadan ve mevcut kompakt kodlamalarla tasarım yapın

Uygulayıcılar, uygulamayı desteklemek için minimum kod yazacak - hızlı geliş tirme

Birlikte çalışma zamanı sorunları olmadan hatalı kodlama/kod çözme.

Bölüm 2

ASN.1'e Giriş

(Veya: Yazmadan önce okuyun!)

Özet:

Herhangi bir dili veya gösterimi *öğ* renmenin en iyi yolu, bir kısmını okumaktır. Bu bölüm, ASN.1 tip tanımlarının küçük bir örneğ *ini* sunmakta ve aş *ağ* ıdakilerin ana kavramlarını tanitmaktadır:

yerleş *ik* anahtar kelimeler,
inş *aat* mekanizmaları,
tip-referans-adlarına sahip kullanıcı tanımlı tipler,
tanımlayıcılar veya "alan adları",
alternatifler.

Bölüm II'de daha ayrıntılı olarak tartış *ilan* "etiketleme"ye bir atıf vardır.

Bu bölüm, ASN.1'e yeni baş *layanlar* için hazırlanmış *tır* ve gösterime önceden maruz kalmış *olanlar* tarafından atlanabilir.

1. Giriş

Ş ekil 13'e bakın. Buradaki amaç, tanımladığı *i* veri yapısını - bu yapının bir değ *erinin* iletiminin ileteceğ *i* bilgiyi - basitçe anlamlandırmaktır.

Ş ekil 13, ASN.1'in özelliklerini göstermek için tasarlanmış "yapay" bir örnektir. Ele aldığı *i* soruna mutlaka en iyi "iş *çözümünü*" temsil etmez, ancak ilgili okuyucu onun daha ilginç özelliklerinden bazıları için makul bir gerekçe bulmaya çalış abilir.

Örneğ *in*, "yurtdış *ı*" davası "İ *ngiltere*" davasının tutabileceğ *i* herhangi bir bilgiyi barındırabilirken neden "İ *ngiltere*" ve "denizaş *ırı*" için farklı "ayrınlıklar" kullanılmış *tır*? Makul cevap, "İ *ngiltere*" vakası 1. versiyondaydı ve "denizaş *ırı*" daha sonra iş *geniş* lediğ *inde* eklendi ve tasarımcı aynı bitleri "İ *ngiltere*" davası için on-line tutmak istedi.

Bu örnek, bu kitap ilerledikçe oluş *turulmuş tur* ve bu "Wineco protokolü" senaryosu, Ek 2'deki tüm protokolle birlikte Ek 1'de yer almaktadır.

ASN.1, elbette, normalde birden fazla yazı tipinde değ *il*, yalnızca bir yazı tipinde yayınlanır (çoğ *unlukla* Courier). O noktaya daha sonra döneceğ *iz!* © OS, 31 Mayıs 1999

2 Örnek

Sürekli olarak ş ekil 13'e bakın! Dört noktalı satırların ASN.1 sözdiziminin bir parçası olmadığı ina dikkat edin - bunlar yalnızca, belirtimin o bölümünü tamamlamadıg im anlamına gelir.

```

Stok sipariş i ::= SIRALI {sipariş no
                                TAM SAYI,
                                ad-adres BranchIdentification, ayrıntılar
                                Dİ Zİ Sİ
                                SIRALI {çğ e
                                NESNE TANIMLAYICI,
                                vakalar
                                TAM SAYI},
                                aciliyet
                                SAYILANDIRILMIŞ
                                {yarın(0),
                                üç gün(1), hafta(2)}
                                VARSAYILAN hafta, doğ rulayıcı Güvenlik-
                                Türü}
                                ...
                                ...
                                ...
BranchIdentification ::= AYARLA
{unique-id NESNE TANIMLAYICI, ayrıntılar
SEÇİ M {uk
[0] Dİ Zİ {isim
                                GörünürDize,
                                Adresi yazın}, denizde
                                OutletType{OUTLET
                                {ad
                                UTF8Dizesi, tür
                                OutletType, yer
                                Adresi},
                                depo [2] SEÇİ M {kuzey [0] BOŞ ,
                                güney [1] NULL} } }
                                ...
                                ...
Güvenlik Tipi ::= SET { ...
                                ...
                                ...
}
Ş ekil 13: ASN.1'in (yapay) bir örneğ i

```

2.1 En üst düzey tip

Örnekte, okuyucuya "Stok için sipariş "in üst düzey tür, değer erleri soyut sözdizimini oluştururan tür, kodlandığından mesajları sağlayan tür olduğunu açıkça söyleyecek hiçbir şevey yoktur (ilk önce göründüğü dışında). Bu uygulama tarafından iletilir. Gerçek bir ASN.1 spesifikasiyonunda, bunu spesifikasiyonla ilişkili insan tarafından okunabilir metinden veya 1994 sonrası ASN.1'de bir ifade bularak keşfedebilirsiniz:

benim-soyut-sözdizimi ÖZET-SÖZDİ Zİ Mİ ::=
 {Stok için sipariş TANIMLAYAN
 {joint-iso-itu-t uluslararası organizasyon(23) set(42) set-vendors(9) wineco(43) abstract-syntax (1)}}

Bu basitçe, soyut sözdizimini "benim-soyut-sözdizimi" olarak adlandırdığımızı, "Stok için sipariş " türündeki tüm değerlerden oluşan tughunu ve bu soyut sözdizimini bir örnekte tanımlamanın gerekli olup olmadığıını söyleyelim.

Bu, ASN.1'in "NESNE TANIMLAYICI değil eri" (ASN.1 spesifikasiyonlarında sıkılıkla bulacağınız) olarak adlandırılan bir parçasıyla ilk karşılaşılmazdır. Bu üçüncü satırın tamamı aslında bir sayı dizisi yazmaya eşdeğerdir:

{2 23 42 9 43 1}

Ancak şimdilik NESNE TANIMLAYICI değil erini yok sayalım ve şekil 13'teki ana örneğe geri dönelim.

2.2 Önemli olan cesur!

Kalın yazılmış kısımlar ASN.1 dilinin kalbidir. Bunlar ayrılmış sözcüklerdir (esas olarak hepsinin büyük harf olduğunu unutmayın - ASN.1'de büyük harf önemlidir) ve yerleşik türler veya yapı mekanizmalarına başlıyor. Daha sonraki bir bölüm, her yerleşik tip ve yapı mekanizmasını ele alıyor!

2.3 İtalik isimler bir şeyle birbirine bağlamak için kullanılır.

İtalik kısımlar, yazarın uygulamanın türlerini adlandırmak için özürde seçtiği adlardır. Genellikle bir insan okuyucuya, bu türün taşıması amaçlanan bilgi türü hakkında iyi bir ipucu taşırlar, ancak bir bilgisayar için tek amaçları, spesifikasiyonun farklı bümelerini birbirine bağlamaktır.

Örneğin, "Stok için Sipariş "in üçüncü satırında "BranchIdentification" tip-referans-adına sahibiz. Bu, ancak ve ancak şartnamede başka bir yerde (bu durumda daha aşağıda, ancak önce olabilirdi) tam olarak bir "tip ataması" varsa yasaldır.

Üst düzey tip

Tüm uygulama belirtimleri, o uygulama için mesajları tanımlayan (tek) bir ASN.1 türü içerir. Genellikle (ancak zorunlu değil) şartnamede ilk sırada yer alır ve okumaya başlamak için iyi bir yerdir!

Bir spesifikasiyonda bulunan isimlerin çoğu veya

- yerleşik türlerin veya diğer yerleşik anahtar sözcüklerin adları (genellikle tamamı büyük harf) veya - tür referans adları (karışık durum, büyükten başlayarak) veya
- karmaşık türlerdeki öğelerin veya alternatiflerin adları (karışık durum, düşükten başlayarak) veya - (daha az görülen) değil er-referans adları (karmaşık durum, düşükten başlayarak) veya
- numaralandırma adları (alttan başlayarak karışık durum).

"BranchIdentification" iç in yazın . Bir bilgisayar söz konusu olduğ unda, aş ağı ıdaki metnin tamamı

Şube Kimliği ::=

"SET" ile başlayarak ve "SET" ile eşleşen kapanış kıvrık parantezine kadar, göründüğü her yerde "BranchIdentification" tür-referans-adi metinsel olarak değil iş tirmek iç in kullanılabilir . Ortaya çıkan ASN.1 değil iş meyecektir. Elbette, birçok farklı yerde "BranchIdentification" a atıfta bulunulursa, ilişkili türdeki metnin birden çok kopyasına sahip oluruz, bu da hataya çok açık olur ve belirtimin okunmasını zorlaştırır, bu nedenle tür referansı- bu gibi durumlarda isimler "iyi bir şe" dir. Ancak bu, daha sonraki bir bölümde ele alınacak olan bir tarz meselesidir.

2.4 Normal yazı tipindeki adlar, alanların/ş egerin/ş egerin adlarıdır

Normal yazı tipindeki adlar yine uygulama tasarımcısı tarafından keyfi olarak seçilir ve yine bilgisayarla ilgisizdir, ancak bir insan okuyucunun belirtimi anlamasına yardımcı olur. Ayrıca, spesifikasyonun karşılık gelen kısmıyla ilişkili semantiği açıkça belirtmek iç in insan tarafından okunabilir metin iç in bir "tutma yeri" sağlarlar.

Başlangıçta, normal yazı tipi sözcüklerini bir kayıt yapısındaki alanların adları olarak düşünen yararlı olabilir; aş ağı ıdaki kalın veya italik sözcük o alanın türünü verir. Doğru ASN.1 terminolojisi, normal yazı tipi sözcüklerinin aş ağı ıdakilerden biri olduğunu söylemektedir:

bir dizinin elemanlarını adlandırma,

bir kümenin elemanlarını adlandırma,

bir seçenekimin alternatiflerini adlandırmak veya

(yalnızca bir durumda) adlandırma numaralandırmaları.

ASN.1 spesifikasyonunu bir programlama dilindeki bir veri yapısı tanımına eşlemeden iç in bir ASN.1 aracı kullanılırsa, bu normal yazı tipi adları, seçilen dildeki tanımlayıcılara eşlenir ve uygulama kodu, bu adları kullanarak veri yapısının karşılık gelen bümeleri.

Uyarı okuyucusu - yine! - hemen bu adların uzunluğuunu ve bunlara izin verilen karakterleri merak edecek ve belirli bir programlama diliyle eşleme yaparken ilgili sorunları soracaktır. Bunlar iyi sorular, ancak tüm ASN.1 adlarının isteğiyle olarak uzun olabileceğini ve yalnızca yüzüncü karakterleri, hatta bininci karakterleri (veya daha sonraları) farklı olsalar bile farklı olduklarını söylemek dışında şimdilik göz ardı edilecekler! ASN.1 belirtimlerinde oldukça uzun adlar oldukça yaygındır.

2.5 Örneğe geri döner!

Öyleyse "Stok iç in sipariş" türündeki bir deger , satırda gönderildiğiinde hangi bilgileri taşıır?

"Stok iç in sipariş" , bir dizi alan veya "ş e" içeren bir yapıdır (değerleri verilen sırada satırda gönderilecek olan türlerin sıralı bir listesi). İlk alan veya eleman denir

"order-no" ve bir tamsayı değil eri tutar. İlkinci, "isim-adres" olarak adlandırılır ve daha sonra tanımlanan oldukça karmaşık ikinci türdür ve birçok iç yapıya sahiptir. Bir sonraki üst düzey alan "ayrintılar" olarak adlandırılır ve aynı zamanda oldukça karmaşık ikinci yapılandırılmış alandır, ancak bu sefer tasarımcı, tamamen bir stil meselesi olarak, başka bir tür kullanmak yerine "in-line" türünü yazmayı seçmiş tir. tip referansı isim.

Bu alan bir "SEQUENCE OF", yani "SEQUENCE OF" (sıfır olabilir) ifadesini takip eden rastgele sayıda tekrardır. Minimum veya maksimum tekrar sayısını zorunlu kılan ASN.1 gösterimi vardır, ancak buna sıkılıkla rastlanmaz ve daha sonraya bırakılır.

Aşağıda, "öğe" adı verilen bir "NESNE TANIMLAYICISI" alanını ve "vakalar" adı verilen bir "TAMSAYI" alanını birbirine bağlı olan başka bir "SEKANS" yer almaktadır. (Unutmayın, şarap stokları - kasaları - sipariş ediyoruz!). Dolayısıyla, "ayrintıların" tamamı, bir çift öğe enin - bir nesne tanımlayıcı değil eri ve bir tamsayı değil eri - keyfi olarak birçok tekrarıdır.

Bu uygulama için soyut sözdiziminin tanımlanmasını tartıştığımızda zaten nesne tanımlayıcı değil erleriyle tanıştiniz. Nesne tanımlayıcıları, dünya çapında kesin adlardır. Herkes (oldukça!) nesne tanımlayıcı ad alanının bir kısmını kolayca alabilir ve bu tanımlayıcılar ASN.1 tabanlı uygulamalarda çok çeşiti nesneleri adlandırmak için sıkılıkla kullanılır. Bu örnekte, bir "ürün"ü tanımlamak için bu formun adlarını kullanırız (bu durumda, "ürün" muhtemelen bir stok öğesi - belirli bir şarabın tanımıdır). Daha sonra, uygulama tasarımcısının bir dal için "benzersiz bir kimlik" sağlamak üzere "BranchIdentification"da bu aynı formun kimliklerini kullanmayı seçtiğimizde görüyoruz.

"Ayrıntılar" üst düzey alanının ardından, "SAYILANDIRILMIŞ" yerleşik ikinci tip "urgency" adlı bir alanımız var. Bu tür adının kullanılması, numaralandırmalar için bir ad listesi (ürün olası değil erleri) tarafından takip edilmesini gerektirir. Çok u programlama dilinde olmasa da ASN.1'de, bu örnekte olduğu gibi, genellikle adın ardından yuvarlak parantez içinde bir sayı gelir. Bu numaraların 1994 yılına kadar mevcut olması gerekiyordu, ancak artık uygulama tasarımcısı isterse otomatik olarak atanabilir. Her numaralandırmayı tanımlamak için satır boyunca iletilen gerçek değil erleri sağlarlar, bu nedenle "aciliyet" "yarin teslim et" ise, bu alan konumunda satıldan aşağı gönderilen şevey bir sıfırdır. (ASN.1'in ilk belirtimlerinde tasarımcı tarafından atanacak numaraların gereklilik olmasının nedeni daha sonra tartışılacaktır, ancak temel olarak, sürüm 1 belirtiminde sürüm 2'de eklenmiş fazladan bir numaralandırma varsa, birlikte çalışma sorunlarından kaçınmaya çalışmakla ilgilidir - genişletilebilirlik Yeniden!)

Yine "urgency" alanı, programlama dili veri-yapı tanımında bulunmayan bir özelliği sahiptir. "VAR SAYILAN" anahtar kelimesini görüyoruz. Bunun Temel Kodlama Kuralları (BER - orijinal ASN.1 Kodlama Kuralları) için anlamı, gönderenin seçeneğ olarak, amaçlanan değil er "VAR SAYILAN" kelimesinden sonraki değil er ise bu alanın ilettilmesine gerek olmamasıdır - bu durumda "hafta". Bu, tek bir soyut değil erin karşı ilk gelen birden fazla bit modelinin olduğu bir örnektir - "VAR SAYILAN" bir değil erin kodlanması kodlanmayıcağıını seçmek kodlayıcıların bir seçeneğidir. Daha sonraki Paket Kodlama Kuralları için, değil er "hafta" ise kodlayıcının bu basit alanı atlaması gereklidir ve kod çözücü bu değil eri varsayar. ("Aciliyet" daha karmaşık ikinci veri türü olsaydı, durum biraz farklıydı, ancak bu, Bölüm III'ün konusu.)

Anahtar Sözcük VAR SAYILAN: Bir DİZİ veya KÜME öğesinin bir öğesi için, o öğe için bir değil er dahil edilmemiş seversayılmak bir varsayılan değil eri tanımlar.

Anahtar Sözcük İSTEÇ BAĞLI: Bir değil eri atlanabileceğinin bir öğeyi tanımlar. Atlama, öğe enin herhangi bir normal değil erinden farklı anımlar taşıır.

"VAR SAYILAN"a benzer başka bir ASN.1 anahtar sözcüğü vardır, yani "İSTEÇ BAĞLI"dır (şekil 13'teki örneğe dahil değil ildir). Yine, anlam oldukça açık: alan atlanabilir, ancak orada © OS, 31 Mayıs 1999

herhangi bir varsayılan dē erin varsayımları dē ildir. Anahtar kelime, önē in adı "ek bilgi" olan bir alan/ȫ e ile ilişkilendirilebilir.

"Türdeki kesin soyut dē erler kümesi nedir?" sorusuna kısaca dönecek olursak, DEFAULT'un varlığı inin soyut dē erlerin sayısını dē iş tırmadī i, yalnızca kodlama seçeneklerini etkiledī i, ancak İ̄ STĒE BĀLI soyut dē erlerin sayısını artırır - istē e bāglı alanı olmayan soyut bir dē er, bir miktar dē erle mevcut olduğ u herhangi bir soyut dē erden farklıdır ve onunla ilişkili farklı uygulama anımlarına sahip olabilir.

Son olarak, "Order-for-stock"ta, son eleman "authenticator" olarak adlandırılır ve uygulama tasarımcısı tarafından "Sipariş - stok için". § ekil 13'te içerī i önēkte belirtilmeyen bir "SET" olarak gösterilmiş tir (gerçek bir spesifikasīyonda, elbette "SET" içerī i tam olarak tanımlanmış olacaktır). "SET", "SEQUENCE" ile çok benzerdir. BER'de (orijinal ASN.1 kodlama kuralları), yine bir gönderenler (kodlayıcılar) seçenekleri işaret eder. SET'in üst düzey ȫleri (alanları), metinde verilen sırayla (SEQUENCE iç in olduğu gibi) iletilmek yerine, gönderen/kodlayıcı iç in uygun olan herhangi bir sırada iletilir. Bugün, kodlayıcı seçeneklerin hem güvenlik nedenleriyle hem de alıcılara yükledikleri ekstra maliyet ve özellikle ayrıntılı testler nedeniyle "KÖTÜŞ EY" olduğu kabul edilmektedir ve "SET" (ve karşılık gelen "SET OF") asla uygulama tasarımcıları tarafından kullanılmamalı ve ASN.1'den çıkarılmalıdır! Ama lütfen tekrar § ekil 999'a bakın!

§ ekil 13, spesifikasīyonda daha sonra tanımlanan "Güvenlik Tipi"ni göstermektedir, ancak aslında, bu tam olarak, bir uygulama tasarımcısı tarafından tipleri (ve bunların semantik) güvenlik özelliklerini desteklemek iç in tasarlanmış tir.

ASN.1'de (daha sonra ele alınacaktır) bir tasarımcının dīg er spesifikasīyonlarda görünen tanımlara baş vurmasını sağlayan mekanizmalar vardır ve bu mekanizmalar sıkılıkla kullanılır. Bununla birlikte, bazı uygulama tasarımcılarının, kısmen bir uygulayıcının ek metinler almasına (belki satın almasına!) gerek kalmadan kendi metinlerini tamamlamak iç in, kısmen de üzerinde kontrol ve "sahiplik" sağlamak iç in dīg er spesifikasīyonlardan tanımları kopyaladıklarını da göreceksiniz. tanım. Bu kitabı bir meslektaşınızla birlikte veya bir kursun parçası olarak kullanıyorsanız, bunu yapmanın iyi bir şeyle olup olmadığı konusunda ilginç bir tartışma yaşayabilirsiniz!

2.6 BranchIdentification tipi

Şimdi, ASN.1 gösteriminin birkaç ek özelliği iç in gösteren "BranchIdentification" tipine kısaca bir göz atalım. (Şimdilik, lütfen bu tanımdaki köşeli parantez iç indeki sayıları tamamen göz ardı edin. Bunlar "etiketler" olarak adlandırılır ve bu bölümün sonunda ele alınacaktır.)

Bu kez bir "SET" olarak tanımlandı, bu nedenle BER'de ȫler herhangi bir sırayla iletilir, ancak biz bunları metinsel sırayla alacağımız.

Bir kenara (ama önemli bir kenara), B'düm 1'de BER'in tüm ȫler iç in bir TLV tipi kodlama kullandığı inanın bahsetmiş tı. Açıkcısı, ēg er gönderici bir "SET" in ȫlerini herhangi bir sırada iletebiliyorsa, her bir ȫ enin TLV'sindeki "T" iç in kullanılan dē erin farklı olması gereklidir. (Varlığı veya yokluğu tespit edilmesi gereken İ̄ STĒE BĀLI veya VARSAYILAN ȫler olmadıkça, bu Dİ̄ Zİ̄ iç in gereklilī olmayacağı). Aşağıda kısaca tanıtılan ve daha sonra daha ayrıntılı olarak ele alınan "etiket" kavramının ortaya çıkışına neden olan bu gerekliliklerdir.

Listelenen ilk **öğe**, daha önce tartışılan bir "OBJECT TANIMLAYICI" değil eri olan "unique-id" dir. Diğer er tek unsur "detaylar" dir. "Ayrıntılar" adının "Stok İçin Sipariş" te kullanıldığına dikkat edin. Bu oldukça normal ve tamamen yasaldır - bağlamlar farklıdır.

Uygulama tasarımcılarının bir SEQUENCE veya SET'teki üst düzey **öğeler** içinden farklı adlar kullanması normaldir, ancak bu aslında 1994'ten önce bir gereklilik değil idi.

Artık hem

"SEQUENCE" hem de "SET" **öğeleri** için farklı adlara sahip olmak bir gereklilik (ve bir "SEQUENCE" in alternatifleri için - aşağıya bakınız).

Gereksinim, kısmen mantıklı olduğunu için, ancak esas olarak, bir türün değil erleri için ASN.1 gösterimi, bu kural izlenmediği takdirde bazı durumlarda belirsiz olabileceğin için eklenmiş tir.

"Ayrıntılara" bakıldığıında: bu bir "SEQUENCE" dir, yani bu alan konumunda olan şayi bir dizi olası alternatiften biridir - bu durumda üç olasılık vardır: "İçiniltere", "denizaşırı" ve "depo" alternatifleri. (Yine, uyarı okuyucusu, BER'de kullanılan TLV yaklaşımıyla, alıcı/kod çözücünün hangisinin iletildiğini doğrulayın bir şekilde belirlemesi gerekiyorsa, bu alternatiflerin her birine atanmış "T"nin farklı olması gerekiyor.)

"İçiniltere" alternatif, üç **öğe** eden oluşum bir "SEKANS"tır: "ad", "tür" ve "konum".

Son iki **öğe**, italik yazılmış tür adlarına sahiptir ve bu nedenle muhtemelen oldukça karmaşık olacaktır ve şartnamede daha önce veya sonra tanımlanacaktır. Burada daha fazla tartışılmazlar. "ad" bir "VisibleString"dir. Bu, "karakter dizileri" olan ASN.1 türlerinin oldukça uzun bir listesinden (yaklaşık bir düzine) biridir - belirli bir karakter repertuarından karakter dizileri. Bu türlerin adlarının tümü büyük-küçük harflerle karışık olacaktır ve ASN.1'deki yerleşik türlerin (takvim tarih ve saatini taşıyan türler değil er ana istisnadır) birkaç istisnasından biridir (takvim tarih ve saatini taşıyan türler). Kullanıcı tarafından yeniden tanımlanabilecek her zaman tamamen büyük harflerle yazılır ("INTEGER", "BOOLEAN", vb. gibi).

"VisibleString" türündeki değil erler, yazdırılan ASCII karakterleri artı "boş luk" dizeleridir. Bu nedenle, Birleşik Krallık veya ABD adları için uygunlardır, ancak diğer Avrupa ülkeleriyle iyi başa çıkamazlar ve dünyanın diğer yerlerinden gelen adlarla çok kötü bir şekilde başa çıkmabilirler!

Buna karşılık, "denizaşırı" alternatifi için "ad" **öğe** esinin "UTF8String" türü vardır. Karakter kodlama şartnamelarıyla ilgileniyorsanız, UNICODE (ve/veya ISO 10646!) ve UTF8'i duymuşsunuzdur! Eğer değil ilseniz, alan daha sonra daha ayrıntılı olarak tartışılacaktır! "UTF8String" in dünyadaki herhangi bir dilden karakterler içerebileceğiini söylemek yeterlidir, ancak ilginç bir özellik ile, eğer karakterler sadece ASCII karakterleri yiyse, kodlama tam olarak ASCII'dir!

Karakterler için UTF8 kodlama şartnameyi nispeten yenidir ve yalnızca 1998'de ASN.1'e eklenmiş tir. Yasal olarak yalnızca uygulama tasarımcısı 1998 (veya üstü) ASN.1 belirtimine başvuruyorsa kullanılabilir.

ASN.1, saf ASCII metninden dünyadaki herhangi bir dilden karakterler içeren metne kadar destek sağlayan bir çok karakter dizisi türüne sahiptir.

- 1994'te bazı kısıtlamaların eklendiği ini zaten belirtmiş tık (a Ama **öğe** esinin **öğelerinin adları**). Örneğin "SEQUENCE", "SET" vb.'nin farklı olması gerekiyordu. (300 sayfa uzunluğunda!) belirtiminizi 1994 veya sonrasında uyacak şekilde yükseltme zahmetine katlanmadığınıza, ancak yine de UTF8String'in yeni bir sürümde kullanmak istediğinizizi varsayıyorum. Yasal olarak, yapamazsınız. ("Aa öyle mi?", "Hangi hükümet çıkardı o yasayı?", "Beni çiğ nersem hangi kolluk kuvveti beni cezalandırır?" diyorsunuz. Susuyorum!) Ama bir uygulayıcı/okuyucu olarak ve görüyorsanız oluyor, ne olduğunu bileyebilirsiniz © OS, 31 Mayıs 1999

anlamına geliyor! Tabii ki, bir uygulama tasarım ekibinin parçası olarak, bunun spesifikasyonlarınızda olmadığı İndan kesinlikle emin olursunuz, değil mi?

Şekil 13'e geri dön! "Ayrıntılar" daki üçüncü alternatif "depo" dur ve bu da baş ka bir "SEÇİ M" dir, yalnızca iki alternatif vardır - "kuzey" ve "güney", her biri "BOŞ" tipindedir.
 "BOŞ" nedir? "NULL" resmi olarak yalnızca tek bir değil ere sahip bir türdür (kendisi belki de kafa karışır tırıcı bir şeilde "NULL" olarak adlandırılır). Bir türde sahip olmamız gereken ancak eklenecek ek bilginin olmadığı yerlerde kullanılır. Bazen "yer tutucu" olarak adlandırılır. "Depo" durumunda, "kuzey" ve "güney" arasında karar vermek için bir BOOLEAN veya bir SAYILANDIRILMIŞ olabiliriz. Sadece bir stil meselesi olarak (ve "NULL" kullanımını göstermek için!) bunu bir NULL seçimi olarak yapmayı seçti.

2.7 Bu etiketler

Şimdi köşeli parantez içindeki sayıları - "etiketleri" tartış alım.
 1994 sonrası ASN.1'de bu sayıları dahil etmek hiç bir zaman gereklidir. 1994'ten önce gerekli olsaydı, (1994'ten sonra) otomatik olarak oluş turulmalarını isteyebilirsiniz (OTOMATİ K ETİ KETLEME olarak adlandırılır) ~~ve genellikle bir~~ zaman fiilen dahil etmenize

Bununla birlikte, mevcut yayınlanmış spesifikasyonlarda, etiketlerle sık sık karşılaşırsınız ve bunlar hakkında biraz bilgi sahibi olmanız gereklidir.

1994'ten sonra gerekli olmayan köşeli parantez içindeki Etiketler Numaraları, net kodlamalar sağlamak için ASN.1 tipi değil erlerin taşıyabileceğ i bilgileri etkilemezler.

En eski ASN.1 tabanlı uygulama spesifikasyonlarından bazlarında, etiketten sonra sık sık "IMPLICIT" anahtar sözcüğü ünү ve günümüzde nadiren de "EXPLICIT" anahtar sözcüğü ünү bulacaksınız. Bunlar, etiketin anlamını nitelendirir ve Bölüm 3'te tam olarak açıklanır.

Neden etiketlerimiz var? BER'in temel yapısını hatırlayın: bir "SEKANS" için, dizinin her elemanı için bir TLV vardır; bunlar, bir dış seviye TLV'nin "V" kısmını oluşturmak için üç uca yerleş tirilir. Varsayılan olarak, "INTEGER" veya "BOOLEAN" gibi herhangi bir temel ASN.1 türü için TLV'nin "T" kısmı, ASN.1 belirtiminde belirtilen bir değil ere sahiptir ve dış değil erin "T" kısmı, bir "SEQUENCE" için seviye TLV yine ASN.1 spesifikasyonunda belirtilen bir değil ere sahiptir.

Bu, varsayılan olarak, "kuzey" "BOŞ" ve "güney" "BOŞ" kodlamasının aynı olacağı anlamına gelir - alıcı/kod çözücü hangisinin gönderildiğiini bilmeyecektir. Kodlama, bir "SEÇİ M" in her alternatifinde her alternatif için kullanılan "T"nin farklı olması gerektiğin şeildeki gereklili ve açık kuralı ihlal etmiş tir. Etiketin amacı, varsayılan "T" değil erini etikette belirtilen bir değil erle geçersiz kılmaktır. Dolayısıyla, yazıldığı gibi örnekle, "kuzey" "T" sıfır içerir ve "güney" "T" bir içerir. Benzer şeilde, "uk" ve "denizaltı" "SEQUENCE" kodlamalarından en az biri için dış düzeyde "T" üzerindeki varsayılan etiketi geçersiz kılmak önemlidir. (Tarz olarak, ikisini de abartmayı seçti).

Daha sonraki bir bölüm, etiketlerin ne zaman eklenmesi gerektiğine inelik kin kuralları tam olarak açıklar. (1994 öncesi, şeilde 13, köşeli parantez içindeki sayıların en azından bir kısmı - etiketler olmadan yasa dışı olurdu). Kurallar, "belirsizlik" i önlemek için gereken minimum kurallardır ve bu anlaşılıktan sonra okuyucu, ayrıntılı kuralları yeterince kolay bir şeilde hatırlayabilecektir. Bununla birlikte, (normalde) varsayılan bir etiketi aşınmanın bir cezası yoktur ve bir tarz ve "bunu düşünme, sadece yap!" felsefede, kesinlikle gereklili olsun veya olmasın, her "SEÇİ M" yapısının öğelerinin her birine sırayla atanmış etiketleri (şeilde olduğu gibi) görmek oldukça yaygındır. Benzer şeilde (ancak şeilde yapılmamış tir)

13), "SEQUENCE" ve "SET" yapılarının tüm öğelerine sıralı etiket numaralarıyla uygulanan etiketleri görmek oldukça yaygındır (1994 öncesi).

Giriş niteliğinde son bir yorum: Yukarıdakiler, etiketlerin sadece düz eski sayılar olduğuunu ima etti. Aslında, bir TLV'nin "T" bölümünde kodlanan değil olan etiket ad alanı, bundan biraz daha karmaşık. Bazen "UYGULAMA" veya "ÖZEL" veya "EVRENSEL" anahtar sözcüklerini açılış köşeli parantezinden sonra bulacaksınız, örneğin:

Tagged-type ::= [UYGULAMA 1] Stok İçin Sipariş

Bu anahtar sözcükler, etiketin "sınıfını" tanımlar. Bunların yokluğuunda, "sınıf", uygulanan en yaygın etiket sınıfı olan "bağlama özel" olarak adlandırılır. Etiketlemenin tüm ayrıntıları Bölüm II, Bölüm 4'te yer almaktadır.

3 Farklı yazı tiplerinden kurtulmak

Tek bir yazı tipi kullanan normal bir ASN.1 tabanlı uygulama belirtiminiz olduğuunu varsayıyalım. Şekil 13'teki gibi yazı tiplerini nasıl uygularsınız?

İlk olarak, prensip olarak, karakter dizisinin adları ve tarih/saat türleri dahil olmak üzere dilde ayrılmış sözcüklerin neler olduğuunu bilmeniz ve bunların kalın olduğuundan emin olmanız gereklidir! Uygulamada, tamamı büyük harf olan herhangi bir adım kalın yazılmak üzere konusunda iyi bir tahminde bulunabilirsiniz, ancak bu bir gereklilik değil. Şekil 4'teki "Adres" tipi-referans-adı "ADRES" olabilir ve şartnamede her yerde değil iş iklik yapılması şartıyla sonuç aynı ve tamamen yasal bir şartnameidir. Ancak bir stil meselesi olarak, tür referans adları için tamamı büyük harf nadiren kullanılır.

İkinci olarak ayarladığınız ilk büyük harfle başlayan diğer adlar - bu bir tür referans adıdır. Tip-referans-adlarının büyük harfle başlaması gereklidir. Bundan sonra birbirinin yerine büyük veya küçük harf içerebilirler.

Şekil 13'te iki farklı stilin bir karışımı göreceksiniz. Bir durumda, üç kelimedenden oluşan bir tip-referans-adı ("Stok için sipariş"), kelimeleri bir tire ile ayırmak gereklidir. Başka bir durumda, bir tür-referans-adı ("OutletType"), sözcükleri ayırmak için başka bir büyük harf kullanır ve kısa çizgi kullanmaz. "Güvenlik Tipi" her ikisini de kullanır!

Normalde tek bir spesifikasyonda bu üç stilin bir karışımı görmeyiniz, ancak hepsi tamamen yasaldır. İlk onaylanan ASN.1 belirtiminden itibaren adlarda kısa çizgilere (ancak bitişik konumlarda iki değil il, yorumla ilgili belirsizlik için - aksa ağıyla bakın) izin verilmişdir, ancak bu ilk onaylanan belirtimden önceki taslaklarda izin verilmemiştir. Bu nedenle ilk yazarlar seçim yok ve "OutletType" stilini kullandı. Tabii ki, hiç kimse ASN.1 spesifikasyonunu okumaz - sadece herkesin yaptığıını kopyalarlar! Yani bu tarz bugün hala en yaygın olanıdır. Bununla birlikte, sadece bir tarz meselesi ve bu konuda öneksiz - her üç form da yasaldır ve daha düzenli veya daha net göründüğünü düşündürünüz ki isel bir tercihtir.

Ve son olarak, normal yazı tipi: küçük harfle başlayan adların çoğu, öğelerin veya alternatiflerin adlarıdır ("sipariş no", "aciliyet", vb.) ve yine bu tür adların küçük harfle başlaması gereklidir. Harf, ancak daha sonra büyük veya küçük harf içerebilir.

Değil er adları için küçük harfle başlayan adlar da gereklidir. Basit bir örnek, "aciliyet" için "hafta" değil eridir.

Uygulama belirtimleri, ş ekil 13'te görünenler gibi (ve genellikle çoğu uygulama belirtiminin büyük bölümünü oluştur) tip atama bildirimlerini içermekle kalmaz, aynı zamanda "değ er-referans-adalarına" değ erler atayan ifadeler de içerebilir. Bir değ er referans atamasının genel biçimi aşağıda gösterilmektedir:

varsayılan-durumlarım INTEGER ::= 20

"20" tamsayı değ erine baş vurmak için "INTEGER" türünde değ er-referans-adı "varsayılan-durumlarım"ı tanımlayan . Daha sonra ş ekil 13'teki "durumlar" ög esinde ş u ş ekilde kullanılabilir:

INTEGER DEFAULT my-default-case'ler

4 Bazı kayıp iş leri birleş tirmek

4.1 Tip ve değ er atamalarının özetİ

İlk olarak, ş imdiye kadar gördüklerimizi özetleyelim. ASN.1, bir ASN.1 tipini tanımlayan bir dizi gösterim parçasını (tip gösterimi) belirtir. Bazları "BOOLEAN" gibi çok basittir, diğ erleri numaralandırılmış bir türü veya bir dizi türünü tanımlamak için kullanılanlar gibi daha karmaş iktir. Tip-referans-adı aynı zamanda, ASN.1'in bir tip-notasyonu gerektirdiği her yerde kullanılabilecek bir tip-notasyonu parçasıdır.

Bir uygulama belirtimi, çok sayıda tip atama deyimi ve ara sıra (ancak nadiren) bazı değ er atama deyimleri iç erir.

Benzer ş eklide ASN.1, bir dizi değ er gösterimi belirtir (ASN.1 ile yazabileceğiniz herhangi bir tür, tüm değ erleri için tanımlanmış bir değ er gösterimine sahiptir). Yine, tamsayı değ erler için "20" gibi değ erler için bazı gösterimler çok basittir, diğ erleri daha karmaş iktir, örneğ in bu bölümün başında gördüğünüz nesne tanımlayıcı değ erleri veya dizi türlerinin değ erleri için gösterim . Yine, ASN.1'in değ er gösterimi gerektirdiği her yerde, bir değ er referans adı kullanılabilir (bir yerde bir değ er atanmış olması koşuluyla).

Tip atamasının genel biçimi ş öyledir:

tip-referans-adı ::= tip gösterimi

ve bir değ er ataması:

değ er-referans-adı tip gösterimi ::= değ er gösterimi

burada değ er gösterimi, tip gösterimi tarafından tanımlanan tip için "doğru" değ er gösterimi olmalıdır. Bu önemli bir kavramdır. ASN.1'de tip gösterimini kullanabileceğiniz herhangi bir yerde (örneğ in, bir "SET" veya "SEQUENCE" ög esinin türünü tanımlamak için, herhangi bir yasal tip gösterimini kullanabilirsiniz. Ancak, değ er gösterimine izin verilen yerlerde (örneğ in, değ er atamalarında veya VARSAYILAN'dan sonra), her zaman, değ er gösteriminin sözdizimini tanımlanan tip için izin verilenle sınırlayan, vali adı verilen (bir tip referans adı olabilir) karşılık gelen bir tip gösterimi vardır. tip notasyonuna göre.

Şimdiye kadar, bölümün başındaki "TANIMLAYAN" bölümünde ve DEFAULT sözcüğüinden sonra kullanılan değer er gösterimini gördünüz. Daha sonra açıklanacak başka kullanımlar da vardır, ancak değer er notasyonu tip notasyonundan çok daha az sıkılıkla kullanılır.

4.2 İ simlerin biçimleri

ASN.1'deki tüm adlar karışık büyük/küçük harfler ve rakamlar ve kısa çizgilerdir (ancak yorumla karıştırılmaması için iki birleşik veya bir sonda değil), bağılı olarak büyük harfle veya küçük harfle başlar. adının ne için kullanıldığıına bağlıdır. (Şimdiye kadar tahmin etmişsinizdir, boş luk karakteri içeremezler!) ASN.1'de her isimlendirmede ilk harfin durumu sabittir. Büyük harf yasalsa, küçük harf geçerli olmayacağı ve bunun tersi de geçerlidir. İ simler keyfi olarak uzun olabilir ve ismin herhangi bir konumunda içerik veya durum bakımından farklılık gösterirlerse farklı isimlerdir.

Adlar yalnızca harf, rakam ve kısa çizgi içerebileceğiinden, ardından başka herhangi bir karakterin (kıvrık ayraç veya virgül gibi) geldiği bir adın yanında boş luk veya yeni satır olmadan aşağıdaki idaki karakter bulunabilir. veya tamamen kişi isel bir tarz olarak, bir veya daha fazla boş luk veya yeni satır eklenebilir.

4.3 Düzen ve yorum

Düzen "serbest biçimlidir" - boş luk koyabileceğiniz her yere yeni bir satır koyabilirsiniz. Yeni satırınız olan her yerde onu kaldırabilir ve bir boş luk bırakabilirsiniz. Böylece, eksiksiz bir uygulama belirtimi tek bir metin satırı olarak görünebilir ve geri ekten de temelde bir bilgisayar onu böyle görür!

Tarz olarak, herkes her tür veya değer er atama ifadesi arasına ve genellikle bir kümenin veya dizinin her bir öğesi ile bir seçimin alternatifleri arasına yeni bir satır koyar. Şekil 13'te gösterilen yerleşim stilini, süslü parantezlerin eşleşmesini çok açık hale getirdiğinden bu yazar tarafından tercih edilen stilidir, ancak belki biraz daha yaygın olan bir yerleşim stilini, "SEQUENCE" ile aynı satırda "SEQUENCE"tan sonra açılan süslü parantezi dahil etmektedir. anahtar kelime "SEKANS", örneğin:

```
Dİ Zİ { ögeler
    NESNE TANIMLAYICI, durumlar TAM SAYI }
```

Yine de diğer yazarlar (daha az yaygın), kapanış kıvrık parantezini kendi başına bir çizgiye koyar ve eşleşen açılış dırseğiyle dikey olarak hizalar. Tamamen saf (ve tamamen öneksiz) üslup meseleleri.

Biraz daha ciddi bir şekilde, "CHOICE" tipi için 1994 öncesi değer er gösterimi vardı. Aşağıda izin veren "BranchIdentification":

ayrintilar ambar kuzey değer eri-ref

bir değer er gösterimi parçası olarak ("değer er-baş vuru", "NULL" değer eri için bir değer er referans adıdır). ASN.1'in, adların bir tür veya değer er atamasında atanmadan önce kullanılmasına izin verdiğiini unutmayın ve zayıf aptal bir bilgisayar, belirtimin başlangıcında şuna benzer bir şeyle karşılaşabilir:

Adlar ve düzen Adlar

harf, rakam veya kısa çizgi iç erir. İsteğe bağlı olarak uzunlar. Durum önemlidir. Düzen serbest biçimdir. Açıklama, bir çift bitiş ikinci çizgi ile başlar ve bir çift bitiş ikinci çizgi veya bir yeni satır ile biter.

joe Fred ::= jack jill joseph Mary ::= vb vs

Bu durumda, ilk atamanın nerede biteceğ ini belirleyemez - "kriko"dan sonra mı yoksa "jill"den sonra mı yoksa "joseph"ten sonra mı - daha sonra tanımlanan gerçek "Fred" türüne bağlıdır. Bu, bir bilgisayara zor anlar yaşatabilir! İlk araç satıcılarından bazıları bununla başa çıkmadı (muhtemelen gerçekte hiç gerçek ekleşmemiş olsa bile!) ve "noktalı virgül" karakterinin ASN.1'de bir ifade ayırcı olarak kullanılmasını istedi. Bugüne kadar, bu araçları kullanırsanız, tüm yazım atamalarınızın arasına noktalı virgül koyma gerekecektir. ("OSS ASN.1 Tools" paketi bu gereklilik i getirmez). ASN.1 spesifikasyonlarına noktalı virgül ekleme gereklilik ine karşı çıksamı, ancak araç satıcılarına yardımcı olmak için "CHOICE" de değil er notasyonuna bir "iki nokta üst üste" eklendi, böylece 1994 sonrası yukarıdaki değil er notasyonu yazılmacaktı:

detaylar : depo : kuzey : değil er-ref

(Boş luklu veya boş luksuz, ancak iki nokta üst üste.) Ve (örneğin):

joe Fred ::= jack : jill joseph Mary ::= vb vs

"jill" den sonraki ilk atamanın sonuna sahipken:

joe Fred ::= jack : jill: joseph Mary ::= vb vs

"joseph" ten sonraki ilk atamanın sonuna sahiptir. Bu, 1994 spesifikasyonunun 1994 öncesinde mevcut olmayan ek gereklilikler getirdiği başka bir küçük alandır.

Boş luktara ve yeni satırlara izin verilen her yere yorum eklenebilir. Yorum bir çift tire ile başlar (arasında boş luk olmadan) ve ilk yeni satırda veya başka bir kısa çizgi çifti ile biter. ("Yeni satır"ın değil er boş luk biçimlerinden farklı olduğu tek durum budur.)

Bu tamamen iyi ve tutarlı bir kuraldır, ancak belirli bir iyi bilinen programlama dili için kullanılanla tamamen aynı değildir, bu yüzden dikkatli olun! Birkaç satırda yayılmış bir yorum blogu istiyorsanız, her satırın başında bir çift kısa çizgiye ihtiyacınız vardır.

5 Peki başka ne bilmeniz gerekiyor?

Gerçekten, artık oldukça iyi bir şekilde uzaklaşıp ASN.1 spesifikasyonlarını okuyabilirsiniz! Ancak bu metni elde etme zahmetine katlandığınız için (belki parasını bile ödediğiniz için!), biraz daha devam etmesini bekleyeceksiniz.

Sonraki birkaç bölümde, ASN.1 tabanlı bir uygulama belirtiminin dış düzey yapısına bakacağınız ve çeşitli itli yerleşik türleri ve yapı mekanizmalarını ("SEQUENCE" gibi) ve ilişkili de değil er gösterimlerini inceleyeceğiz. Bu metin sıkıcı! Hızlı bir şekilde okumanız gerekecek!

Bu, okumanız gereken her şeyi tamamlayacaktır. 1994'ten önce üretilen ASN.1'in çoğu ve, Bölüm II'ye bırakılan alt tipleme ve "delikler" için mekanizmalar gibi daha az yaygın olarak kullanılan "gelişmiş" özellikler dışındadır. Bölüm II ayrıca 1994'te tanıtılan "yeni" özelliklerin tartışmalarının çoğuunu içerir ve uygulama belirtimlerini yazmaya dahil olan herkes için önemli bir okumadır.

Bölüm I, "ASN.1 derleyicileri" kullanılarak uygulamaların nasıl üretileceği ile dair daha ayrıntılı bir tartışma ve uygulamaya ilişkin bazı başka yönergelerle sona ermektedir.

Bölüm 3 Bir ASN.1 spesifikasyonunun yapılandırılması

(Veya: Duvarlar, zeminler, kapı girişleri ve
asansörler, evreyle ilgili bazı hususlar dikkate alınarak!)

Özet:

ASN.1 tabanlı uygulama belirtimleri, Bölüm 1 Bölüm 2'de gösterildiği gibi temel olarak tip tanımlarından oluşur, ancak bunlar normalde (ve resmi olarak böyle olması gereklidir) modül adı verilen koleksiyonlar halinde grupperler.

Bu bölüm:

modül yapısını tanıtır,

modül başlıklarının biçimini açıklar,

modüllerin nasıl tanımlanacağıını gösterir,

tip tanımlarının modüller arasında nasıl dışa ve içe aktarılacağıını açıklar.

Bölüm ayrıca şunları tartışır:

tam bir uygulama belirtimi için bazı yayın formatı sorunları ve

ASN.1 parçalarının makine tarafından okunabilir kopyasını kullanıma sunmanın önemi.

Bir modülün tanımının bir kısmı, ağ ıdakilerin oluşturulmasıdır:

etiketleme ortamı,

geniş letilebilirlik ortamı

bu spesifikasyonda görünen tip notasyonları için. Bu terimlerin anlamı ve önemi bu bölümde tartışılır ve son ayrıntıları Bölüm II'de verilmektedir.

1 Bir örnek

Ş ekil 13'te verdiği imiz örnekte bir üst düzey tip ("Stok için sipariş ") ve çoğunu eksik bıraktığımız bir dizi destekleyici tip vardı. Destekleyici türleri yine de eksik bırakacağınız (ve aslında, tekrarı önlemek için tüm türlerin gödesi için dört noktadan oluşan üç satır kullanacağınız), ancak şimdiden Ş ekil'deki örneğin tersine evireceğiniz.

Modüller

Tüm ASN.1 tip ve değerler atamalarının, bir modül başlığıyla başlayan ve "END" ile biten bir modül içinde görünmesi gereklidir.

Wineco sipariş protokolü {joint-iso-itu-t internationalRA(23) set(42) set-vendors(9) wineco(43) modülleri(2) sipariş (1)}

TANIMLAR

OTOMATİK ETİ KETLER ::=

BAŞ LAMAK

Stok sipariş i ::= SIRALI {

....
....}

BranchIdentification ::= SET {

....
....}

Güvenlik Tipi ::= SET {

....
....}

Çıkış Tipi ::= SIRALI {

....
....}

Adres ::= SIRALI {

....
....}

SON

Ş ekil 14: Eksiksiz bir tek modül ASN.1 belirtimi

13'ü, dilin kurallarına uyan ve bir ASN.1 derleyici aracına beslenebilen eksiksiz bir ASN.1 spesifikasyonuna dönüşür.

NOT — Ş ekil 13 ve 14'te kullanılan dört noktalı üç çizgi kullanımı yasal değil ASN.1! Bu kitapta tamamen tembellikten kullanılmıştır! Gerçek bir belirtimde, adlandırılmalı ve tam olarak belirtilmiş (doğrudan veya tür-referans-adalarına göre) öğelerin tam bir listesi olacaktır. Ş ekil 14'te, bu türlerin gödesinde başka tür referans adalarının kullanılmadığı varsayılmıştır - bunlar yalnızca INTEGER, BOOLEAN, VisibleString, vb. dilin yerleşik türlerini kullanır.

Tüm özellikler ş ekil 14'te gösterilmektedir.

Bu örnek, altı satırlık (bu - basit! - durumda) modül baş liğ indan, bir dizi tip (veya dē er) atama deyiminden ve bir "END" ifadesinden oluş an ASN.1 modülünü oluş turur.

Bu, ASN.1 spesifikasiyonunun en küçük yasal parçasıdır ve ilk spesifikasiyonların ç oğ u bu formdaydı - tek bir modül. Bugün, karmaş ık bir protokolün birkaç ASN.1 modülünde (genellikle tek bir fiziksel yayında veya Web sayfaları kümelerinde) sunulması daha yaygındır.

Bu daha sonra tartış ılacaktır.

Gerçek bir yayında, modül baş liğ inin bir sayfanın baş ında görünmesi çok yaygındır, çünkü o zaman on veya daha fazla sayfaya kadar tip atamaları (belki ara sıra dē er atamasıyla) ve ardından END ifadesi, bu da modülü sonlandırır. Normalde, baş ka bir modül tarafından takip edilse de, basılı bir belirtimde END ifadesinden sonra bir sayfa sonu olacaktır.

ya da dē il.

Ancak Ş ekil 14, toplam protokol belirtiminin ASN.1'in muhtemelen yalnızca birkaç sayfası olduğunu ve tüm belirtim için tek bir bağı ımsız modülün kullanıldığı ilk ASN.1 belirtimlerinin tipik bir örneğ idir.

Bu örneğ in baş ındaki yeni satırların ve girintilerin kullanımı yaygın olarak kullanılan ş ey olsa da, boş luk ve yeni satırların birbirinin yerine geçebileceğ ini belirten normal ASN.1 kuralı burada da geçerlidir - modül baş liğ i bir tek çizgi.

Modül baş liğ inin farklı işe elerine bu bölümde daha sonra ayrıntılı olarak bakacağ iz, ancak önce yayın stili hakkında biraz daha tartış acağ iz.

2 ASN.1 spesifikasiyonları iç in yayın stili

Yıllar boyunca, farklı gruplar, yayınlanmış belgelerde ASN.1 spesifikasiyonlarının sunumuna farklı yaklaş ımlar benimsemiş tir. Sorunlar ve çeşitli itlilik, çeliş en arzulardan kaynakları:

- a) Toplam belirtimi oluş turan çeşitli itli ASN.1 türlerini, farklı türlerin ve alanların anlamlarını açıklayan, insan tarafından okunabilen normal metin içinde kademeli olarak (genellikle "aş a ğ ıdan yukarıya" bir tarzda) tanıtmaya isteğ i.
- b) Spesifikasiyonda, ASN.1 sözdizimine uyan ve tip-referans-adı alfabetik sırasına göre tip tanımları ile bir ASN.1 aracına beslenmeye hazır olan eksiksiz bir ASN.1 parçasına sahip olma isteğ i veya "yukarıdan aş a ğ ıya" sırayla.
- c) İş stenmeyen farklılıklardan kaçınmak için metni tekrar etmemeye isteğ i ve nihai ürünlerde farklılıklar kalırsa hangi metnin öncelikli olduğunu soruları.

Tek bir mükemmel yaklaş ım yoktur - uygulama tasarımcıları bu alanlarda kendi kararlarını vermelidir, ancak aş a ğ ıdaki iki alt bölümde bazı ortak yaklaş ımlar tartış ılmaktadır.

Referanslara ve çapraz referanslara yardımcı olması için ASN.1'inize satır numaraları eklemeyi düş ünebilirsiniz... ancak bunlar dilin bir parçası dē ildir!

2.1 Satır numaralarının kullanımı.

Bir yaklaşım, ş ekil 15'te kısmen gösterildiği gibi, tüm ASN.1 spesifikasyonuna sırayla satır numaraları vermektedir (yine, spesifikasyonun dışarıda bırakılan parçalarını belirtmek için dört noktalı çizgiler kullanılır).

```
001 Wineco sipariş protokolü 002 { ortak iso-
itu-t uluslararası RA(23) set(42) set-vendors(9) 003 wineco(43) modülleri(2) sipariş (1)}
```

```
004 TANIMLAR 005
```

```
OTOMATİK ETİ KETLER ::=
```

```
006 BAŞ LANGİÇ
```

```
007 008 009 010
```

```
Stok sipariş i ::= SIRALI {sipariş no
```

```
INTEGER, ad-
```

```
adres BranchIdentification,
```

```
....
```

```
....
```

```
159             özet OCTET STRING}
```

```
160
```

```
161 SON
```

Ş ekil 15: Satır numaralı modül

Bu belirtim bir ASN.1 aracına beslenirse, satır numaralarının kaldırılması gerektiğiine dikkat etmek önemlidir - bunlar ASN.1 sözdiziminin parçası değildir ve yazar, göz ardı edilecek bir yönerge sağlayan hiçbir araç bilmez. onlara!

Bunu üretmeye yardımcı olacak araçlarınız varsa (ve bunlar varsa), bu satır numaralı yaklaşım ayrıca, her tip referans adı için satır numarasını veren belirtimin sonunda bir çapraz referans sağlama mümkün kılın. bir tipin verildiği tip ataması, ardından bu referansın kullanıldığı tüm satır numaraları. Geniş bir spesifikasyon için, bu yaklaşım okuyucular için ÇOK faydalıdır. Bunu yapmazsanız, tanımlarınızı alfabetik sıraya göre yeniden sıralamak isteyebilirsiniz.

Satır numaralarını kullanmaya karar verdığınızda, iki ana olasılık vardır. Yapabilirsiniz:

ASN.1'i yalnızca tam bir belirtim olarak (genellikle sonunda) tek bir yere koyn ve anlambilimi belirten, insan tarafından okunabilen normal metin içinden ASN.1 metnine baş vurmak için satır numaralarını kullanın.

Satır numaralı ASN.1'i bir dizi "ş ekil"e ayıran ve bunları, daha spesifik referanslar için yine satır numaralarını kullanarak, insanlar tarafından okunabilir metinde uygun yere yerles tirin.

İkinci yaklaşım, yalnızca (toplam belirtimde) tür tanımlarına sahip olduğunuz sıra, bunları ana metinde tanıtmak ve tartışmak istediğiniz sırayla aynıysa iyi çalışır.

2.2 ASN.1 metninin çoğ altılması

ASN.1 metnini çoğ altmak için bir dizi spesifikasyon seçilmiş tir (mutlaka olmasa da genellikle satır numaraları kullanılmadan). Bu durumda tipler, insan tarafından okunabilir metin içine yerleş tırılmış ASN.1 parçalarıyla tanıtılır ve modül baş lig i ve "SON" ile tam modül belirtimi, belgenin son maddesi olarak veya bir Ek'te sunulur. .

ASN.1 metnizi, belirtiminizin gödesinde parçalanmış olarak ve bir ekte tamamlayarak tekrar etmeyi seçebilirsiniz - ancak metinlerin aynı olmasına dikkat edin!

ASN.1 metninin, insan tarafından okunabilen normal bir metne gömülü olduğunu durumlarda, ona ayırt edici bir yazı tipi verilmesinin çok istendiğini unutmayın. Bu, ASN.1 tiplerinin veya dizi (veya küme) çoğ elerinin veya seçim alternatiflerinin tek tek adlarının bir cümleye gömülü olduğunu durumlarda özellikle önemlidir. Ayırt edici bir yazı tipinin mümkün olmadığı durumlarda, bu tür durumlarda italik veya tırnak işaretini kullanımı yaygındır. (Bu metinde genellikle tırnak işaretleri kullanılmıştır.)

ASN.1 metni birden fazla yerde görünebilir, o zaman Ek'te toplanan metnin "farklılıklar varsa önceliğe sahip olduğunu" söylemek yaygındır. Günümüzde "iki metinde farklılıklar bulunursa, bu şartnamede bir hatadır ve bu şekilde bildirilmelidir" demek daha yaygındır.

2.3 Makine tarafından okunabilir kopya sağlama

ASN.1'in tamamını bir arada toplayan bir ek, uygulamanın nasıl ele alınacağı önemli olmaksızın, onu basılı metnin birçok sayfası içinde tamamen parçalanmış halde bulundurmaktan açıkça daha iyidir.

Uygulayıcılarınız araçlar kullanıyorsa, makine tarafından okunabilir kopya isteyebileceklerdir: Bunu nasıl sağlayacağınızı düşündürün ve onlara nerede olduğunu söyleyin!

ASN.1 araçlarının varlığıından önce, ASN.1 spesifikasyonu bir uygulayıcıya neyi kodlayacağıını söylemek için oradaydı ve nadiren bir bilgisayara beslenmesi gerekiyordu, bu nedenle basılı metin yeterliydi. Uygulamanın büyük bir bölümünün doğrudan ASN.1 belirtiminin makine tarafından okunabilen bir versiyonundan otomatik olarak üretilmesini sağlayan ASN.1 derleyicilerinin gelişimiyle, bu tür materyallerin sağlanması biraz dikkat edilmesi gerekiyor.

"Yayınlanan" belirtim elektronik biçimde olsa bile, yayın için kullanılan format nedeniyle veya yukarıda tartışılan satır numaralarını çıkarma ihtiyacı nedeniyle bir kullanıcının resmi ASN.1 tanımını çıkarması kolay olmayabilir. veya "şekillerden" malzemeyi çıkarmak için.

Mümkün olan her yerde, "yayınlanmış" belirtim, tüm belirtim için makine tarafından okunabilir metnin yetkili bir kaynağıını tanımlamalıdır. Bu, şuan (1998), biçimlendirme karakterleri olarak yalnızca boş luklar ve yeni satırlar ile ve değer notasyonlarında ASCII olmayan karakterler için karakter adları kullanılarak (bkz. Bölüm II Bölüm 2) ASCII kodlu olmalıdır. Bununla birlikte, herhangi bir karakterin doğrudan temsiline izin veren sözcüklerin UTF8 kodlamalarının (yine bkz. Bölüm II Bölüm 2), giderek daha fazla kabul edilebilir, hatta tercih edilir hale gelmesi muhtemeldir.

Birçok erken ASN.1 spesifikasyonunun, basılı kopya spesifikasyonlarının satışından para kazanma geçmiş olan ve ilk günlerde makine tarafından okunabilir materyal sağlama ISO ve ITU-T tarafından yayınlanması talihsiz bir durumdu. Bununla birlikte, ilgili Standartların ve Tavsiyelerin bir dizi Editörü, ASN.1'in (genellikle ASCII metni olarak) makine tarafından okunabilen bir kopyasını (genellikle ücretsiz olarak) dağıtmak için izin aldı, ancak bu tür materyallerin mevcudiyeti her zaman geniş çapta duyurulmadı.

```
001 Wineco sipariş protokolü 002 {joint-iso-itu-
t internationalRA(23) set(42) set-vendors(9) 003 wineco(43) modüller(2) sipariş (1)}
```

```
004 TANIMLAR 005
```

```
OTOMATİK KETİ KETLER ::=
```

```
006 BAŞ LANGIÇ
```

```
007 008 009 010
```

```
Stok sipariş i ::= SIRALI {sipariş no
```

```
INTEGER, ad-
```

```
adres BranchIdentification,
```

```
....
```

Şekil 16: Modül başlığı

Birçok ASN.1 spesifikasyonunun, neden olabilecek tüm hatalarla birlikte araçlarda kullanılmak üzere basılı kopyalardan yeniden anahtarlanması gerekmesi talihsiz bir durumdur. Daha iyi araç satıcıları, zaman içinde en yaygın protokoller için makine tarafından okunabilen (Editörlerden elde edilen veya kendilerini yeniden anahtarlayarak) bir spesifikasyon stoğu oluşturmuş tur ve bunları talep üzerine müşterilerine sağlayacaktır. (Ek 5'teki URL, birçok ASN.1 tabanlı belirtimin bir listesine ve bazı durumlarda var olduğu bilinen makine tarafından okunabilir belirtim kaynaklarına bir bağlılık tırtılar.)

3 Modül başlığı ina dönüyoruz!

3.1 Sözdizimsel tartışma

Şekil 16, modül başlığı satırlarını (satır numaralarıyla birlikte) tekrarlamaktadır.

Sırasıyla maddeleri ele alalım. İlk satır modül adını içerir ve büyük harfle başlayan herhangi bir ASN.1 adıdır.

İnsanlar için modülü ve içeriğiini tanımlamayı amaçlamaktadır ve normal olarak aynı uygulama belirtimindeki diğer herhangi bir modül adından farklı olacaktır. Ancak bu bir gereklilik değil, çünkü ASN.1 tam bir uygulama spesifikasyonuna ilişkin gerçek bir konsepte sahip değil (yalnızca eksiksiz ve yasal bir modül)! "Tam bir belirtim" sorusuna daha sonra geri döneceğiz.

Modül başlığı sağlar

Bir modül adı

Eş siz bir modül tanımlaması

Etiketleme ortamının tanımı

Geniş letilebilirlik ortamının tanımı

İkinci/üçüncü satır, modül tanımlayıcı olarak adlandırılır ve bir nesne tanımlayıcı değil erinin başlığı bir halidir. Bu isim formunun diğer modüllerden farklı olması gereklidir - sadece aynı uygulama spesifikasyonundakilerden değil, dünya çapında sınımlı kadar yazılmış veya yazılacak olan herhangi bir ASN.1 modülünden! (Bazları Şekil 999'un geçerli olduğuunu söylese de, bu modülün sonraki herhangi bir sürümü dahil.)

Açıkçası, bu ikinci/üçüncü satırı eklemenize gerek yok. Yaklaşık 1988'de ASN.1'e dahil edildi ve kısmen geriye dönük uyumluluk nedenleriyle ve kısmen de nesne tanımlayıcısının bir parçasını almakta güçlük çeken (veya almaya çalışmak için çok tembel olan!) kişilere hesaba katmak için isteği bağlı bırakıldı. isim alanı. © OS, 31 Mayıs 1999

Bugün, yazdığınızınız herhangi bir modüle dünya çapında belirsiz olmayan adlar vermenizi sağlamak için bazı nesne tanımlayıcı ad-uzayı elde etmek nispeten kolaydır, ancak bunun nasıl yapılacağı ina (ve bir nesne tanımlayıcısının ayrıntılı biçimine) ilişkili bir tartışmayı erteliyoruz. değeri Bölüm II'ye. Bu kitapta kullanılan nesne tanımlayıcı değeri erlerinin "meşru" olduğunu ve dünyadaki diğer ASN.1 modüllerini adlandırmak için kullanılan diğer erlerinden (yasal olarak!) farklı olduğunu söylemek yeterlidir. Bu nispeten öneşiz kitap için isim alanını elde edilebilirse!

Dördüncü satır ve altıncı satır "boiler-plate" dir. Hiçbir şunu söylememiştir ama orada olmaları gerekiyor! Alternatif bir sözdizimi mümkün değil ildir. (Aynı durum modülün sonundaki "END" ifadesi için de geçerlidir.)

Beşinci satır, birkaç olasılıktan biridir ve modülün göndesi içinde metinsel olarak görünen tip gösteriminin (ancak tip referans adlarının değil) ayrıntılı yorumunu etkileyen modülün "ortamını" belirler.

Tasarımcılar lütfen dikkat edin: ASN.1'in bir modül başlığı ve bir "END" ifadesi olmadan bir belirtim yazmak yasa dışı olmasının yanı sıra, tip notasyonunun "ortamı" belirlenmediğinden çok belirsiz olabilir.

Öyleyse ... "ortamın" hangi yönleri belirtilebilir ve bu beşinci satırda hangi sözdizimi mümkündür?

"Ortamın" (bu kitapta) "etiketleme ortamı" ve "geniş letilebilirlik ortamı" olarak adlandırılan iki yolu vardır. Okuyucu, bunların her ikisinin de daha önce kısaca bahsettiğimiz, ancak hiç bir zaman tam olarak açıklamadığımız terimler içerdigini fark edecektir! Lütfen hayal kırıklığı ina uğradığını, ancak buradaki açıklama yine kısmi olacak - bu kavramların tam bir tartışıması için Bölüm II'ye gitmeniz gerekiyor.

Etiketleme ortamı (parantez içinde belirtmek için 4. satırda kullanılan dize ile) aşağıdaki idakilerden biridir:

Açık etiketleme ortamı (EXPLICIT TAGS).

Gizli etiketleme ortamı (IMPLICIT TAGS).

Otomatik etiketleme ortamı (OTOMATİK ETİKİTLER).

Bunların hepsinin ihmali edilmesi, açık bir etiketleme ortamı anlamına gelir. (Açık etiketleme ortamı 1988 spesifikasiyonuna kadar mevcut olan tek etiketleme ortamı olduğunu undan, bu tarihsel nedenlerden dolayıdır).

Geniş letilebilirlik ortamı (parantez içinde verildiğiini belirtmek için 4. satırda kullanılan dize ile) aşağıdaki idakilerden biridir:

Açık geniş letilebilirlik işaretleri gerektiren bir ortam (4. satırda geniş letilebilirlikten bahsedilmiyor).

Örtülü geniş letilebilirlik işaretlerinden oluşan bir ortam (EXTENSIBILITY IMPLIED).

Bu ortamları aşağıdaki tartışılmıştır. Hem etiketleme hem de geniş letilebilirlik ortamı belirtilmişse, bunlardan birinin metni önce gelebilir.

3.2 Etiketleme ortamı

Buradaki tedavi, ağırlıklı olarak TLV tarzı bir kodlamada etiketlemenin etkisine ve özellikle BER'e dayanmaktadır. Etiketlemenin ASN.1'e dahil edilmesi, böyle bir kodlama şemasına yardımcı olmak içindi. Herhangi bir kodlama kuralına uygulanabilen etiketlemenin daha soyut bir uygulaması Bölüm II'de verilmektedir.

Etiketlemenin etkilerine daha yakından bakmak için, şekil 17'de tekrarlanan şekil 13'teki bir bölümü gözden geçirelim.

Üç etiketleme ortamı

açık etiketleme

örtük etiketleme

otomatik etiketleme

BER'de bir SEQUENCE'in bir TLV olarak kodlandığıını ve "V" kısmının, dizinin her elemanı için bir TLV dizisi olduğunu daha önce belirtmiş tık. Dolayısıyla "deniztaşı İri" öğesi, üç öğenin her biri için bir tane olmak üzere üç TLV'den oluşan "V" kısmı ile bir TLV'dir. "[1]" etiketinin, "deniztaşı İri" dizisi için en dış taki "T" etiketin değerini geçersiz kıldığını da belirtmiş tık.

Benzer şekilde, NULL'lar üzerindeki [0] etiketinin ve [1] etiketinin, her bir NULL için TLV'deki varsayılan etiketi geçersiz kıldığını not ettiğim. Bu durumda, kodlama artık NULL için varsayılan etiketi içermeyen ve bu TLV'nin aslında bir NULL'u (veya diğer durumlarda bir INTEGER veya bir BOOLEAN vb.) temsil ettiğimde gerçekliği artık yalnızca "T" içindeki etiket tarafından ima edilir. Bölüm - [0] öğesinin bu durumda bir NULL'a atıfta bulunduğuunu anlamak için tür tanımını bilmeniz gereklidir. "Örtülü olarak NULL'u etiketlediği imizi" söyleyorum. Benzer şekilde, "deniztaşı İri" "SEKANS" dolaylı olarak "[1]" etiketiyle etiketlendi.

deniztaşı İri [1] Dİ Zİ {ismi
UTF8Dizesi,
konum Adresi ~~Paralel~~ Type,
depo [2] SEÇİ M
{kuzey [0] NULL, güney [1] NULL}}

Şekil 17: Şekil 13'ün bir parçası

Peki ya "depo" "SEÇİ M" üzerine yerleştiğimiz etiket? "CHOICE" ve "SEQUENCE" arasında yüzeysel bir benzerlik vardır (hemen hemen aynı söz dizimine sahiptirler), ancak aslında BER kodlamalarında çok farklıdır. "SEQUENCE" ile, ağırlıklı olarak daha önce açıklandıktan gibi bir dış seviye TLV sarmalayıcısına sarılır, ancak "CHOICE" ile, "CHOICE" alternatiflerinden biri için yalnızca TLV kodlamalarından herhangi birini alırız ve kullanırız "SEÇİ M" in kendisi için tüm kodlama (TLV) olarak.

Bu, "depo" etiketlemesini nerede bırakıyor? İlk bakışta, "CHOICE" (hangi alternatifin seçildiği) ile bağlı olarak "[0]" veya "[1]" olanı için TLV etiketini "[2]" etiketiyle geçersiz kılacaktır. Biraz düşündürün ve sonra bunun bir BUST spesifikasyonu olacağıını anlayın! Bir iletişim imzayı inden hattan hangisinin gönderildiğiini bilebilmek için alternatiflere özel olarak (BOŞları etiketleyerek) farklı etiketler verildi, ancak şimdiden her ikisini de ortak bir değeriyle aşıyoruz ("[2]")! Buna izin verilemez!

Uzun lafın kısası - ASN.1'de iki tür etiketleme mevcuttur:

örtük etiketleme: (şimdiden önce açıklanan budur), burada yeni etiket eski etiketi geçersiz kılar ve eski etiket tarafından taşıyınca tür bilgisi artık yalnızca kodlamada örtülüdür; buna bir "SEÇİ M" türü için izin verilemez; ve

özellikle bu paketin "T" kısmında yeni etiketi taş ımak için yeni bir TLV ambalajı ekliyoruz ve orijinal TLV'nin tamamını (eski etiketle birlikte) bu ambalajın "V" kısmında taşıyoruz; açıkça bu "SEÇİ M" için TAMAM.

"CHOICE" türleri için örtülü etiketleme yasak olsa da (bunu istemek yasa dışı bir ASN.1 belirtimidir), hem örtülü hem de açık etiketleme diğer tüm türlere uygulanabilir.

Bununla birlikte, açık etiketleme, maksimum tür bilgisini korurken ve aptal bir hat monitörünün mantıklı bir görüntü üretmesine yardımcı olabilirken, örtülü etiketlemeden açıkça daha fazla ayrıntılıdır.

Şimdi, farklı etiketleme ortamları ne anlama geliyor?

3.2.1 Açık etiketleme ortamı

Açık bir etiketleme ortamında, etiketin (sayı köşeli parantez içinde) hemen ardından "IMPLICIT" anahtar sözcüğü gelmediğin sürece, tüm etiketler açık etiketleme üretir.

Erken ASN.1 spesifikasyonlarında mevcut olan tek açık etiketleme ortamıydı, bu nedenle "IMPLICIT" kelimesini hemen hemen her yerde görmek yaygındı ve bu da okunabilirliği azaltıyordu. Elbette, bir "CHOICE" tip notasyonuna veya bu tür bir gösterim için bir tip referans adına uygulanan bir etikete "IMPLICIT" koymak yasa dışıydı ve öyledir de.

3.2.2 Gizli etiketleme ortamı

Örtük etiketleme ortamında, aşağıdaki idakilerden biri (veya her ikisi) geçerli olmadıkça tüm etiketler örtük etiketleme olarak uygulanır:

Etiket, bir "CHOICE" tip gösterimine veya böyle bir gösterim için bir tip referans adına uygulanmaktadır; veya

"EXPLICIT" anahtar kelimesi, etiket gösterimini takip eder.

Yukarıdaki durumlarda, etiketleme hala açık etiketlemedi. Uygulamada, yaklaşık 1986 ve 1995 yılları arasında yazılan spesifikasyonların çoğu, modül başlıklarında bir örtük etiketleme ortamı belirledi ve bir etiketten sonra "IMPLICIT" anahtar kelimesini veya "EXPLICIT" anahtar kelimesini görmek alıştırmadık bir durumdu. Ara sıra, EXPLICIT takviye için ve ara sıra (esas olarak güvenlik dünyasında fazladan bir TLV sarmalayıcıyı garanti etmek için) örtük etiketleme ortamında belirli türlerde kullanıldı.

3.2.3 Otomatik etiketleme ortamı

Açık ve örtük etiketleme ile ilgili kurallar, etiketlemenin ne zaman gerekli olduğunu ilişkili zaten karmaşık olan kurallara eklenir ve 1994 spesifikasyonunda, kısmen uygulama tasarımcısı için işleri basitleştirmek için ve kısmen de yeni Paketlenmiş Kodlama Kuralları (PER) olduğunu undan TLV tabanlı değil ve etiketlerden çok az yararlanılmıştır, otomatik bir ortam belirleme yeteneği i

örtülü etiketleme - "T" bölümünü geçersiz kılar

özellikle - fazladan bir TLV sarıcı ekler

Örtülü etiketleme ortamı, yalnızca yasal olduğu durumlarda örtülü etiketleme üretir - bir "SEÇİ M" de "AÇIK" demeye gerek yoktur.

Otomatik etiketleme

Bu ortamı kurun ve etiketleri unutun!

etiketleme eklendi.

Bu durumda, etiketler otomatik olarak her dizinin (veya kümenin) tüm öğelerine ve bir seçimin her alternatifine “[0]”dan başlayarak sıralı olarak eklenir (her “SEQUENCE”, “SET” veya “CHOICE” için ayrı ayrı). inşaat). Belirli bir “SEQUENCE” (veya “SET”) öğesinin veya “CHOICE” alternatifinin öğelerinden herhangi birinde etiket gösterimi mevcutsa, o zaman tasarımcının kontrol altına alındı ve etiketlerin HİÇBİR otomatik uygulaması olmayacak. (Mevcut etiket gösterimi, bu durumda örtük etiketleme ortamında yorumlanır.)

Bugün genellikle "OTOMATİ K ETİ KETLER"in modül başlığına yerleş tirilmesi tavsiye edilir ve böylece tasarımcı etiketleri tamamen unutabilir! Ancak (lütfen şekil 999'a bakın!), "OTOMATİ K ETİ KETLER"in BER'de gereğinden fazla ayrıntılı olabileceğine ve ASN.1 araçları kullanılmadığında uygulama hataları için daha fazla kapsam sağlayabileceğine dair bir karşılıkça var. Seçimi yap! Ama benimkinin ne olacağıını biliyorum!

3.3 Geniş letilebilirlik ortamı

Versiyon 2'deki **öğ** elerin eklenmesine izin vermek için bir TLV stili kodlamanın gücünü zaten tartışmıştık, sürüm 1 spesifikasyonları bu tür ek **öğ** eleri atlayabilir ve yok sayabilir.

(Bu geniş letilebilirlik kavramı aslında diziler ve kümelerin dışındaki şeylelere genelleme yapıyor ama şimdilik bunlar yeterli.)

geniş letilebilirlik iş aretīsi

Sürüm 2 malzemesinin, sürüm 1 sisteminin sürüm 2 kodlamalarını çöme yeteneğ ini etkilemeden eklenebileceğ i bir ekleme noktasını tanımlayan bir üç nokta (veya bir çift).

ASN.1'de bir miktar geniş letilebilirlik yeteneğini koruyacağın ve BER'in TLV'sinden (yeni PER gibi) daha az ayrıntılı olan kodlama kurallarını tanıtacağın, bir tasarımcının geniş letilebilirlik gereksinimleri uygulama belirtiminde yapılmalıdır. açık.

Ayrıca, yalnızca kodlama kurallarının bir sürüm 1 sisteminin eklenen sürüm 2 materyalinin sonunu bulmasına (ve belki de yok saymasına) izin vereceğinden değil, aynı zamanda uygulama tasarımcısının bir sürüm 1 sisteminden beklenen eylemleri açıkça belirttiğinden de emin olmamız gereklidir. böyle bir malzeme alır.

Bunu mümkün kılmak için, 1994 belirtimi ASN.1 notasyonuna bir geniş letilebilirlik işareti koydu. Bunun en basit kullanımında "Sipariş -stok" tip qüsterimi şekil 18'deki gibi yazılabılır.

Burada, "urgency" "authenticator" arasına dış düzeyde elerin daha sonra eklenmesine ve sürüm 2'de ek numaralandırmalı, genelindeki istenilende kötü bir etki olmaksızın izin vermek için kodlama kurallarına ihtiyacımız olduğumu tespit ediyoruz. (Tüm ayrıntılar Bölüm II'dedir.) ("Doğ rulayıcı" ifadesinin sonuna sürüm 2'de elerini eklemekten memnun olsaydık, o zaman tek bir üç nokta yeterli olurdu.)

Stok sipariş i ::= SIRALI {sipariş no
adres BranchIdentification, ayrıntılar
Dİ Zİ Sİ
Dİ Zİ {çözelme
NESNE TANIMLAYICI, vaka TAM SAYI},
aciliyet NUMARALI
{yarın(0), üç günlük(1),
hafta(2), ... }
VAR SAYILAN hafta,
...,

Şekil 18: Geniş letilebilirlik işaretleri ile stok sipariş i

Elipslerin yerleş tirildiği ve yeni sürüm 2 malzemesinin, konuş landırılmış sürüm 1 sistemlerini bozmadan güvenli bir şekilde eklenebileceği yere (sürpriz, sürpriz!) ekleme noktası denir.
Herhangi bir dizide, sette, seçimde vb. yalnızca bir ekleme noktasına sahip olmanızı izin verilir.

Uyarı okuyucusu (şimdije kadar bu ifadeye alışmış olmalısınız, ancak muhtemelen hala can sıkıcıdır - özür dilerim!), provizyon yapmak için uyarı kodlama kurallarına ek olarak sürüm 1 sistemlerine ne yapması gerektiğiini söylemenin de gerekli olduğuunu fark edecktir. ilave malzeme ile yapının Yerleştiğindeki ekleme noktası konusu olduğuunda, gerekli eylemin eklenen öğeleri sessizce yok saymak olacağ "açık" görünebilir. Ancak sürüm 1 sistemi, bilmediğim bir "aciliyet" değil eri alırsa ne yapmalıdır? Ekleme noktası, sürüm 2 tipi ekleme noktası, üç noktadan hemen sonra eklenebilen baş ka bir gösterim parçası gösterimdeki üç noktaların sistemdeki II. ayrıntılar istiyorsanız! (İstisna belirtimi bir ünlem işaretiyle başlar yanına giden enuze anlayacaksınız!).

Uygulama tasarımcıları, zorunlu hale getirilmemesine rağmen, geniş letilebilirlik işareti kullanıklarında istisna belirtimleri sağlama için teşvik edilir.

Açık geniş letilebilirlik işareti, üç nokta ve bir üç noktanın varlığından kaynaklanan kodlama kuralları ve sürüm 1 davranış üzerindeki tüm çıkarımları gerektiren bir ortamda, yalnızca üç nokta, gerekli olduğu yerde belirtimde metinsel olarak mevcutsa oluşur.

Zımmi geniş letilebilirlik işareti bulunan bir ortamda, bu tür işaretleyle otomatik olarak izin verilen yapılarda bir geniş letilebilirlik işareti içermeyen, o ortamda tüm tip gösterimlerinde yapının sonuna bir tane eklenir.

Dolayısıyla, şe 18'deki tip gösterimi, ima edilen geniş letilebilirlik ortamındaysa, "ayrintılar" "SEQUENCE OF" içindeki "SEQUENCE{...}" yapısının sonuna otomatik olarak ek bir uzanti işareti eklenir.

Bu metnin yazıldığı sırada, uzanti işaretleycileri yaygın olarak kullanılıyordu, ancak çok az tasarımcı, sürüm 2 malzemesinin eklenmesi için ek, belki de gereksiz ekleme noktalarına sahip olmanın maliyetine rağmen, zımmi geniş letilebilirlik işareti bulunan bir ortamını belirtmeyi seçti. hattaki bit cinsinden düşüktür.

Zımmi geniş letilebilirlik işareti bulunan ortam: geniş letilebilirlik işareti olmayan (ve izin verilen) herhangi bir yapının eklenmesi (sonuna) bir ortam.

Sorun muhtemelen bu ortamı kullanmayla ilgili üç sorundan kaynaklanmaktadır:

Ekleme noktası her zaman sondadır - konumu üzerinde hiçbir kontrolünüz yoktur.

Sürüm 2 spesifikasyonunu üretirken, eklediğiniz öğelerin önüne gerçekte üç nokta eklemeniz gereklidir - ve unutabilirsiniz!

Uzanti işaretleycili bir istisna belirtiminin varlığına ilişkili bir hükmü (bu ortam kullanıldığından) yoktur, bu nedenle sürüm 2 öğelerinin veya değil erlerinin varlığında sürüm 1 sistemlerinin gerekliliği davranışına ilişkili tüm kurallar, tümüne genel olmalıdır. Şartname.

Sonuç tavsiyesi: Uzantı iş aretçilerinin nerede olmasını istediği inizi ve sürüm 1 sistemlerinin sürüm 2 ög elerine ve deг erlerine vermesini istediği iniz iş lemeyi (bu kararları yerelleş tirmek ve açık hale getirmek için istisna belirtimlerini kullanarak) dikkatlice düş ünün, ancak genel bir çözüme giriş meyin. zınni geniş letilebilirlik ortamı.

4 İ hracat/ithalat bildirimleri

Altı satırlık bir baş lıg in etkilerini açıklamak çok fazla metin gerektirdi! ASN.1 Standardı/Önerisinde çok daha az metin var! Ama henüz iş imiz bitmedi!

Altıncı satırın ("BEGIN") ardından ve (yalnızca) herhangi bir tür veya deг er atama ifadesinden önce, bir export bildirimi (ilk) ve/veya bir imports bildirimi ekleyebiliriz. Bunlar genellikle modül baş lıg inin bir parçası olarak kabul edilir.

Bu noktada, daha önce yalnızca ima edilen ş eyi vurgulamak önemlidir: ASN.1 repertuarında, referans adları olan ş eylerin yalnızca türler ve deг erlerden daha fazlası vardır, ancak bunlar açık ara en önemlileri (veya en azından, çoг u spesifikasiyonda en üretken!).

1994 öncesi (yalnızca) makro adlarını ekleriz ve 1994 sonrası bilgi nesnesi sınıflarının, bilgi nesnelerinin ve bilgi nesnesi kümelerinin adlarını ekleriz. Bunların tümü bir dış a aktarma veya içe aktarma ifadesinde görünebilir, ancak ş imdilik yalnızca tür referans adları ve deг er referans adları üzerinde yoğunlaş iyoruz.

Bir ihracat beyanı nispeten basittir ve ş ekil 19'da gösterilmektedir.

```

Wineco ortak türleri
  { ortak iso-itu-t uluslararası(23) set(42) set-vendors(9)
  wineco(43) modüller(2) ortak(3)}
  TANIMLAR
    OTOMATİ K ETİ KETLER ::=

    BAŞ LAMAK

    İ HRACAT Çıkış Türü, Adresi;

    Çıkış Tipi ::= SIRALI { ....
      ...
      .... }

    Adres ::= SIRALI { ....
      ...
      .... }

    ....
    SON
  
```

Ş ekil 19: Genel tipler modülü (ilk deneme)

İ hracat/İ thalat beyanları

Bir modülün baş inda, diг er modüllerde tanımlanan türlerin kullanımını belirten (içe aktarma) veya bu modülde tanımlanan diг er modül türleri için kullanılabilir hale getiren (dış a aktarma) bir çift isteг e baг lı ifade.

"OutletType" ve "Address" için tip tanımlarını yapın, bunları yaygın olarak kullanılan tiplerden oluş an bir module koyun ve dış a aktarın, yani baş ka bir module kullanılabilir hale getirin.

Gerçekte, "Wineco-common-types" içinde dış a aktarmamayı seçtiğ imiz daha fazla destekleyici tip olacaktır - bunlar diğ er modüllerde kullanılamaz. Muhtemelen ihraç edilen daha fazla tür de olacaktır.

"EXPORTS" ifadesi için bir ifade sonlandırıcı olarak noktalı virgülün varlığı ina dikkat edin. Bunun "IMPORTS" ifadesini de sonlandırmak için kullanıldığı ini göreceğ iz. Bunlar, ASN.1'in bir ifade sonlandırıcıya sahip olduğu yalnızca iki durumdur.

Ayrıca, tarihi nedenlerle

("EXPORTS" yalnızca 1988'de eklenmiş tir) bir "EXPORTS" ifadesinin atlanması, "her ş ey baş ka bir modül tarafından içe aktarılabilir" anlamına gelirken:

EXPORTS ifadesinin olmaması, "HER Ş EYİ ihraç eder" anlamına gelir. İ HRACAT ; "Hİ ÇBİ R Ş EY İhraç etmez" anlamına gelir.

İ HRACAT ;

"Baş ka bir modül tarafından içe aktarılabilen hiç bir ş ey yok" semantiğ ine sahiptir.

Daha sonra, ilk olarak ş ekil 13'te kullandığımız "Security-Type"ın Secure Electronic Transactions (SET) spesifikasiyonundan (tamamen ayrı bir yayın) alındığı in ve "Wineco-common-type" modülünde deş il, aynı zamanda diğ er modüllerimizde de mevcuttur. Bunu "Wineco-common-types" modülünde kullanmak için içe aktarıyoruz, ancak diğ er modüllerimizin içe aktarma maddelerini daha basit hale getirmek için tekrar dış a aktarıyoruz (yalnızca "Wineco-common-types"tan içe aktarmaları gerekiyor). Tip tanımlarının bu "aktarılması" yasalıdır.

Wineco ortak türleri

{ ortak iso-itu-t uluslararasıRA(23) set(42) set-vendors(9)
wineco(43) modüller(2) ortak(3)}

TANIMLAR

OTOMATİ K ETİ KETLER ::=

BAŞ LAMAK

İ HRACAT Çıkış Türü, Adresi, Güvenlik-Tipi;

İ THALATLAR Güvenlik Tipi
SET modülü

{joint-iso-itu-t uluslararasıRA(23) set(42) modülü(6) 0};

Çıkış Tipi ::= SIRALI {

....
.... }

Adres ::= SIRALI {

....
.... }

.....

SON

Ş ekil 20: Ortak türler modülü (geliş tirilmiş)

Bu, § ekil 19'u § ekil 20 olarak değil iş tirir.

İ HRACATTA olduğ u gibi, "İ THALATLAR" ve "Kİ MDEN" arasındaki metin, referans adlarının virgülle ayrılmış bir listesidir. Birden fazla baş ka modülden nasıl import yapacağ ımızı bir sonraki § ekilde göreceğ iz.

Bu noktada, bir tür, belirli bir etiketleme veya geniş letilebilirlik ortamına sahip bir modülden farklı bir etiketleme veya geniş letilebilirlik ortamına sahip bir modüle içe aktarılırsa, içe aktarılan bu tür için tip gösterimi, modülün ortamı ile yorumlanmaya devam eder. ki baş langıç ta tanımlandı. Bu, çevre kavramının sunulma biçiminden bariz görünebilir, ancak noktayı güçlendirmeye değil er - ithal edilen § ey, bir anlamda, türün metni değil il, tür gösteriminin tanımladığı i "soyut tür"dür. gösterim.

5 Yapımızı iyileş tirmek

son örnek

Artık birkaç modül kullanıyoruz, üst düzey türümüz olarak bir SEÇİ Mimiz var ve bunu üst düzey türümüz olarak açıkça tanımlıyoruz, Bir nesne tanımlayıcı değil er-referans-ad kullanıyoruz, UYGULAMA sınıf etiketleri kullanıyoruz, geçersiz kodlamaları iş liyoruz, istisna iş leme ile en üst düzeyde geniş letilebilirlik e sahibiz. ASN.1 kullanımımızda oldukça karmaş ık hale geliyoruz!

Şimdi epeyce değil iş iklik yapacağ iz! Baş ka bir modülde tanımlanan "Return-of-sales" adlı ikinci bir üst düzey mesaj ekleyeceğ iz (ve daha fazlası için hazırlık yapacağ iz) ve şimdi "ABSTRACT-SYNTAX" ifadesini (Bölüm 2'de bahsedilen) dahil edeceğ iz. ilk koyacağ ımız baş ka bir modülde yeni üst düzey tip.

Biraz daha gelişmiş bazı özellikleri göstermek için bu üst düzey modülde birkaç kozmetik değil iş iklik daha yapacağ iz. Yapacağ iz:

üst düzey mesajlarımız için "UYGULAMA" sınıf etiketlerini kullanın. Bu gereklidir, ancak genellikle yapılır (etiket sınıflarının sonraki tartışmalarına bakın)

uzun nesne tanımlayıcılarımızın ilk bölümünü "wineco OID" değil er-referans-adına atayın ve bunu ASN.1'in yaygın olarak kullanılan bir özellig i olan nesne tanımlayıcılarımızın baş langıcı olarak kullanın.

Kod çözücünün gelen materyalin geçersiz bir kodlamasını tespit etmesi durumunda, metnimizin sistemin nasıl davranışacağı in tam olarak belirleyeceğ in netleş tirmek için "ABSTRACT-SYNTAX" a metin ekleyin.

Nihai sonuç, § ekil 20'deki metnin takip ettiğ i varsayılan § ekil 21'de gösterilmektedir.

§ ekil 21'e iyice bakın ve ardından "bu konuyu anlatan" aşağ idaki metni okuyun.

001'den 006'ya kadar olan satırlar yeni bir § ey değil il. 10. ve 13. satırlarda, nesne tanımlayıcı değil erimizi kısıtlamak için "wineco-OID" (015 ve 016 satırlarında tanımlanmış tir) kullanacağ ımızı, ancak henüz kapsam içinde olmadığı i için bunu modül baş lig i nda kullanmamıza izin verilmediğ in unutmayın., ve nesne tanımlayıcı değil eri tam olarak yazılmalıdır.

Satır 007, diğ er modüllerden referans için hiçbir § eyin bulunmadığı in söylüyor.

```

001 Wineco-common-top-level { ortak-iso-itu-t
002     internationalRA(23) set(42) set-vendors(9) wineco(43) modüller(2) top(0)}
003
004 TANIMLAR
005     OTOMATİ K ETİ KETLER ::=
006     BAŞ LAMAK
007     İ HRACAT ;
008     İ THALATLAR Stoka göre sipariş
009     Wineco sipariş protokolü
010     {wineco-OID modüller(2) sıralama(1)}
011     DAN satış iadesi
012     Wineco-dönüş protokolü {wineco-OID
013     modüller(2) dönüş ü(2)};
014
015 wineco-OID NESNE TANIMLAYICI ::=
016     { ortak-iso-itu-t uluslararasıRA(23) set(42) set-vendors(9) wineco(43)}
017     wineco-soyut-sözdizimi ÖZET-SYNTAX ::=
018
019     {Wineco-Protocol TANIMLAYAN
020             {wineco-OID özet sözdizimi(1)}
021             HAS PROPERTY
022             {geçersiz kodlamaları iş ler}
023             -Madde 45.6'ya bakın --
024
025     Wineco-Protokol ::= SEç M
026     {sipariş [UYGULAMA 1] Stok sipariş i,
027     satış [UYGULAMA 2] Satışların iadesi,
028     ... ! PrintableString : "45.7 maddesine bakın"
029     }
030
031     SON
--Yayınlanan özelliklerde yeni sayfa.
032 Wineco sipariş protokolü 033 { ortak iso-itu-t
uluslararasıRA(23) set(42) set-vendors(9) 034 wineco(43) modüller(2) sipariş (1)} 035 036 037 038 039 040 041 042 043 044
045 046 047 048
049 TANIMLAR
050     OTOMATİ K ETİ KETLER ::=
051     BAŞ LAMAK
052     İ HRACAT Stoka göre sipariş ;
053     İ THALATLAR OutletType, Address, Security-Type FROM
054         Wineco-ortak türleri {wineco-OID
055         modüller(2) ortak (3)};
056
057 wineco-OID NESNE TANIMLAYICI ::=
058     { ortak iso-itu-t internationalRA(23) set(42) set-vendors(9) wineco(43)}
059
060     Stok sipariş i ::= SIRALI { ....
061
062     ....
063     .... }
064
065
066
067
068
069
070     BranchIdentification ::= AYARLA
071     { ....
072     ....
073     .... }
074
075
076
077
078
079
080
081     SON

```

Ş ekil 21 (ilk kısım): Bu bölüm için son ş ekil!

008'den 013'e kadar olan satırlar, diğ er iki modülümden beklediği imiz ithalatlardır. Buradaki sözdizimine dikkat edin: aynı modülden içe aktarılan daha fazla tür olsaydı, 039 satırındaki gibi virgülle ayrılmış bir liste olurdu, ancak iki farklı modülden içe aktardığı imizda 011'den 013'e kadar olan satırlar sadece 008 ve 010 satırlarından çalışır, ayırıcı yok.

015 ve 017 satırları, nesne tanımlayıcı değil er-referans-adımıza bir değil er ataması sağlar. Tam bir nesne tanımlayıcı değil erinin ilk kısmına "geniş leyen" bir nesne tanımlayıcı değil er-referans-adı ile başlayabilmesi ve daha sonra, 010, 013 ve 020 satırlarında gördüğüümüz gibi. İleriye atlamak istiyorsanız ve ilgileniyorsanız, OID ağacı Bölüm II, Bölüm 1'de daha ayrıntılı olarak açıklanmaktadır.

018'den 023'e kadar olan satırlar, üst düzey türü tanımlayan, soyut sözdizimini adlandıran ve ona bir nesne tanımlayıcı değil eri atayan "sihirli parça" sözdizimidir - daha eski spesifikasyonlarda insan tarafından okunabilir metinde yapılacak bir şeyle. Aslında, bu sözdizimi "ad hoc" değil ildir, Bölüm II'de tartışılacak olan bir bilgi nesnesi atama ifadesinin bir örneğidir.

"MÜLKİ YETİ VAR" ve 22 ile 23. satırlar, şeyle anda belirtilebilen tek "özellik"tir.

Bu sözdiziminin dahil edilmesi, kısmen, kod çözmenin uygulamadan ayrı bir katman olduğunu ve kod çözmenin tanınan bir soyut değil er üretememesi durumunda yapabileceğiniz tek şeyle bağlanmayı iptal etmek olduğunu şeyle eklindeki eski bir OSI bakış açısına karşı koymaktır! (Şekil 999'u tekrar kontrol edin!) Aşağıda bir fikir! Ancak 20'den 23'e kadar olan satırların dahil edilmesi, okuyucuya, spesifikasyonun gerçekten de (madde 45.6'da) bu durumda ne yapılması gerekiyor ince kapsayan bir metin içeriği konusunda güvence verir.

025'ten 029'a kadar olan satırlar, her bir kodlamanın (ana mesaj türlerimizden herhangi birinin veya ana mesaj türlerimizin) net olmasını sağlama için üst düzey iletlerimiz için ihtiyaç duyduğumuz tek ASN.1 türünü tanımlar. BER'i "Stok sipariş i" ve "Satış verilerinin iadesi" olmak üzere iki tür basitçe uygularsak, aynı zamanda kullanılan bir tür değil er için kullanılan bir bit modeli elde edebiliriz (ve muhtemelen elde ederiz). diğ er türden bir değil er için bir kodlama. Yeni bir CHOICE tipi oluşturarak, bir CHOICE tipinin etiket benzersizliği kuralları bu sorunu çözer. 005 satırında "OTOMATİK KETİ KETLER" kullandığı imizde dikkat edin, bu nedenle 026 ve 027 satırlarında herhangi bir etiket eklemeye gerek kalmadı, ancak kişiisel tercih ve tarz gereği i "T" değil erinin tam kontrolünü ele almayı seçti. Orijinal etiketler ne olursa olsun, mesajlarımızın en dış taktiği TLV'sinde "[UYGULAMA 0]" ve diğ erini "[UYGULAMA 1]" kodlaması yapın. Bazı tasarımcılar bunun el kodlayıcıları için yararlı olduğunu iddia ediyor - bir araç kullananlar için kesinlikle alakasız. 026 ve 027 satırlarında etiketlerin bulunmasını, 025 satırındaki SEÇİM için otomatik etiketlemeyi devre dışı bıraklığına ve etiketleme ortamını geçici olarak örtük etiketleme ortamıyla değil iş tirdiği ince dikkat edin.

Satır 028, sürüm 2'de daha fazla dış düzey mesaja ihtiyaç duyabileceğimizden şeyle phelendiği imizi ve kodlama kurallarının bu tür mesajları eklemenin sürüm 1 sistemlerinin sürüm 1'deki mesajları doğrudu şeyle ekilde olmasını engellememesini sağlama gereği ince söyler. 028 satırındaki materyal (istisna belirtimi - Bölüm II'de ayrıntılı olarak açıklanmışdır) bize, madde 45.7'nin, sürüm 2'de (veya daha sonra) eklenen mesajları alması durumunda sürüm 1 sisteminin gerçekleştirmesi gereken eylemleri ayrıntılılarıyla açıkladığı ince söyler.

032'den 101'e kadar olan satırlar bizim ikinci modülümdür (orijinal Şekil 13'ün gelişimi) ve yeni hiç bir şeyle icermez. Ancak, 043 ve 045 satırlarının 015 ile 017 arasındaki satırların tekrarı olduğunu ve bu istenmeyen görünebilir. "Wineco-OID"yi başka bir modülde (ihtiyaç duyabileceğimiz birçok başka değil er-referans-adıyla birlikte) tanımlamak ve bu adı o modülden almak mümkün olabilirdi.

Ancak, (bariz "sonsuz özyineleme" nedeniyle) bu içe aktarma için "FROM"da "wineco-OID" kullanmamıza izin verilmeyecek, bu nedenle olabildiğince çok metin yazacağımız (ve bunu her modülde tekrarlayacağımız) ithalatı yapmak istiyoruz) 015 - 017 ve 043 - 045 satırlarında yazdırıldığımız gibi. Elimizdeki, alabildiğimiz kadar az.

102'den 139'a kadar olan satırlar, yapısal olarak 032'den 101'e kadar olan üçüncü modülü müzdür ve yeni bir şeyle getirmez. Daha sonra tüm belirtim, daha önce tartıştığımız "ortak tip" modülü müzü veren Şekil 20'deki metinle sona erer.

```
--Yayınlanan özelliklerde yeni sayfa. 102 Wineco
iade protokoli 103 104 105 106 107 108 109
    { ortak-iso-itu-t uluslararası RA(23) seti(42)
      set-vendors(9) wineco(43) modüller(2) döner(2)}
TANIMLAR
    OTOMATİK ETİ KETLER ::=

    BAŞ LAMAK
    İ HRACAT Satışların geri dönüşü Ü;
    İ THALATLAR OutletType, Address, Security-Type FROM
        Wineco-ortak türleri {wineco-OID
        modüller(2) ortak (3)};

    wineco-OID NESNE TANIMLAYICI ::=

        {iso tanımlı kuruluş icd-wineco(10)}
115
116    Satışların geri dönüşü Ü ::= SIRALI { ....
117
....      ....
....      ....
....      ....
....      ....
139    SON
```

Şekil 21 (son kısmı): Bu bölüm için son Şekil!

6 Tam Özellikler

Daha önce belirtildiği gibi, ASN.1'de "tam belirtim" kavramı yoktur, yalnızca doğrudan (tam) modüller vardır; bunlardan bazıları üst düzey bir tür (veya hangi okunabilir metinde tanımlanan üst düzey bir tür içerebilir).

Çoğu durumda, bir modül başka bir modülden bir tür alırsa, iki modül aynı yanında olacaktır (genellikle, aynı belirtimin parçası), ancak bu bir gereklilik değildir. Türler herhangi bir modülden herhangi bir yere aktarılabilir.

Bir modülde üst düzey bir tip aldığıımızı ve kendi modülü içinde (doğrudan veya dolaylı olarak) kullandığımız tüm tip-referans adlarının zincirini ve türlere alma ve verme bağları (yine herhangi bir derinliği zincirlenmiş) yoluyla takip ettiğimizi varsayıyalım. Diğer modüllerde. Bu bize, bunun en üst düzey tür olduğunu uygulama için "tam belirtimi" oluştur. Tüm tür setini verecektir ve tüm bu türlerin belirtimleri (tabii ki) bu uygulamanın herhangi bir uygulayıcısına tarafından kullanılabilir olmalıdır. ve uygulamaya yardımcı olan herhangi bir ASN.1 derleyici aracına. Tamamen bu kitabın bu bölümünün son bölümünün amaçları doğrultusunda, bu tip tanım açısından uygulama için gerekli tipler adı verilecektir.

Uygulama için gerekli tüm türlerin tanımlarını elde etmek için hangi ek (fiziksel) belgelerin gerekli olduğunu tam olarak herhangi bir uygulama belirtiminin metninin başında net bir şekilde belirtmesi, herhangi bir uygulama tasarımcısına önemli bir tavsiyedir.

Ancak şimdiden, bu uygulama için gerekli türlerin tanımlanmış modüller kümesini ele aldığıımızı varsayıyalım. (Yine, yalnızca sonraki birkaç paragrafta, bunları uygulama için gerekli modüller olarak adlandıracagız).

Genel olarak, üst düzey tipi metinsel olarak içeren modül, muhtemelen uygulama için gerekli tipler dışında herhangi bir tip içermez (bunun böyle olması gerekliliği olmamasına rağmen). Ancak, özellikle diğer yayınlardaki belki de daha genel gereksinimleri karşılamak için üretilmiş modüllerden içe aktarmaya başlar başlamaz, o zaman uygulama tarafından gerekli olmayan, uygulama için gerekli modüllerde tanımlanmış bazı tipler olabilir!

Daha sonra göreceğimiz gibi, aletlerin zekâları farklıdır. Başvurulan türleri fiziksel olarak ayıklamanızı ve her şeyi aynı module ilk önce en üst düzey türle koymamızı gerektiren bazı araçlar var! Bu son derece küçük bir uçtur ve farklı modüllerin etiketleme veya geniş letilebilirlik ortamları farklısa gerçek sorunlara yol açabilir.

En iyi araçlar, onlara uygulama için gerekli tüm modüller (ve üst düzey türü tanımlayan bir yenergeyi) içeren makine tarafından okunabilir metin (belki birkaç dosyada) sunmanıza izin verir ve bu modüllerden yalnızca uygulama- gerekli türler, yalnızca seçtiğiniz programlama dilindeki veri yapılarına eşleme. (Bu, uygulama için bellek gereksinimini minimumda tutar).

Diğer modüllerden türleri referans almanın (ice aktarmanın) veya metinsel olarak kopyalamanın artıları ve eksileri hakkında daha önce (potansiyel bir uygulama tasarımcısı olarak) kendinizle yaptığıınız tartışmayı hatırlıyor musunuz? Bu tartışmayı yeniden açabilirsiniz!

7. Karar

Şekil 13'teki basit tip atamalarımızdan çok yol kat ettik!

ASN.1 tabanlı bir uygulama belirtiminin üst düzey yapısı tanımlanmış ve araştırmış ve önemli kavramların çoğu tanıtılmıştır.

Ancak bir uyarı: Burada öneklemek için kullandığımız basit protokol, Şekil 14'te özetlenen tek ASN.1 modülü olarak daha iyi yapılandırılabilir. Dış aktarma/ice aktarma ile birden çok modülün ek gücü (ancak karmaşıklığı) önemlidir. ancak gereksiz yere kullanılmamalıdır - mümkün olduğunda basit tutun! Şekil 14 yapısı iş görürse, Şekil 14 ile kalın!

Şimdiden, basit gönme tipleri ve yapım mekanizmaları için ASN.1 tipi ve değerler notasyonları (bu bir sonraki bölümde yapılmıştır) ve (Bölüm II'de - Bölüm 5'te bir giriş ile) tartışmasını tamamlamaya kalmıştır. Bahsettiğimiz daha gelişmiş kavramların daha kapsamlı bir şekilde ele alınmasını sağlayarak 1994'te eklenen özelliklerin daha fazlasını tartışmak için.

Bununla birlikte okuyucu, 1994'ten önce üretilen çoğu gerçek ASN.1 spesifikasyonunu okuyabilmeli ve anlayabilmeli ve 1994 ASN.1'de tanıtılan bazı özelliklerin kullanımını tanımalıdır. Okumaya devam etmek!

4. Bölüm

Temel veri türleri ve oluş turma mekanizmaları - kapatma

(Veya: Çeşitli şekil ve boyutlarda tuğlalara ihtiyacınız var!)

Özet:

ASN.1'de önceden tanımlanmış bir dizi tür vardır, örneğin:

TAM SAYI,

BOOLE,

UTF8Dizesi.

Bunlar, aşağıldakiler gibi yapı mekanizmalarıyla daha karmaşık ikinci kullanıcı tanımlı türler oluş turmak için kullanılır:

SEKANS,

AYARLAMAK,

SEÇİM,

DİZİSİ,

SET,

vb.

Bu işaret mekanizmalarının birçoğu önceki bölümlerdeki örneklerde ve resimlerde yer almıştır.

Bu bölüm, tüm temel ASN.1 türlerinin ayrıntılı sunumunu tamamlar ve her durumda aşağıldakilerin açık bir tanımını verir:

tip için tip gösterimi,

türdeki soyut değerler kümesi ve

bu türdeki değerler için değer gösterimi.

Tip/değ erle ilgili ek notasyon parçaları da ele alınarak, 1994 öncesi spesifikasyonlarda yaygın olarak kullanılan sözdizimi tartış masını büyük ölçüde tamamlıyor.

Bdüm, iş lenmesi sonraki bdüme (Gelişmiş özelliklerin tartışılması) veya Bdüm II'ye ertelenen ek kavramların bir listesiyle sona erer.

1 Örnek çizim

Bazı tip ve değerler gösterimlerini göstermek için Satış İ adesi mesajımızı § ekil 22'deki gibi tanımlayacağımız. § ekil 22, NULL dışındaki tüm temel ASN.1 tiplerini içerecek şekilde tasarlanmışdır ve kancayı sağlar. Bu türlerin daha fazla tartışılması.

Şekil 22, tüm temel ASN.1 türlerine girişiniz tamamlamak için dikkatlice oluş turulmuş tur - işte bu kadar millet!

§ ekil 22'ye iyi bakın. Anlamını anlamak sizin için şimdiden kadar oldukça kolay olmuştur. Bununla ilgili bir sorununuz yoksa, ASN.1'i üzerine bir kitap yazacak veya bir kurs verecek kadar iyi anlamak istemiyorsanız, muhtemelen bu bölümün geri kalanını atlayabilirsiniz! (Daha sonra § ekil 23'teki nesne tanımlayıcı değerlerinin ayrıntılarını azaltmak için wineco öğelerini § ekil 22'ye dahil ettim!)

Satış getirisi ::= Dİ Zİ {versiyonu

Bİ T Dİ Zİ Sİ

{sürüm1 (0), sürüm2 (1)} VARSAYILAN {sürüm1},

bildirilen gün sayısı INTEGER

{hafta(7), ay (28), maksimum (56)} {1..56} VARSAYILAN hafta,

rapor tarihi ve saatı SEÇ Mİ {iki haneli yıl UTCZamanı,

dört haneli yıllık GeneralizedTime},

-- Sistem saatı dört basamaklı bir yıl sağlıyorsa, -- ikinci alternatif kullanılacaktır. -- ilk alternatif ile

zaman, -- kayan bir pencere olarak yorumlanacaktır. Gecikme nedeni SAYILANDIRILMIŞ {bilgisayar

arızası, ağ arızası, diğ er} İSTEÇ BAĞLI, -- Bu alanı yalnızca ve yalnızca -- rapor edilen gün sayısı

yediği geçiyorsa dahil edin.

ek bilgi İSTEÇ BAĞLI PrintableString Dİ Zİ Sİ,

-- Bu alanı, ancak ve ancak -- gecikme nedeni "diğ er" ise dahil edin.

Satış verileri SET OF Rapor öğesi,

... ! PrintableString : "Wineco kılavuzunun 15. bölümüne bakın")

§ ekil 22 (bdüm 1): Temel ASN.1 türlerinin kullanımının gösterimi

```

Report-item ::= SEQUENCE {item item-
description -- Yeni stoklanan herhangi   NESNE TANIMLAYICI,
                     bir ürüne eklenecek.   ObjectDescriptor İ STEĞE BAĞLI,
barkod verileri          OCTET Dİ ZGİ Sİ , --
Wineco kılavuzunun 29. bölümünde belirtilen -- ög enin barkodunu temsil eder. stok tükendi
BOOLEAN DEFAULT FALSE,
-- Ög e için stok, rapor edilen dönemde boyunca herhangi bir zamanda tükenirse DOĞRU gönderin.
minimum stok seviyesi GERÇEK, maksimum stok seviyesi GERÇEK, ortalama stok seviyesi GERÇEK

```

-- Dönem boyunca minimum, maksimum ve ortalama seviyeleri normal hedef stok seviyesinin yüzdesi olarak verin-- }

wineco-items NESNE TANIMLAYICI ::=
{ common-iso-itu-t internationalRA(23) set(42) set-vendors(9) wineco(43) stock-items (0)}

Ş ekil 22 (bölüm 2): Temel ASN.1 türlerinin kullanımının gösterimi

2 Yerleşik türlerin tartışılması

2.1 BOOLE türü

(Ş ekil 22'deki "stok tükendi" bölümune bakın). Burada eklenecek bir şe y yok. Bir "BOOLEAN" türü, doğru ve yanlış olmak üzere bariz iki soyut değer sahiptir, ancak değer er göstergesinin tamamı büyük harflerle yazılmış "DOĞRU" veya "YANLIŞ" sözcükleri olduğu una dikkat edin. Büyük harf kullanımını, ASN.1'deki (neredeyse) tüm yerleşik adların tamamının büyük harf olduğu gerçeğe ıyle tutarlı veya ASN.1'in bu değer er referans adlarını gerektirdiği gerçeğe ıyle tutarsız olarak kabul edebilirsiniz. küçük harfle başı la! ASN.1 her zaman kendi kurallarına uymaz!

2.2 INTEGER türü

(Bkz. "rapor edilen gün sayısı", ş ekil 22). Bu örnek, ş ekil 13'te gördüğümüz basit "INTEGER" kullanımından biraz daha karmaşık看起来!

Buradaki örnek, ayırt edici değerler olarak adlandırılan değerlerleri içerir. Bazı erken ASN.1 spesifikasiyonlarında (SAYILANDIRILMIŞ) (örneğin, ti) ayırt edici değerler listesiyle birlikte "INTEGER" türünü kullanırdı, bugün ise "SAYILANDIRILMIŞ" ifadesini kullanırlardı. Aslında, sözdizimi oldukça benzer görünebilir, bu nedenle ş ekil 13'teki örneğin karşılığı inşası şe ekilde yazabiliriz:

tamsayı türü

Sadece INTEGER kelimesi, güzel ve basit!; ve/veya

Seçkin bir değer listesi ekleyin; ve/veya

Bir aralık belirtimi ekleyin (alt tipleme); sonra

Aralık belirtimine bir uzantı işaretçisi ve istisna belirtimi koyun! (Yine karmaşıklaşır!)

aciliyet TAM SAYI {yarın (0),

üç günlük (1), hafta
(2)} VARSAYILAN hafta

Bununla birlikte, burada bazı önemli farklılıklar fark etmek önemlidir. "INTEGER"den sonra gelen listenin varlığı tamamen isteği e bağılıdır ("SAYILANDIRILMIŞ" için gereklidir) ve listenin varlığı, türdeki soyut değerler kümescini hiç bir şekilde etkilemez.

Aşağıdaki iki tanım neredeyse eşdeğerdir:

My-integer ::= INTEGER {yarın(0), üç günlük (1), hafta(2)}

ve

Tamsayım ::= TAM SAYI yarın
Tamsayım ::= 0 üç günlük Tamsayım ::= 1
hafta Tamsayım ::= 2

Fark, ASN.1 kapsam kurallarında yatkınlıkta. İlk örnekte "yarın" vb. adlar, modül içinde yalnızca bir kez atanabilen değer er-referans-adalarıdır ve bu modül içinde bir tamsayı değerinin gerekliliği herhangi bir yerde kullanılabilir (aslında, bir sayı olarak bile) kullanılabilir. numaralandırmada veya başka bir ayırt edici değer er listesinde veya bir etikette - ancak tüm bu kullanımlar alış習慣 olurdu!) ve modülün başındaki bir EXPORTS ifadesinde görünebilir. Öte yandan, ilk örnekte, "yarın" vb. adları dışarı aktarılabilir, ancak er ayırt edici değer er listelerinde (aynı veya farklı değerlerle) görünebilir veya gerçek ekten de tamamen farklı bir değer er içinde er-referans adları olarak görünebilir. tip. İlk örnekteki "yarın" adı, YALNIZCA "Tamsayım" türü tarafından yönetilen değer er notasyonunda göründüğü içinde, örneğin "Tamsayım"ın sıfır değerini belirleme anlamına gelir. Bu türdeki bir dizi değer eseri için "VARSAYILAN" değer er.

Ayırt edici değer er listelerinde sayıları artan düzende kullanıyor olsak da, bunun için bir gereklilik olmadığı inanı da dikkat edin - sıra önesizdir ve ortaya çıkan tanımları etkilemez.

Pozitif bir tamsayı değer eri için değer er gösterimi olarak bir ondalık sayının kullanılabilmesiini gördük. Negatif değerler, örneğin:

eksi-iki TAM SAYI ::= -2

ancak "-0" yazmanıza izin verilmeyen ve "INTEGER" türü için değer er gösterimi olarak herhangi bir ikili veya onaltılık notasyon geçerli değildir.

"INTEGER" için soyut değerler kümesci nelerdir? ASN.1 belirtiminin erken bir taslağı, BER kodlamalarının dayattığı kısıtlamalara dayalı olarak ASN.1 tamsayılarının maksimum ve minimum değerlerini filen belirtiyordu. Ancak yapılan bir hesaplama, saniyede bir terabit hızında çalışıp bir iletişim hattıyla en büyük veya en küçük değer eri iletmenin yaklaşık 100 milyon yıl sürecekini gösterdi! ASN.1 tamsayıları "etkin bir şekilde sınırsızdır". (Ve daha yeni PER kodlamalarında, bir tamsayı değerinin boyutunda herhangi bir sınır yoktur.)

Bu, sonraki bir bölümde daha uygun bir şekilde ait olan bir tartışmanın başlangıcını gündeme getiriyor - keyfi olarak büyük tamsayıları işlemek için uygulama kodunuzu gerçek ekten yazmanız gerekiyor mu? Şekil 13'teki "bildirilen gün sayısı"na tekrar bakarsak, ayırt edici değer er listesinin ardından "(1..56)" yazısını görüyoruz. (Bu, ayırt edici bir değer er listemiz olsa da olmasa da mevcut olabilir).

Bu, bir alt tür kısıtlamasına iliş kin ilk örneğ imizdir - tamsayıımızın aralığı inı sınırlayan veya onu altkümeleyen bir notasyon. Bu durumda, uyumlu bir göndericinin göndermesine izin verilen tek değ erlerin 1 ila 56 aralığıındaki değ erler olduğunu söylemek ve bir uygulayıcının bu alan için yalnızca bir bayt ayırması gerektiği açıktır. Alt tip notasyonu (tamsayı türü için olduğu kadar diğer er türler için) hakkında daha kapsamlı bir tartışma daha sonra görünür, ancak bir tamsayı aralığı inin bu basit kısıtlaması, bu gösterimin açık ara en yaygın kullanımıdır. Uygulama tasarımcılarının, yapabildikleri her zaman "INTEGER" türlerine bunun gibi bir aralık kısıtlaması koymaları ve uygulayıcıların keyfi olarak büyük tamsayıları gerçekten iş lemesini beklediklerini yorumda açıkça belirtmeleri önerilir. Bununla birlikte, bir uygulayıcı olarak, aralık kısıtlaması ve açıklayıcı metin olmadan basitçe "INTEGER" görürseniz, alacağınız en büyük değ erin dört sekizli bir tamsayı olacağının genellikle güvenli bir varsayımdır.

Son bir nokta: Ayırt edici değ erleri tanımlamaya yönelik sözdiziminin numaralandırmaları tanımlamaya yönelik söz dizimiyle benzerliği i kafa karıştıracı olabilir. Ayırt edici değ erlerin tanımı, türdeki soyut değ erler kümesini veya kodlanma biçimlerini hiç bir şekilde değ iş tırmadiğinden, sürüm 2'ye geçişte hiç bir zaman "geniş letilebilirlik" sorusu yoktur - ek ayırt edici değ erler eklenirse, bu sadece göstergimsel bir kolaylıktır ve hattaki bitleri etkilemez. Bu nedenle, üç nokta geniş letilebilirlik işaretçisi (numaralı türdeki liste için kullanılabilir), ayırt edici değ erler listesinde ne gerekli ne de buna izin verilir (daha sonra göreceğiz gibi, bir aralık kısıtlamasında görünebilmesine rağmen men).

2.3 SAYILANDIRILMIŞ tip

(Bkz. ş ekil 13'teki "aciliyet" ve ş ekil 22'deki "gecikme nedeni"). Önceki tartışmalarımıza eklenecek çok az ş ey var. Yuvarlak parantez içindeki sayılar 1994'ten önce gerekliydi ve 1994'ten sonra isteğe bağlıdır. Tür, tam olarak ve yalnızca listelenen adların her birine karşı ilişkili gelen değ erlerden oluşur.

Sayılar başlangıçta geniş letilebilirlik sorunlarından kaçınmak için mevcuttu - sürüm 2 yeni bir numaralandırma eklediyse, bunun orijinal numaralandırmaları belirtmek için kullanılan (kodlamalarda) değ erleri etkilememesi öneği idi ve bunu sağlamanın en kolay yolu uygulamanın izin vermesiydi. Tasarımcı kullanılacak numaraları listeler. 1994 sonrası, geniş letilebilirlik daha belirgindir ve şunları görebiliriz:

Aciliyet türü ::= HAYIRLANMIŞ {yarın,

üç gün, hafta,

...

-- Sürüm 1 sistemleri başka herhangi bir değ er almalıdır -- "hafta" anlamına gelir. ay}

numaralandırılmış

Bir uzanti işaretçisine sahip olabilir.

1994'ten önce gerekli olan kodlamalar için sayılar, 1994'ten sonra isteğe bağlıdır.

Burada sürüm 2'de "ay" eklenmiş tir, ancak sürüm 1 ilk kez belirtildiğiinde sürüm 1 sistemlerine getirilen gereksinim aslında bu tür değ erlerin sistemlerin "ay" "hafta" olarak ele alacağının anlamına gelir. Bu, sürüm 1 sistemlerinden istediginiz özel durum iş leme hakkında derinlemesine düşünmenin önemini göstermektedir. Bunun yerine sürüm 1 spesifikasyonu "herhangi bir bilinmeyen numaralandırmayı yarın olarak ele al" deseydi, sürüm 2'ye "ay" eklemenin etkisi daha az tatmin edici olabilirdi! Bu durumda, bir istisna belirtimi kullanmak yerine, yorumda istisna iş leme davranışını üç noktadan sonra vermemi seçtiğimizde dikkat edin - bu, özellikle istisna iş leme bu alana özgüyse, oldukça tatmin edicidir. Uygun istisna iş leme seçimi, Bölüm 7, 2.6'da daha ayrıntılı olarak ele alınmış tir.

Son olarak, gerçekten garip olmak istiyorsanız, bazı numaralandırmalar için sayı girebilir, diğer erleri için giremezsiniz. Şanslısanız, sonuç yine de yasal olacaktır! Böyle aptalca şeyle yapmak istiyorsanız gidin ve ASN.1 spesifikasyonunu okuyun, bu kitap size yardımcı olmayacaktır!

Gerçek

2.4 GERÇEK tip

(Bkz. Şekil 22'deki "min. stok düzeyi" vb.). "REAL" tipi için tip gösterimi Şekil 22'de verilmiş tir. Bu tek seçenek.

Matematiksel olarak eşit olsa bile birbirinden farklı iki soyut değer er kümesi, Taban 10 ve Taban 2. Değer er gösterimi mantis, taban (2 veya 10) ve üs için virgülle ayrılmış bir tamsayı listesidir. Ayrıca ARTI-SONSUZ ve EKSİ-SONSUZ.

Değer er gösterimi biraz meraklı. İşte gerçek tür için bazı değer er notasyonu örnekleri:

```
v1 GERÇEK ::= {mantis 314159, taban 10, üs -5} v2 GERÇEK ::= {mantis 3141590, taban 10, üs -6} v3
GERÇEK ::= {mantis 1, taban 2, üs -1} v4 GERÇEK ::= {mantis 5, taban 10, üs -1} v5 GERÇEK ::= 0 v6 GERÇEK ::=
{mantis 0, taban 2, üs 100} v7 GERÇEK ::= {mantis 0, taban 10, üs 100}
```

V5 dışında bunların hepsinin virgülle ayrılmış üç sayıdan oluşan an listeler olduğu una dikkat edin. (Virgülle ayrılmış listeler ASN.1 değer er notasyonunda sık sık ortaya çıkar ve REAL tipi için seçilmiş tir, çünkü bir ASN.1 aracı, vali henüz tanımlanmamış bir tip-referans adı olduğuunda değer er notasyonuyla karşılaşıbılır ve aracın bir gösteriminin sonunu bulmanın basit yolu). $\{x, y, z\}$ ile tanımlanan matematiksel değer er ($x \times y / z$), ancak y'nin yalnızca 2 ve 10 değer erlerini almasına izin verilir.

Ayrıca, aşağıdaki değer er gösterimiyle açıkça dahil edilmiş (ve özel olarak kodlanmış) iki değer er vardır:

ARTI-SONSUZ
EKSİ-SONSUZ

Yine, hepsi büyük harfler. "REAL" ilk tanıtıldığından, "OVERFLOW", hatta "PI" vb. gibi ek özel "değer erler" eklenmesi tartışıldı, ancak bu asla olmadı.

Gerçek uygulama belirtimlerinde "REAL" tipi nadiren kullanıldığından, gerçekten bilmeniz gereken tek şebeke budur. "GERÇEK" türündeki tartışmanın geri kalanı biraz akademiktir ve sağlığına "gerçek" bir zarar vermeden bunu atlayabilirsiniz! Ancak v1'den v7'ye kadar olanlardan hangisinin aynı soyut değer eri ve hangilerinin farklılığından bilmek istiyorsanız okumaya devam edin!

İşte simden soyut değer erlerin (matematiksel) gerçek sayılar olmasını bekleyebilirsiniz, ancak matematiksel değil ilimli olanlar için yalnızca rasyonel değer erler dahil edilmiş tir.

Biçimsel olarak, tür, iki soyut değer er kümesi içerir; bir küme, 10 tabanını kullanan sonlu bir gösterime sahip tüm sayıları içerebilir ve diğer bir küme, 2 tabanını kullanan sonlu bir temsile sahip tüm sayıları içerebilir. , sonraki değer erler, öncekiinin katı bir alt kümesidir, ancak önceki, ikinci kümede olmayan değer erleri içerebilir. Tüm ASN.1 kodlama kurallarında, "REAL" için ikili kodlamalar vardır ve ayrıca ISO standartı ISO 6093'te belirtildiği gibi ondalık kodlamalar vardır. Bu standart, değer eri temsil edecek bir karakter dizisini belirtir ve bu daha sonra ASCII kullanılarak kodlanır. Bu kodlamalara bir örnek:

56.5D+3

ancak ISO 6093 birçok seçenek iç erir!

"REAL"deki soyut değer erler kümesini yalnızca 10 tabanlı veya yalnızca 2 tabanlı küme olacak şekilde kısıtlamak (1994 sonrası) mümkün, bu da uygulama tasarımcısına ikili veya ondalık kodlamaların kullanılıp kullanılmayacağı konusunda etkili bir şekilde kontrol sağlar. . Türün sınırsız olduğu durumlarda, matematsel olarak eşit olan 2 tabanlı değerlerden 10 tabanlı bir değerere farklı uygulama semantikini koymak teorik olarak mümkün, ancak muhtemelen kimse yeterince aptal olmaz! (Aslında "REAL" zaten gerçek protokollerde pek kullanılmaz).

Ancak bu tartışmayı toplamak için... yukarıdaki v1'den v7'ye kadar olan değerlerle baktığımızda, aşağıda aynı satırda listelenen değer referans-adlarının aynı soyut değerler için değer gösterimi olduğuunu ve farklı satırlardakilerin ise farklı soyut değerler için isimler:

v1, v2 v3
v4 v5, v6
v7

(V5, V6'ya eşittir çünkü V5, base2 sıfır değerini temsil edecek şekilde tanımlanır.)

2.5 BIT Dİ ZGİ Sİ türü

(Bkz. "versiyon", § ekil 22). Bit dizisi türünün iki ana kullanımı vardır. İki, türle ilişkili adlandırılmış bitlerin bir listesine sahip olduğu umuz "sürüm" için verilendir. İkiinci ve en basit tip gösterimdir:

BIT Dİ ZGİ Sİ, sürüm anlaşması için bir bit haritasını desteklemek için genellikle adlandırılmış bitlerle birlikte kullanılır.

Bi TDİ Zİ Sİ

Beklediğimiz gibi, bunun tamamen büyük harf olduğuunu unutmamak, ancak beklemeyebileceğimiz gibi, türün adı (etkili bir tür-referans-adı) bir boş luk içerir! Alana sadece izin verilmez, gereklidir! ASN.1 yine kendi kurallarını yıkıyor!

Şekil 22'ye birazdan döneceğiz. İki simlendirilmiş bitler listesinin olmadığı daha basit durumu ele alalım.

Bir dizinin (diyelim ki) bir alanı basitçe "BIT Dİ ZGİ Sİ" olarak tanımlanırsa, bu durumda, bir protokoldeki herhangi bir alana semantik uygulanması gerektiğiinden, bu yetersiz tanımlanmış bir protokolün işareti olabilir. Daha fazla açıklama içermeyen "BIT Dİ ZGİ Sİ", ASN.1 belirtimlerinde yasal olarak "deliklerin" bırakılabileceği, ancak bir bütün olarak belirtimin zararına olacak şekilde birkaç yoldan biridir.

Daha sonra, herhangi bir "deliği in" kaldıgında yerde, bir iletişim imzayı içeriğinde deliği içeriğinin açıkça tanımlayacak alanlar sağlar. Lamanın ve iletişim imzaların tüm ortakların tüm tanımlamaları (ve sonuçta ortaya çıkan sonucu) anlamasını sağlar. Lamanın önemli olduğuunu göreceğiz. Deliği içeriğinde veya bilinmeyen bir tanımlayıcı üzerindene yapılması gerektiğiini bileyebilir. ASN.1, bu tür "delikleri" ve ilişkili tanımlamayı sağlar ve kendi "deliklerinizi" büyütmek için "BIT STRING" kullanmak iyi bir fikir değildir (ancak bazı insanlar yapar)!

tanımlanır. Adlandırılmış bitlerin olmayan BIT Dİ ZGİ Sİ de taş imak ve özellikle gizleme veya imza amaçları için ş ifrelemeleri taş imak için So ... meş ru bir kullanıma sahiptir. Ancak bu durumda bile, genellikle uygulanacak güvenlik algoritmasını açıkça tanımlamaya ve belki de kullanımında olan belirli anahtarlarla dolaylı olarak baş vurmaya ihtiyaç vardır. BIT STRING veri türü, protokollere güvenlik geliş tirmeleri sağlayanlar için (meş ru olarak) önemli bir yapı taşıdır, ancak genellikle onunla birlikte daha fazla veri taşınır.

Adlandırılmış bitlerin olmayan BIT Dİ ZGİ Sİ de sıklıkla ş ifrelenmiş bilgileri taş imak için daha karmaşık bir yapının parçası veya daha karmaşık bir yapısı olarak kullanılır.

Şekil 13'teki "versiyon" için olduğu gibi adlandırılmış bitlerle BIT Dİ ZGİ Sİ'nin kullanımı yaygındır. Kırık parantez içindeki adlar, yalnızca bit dizisinin bitleri ve ilişkili bit sayısı için adlar sağlar. Adlandırılmış bir bit listesinin varlığıının (tamsayılar için ayırt edici değil erlerde olduğu gibi) türü etkilemediğine dikkat etmek önemlidir. Liste hiç bir şekilde bit dizisinin olası uzunluğuunu kısıtlamaz ve bitlerin sırayla adlandırılması gerekmektedir.

ASN.1, "baş taki bit"ten "sondaki bit"e kadar "bit sıfır" olarak bahseder. Kodlama kuralları, kodlama sırasında bir bit dizisi türünün "öndeki biti" "sondaki bit" ile seziklilere eşler.

(BER - keyfi olarak, zıt kuralı seçebilirdi - baş taki bitin kodlamanın ilk seziklisinin en anlamlı bitine yerleş tirileceğini belirtir, vb.)

Bu bit adları nasıl kullanılır? Her zaman olduğu gibi, insan tarafından okunabilir metin tarafından belirli bitlere referans için bir tutamaç sağlayabilirler. Bununla birlikte, değil er notasyonunda da kullanılabılır.

Bir bit dizisi için açık (ve en basit) değil er gösterimi, değil er ikili olarak belirtmektir, örneğin:

'101100110001'B

Değil er dört bitin katıysa, onaltılık kullanımına da izin verilir:

'B31'H

(ASN.1 onaltılık gösterimde yalnızca büyük harflere izin verildiğine dikkat edin.)

Bununla birlikte, adlandırılmış bitler mevcutsa, o zaman süslü parantezler içindeki bit adlarının virgülle ayrılmış bir listesi olan ek bir değil er gösterimi mevcuttur (örneğin, Şekil 22'deki "sürüm"ün "VAR SAYILAN" değil erine bakın). Tanımlanan değil er, listelenen her bit adı için bitin bire ayarlandığı ve diğer tüm bitlerin sıfır ayarlandığı bir değil erdir.

Uyarı okuyucusu (yine yaptım!), bu ifadenin bir bit dizisi değil eri tanımlamak için yeterli olmadığıını anlayacaktır, çünkü değil erde kaç (varsayı) sondaki sıfır bitin bulunduğuunu belirsiz bırakır.

Bu nedenle, eğer bit dizisinin uzunluğu kısıtlanmamışsa, böyle bir "değil er notasyonu" kullanımını gerçekten bir değil er tanımlamaz - bir dizi değil er tanımlar! Hepsi aynı bir bite sahip, ancak sıfırdan sonsuza kadar sıfır bit var!

1986 civarında yayınlanan ASN.1 belirtimleri, bazı gelincik sözcüklerle (farklı sürümlerde biraz değil iş tirilmiş) bu sorunu çözmektedir: "Adlandırılmış bir bit listesi varsa, sondaki sıfır bitlerin anlamsal bir önemi olmayacağı"; daha sonra "kodlama kuralları, kodlanan değil erlere (veya bunlardan) sondaki sıfır bitleri eklemekte (veya çıkarmakta) serbesttir" ile artırılmıştır!

Bu sorun, normal BER için büyük bir sorun değil ildir; burada bazı değer erlerin "VARSAYILAN" değil erle tam olarak eş leş ip eş leş medig i konusunda ş üphe olup olmadığı önemlidir, ancak daha sonra açıklanan kanonik kodlama kurallarında daha çok önemlidir.

Adlandırılmış bitlerin en yaygın kullanımı, ş ekil 13'te gösterildiği gibi bir "sürüm" haritası gibidir. Burada bir uygulamaya, destekleyebildiği sürümlere karşı ilişkili gelen bitleri ayarlaması talimatı verilir ve - tipik olarak - bazı alicinin tam olarak bir bit (orijinal mesajda ayarlananlardan biri) ayarlayacağı veya bir tür ret mesajı göndereceği cevap mesajı.

Resmi/İleri düzey tartışma

NOT — Çok u okuyucu bir sonraki kısmı atlmalıdır! Daha az sorunu olan OCTET STRING'e geçin! Okumak konusunda ısrarcısanız, lütfen ş ekil 999'u tekrar okuyunuz!

Son 15 yılda ASN.1 spesifikasiyonlarında, adlandırılmış bitlerle "BIT STRING" tanımlarıyla ilişkili birçok farklı metin bulunmaktadır. Çok u arzu tarafından kısıtlandırılmıştır.

a) belirtilenleri gerçekten değil iş tirmemek veya en azından mevcut konuşlanır adlandırılmış uygulamaları bozmamak; ve

b) gerçekten yapmasa bile yukarıda a)'yi ima etmemek gibi görünen çok miktarda metin eklememek!

Sonuç olarak, uyanık ve akıllı bir okuyucu(!) olarak, okuduğunuza spesifikasiyonun eski haline ve/veya insanların size "ASN" demekte ısrar edip etmediklerine bağlı olarak, aşağılardan pekala sorun yaşayabilirsiniz. .1 Uzman!"

Adlandırılmış bitlere sahip BIT STRING, böyle bir türün kesin soyut değerlerinin ne olduğunu hakkında ilginç sorunlar ortaya çıkarır:

BU TÜR SORULARI GÖRMEYİ N, önemli değer iller!

ASN.1 Standardı, adlandırılmış bir bit listesinin (ve böyle bir listenin kapsamının) varlığıının, tanımlanmakta olan türdeki soyut değerler kümesi üzerinde hiçbir etkisinin olmadığıını ima ediyor gibi görülmektedir. Bununla birlikte, soyut değerler, uygulama tasarımcılarının, her değer erin farklı bir kodlamaya sahip olacağı güvencesiyle ve kanonik kodlamalar için her değer er için tam olarak bir kodlama olacağı güvencesiyle, kendileriyle farklı uygulama semantiklerini ilişkilendirmelerini sağlama konusunda bir sorun yoktur.

(İhtiyaçlı açıklama aşağıdadır!) Spesifikasiyon, "uygulama tasarımcılarının, farklı (uygulama) anımlarının ... yalnızca sondaki sıfır bitlerinin sayısında farklılık gösteren ... değerlerle (adlandırılmış bitlere sahip türlerin) ilişkilendirilmemesini sağlama gerekliliğiini" belirtir. Bunun aslında söylediğimizde, "değerlerin farklı soyut değerlerin aslında tek bir soyut değer er olduğu"udur.

Geriye kalan tek sorun, bu tür soyut bit dizisi değerlerinin kodlama kuralları tarafından nasıl temsil edilmesi gereklidir. Standart rehberlik sağları: "kodlama kuralları, kodlanan veya kodu çözülen değerlerle (veya bunlardan) keyfi olarak birçok sondaki sıfır biti eklemekte (veya kaldırırmakta) serbesttir". Belki de bunu ifade etmenin en iyi yolu değer il, ama prensipler açık:

adlandırılmış bir bit listesi mevcut olduğuunda, yalnızca sondaki sıfır bitlerinin sayısında farklılık gösteren farklı bit modellerine karşı ilişkili gelen yalnızca bir soyut değer erimiz olur;

kodlama kuralları (elbette!) bu soyut değerleri istedikleri gibi temsil etmeyeceğinden, ancak bir seçenek, yalnızca sondaki sıfır bitlerinde farklılık gösteren bu bit modellerinden herhangi birini kodlamaktır.

Her bir soyut değil için tek bir kodlama sağlama iddiasında olmayan BER için kurallar, kodlamada keyfi olarak çok sayıda sondaki sıfır bitine izin verir. (Bu oldukça soyut(!) sorun ilk kez anlaşıldığında, mevcut uygulamaları bozmamak için buna izin verme kararı gereklidi.) Mevcut BER uygulamaları, adlandırılmış bir bit dizisi türündeki bir değil erin kodlanması genellikle sondaki sıfır bitleri içerecektir. liste.

Bununla birlikte, PER dahil kanonik kodlama kuralları için, tek bir kodlama gereklidir ve ilk bakışta, bu tür kodlama kurallarının kodlamada hiçbir zaman sondaki bitlere sahip olmadığıını söylemek iyi bir çözüm gibi görünür.

Ancak, bit dizisinde soyut düzeyde uzunluk kısıtlamaları varsa, kodlama seçimi (ve aslında kodlama için kullanılacak olan aynı semantik e sahip soyut değil erler kümesinden kesin soyut bit dizisi değil erinin seçimi) karmaşık olabilir.

Konu daha da karmaş ıktir çünkü BER ile ilgili kodlama kurallarında uzunluk kısıtlamaları "görünmez"dir - kodlamayı etkilemez! PER'de görünür olabilirler veya olmayabilirler!

Tüm bunların sonucu, BER'in kanonik sürümlerinde, sondaki sıfır bitlerinin hiç bir zaman bir kodlamada iletilememesidir, ancak uygulamaya iletilen değil erin, tatmin edici olmasını sağlamak için yeterli sıfır bitinin (gerekli minimum) eklenmesi gereklidir. uygulanmış olabilecek herhangi bir uzunluk kısıtlaması. (Bu tür kısıtlamaların, kodlama kuralları tarafından görünür olsun veya olmasın - etkilensin - uygulama ve Uygulama Programı Arayüzü -API- kodu tarafından görülebileceği varsayılmaktadır.)

(Bazı) uzunluk kısıtlamalarının PER-görünür olduğunu PER, bunu biraz değ iştir: iletilen şey her zaman PER-görünür kısıtlamalarla tutarlıdır - bu nedenle (minimum sayıda) sondaki sıfır bitleri, eğer bir tatmin için gerekliyse aktarımında bulunur. uzunluk kısıtlaması Bu nedenle, PER-visible kısıtlamaları uygulanmaması koşuluyla, kodlama uygulamaya dege iş meden teslim edilebilir, aksi takdirde kanonik BER kuralları geçerli olur - uygulama, kısıtlamaların izin verdiği bir dege er alır ve bununla aynı uygulama semantiği ini taşıır. doğrudan iletilen kodlamadan elde edilir.

Ve buraya kadar okuduysanız, bahse girerim okumamış olmayı diliyorsunuz! Hepsi işe yarıyor ama basit değil!

Bunun gibi sorunlar normal uygulama tasarımcısını etkilemez - sadece bariz şayeleri yapın ve hepsi işe yarayacaktır, ayrıca iyi bilinen kurallara uyan normal uygulayıcıyı da etkilemezler: bariz kodlamayı kodlayın; kod cözmeye liberal olun.

Ancak bu sorunlar, gelen materyalin hatalı olduğunu algılanırsa "katı teşhis" seçeneği sunan alet satıcıları için önemlidir. Bu gibi durumlarda neyin "hatalı" olduğunu dair çok kesin bir açıklama gereklidir!

2.6 OCTET STRING türü

(Bkz. "barkod verileri", ş. ekil 22). Bir kez daha "OCTET" ile "STRING" arasında bir boşluk olması gerekiyor!

Ve bir kez daha, bir sekizli dizi "herhangi bir şeysi taş ımak" için cazip bir adayı - sınırlanmış bir delik.

(Ama kendinizi kandırmayın!) Yine, tanımlama alanları ve istisna isleme ile desteklenmediği sürece uygun değildir.

İstisna İş İmre ile desteklenmediği sürece uygun değil. ASN.1 daha iyisini sağlar

OCTET STRING türü basittir - ancak onu kullanmayın!

bir şeilde desteklenen bir "deliğ" temsil eder ve önceden üretilmiş bir "delik" kullanmak daha iyidir - daha sonra bakın!

"delikleri" desteklemek için mekanizmalar.

Ş ekil 22'de gösterilen durumda, sekizli dizenin kesin içeriğ i (umarız!) "wineco kılavuzunun 29. bölümünde" iyi bir ş ekilde belirtilmiş tir. Ancak bu belirtim çok genel değil ildir. Amaç, açıkça, ASN.1'in dışında bazı kodlamalar kullanarak ek tanımlama bilgileri için bir kap sağlamaktır. Genel olarak ve zamanla, tasarımcının bu oktet dizisinde taş ımkı isteyebileceğ i çesitli tanımlama biçimlerinin bir dizi farklı kodlaması olabilir ve yine "bu bir barkod versiyonudur" diyen ek tanımlama alanlarına ihtiyaç olduğunu görüyoruz. 1" - veya başka bir şeyle ve bu kararları "29. bölüm" fiziksel olarak bağlamak yerine "bugün böyle kodlanıyor". Bir kez daha "delikleri" tartıştığımızı görüyoruz.

Özet olarak (ama yine Ş ekil 999'a bakın!), ne yaptığınızı gerçekte bilmiyorsanız ve gerçekte "kazmak" istemiyorsanız, uygulama belirtimlerinde OCTET STRING veya BIT STRING (sürüm bit haritaları dışında) alanlarına sahip olmak KÖTÜ BİR ŞEYDİR. R. kendi deliğiniz. Ama elbette, belki de yaparsın!

OCTET STRING için değer er gösterimi, daha önce bit dizisi için gösterildiği gibi her zaman onaltılık veya ikili ş eklindedir. Sonuç, sekiz bitin tam katı değil ise, sonuna sıfır bit eklenir.

2.7 NULL türü

(Bkz. ş ekil 13'teki "depo"). Resmi olarak NULL, yalnızca bir değer eri olan bir türdür. Bu değer er için değer er gösterimi oldukça kafa karıştırıcıdır:

NULL için her şeyle biliyorsunuz - bir yer tutucu: sorun yok.

HÜKÜMSÜZ

Yine, tamamı büyük harflerle, ilk başta küçükük harf beklenebilir.

Normal kullanım, ş ekil 13'teki gibidir - burada bir TLV sağlamak için bir türde ihtiyacımız vardır (varlığı veya yokluğu bazı anlamlar taşıır), ancak türle birlikte taşınamak ek bilgi yoktur. NULL, ASN.1 kurslarında genellikle "yer tutucu" olarak anılır.

2.8 Bazı karakter dizisi türleri

(Ş ekil 22'de "ek bilgi" ve ş ekil 13'te "isim" (iki kez) bölümne bakın). Şimdide kadarki örneklerde, "PrintableString" (en eski ASN.1 taslaqlarında mevcut), "VisibleString" (kullanımdan kaldırılan "ISO646String" eş anlamlısı) ve "UTF8String" (1998'de eklenen) ile karşılaştırınız. Birkaç tane daha var.

Tamamı büyük olmamalarına rağmen, bunlar (ve diğer karakter dizisi türü adları) yaklaşık 1988/90'dan beri ayrılmış sözcüklerdir (kendi türleriniz için kullanamayacağınız adlar). ASN.1'in ilk tasarımcıları (haklı olarak!) karakter dizisi türlerinin ve adlarının biraz "ad hoc" olduğunu düşündüler ve onlara biraz azaltılmış bir statü verdiler!

Aslında, en eski ASN.1 belirtiminde, "Kullanışlı Türler" kavramı vardı, yani saf insan dili yerine ASN.1 gösterimi kullanılarak tanımlanan türler ve bunların tümü karışık büyük/küçük harf kullanıyordu. Karakter dizisi türleri başlangıçta "Yararlı türler" olarak dahil edildi ve etiketli bir OCTET DİZİSİ olarak tanımlandı. Bugün (yaklaşık 1990'da ayrılmış kelimeler haline geldiklerinden beri), aşağı yukarı INTEGER veya BOOLEAN'inkine eşit bir statüye sahip oldukça temel türler olarak kabul ediliyorlar.

"PrintableString" dē erlerindeki karakter kümesi, ASN.1'e "bāg l̄d̄r" ve kabaca eski teleks karakter kümesi artı küçük harflerdir. TLV'nin "V" kısmındaki BER kodlaması ASCII kodlamasıdır, bu nedenle "VisibleString" (aşağıda) üzerinden ayarlanan azaltılmış karakter, bir dizi uygulama özell̄ī i "PrintableString" kullanmasına rağmen gerçekten kullanışlı dē ildir.

"VisibleString" dē erlerindeki karakter kümesi, yalnızca yazdırılan ASCII karakterleri artı "boş luktur". TLV'nin "V" kısmındaki BER kodlaması elbette ASCII'dir.

"UTF8String"deki karakter grubu, Mısır hiyerogliflerinden en derin Amazon ormanlarında tahtaya oyulmuş ş eylere ve zamanı gelince Mars'ta bulacağımız ş eylere kadar uygun ş ekilde artış tırılmış ve belgelenmiş herhangi bir karakterdir (ASCII kontrol karakterleri dahil). Karakter başı ina BER (ve tür azaltılmış bir karakter kümesiyle sınırlanılmamışsa PER) kodlaması dē işken uzunluktadır ve ASCII karakterler için karakter başı kodlamanın bir sekizli olduğu ve tüm karakterler için üç sekizliye uzanan "nice" özell̄ī ine sahiptir ş imdiye kadar artış tırıldı ve belgelendi ve galaksinin tüm dillerini orada bulduktan sonra karakter başı ina en fazla altı sekizli olacak! Karakter seti konularına "ilişkin" olanlar, "Unicode" adını tanyabilir. UTF8, iletişim ve karakter bilgilerinin depolanması için (1999 dolaylarında) son derece popüler hale gelen Unicode'un tamamını (ve daha fazlasını) kapsayan bir kodlama şemasıdır. Tavsiye: Yeni bir protokol tasarlıyorsanız, bunu yapmamak için çok iyi bir nedeniniz yoksa, karakter dizisi alanlarınız için UTF8String kullanın.

2.9 NESNE TANIMLAYICI tipi

(Bkz. ş ekil 22'deki "item" ve "wineco-items" ve ş ekil 21'deki modül tanımlayıcıları.) Nesne tanımlayıcı tipinin dē erleri bu kitabın başından itibaren kullanılmış ve tanıtılmıştır. Ama yine de bu türden ayrıntılı bir tartışmayı daha sonraki bir bölümde erteleyeceğiz!

NESNE TANIMLAYICISI - belki de dē er tüm temel ASN.1 türlerinden daha fazla kullanılır - birçok yoldan biraz ad alanı elde edebilirsiniz, ancak buna rağmen ihtiyacınız yok

NESNE TANIMLAYICI türü, tüm ASN.1 türleri arasında en çok kullanılanı olma iddiasında bulunabilir (elbette SEQUENCE, SET ve CHOICE yapıcları hariç). ASN.1 tabanlı bir belirtimde dünya çapında kesin tanımlamaya ihtiyaç duyulan her yerde, nesne tanımlayıcı tipi kullanılır.

Dē er gösteriminin görünürdeki ayrıntılarına rağmen, tip nesne tanımlayıcısının dē erlerinin kodlaması aslında çok derli topludur (dē er gösteriminde bulunan insan tarafından okunabilecek adlar kodlamada görünmez). Bir nesne tanımlayıcı dē erinin ilk bileşenleri için, adların tamsayı dē erlere eşlenmesi "iyi bilinir" ve herhangi bir dē er notasyonundaki sonraki bileşenler için karşı ılık gelen tamsayı dē eri mevcuttur (genellikle yuvarlak parantez içinde).

Temel ad uzayı, hiyerarşik olarak tahsis edilmiş bir ağ yapısıdır; üst düzey yolların tahsisinden sorumlu küresel yetkililer ve alt düzey yaylardan giderek daha fazla yerel makam sorumludur.

Sizin (uygulama tasarımcısı olarak) nesne tanımlayıcı ad alanından dē erler tahsis edebilmeniz için, bu ağ açıtan "asılmanız" yeterlidir. Nereden "asılı" olduğu unuz gerçekten önemli dē il (gerçi siz zirveye yaklaşışıkça dē erlerinizin kodlamaları daha kısa olacaktır ve uluslararası kuruluşları nerede "asılı kaldıkları" konusunda hassas olma eğilimindedirler!).

Standart oluşuran bir grup, özel bir şirket ve hatta bir birey için, bu ad alanının bir kısmını elde etmek için bir dizi mekanizma vardır ve bunların çoğu hiçbir idari çaba gerektirmez (muhtemelen zaten bir tahsisiniz vardır!). Bu mekanizmalar daha sonra açıklanacak olsa da,

OID ağ acının dallarının çoğ almasıdır (sık sık anlatıldığı gibi), tüm ince parçaları tanımlamanın zor olması!

ASN.1 modüllerini yetkili bir şekilde yazabilmek için biraz OID alanı almanız gerektiğ i, ASN.1'e yönelik bir eleş tiri olmuş tur. Bu aslında doğru değil il - modül tanımlayıcısı gerekli değil il . Ancak, ASN.1 modüllerini üreten çoğ u kişi (baş arılı bir şekilde) OID alanından bir parça almayı dener ve modüllerini OID değil erleriyle tanımlar. Ancak bu size sorun çıkarırsa, bu bir gereklilik değil ildir .

2.10 ObjectDescriptor türü

(Şekil 22'deki "ürün açıklamasına" bakın). ObjectDescriptor türü için tür gösterimi şöyledir:

Nesne Tanımlayıcı

boş luksuz ve karışık büyük ve küçük harf kullanarak!
Bu büyük ölçüde tarihsel bir kazadır.

Bu tür resmi olarak etiketli bir "GraphicString" (dünya dillerinin çoğunu taşıyabilen, ancak bugün eskimiş olarak kabul edilen baş ka bir karakter dizesi türü) olarak tanımlanmıştır. Tanımı bir ASN.1 tip atama deyi̇mle yapıldığından, başlangıçta yalnızca "Yararlı Tip" olarak kabul edildi ve boş luksuz karışık büyük/küçük harf adı verildi. Bugün, ASN.1 spesifikasyonunda "Kullanışlı Tip" terimi kullanılmamaktadır ve bu yerleşik tip için karışık durum kullanımı biraz anakronizmdir.

Nesne Tanımlayıcı

Evet, karışık durum! Bunu bir spesifikasyonda asla görmezsiniz ve muhtemelen onu kullanmak istemezsiniz - bu metni dikkate almayın!

Türün varlığı, OBJECT TANIMLAYICI türünün birimi üzerindeki argümanlardan kaynaklanır.

Hatta kodlandığındaki kısa, sayısal tanımlayıcılar üreten bir tanımlama mekanizmasını (baş arılı bir şekilde) savunanlar vardı. "İnsan dostu" ve çok sayıda metin içeren (örneğin, daha önce tanıştığımız değil) notasyonunun basit bir ASCII kodlaması gibi bir şeyle) ve belki de hiç sayı içermeyen bir tanımlama mekanizması için (baş arızı bir şekilde) tartışan baş kaları da vardı. . Tartışma geliştiğinde, "NESNE TANIMLAYICI" türünün tanıtılmamasını içeren bir tür uzlaşmaya varıldı - kısa, sayısal, dünya çapında net olduğunu garanti edildi, ancak süresiz olarak uzun (ama genellikle yaklaşık 80 karakter) karakter dizisi artı bir nesneyi "tanımlamak" için boş luk. "ObjectDescriptor" değil erinin dünya çapında kesin olduğunu hiçbir şekilde garanti edilmez (dizi, bir nesneyi tanımlamak isteyen her tasarımcı tarafından keyfi olarak seçilir), ancak dizinin uzunluğu nedeniyle, genellikle belirsiz değil ildir.

ASN.1 spesifikasyonunda, bir nesneyi tanımlamak için bir nesne tanımlayıcı değil eri tahsis edildiğiinde, onu tanımlamak için bir nesne tanımlayıcı değil erinin de tahsis edilmesi gerektiğiine dair güçlü bir öneri vardır. Daha sonra uygulama tasarımcılarının protokollerine (bir nesneye atıfta bulunurken) yalnızca bir "NESNE TANIMLAYICI"lığı eski veya hem bir "NESNE TANIMLAYICI" hem de bir "ObjectTANIMLAYICI" eklemesi, belki de ikincisinin dahil edilmesini "İSTEĞE BAĞLI" hale getirir.

Uygulamada (Şekil 22'deki yapay örneğinde) bir uygulama belirtiminde asla bir "ObjectDescriptor" ile karışıklaşmaz! Tasarımcılar onu kullanmamayı seçerler.

Ayrıca, bir nesne için bir nesne tanımlayıcı değil eri tahsis edildiğiinde, atanmış bir nesne tanımlayıcı değil eri de olması gerektiği kuralı sıkılıkla bozulur.

Bir ASN.1 modülünün başlığında nesne tanımlayıcı değil erlerinin en görünür kullanımını ele alın: karışık gelen nesne tanımlayıcı değil eri nedir? Açıktır belirtildi, ancak çoğukışışlığı öyle derdi:

başlıkta nesne tanımlayıcısından hemen önce görünen modül adının ilgili nesne tanımlayıcısını oluşturması. İyi tamam!

Ancak, orijinal olarak ASN.1 spesifikasyonunda atanmış başlık nesne tanımlayıcı değil erleri vardır, örneğin:

{iso standardı 8571}

Bu, numaralandırılmış standardı (aslında çok parçalı bir standarttır) tanımlar ve ayrıca bu standarttan sorumlu olanlara nesne tanımlayıcı ad alanını verir. Ancak buna karşılık gelen bir nesne tanımlayıcı değil eri atanmamıştır!

2.11 İki ASN.1 tarih/saat türü

Evet, gerçekte de § 22'yi doğrudu yorumladınız - UTCTime yalnızca iki basamaklı bir yıl taşıyan bir tarih/saat türüdür!

Ayrıca hem "UTCTime" hem de "GeneralizedTime" in yine karışık büyük/küçük harf olduğunu fark edeceksiniz. Yine bu tarihsel bir tesadüf: ASN.1 tip atama deyişi kullanılarak "VisibleString" etiketli olarak tanımlanılar ve orijinal olarak "Yararlı Tipler" olarak listelendirler.

Neden ikisi de? GeneralizedTime sonradan mı eklendi? Evet ve hayır! 1982'deki ilk taslaklarda, UTCTime mevcut olan tek şeysi ve tarihleri ve saatleri ASN.1 belirtimine "bağlı" olarak temsil etmek için kullanılacak karakter dizisinin belirtimini içeriyordu: yani tam metin bu türün tanımlanması ASN.1 belirtiminde mevcuttu.

UTCTime ve GeneralizedTime

Konsept olarak basit, kullanımı kolay, ancak sorunsuz değil!

GeneralizedTime, 1984'te ilk ASN.1 spesifikasyonu yayınlanmadan önce eklendi, ancak tüm spesifikasyonu içermiyordu - o zamanlar yeni bir ISO Standardı'na (ISO 8601) atıfta bulunuyordu.

Ancak, ASN.1'in ilk kullanıcıları metinlerini UTCTime kullanımına dayalı olarak zaten sonlandırmıştı ve bu, ASN.1 belirtiminde bırakılmıştı. UTCTime'in sadece iki haneli, GeneralizedTime'in ise dört haneli bir yıl kullanması 1982'de tartışma konusu bile değildi! (İki tür arasındaki diğer fark, zamanın kesinliği idi - UTCTime için en iyi ihtimalle bir saniyelik bir kesinlik, daha çok GeneralizedTime için).

1988'e kadar yayınlanmayan, ancak UTCTime'ı da kullanan Dizin çalısması biraz daha az affedilebilirdi! "Kablolu" bir spesifikasyonun cazibesinin - ne elde ettiğinizizi görmek için başlık bir yayın aramanız gereklidir - tasarımcıları UTCTime'ı (GeneralizedTime yerine) kullanmaya teşvik etmede bir etkisi olabilir. 1980'ler.

Ş 22'de bir UTCTime değilinin "kayan pencere" olarak yorumlanmasıyla ilgili açıklama, genellikle iki basamaklı yıl alanları için yapılan üç farklı öneriden biridir:

(geçmiş te VARSAYILAN). 1900 ile 1999 arası bir yıl olarak yorumlayın - varsayılan ayar ve kesinlikle 1982'deki amaç, ancak bugün için küyü bir fikir!

(Şimdilik BASİT öneri). 1950 ile 2049 arası bir yıl olarak yorumlayın - basit ve bize 50 yıl daha kazandırıyor!

(SÜRGÜLÜ PENCERE - sonsuza kadar çalışır!). İçinde bulunan yılın alt iki basamağıyla eşleşen herhangi bir 2 basamaklı yılı, içinde bulunan yıl olarak yorumlayın. Diğer tüm değil erleri, cari yıl eksili yıl ile cari yıl artı 49 yıl arasındaki bir pencere içinde yollar olarak yorumlayın

(veya eks 49'dan art 50'ye - bir seçim meselesi - ancak açıkça tanımlanmalıdır). Bu, her yıl 31 Aralık'ta, elli yıl önceki tarihlerin yorumunun, elli yıl sonraki bir tarih olarak yorumlanması anlamına gelir. Eğ er sisteminizde elli yıl öncesi hiç tarih yoksa (ve kirk dokuz yıldan daha ileri bir tarihe baş vurmanıza gerek yok), bu sistem açıkça çalışır ve iki basamaklı yılların kullanılmasına izin verir. süresiz Düzgün bir çözüm!

"UTC" ne anlamına geliyor? CCIR'den (Uluslararası Radyo Danışma Komitesi) gelir ve "Eş güdümlü Evrensel Zaman" anlamına gelir (baş harflerin tuhaf sırası diğ er dillerdeki addan gelir). Aslında, farklı isme rağmen, "GeneralizedTime" aynı zamanda Eş güdümlü Evrensel Zamanı da kaydeder. Bu zaman standardı nedir? Temelde Greenwich Ortalama Saati'dir, ancak kesin doğruluk için, Greenwich Ortalama Saati yıldızlara dayalıdır ve Paris'te bir atom saatine dayalı ayrı bir zaman standardı vardır. Eş güdümlü Evrensel Zaman, atom saatine dayalı bireysel "tıklere" sahiptir, ancak zaman zaman bir yılın sonuna (veya Haziran sonuna) bir "artık saniye" ekler veya bir saniyeyi kaldırır. , küresel bazda zamanın dünyanın güneş etrafındaki konumuyla aynı hızada kalmasını sağlamak için. Ancak bunun herhangi bir ASN.1 protokolünü etkilemesi olası değil!

UTCTime'in tam değil er kümesi nedir? Türün değil erleri, aşağıdaki biçimdeki karakter dizeleridir:

```
yymmddhhmmZ
yymmddhhmmsZ
yymmddhhmm+hhmm
yymmddhhmm-hhmm
yymmddhhmmss+hhmm
yymmddhhmmss-hhmm
```

"yymmdd" yıl (00 - 99), ay (01 - 12), gün (01 - 31) ve "hhmmss" saat (00 - 23), dakika (00 - 59), saniye (00 - 59) .

"Z", "Greenwich Ortalama Saati"ni (veya UTC saatini) belirtmek için zaman değil erlerinde yaygın olarak kullanılan bir sonektir; diğ erleri bir saat ileri "A", bir saat geri "Y" vb. belirtir, ancak bunlar DEĞERLDİR ASN.1'de kullanılır.

"+hhmm" veya "-hhmm" biçimleri kullanılıyorsa (zaman farkı olarak adlandırılır), değil erin ilk kısmı yerel saat ifade eder; UTC saat, "+hhmm" için "hhmm" çıkarılarak ve eklenerken elde edilir. "-hhmm" için ASN.1 belirtimi aşağıda örneği içерir (1994'te eklenen başka bir örnek, 2001'i temsil eden "01"in "yy"ini gösterir!):

Yerel saat 2 Ocak 1982'de sabah 7 ve eş güdümlü evrensel saat 2 Ocak 1982'de öğlen 12 ise, UTCTime'in değil eri şunlardan biridir:

```
"8201021200Z"
veya
"8201020700-0500".
```

GeneralizedTime aynı genel biçimdir, ancak dört basamaklı bir yıla sahiptir ve "ISO 8601'de belirtilen hassasiyetlerden herhangi birine" izin verir.

Bununla birlikte, GeneralizedTime sorunsuz değil. ISO Standartları zaman zaman revizyonu uğrar ve bunları başka bir spesifikasyonlarından referans almak işin ayaklarınızın altında değil işin sağı lajabilir! 1990'ların ortalarında, pek çok kişi inin GeneralizedTime'i, saniyeler için mevcut maksimum kesinlik in ondalık noktadan (milisaniye) sonraki üç basamak olduğunu varsayıarak uyguladığını anlaştı. ISO 8601'in (mevcut sürüm) daha yakından incelemeden, sınırsız kesinlik e izin verildiği açıkktır - ondalık noktadan sonraki basamak sayısında herhangi bir kısıtlama yoktur. BT

keyfi bir kesinlik e ne kadar süreyle izin verildiği ini belirlemek için yazarın ISO 8601'in önceki sürümlerini (ve özellikle 1982'de geçerli olan sürümünü!) bulmaya çalış ması için tamamlanmamış bir ev ödeviydi. Belki bir gözden geçiren araş tırmayı üstlenir? Aksi takdirde okuyucu iç in baş ka bir küçük alış tırma olarak kalır!

Hem UTCTime hem de GeneralizedTime ile ortaya çıkan baş ka bir sorun, kanonik kodlamalarla ilgilidir: farklı hassasiyetler, sondaki sıfırların bulunduğu ("8202021200Z" v "820202120000Z") aynı soyut değil er (belirli bir zaman) iç in farklı kodlamalar olarak mı yoksa farklı soyut değil erler (çünkü kesinlik iletilen soyut bilginin bir parçasıdır)? Zaman farkı ile benzer bir soru ortaya çıkar. Kanonik kodlama kurallarını kullananlar cevabı bildiği i süreçce, aslında hangi yaklaşımın benimsendiği i çok da önemli değil il. Mevcut metin, kesinlik ve zaman farkının bir zamanı (tek bir soyut değil er) kodlamaların farklı yolları olduğunu ve kanonik kodlama kurallarında zaman farkının olmayacağı (ve "Z" olacağ in) ve orada olduğunu söyleyiyor. hassasiyette sonunda sıfır olmamalıdır, bu nedenle "8202022120000Z" örneğ i kanonik kodlama kurallarında yasal değil ildir. Bu, argümanların bu türdeki kesin soyut değil erler kümesi üzerinde devam edebileceğ i baş ka bir alandır.

3 ek gösterim yapıları

3.1 Seçim türü gösterimi

Şekil 22'de örnek yok! Yalnızca bir uygulama belirtiminde kullanılan "seçim türleri" gördüm.
Yaygın değil iller!

ASN.1 spesifikasiyonu "Seçim türü" hakkında konuş ur, ancak bu yan tümcedeki başlık daha doğrudur - bu, bir tür tanımından çok "İTHALATLAR"a benzeyen bir notasyon parçasıdır: mevcut bir tanıma atıfta bulunur.

SEÇ M TI PI gösterimi - bunu muhtemelen görme ihtimaliniz yok - unut gitsin!

Seçim tipi gösterimi aşağıdaki formu alır:

seçim-alternatif tanımlayıcısı < Tip-notation-for-a-CHOICE

Örnek in, verilen:

```
Örnek seçim ::= SEÇ M {alt1
                      Tip1, alt2
                      Tip2, alt3
                      Tip3}
```

Ardından, "Örnek seçim" in mevcut olduğu kapsam (modül) içinde tip gösteriminin gerekliliği her yerde aşağıda türkçe tip gösterimi kullanılabilir:

```
alt1 < Örnek seçim alt2 < Örnek seçim
veya
alt3 < Örnek seçim
veya
```

Bu notasyon, tanımlanan seçim tipinin adlandırılmış alternatif olarak tanımlanan tipi referans alır ve tip-referans-adının başka bir biçimi olarak görülmelidir. Seçim tipi notasyonu, "Örnek seçim" in orijinal olarak tanımlandığı i modülden farklı bir modüldeyse, herhangi bir etiketleme veya

baş vurulan türe uygulanan geniş letilebilirlik ortamı , seçim türü notasyonunun dē il Örnek seçimimin orijinal tanımını içeren modülündür.

"Bir seçim türü" için dē er göstergemi, yalnızca seçilen tür için dē er göstergemidir.

Baş ka bir deyī le, "alt3 < Örnek-seçim" tip göstergemi için dē er göstergemi, "Tip3" için dē er notasyonudur. ("alt3" tanımlayıcı, "seçim türü" için dē er notasyonunda görünmez ve iki nokta üst üste iş areti de yoktur.)

3.2 Gösterimin Bİ LEŞ ENLERİ

Bu, bir dizinin veya kümenin iç kısmına atıfta bulunan, nadiren kullanılan bir notasyon parçasının baş ka bir örnē idir. Bunu kullanmanın tek nedeni, BER'de fazladan bir TLV sarmalayıcıdan kaçınabilmenizdir! Yine ş ekil 22'de gösterilmemī tir!

Gösterimin Bİ LEŞ ENLERİ - bunu da sık sık görmeyeceksiniz, o yüzden bunu da unutun!

Aşağıda "SEKANS" ile ilgili olarak açıklanmış tir, ancak aynı ş ekilde "AYAR" için de geçerlidir. Bununla birlikte, bir "SEQUENCE" içindeki "Bİ LEŞ ENLERİ" nin ardından bir dizi-tipi için (bunun bir tip-referans-adı olabileceğini ve genellikle olacağını hatırlayın) ve benzer ş ekilde SET için tip notasyonu izlemelidir.

Uygulama spesifikasyonumuzdaki pek çok dizi tipine dahil etmek istediği imiz bir ş eger koleksiyonumuz (tanımlayıcılar ve tip notasyonu) olduğuunu varsayılmı̄. Açı̄k ası̄, tüm bariz sebeplerden dolayı onları birkaç kez yazmak istemiyoruz. Elbette bir tür tanımlayabiliriz:

```
Ortak ş eger ::= Dİ Zİ
{ş e1 Tür1, ş e2 Tür2,
 ...
ş elemen23 Tip23}
```

ve bu türü "gerçek" dizilerimizin her birinin ilk (veya son) ş egī esı̄ olarak dahil edin:

```
İ lk-gerçek-dizi ::= Dİ Zİ {kullanılan-tarafından-tüm Ortak-
ş eger, sonraki-ş e Bazi-özel-tür, sonraki-tekrar Özel2,
```

vb

Son}

Aynı ş eyi, bu ortak ş egerlere ihtiyacımız olan tüm diziler için yapıyoruz. Sorun dē il. (Ve PER ile gerçekten iyi!) Ama BER ile, BER'in çalışma ş eklini hatırlarsanız, "First-gerçek-dizi" için dış düzey bir TLV ve "V" bromiumde her biri için bir TLV elde ederiz. ş egerleri ve özellikle "herkes tarafından kullanılan" ş e için bir TLV. Bunun "V" kısmı içinde, "Ortak unsurlar" ş egerinin TLV'lerini alıyoruz. Ancak - metinsel olarak - "Ortak-ş eger" gödesini "İ lk-gerçek-dizi"ye kopyalasaydık, "Ortak-ş eger" için TLV olmazdı - (BER ile) iki veya üç - belki kurtarırdı döt! - sekizli!

"Bİ LEŞ ENLERİ" kullanırsak ş unu yazabilirmiz:

```
İ lk-gerçek-dizi ::= Dİ Zİ
Ortak elementlerin bileşenleri,
{next-element Bazi-özel-tür, sonraki-tekrar Özel2, vb.
Son}
```

"Bİ LEŞ ENLERİ" göstergemi, metinsel olarak kopyalama olmaksızın bu tür bir kopyalamayı sağlar - baş vurduğ u dizi tipini "açar".

"Öğ enin Bİ LEŞ ENLERİ" nde tanımlayıcı olmadığı ina dikkat edin. Bu isteğ e bağılı değildir - "tanımlayıcı" atlasmalıdır. "Bİ LEŞ ENLERİ" OF gerçekten Dİ Zİ 'nin bir öğesi değildir - öğeleri ayıklayan veya açan bir notasyon parçasıdır. Genellikle "metinsel ikame" olarak adlandırılır, ancak bu tam olarak doğrudur (uyarı okuyucu!) çünkü Ayıklanan öğeler iç etiketleme ve geniş letilebilirlik ortamı, orijinal olarak tanımlandıkları modülün ortamı olarak kalır.

Otomatik etiketleme uygulanırsa ve Bİ LEŞ ENLERİ kullanılsa biraz karmaşıklichkeit vardır. Okuyucunun iki seçenek i vardır: sadece unutun ve hepsinin işe yaradığını not edin (el ile kodlama uygulayıcısı değil iseniz, bu durumda bir sonraki seçenek e bakın!) veya iyi bir alıştırma olarak (hiçbiri bu kitapta resmi olarak ayarlanmamıştır!) ASN.1 spesifikasyonuna gidin ve cevabı bulun!

3.3 SIRALI mı yoksa SET mi?

SEQUENCE, SET, SQUENCE OF ve SET OF için tip notasyonu, "VARSAYILAN" ve "İSTEÇ BAĞLI" kullanımıyla birlikte önceki metin ve örneklerde iyi bir şekilde gösterilmişdir. BER'de (CER/DER/PER değil il) varsayılan değil erin esasen tavsiye niteliğinde olduğunu unutmayın. Bir kodlayıcının, tamamen bir kodlayıcı seçenek i olarak bir varsayılan değil eri açılıcka kodlamasına veya karşılıkla gelen TLV'yi atlamasına izin verilir.

Bir uygulama tasarımcısı genellikle SEQUENCE veya SET kullanmayı az çok keyfi olarak seçebilir. Bu metni okuyun ve ardından her zaman SEQUENCE kullanın!

arasındaki farkları kısaca tartışmıştık.

Dİ Zİ { } ve AYARLA { }

BER'deki kodlama açısından (TLV'ler SEQUENCE için metinsel sıradadır, kodlayıcı tarafından SET için seçilen bir sıradadır) ve ayrıca SET'te "sira semantik olarak anlamlı değil" şeclineki daha teorik bir bakış açısından.

Sorun şunu ki, soyut değil eri sıralanmamış bir bilgi koleksiyonu olarak kabul edersek ve bunu bir kodlamada temsil etmek için tek bir bit modeli istiyorsak, koleksiyonu sıralamak için az çok keyfi kriterler icat etmemiz gereklidir. tek bir bit-desen kodlaması oluşturmak için! Bu, pahalı (CPU'da ve belki de bellek açısından) kodlama kuralları oluşturabilir. SET { } durumunda, kodlayıcı seçeneklerini kaldırırmak istiyorsak, metin sırasını (gerçekten iyi bir fikir değil il) veya etiket sırasını (etiketlerin öğeleri arasında farklı olması gereklidir) kullanmak mümkündür. SET sıralamayı statik bir karar olarak sağlamanın iyi bir fikirdir. Bununla birlikte, "SET OF" söz konusu olduğunda, hiç kimse, her bir öğenin kodlamalarının çalışması zamanı sıralamasını yapmadan tam bir degré kümlesi için tek bir bit modeli sağlamanın bir yolunu bulamadı! Bu pahalı olabilir!

Bölüm III'te kanonik (CER) ve ayırt edici (DER) kodlama kurallarını tartışırken bu noktaya geri döneceğiz, ancak bugünkü tavsiyemiz (ancak şebeke 999'a bakın!): En iyisi "SET {"den uzak durmak ve "den kaçınmak SET OF" veya gibi!

Burada belirtmesi gereken çok küçük bir ayrıntı: "SET {" ve "SET OF" için sağlanan varsayılan etiket aynıdır. "SEQUENCE {" ve "SEQUENCE OF" için sağlanan farklıdır, ancak bunlar da aynıdır. Bu yalnızca, etiketleri asgari düzeyde etiket uygulamasıyla SEÇİMLER ve SET'ler vb. içinde dikkatli bir şekilde uyguluyorsanız önemlidir. Bu durumda, etiketleme hakkında daha sonra okuyacak ve bundan memnun kalacaksınız ve © OS, 31 Mayıs 1999'u belirlemek için ASN.1 spesifikasyonunu dikkatlice kontrol edeceksiniz.

tüm türler için varsayılan etiket! Bununla birlikte, normal bir dümlüseniz, her şeye rutin olarak etiketler uygulayacaksınız (1994 öncesi) veya "OTOMATİK ETİ KETLER" (1994 sonrası) kullanacaksınız ve "SEQUENCE {" için varsayılan etiketin aynı olduğunu gerçeğin "SEQUENCE OF" için olduğunu gibi, her iki durumda da sizin endişenizi elendirmez!

3.4 Dİ Zİ , AYAR ve SEÇİM (vb) değerlerini gösterimi

Bu yapılar için tip gösterimini neredeyse bu kitabın ilk sayfasından itibaren kullandık, ancak şimdiden bunların değerlerini notasyonuna bilmemiz gerekiyor. (Aslında, bununla kurslar dışında veya ASN.1 spesifikasyonunun açıklayıcı bir eki dışında asla karşılaşılmaz, ancak bu, ASN.1 ile tanımlayabileceğiniz herhangi bir tip için tüm değerleri için iyi tanımlanmış bir gösterim olduğunu noktasını güçlendirir. .)

SEQUENCE, SET, CHOICE, vb. değerlerini gösterimi

Onu hiç bir zaman yazmanıza gerek kalmayacak ve yalnızca kurslarda, ASN.1 eğitimlerinde ve bunun gibi saçılıma sapan kitaplarda okuyacaksınız, ama işte burada. Eğitiminizi tamamlamak güzel!

Basitçe söylemek gerekiyor: "SET {" ve "SEQUENCE {" için değerlerini, virgülle ayrılmış bir liste içeren bir çift kaşıtlı ayraçtır. Listedeki her öğe, "SEQUENCE {" (sırayla alınır) veya "SET {" (herhangi bir sırayla) öğesinin tanımlayıcısıdır ve ardından bu öğenin bir değerleri için değerlerini notasyonu gelir. Elbette bu kural, iç içe geçmiş "SEQUENCE {" yapıları varsa yinelemeli olarak uygulanır.

"SET OF" ve "SEQUENCE OF" için yine virgülle ayrılmış bir liste içeren bir çift kaşıtlı ayraç elde ederiz, her öğe "OF"tan sonra gelen tip gösteriminin bir değerleri için değerlerini notasyonudur.

```
bugünün dönüşüm Satış getirisi ::= {versiyon {versiyon2},
raporlanan gün sayısı 8, raporun tarih ve saatbâkemâkî
yıl:"9901022359Z", nedeni -gecikme {açıklama hatası},
```

-- ek bilgiler, satış verileri dahil değildir

```
{--Rapor-item 1: {item item-
description -- Yeni stoklanılmış {wineco-items özel tip (112)},
bir ürün. barkod verileri "Tio Pepe Özel Rezerv Alımı",
'A0B98764934174CDFH, -- stok
tükendi, varsayılan olarak YANLIŞ olarak ayarlanmış tir. minimum stok
seviyesi {mantis 2056, taban 10, üs -2}, maksimum stok seviyesi {mantis 100,
taban 10, üs 0}, ortalama stok seviyesi {mantis 7025, taban 10, üs -2} }
```

```
--Report-item 2: {item {wineco-
items own-dry-sherry (19)}, bar-code-data 'A0B897632910DFE974'H, tüketilmiş -of-stock TRUE, min-stock-
level {mantis 0, taban 10, üs 1}, maksimum stok seviyesi {mantis 5032, taban 1051, üs 0}, ortalama stok
```

-Bu çizimde yalnızca iki rapor öğesi

}

Şekil 23: "Satışların geri dönüşümü" için bir değerler

Son olarak, "SEÇİ M" için beklediğiniz gibi DEĞİL - R - kaşlı ayraç yok! Bunun yerine, alternatiflerden birinin tanımlayıcısını, ardından iki nokta üst üste (:), ardından o alternatifin değil eri için değil er gösterimini alırsınız. Etiketlerin herhangi bir oluşum için veya geniş letilebilirlik işaretçileri veya istisna belirtimleri için değil er gösterimi yoktur. Seçim değil erlerinde iki nokta üst üste 1994 öncesinde mevcut değil idi.

Bu, okuyucunun şekil 22'de verilen "satışların getirisi" türü için "günümüzün getirisi" a (rastgele) bir değil er olarak atılan şekil 23'ü incelemesi için yeterli olmalıdır.

4 X.680/ISO 8824-1'de başka neler var?

Bu bölüm "Temel ASN.1'i - ASN.1 notasyonunu belirten dört belgenin ilkinde bulunan ve günümüzde spesifikasyonlarda ortak kullanımda olan materyali - ele almayı çalıştık. Bununla birlikte, sonraki böümlere ertelenen ASN.1 belgelerinin bu ilkinde bazı ek materyaller bulunmaktadır. Bu bölümün eksiksiz olması için, bundan aşağıda kısaca bahsedilmiş tır.

Ek alanlar şunlardır:

Geniş letilebilirlik ve sürüm parantezleri: Bu büyük bir konudur, kısaca değil inilmiş tır ve ilk olarak 1994'te tanıtılmış tır, sonra tedavi edilir.)

Etiketleme: Zaten kısaca değil inildi. Bu geçmişte önemliydi, ancak 1994'te otomatik etiketlemenin kullanıma sunulmasıyla birlikte şimdiden çok daha az önemli.

Nesne tanımlayıcı türü: Bu, 1998 öncesi X.680/ISO 8824-1'de tamamen ele alınmış tır, ancak malzemenin bazı böümleri artık başka bir Tavsiye/Standart olarak ayrılmış tır. Bu kitabın önceki böümleri pek çok giriş malzemesi üretti, ancak tartışmaya hala eksik!

Delik türleri: Bu terim, daha resmi ASN.1 terimleri EXTERNAL, EMBEDDED PDV, KARAKTER DİZİSİ ve "Açık Türler" (1994 sonrası) için kullanılır. Ve HERHANGİ BİR TANIMLANAN HERHANGİ BİRDEN (1994 öncesi) bahsetmeye cesaret edebilir miyiz? HERHANGİ BİR veya HERHANGİ BİR TANIMLANAN BY'yi hiç duymadıysanız, bu iyi bir şevidir. Ancak daha sonraki metinle lekelenmeniz gerekecek - üzgünüm!

Karakter dizisi türleri: Dünya çapındaki çeşitli itli karakter kümelerinden karakter dizilerini taşımak için yaklaşık iki bir düzine farklı tür vardır. Şimdiden kadar PrintableString, VisibleString, GraphicString ve UTF8String ile tanışın ve bunları kısaca tartışın. Daha söylemeyecek çok şevidir!

Alt tiplendirme veya sınırlandırılmış tipler: Bu, X.680/ISO 8824-1 ve X.682/ISO 8824-3 arasında böünmüş muameyle büyük bir alandır. Şekil 22'de "rapor edilen gün sayısı" üzerindeki "(1..56)" aralık kısıtlamasına sahip bir örneği inceleyelim: Bu form, en sık karşılaşılanınız veya kullanmak isteyeceğiniz formdur., ancak ihtiyacınız varsa başka birçok güçlü notasyon da mevcuttur.

Macros: Bu bölüm bir müstehcenlikle bitirmek zorundayız! Bazı eleşti menler, "Kitabı bu kelimeyle kırletme!" Ancak makrolar, 1980'lerin sonlarına kadar ASN.1'de çok önemliydi (ve değil eriydi) ve bugün de sık sık karşılaşılmaktadır. Ama umarım hiçbiriniz bir tane yazmaya zorlanmazsınız! Bölüm I ve II size makrolar hakkında daha fazla bilgi vermeyecek, ancak

Bölüm IV'teki tarihi materyal, ASN.1'in ömrü boyunca bunların tanıtılmasını ve gelişimi imini tartışır. Bu büyüleyici bir hikaye!

Ek olarak, X.681/ISO 8824-2, X.682/ISO 8824-3 ve X.683/ISO 8824-4'te (1994'te yayınlandı) görünen bir dizi yeni kavram ve notasyon vardır. Bunlar: bilgi nesnesi sınıfları (bilgi nesnesi tanımı ve bilgi nesnesi setleri dahil) ve parametrelendirme.

Yukarıdaki maddeler zaten tanıtılmışsa (bu bölümde veya daha önce), bunların ayrıntılı olarak ele alınması Bölüm II'nin bir bölümüne bırakılmıştır. Henüz tartışılmadıkları yerlerde, aşağıdaki kısa bölümde kısa bir giriş yer almaktadır.

Bölüm 5

Daha karmaşık ik alanlara referans

(Veya: Her zaman ög renecek daha çok ş ey vardır!)

Özet:

Bu bölüm, Bölüm II'de daha kapsamlı olarak ele alınan kavramlara ve notasyona bir giriş sağlar. Bu özelliklerin bazlarından daha önce kısaca bahsedilmiş ti, ancak tam olarak ele alınmamış ti. Bu içerir:

nesne tanımlayıcıları

Karakter dizisi türleri

alt tipleme

etiketleme

Geniş letilebilirlik, istisnalar ve bölüm parantezleri

Burada ilk kez tanıtılan diğer konular şunlardır:

delik türleri

makrolar

Bilgi nesnesi sınıfları ve nesneleri ve nesne kümeleri

Diğer kısıtlama türleri

parametreleş tirme

ASN.1 semantik modeli

Bölüm II'yi göz ardı etmek isteyen okuyucu için burada bir giriş yapılmıştır. 1998'in ortalarında olduğum gibi, ASN.1 notasyonu ile ilgili, en azından bu bölümün sonuna kadar tanıtılmamış hiçbir alan veya kavram yoktur.

Bu bölümdeki metnin amacı şudur:

ele alınan kavramı ve sorunu tanımlamak;

yayınlanmış bir protokolde bu özelliklerin mevcudiyetinin kolayca fark edilebilmesi için notasyonel desteği in gerekli anahtar yönlerinin gösterilmesi; ve

Bölüm II'de bulunan ek metni özetlemek için.

Belirli bir konuda daha fazla ayrıntıya ihtiyaç duyulursa (okuyucunun ilgisini çeken bir şeyle veya varsa), o zaman Bölüm II'deki uygun bölüm başı vurulabilir. Bölüm II bölümü, aksi belirtilmediği sürece, bu bölümde belirtilen tüm kalemlerin "kapanış"ını sağlar.

1 nesne tanımlayıcısı

NESNE TANIMLAYICI türü, bu bölümün geniş amacının ve kullanımının açıklandığı (tip notasyonu ile) Bölüm 4'te (madde 2.9) kısaca tanıtılmıştır.

Değerler gösteriminin örnekleri, bu değerler gösteriminin tüm olası biçimlerini tam olarak göstermemiş olsalar da, metin boyunca ortaya çıkmıştır.

Nesne tanıtıcı ağın (isim-uzayı) biçimine ilişkili daha ayrıntılı bir tartışma, değerler notasyonunun olası biçimlerinin tam olarak ele alınmasıyla birlikte, Bölüm 2 (Daha Fazla Ayrıntı) Bölüm 1'de verilmişdir.

Önceki metin, bu türün normal bir şekilde anlaşılması ve mevcut spesifikasyonların okunabilmesi için yeterince şeyle verdi. Yalnızca bir nesne tanımlayıcı ad alanına ihtiyacınız olduğunun hissediyorsanız ve nasıl alacağınızı bilmiyorsanız, "Daha Fazla Ayrıntı" materyali faydalı olacaktır. Bu materyal ayrıca, tüm adları atlayan ve yalnızca sayıları kullanan (yasal) nesne tanımlayıcı değerler notasyonu ve değerlerin nerede yayınlandıktan sonra bağlı olarak bileşenlerle farklı adların ilişkilendirildiği (tartışma) değerler notasyonu hakkında bazı tartışmalar içerir. veya alt yolların doğası.

NESNE TANIMLAYICILARI, basit bir tip notasyonuna ve daha önce görülen bir değerler notasyonuna sahiptir. "Daha Fazla Ayrıntı" bölüm, size ad alanının biçimini ve bazlarının nasıl alınacağıını anlatır ve değerler notasyonuyla ilgili tartışmalar sağlar.

2 Karakter dizisi türü

Değerler belirli bir karakter repertuarından karakter dizileri olan türlerin adları, önceki metin boyunca yer aldı ve bu bölümün Bölüm 4, Madde 2.8'i, tip notasyonlarını biraz ayrıntılı olarak tartışmıştır:

YazdırılabilirDize
GörünürDize
ISO646Dizesi
UTF8Dizesi

tedavi, bazı okuyuculara yabancı olabilecek "Unicode" gibi terimler getirmesine rağmen.

Şimdiye kadar bu türler için değerler gösterimi çok az ele alındı ve her bir repertuardaki kesin karakter dizisi tam olarak tanımlanmadı.

Şimdiye kadar tanıştığınızdan çok daha fazla karakter dizisi tipi ve özel tipler ve karakter dağıarcıının çalışması zamanına kadar tanımlanmadığı tipler oluştu. turmak için mekanizmalar vardır. Değerler gösterimi, hem basit bir "tırnaklanmış dize" mekanizması hem de "komik" karakterlerle başa çıkmak için daha karmaşık ikinci bir mekanizma sağlar.

Bölüm II (Daha Fazla Ayrıntı) Bölüm 2, deň er notasyonunun tam olarak ele alınmasını sağlar ve tüm karakter dizisi türleri için karakter repertuarlarının kesin tanımlarına referanslar sağlar. Yayınlanan belirtimlerde karşılıklaş acağıınız aşağidakı ek karakter dizisi türlerini açıklar (tüm karakter dizisi türleri, yayınlanan en az bir belirtimde kullanılır):

sayısal dize
 IA5Dizesi
 TeleteksDizesi
 T61Dizesi
 VideotexDizesi
 grafik dizesi
 GenelDize
 evrensel dize
 BMPString
 UTF8Dizesi

Karakter dizisi türleri için en basit deň er gösterimi, basitçe tırnak işaretleri içine alınmış gerçek karakterlerdir (ASCII karakteri QUOTATION MARK, genellikle karakterin üst çeyreğinde iki dikey çizgi olarak temsil edilir). Örneğin:

"Bu örnek bir karakter dizisi deň eridir"

(Uyarı - umarım hala biraz vardır!) Okuyucu dört soru soracaktır:

ASN.1 spesifikasyonunu yayınlamak için kullanılan karakter seti repertuarında olmayan karakter dizisi deň erlerinde görünen karakterleri nasıl ifade edebilirim? (ASN.1 spesifikasyonlarının ASCII metni olarak yayınlanması yaygındır).

ASCII QUOTATION MARK karakterini ("") bir karakter dizisi deň erine nasıl eklerim?

Yayınlanmış bir belirtimde uzun karakter dizisi deň erlerini birkaç satırda bölebilir miyim?

Bir karakter dizisi deň erindeki boş luk karakterlerini ve kontrol karakterlerini tam olarak nasıl tanımlarım?

Bunlar, "Daha Fazla Ayrıntı" bölümünde ele alınan konulardır.

Özetle:

TIRNAK İŞARETİ karakteri, bitişik tırnak işaretlerinin varlığıyla dahil edilir (programlama dillerinde çok yaygın bir teknik).

ASN.1 (karakter seti standartlarına atıfta bulunarak), dünyadaki tüm karakterler için adlar (bu karakterlerin adlarında yalnızca ASCII karakterleri kullanılır) ve bu adların kullanımına izin veren bir deň er gösterimi sağlar.

ASCII'de görünen farklı boş luk biçimlerinin ve kontrol karakterlerinin kesin belirtimini sağlamak için ISO 646 ve ISO 10646 için hücre referansları da mevcuttur.

Daha karmaşık ik bir karakter dizisi deň er gösterimi örneği, "Daha Fazla Ayrıntılar" bölümü:

{ null, {0,0,4,29}, cyrillicCapitalLetterTe, "ABC"}

bunun neyi temsil ettiğ ini $\ddot{\text{a}}$ renmek istiyorsanız "Daha Fazla Detay"a gidin!

Bununla birlikte, yukarıdaki hüküm hikayenin sonu dē ildir. UniversalString veya BMPString veya UTF8String kullanılırsa, ASN.1'de yaklaşık 80 sözde "koleksiyon" karakter için yerleşik adlar bulunur (yine karakter seti standartlarına göre tanımlanır). İşte bu koleksiyonlardan bazılarının isimleri:

TemelLatince
TemelYunanca
Kiril
Katakana
Ipa Uzantıları
MatematikselOperatörler
Kontrol Resimleri
Dingbat'lar

Resmi olarak, bu koleksiyonlar BMPString türünün alt kümeleridir (alt türler - bu bölümün bir sonraki maddesine bakın) ve bu önceden tanımlanmış türlerin kombinasyonlarını kullanarak özel karakter dizisi türleri oluşturmak mümkündür.

Kısim II Kısim 2, bu özelliklerin tam kapsamını sağlar, ancak karakter seti standardizasyonunun biçimini ve tarihsel ilerleyiş i hakkında daha ayrıntılı bir tartışma, Bölüm IV'te (Geçmiş ve Uygulamalar) yer almaktadır. Bu alanı tam olarak anlamak isteyen okuyucular, Bölüm II'yi okumadan önce Bölüm IV'teki ilgili bölümü okumak isteyebilirler.

Son olarak, ASN.1 ayrıca ş u türü de içerir:

KARAKTER Dİ ZESİ

bir karakter dizisini içerecek bir alanı belirtmek için bir SEQUENCE veya SET'e (örneğ in) dahil edilebilir, ancak (bu aşamada) karakter dağarcığı in veya kodlamayı belirlemeden.

Bu, tamamlanmamış bir belirtim veya "deliktir" ve Kısim II Bölüm 7'de ele alınmışdır. Bu karakter dizisi tipi kullanılırsa, hem repertuar hem de kodlama duyuru ile belirlenir (veya OSI yiğini kullanımdaysa, müzakere ile) çalışma zamanında, ancak "kısıtlamalar" (aşağıdaki "Diğer kısıtlama türleri" bölümune bakın), birincil belirtim zamanında veya "profiller" (bir grup tarafından üretilen ve seçenekleri azaltan ek belirtimler) kullanılarak ek belirtimlerle kısıtlanabilir. temel standart).

3 Alt tipleme

Şimdide kadar bu konuda çok az metin var. Bir örnek in görduk:

TAM SAYI (1..56)

tamsayı dē erlerinin yalnızca bir alt kümescini içeren bir tamsayı türü belirtmek için - 1 ila 56 aralığındakiler dahil. Buna "basit alt tipleme" denir ve yaklaşık olarak ASN.1 Spesifikasiyonlarında sağlanmıştır.

Basit alt tiplemeden ilis, kisel kısıtlamalara kadar. ASN.1, bir ASN.1 türünün dē erlerinin bir alt kümescini seçmek için güçlü mekanizmalar ve (PER'de) bu seçilen alt kümeyi çok verimli bir şekilde kodlamak için sağlar.

1986 sonrası.

Basit alt tipleme, oldukça güçlü eşitli mekanizmalar kullanarak yeni bir tür tanımlamak için herhangi bir ASN.1 türünün değer erlerinin bir alt kümesinin seçilmesini sağlar. Bir "Tam Sınıf" protokolü için soyut bir sözdiziminin (iletilebilen soyut değer erler kümesi) normalde tek bir ASN.1 tipinin değer erler kümesi olarak tanımlandığıını unutmayın (bkz. Bölüm 1, madde 2.1, 2.3 ve 3 ve Bölüm 3 madde 4). Bir "Temel Sınıf" protokolüne ihtiyaç duyulursa, bu, uygun şekilde bu değer erlerin bir alt kümesi olarak tanımlanabilir. Bölüm II Bölüm 3'te açıklanan "basit alt tipleme" mekanizmaları, ASN.1 notasyonu kullanılarak resmi olarak sağlanacak böyle bir belirtimi mümkün kıracak kadar yeterli güce sahiptir.

Daha karmaşık ik bir alt tipleme biçiminin bir örneği şöyle olabilir:

```
Temel Sipariş Sınıfı ::= Wineco Protokolü
(Bİ LEŞ ENLER İ LE
    sipariş (Temel-Sipariş ) MEVCUT, satış
    MEVCUT OLMAYAN } )
```

Tüm alt tiplemenin (ve kısıtlamaların uygulanması - aşağın yaya bakın), yuvarlak parantezler içine alınmış ve bazı tip notasyonlarını (sıklıkla bir tip referans adı) takip eden sözdizimi ile yapıldığıını unutmayın.

Bununla birlikte, gösterimi de görüntülemek mümkündür:

TAM SAYI (1..56)

tamsayı alanına bir kısıtlama koymak gibi ve bu, alınan materyalde kısıtlamanın ihlal edilmesi durumunda ne yapılacağına dair düşüncelere yol açar. (Bu, normalde yalnızca gönderen, kısıtlamanın gevşetildiği protokolün daha sonraki bir sürümünü uygulamışsa gerçekleşme melidir. Bu, Bölüm II, Bölüm 5'te ele alınmaktadır (aşağı yaya bakın)).

1994 yılında ASN.1'e bir "boş luğ u" neyin doldurabileceğini kısıtlamak veya o "boş luğ un" içeriğini başka bir alanın değer eriyle ilişkilendirmekle ilgili bir dizi başka kısıtlama biçimini getirilmişdir. Kısıtlamanın bu diğer bir biçimleri Bölüm II Bölüm 9'da ele alınmaktadır.

4 Etiketleme

Daha önceki metin, etiketlemeye girip çıktı, ancak hiçbir zaman tam bir işlem yapmadı. TLV konsepti (etiketlemenin altında yatan) Bölüm 1 Madde 5.2'de tanıtıldı ve ASN.1 etiketleme hakkında daha fazla metin, etiketlemenin tamamen TLV kodlama felsefesiyle ilişkili olarak tanımlandığı ve Bölüm 2 Madde 2.7 ve Bölüm 3 Madde 3.2'de yer aldığı ve "örtük etiketleme" ve "açık etiketleme" kavramları tanıtıldı.

1994'e kadar, etiketlerinizi doğrudan kullanmak, doğrudan bir spesifikasiyon yazmak için temeldi. 1994 sonrası modül başlığındaki OTOMATİK ETİKETLER bunların unutulmasını sağlar. Bu nedenle ayrıntılar Bölüm II'ye düşer. Bir spesifikasiyonu okumak ve anlamak (hatta bir spesifikasiyonu uygulamak) istiyorsanız, etiket kavramı hakkında zaten yeterince bilginiz var demektir, ancak etiketlerinizin kontrolünü ele almak istiyorsanız (1994'ten önce yapmak zorunda olduğunuz gibi), Bölüm II materyali

Aşağıda türler gibi sözdizimsel yapılarla birlikte, etiketin farklı "sınıflarından" da bahsedilmişdir:

```
[3] INTEGER
Yararlı-Türüm ::= [UYGULAMA 4] SIRALI { .... }
[PRIVATE 4] INTEGER
[EVRENSEL 25] GraphicString
```

Bölüm II Bölüm 4:

Farklı etiket sınıflarının eksiksiz bir şekilde ele alınmasını sağlar.

TLV tabanlı olmayan kodlama kuralları kullanılsa bile, açık ve örtük etiketleme kavramlarını anlamlı kılan türlerin ve değerlerin soyut bir modelini sağlar.

Bir belirtimde kullanılan etiket sınıfı seçiminde stil konularını tartışır.

Kümelerin ve dizilerin farklı öğeleri veya seçim alternatifleri üzerindeki etiketlerin ne zaman farklı olması gerekiğine ilişkin ayrıntılı kuralları verir.

5 Geniş letilebilirlik, istisnalar ve sürüm parantezleri

İlk iki terim - geniş letilebilirlik ve istisnalar - zaten birkaç yerde bahsedildi.

Giriş 'in 2. Maddesi, "geniş letilebilirliği i", konuşlandırılmış "sürüm 1" sistemleri ile yıllar sonra tasarlanan ve dağıtılan "sürüm 2" sistemleri arasında karşılıklı çalışmayı sağlama aracı olarak tanımlamıştır.

Geniş letilebilirlik için çok büyük bir hükmü verilirse, her iki taraf da (tüm özellikleri biliyorlarsa) bunların tamamen farkında olsalar bile, bir kodlamadaki hemen hemen her öğe enin bir uzunluk alanı ve bir tanımlama ile "sarılması" gereklidir. sabit değerler Başka bir deyişle, "TLV" (bkz. Bölüm 1 madde 5.2) stilini kodlamaya zorladık. Bununla birlikte, bir sürüm 2 spesifikasyonunun yeni malzeme ekleyebileceğini yerleri kısıtlarsak (ve yalnızca yeni sürüm 2 malzemesini tamamlarsak), çok daha verimli bir kodlama üretelim. Bu, Paketlenmiş Kodlama Kuralları (PER) tarafından sağlanır.

Geniş letilebilirlik hükmünün kullanımını üç nokta (üç nokta), bir ünlem işareti (!) kullanımıyla istisna belirtimi ve eşleşen bir bitiş ikinci çift ile bitiş ikinci bir açık kör eli parantez çifti kullanarak sürüm parantezlerinin kullanımını tanıyalısınız. kör eli parantez kapatma.

Uzatma işareti, Bölüm 3 madde 3.3'te, sürüm 1 sistemlerinin eklenen herhangi bir malzeme ile gerçekleştirmesi gereken eylemleri tanımlayan istisna belirtimi ile birlikte kısaca tanıtıldı.

Bölüm 2 Bölüm 5:

Bölüm 3 metnini genişletir;

uzantı işarelerinin yerleşilebileceğini tüm yerleri açıklar;

istisna belirtimini gösterir; ve

"versiyon parantezleri" kavramını tanıtır ve açıklar (aşağıya bakın).

Geniş letilebilirlik hükmü ASN.1'e ilk kez eklendiğinde, eklenen her dizi veya set $\{\}$ eski "tamamlandı", ancak daha sonra bunun gerekli olmadığı anlaşıldığı - "tamamlanması" gereken tek şeyle, buna eklenen malzemenin tamamışıydı. yeni sürümde yerleşti. Bu nedenle, bu materyali içinde "versiyon parantezleri" ile birlikte parantez içine alma konseptine sahibiz. Bu

```

Dİ Zİ {alan1
    TipA, alan2 TipB, !
    PrintableString : "59.
    ...
    maddeye bakın", -- Aşağı İdakiler, 59. maddede belirtildiği gibi
    eski sistemler tarafından işlenir. [[ v2-field1 Type2A,
        ...
        v2-field2 Type2B ]],
    [[ v3-field1 Type3A,
        v3-field2 Type3B ]],
    ...
    -- Aşağı İdakiler sürüm 1 materyalidir. alan3 TypeC}

```

Şekil 24: Geniş letilebilirlik işaretlerinin ve sürüm braketlerinin gösterimi

Bölüm II Bölüm 5'te tekrarlanan ve daha ayrıntılı olarak açıklanan Şekil 24'te gösterilmişdir.

Sürüm parantezlerini dahil etmenin zorunlu olmadığına dikkat edin. Bunlar yoksa, etki, dizinin her bir $\{\}$ esinin bir dizi versiyonda ayrı ayrı eklenmiş gibi olmalıdır.

Ayrıca, başka sürüm 1 materyali yoksa (Şekil 24'teki "field3 TypeC" mevcut değil), bu durumda son üç noktanın gerekli olmadığıını ve sıkılıkla atlanacağıını unutmayın.

6 delik tipi

Bölüm 2 Madde 2.1, "delikler" kavramını tanıttı: diğeri grupların özellikleri kendi ihtiyaçlarına göre "özelleştirmelerine" izin vermek veya çok çeşidi gibi diğeri malzeme türleri için bir taş ıyci mekanizma sağlamak için tanımsız bırakılan bir spesifikasyonun parçaları.

Birkaç ASN.1 türünden birini kullanarak bir delik bırakabilirsiniz, ancak **Bilgi Nöbetçi Sınıflarını kullanmak daha iyi olabilir!**

Genel olarak belirticiler, protokollerine herhangi bir ASN.1 türü ekleyebilir ve semantik bu türdeki değil erlerle ilişkilendirilecek şekilde tanımsız bırakabilir. Bu bir "delik" oluşturacaktır. Böylece prensip olarak INTEGER veya PrintableString kullanılarak "delikler" sağlanabilir! Ancak genellikle belirticiler bir "delik" bıraktığıında, kabın keyfi bir bit deseni taşıyabilmesini isterler. Bu nedenle, bir "delik" oluşturmak için OCTET STRING veya BIT STRING kullanmak daha yaygın olacaktır. Bir delikin varlığıını açıkça tanımlamak ve bazı durumlarda "delik" içindeki malzemeyi tanımlayacak ilişkili bir tanımlama alanı sağlamak için tanıtılan belirli ASN.1 türleri olduğunu, bu genellikle önerilmez.

ASN.1'in ömrü boyunca "delikler" için sağlanan hükümler aşamalı olarak zenginleşti. Tirilmiş tir ve eski mekanizmaların bazıları artık kullanımdan kaldırılmıştır. Aşağı İdakiler normalde "delik" türleri olarak kabul edilen türlerdir ve Bölüm 7'de tam olarak açıklanmıştır:

HERHANGİ Bİ R (1994'te kaldırıldı)
 HERHANGİ Bİ R TANIMLANAN (1994'te kaldırıldı)
 HARİ Cİ (kullanımdan kaldırıldı)
 Görülü PDV
 KARAKTER Dİ ZESİ

7 Makro

ASN.1 (1984'ten 1994'e kadar) "makro gösterimi" adı verilen çok karmaşık bir sözdizimi parçası içeriyordu. 1994 yılında, "Bilgi Nesnesi Sınıfı" ve ilgili kavramlar (aşağıya bakın) tarafından sağlanan eşdeğeri (ancak çok daha iyileştilmiş) olanaklarla kaldırıldı.

Birçok dilde, grafik paketinde ve kelime işlemcide bir makro özelliği bulunur. "Makro" adı çok yaygın. Ancak, bu terimin ASN.1'deki kullanımı, diğer paketlerdeki kullanımıyla çok az ilişkilidir.

Makroları çevreleyen çok fazla tartışmalar var. İlk on yılında ASN.1'in bir parçasıydılar, ancak birçok sorun ürettiler ve 1994'te yerini Bilgi Nesnesi Sınıfları aldı.

Sık sık bir makro tanımlayan metin göremezsiniz (ve bugün kesinlikle herhangi bir metin yazmamalısınız), ancak daha eski teknik özelliklerde, birimi bir modüle aktarılan bir makro tanımına bağlı olan metinler görmeye devam edebilirsiniz.

```
MY-MAKRO MAKRO ::=  

  BAŞ LAMAK  

    Tİ P NOTASYONU ::= ....  

    ....  

    DEĞER GÖSTERİ Mİ ::= ....  

    ....  

  SON
```

Şekil 25: Bir makro tanımının yapısı

Bölüm IV ("Geçmiş"), makroların ne hakkında olduğunu hakkında biraz daha bilgi verir. Okuduğunuza spesifikasyonlarda bir makro tanımına (makro gösteriminin kullanımı) uymanız olası değil, ancak Şekil 25 genel yapıyı göstermektedir (formu makro notasyon spesifikasyonu tarafından tanımlanan ek metni temsil eden dört nokta). Bu sözdizimi parçası, bir tür referans atamasının meydana gelebileceği bir modülde herhangi bir yerde görünebilir ve makronun adı (geleneksel olarak her zaman büyük harfle yazılır) diğer modüllerde kullanılmak üzere modülden dışa aktarılabilir (ve genellikle verilir).

Makro gösterimi, ASN.1'in bu kitapta tam olarak ele alınmayan tek kısımdır! Bu kitabı okuyucuları ASLA makro yazmamalıdır! Ancak, bir makro adını içe aktaran ve ardından o makronun çağırılması olan sözdizimine sahip modüllerle karşılaştıracaksınız. Yine, bir tür tanımının görünebileceği her yerde bir makro çağrı rısı görünebilir.

Çok sayıda "delik" içeren standartlardan biri "Uzaktan İşlemler Hizmet Öğesi (ROSE)" olarak adlandırılır. ROSE, kullanıcılarının ROSE protokolünü tamamlamak için bilgi kümeleri sağlamasını sağlamak için OPERATION makrosu adı verilen bir makro tanımlar (ve dışa aktarır). OPERATION makrosunu kullanan tipik bir sözdizimi parçası Şekil 26 gibi görünecektir (ancak gerçek örneklerin çoğu çok daha uzundur).

```
arama İ Ş LEMİ  

  ARGUMENT İA5Dizesi  

  SONUÇOKTET Dİ Zİ SI  

  HATALAR (geçersiz sim, isimBulunamadı) ::= 1
```

Bunu tam olarak anlamak için biraz bilgiye veya ROSE'a ihtiyacınız var. ROSE, Kısmen yaygın kullanımından dolayı, ancak esas olarak makro kullanımı, Bilgi Nesni Sınıfı belirtimi ve istisna yönetimine ilişkin iyi örnekler sağladı için Kısım II Bölüm 7'de kısaca açıklanmıştır.

İş LEM makro tanımı, 1994 ROSE belirtiminde bir OPERATÖR Bilgi Nesni Sınıfı belirtimi ile değil iş tırılmıştır ve şekil 26 gibi sözdizimi dahil olmak üzere belirtimler yavaş yavaş değil iş tırılmekte ve bizi OPERATÖR Bilgi Nesni Sınıfı yapmaktadır.

8 Bilgi nesnesi sınıfları ve nesneleri ve nesne kümeleri

Protokol belirticileri, belirtimlerinde "delikler" bırakıklarında, genellikle bu türden birkaç boş luk vardır ve belirtimin kullanıcılarının, bu boş lukları doldurmak için belirli nitelikteki bilgileri sağlamaları gereklidir. Makro gösteriminin kullanımlarının çoğu u, bu kullanıcıların bu ek bilgileri belirtmek için bir gösterime sahip olmalarını sağlamaktır.

Bilgi Nesni Sınıfları (nesneler ve nesne kümeleriyle birlikte), 1994'te ASN.1 notasyonuna yapılan ana eklemeydi ve makroları çok daha gelişmiş bir işlevsellikle değil iş tirdi.
 Bu alanlardaki ayrıntılar Bölüm II'ye bırakılmıştır, ancak artan sayıda eski belirtim bu gösterimi kullanmak için revize edilmektedir ve çoğu u yeni belirtim bunu kullanmaktadır.
Bu alanlar önemli!

Bilgi Nesni Sınıfı konsepti, "delikleri" terk eden belirticilerin bu deliklerin nerede olduğunu açıkça belirlemesi gereklidir, ancak daha özel olarak "deliği i" tamamlamak için gereken bilgileri listeleyebilmesi gereklidir. En basit durumda, ihtiyaç duyulan bilgi, bir tamsayı veya o tip ve onun semantik ilişkili semantik bir nesne tanımlayıcı değil eri ile birlikte boş luğ u doldurabilecek bir dizi ASN.1 tipi (ilişkili semantik ilişkili birlikte) olacaktır. Tanımlayıcı, türün bir değil erinin yanı sıra taşıyıcı protokolünde taşıınacaktır.

ASN.1, toplanacak bilgi biçimini tanımlamak için bir sözdizimi sağlar. Bu, şekil 27'de gösterilmektedir:

```
SINIFIM ::= SINIF
          {&Doldurmak için yazın,
          &tanımlayıcı INTEGER}
```

Şekil 27: Bir Bilgi Nesni Sınıfı tanımlamak için notasyon

"&" karakterinin kullanımına dikkat edin. ASN.1'de "&"nin kullanıldığı tek yer burasıdır ve varlığı, Bilgi Nesni Sınıfları ile ilgili Bölüm II materyalini okumanız gereklidir. inin açık bir göstergesidir!

Belirleyici bir Bilgi Nesnesi Sınıfı tanımladıktan (ve genellikle referans adını dış a aktardıktan) sonra, kullanıcılar o sınıfın nesne kümelerini tanımlayabilir ve bunları temel protokole bağılayabilir. Bu, Bölüm II'de genişletilmiş ve gösterilmişdir.

9 Diğer kısıtlama türleri

Daha önce tartışılan basit alt tiplemeden biraz daha karmaşık ik olan kısıtlama biçimleri vardır. Bunlar "tablo kısıtlamaları", "ilişkisel kısıtlamalar" ve "kullanıcı tanımlı" kısıtlamalar olarak adlandırılır. İlk ikisi, boş lukları tutarlı bir şekilde doldurmak için tanımlanmış bir bilgi nesneleri kümelerinin kullanımıyla yakından ilgilidir. Sonucusu, ASN.1 gösterimi içinde tamamen resmi bir şekilde yapılamayan delik iç eriklerinin belirtilmesi ile ilgilidir. Basit alt tipleme gibi, bu kısıtlamalar her zaman bir tip adının (veya bir delik özelliğinin) ardından yuvarlak parantez içinde görünür. Bölüm II'de gösterilmiş ve açıklanmışdır.

Tablo kısıtlamaları, ilişkisel kısıtlamalar - bir Bilgi Nesnesi Kümesinin tanımlıyla tutarlı bir şekilde delikleri kısıtlamanın yolu. Bölüm II'ye gidin.

Kullanıcı tanımlı kısıtlamalar - ihtiyacınız olan diğer tüm kısıtlamalar için hepsini bir araya toplayın!

10 Parametreleş tirme

Bir ASN.1 spesifikasiyonunu parametreleş tirme yeteneği, çok basit ama son derece güçlü bir mekanizmadır. 1994 yılında tanıtıldı. Bir programlama dilinde işlev veya yöntemlerin sahte parametreleri kavramı, işlev veya yöntem çağırıldığında sağlanan gerçek parametrelerle oldukça yaygındır.

Parametreleş tirme - çok basit ama çok güçlü. Tüm ASN.1 referans adlarının sahte bir parametre listesi olabilir, gerçek parametreler kullanıldıklarında sağlanır.

Benzer bir şekilde, bir ASN.1 tip-referans adına, söz konusu tip kullanıldığıında sağlanan gerçek parametrelerle, sahte parametreler verilebilir.

Örnek in:

```
Benim Tipim {INTEGER:dummy1, Dummy2} ::= SEQUENCE
{birinci alan Dummy2, ikinci alan INTEGER
(1..dummy1)}
```

Burada "Türüm" iki yapay parametreye sahiptir; birincisi "ikinci alan" üzerinde bir sınır sağlamak için kullanılan bir tamsayı ve ikincisi birinci alan için türü sağlayan bir saniye. Tipik olarak, My-Type, toplam spesifikasiyonda birkaç farklı yerde kullanılacaktır ve her birinde farklı gerçek parametreler olacaktır.

Parametreleme, kullanımı bundan daha geniş olmakla birlikte, kullanıcı grupları tarafından tanımlanan Bilgi Nesnesi Kümelerinin orijinal belirtici tarafından bırakılan deliklere bağlanmasını sağlamak için önemli bir araçtır.

12 ASN.1 semantik modeli

ASN.1'de "ile aynı türde olması gereken" ifadesinin olduğu u birçok yer vardır. parametre bir ~~türde olmalıdır, aynı türde olmalıdır~~ parametre "kukla parametre ile aynı türde olmalıdır". DEFAULT'tan sonraki bir deger "DEFAULT" kelimesinden önceki türle aynı türde olmalıdır. Söz konusu tipler aynı tip-referans adı ise, o zaman "aynı tip" oldukları açıktır. Ancak, söz konusu iki türün metinsel olarak farklı ancak aynı metinle belirtildiği ini varsayılmı? Veya metinsel olarak farklı ancak metinde bazı küçük farklılıklar var mı? Hala "aynı tip" mi? Hangi "küçük deger iş ikiliklere" izin verilebilir? 1999'a kadar olan ASN.1 metninin bu soruları açıklığı a kavuş turmak için söyleyecek çok az şeysi vardı!

Soyutlamalar, soyutlamalar, modeller, modeller.
Herkesin kendine ait.

Ancak bazen neyin yasal olup neyin olmadığıını açıkça ifade etmek için açık olmaları gereklidir.

Neyse ki, gerçek spesifikasyonlarda zor durumlar nadiren görülür, ancak ASN.1 araçlarının yazarlarının neyin yasal olup neyin olmadığıını bilmesi gereklidir (veya varsayımlarda bulunulması gereklidir).

1990 yılında bu tür tüm ifadeleri kaldırırmak ve bu alanlarda daha fazla titizlik sağlanmak için bir girişimde bulunuldu, ancak zamanında üzerinde anlaşmaya varılan tatmin edici bir metin elde etmenin imkansızlığı ortaya çıktı ve son dakikada 1994 spesifikasyonu için metin orijinaline geri döndürüldü "olmalı aynı tip".

Ancak bu alandaki çalışmalar devam etti. Sorunu çözmek için, bir parça tarafından temsil edilen temeldeki soyutlamaları tanımlamak için iyi tanımlanmış bir "soyut model" veya "zihinsel model" veya "anlamsal model" (en sonunda ikinci terim seçildi) olması gerektiği kabul edildi. ASN.1 metni, başlangıç noktası, ilk olarak Bölüm 1 Madde 3.1'de açıklandı gibi bir dizi soyut degerin kapsayıcısı olarak bir tür kavramıdır.

Yazma sırasında (1999'un başlarında), çalışmaları tamamlandı ve üzerinde anlaşmaya varıldı ve 1999'un sonrasında yayına bekleniyor.

13 Sonuç

Bölüm I "ASN.1'e Genel Bakış" için ASN.1 notasyonu tartışmasını tamamlar (kalan bölgeler ASN.1 araçlarını ve yönetim ve tasarım konularını tartışır). Bu bölümde tam olarak açıklanmayan konulardan herhangi biri hakkında daha fazla ayrıntıya ihtiyaç duyulursa, Bölüm II'nin uygun bölümüne başvurulmalıdır. Bunlar büyük ölçüde bağımsızdır ve herhangi bir sırayla alınabilir.

Kodlama Kuralları hakkında daha fazla ayrıntı için Bölüm III'e bakın ve ASN.1'in ve bazı uygulamalarının gelişimi geçmiş i için Bölüm IV'e bakın.

Bölüm 6

ASN.1 derleyicisi kullanma

(Veya: Neyle ilgili - hattaki bitleri üretmek!)

Özet:

Bu bölüm:

ASN.1 tanımlı protokollerin uygulanmasına yönelik yaklaşımları açıklar,
ASN.1 derleyicisi yoksa ne yapılması gerekī inı kısaca açıklar,
ASN.1 derleyicisinin konseptini ve işleyiş inı ayrıntılı olarak açıklar,
"OSS ASN.1 Tools" ürünü tarafından üretilen programlama dili yapılarının önekleriyle
uygulama sürecini (bir ASN.1 derleyici kullanırken) gösterir,
bir ASN.1 derleyicisinde "en iyi satın alma" ararken nelere bakılacağı inı tartışır.

Bu bölüm, uygulama mimarileri, strateji vb. hakkında konuşur. Bu nedenle kaçınılmaz olarak eksik
ve kısmıdır. Tartıştı̄ konular standart değil ve farklı uygulayıcılar farklı yaklaşımalar
üretecektir. Ayrıca bir platformda "en iyi" olan başka bir platformda "en iyi" olmayabilir.

Bu bölüm, ASN.1 kullanılarak belirtilen protokollerin uygulanmasına ilişkin bir fikir verir, ancak
ayırtı̄ların çoğu, Bölüm III'te ele alınan C ve Java gibi programlama dilleri ve BER kodlamaları
bilgisine bağlıdır. Bununla birlikte, böyle bir bilgiye sahip olmayanlar, yine de bu bölümde faydalı
bilgiler edinebilirler. Ancak bir programcı değilseniz, bir sonraki maddeyi okuyun ve gerisini
tamamen atlayın!

1 Bir uygulamaya giden yol

Bir ASN.1 derleyici kullanarak uygulama sürecini Bölüm 1 madde 5.6'da ele aldık (ve bunu şekele 12'de gösterdik). **Bir derleyici ile her şey çok basit!**
Bu bölüm okumadan önce, o materyali gözden geçirmek isteyebilirsiniz. ASN.1'inizi seçtīinizde bir programlama dilinde "derlersiniz", derleyici çıktı̄sını
uygulamanın semantīği ile ilgilenen uygulama koduyla birlikte eklersiniz, (gerçekten) derler ve
bağlarsınız. Kendi kodunuz dil veri yapılarını okur/yazar ve gerekīinde ASN.1 derleyici satıcısı
tarafından sağlanan ENCODE/DECODE çalıştırma zamanı rutinlerini çağırırsınız (ve alt katman
API'lerinize bir arabirim sağlarsınız).

Aslında bilmeniz gereken tek şey bu, ancak daha fazla ayrıntı istiyorsanız okumaya devam edin!

2 ASN.1 derleyicisi nedir?

"Derleyici"nin normalde ne anlamına geldiğini hepimiz biliyoruz - üst düzey bir dilde yazılmış bir programın metnini okuyan ve onu bilgisayar belleğine yüklenen ve belirli bir bilgisayar donanımı tarafından uyulabilen talimatlara dönüşen bir program. Genellikle çalışma zamanı kitaplıklarını birleştirmek için başka bir bağlayıcı yükleyici aşaması içerir.

Bir veri yapısı tanımını "derlemek" ne anlamına gelir?

Ancak ASN.1 bir programlama dili değil ildir. Bu, veri yapılarını tanımlamak için bir dildir, öyleyse ASN.1'i nasıl "derleyebilirsiniz"?

Derleyici terimi biraz yanlış bir adlandırmadır, ancak ilk olarak ASN.1 tanımlı protokollerin uygulanmasını destekleyen çok gelişmiş araçları, sözdizimi denetimi ve güzel yazdırma yeteneğinden biraz daha fazlasını sağlayan eski araçlardan ayırmak için kullanılmıştır. Bu bölümün geri kalanında, "derleyici" yerine "ASN.1-derleyici-aracı" terimini kullanacağımız.

ASN.1 kullanılarak tanımlanan bir protokolü uygulamanın birkaç yolu vardır. Üç ana seçenek arasında tartışılmaktadır.

Değerleri geçici bir şekilde kodlamak ve kodunu çözmek için gerekli tüm kodları yazın. Bu, yalnızca en basit ASN.1 spesifikasyonları için uygundur ve kodlama kodunuzdaki hataları ayıklamak ve kod çözmeyle ilgili tüm seçenekleri kullanabilmenizi sağlar. (Aynı ifade, size yardımcı olacak bazı araçların bulunduğu BNF kullanılarak tanımlanan karakter tabanlı protokoller için de geçerlidir, ancak bunlar ASN.1 tabanlı bir ASN.1 derleyici aracı kadar destek sağlar. `§ artname`). Bu seçenek iyi daha fazla tartışmaya izin verir.

gibi çağrıları önceden oluşturmuş ve önceden test edilmiş bir dizi genel amaçlı kitaplık yordamı kullanın. olarak:

```
encode_untagged_int (int_val, output_buffer);
```

Bununla birlikte, yukarıdaki, alacağınız en basit çağrıyı hakkında bir etiket (üç olası sınıftan birinden) sağlar. isteyeceksiniz ve SEQUENCE gibi oluşturmuş türler için bu şekilde destek oldukça karmaşık olabilir.

Bu yaklaşım ayrıca yalnızca kısıtlamaların alakasız olduğu ve etiketlerin ve uzunlukların nispeten katı bir kodlamasının olduğu BER ile gerçekten işe yarar. Bu yaklaşım, ASN.1-derleyici-aracılarının gelişmesinden önceye dayanmaktadır ve biraz sonra tartışılacaktır.

ASN.1 türüne karşılıkla gelen (ve ASN.1-derleyici aracı tarafından ASN.1 türünden otomatik olarak oluşturulan) bir programlama dili veri yapısına değerler koymaya izin veren bir ASN.1-derleyici aracı kullanın ve ardından tüm değerlerin yerinde olduğuunda, o türden değerlerin tam bir kodlamasını üretmek için tek bir "kodlama" çağrıası.

Bu, uygulama kodunun yapısında en az kısıtlamayla en basit uygulamayı sağlar ve bu bölümde en çok tartışılabilir yaklaşımındır. BER için olduğu kadar PER, DER ve CER için de eşit derecede iyi çalışır ve kodlamanın tüm yolları için test edilmiş ve hataları ayıklanmış kodu maksimum düzeyde kullanır.

Ancak, genellikle kodlamanın yanı sıra kodu çözmemiz gerekiyor. Üçüncü seçenek durumunda (bir ASN.1 derleyici aracının kullanımını), kod çözme, kodlamadan daha zor değildir. ASN.1-derleyici-aracı tarafından sağlanan çalışma zamanı yordamları, bir ASN.1 türünün değerlerinin kodlamasını alacak ve o türle karşılıkla gelen programlama dili veri yapısının tüm alanlarını ayarlayacaktır.

Orta seçenekte, kodlama temel olarak uygun kitaplık rutinlerinin bir dizi çağırmasıdır, ancak kod çözme için alınan bit dizisini ilkel değil erlerden oluşan bir ağ aç yapısına ayırtırma ve ardından bulmak için bu ayırtırma ağ acını ağ açta gezdirme sorunu vardır. ilkel değil erler Yine, bu BER ile PER'den daha kolay mümkündür, çünkü BER ile ayırtırma ağ acı, kodu çözülen değil erin türü bilinmeden oluşturulabilir.

Bir kodlama yordamları kitaplığıının ve bir ayırtırma ağ acının kullanımını aşan (kısaca) daha ayrıntılı olarak ele alınmışdır, ancak bu bölüm, ASN'nin uygulanmasına basit bir yaklaşım sağladığını inden, esas olarak bir ASN.1-derleyici aracının kullanımına odaklanmaktadır. %100 garanti (ASN.1-derleyici aracının hatasız olduğu varsayılarak!) 1 tabanlı belirtimler:

Yalnızca doğrudan değil kodlamaları üretilecektir.

Hiçbir doğrudan kodlama, kod çözücü "patlatmaz", değil erler olası tüm doğrudan kodlamalardan doğrudan ekilde çıkarılır.

ASN.1-derleyici-araçlarının ürettiğinin bir örneği olarak, "Rapor-çığı esine" atıfta bulunan "Satış İarın İadesi" için olan wineco belirtimimizin bir bölümünü kullanacağımız. Bunlar ilk olarak bu bölümün 4. Bölümünde Şekil 22'de (kısım 2) gösterildi ve burada yorumlar yapılmadan tekrarlandı.

"OSS ASN.1 Tools" ürünü (ASN.1-compiler-tool ürününe iyi bir örnek) tarafından üretilen C ve Java yapıları ve sınıfları Ek 3 ve 4'te verilmişdir ve C ve Java'ya aşınanlar isteyebilir bu yapıları ve sınıfları şekil 28 ile karşılaştırır.

```

Satış getirisi ::= Dİ Zİ {versiyonu
                           BIT Dİ ZGİ Sİ
                           {sürüm1 (0), sürüm2 (1)} VARSAYILAN {sürüm1},
                           bildirilen gün sayısı INTEGER
                           {hafta(7), ay (28), maksimum (56)} (1..56)
                           VARSAYILAN hafta, rapor
                           saatı ve tarihi SEÇİ M {iki haneli yıl UTCTime, dört haneli yıl
                           GeneralizedTime}, gecikme nedeni SAYILANDIRILMIŞ
                           {bilgisayar arızası, ağ arızası, diğer} İ STEÇE BAĞLI,
                           maksimum stok seviyesi
                           GERÇEK,
                           ortalama stok seviyesi GERÇEK}

Ek Bilgiler
PrintableString Dİ Zİ Sİ İ STEÇE BAĞLI, satış verileri SET OF Rapor
çığı esı, ... ! PrintableString : "Wineco kılavuzunun 15. bölümne bakın"

Report-item ::= SEQUENCE {item item-açıklama
                           bar-kod-veri stok tükendi min-stok-düzeyi NESNE TANIMLAYICI,
                           ObjectDescriptor İ STEÇE BAĞLI,
                           SEKTÖRLÜ Dİ Zİ,
                           BOOLE VARSAYILAN YANLIŞ,
                           GERÇEK,
                           maksimum stok seviyesi
                           GERÇEK,
                           ortalama stok seviyesi GERÇEK}

```

Şekil 28 - Uygulanacak bir örnek

3 Bir ASN.1 derleyici aracının genel özellikleri

Bir ASN.1 derleyici aracı, uygulamanıza dahil edilecek uygulamadan bağımsız programlama dili metni (C için, bu .H ve .C dosyalarıdır) ve son dosyanıza bağılanacak kitaplardan oluşan bir "derleyici"den oluşur. yürütülebilir. Bazı platformlar için derleyici, çalışma zamanında kullanılacak bir DLL oluşturmak için derlenmesi gereken metin de yayinskyabilir.

Bu her şeyi yapar. ASN.1 türünüzü alın. Bir dil veri yapısına "derleyin". Onu dege erlerle doldurun. ENCODE'u arayın. Tamamlandı! Kod çözme de aynı derecede kolaydır.

Genel model, "derleyici" aşamasının ASN.1 modüllerini alması ve iki ana çıktı üretmesidir. Bunlar:

ASN.1 tipine karşı ilişkili gelen veri yapısı tanımları (seçtiğiniz dil için).

Kaynak metin (seçtiğiniz dil için), sağlanan kitaplardaki çalışma zamanı yordamlarının kodlama/kod çözme işlemelerini gerçekleştirmek için kullanabileceğiniz tabloları veya kodu, yalnızca bu bilgilere ve çekirdek içi gösterime yönelik işaretle verildiğiinde üretecek olan kaynak metin (seçtiğiniz dil için) kodlanacak değillerin (ve kodlanacak arabelik için bir tutamaç) listesi. Bu metin, ASN.1 türlerinizdeki etiketlerin tüm ayrıntılarını içerir, böylece uygulama kodunuzdaki etiketler hakkında asla endişelenmenize gerek kalmaz.

Bazı platformlar için durum biraz daha karmaşık olabilir. Derleyici, uygulamanız tarafından kullanılmak üzere bir DLL oluşturmak için derlemeniz gereken metni verebilir.

Bir sonraki bölüm, basit bir kodlama/kod çözme rutinleri kitaplığıının kullanımına bakar ve ardından "OSS ASN.1 Araçları" derleyicisinin "derleyici" kısmından gelen çıktıya ve bu aracın kullanımına bakarız.

4 Basit bir kodlama/kod çözme rutinleri kitaplığıının kullanılması

ASN.1 içindeki uygulamalar (basit sözlü basılı "baskı" programları üretildikten sonra), BER etiketi (tanımlayıcı) alanlarının, BER uzunluk alanlarının ve BER ilkelinin kodlanması yardımcı olan bir rutin kitabıydı. türleri.

Kodlama/kod çözme yordamlarından oluşan bir kitabı (her ASN.1 türü için bir tane) hiç yoktan iyidir. Ancak iş içe Dİ Zİ türlerinin işlenmesinde, özellikle uzunluk alanlarıyla ilgili olarak zorluklar ortaya çıkar.

Bugün bazı uygulamalar hala bu yaklaşımı kullanıyor. Her şeyi sıfırdan yapmaktan daha iyidir!

Yaklaşım, bir BER kodlaması cinsinden açıklanmıştır. Bir PER kodlaması için daha az iyi çalışma eğilimindedir ve burada ASN.1-derleyici-arac yaklaşımı daha uygun olacaktır.

4.1 Kodlama

Etiketlenmemiş ilkel öğelerin kodlanması önelsizdir - ancak etiketleme ekleyin ve baş ka bir Dİ Zİ (vb.) içinde Dİ Zİ içinde Dİ Zİ iç içe geçmiş inş a edilmiş türler ekleyin ve peki, elinizdeki tek ş ey bir dizi ise hayat o kadar basit de ildir sizin için tanımlayıcı ve uzunluk kodlamalarını (ve ilkel de ırlerin kodlamalarını) yapan kitaplık.

Bir rutin kitaplığı i kullanarak kodlama karmaş ık bir hal alabilir, çünkü genellikle bir kodlamayı kodlamadan önce uzunlu ıunu bilmeniz gereklidir!

ASN.1-derleyici-araçlarının ortaya ıkmasından önce, "Rapor-öğ esi" (bkz. § ekil 28) gibi bir diziyi kodlamaya yönelik yaygın bir yaklaş ım, kodun § ekil 29'a benzeyen bir ş eye sahip olması (sözde kod kullanılarak) olurdu.

```

x1 encode-oid (x1 , buffer_x[1]) içine "item"
için de ır er alın
"item-description" de ır erini x2 encode_obje_desc ( x2,
buffer_x[2] ) içine alın
"Barkod verilerini" x3'e alın
encode_octet_str ( x3, tampon_x[3] )
"Stok tükendi" de ır erini x4'e alın
E ČER x4 doğ ruya SONRA
    encode_boolean ( true, tampon_x[4] )
BAŞ KA
    Buffer_x[4] ı̄j esini boş bir dizge olarak ayarlayın
E ČER SON
.....
vb, son ı̄j eyi buffer_x[7] olarak kodlayarak söyleyin.
.....
kodlama dizisi (buffer_x, 1, 7, buffer_y)
-- Bu, 1'den 7'ye kadar buffer_x'in içeriğ ini -- bir "SEQUENCE" sargası ile buffer_y'ye
kodlar.
-- Pratikte SEQUENCE'in etiketlenebileceğ ini unutmayın -- daha karmaş ık bir
çağ rı dizisine yol açar Buffer_y'yi iletim iç in alt katmanlara geçirir.

Buffer_x, buffer_y'yi temizle

```

§ ekil 29 - "Rapor ı̄j esini" kodlamak iç in sözde kod

Burada, satın aldığ ı̄miz bir kütüphanede, herhangi bir ASN.1 ilkel türünün de ırini alacak (o ilkel türü destekleyebilen dilde bazı veri türlerini kullanarak) ve bir arabellekte bir kodlama döndürecek rutinlerimiz olduğunu varsayıyoruz. Son olarak, tüm arabellekleri bir araya getirecek (burada yer alan kopyalamaya dikkat edin) ve bir Dİ Zİ iç in "T" ve "L" üretecek (BER kullandığ ı̄mizi varsayıarak) son kodlamayı döndürecek baş ka bir kitaplık rutini çağ ırıyoruz. buffer_y'de.

Açıktası, ASN.1'imizde daha karmaş ık iç içe yapılarımız varsa, tam özyinelemeye izin veren bir programlama dili kullanmıyorsak, bu oldukça karış ık hale gelebilir. ASN.1 yapısını kodumuzun yapısına etkili bir ş ekilde bağ ıadık, bu da protokolün 2. sürümünde olası de ır iş iklilikleri daha zor hale getirdi.

Kopyalamanın bir kısmını ortadan kaldırırmak iç in yapılabilecek bazı ş eyler var. Sorunun bir kısmı, bir Dİ Zİ Nİ N tüm ı̄j elerini kodlayana ve bu kodlamanın uzunlu ıunu sayana kadar, bir Dİ Zİ Nİ N uzunluk sekizlileri iç in BER sekizlilerini oluş turamamamızdır.

Bir Dİ Zi kodlamak için, kodlamaları bir tampondan diğ erine kopyalamak zorunda kalma sorununu azaltmanın/ortadan kaldırmanın (en azından!) dört yolu vardır. Bunlar:

Dizinin her bir α esinin uzunluğunu belirlemek için yeterli olan bir "deneme kodlaması" yapın (yapımız birçok SEQUENCE veya Sequence of düzeyi içeriyorsa, bunun gerçekten yinelemeli bir α rı olması gereklidir), ardından SEQUENCE başlığıını finalde oluşturun. arabelleğe alın, ardından SEQUENCE α elerinin her birini bu arabelleğe kodlayın.

Bilerrsiz uzunluk formunu kullanın, bu durumda dizi baş I^{G} ini son arabelleğ imize oluş turabilir ve ardından dizinin G^{I} elerinin her birini sonunda bir çift sıfır olacak şekilde bu ara belleğe kodlayabiliriz.

Uzunluğu 2'ye eşit olan uzun biçimli bir uzunluk kodlaması için alan tahsis etme "hilesini" kullanın, ardından uzunluk bilindiğinde daha sonra dolduracağımız iki boş sekizliyi takip edin ve ardından her üçüncü eyi aynı şekilde kodlayın. Son tampon.

Alt katman yazılımının arabiriminde, bitiş ik tek bir bellek parçası yerine bir arabellek zincirini bu yazılıma geçirmenize olanak tanıyan bir "toplama" yeteneği kullanın (mevcut olduğu varsayılarak!).

Bu yaklaşımın BER için iyi çalıştığı gösterilmişdir, ancak CER/DER/PER için mümkün olmamıştır (CER/DER, uzunluk kodlaması için minimum sezikli gereklidir) veya daha zor/karmaşık olabilir.

4.2 Kod Çözme

Kütüphane rutinlerini kullanarak kod çözme o kadar kolay değil. Genel amaçlı bir ayrıştırıcıya ihtiyacınız var - BER için nispeten kolay (PER için daha az kolay), ağ aç yüreme kodu ve ardından ilkel türler için temel kod çözme yordamları.

Bu, karakter tabanlı kodlamalarla yapmanız gerekenlere oldukça paraleldir - ancak karakter tabanlı kodlamalarda, gelen karakter dizisini (BNF'nin girişine dayalı olarak) "yaprak" bileşenlerinden oluşan an bir aç yapısına bölmek için oldukça karmaşık ik bir araca ihtiyacınız vardır. İşlem için, Bir BER ayrıştırma açı üretmek oldukça kolaydır.

Kod çöme için genel amaçlı bir ayrıştırıcıya ihtiyacınız var, ardından ağ ta yürüyorsunuz. TLV yapısı veri tipinden bağımsız olduğundan, BER ile kütüphane yaklaşımı PER ile olduğundan daha kolaydır.

Genel olarak, ASN.1 ile basit bir kodlama-kod çözme rutinleri kitaplığıının kullanımı, orijinal ASN.1 tanımının daha okunabilir olduğunu tartışılabilir olsa da, ne karmaşık ne de BNF kullanılarak tanımlanan karakter tabanlı protokoller için ayrış tırıcılarının kullanımından daha basittir. Karakter tabanlı bir protokolün BNF tanımından daha "meslekten olmayan" birisine,

Gelen bir BER kodlamasını bir ağ açı yapısına (burada her yaprak ilkel bir türdür) ayrış tırmanın, BNF kullanılarak tanımlanan karakter tabanlı bir kodlamadan bir sözdizimi ağ açı üretmekten çok daha kolay olduğunu da bir durumdur.

BER için kod çözme uygulamaları, aşağıdaki "V" bittiğinin yapılmış yapılmadığıını belirlemek için tanımlayıcı oktetlerin 6. bitinin kullanılmasından faydalananabilir ve uygulamadan bağımsız kodun yapraklarında ilkel tiplerle bir ağaç yapısı üretmesini sağlar. Bu ağaç yapısı daha sonra alınan değerlerin belirlemek için uygulanmaya özel kod tarafından "yürüttür".

Bu "yararlı rutinler kitaplığı" yaklaşımı, her şeyi sıfırdan yapmaktan kesinlikle daha iyidir! Ancak aşağında açıklandığı gibi bir ASN.1 derleyici aracıyla işler çok daha basit.

5 Bir ASN.1 derleyici aracı kullanma

5.1 Temel hususlar

Bir ASN.1 derleyici aracı, her şeyi tek adımlı bir iş lemeden çok daha fazlası haline getirir (aracın kullanıcısı için). Nasıl kodlanacağına ilişkin tüm kararlar (arabellekleri kopyalama, deneme kodlamaları yapma, belirsiz uzunluk kullanma, iki uzunlukta uzun biçimli belirli uzunluk kullanma), ASN.1-derleyici-aracının çalışma zamanı desteği içinde şunu ekilde gömülüdür: gelen bir kodlamayı, daha sonra bir programlama dili veri yapısıyla eşleşen bir biçimde belleğe yerlesıtirebilecek bileşenlere ayrılmıştır.

Bir ASN.1 derleyici aracıyla hayat biraz daha kolaylaşıyor! Ne kadar basit yapabilirsin? ASN.1'in resmi gösterimde semantik ele almadığını göz önüne alındığında, daha fazlası... (örneğin, ASN.1'in resmi gösterimde semantik ele almadığını göz önüne alındığında, daha fazlası...)

ASN.1-derleyici-araçları belirli bir platforma özgüdür (donanım, iş letim sistemi, programlama dili ve hatta gelişmiş türme ortamı anlamına gelir) ve kullanmakta olduğunuz platform için uygun olanı bulmanız gerekecektir. Yaygın olarak kullanılan donanım ve iş letim sistemlerinde C, C++ veya Java kullanıyorsanız sorun yaşayamazsınız, ancak oldukça eski bir dile kilitlenmişseniz (kaba konuşuyorsam kusura bakmayın!), hayat daha zor olabilir.

Belirli bir ürün, "derleyici yöneticileri" kullanılarak tek bir yazılım paketinde bu dillerden birkaçını destekleyebilir veya birden çok platform (örneğin C ve Java) için destek istiyorsanız, bir ürünün birkaç sürümü için ödeme yapmanız gerekebilir. Bazı durumlarda "çapraz derleme" (bazı ASN.1-derleyici araçlarının desteklediği) eski platformlarda uygulama desteği sağlayabilir. Temel olarak, mevcut araçları, kullanmak istediğiniz/kullanmanız gereken platformu doğrudan veya çapraz derleme yoluyla destekleyip desteklemediklerine göre "filtrelemeniz", ardından "en iyi"yi seçmeniz gereklidir (bu bölümün sonraki bölümne bakın).

Burada "istek/ihtiyaç" önemlidir. Bazen uygulama platformu sabittir ve gerek tarihsel nedenlerle gereksiz birket politikası nedeniyle de iş tirilmesi neredeyse imkansızdır, ancak daha sıkılıkla, farklı platformların kullanımıyla ilişkili maliyetler vardır ("şirket içi" olmayan donanımın satın alınması, eğitim itim bu platformlar için mevcut araçların "kalitesine" (ve maliyetine) karşı dengelenmesi gereken programcıların maliyetleri vb.).

5.2 Takım tasarımcıları neye karar vermelidir?

İşte bir ASN.1 derleyici aracının tasarımında çok kritik üç karar vardır - ASN.1 veri yapılarının programlama dili veri yapılarıyla nasıl eşleneceğini, genel çalışma sırasında CPU/bellek takaslarının nasıl yapılacağını zaman desteği ve kodlama/kod çözme iş lemleri sırasında bellek ayırma ve arabellek yönetiminin nasıl ele alınacağı. Ancak diğer önemli kararlar, bu alanlarda ne kadar kullanıcı kontrolü, seçenek ve esneklik sağlanacağıdır. Bu faktörlerin tümü, herhangi bir aracın "kalitesine" katkıda bulunur.

Programlama dili, CPU/bellek desteği, tokusları, bellek tahsis ve arabellek yönetimi ve kullanıcı kontrolü ile nasıl eşlenir - bunlar ana konulardır.

ASN.1 derleyici aracının tasarımcıları bazı önemli kararlar almış olacaklar. Daha sonra bu kararların kalitesinin ASN.1 derleyici aracının kalitesini (ve protokol uygulamalarını üretmenize yardımcı olmak için kullanma kolaylığı ve esnekliği in) çok etkilediği ini göreceğiz.

Ele almaları gereken (ve ortaya çıkan ASN.1-derleyici aracının kalitesini etkileyen) en önemli alanlar şunlardır:

ASN.1 programlama dili veri yapılarına nasıl eşlenir?

Çalışma zamanı kodlama/kod çözme hızı ve bellek gereksinimleri arasındaki doğruluğunu nelerdir?

Kodlama ve kod çözme işlemelerini gerçekleştirirken bellek tahsisini nasıl yapılır?

Eşlemeler ve çalışma zamanı işlemi için aracın davranışları üzerinde ne kadar kullanıcı denetimi sağlanmalıdır (ve nasıl - genel yönergeler veya yerel denetim)?

Bu kararların hiçbiri kolay değil, ancak en iyi araçlar, ideal olarak hem genel varsayılan ayarlar hem de belirli yerel geçersiz kılmalar açısından "derleyici yönergelerinin" kullanımı yoluyla tüm bu alanlarda bir dereceye kadar kullanıcı kontrolü sağlayacaktır. (Örneğin, iki sekizli, dört sekizli veya tamamen belirsiz uzunluktaki tamsayılar içindir).

5.3 Bir programlama dili veri yapısına eşleme

ASN.1 derleyici aracının tasarımcıları, rastgele karmaşık ASN.1 türleri kümesinden, seçtiğiniz dildeki ilgili (ve benzer şekilde karmaşık ikil) veri türleri kümesine bir eşleme belirlemiş olacaklardır.

Ve bir ASN.1 modülünün (veya İHRACAT ve İTHALATA bağlı birkaç modülün) metnini alacak ve modül(ler)i işleyecek bir program (bu genellikle "derleyici" olarak adlandırılan bittir) yazmış olacaklardır.) bu modüllerdeki türlerin seçilen hedef dile eşlemesini çıktı olarak oluşturmak.

Bu belki de en önemli tasarım kararıdır. Genellikle "ASN.1 için API'yi tanımlama" olarak adlandırılır ve C++ söz konusu olduğunda bunun için bir X-Open standartı vardır. Bunu yanlış anlarsanız, programlama dili veri yapısının değil erleriyle temsil edilemeyen ASN.1 türünde bazı soyut değerler olacaktır. Veya belki de oluş turulan ~~özelliklerini~~ onu kullanmaya çalıştığınızda sadece programlama dili derleyici hata mesajları üretecektir!

Bu sına nasıl yardımcı olur? Pekala, "Rapor maddesi"ni kodlamak için sözde kodunuz artık daha çok şeyle 30'a benziyor.

```
"item" için deḡ eri Report-item.item'e alın "item-description" için deḡ eri Report-item.item-description'a alın "bar-code-data"yı Report-item.bar-code-data'ya alın
"ran- "out-of-stock" deḡ erini Report-item.ran-out-of-
stock olarak deḡ iş tirin
```

```
...
vb., Rapor ȫj esinin tüm alanlarını ayarlama
...
Çağrı Kodlaması (CompilerInfo, Report-item, Buffer)
İletim için araballegḡ i alt katmanlara geçirin
Tamponu Temizle
```

Şekil 30 - Bir ASN.1 derleyici aracı kullanarak kodlamak için sözde kod

SEQUENCE OF türlerinin veya tekrarlarının iç içe geçmiş bir yapısı ne kadar karmaşık olursa olsun, programlama dili veri yapınızda belirlediğinizde değil erlerden tam mesajınızı kodlamak için sonunda yalnızca bir "Kodla" çağrısı olduğumu unutmayın.

Gelen mesajlar için süreç tersine çevrilir. Kendi kodunuz ayırtırma yapmaz ve açıktır yürümez. Yalnızca, aracın "derleyici" bölümünün sizin için oluşturduğu programlama dili veri yapısının alanlarına erişir.

"Kodlama" çağrılarındaki "CompilerInfo", aracın "derleyici" kısmından çalışma zamanı rutinlerine aktarılan bilgidir. Bu, (diğer erlerinin yanı sıra) BER için uygulanacak etiketlemeyi geçer. Sizin için büyük ölçüde görünmez olsa da (bu bilginin biçimini anlamamanız gerekmek), kodlama/kod çözme işlevlerini sağlamak için çalışma zamanı rutinlerini etkinleştirmek kesinlikle önemlidir.

5.4 Çalışma zamanında bellek ve CPU takasları

Bu "CompilerInfo" parametresi nedir? Bu hayatı bir sihirli bilesen! Bu, derleyici tarafından üretilir ve "Satış İadesi" ile işaret edilen belleğin içeriğinin (örneğin) alınmak, bu bellekten ASN.1 türü için gerçek ek değil erleri bulmak ve kodlamak için "tarifi" içeriir. Bu değil erler doğrudan etiketler, DEFAULT'un doğrudan kullanımı vb. bilgiler ASN.1 tip tanımında mevcuttur.

Tabloların yorumlanması, bir görevi gerçekleştirmenin oldukça kompakt bir yoludur, ancak açık kod daha hızlidır. Seçtiğiniz en iyi araçlarla.

Esasen tamamını içeriir

Bu "CompilerInfo" nun alabileceğiniz (en azından!) iki biçim vardır:

(Örneğin) değil erini içeren belleğin içeriğinin nasıl kodlanacağıını belirlemek için "Kodlama" tarafından yorumlayıcı bir şekilde kullanılan çok kompakt bir tablo kümesi olabilir. "Satış İadesi" (ve benzeri şekilde "kod çözme" için).

O alanın kodlamasını yapmak için sırayla her alanın değil erini almak (ve parçaları daha büyük SEQUENCE, SEQUENCE OF, vb. yapılarında birleştirmek) (daha ayrıntılı, ancak daha hızlı) gerçek kod olabilir. Genellikle, açık kod PER için BER'den daha uygundur, çünkü PER'de etiketler ve uzunluklar genellikle ihmal edilir, oysa kodlanacak etiketleri tanımlayan ve yorumlayıcının uzunlukları oluşturmasına izin veren tablo odaklı bir yaklaşım daha uygun olabilir. BER için. Kurslar için atlar!

Elle kodlama için birçok farklı uygulama mimarisi olduğunu gibi, araçların tasarımları için de birçok farklı olası mimari vardır. Uygulama mimarilerinde önemli olan tek şey hat üzerindeki bitlerin doğru olmasıdır. ASN.1 derleyici aracında da benzer şekilde, gerçekten önemli olan tek şey, ASN.1 türünün tüm soyut değerlerini temsil edebilen bir programlama dili veri yapısı üretmesi ve yerleşik türlerin değerlerin için verimli bir şekilde doğru kodlamalar üretmesidir. Bu veri yapısında. (Kod çözme ile ilgili benzer açıklamalarla birlikte.) "OSS ASN.1 Tools" ürününün bir kodlama (veya kod çözme) üretme konusunda tam olarak nasıl çalıştığını bilmiyorum, ancak doğru sonuçları veriyor!

5.5 Bir aletin kontrolü

Bir ASN.1 derleyici aracına (ve/veya onu destekleyen çalışma zamanı kitaplıklarına) dahil edilebilecek birçok seçenek vardır. Örneğin:

Kaçınılmaz olarak kullanıcıya bırakmak istediğiniz seçenekler vardır. Bunu en iyi nasıl yapabilirim?

"Derlenecek" dil veya platform.

ASN.1 INTEGER türlerinin programlama dili veri yapılarında nasıl temsil edileceği.

ASN.1'den programlama dilinize eşlemede dizilerin mi yoksa bağlı listeli yapılarının mı kullanılacağı (örneğin, "SEQUENCE OF" içinden).

Kodlama için hangi kodlama kurallarının kullanılacağı (ve kod çözme için varsayılmak).

(Biraz daha incelikli) Çalışma zamanında hangi kodlama kuralları seçilebilir - tamamı mı yoksa yalnızca bir alt kümeli mi? (Bu, dahil edilen kitaplık yordamlarını ve dolayısıyla yürütülebilir dosyanın boyutunu etkiler.)

Kanonik olmayan kodlama kurallarında hangi kodlamaların kullanılacağı.

Kullanıcının mümkün olan en hızlı kodlamayı/kod çözmemeyi mi yoksa en küçük yürütülebilir dosyayı mı tercih ettiği.

(Oldukça öbensiz) Hem derleme zamanında hem de çalışma zamanında kullanılacak dizinlerin ve dosyaların adları.

Ve birçok diğerleri.

Kullanıcının kontrolü, genel bir yapılandırma dosyası, komut satırı yönergeleri, Windows tabanlı bir ürünündeki bir "seçenekler" düğmesi, ASN.1 kaynağına girmeli "derleyici yönergeleri" veya çalışma tırma tarafından ifade edilebilir. zaman çağrısı parametreleri veya bunlardan birkaçı tarafından, biri genel bir varsayılan sağlar ve diğer biri bu varsayılanı yerel olarak geçersiz kılar. "OSS ASN.1 Tools" ürünüyle, derleyici yönergeleri, özel bir yorum biçimi olarak bir tür tanımından sonra (bir alt tür belirtiminin gidebileceği yer) eklenir. Örneğin:

SET --<BAĞLANTILI>-- TAM SAYI

6 "OSS ASN.1 Tools" ürününün kullanımı

Burada, belirli bir araçla dē erlerin nasıl kodlanacā inı açı̄lıyoruz. Dīg er ASN.1- derleyici araçları ile süreç benzerdir.

Dē erlerinizi dil veri yapısına koyun ve ENCODE'u çagırın. Hepsi bu kadar! Az çok!

C programlama dili kullanılarak yazılmış bir uygulamayı desteklemek için "OSS ASN.1 Araçları" ürününü kullandığınızda, bir ASN.1 belirtimi girersiniz (ve soyut sözdizimini veya PDU'yu olū turan üst düzey türü derleyiciye tanımlarsınız) bir derleyici yönergesi aracılığyla. Bu, tek bir modül veya birkaç modül kullanılarak tanımlanabilir. Dört çıkış vardır (ancak doğru ASN.1 giriş īçin yalnızca son ikisi önemlidir):

"Güzel basılmış" bir liste (gerçekten çok önemli dē il).

ASN.1'iniz biraz "komik" ise hata ve uyarı mesajları.

ASN.1 türlerinizin C dili verilerine eşlenmesini içeren bir ".h" başlık dosyası yapılırlar.

Derleyiciden, kodlamak ve kodunu çözmek īçin çağıracağınız çalışma zamanı yordamlarına bilgi aktaran bir ".c" kontrol dosyası.

İkincisi oldukça anlaşılır (ancak hayatı derecede önemli) ve onu C derleyicinizle derlemek ve uygulamanızın bir parçası olarak ortaya çıkan nesne dosyasına bağlamak dışında onu görmezden geliyorsunuz.

".h" dosyası kendi kodunuzla birlikte gelir ve uygulamanızın "kodlama" ve "kod çözme" çağrılarını içeren ana bölümünü oluşturmak üzere derlenir. Ayrıca bir çalışma zamanı kitaplığında da bağlantı kurarsınız. Bu aşamada, hacimleri nedeniyle bu bölümde yer almayan Ek 3 ve 4'e bakmak isteyebilirsiniz.

Ek 3, "Satışların iadesi" īçin ".h" dosyasının çoğuunu ve C dili uygulaması īçin "Rapor-öğesini" vermektedir (ve ilgili "include" dosyalarının bazı kısımlarını). Ek 4, bir Java uygulaması īçin eşdeğerini verir.

Bu ekler hakkında hiç bir açıklama veya tartışma sunmuyorum - eğer bir C veya Java programcısısanız, metin (ve ASN.1 tanımlarıyla ilişkili) oldukça anlaşılır olacaktır. Eğer dēilseniz, onları görmezden gelin!

İşte buyur! Elbette, orijinal uygulama standarı ASN.1 kullanmak yerine "pseudo-C" veya Java'da yayınlanabilirdi, ancak bu gerçekten iyi bir fikir olur muydu? Bir kez olsun bir görüş belirteceğim - HAYIR. Aynı soruyu 1982/4'te sorsaydınız, COBOL veya Pascal (veya belki de Modula) hakkında konuşuyor olurduk. Ve yapılarınızı "pseudo-C" de tanımlasınız bile, yine de bu yapıların kodlanması hakkında açıklamalar yapmanız gereklidir, en önemlisi satır boyunca iletildiğiinde bir tamsayıdaki baytların sırası, düzleştirmeye hakkındadır. Tamsayıların ve işareti ̄lerin boyutu hakkında oluşturunuz herhangi bir ağacı yapısı vb. ASN.1 ile gerçekten çok daha basit - ASN.1-derleyici aracının yükü almasına izin verin!

Ekler elbette tüm derleyici çıktıları dēildir. Çalışma zamanı yordamları tarafından kodlama/kod çözme işlemi gerçekleştirmek için kullanılan kontrol bilgileri dē vardır, ancak uygulayıcının buna asla bakması gerekmek ve burada gösterilmez.

7 Bir ASN.1 derleyici aracını diğ erinden daha iyi yapan nedir?

Bir aletin kalitesinin değ erlendirilebileceği bir çok boyut vardır. İ ncelenecek baş lica alanlar ş unlardır:

Tam ASN.1 gösterimi için destek in kapsamı.

TAMAM. Demek bir ASN.1- derleyici aracı satın almak istiyorsunuz? Best-buy'da nelere dikkat edilmeli? Çamaşır makinesi almak kadar kolay değ il! İ ş te aramak veya dikkat etmek isteyebileceğiniz bazı ş eyler.

Programlama dili veri yapılarına eş lemeler.

Çalışma zamanı belleğ i/CPU değ iş tokuşları.

Bellek ayırma mekanizmaları.

Seçenekler üzerindeki kullanıcı kontrolünün derecesi.

Bir araç satıcısının alması gereken karar türlerini tartıştığımızda, bu alanların çoğu u hakkında zaten bazı tartışmalar yapmıştık. Burada birkaç ayrıntı noktasını vurguluyoruz. Bununla birlikte, en iyi araçlarla aşağın da listelenen sorunlardan kesinlikle hiç birinin ortaya çıkılmayacağıını kabul etmek önemlidir. Gerçekten de sorunların çoğu u, tam olarak geliş tirilmeden önce yalnızca ilk araçlarda meydana geliyordu.

Bazı eski araçlar ASN.1 değ er gösterimi için destek sağ ılamadı, bu nedenle modülünüzden tüm değ er atamalarını kaldırmanız ve uygulama kodunuzdaki varsayılan değ eri işleyerek "VAR SAYILAN"ı "İSTEĞE BAĞLI" ile değ iş tırmeniz gerekiyordu.

Diğ er eski araçlar yalnızca tek bir modülü işleyebiliyordu (İ THALAT ve İ HRACAT için destek yok), bu nedenle tek bir modül oluş turmak için metni fiziksel olarak kopyalamanız gerekiyordu. Bugünün daha iyi araçları birden fazla modülü idare edecek ve (onlara üst düzey mesajınızı belirledikten sonra) bu modüllerden tam olarak ve yalnızca üst düzey mesajınızı desteklemek için gereken türleri çıkaracaktır.

Diğ er bir konu da ASN.1 tanımını yayınlandı iş ekliyle kullanıp kullanamayacağınız veya modülünüzdeki atama ifadelerinin her birinin sonuna noktalı virgül ekleyerek araç taki ayrıştırıcıya yardım etmeniz gerekip gerekmediğidir.

Yalnızca belirli bir protokolü desteklemek için tasarlanmış ve yalnızca o protokolde görünen türleri tanıyan başka araçlar da vardır. Bu protokol, sürüm 2'de daha fazla tür kullanmak üzere genişletilirse, sürüm 2'yi uygulamadan önce aracınızın yükseltilmesini beklemeniz gerekebilir!

Bölüm II'de açıklanan ASN.1 - Bilgi Nesnesi Sınıfları vb. için 1994 uzantıları sorunu da vardır. Bu, muhtemelen bazı araçlarda hala destek eksikliği bulacağınız alandır.

Programlama dili veri yapısına eş leme çok kritik bir alandır. Bu yanlış sa, ayarlayabilmeniz gereken tüm değ erleri ayarlayamayabilirsiniz!

Ayrıca, ASN.1'in tanımlayıcılar için (tüm karakterler anlamlı olacak şekilde) keyfi uzunluk adlarına izin verdiğiini ve büyük/küçük harfe duyarlı olduğunu unutmayın. Bazı programlama dillerinde, (örneğin) 31'inci karakterden sonraki karakterler basitçe atılır. Araç, (ASN.1'de oldukça yaygın olan) uzun adların sizin anlayabileceğiniz ergonomik bir şekilde farklı programlama dili adlarına eşlenmesini sağlıyor mu?

INTEGER türleri ne olacak? İ yi bir araç, INTEGER türlerinin örneğ in kısa, normal, uzun veya çok büyük (ş u ş ekilde temsil edilir) olarak eş lenmesi üzerinde size kontrol sağ lar (genellikle belirli bir türde karşı ASN.1 metnine göndü ünuz genel yönergeler veya yönergeler aracılığ yla) bir dizi) tamsayı.

Eş lemelerde verimlilik hususları da vardır. Bazı platformlarda "yerel" tamsayı türleri kavramı vardır. Doğrudan bunlara eş leme yapmak, daha genel (platformdan bağımsız) bir ş ekilde ilerlemekten çok daha verimli olabilir.

ASN.1'den bir programlama diline (genellikle "ASN.1 Uygulama Programı Arayüzü (API)" olarak adlandırılır) eş lemelerin genel olarak standartlaşdırılmadığı in, bu nedenle her araç satıcısının kendi işini yaptığı in hatırlamak önemlidir. ASN.1/C++ API olarak adlandırılan C++ ile eş lemenin standartizasyonu üzerine X-Open içinde yapıldı, ancak belgenin nihayet onaylanıp onaylanmadığı inanınca emin değilim. Uygulama diliniz olarak C++ kullanmak istiyorsanız, sormak isteyebilirsiniz. Bu eş lemeyi kullanıp kullanmadıkları konusunda araç satıcınız.)

Çalışma zamanı kodlama ve kod çözme için oluşturulan koda dayalı (daha fazla bellek alan ancak daha hızlı) bir yaklaşımın ina karşı büyük ölçüde yorumlayıcı tablo tabanlı bir yaklaşım (az bellek kullanarak) seçeneğ ini daha önce tartışmıştık. Bu, muhtemelen her uygulama veya platform için hangi yaklaşımın benimsenmesini istediği inizi seçmenizi sağlayacak aracın kullanımında seçenekler arayacağınız bir alandır.

Ve son olarak, daha önce araç seçenekleri üzerinde kullanıcı kontrolü sağlama araçlarını ve kontrol edilebilecek bu tür seçeneklerin kapsamını tartışmıştık.

Tüm bu faktörler bir aletin "kalitesine" katkıda bulunur, ancak kesinlikle maliyetine de bakmak isteyeceksiniz! Çoklu araç satıcısı, size ASN.1-derleyici aracının yalnızca bir kopyasını, ancak çalışma zamanı desteği inin sınırsız kopyalarını (sonuçta ortaya çıkan uygulamanızı dağıtmak istiyorsanız kesinlikle ihtiyacınız olan!) sağlayan bir lisans ücreti alır.

8 Sonuç

Bu bölümde, ASN.1 kullanılarak tanımlanan bir protokol için gerçek bir uygulamanın nasıl oluşturulacağı ele alınmışdır. Bunu, bu kitabın I. Bölümünü tamamlamak için yöneticiler, belirleyiciler ve uygulayıcılar tarafından değerlendirilmek üzere yönetim ve tasarım konularının bazı tartışmaları takip eder.

Bölüm 7

ASN.1 belirtimi ve uygulaması için yönetim ve tasarım sorunları

(Veya: Düşünmeniz gereken şeyler!)

Özet:

Bu bölüm:

metnin başka yerlerinde bahsedilen pek çok konuyu ve "üslup" kararlarını bir araya toplar;
yönetim kararları için bazı küresel sorunları tanımlar;
belirticilerin göz önünde bulundurması gereken hususları tanımlar;
uygulayıcıların dikkate alması gereken hususları tanımlar.

Yönetim kararları ile ilgili bölüm, Bölüm I'ı okuyan herkes tarafından anlaşılabilir olmalıdır. Geri kalan böümler, Bölüm II'de kapsanan materyal bilgisini gerektirecek, ASN.1 hakkında oldukça detaylı bilgi sahibi olunacağıını varsayıp ve oldukça anlaşılması güç bazı alanları kapsayacaktır.

Bir uyarı: Yönetim gurularına ve ayrıntılı "metodolojilere" inanan biri değilim. Aşağıdaki başlıkların çoğu "sorunlar" kelimesi bulunur. Aşağıdaki metin, okuyucuya seçenekler ve göz önünde bulundurmaları gereken şeyler hakkında bir fikir vermek için tasarlanmıştır. Günün sonunda kararları sen verirsin, ben değilim! Size ne yapmanız gerektiğiini düşündüğümü söylemek yerine, mümkün olduğunca üzerinde düşünmeniz gereken alanları öphemeye çalışıyorum. Ara sıra ikincisine yöneliksem, özür dilerim ve lütfen tavsiyemi görmezden gelmekten çekinmeyin!

Bu bölümde söylenenlerin çoğu fikirdir (yine de ekil 999!), gerçek değil il ve burada öne sürülen bazı önerilere karşı farklı ve belki de zıt görüşlere sahip başkaları da olabilir.

1 Yönetim kararları için küresel sorunlar

1.1 § arname

1.1.1 ASN.1'i kullanmak ya da kullanmamak!

Bu, protokollerini tanımlamaya yönelik çeşitli tekniklerin açıklandığı 1.Bölüm 1'de iyi tartışılmıştır. Bu elbette 1 numaralı karardır, ancak protokol spesifikasyonunun yapıldığı kültür veya diğer ilgili protokoller için kullanılmış olan spesifikasyon göstergesini ile daha fazla şartlandırılmış olabilir.

Buraya kadar okuduysanız ve bir protokol için kullanılan belirtim dilini etkileyebiliyorsanız, eminim ki ASN.1'i ciddi şekilde dikkate alındığından emin olacaksınız. Bir sonraki maddeye geçin!

Şimdiye kadar, ASN.1 kullanarak bir belirtim üretmenin ve bir ASN.1 aracının mevcut olması koşuluyla böyle bir protokolü uygulamanın kolaylığıını net bir şekilde görmüş olmalısınız.

Karşılık argüman, kullanım kolaylığı nedeniyle, ASN.1'in sizin belirtiminizi basit tutmaya zorlamamasıdır (ama elbette bunu yapmanızı engellemez!) ve protokol ne kadar karmaşık olسا o kadar karmaşık olacak hale gelir. uygulayıcılarınızın araç desteği ine ihtiyacı olacak ve araçlar maliyetlidir!

Bununla birlikte, protokolünüzün ticari firmalar tarafından uygulanmasını bekliyorsanız, uygulamaya harcanan belki on ila yirmi yıllık bir çabayla, bir araç satın alma maliyeti tamamen öneşiz hale gelir. Profesyonel olarak geliş tirilmiş, desteklenen ve sağlam bir araç için para ödemek, genellikle uzun vadede bir "bedava" kullanmaktan daha etkilidir. (Bunun ana karşılık argümanı muhtemelen Apache Web sunucusudur - muhtemelen bugün kullanılan en popüler Web sunucusudur ve ücretsizdir! Ancak 1. ngilizcede "istisna kuralı kanıtları" diye bir söz vardır.)

1.1.2 Kopyalamak mı yoksa kopyalamamak mı?

Başka bir standartta tanımlanan (ve dışa aktarılan) bir ASN.1 türüne ihtiyacınız varsa, bu türü kendi modül(ler)inize almak için açık bir argüman vardır. Bu genellikle ROSE veri türleri ve nesne sınıfları ile X.500 Dizin Adları ve X.509 sertifikaları için yapılır.

Bu durumda, elbette, içe aktardığınız kaynakta açık bir referans da eklersiniz.

Kopyalamak yanlış, değil mi? İzin almanın mümkün olabilir ve bu daha iyi bir çözüm olabilir. Aşağıdaki konulara bakın.

Bununla birlikte, bazı belirticiler tarafından alınan başka bir seçenek daha vardır ve bu, bir tür tanımını basitçe kendi belirtiminize kopyalamaktır (elbette alanlarla ilgili semantik de verir). Spesifikasyonuz, kopyaladığınız spesifikasyonla aynı standartlar kuruluşu tarafından yayınlanacaksa, ancak bu durum bir dizi spesifikasyonda ortaya çıkmışsa bile, bu muhtemelen telif hakkı yasalarını veya en azından fikri mülkiyet haklarını ihlal etmektedir. Yukarıdaki uyarı geçerli değildir!

İçé aktarma ve referans verme yerine kopyalamanın (yerleştiğinde) üç ana nedeni vardır:

Malzeme üzerinde kontrol sahibi olmanızı sağlar, başı vurulan malzeme daha sonraki bir sürümde kendi spesifikasyonlarınızla uyumlu olmayan bir şekilde de factis sorunları ve karışıklıkları önlüyor.

Bu, uygulayıcılarınızın yalnızca belgelerinizi alması gereki̇g i anlamına gelir - belirtiminiz eksiksiz ve bȧg ımsızdır.

Kopyalanan materyalin yalnızca basitleş tirilmiş bir versiyonunu istiyorsunuz (DOĞRUDAN İ THALAT kullanımından ziyade di̇g er spesifikasyonlarda ROSE materyalinin kopyalarını bulmanızın nedeni genellikle budur).

Bu konudaki kararlar kolay dėg ildir ve uygun tartış malardan sonra bilinçli olarak alınmalıdır.

Spesifikasyonla ilgili baş ka gerçek yönetim sorunu yoktur (ancak belirticiler i̇çin daha fazla ayrıntı aş ağı ıda tartış ılmaktadır), bu nedenle ş imdi uygulama ile ilgili konulara dönuyoruz.

1.2 Uygulama - bütçeyi belirleme

Herhangi bir ticari projenin ayrıntılı maliyetlere ihtiyaçı vardır, ancak ASN.1 tabanlı bir spesifikasyonun uygulanmasını üstlenirken bazı gizli maliyetleri (veya akıllıca para harcama fırsatlarını!) gözden kaçırma kolay olabilir. Bunlardan bazıları aş ağı ıda belirtilmiş tir.

Maliyetlendirmelerinizi yaparken unutmamanız gereken birkaç ş ey...

1.2.1 Teknik özellikleri alma

Gereksinim duyduğ unuz iki özellik seti vardır - uyguladı̇g iniz protokol i̇çin olanlar ve ASN.1'in kendisi i̇çin olanlar.

Elbette protokolünüz i̇çin spesifikasyona ihtiyacınız var. Ama aynı zamanda ASN.1 i̇çin ve muhtemelen bu referanslardan herhangi biri i̇çin.

Çȯg u durumda, hem protokol belirtiminin hem de ASN.1 belirtiminin en son sürümlerini kullanmak isteyeceksiniz, ancak bazen eski sürümlerin kullanımına iliş kin bazı endüstri veya topluluk ç kar anlaşı maları olabilir. (ASN.1 1990 sorunu, Bdüm IV, Bdüm 1'de tartış ılmaktadır). Ş artnamelerdeki düzeltme ve eklerde dikkat etmek i̇çin de dikkatli olun. Spesifikasyonlarınızı alı̇g iniz yer sizi bu konuda uyarabilmelidir. Bazı durumlarda dolaş imda taslak düzeltmeler veya zeyilnameler olabilir. Bu ikinci durumda, daha fazla araştırmalar yapmanız ve belki de bu belgelerin istikrarını keş fetmek i̇çin standartların baş kanı veya raportörü veya editörü ile iletişim imle geçmeyi denemeniz gerekebilir. Taslak düzeltmeler ve ek taslaklar her zaman onaylanmış düzeltme veya ek haline gelmez (en azından bazen önemli dėg iş iklikler olmaksızın).

ITU-T'nin artık (bir hesap olu̇ turmanız koş uluyla) tüm ITU-T spesifikasyonlarını satın alabileceğ iniz ve Web üzerinden kopyalarını indirebileceğ iniz bir Web sitesine sahip olduğ unu unutmayın. ETSI'nin (Avrupa Telekomünikasyon Standartları Enstitüsü) benzer bir sitesi var ama ETSI standartları ücretsiz! Bunların çȯg u, belirtim dili olarak ASN.1'i kullanır. Bu sitelere bȧg lantılar Ek 5 aracılığ iyla elde edilebilir.

Protokol spesifikasyonlarınız söz konusu olduğ unda (ancak ASN.1 spesifikasyonlarının kendileri dėg il), bir araç kullanacaksanız spesifikasyonun ASN.1 bdümlerinin elektronik bir kopyasını elde etmeye çalış mak önemli olacaktır, aksi takdirde bu metni girmek gibi sıkıcı ve hataya açık bir görevi olacaktır.

Aletinizin satıcısı muhtemelen size burada yardımcı olabilir ve ASN.1 spesifikasyonlarının elektronik kopyaları genellikle ücretsiz olarak dolaş maktadır ve bazen Web'de bulunmaktadır. Elektronik kopyanın baş ka bir kaynağı, elektronik sürümlerin ticari satıcıları olmaması ve belirtimlerin basılı sürümünü satın alı̇g inizi bilmesi koş uluyla genellikle bir tane sağ lamaktan mutluluk duyacak olan protokol belirtiminin Düzenleyicisidir.

Projeniz için bu spesifikasyonları zamanında almanız gerekecek ve her iki durumda da (ASN.1 spesifikasyonları ve sizin protokol spesifikasyonlarınız) muhtemelen bazı destekleyici spesifikasyonlara da ihtiyacınız olduğunu göreceksiniz ve bunların erken aşamada tanımlanması gerekiyor. proje.

ASN.1 spesifikasyonları söz konusu olduğunda, REAL, GeneralizedTime ve çoğu karakter seti tipinin kodlamasının tüm ayrıntıları, ek ayrı spesifikasyonlara atıfta bulunulmasını gerektirir; bu nedenle, protokol spesifikasyonunuzda bu tipler kullanılıyorsa, bu da er özelliklerini de elde edin.

Bir Standart da erine atıfta bulunduğuunda, her zaman referans verilen Standardın en son sürümünü kullanmanız ISO tavsiyesidir. Ancak bu bazen tehlikeli olabilir ve referans Standardın yayınlandığı sırasında referans Standardın hangi sürümünün güncel olduğunu görmek ve yapılan değişikliklerin protokolünüz üzerinde ne gibi bir etkisi olabileceğini görmek için yayın tarihlerini kontrol etmek her zaman iyidir..

1.2.2 Eğitim kursları, öğretime reticiler ve danışmanlar

Kolayca gözden kaçan (ve bunun için projeye dahil edilmeyen) da er bir maliyet de sizin için eğitim süresi ve kursların maliyeti süresidir.
uygulama ekibi.

Ticari kurslar ticaridir! (Ancak araç satıcınız, sizin için bazı kurslar ve eğitim materyalleri içeren bir pakete sahip olabilir).

ASN.1 ile ilgili bir "yalnızca teori" kursu (bu kitapla aşağı yukarı aynı teknik materyali kapsayan, ancak bu bölümde ve da er birkaç yerde görülen türden tartışılmalar olmaksızın) yaklaşık iki gün sürecektir. ASN.1 spesifikasyonlarını yazan ve bir araç kullanan bazı uygulamalı çalışmaların olduğu bir kurs, dört gün kadar uzun olabilir.

Bu tür kursları bu kitabı satın alarak da desteklemek isteyebilirsiniz! (Ya da Olivier Dubuisson'un eşlik eden cildi - hem Fransızca hem de İngilizce olarak mevcuttur. Bağlantı için Ek 5'e bakın.)

Benzer şekilde, ASN.1 kullanılarak belirtilen protokollerin çoğu una iyi bir giriş sağlayan ticari kurslar mevcuttur ve uyguladığınız protokol için bunlar mevcutsa, muhtemelen bunları kullanmak isteyeceksiniz. Sık sık konuşulan eğitmen/sunum yapan kişi bu protokolün standartlaşdırılmasında aktif olacaktır ve dolaşımada olabilecek herhangi bir ek ve düzeltmenin durumu konusunda sizin uyarabilir.

Son olarak, kendilerini "ASN.1 danışmanları" olarak tanıtan (az) sayıda insan vardır.

Uygulama tavsiyesi verecekler veya yazılmasını istediğiniz iniz bir protokolün ana hatlarını alıp sizin için ASN.1'i üretecekler. Ama danışmanlık ücreti ödüyorsunuz!

1.3 Uygulama platformu ve araçları

Mevcut bir sistemi genişletme ihtiyacı veya firmanızın küresel politikalarla uyumlu olmamak gereken bir nedeniyle uygulama platformunda (donanım, iş letim sistemi, programlama dili) seçenekleriniz olmayabilir. Çalışanlar.

Uygulama platformları hakkında karar almayıla ilgili birçok faktör vardır, ancak araç seçimi ile platform seçimi arasında etkileşimler olabilir.

Ancak bir seçenekiniz varsa, bir aracın kullanılıp kullanılmayacağına ve kullanılacaksa hangisinin kullanılacağına ilişkili kararla birlikte platform hakkında bir karar alınmalıdır. (Bir aracın "kalitesinin" özellikleri önceki bölümde tartışılmıştır ve burada dikkate alınmalıdır.)

En az bir araç satıcısı, o platform içinden bir C derleyicisi veya bir C çapraz derleyicisi olması koşuluyla, araçlarını herhangi bir platform içinden sağlayacaktır. C, C++ ve Java'da programlamayı destekleyen araçların tümü mevcuttur.

Belirleyiciler içinden 2 Sorun

Bu madde, ASN.1 kullanan protokol spesifikasyonunda yer alanların dikkate alınması gereken birkaç noktayı tartışır.

2.1 Kılavuz ilkeler

Akılda tutulması gereken dört ana ilke vardır (bazıları ASN.1 kullanılsın veya kullanılmamasın tüm protokol tasarıımı için geçerlidir). Bu ilkeler çok açık görünebilir, ancak çoğu zaman gözden kaçırılırlar:

Dört ilke? Elbette sihirli yedi ya da sadece bir olmalı! Şimdilik bu dördü yeter.

Sadelik: Gerektiği kadar genel ve esnek olurken, olabildiğince basit tutun.

Belirsiz ve eksiksiz: Belirtinizde hiç bir belirsizlik bırakmadığınızdan ve bilinçli olarak karar vermediğiniz sürece belirtiminizde hiç bir uygulama bağımılığı bırakmadığınızdan kesinlikle emin olun. İlkinci durumda, bu tür bağımlılıkların yalnızca örtülü veya gizli değil, açıkça ifade edildiğiinden ve bu tür bağımlılıkların birlikte çalıştırma sorunlarını tam olarak değil erlendirdiğiinizden emin olun.

Kaçınma seçenekleri: Kod özgücü uygulama maliyetlerini ve test maliyetlerini azaltacağının, çok iyi bir neden olmadıkça kodlayıcı seçeneklerinden kaçınmaya çalışın. Tüm spesifikasyonun hangi böümlerinin uygulanması gerektiğiine dair seçeneklere izin vermek de tehlikelidir ve dikkatli bir şekilde yapılmadığı takdirde birlikte çalıştırmayı ciddi şekilde sınırlayabilir (ancak genellikle yapılır!). Bu ilkenin ayrıntılı bir uygulaması "SET veya SET OF kullanmayın, bunun yerine her zaman Sequence veya Sequence of kullanın" der.

Bir sonraki sürümü düşündürün: Protokolünüz yükselirse her zaman bir sonraki sürüm olacaktır. Nasıl farklı olabilir? Eklenen malzemenin sürüm 1 sistemleri tarafından nasıl işlenmesini istiyorsunuz?

Bu ilkelerin çoğu, aşırıda daha ayrıntılı olarak açıklanan bazı özel ASN.1 özellikleri ve bunların kullanımıyla eşleşir.

2.2 Tarza iliş kin kararlar

En iyi tavsiye, olabildiğ ince çok farklı özellik e bakmanız ve çeşitli stil konularında bilinçli bir karar vermenizdir.

İngilizce metinden ayırmak için farklı yazı tiplerinin kullanılması, takip edilmesini kolaylaşdırır, käü olan ise zorlaşdırır. Hattaki gerçek bitler aynı olabilir!

Dikkate alınması gereken bazı basit şeyleler şunlardır:

Yazı Tipleri: Resmi materyali İngilizce metinden ayırmak için farklı yazı tiplerinin kullanılması.

Tanımların sırası: Tip tanımlarının yukarıdan aşağıya listesi mi yoksa alfabetik liste mi?

Modül yapısı: İlgili tanımların modüller halinde gruplandırılması ve modüllerin sırası ve genel yapısı.

Satır numaraları ve dizinler: Spesifikasyon için satır numaralandırmanın olası kullanımı ve bir dizinin sağlanması (tanımlandığı yerde ve her bir referans adı için kullanıldığı yerde gösteriliyor).

Referans adlarının uzunlukları: Uzun adlar daha net olabilir, ancak bir belirtimi karmaşıklaşdırabilir. İlişkili semantiği tanımlamak (ima etmek) için yalnızca isme güvenmeyin.

Yinelenen metin: Birkaç mesajın ortaklığı elere sahip olduğum durumlarda metni çok altmamaya çalışın, ancak bunun (örneğin) parametrelendirme kullanmaktan daha net olduğum durumlarda, belirtimi daha basit hale getiriyorsa bundan korkmayın.

Parametre sayısı: Bir referans adı tanımında çok sayıda parametreniz varsa, Bölüm II Bölüm 7'de açıklanmış gibi, bunları tek bir parametrede bir araya getirmek için bir Bilgi Nesnesi Sınıfı tanımlamayı düşünen.

Web yayını: Artık Web'de ASN.1'lerine (hatta tam belirtimine) sahip birçok standart vardır. Bazlarının benimsediği bir yaklaşım, bir referans adının her kullanımından bu adın tanımına hiper metin bağlanıları sağlar, ancak elbette bu durumda HTML'yi sizin için oluşturmak için bir ASN.1 aracına ihtiyacınız vardır, yoksa bu çok fazla olurdu sıkıcı ve hataya açık. Ayrıca bir ASN.1 derleyici aracına girdi giriş için belirtiminizin "ASCII" txt'sini sağlamanız gereklidir.

Diğer konular "üsluptan" biraz daha fazlasıdır veya bir madde işaretiyle sağlanabilecek olanlardan daha uzun bir tartışmaya garanti eder. Bunlar aşağıda tartışılmaktadır.

2.3 En üst düzey tipiniz

Mesajlarınızı tanımlayan üst düzey türün ne olduğunu çok net bir şekilde belirtmeniz gereklidir.

Bu, tek bir tür olmalıdır ve neredeyse her zaman geniş letilebilir bir SEÇİ M türü olacaktır.

Bu SEÇİ M'e, örneğin açık türler üzerindeki kısıtlamalarda kullanılabilecek türleri değil, tüm dış düzey mesajlarınızdan birini tanımlayan tüm türleri dahil edin.

Bu sizin mesaj grubunuzdur. Ona hak ettiğiniz önemini ve önemini verin.

Diğer tüm türler, bu türü desteklemek için vardır.

Bu üst düzey türü tanımlamak için ABSTRACT-SYNTAX göstergesini kullanabilir veya İngilizce metinle ve onu göze çarpan bir konuma - belki de kendi modülüne - yerleş tirerek çok net hale getirebilirsiniz.

ÖZET-SYNTAX, kısmen ASN.1'e görece geç bir tarihte eklendiğ inden ve kısmen de iliş kili nesne tanımlayıcı değil erine yalnızca tam OSI yığını kullanılyorsa iletişim imlerde ihtiyaç duyulduğ u için mevcut spesifikasyonlarda sıkılıkla kullanılmaz. üst düzey türlerinizi tanımlamanın çok net bir yolunu sağlar.

Geniş letilebilirlik işaretçisini kullandıñınız tüm durumlarda olduğ u gibi, sürüm 1 sistemlerinin sürüm 2'de eklenmiş mesajları almaları durumunda ne yapmasını istediğinizizi düşünmeli ve açıkça belirtmelisiniz. Bunu tanımsız bırakırsanız (uygulamaya bağlı), yukarıdaki dört ilkedenden birini ihlal ettiniz ve muhtemelen sizı isıracak!

2.4 Tamsayı boyutları ve sınırları

Bu ayrıntılı bir konudur ve yalnızca tamsayıların boyutuyla değil il, aynı zamanda dizilerin uzunluğu ve SEQUENCE OF ve SET OF yinelemeleriyle de ilgilidir.

Boyutları ve sınırları düşünen.
Çoğu durumda sonsuzluğ u kastetmiyorsunuz!

PER kullanıyorsanız, araçlar sınırlara göre kodlamalar gerçekleştiğ inden, sınırların resmi olarak alt tip notasyonu kullanılarak ifade edilmesi çok önemlidir. BER kullanıyorsanız, sorun kodlama değil il, ağız idakilerle ilgilidir:

Bu alanların dahili işlenmesinde hangi boyutta tamsayı kullanılmalıdır? (Seçtiğiniz programlama diline nasıl eşlenmelidirler?)

Herhangi bir belirtim sağlamazsanız, bir uygulama 4 sekizli tamsayılarla eşlenebilir (ve kodlayabilir ve iletебilir), diğer eri ise yalnızca 2 sekizli tamsayıları destekleyebilir. Yine de diğerleri, keyfi olarak büyük tamsayıları veya keyfi olarak uzun dizileri işlemek için yorucu ama gereksiz çabalar göstererek uygulama maliyetlerini önemli ölçüde artırabilir.

Bunlar elbette PER ile ilgili sorunlardır, ancak INTEGER türlerine sınırlar koyduysanız, uygulayıcı dahili olarak kullanmak için uygun tamsayı boyutunu anlayabilir.

Bir belirtim sınırlarla doluya, özellikle bunlar tek bir modülde ayarlanmışsa ve içe aktarılmışsa veya parametre olarak geçirilmişse, bu, belirtimi (bilgisayar için tamamen anlaşılırken!) bir insan tarafından daha az okunabilir hale getirebilir. Bir alternatif, kendi INTEGER4 türünü tanımlamak olabilir, ancak daha sonra bunun, onu kullanmak istediğiniz yere dışa ve içe aktarılması gereklidir.

ASN.1 araçları genellikle, ASN.1 INTEGER türünün eşleneceği programlama dili tamsayılarının boyutuna ilişkili genel ifadelere izin verir, böylece aksi belirtimedikçe, INTEGER alanlarının 4 olarak uygulanmasının bekleniği sırada 1'ningizde açık bir ifade sağlanır. sekizli tamsayılar yeterli olabilir.

Burada, PER kodlamaları kullanılırken hatta mümkün olan en küçük bit sayısını sağlamak için sınırların belirtilmesi ile programlama dili tamsayılarına ve dahili işlemeye eşleme için ne kullanılacağına ilişkili rehberlik arasında belirli bir gerilim olduğuna dikkat edin.

Bununla birlikte, kesinlikle hayatı olan, ASN.1 INTEGER tipi için çok büyük tamsayıların (X.509 sertifikalarındaki imzalarda görünenler gibi) ne zaman desteklenmesi gerektiğiinin açıklığa kavuşturulmasıdır.

Yukarıda esas olarak INTEGER üzerinde yoğunlaştık, ancak ağız idakilerin tümü ile ilgili sınır sorunları olduğunu unutmayın:

INTEGER değil erleri.

BIT STRING, OCTET STRING, karakter dizisi türleri ve GeneralizedTime uzunlukları.

Her SEQUENCE OF ve SET OF yineleme sayısı.

Ve her durumda, yukarıda belirtilen iki ana soruna sahipsiniz: optimum PER kodlamalarının sağlanması ve birlikte çalışmanın sağlanması. İkinci tartışıması siz daha önemlidir.

Kısım II Bölüm 7'de belirtildiği gibi, gerçek ekten bazı sınırları (veya başka herhangi bir şeysi) uygulamaya bağlı olarak bırakmaya karar verirseniz, soyut sözdiziminin bir parametresinin dahil edilmesi bunu açıkça işaretler ve ardından bir istisna işaretisi dahil edebilirsiniz. İki uygulama seçenekleri aynı değil ilse bir alıcının ne yapması gerekiyor ini belirtmek sınırlı. Bu rotayı izlerseniz, ıngilizce metinde neyi amaçladığınızı, uygulama bağlılığını ini bırakma nedenlerini ve bunun ne zaman birlikte çalışma sorunlarına yol açmasını beklediğini inizi (veya beklemeyi inizi) açıkça açıklamanız iyi olur.

2.5 Geniş letilebilirlik sorunları

Sürüm 2'de hangi uzantılara ihtiyaç duyacağınızı düşünenin öneminden ve uygun noktalara üç nokta koymanın öneminden daha önce bahsetmiştım.

Geniş letilebilirlik önemlidir ve işinize yarayacaktır - ancak yalnızca sürüm 2'yi yazarken kurallara uyarsınız!

Çoğu kişi, modül başlığıında EXTENSIBILITY IMPLIED kullanmaz, asırlı ödürlere yerine gerekiyorinde üç noktayı açıkça dahil etmeyi tercih eder. Bu muhtemelen daha açık ve istenirse her durumda ayrı istisna işlemeye izin verir (asırlı iyi bakın).

Dağıtılmış sürümler 1 sistemleriyle birlikte çalışmanın, bazı ayrı sürümler müzakereleri olmadan mümkün olmasını istiyorsanız veya sürümler 2 uygulayıcılarının "ikili yapıları" desteklemesini gerektirmenizi istiyorsanız, sürümler 2 belirtiminizde hangi değil iş iklikleri yapabileceğinizi ve yapamayacağınızı bilmek önemlidir.

Yalnızca sürümler 1'de elipslerinizi koymuş unuz yere malzeme ekleyebilirsiniz. Orijinal olarak "EXTENSIBILITY IMPLIED" yazmadığınız sürece, sürümler 2'de yeni üç nokta ekleyemezsiniz (elbette uzantı olarak eklediğiniz yeni türler dışındadır) veya kaldırılamazsınız elipsler. Ve mevcut türleri değil iş tıremezsiniz, örneğin:

TAM SAYI

ile

SEÇİ M { TAM SAYI , NESNE TANIMLAYICI }

"Yapamayacaklarınız"a son bir eklemeye (ancak bu liste kapsamlı değil ildir!) isteğe bağlılarından OPTIONAL veya DEFAULT ekleyemez veya mevcut işlerden çikaramazsınız (ancak isterseniz başka bir zorunlu iş ekleyebilirsiniz) daha önceki bir İSTEKI BAĞLI iş eylemi aynı türde olan üç noktanızda).

2.6 İstisna iş leme

2.6.1 Gereksinim

Üç nokta kullandığınızda, hangi davranışını beklediğinizin net bir ifade vermeniz kesinlikle hayatı önde taşıır:

Sürüm 1'e, sürüm 2'ye çarptığında ne yapması gerektiği söylemenmelidir - ve sürüm 2'yi yazarken ona ne yapmasını söylediğiniz hatırlamalısınız!

Ek malzeme alırlarsa sürüm 1 sistemlerinden.

Zorunlu alanların eklendiği sürüm 2 sistemleri, sürüm 1 sistemlerinden gelen mesajları nasıl işleyecektir.

Sürüm 2 eklemeleri genellikle İSTEĞE BAĞLI olarak işaretlenme eğiliminde olduğundan, ilki daha yaygın olan durumdur.

2.6.2 Yaygın istisna iş leme biçimleri

2.6.2.1 SIRALI ve AYAR

Bir SEQUENCE veya SET'e ilk eklenen öğeleri göz önünde bulundurun. Burada, bunların sürüm 1 sistemleri tarafından sessizce göz ardı edileceğini belirtmek son derece yaygındır (o zaman bunun sürüm 2 protokolünüzdeki sonuçlarını düşüneniz gereklidir).

Kapsamlı bir seçenek listesi bile yok.
Kesinlikle size ne yapacağınızı söylemiyorum! Ama belki de seni düşündürmeye yetecektir.

Önce en basit durumlar - sessizce girmezden gelin.

ASN.1 araçlarının kod çize yordamları içindeki bu tür malzemelerin kaldırılmasını desteklemesi muhtemeldir, böylece araca özel olarak belirtmek için işlem yapılmadığı sürece uygulama kodu, bir sürüm 2 mesajı tarafından vurulduğunda asla farkında bile olmaz. materyal aktarılmalıdır (örneğin, aktarma için).

2.6.2.2 SEÇİ M

SEÇİ M söz konusu olduğunda durum daha zordur ve protokolünüzde meydana gelen kesin etkileşimlere bağlı olacaktır.

Biraz daha zor. Ve üst düzey SEÇİ M genellikle biraz öyledir.

En basit durum, üst düzey SEÇİ M'İ Zdir; burada bir alış verişin başlangıçının üst düzey mesajlarına muhtemelen bazı tanımlanmış yanıtlar vardır ve bu yanıtlarında "Üzgünüm, bunu uygulamadım. Ben sadece sürüm 1 sistemiyim" göstergesi. (Böyle bir hükmün elbette 1. versiyon yanıt mesajlarında yapılması gereklidir.)

Şimdi, geniş letilebilir bir SEÇİ M'in bir diziye gönüllü olduğumu ve belki de sürüm 2'de yeni türlerin eklendiği bazı karakter dizisi türlerinin geniş letilebilir bir seçimi olduğumu düşünün.

Böyle bir türde bir sürüm 2'de eri alan bir sürüm 1 sisteminin, bu değil eri boş bir dize olarak ele alması - etkili bir şekilde yok sayması ve sonraki işlemde "Bu alan için uygun değil er yok" demesi mümkün olacaktır. Elbette detaylı protokolünüzü ve SEÇİ M alanının önemine bağlı olarak başka birçok işlem mümkün değildir. Neyin uygun olacağına yalnızca siz karar verebilirsiniz.

2.6.2.3 TAM SAYI ve SAYILANDIRILMIŞ

INTEGER üzerindeki geniş letilebilir aralıklar veya geniş letilebilir SAYILAR için durum net değil. Bir seçenek, herhangi bir yeni sürüm 2 değil erinin belirli bir sürüm 1 değil eriyle eş lenmesini (sürüm 1'de) tanımlamak ve bu değil erin iş lenmesini sürüm 1 davranış olarak belirtmek olabilir.

Başka bir zor. Tüm sürüm 2 değil erlerinin çok fazla soruna yol açmadan eşlenebileceği bir sürüm 1 değil eri var mı? Aksi takdirde, tamsayı veya numaralandırmanın sonraki iş lemleri nasıl etkileyeceğine bakmanız gereklidir.

Sürüm 1'i yazarken neden uzantıyı sürüm 2'de yapıyor olabileceğini inizi ve bu davranışın iş e yarayıp yaramayacağıını düşünmeye çalışın. Sonunda sürüm 2 eklemeleri yaptıgınızda bu tartışmayı tekrar ziyaret etmeniz gereklidir!

Bir sürüm 1 değil erine eş leme her zaman doğrudu olmayacağı ve bir sürüm 2 değil erinin varlığıının, birkaç ileri iş leme aşaması (hatta belki bir veritabanına) yoluyla "bilinmeyen bir değil er" olarak taşınması ve bunun daha sonra üzerindeki etkisi gerekebilir. Bu değil erin işleyen kod, sürüm 1'de tam olarak belirlenmelidir.

2.6.2.4 Geniş letilebilir diziler

Dikkate almamız gereken bir sonraki durum, sürüm 1'de sınırlı (ancak geniş letilebilir) bir maksimum boyutu sahip olan ve sürüm 2'deki boyut artırılmış dizelerdir.

Her ikisi de bariz olan iki ana seçenek: Sürüm 1'in iş leme düzeyinde daha uzun dizileri desteklemesini zorunlu kılmak veya kısaltın.

Burada yine sıkı bir PER kodlaması elde etmek için kısıtlamaları kullanma ihtiyacı ile uygulayıcıların sonraki iş lemlerde gerçekten desteklemesini istediğiımızda şunlar arasında bir çelişki görüyoruz.

Bu durumda (sürüm 1'de) kısıtlamanın sürüm 1 göndericileri için maksimumu belirlediğiini söylemek mümkün olabilir (şuna da gerekli görülen tek şunun budur), ancak sürüm 1 alıcılarının kendi uygulama boyutlarında işleyebilmesi gereklidir. Sürüm 1 sınırlarının iki katına kadar (diyalim) - ve belki bundan sonra kesilebilir.

Ancak yine de, dize alanının sonraki kullanımına ve iş lenmesine bağlı olarak, sürüm 2 değil erini "bilinmeyen değil er" olarak ele almak gibi seçenekler de uygun olabilir.

2.6.2.5 SET OF ve SEQUENCE OF üzerindeki geniş letilebilir sınırlar

Bu durum, dizilerdeki sınırlara çok benzer.

Beklediğiniz gibi dizelere çok benzer.

Alındıktan sonra daha büyük yinelemeleri desteklemek için sürüm 1 sistemlerinin zorunlu kılınması açıkça mümkündür. Yineleme sınırlarına kadar iş lediklerini ve ardından muhtemelen bir tür hatyla malzemenin geri kalanını (bir dizgenin kesilmesine eş değil er) yok sayıklarını belirtmek de mümkündür. dönüş.

Bununla birlikte, SET OF ve SEQUENCE OF yinelemelerindeki sınırlar nispeten nadirdir (uzatma işaretçileri olsun veya olmasın), bu nedenle bu durum sıklıkla ortaya çıkmaz. Ancak okuyucu, önceki metinden bunun potansiyel birlikte çalışma sorunları veya pahalı uygulamalar anlamına geldiğinin farkında olacaktır: Çok az uygulama, kendilerine böyle yapılması gerekiyor i söylenmedikçe sınırsız sayıda yinelemeyi gerektiren destekleyecektir.

Bununla birlikte, sorun, gerçek uygulama sınırlarının, bir uygulama programlama dili veri yapısına eşlenmediğinde, yineleme materyalinin toplam boyutunda olma ihtimalinin daha yüksek olmasıdır.

kendi baş ina yineleme sayısından ziyade. Bu belki de yineleme sayımlarındaki sınırların neden genellikle belirtilmeden bırakıldığı ina açıklar.

2.6.2.6 Kısıtlamalarda geniş letilebilir nesne kümelerinin kullanımı

Son olarak, ROSE'da olduğ u gibi geniş letilebilir bir Bilgi Nesnesi Kümesinin tablo veya iliş kisel kısıtlama olarak kullanıldığı durumu ele alıyoruz. Burada, bir sürüm 2 nesnesi alınırsa, ROSE REJECT mesajı gibi bir tür hata yanıtının olması yaygın bir durumdur.

Son öneğ imiz, hem en karmaşık hem de en basit!

Ancak diğ er durumlarda, sürüm 2 nesnesini sessizce yok sayma (belki ek bir "kritiklik" alaniyla bağ lantılı olarak) veya onu sürüm 1 nesnesi olarak ele alma seçeneğ i de olasılıklar olabilir.

2.6.2.7 Özет

Yukarıda altı ana mekanizma kullandık:

Sessizce görmezden gelin.

Bir çes it hata yanıtı verin.

Sürüm 1 değ erine veya nesnesine eş leyin.

Sürüm 1'e özel bir "bilinmeyen değ er" ekleyin ve iş lenmesini belirtin.

Eklenen malzemeyi veya bilinmeyen seçimi veya değ eri alın ve değ iş tirmeden aktarın.

Mümkün olduğ u kadar çok iş leyin, ardından kesin (sessizce veya bir tür hata yanıyla).

Daha önce altı mekanizma tarif edildi - lütfen biri baş ka bir tane bulsun ve sihirli yediye sahip olacağ iz!

Gerçek geniş letilebilir yapıya, bu yapının nerede kullanıldığı ina, onunla iliş kili anlambilime ve daha sonraki (belki çok daha sonra) iş lemeyle nasıl etkileştiğ ina bağ lı olarak, bu davranışlardan birini seçebilir veya belki de uygulamaya özel baş ka bir iş lemenin daha önemli olduğ unu belirleyebiliriz. uygun.

2.6.3 ASN.1 tarafından belirtilen varsayılan istisna iş leme

ASN.1, varsayılan istisna iş leme davranış inin belirtmediğ i için eleş tirildi, ancak umarım yukarıdaki seçenekler tartış ması, iyi ve uygun istisna iş lemenin belirli bir protokolün gereksinimleriyle ilgili olması gerektiğ ini ve sıkılık farklı yerlerde farklılık göstereceğ ini açıkça ortaya koymaktadır. protokol.

"Bu tur ş eylerle ug raş mak istemiyorum - neden sadece üç nokta koyup varsayılan istisna iş leme prosedürlerini baş latamıyorum?"
"Üzgünüm, iş e uygun - istisna iş lemeden (ve hakkında) sorumlu olma siziniz!"

Belirleyicilerin, farklı türde sürüm 1 istisna iş leme davranış inin sonuçları üzerinde düş ünmeden eksiltiler koymasına izin vermek pozitif olarak tehlikeli olacaktır. Üç nokta kolay bir seçenek de il . Baş langışta, verimli PER kodlamalarının, sürüm 1 ve sürüm 2 sistemleri arasında bir miktar iç çalış manın hala mümkün olacağı ş ekilde olmasını sağ lamak için tanıtıldı, ancak sürüm 2 eklemeleri, sürüm 1 davranış inin açık (önceki) bir özellikle olmadan yapılrsa, BER ile bile , ciddi sorunlar ortaya çıkar.

Zor olabilir, angarya olabilir, ancak geniş letilebilirlik sorunlarına ve ilgili istisna iş lemeye ciddi ş ekilde dikkat etmek, bir protokol belirleyicinin iş inin bir parçasıdır - iş , birkaç veri yapısını tanımlamaktan daha fazlasıdır!

Ne yazık ki, sürüm 1'de istisna iş leme konusunda kötü bir iş yapılrsa, sürüm 1'in kötü tasarımdan zarar görecek olan sürüm 2'yi üreten tamamen yeni (ve masum!) bir belirtici grubu büyük olasılıkla olacaktır. Ama korkarım ki hayat bu!

2.6.4 Resmi istisna belirtim gösteriminin kullanımı

Geniş letilebilirlikle ilgili bu tartışmayı bitirmeden önce, resmi istisna belirtim gösteriminin ("!" ile başlayan gösterim) kullanımına biraz değil inmeliyiz.

Resmi olarak söylemek ya da söylememek? Peki neden olmasın

Önemli olan (önceki maddede vurgulanan) istisna iş lemenin çok net bir ş ekilde belirtilmesi ve protokolde belirli iş lemenin kullanılacağı yerlerin açıkça tanımlanmasıdır. Üç noktanın nispeten az kullanımı varsa ve özellikle de gerekli istisna iş leme hepsi için aynıysa, zaman resmi istisna belirtim gösterimini dahil etmede gerçek bir kazanç yoktur ve İngilizce metin yeterli olabilir. (Yalnızca elipsler SEQUENCE yapılarının sonundaysa ve her durumda gereken davranış eklenen materyali sessizce yok saymaksa bu durum söz konusu olabilir).

(Aslında, bu tam olarak doğrudu - resmi notasyonun dahil edilmesi, bir okuyucuya istisna iş lemenin düşüncüsü ünü ve metinde bir yerlerde gerekli davranışın ayrıntılarının olduğuunu ve benim kişisel görüşüm resmi olması gerektiğiini söyler. geniş letilebilirlik in olduğu her yerde istisna belirtim gösterimi, ancak benimle aynı fikirde olmayan başkaları olduğuunu biliyorum!)

Belirtilen belki dört veya beş farklı istisna iş leme prosedürüne sahip bir protokolde (farklı üç nokta örnekleriyle kullanılacak, her davranışın birkaç üç nokta örneği ne uygulanacaktır), ardından resmi notasyonun kullanımı (belki sadece "!1", "!2, vb.) her üç nokta üzerinde, hangi davranışın hangisi için geçerli olduğuunu açıkça belirlemenin basit ve kullanımı bir yolu olabilir. Buna benzer bir şeyle, ROSE protokolünde çok etkili bir ş ekilde yapılır ("1", "2", vb. içindede referans adları kullanılarak).), Bölüm II Bölüm 6'da açıklandığı gibi.

2.7 Parametrelendirme sorunları

Parametrelendirme güçlündür ve özellikle bir grubun bir taşıyıcı protokolü sağladığını ve diğer birkaç grubun tam bir belirtim oluşturmak için boş lukları farklı şekilde doldurduğu durumlarda belirli "yeniden kullanılabilirlik" hedeflerine ulaşmanın tek yolu olabilir.

Bu kitabın başka yerlerinde söylenenlerin sadece bir tekrarı.

Ancak, parametreleşmiş bir tür, tek bir belirtim içinde yalnızca sınırlı sayıda başlıtlırsa, o zaman parametreleşmiş türme gereksiz olabilir ve aynı etki, farklı (ancak benzer) tür veya de farklı tanımları kullanılarak daha net bir ş ekilde elde edilebilir.

Soyut sözdiziminin Nesne Kümesi parametreleri, "tümünü uygulamalı, ancak ekleyebilir" ile "bir altkümeyi uygulayabilir, ancak ekleyemez" ile "bu bir kılavuzdur, ekle veya çıkar" gibi kesin belirtimler sağlamanın çok iyi bir yoludur. , ancak şunu andırır ASN.1 okuyucusuna yabancıdır ve açıklayıcı metinle birlikte sunulmalıdır.

Soyut sözdiziminin tamsayı parametreleri (sınırında kullanılır), belirtiminizde uygulamaya bağlı özellikler bırakmayı seçtiğiniz (her ne sebeple olursa olsun) açıkça belirtmenin çok iyi bir yoludur.

Ancak her iki durumda da, daha önce tartışıldığ gibi, istisna iş leme prosedürlerinin tam olarak belirtilmesi önemlidir.

{...} gösteriminin kullanımı, kullanılacak nesne setinin uygulamaya bağlı olduğunu ve genellikle parametreleş tirmeden daha az net ve kesin bir gösterim olduğunu bildiren bir parametreleş tirme biçimidir (ancak aynı fikirde olmayanlar da vardır!).

Bu notasyonun kullanılması önemlidir, bu metin, spesifikasyonun nasıl (kim tarafından ve nerede) tamamlanmasının amaçlandığıını ve birlikte çalışmayla ilişkini hangi çıkarımların olduğunu ve hangi istisna iş lemenin uygulanacağıını açıkça belirtir. Bu yapılsa, bu gösterim, en üst düzey türden bir kısıtlama olarak kullanıldıkları yere kadar iletilen çok sayıda farklı parametrede (çeşitli sınıflardan nesne kümeleri) daha az karmaşık bir belirtim üretebilir.

Son olarak, (Bdüm II, Bdüm 7), parametreleş tirilmiş bir türde (veya başka bir referans adı biçiminde) çok sayıda parametreniz varsa, bunları uygun bir Bilgi Nesnesi Sınıfı tanımlayarak tek bir nesne kümesi parametresine indirgelyebileceğinizini unutmayın (Kısim II, Bdüm 7). her parametre için eksiksiz bilgi setini taşıır. Bu çok yararlı bir basitleş tirme ve konuşmanızdaki ayrıntıları azaltma olabilir.

Metin.

2.8 Kısıtlanmamış açık tipler

Kısıtlanmamış açık tipler - şuna benzeyen dizilerin ögeleri, örneğin:

Söylenecek tek bir şevey var - YAPMAYIN!

Operasyon türü

ASN.1'de Bilgi Nesnesi Sınıfı kavramının Seul'de tanıtılmasının bir parçası olarak (bkz. "ve umarım okuyucu (en azından Bdüm II'yi okumuş olanlar) şimdiden kadar "ham HERHANGİ" (ve dolayısıyla sınırsız bir açık tip) KÖTÜ ŞEY olduğunu anlamıştır.

Bir aracın bu yapı için sunabileceğini tek şevey bir sekizli dizedir. Ve uygulamanın uygulayıcısı bile, bu ögelerde meydana gelebilecek olası türleri, bu türlerle ilişkili semantik ve belirli bir iletişiminde gerçekte hangi türün ortaya çıktığini bulmak için nereye bakılacağına dair net bir göstergeye sahip değil. , sekizli dizinin kodunun nasıl çözüleceğini ve yorumlanacağını.

2000'li yıllarda itibaren belirtici olarak, izin verilse de lütfen bu formu kullanmayın! Bu tür yapıların daha kesin ve uygulanabilir bir belirtiminin nasıl verileceğini görmek için ROSE bdümüne (Kısim II Bdüm 6) bakın. ASN.1 2010'da hala gücleniyorsa, kimsenin "1990, 1990!" (yine, Bdüm IV Bdüm 1'e bakın!).

2.9 Etiketleme sorunları

Yeni bir belirtim yazıyorsanız, OTOMATİ K ETİ KETLER kullanmalısınız (ve - bir yana - numaralandırmalar için numaralandırma değil erleri belirtmemelisiniz). Ancak mevcut bir spesifikasyona ekleme yapıyorsanız, hayat daha karmaşık olabilir.

Söylenecek tek bir şeyle var:
OTOMATİ K kullanın
ETİ KETLER

Metinsel olarak mevcut bir etiket yapısının bir SEÇİ M, SIRALI veya SET'te otomatik etiketlemeyi otomatik olarak devre dışı bırakıdıktan sonra unutmayın - kontrol tekrar sizdedir (ÖRTÜLU etiketleme ile).

OTOMATİ K ETİ KETLER kullanmamak için iyi nedenleriniz varsa, etiketleme konusunda çok daha iyi bir anlayış aşıhp olmanız gereklidir, ancak bu durumda modül başlığı inzada her zaman IMPLICIT ETİ KETLER kullanmalısınız. Modern spesifikasyonlarda açık bir etiketleme ortamı kullanmak kafa karıştırıcı olabilir ve ya çok ayrıntılı bir protokole (BER ile) ya da IMPLICIT kelimesiyle dolu bir spesifikasyona sahip olursunuz.

Belirli etiketlerin AÇIK olduğunu belirtmeyi seçerseniz, bunun nedenleri çoğu okuyucu için anlaşılması olactır ve bunun neden yapıldığıını metninizde belirtmelisiniz.

Genellikle iki olası neden vardır: Örtük etiketleme ortamında, bir seçenek türündeki etiketler aslında açık etiketler haline gelir. Bu, spesifikasyonda AÇIK kelimesini yazarak açıkça belirttilerse (bir bilgisayar için gereksizdir, ancak bir insana yardımcı olabilir) bir araç olmadan uygulama yapan kişi ilere yardımcı olabilir.

Diğer neden, bazı semantikleri veya kategorizasyonları belirli etiket değil erleriyle esasen ilişkilendirme ve (BER'de) tanımlanan gerçek türün etrafında bir uzunluk sargası olmasını sağlama arzusudur. Benzer bir motivasyon, PER kullanıldığıında açık bir tür üzerinde bir tür kısıtlamasının kullanılmasından gelir. Bu (olukça belirsiz) cihazların her ikisi de bazı güvenlik belirtimlerinde görünür.

Tabii ki, yukarıdaki tüm etiketleme tartışmaları, tip tanımlarınızı yalnızca bağ ımsız olarak yazmakla kalmayıp, tanımlanmış ASN.1 modül çerçevesi içinde yazdığını varsayar! Bu kitabın okuyucularının bunu asla yapmayacağından eminim!

2.10 Basit tutmak

ASN.1, net özellikler sağlama için bir dizi güçlü mekanizmaya sahiptir, ancak çoğu zaman, bazlarının daha basit bir belirtim çırakları için kullanılmamasını tavsiye eden insanlar bulacaksınız.

Bu yine Şekil 999 olayı - diğer erleri "basit olmayan"ın farklı bir listesine sahip olacak ve farklı önerilerde bulunacaktır!

Bunda bazen haklılık payı olabilir, ancak basit görünen şeyle, geçmiş te sıklıkla karşılıkla ilanlara çok bağlı olma eğilimindedir ve yeni notasyon yapılarının hazır bir kabul ve tanınma kazanması biraz zaman alabilir. Bir kez anlaşılıp tanıındıklarında, İngilizce metin alternatifinden daha net (ve dolayısıyla daha basit) bir belirtim sağlayabilirler.

Belirli yapıları kullanmamak için bazen ileri sürülen ikinci bir neden daha vardır; o da, bazı günümüz araçlarının bu yapıları kabul etmesi, ancak sadece "derleyici yönetgelerine" (genellikle özel bir biçim) güvenmek yerine bunları kullanmamasıdır. ASN.1 yorumu aynı etkiyi sağlar (ve bazı durumlarda notasyonun ASN.1'e girmesinden önce).

Bir veya her iki nedenden dolayı bu kategoriye giren gösterimler şunlardır (belirli bir sıra olmadan):

ABSTRACT-SYNTAX kullanımı.

Soyut sözdiziminin parametrelerinin kullanımı (değ iş ken kısıtlamalar).

Bir Açık Tür üzerinde bir tür kısıtlamasının kullanımı.

{...} gösteriminin kullanımı.

! istisna belirtim gösterimi.

Bunlardan herhangi birinden kaçınılmamasını tavsiye etmem, ancak bu yapıların (veya - henüz - yaygın olarak kullanılanmayan herhangi bir baş ka yapının) kullanıldığı i yerlerde, bir ASN.1 yorumu veya giriş eklemenin mantıklı olabileceğ i konusunda uyarıcı bulunmak isterim. Yapıların nasıl ve neden kullanıldığı ini ve bu protokol için kesin anlamlarını belirten, spesifikasyonun ana gödesindeki metin. Bu ş ekilde, bu tür yapılar herkes için tanındık hale gelecek ve "basit" hale gelecektir!

Uygulayıcılar için 3 Sorun

Bu bölüm, "belirleyiciler için sorunlar" bölümünden biraz daha kısadır, ancak daha önceki konuların birçoğ u burada yinelenmektedir. Aradaki fark, sizin (uygulayıcı) alıcı tarafta olmanızdır ve belirleyiciler belirsizlikler ürettiyse veya uygulama bağ ımlılıkları bırakırsa, bunları çözeniz gereklidir! (Uygulayıcılara ayrıca, bu bölümün önceki iki bölümünü ve tabii ki II. Bölüm tamamını dikkatlice okumaları tavsiye edilir.)

3.1 Kılavuz ilkeler

İ nternet uygulayıcıları için ilkeler genellikle ş uş ekilde ifade edilir:

Aslında vermek istediği imden emin değilim! Uygulama, özellikle bir araç kullanmayıorsanız, çok detaylı ve dağınık bir iş tir!

Gönderdiğ iniz üzerindeki spesifikasyona kesinlikle uygun.

Aldıklarınızda bağ iş layıcı olun.

Kulağ a iyi bir tavsiye gibi geliyor ve kesinlikle geçersiz olan ş eyleri anlayan ve iş leyen kodlar yazmak çok u zaman mümkündür.

Bu durum, İ nternet protokollerinde ASN.1 tabanlı protokollerden daha sık ortaya çıkar, çünkü metin tabanlı bir formatın kullanılması genellikle daha fazla fazlalık ve dolayısıyla resmi olarak yanlış kodlamaları "anlama" kapsamı sunar ve çoğu u İ nternet protokolü buna güvenir. Bir spesifikasyonun 1. versiyonu ile 2. versiyonu arasında karışılıkla çalışmayı sağlama prensibi. Durum, neredeyse hiç fazlalık i olmayan ve açık bir uzantı biti olan PER ile nadiren ortaya çıkacaktır!

İ lkel/yapıcı biti "ilkel" olarak ayarlanmış bir evrensel sınıf 16 etiketiniz (SEQUENCE) varsa, BER ile bağ iş layıcı olmaya karar verebilirsiniz. Veya evrensel sınıf 16'yi tespit ettikten sonra o biti kontrol etmek için kod yazmaya zahmet etmeyerek yanlışlıkla yasaklıyor olabilirsiniz!

Ancak hataları affediyorsanız (ilkel bir dizi veya belirtilen sınırları aşan tamsayılar diyelim), affedici olmanın etkisini dikkatlice düşünmelisiniz. Bu konu çok yakından ilgili

geniş letilebilirlik - sahip olduğunuz şeyle, zımnı geniş letilebilirliktir (kendinizin uygulamaya koymaya karar verdiğiiniz) ve en iyi istisna iş leme prosedürlerini tanımlamak için kendi başınızınız.

ASN.1 tabanlı protokoller söz konusu olduğunda, anlam verebildiğiniz (kendi uzantılarınız) yanlış kodlamaları sessizce yok saymanın ve iş lemenin nadiren iyi bir fikir olduğunu tavsiye ederim. İş leme devam etmeyi seçebilirsiniz, ancak hata (gönderenin ayrıntılarıyla birlikte) en azından bir yere kaydedilmeli ve protokol izin veriyorsa göndericiye bir tür hata mesajı olarak geri gönderilmelidir.

3.2 Aletinizi tanıyın

Herhangi bir geliş türme ortamında, seçilen araçta bir uygulayıcının hayatını kolaylaş tıtabilecek çok sayıda özellik vardır. Aracın bu özelliklerine/sekveneklerine/parametrelerine aşağı ina olmak önemlidir.

Bu büyükanneye yumurta emmeye öğretiyor!

Bir aracın "kalite" yönlerinden biri, sağladığınız işlevleri anlamaya kolaylığı ve bu işlevleri elde etmek için gereken ayrıntılı söz dizimiidir. Elbette, sağladığınız gerçek işlevleri daha önemli olarak düşünebilirsiniz, ancak ilgili belgelerde veya yardım dosyalarında açık olmayan veya çağrı ırması kolay olmayan işlevler, neredeyse eksik işlevler kadar kötüdür.

3.3 Tam sayıların boyutları

Bu konu, belirticiler (uygulayıcılar için de geçerlidir) bölümünde yoğun bir şekilde tartışılmıştır. Araçlar genellikle size eşlenikleri tamsayı uzunluğu üzerinde genel bir temelde (genellikle komut satırı parametreleriyle) kontrol sağlar, ancak aynı zamanda, genellikle "derleyici yönergeleri" - özel biçimler ASN.1 yorumu.

Tam olarak neyin amaçlandığıını bilmeniz gereklidir. İş eseri, iş arname size söyleyecektir. Aksi takdirde, iyi bir tahmin dört sekizlidir! Ama tahmin ederseniz, arkanızı kollayın - uygulama ekibinizde bir sorun olarak gündeme getirin.

Daha iyi araçlar ayrıca, belirli tamsayı alanlarının, mevcut belleğe bağlı olarak keyfi olarak büyük olmalarına (dinamik bellek təhsisi kullanılarak) izin vermek için dizeler olarak ele alınacağına belirtmenize de olanak tanır.

İki sorununuz var:

Protokol tanımlayıcısının amacını yorumlama.

İşte stediğiniz şeyle resmi iş arnameyi bir parçası değil ilse veya onunla çelişiyorsa, aletinizin istediğiniz işini yapmasını sağlama!

İkinci sorun, aletin kalitesine bağlıdır. Bu nedenle, protokol belirtiminiz bir alanın "INTEGER (0..7)" olduğuunu söyleyorsa, ancak bunun (programlama veya bir veritabanına yazma kolaylığı için) bir alan yerine dört sekizli bir tamsayı ile eşlenmesini istiyorsanız, seçtiğiniz iş programlama dilinde iki veya bir sekizli tamsayı, bunu yapabiliyor musunuz?

İki daha zor problem olabilir! Belirleyiciler, bu bölümde daha önce verilen bu alandaki yönergelere/tavsiyelere uyduysa, sorun yaşamasınız, ancak aksi halde ihtiyacınız olabilir.

(uygulama bilgisinden ve spesifikasyonun diğ er bromiumlarından veya baş kalarından bilgi alarak (aşağı bakın)), niyetin ne olduğunu veya baş kalarının bunu nasıl yorumladığını tahmin etmeye çalışmak.

3.4 Spesifikasyonlardaki belirsizlikler ve uygulama bağımlılıkları

Kutuya inanmayın! Tamamen temiz (özellikle yayınlanan ilk spesifikasyonda) ve uygulamanın her koşulda üretmesi gereken şartındaki bitleri tamamen belirtmiş bir spesifikasyon yazmak zordur. (Bunu söylemekten nefret ediyorum ama iyi yapırsa, belirleyicinin işi uygulayıcınınkinden daha zordur, ancak belirleyicinin durumunda işi kötü yapmak ve fark edilmemek çok daha kolaydır!)

Özellikle ISO ve ITU-T'de olmak üzere protokol üzerinde spesifikasyonu ~~üzerinde~~ ~~çalınan~~ veya gizli uygulama bağımlılıkları olan spesifikasyonlar üretme konusunda iyi beceriye sahip hiç kimse olmayacak!

Uygulayıcılara en önemli tavsiye - ve bu çok önemlidir - söylememiş şeyle bulursanız, bunları bir sorun olarak dile getirin, en azından ekibiniz içinde, ancak tercihen uygun bir posta listesi veya grup aracılığıyla belirticilerin kendileriyle birlikte.

Bazlarınız Alternatif Bit Protokolünü duymuş olacak. 1970'lerin sonlarında belirli bir LAN üzerinden kullanım için çok benzer bir protokol belirtilmiş idi (isim yok, paket tatbikatı yok!), ancak şartname, yanlış numaraya sahip bir ACK alındığında davranışının ne olacağıını söylemedi. Uygulayıcılar, "doğru" eylemin, alıcının kopyaları atacağının güvenerek, son mesajı (aynı sıra numarasıyla) hemen yeniden iletmek olduğunu una karar verdiler. Sonuç: parazit iletimi. Verim, yük geri çekilene kadar yarıya düşer ve her paket iki kez iletildi!

Uygulayıcılara düşen net bir görev varsa, oda şartnamelerin net olmadığından durumlarda kendi kararlarını almamaktır!

3.5 Düzeltmeler

Uygulayıcıların, düzeltmelerin ne olduğunu, durumları ve gelecekte uygulamayı nasıl etkileyebileceğini konusunda daha yönetsel kapasiteye sahip olanlar kadar farkında olmaları gereklidir.

Yönetim sorunları hakkında bu bölümün başında verilen tavsiyeyi okuyun - düzeltmeler için endişelenin!

Bir şeylein geleceğini biliyorsanız, planlanmış sahneyle çok daha az acı verici olabilir!

3.6 Geniş letilebilirlik ve özel durum iş leme

Bu metin kendini tekrar ediyor! Teldeki parçaların ne olması gerekiyor (ve bunlara karşılık olarak ne yapacağınız) ve gelen bilinmeyen şeyleler nasıl başa çıkılacağıınız size açıkça anlatılırsa ve kod çözme aracınız yeterince iyi ve esnekse, o zaman hiç bir şeyle yoktur. problemler.

İyi bir teknik özellik ve iyi bir aletle, hiç problem yaşayamazsınız!

Aksi takdirde endişelen!

3.7 El kodlamalarına özen gösterin

Herhangi bir nedenle INTEGER gibi basit türleri kodlamak için iyi hata ayıklanmış bir rutinler kitaplığına bile erişiminiz yoksa, tam teş ekküllü bir ASN.1 derleyicisine erişiminiz bir yana, o zaman sempatiyi hak ediyorsunuz!

Alet yok mu? Hayat zor olacak.
Dikkat olmak. (İyi olma seçeneğin kaybettin!)

ASN.1 kodlamalarını sıfırdan elle üretmek imkansız değil il ve bir anlamda zor bile değil il. (Ama ne yazık ki, PER'deki çok sayıda optimizasyon nedeniyle, BER ile ilk seferde doğru yapmak muhtemelen PER'den daha kolaydır.) Sadece zaman alıcıdır ve hataya açıktır.

Her şayden önce, Bölüm III'ü normalde okuyacağınızdan çok daha dikkatli okumalısınız! O halde, kullanacağınız gerçek ASN.1 kodlama özelliği ile çok zaman harcamanız gereklidir.

İkincisi, ürettiğiniz şayi, amaçladığınız şayi ürettiğiniz kontrollü etmenizi kolaylaşdırır. bir biçimde görüntülemek için bir tür ad hoc "hat izleme" aracına ihtiyacınız olacak.

Ve son olarak, gerçek ekten bir ASN.1 aracına ihtiyacınız var! Platformunuzda mutlaka çalışın ve değil il (bunun olmaması muhtemelen bir araç kullanmamızın nedenidir), ancak başka bir iletişim platformunda çalışabilen, çıktılarınızı alabilen ve ürettiğiniz düşündüğünüzde de erleri görüntüleyebilen bir platform.

Eh, bu neredeyse son oldu! Tamamen farklı bir tam uygulama ile nihai birlikte çalışabilen testi gibisi yoktur, özellikle de (ve siz!), aldığınız şayle ilgili hatalı olduğumu düşündüğünüzde şayeleri iyi bir şekilde günlüğe kaydediyorsa.

3.8 Posta listeleri

Genel ASN.1 sorguları için kullanabileceğiniz bir posta listesi vardır (buna bağlı lantı için Ek 5'e bakın) ve günümüzde birçok protokol belirtimi posta listeleri, haber grupları, Web sayfaları vb. tarafından desteklenmektedir.

İçeri girin!

Bu kaynaklar sizin için çok değerli olabilir. (ASN.1 ve özel protokol kursları veren, genellikle size e-posta adreslerini bırakmak ve kurslarından sonra soruları yanıtlamak isteyen kişiler gibi).

3.9 İyi mühendislik - sürüm 2 **gelecek**!

Uyguladığınız herhangi bir protokol, sizin veya torunlarınızın (takım bazında) uygulamak zorunda kalacağınız bir sürüm 2 spesifikasyonuna sahip olacaktır.

Geniş letilebilirlik sadece belirleyiciler için değil ildir.

Tüm olağanüstü mühendislik ilkeleri, kodunuzun ve belgelerinizin başkalarının uygulamanızı sürüm 2 spesifikasyonunu destekleyecek şekilde değil iş tırebilmesini sağlamak için uygulanır.

1. sürümün geniş letilebilirlik hükümlerinde, belirticilerin değil iş meyi bekledikleri alanlar hakkında bazı ipuçları alacaksınız. Bu, uygulamanızın yapısını, geldiklerinde bu değil iş iklıklere kolayca uyum sağlayacak şekilde tasarlamanıza yardımcı olabilir.

İstisna işlemeyi mümkün olduğuk kadar doğru yapmak belirticiler için bir zorluk olduğuk gibi, uzantıların kolayca işlenmesini sağlayan bir uygulama mimarisi elde etmek (ve henüz test edilecek sürüm 2 sistemleri olmadığıında sürüm 1'de doğru istisna işlemeyi sağlamak) uygulayıcı için zorluk. Belirleyicilere gelince - bu işinizin bir parçası, doğru anlayın!

4. Sonuç

Ve bu, kitabın bu ilk Bölümünü tamamlıyor. Birçok unuz bu noktada bizden ayrılacaksınız (gerçi Bölüm IV'ün bazı kısımlarını ilginç bulabilirsiniz). Umarım faydalı bulmuşsunuzdur.

Daha teknik düşüncenler şüphesiz Bölüm II ve III'ye ilerleyecektir - okumaya devam edin!

BÖLÜM II

Daha fazla ayrıntı

Bölüm 1

nesne tanımlayıcı türü

(Veya: Bir isimde ne var?)

Özet: Nesne tanımlayıcı türü ve onunla ilişkili hiyerarşik ad alanı, ASN.1'i kullanan protokol belirticileri tarafından yoğun bir şekilde kullanılır. Herkesin kullanabileceği dünya çapında kesin bir adlandırma şeması sağlar ve çok çeşitli "şeyleri" adlandırmak için kullanılmıştır.

Nesne tanımlayıcıları şunları tanımlamak için kullanılır:

- a) ASN.1 modülleri
- b) Özet ve aktarım söz dizimleri
- c) Yönetilen nesneler ve nitelikleri
- d) Dizin Bileşenleri (X.500) adları
- e) MHS mesajlarının başlıklar (X.400) ve MHS Gönde Tipleri
- f) Güvenli Elektronik İşlemlerde Bankalar ve Üye İşyeri
- g) Karakter Repertuarları ve kodlamaları
- h) Kargo firmaları tarafından takip edilen koliler
- i) Ve diğer birçok "şey" veya "bilgi nesnesi".

1. Giriş

Nesne tanımlayıcı türünün son tartışması bu "Daha Fazla Ayrıntılar" Bölümüne ertelenmiştir, ancak bir tür gösterimi olarak BOOLEAN kadar basittir. Sadece şunu yaz:

NESNE TANIMLAYICI

Nesne tanımlayıcıları, 1986'da ASN.1'e, ad alanının herkes tarafından kolayca edinilmesine izin veren, küresel olarak benzersiz kısa tanımlayıcılarla sahip bir ad alanına yönelik artan ihtiyacı karşılamak için dahil edildi

hepsi büyük harf. Karmaşık ılklik, bu türdeki değerler kümelerinde ve değerler notasyonunda ortaya çıkar.

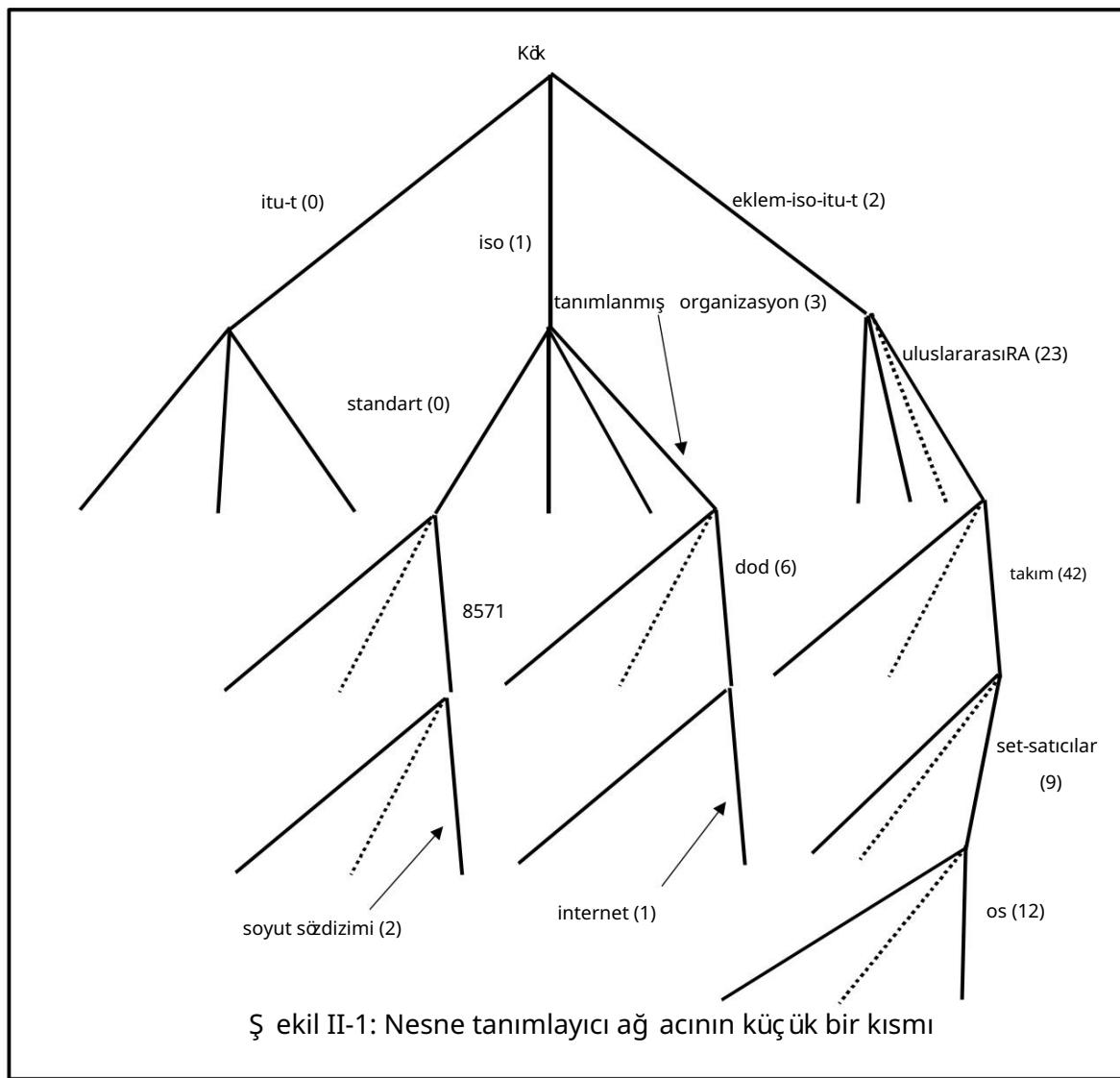
İlk olarak, değer erler dizisinin her gün dinamik olarak değer iş tiğini ve hiç bir bilgisayar sisteminin (veya insanın) tüm yasal değer erlerini bilmesinin beklenmediğini not etmeliyiz. Değer er gösteriminin bir yapısı vardır ve her nesne tanımlayıcı değer eri, bir basit tamsayı değer erlerini dizisine eşlenebilir, ancak bu yapılar önemli değildir. Atomik bir varlık olarak ele alınan bir nesne tanımlayıcı değer eri (ve onunla ilişkili semantik), bir uygulama tarafından bilinir veya bilinmez.

Tüm önemli olan bu.

Bu tür bir bilgisayar protokolünde kullanıldığıında, neredeyse her zaman, alınan bir nesne tanımlayıcı değer eri bilinen bir değerle eşleşmemiye gerekli olan açık bir istisna işlemi özelliğinin olduğunu (veya olması gerekiği) durumlarda kullanılır.

Tüm geçerli ASN.1 kodlama kurallarının, tüm kodlama kuralları için aynı olan ve aynı zamanda sekiz bitin (sekizli dize) tam bir katı olan, nesne tanımlayıcı değerlerinin kanonik bir kodlamasını (kodlayıcı seçenekleri yok) sağladığını unutmayın. Bu nedenle, semantik ASN.1 kodlamasını içeren basit sekizli diziler olarak bilinen bu nesne tanımlayıcı değerlerinin saklanması ve gelen kodlamaların bunlarla karşılaştırılması uygun bir uygulama seçenekidir.

Modüllerini tanımlamanın bir yolu olarak bu türden değer erlerle zaten karşılaştırılmış ve bazı değer er gösterimlerini gördük. Şimdi bu tür değerlerin altında yatan modeli ve



nesne tanımlayıcı ad alanı.

2 Nesne tanımlayıcı ağı acı

Nesne tanımlayıcıları için temel kavram, genellikle ş ekil II-1'de gösterildiği gibi çizilen bir ağı aç yapısıdır. Her nesne tanımlayıcı değil eri, kökten aş ağı iya bir yaprağı (veya muhtemelen dahili bir düğüm) kadar tam olarak bir yola karşı ılk gelir ve değil er notasyonunun her bir bileşeni, bu yolda katedilen yaylardan birini tanımlar.

Ağı acın tek bir kökü vardır (bilgisayarda ağı açlarda olduğ u gibi genellikle tepeden çizilir!) ve bir sonraki seviyeye giden birkaç yay (tüm yaylar sadece bir sonraki seviyeye gider), bu seviyede düğümler sağlar. Bir sonraki seviyedeki her düğüm bir sonraki seviyedeki düğümürlere doğrudan yolları vardır ve bu böyle devam eder. Hem ağı acın derinliği hem de her düğümünden yay sayısının sınırsızdır. Ağı acın bazı dalları alt yollarla yoğun bir şekilde, diğer erleri seyrek olarak doldurulacaktır. Bazı dallar erken bitecek, diğer erleri çok derine inecektir.

Her düğüm bir otorite tarafından yönetilir. Bu otorite, o düğümün altındaki yayları tahsis eder, alt bir düğümüne götürür ve şunları belirler:

Daha fazla tahsis için (alt düğümün altında) sorumluluğu devredildiği otorite veya bu (yaprak) düğümle ilişkili bir bilgi nesnesi. ("Bilgi nesnesi" kavramı aş ağı içinde ayrıntılı olarak ele alınmaktadır.)

Geçerli düğümünden alt düğümü tanımlamak için bir sayı (mevcut düğümünden tüm yaylar içinde net) (sıfır yukarı, mutlaka ardışık değil).

İsteğe bağlı olarak, insanlar tarafından kullanılmak üzere yay ile ilişkilendirilecek bir ad ve yine mevcut düğümünden gelen yaylar içinde tanımlama sağlar.

Üçüncü madde işaretindeki adın, bir değil er-referans-adı için ASN.1 kurallarına uyması gereklidir - yani, küçük harfle başlamalı ve (herhangi bir durumda) harflerle, rakamlarla ve kısa çizgilerle devam etmelidir (ancak iki ardışık tire olmadan).

"ccitt", "itu-t" haline geldiğiinde, ASN.1 standartlaşdırıcıları, adların eş anlamlarını zimnen kabul etti. yaylar.

Belki de bu nedenle, birçok ASN.1 kullanıcısı artık arkadalarının nispeten öneksiz olduğunu (kesinlikle hat üzerindeki bitleri etkilemediklerini) ve (sayısal) bir nesne tanımlayıcı tahsisini elde ettiğinizde, kendinizi tanımlamak veya düğümünüz altında tahsisleri yayımlamak istediğinizde, seçtiğiniz herhangi bir adla o nesne tanımlayıcısını için değil er gösterimini kullanabilirsiniz. Hatta bazıları, daha yüksek seviyeli düğümlelerde kullanılan adları değil iş türmeyi bile ileri sürebilektir.

1999 yılının ortalarında olduğ u gibi, bu alan bir değil iş im halindedir. Daha önceki görüşler, isimlerin bir yayın üstü tarafından tahsis edildiğiini ve değil iş mez olduğunu söyleyerek, aksi takdirde insan kafa karışıklığına çok yer vardır. Bununla birlikte, asıl amacın bu olduğunu bilmeme rağmen, şartnamedeki metin bu görüşü tamamen desteklemiyor!

Aksi görüş (yayınlanan OID'lerde herhangi bir adın kullanılabilmesi) iki gereklilikle desteklenmektedir:

Üst düzey düğümlerin genellikle dahil olmak istemedikleri adların telif hakkı veya ticari marka sorunları vardır, bu nedenle alt yaylarına ad tahsis yapmazlar, yalnızca bir sayı yaparlar.

Alt yaylar bazen, isimleri kendileri ve kdk arasındaki yayları tanımlayan organizasyonlara bağlı (veya onların parçası) görünme konusunda hassas olabilir. Çoklu durumda, böyle bir ilişkilendirme en iyi ihtimalle gevşek bir ilişkilendirme olabilir, bunu isteyen herkese nesne tanımlayıcı alanı verir.

Standardın, yalnızca bir nesne tanımlayıcı için değil er gösteriminde adların isteğiyle bağlı olduğunu söylemek, aynı zamanda bu tür tüm adların, yaynlarda nesne tanımlayıcı değil erleri içerenler tarafından keyfi olarak seçildiğiini iddia edecek şekilde aşıklığı kavuşturulması muhtemeldir. Bununla birlikte, kavislerde yanlıltıcı isimler kullanmak sorumsuzluk olur ve ismi atlamak ya da düğümünü işaret eden kavislerin üzerinde genel olarak tanıtan ismi kullanmak muhtemelen en iyisidir.

3 Bilgi nesneleri

NOT - "Bilgi nesnesi" terimi, NESNE TANIMLAYICISI metninde "Bilgi Nesnesi Sınıfı" kavramlarının tanıtılmamasından çok önce kullanılmıştır ve (belki kafa karıştırıcı bir şekilde) Bilgi Nesnesi Sınıfları ile bağlı olarak kullanılan aynı sözcüklerden daha genel bir kavramı ifade eder.

Bu bağlamda kullanılan bilgi nesnesi terimi, nesne tanımlayıcılarının genellikle ASN.1 modülleri, bir bilgisayarın gerçek ekles tırebleceğinde bazı işlemelerin tanımı, bazı sistemlerin bazı sistemler tarafından manipüle edilebilecek öznitelikleri gibi nispeten soyut nesneleri tanımlamak için kullanıldığı vurgular. bir yönetim protokolü vb. Başka bir deyişle, genellikle bir belirtim parçasını tanımlarlar (ASN.1 kullanılarak yapılması zorunlu değildir). Aslında, bir organizasyon başka bir tür bilgi nesnesi olarak görülebilir ve genel olarak bir düğüm hem bir bilgi nesnesiyle (herhangi bir türden) ilişkilendirilebilir hem de başka alt düzey ümlere sahip olabilir.

Bir organizasyona bir düğüm atanmışsa, onların ağa açtan "asıldıktır" söyleriz. Bir düğümün gururlu sahibi olduğu unuzda cansız nesneleri (ASN.1 modülleri gibi) ağa açtan "asmak" da mümkündür!

Dağıtılmış kayıt yetkilileri, herkes için yeterli alan sağlar. Henüz Nesne Tanımlayıcı ağa acına asıldınız mı? Aksiyondan bir parça alın!

Ağa acın üst kısımlarını öğrenmek ve ardından "hile yapmak" çok kolaydır. Bir düğümden bir yayı "çalmak", bunun altında tahsisleri yayılmamak. Yapma! "Yasal" nesne tanımlayıcı ad alanını elde etmek zor değildir. Ama şe 999'a bakın altındaki yayların yalnızca çok kapalı bir topluluk içinde net olduğunu üst düzey bir yayı savunanlar var - herkes herhangi bir sayı kullanabilir ve dikkat edin! Bunun asıl söylemek istedigi, bazı Nesne Tanımlayıcı değil erlerinin başlama özgü olması gerektiği yönünde bir öneri olduğunu ve bu tür tüm değil erlerin özel bir üst düzey yay tarafından tanımlandığıdır. Ancak, bu teklif sadece - bir teklif. Bodium IV'te tartışılan BAĞL OID tipi benzer bir rol oynamasına rağmen, böyle bir üst düzey yay mevcut değildir (1999 ortası).

Bir organizasyonu veya nesneyi tanımlamak için bir nesne tanımlayıcı değil eri kullanır. Soyut düzeyde, bu basitçe kdkten tanımlanan kuruluş veya nesneye giden bir yoldur. Bu yol, sırasıyla her yayın numarası ve bu yayların her biri ile ilişkilendirme adalar (boş / yok olabilir) verilerek belirtilebilir. Kodlama kuralları yalnızca yay numaralarını kullanır, bu nedenle var olmayan adlar sorun olmaz. Değil er gösteriminin çeşitli imzaları vardır (bkz.

Aşağıda), hem adların hem de sayıların belirtilmesine izin verir. § ekil II-1, aşağıda küçük bir bölümünü, 4 ve 5 yay derinliği ile alınmış iki dalı göstermektedir.

4 Değer gösterimi

Aşağıda tüm örneklerde, herhangi bir sayıyı INTEGER türünde bir değer er referans adıyla değer iş tirmenin yasal olacağıını unutmayın. Bu değer er referans adına aşağıda örneklerde verilen değer er atanmış sa, ortaya çıkan nesne tanımlayıcı değer eri değer iş mez. Ancak bunu yapmak yaygın bir uygulama değildir.

Değer er gösterimi, tanımlanmış bir nesneye giren her yay için bir tane olmak üzere bir dizi bileşenin oluşur. § ekil II-1'de, şe klin altındaki nesneleri şe ekilde tanımlayabiliriz:

ve {iso standardı 8571 soyut sözdizimi (2)}
 ve {iso tanımlı kuruluşu dod (6) internet (1)}
 ve {joint-iso-itu-t internationalRA (23) set (42) set-vendors (9) oss (12)}

veya eş değer er, ancak daha az okunabilir şe ekilde:

ve {1 0 8571 2}
 ve {1 3 6 1}
 ve {2 23 42 9 12}

İlk değer er, ISO Standardı 8571'de bir bilgi nesnesini adlandırır, ikincisi, IETF'ye nesne tanımlayıcı alanı verir ve bunun alt yayları, SNMP (Basit Ağ Yönetimi Protokolü) için İnternet spesifikasyonunda yoğun bir şe ekilde doldurulur. Üçüncü değer er, Güvenli Elektronik İşlemler (SET) konsorsiyumu ile ilişkili bir satıcı olan Open Systems Solutions'a nesne tanımlayıcı ad alanı verir.

Yalnızca sayıların kullanılmasına her zaman izin verilir (ancak yaygın değildir). Bir durumda "8571" bir yayın bir numarası vardır ancak adı yoktur, bu nedenle sayı parantez içinde değer il, tek başına görünür. Diğer er çoklu durumda, ad verilir ve ardından parantez içindeki sayı gelir. (Her ikisi de verilmiş se numara parantez içinde yazılmalıdır). 1988 öncesi ASN.1 spesifikasyonunda listelenen sayısal değer erleri ile "iyi bilinen" yayalar olduğundan, yalnızca üst yaylar içi (iso, standart, ortak-iso-itu-t) sayılar atlanabilir. (artık X.660/ISO 9834-1'de listelenmiş tir). Spesifikasyonların bu üst düzey sayıların çıkarıldığı görülmeli olduğça yaygın olmakla birlikte, özellikle ASN.1 artık ITU-T veya ISO ile yalnızca gevşek bir şe ekilde ilişkilendirilen (veya hiç ilişkilendirilmeyen) kuruluşlar tarafından kullanıldığından, giderek daha fazla uygulama haline gelmektedir. tüm yaylar içi parantez içindeki sayıları listeleyin.

Bu değer er gösteriminin bileşenler arasında virgül içermemiş olduğunu dikkat edin. Bu, ASN.1 değer er gösterimi için alışmadık bir durumdur ve özellikle sayıların çıkarıldığı ilk bileşenlerin kolay okunabilirliğini artırınmak için yapılmıştır.

Nesne tanımlayıcı değer erlerini belirtirken kullanılabilen başka bir olanak vardır. Beş bileşenli bir nesne tanımlayıcı değer eri "wineco-OID" tanımlamayı seçtiğimiz ve ardından bu adı IMPORTS deyimimizde süslü parantezden hemen sonra kullandığımız şe ekil 21'de zaten tanışmıştık. (Yalnızca süslü parantezden hemen sonra izin verilir). Bu oldukça sık yapılan bir şe eydir, ancak referans kapsamı olarak modül tanımlayıcısı içi buna izin verilmemişini unutmayın.

modüldeki adlar henüz girilmemiş tir. Bazı spesifikasyonlar, özellikle bilgi nesnelerinin tanımıyla bağ lantılı olarak çok sayıda nesne tanımlayıcı değil eri tanımlayacaktır ve çok yaygın bir tarz, bu değil erleri tek bir modülde bir dizi değil er referans adına atamak ve bu isimleri dış aktarmaktır. Daha sonra ithal edilecek ve gerektiğinde diğ er modüllerde kullanılacaktır.

5 Nesne tanımlayıcı tipinin kullanımları

"Şeylerin" tanımlanmasına ihtiyaç duyulan bir protokolün yazılması yaygın bir durumdur. Bu "şeyler" şular olabilir:

ne olduğunu:

üzerinde çalışın;

sipariş;

raporlama;

taşındığını bilgiler;

bir mesajın alınması üzerine gerçekleştirecek belirli eylemlerin tanımlanması;

Dizin (X.500) adları gibi bazı daha karmaşık yapıların bileşenleri;

vesaire vesaire.

Bazı mevcut kullanımlar, bu bölümün başındaki "Özet" bölümünde listelenmiş tir.

"Şeyler" için "bilgi nesneleri" terimini kullanıyoruz, çünkü günün sonunda fiziksel bir "şey" bir metin veya belirtimle tanımlanır - bir bilgi parçası ve bazen "şey" fiziksel değil ildir. nesne ama oldukça soyut bir "şey", böyle bir organizasyon, ama "şey" yine de bazı spesifikasyonlarla tanımlanıyor - bir bilgi parçası. Bir nesne tanımlayıcı değil eri tarafından gerçek ekten tanımlanan şey, o şeyin daha ayrıntılı ve kesin belirtimidir - "şey"in kendisinden ziyade bir "bilgi nesnesi"dir, ancak ikisi 1-1 yazışma içindedir, bu nedenle gerçekten hiçbir şey yoktur. ayrılm.

Bir bilgi nesnesinin tanımlanmasına ihtiyaç duyulduğunda:

dünya çapında net olması gereken; ve

bu tür bilgi nesnelerine kimlik tassislerinin hemen hemen herkes için geniş çapta erişilebilir olması gerektiğinde; sonra

ASN.1 nesne tanımlayıcı değil erlerinin kullanılması iyi bir yoldur.

Genel olarak, ASN.1'in hemen hemen tüm kullanıcıları, uygulamalarıyla ilgili bilgi nesnelerini tanımlamak için bir adlandırma şemasına ihtiyaç duymuş ve bu amaç için nesne tanımlayıcı değil erlerini kullanmayı ve protokol alanlarına NESNE TANIMLAYICI olan nesneleri dahil etmeyi seçmiş tir. türler bu tür değil erleri taşımak için. NESNE TANIMLAYICI türü ve onunla ilişkili adlandırma yapısı önemlidir ve yoğun bir şekilde kullanılır.

Bölüm 2

Karakter dizisi türleri

(Veya: Genesis Bölüm 11'in Üstesinden Gelmek!)

Özet: Bu bölüm, karakter dizisi tiplerinin tamamını ele almaktadır:

sayısal dize

YazdırılabilirDize

GörünürDize (ISO646Dize)

IA5Dizesi

Teleteks Dizisi (T61 Dizisi)

VideotexDizesi

grafik dizesi

GenelDize

evrensel dize

BMPString

UTF8Dizesi

Değerler gösterimlerini açıklar ve kullanımları hakkında tavsiyeler verir.

Karakter dizisi "delik" türü - KARAKTER Dİ Zİ Sİ - tartışması bu bölümün 7. Bölümüne kadar ertelenmiş tir.

1. Giriş

Burada, daha sonra "Delik Tipleri" altında açıklanan "CHARACTER STRING" dışında mevcut (1988'e kadar) tüm karakter dizisi tiplerini açıklayacağınız. Bu türlerin tam olarak anlaşılması için okuyucunun, genellikle yıllar içinde bilgisayarlar için karakter kodlama şemaslarına yönelik çeşitli yaklaşımaların farkında olması gereklidir. Bunun ve ASN.1'deki karakter dizisi türleri için destek in tarihsel gelişimin tam bir tartışması Böyük IV'te verilmektedir.

Burada ASN.1 spesifikasyonlarının yazılması ve anlaşılması için yeterli bilgi verilmişdir. Bu materyalin bir kısmını atlamanız istiyorsanız, "Önerilen karakter dizisi türleri" (madde 13) bölümne gidin ve orada belirtilenlerle ilgili paragraflara bakın. Muhtemelen ihtiyacınız olan tek şeys budur!

Ve Tanrı, Babil halkından cennete kulelerini inşa ettikleri için hoşnutsuzdu ve bir yıldırım gönderdi ve insanları farklı diller vererek dünyanın dört bir yanına dağıttı.

Karakter dizisi türleri, bazıları tarafından gereksiz olarak kabul edilir (eski güzel bir OCTET STRING işi yapmaz mı?). (Bkz. ş. ekil 999!). Evet, bir OCTET STRING kullanılabilir. Ancak daha sonra kullanılacak kesin kodlamayı açıkça belirtmeniz ve uygulayıcılara desteklenecek karakter aralığıni netleştirmeniz gereklidir. Ayrıca, bu belirtim normal insan tarafından okunabilir metinde veya ASN.1 yorumunda olacak, uygulamaya yardımcı olan herhangi bir araç tarafından anlaşılması sağlanacak ve (yeni metin olduğu için) potansiyel bir belirsizlik ve birlikte çalışma sorunları kaynağı olacaktır.

ASN.1'de sağ taraflı tipler, en basit gereksinimlerden en iddialı olanlara kadar bir yelpazeyi kapsar. Genel olarak, belirli bir dizi için karakter seti gereksinimleriniz kısıtlıysa, kodlama daha geniş bir karakter dağılığına sahip bir tiple aynı olsa bile, bunu netleştirmek için daha kısıtlı karakter seti tiplerini kullanın.

Ayrıca, en yeni karakter dizisi türlerinden bazılarının yalnızca Unicode kodlama şemasını destekleyen, karakter başına 16 bit kullanan bir programlama dili (Java gibi) tarafından kolayca desteklenebileceğiini unutmayın. (Bu şeyle, Böyük IV'te tam olarak açıklanmaktadır). Bununla birlikte, giderek artan bir şeyle (1990'ların sonrasında) programlama dilleri ve iş letim sistemleri ve tarayıcılar ve kelime iş lemcileri ve'nin tümü, karakter başına 16 bitlik repertuar için veya bazı durumlarda 32 bitlik bir repertuar için Unicode desteği sağlıyor. -karakter başına repertuar.

Bu, uygulama tasarımcısının bir alanı (örneğin) olarak belirttiğine anlamına gelmez. UTF8String veya UniversalString, bu protokolü Unicode desteği olmayan bir dilde (veya iş letim sisteminde) uygulayamazsınız, bu sadece daha zor olabileceğine anlamına gelir!

2 Sayısal Dize

Türün değil erleri, sıfırdan 9'a kadar olan rakamları ve boş luk içeren karakter dizileridir. BER kodlaması ASCII'dir (karakter başına 8 bit) ve PER kodlaması, karakter repertuarı "izin verilen alfabe kısıtlaması" ile daha fazla sınırlanmadığı sürece karakter başına 4 bittir (aşağıındaki Böyük 3'e bakın), daha az olabilir.

3 YazdırılabilirDize

Türün **değ erleri**, ASN.1 belirtiminde bir tabloda tanımlanan ad hoc karakter listesini içeren karakter dizileridir ve buraya **Ş ekil II-2** olarak kopyalanmış tır.

Bu temelde eski teleks karakter seti artı küçükük harflerdir. Sınırlı karakter giriş i veya görüntüleme yetenekleri olan cihazlarla ilişkilendirilecek bir uygulamanız olmadıkça, muhtemelen bugün onu kullanmama eğilimindesiniz.

İ sim	grafik
Büyük harfler	A, B, ... Z
Küçük harfler	bir, b, ... z
rakamlar	0, 1, ... 9
Uzay	(Uzay)
Kesme iş areti	'
Sol Parantez	(
sağ parantez)
Artı iş areti	+
Virgül	,
Tire	-
nokta	.
Solidus	/
Kolon	:
eş itir iş areti	=
Soru iş areti	?

Ş ekil II-2: PrintableString'deki Karakterler

4 GörünürDize (ISO646Dize)

"ISO646String" adı, **VisibleString** ile kullanımdan kaldırılmış bir eş anımlıdir (ad, aslında 1986 sonrası tanımında kullanılmayan bir Standart sayı iç erdiğinden kullanımdan kaldırılmış tır!), ancak bununla karşılık abilirsiniz. Karakter repertuarı, daha iyi bilinen ASCII'nin temelini oluştururan çok eski ISO Standardı ISO 646'da açıklanmış tır. Bu karakter repertuarı başlangıçta

kesinlikle ASCII değil, "ISO 646'nın Uluslararası Referans Sürümü", tüm ASN.1 kullanıcıları ve uygulayıcıları tarafından basit düz ASCII olarak, ancak yalnızca karakterler artı boş luk basılması olarak yorumlandı. Orijinal tanım, ISO 646 Standardına atıfta bulunuyordu, ancak 1986 sonrası tanım resmi olarak "Kaçış Dizileriyle birlikte kullanılacak Uluslararası Kodlu Karakter Setleri Kaydının Kayıt Giriş i 2 (artı boş luk)" idi. (Daha fazla ayrıntı için B'düm IV'e bakın). Bu, 1994 yılında, katı ASCII olan ve ASN.1 kullanıcıları tarafından yapılan normal yorumu tanıyan "Kayıt Giriş i 6"ya referans verecek şekilde değil iş tirildi. BER'deki kodlama karakter baş ina 8 bittir ve karakter aralığı ina kısıtlamak ina türde uygulanan bir alt tipleme yoksa PER'de aynıdır (varsa daha az olabilir).

5 IA5Dizesi

"Uluslararası Alfabe 5", yine bu tür için orijinal referans olan çok eski bir ITU-T Tavsiyesinde belirtilmiş tir. Yine, bu ASCII'ye yakındı (ASCII, Uluslararası Alfabe 5'in "ulusal bir varyantydi", ancak türün yaygın olarak "kontrol karakterleri, boş luk ve del dahil olmak üzere ASCII'nin tamamı" anlamına geldiğ i varsayılıyor. Bugün kesin referans, "Escape Dizileri ile kullanılacak Uluslararası Kodlanmış Karakter Setlerinin Kayıt Giriş İleri 1 ve 6 (artı boş luk ve silme)", katı ASCII'dir. Kodlama yine karakter baş ina 8 bittir (muhtemelen PER'de daha az).

6 Teleteks Dizisi (T61 Dizisi)

Yine, eş anlamlı kullanımından kaldırılmış tir. Başlangıçta CCITT Tavsiyesi T.61, Teletex için karakter repertuarını belirtilmiş tir ve ASN.1 spesifikasyonu tarafından referans alınmış tir. (Bugün ilgili özellikler ITU-T T.50 serisindedir.) Bu türün kesin tanımı, ITU-T teleteks Önerileri tarafından desteklenen artan dil yelpazesini yansıtacak şekilde zaman içinde değil iş miş tir. Bugün Urduca, Korece, Yunanca, Resmi olarak, Kayıt Giriş İleri 6, 87, 102, 103, 106, 107, 126, 144, 150, 153, 156, 164, 165, 168, artı BOŞ LUK ve Sİ L!

Her kayıt giriş inin kodlaması, karakter baş ina 8 bittir, ancak farklı kayıt giriş İleri arasında geçiş yapmak için tanımlanmış çıkış kodları (ASCII "ESC" kodlaması ve ardından bazı tanımlanmış sekizli değil erler) vardır. Bu karakter dizisi türü in tam destek uygulamak oldukça zordur, ancak X.400 ve X.500 çalışmasında yaygın olarak kullanılmaktadır. Referans verilen karakter repertuarları, ASN.1'in her yeni sürümüyle birlikte artmış tir ve kendileri de dünyanın giderek daha fazla karakter setini destekleme baskısı altında olan ITU-T Teleteks Önerileri ile uyumu sürdürme baskısı altında bunu yapmaya devam edebilir. . Bu, bu tür etkili bir şekilde açık uçlu bir karakter repertuarları haline getirir ve herhangi bir "uygunluk" iddiasını tanımlamayı veya sürdürmeyi zorlaştırır. Bugün en iyi kaçınılmazı gereken şeyle, ancak 1980'lerin ortalarında popülerdi ve sık sık karşılaşırsınız.

7 VideotexDizesi

Videotext sistemlerinde ham resimler oluş turmak için kullanılan "karakterlere" eriş im sağılayan, az kullanılan bir karakter dizisi türü. Tipik olarak bir "karakter", her hücrenin ya bir önen rengi ya da bir arka plan rengi (yaklaşık beş kontrol karakterinden birinin iletilmesiyle belirlenir) içerdig i ve resmi oluş turmak için kullanılabilen 64 farklı yazdırma "karakteri" veren 3x2'lik bir dizidir. .

Resmen, yine 17 kayıt girişinin bir listesidir ve TeletexString için belirtilenlerle kısmen örtüşür.

8 Grafik Dizesi

Bu, 1980'lerde üretilen ana OSI (Açık Sistemler Ara Bağlantısı) standartlarında popüler bir dizi türüydü ve karakterlerin yazdırılması için Uluslararası Kayıt'taki Kayıt Girişlerinden herhangi birine izin verdi (ancak kontrol karakter girişlerine izin vermedi). En parlak dönemde, Uluslararası Sicil yaklaşım her ay yeni bir kayıt ekledi ve sonunda dünya dillerinin çoğunu kapsadı. Bu metin akademik bir derste kullanılrsa, bu kadar geniş (ve sürekli genişleyen!) bir tür tanım kullanımının uygulama sonuçlarını tartışmak ilginç bir öreni alıştırması olacaktır. ISO 10646/Unicode'un gelişmesinden bu yana, Uluslararası Sicile yapılan eklemeler çok daha az yaygın hale geldi ve bu Sicile dayalı kodlama şemasını eskimiş olarak kabul edilebilir.

9 Genel Dize

Bu, (çok sayıda olan) kontrol karakterleri için kayıt girişlerinin de kullanılabilmesi dışındadır. GraphicString'e benzer.

10 Evrensel Dize

Bu, ISO Standardı 10646'nın tamamlanmasının ve Unicode belirtiminin yayınlanmasıının ardından 1994 yılında ASN.1'e eklenen bir dize türüdür (ISO 10646 ve Unicode hakkında daha fazla bilgi için Bölüm IV'e bakın). ISO 10646 standarı (ve BER'deki ASN.1 kodlaması), karakter başına 32 bitlik bir kodlama şeması öngörür; Mars ve keşfedilmemiş Evrenin geri kalanının çoğunu! Yalnızca bu tür ve UTF8String (aşağıya bakın), bilgisayar kodlamalarının tanımlandığı tüm karakterleri kapsayabilir (tam olarak doğrudur değil - Uluslararası Kayıtta henüz ISO 10646'ya yerleşmemiş bazı garip glifler var). Ancak bu türün ASN.1 kullanıcıları arasında popüler olduğu kanıtlanmamıştır.

UNICODE, UniversalString tarafından ve BMPString tarafından UTF8String tarafından desteklenir. **VE**

11 BMPSdizesi

Adı, ticari öneme sahip tüm karakterleri (tüm yaş ayan diller) içeren ve karakter başına sabit 16 bit ile kodlanabilen (ve BER'de olan) ISO 10646'nın "Temel Çok Dilli Düzlem"inden (BMP) gelir. Resmi ASN.1 tanımı ISO 10646'ya atıfta bulunurken, karakter seti, Unicode Konsorsiyumu tarafından üretilen Unicode Standardında tanımlananla aynıdır ve daha yaygın olarak Unicode Standardı olarak bilinir. (Hakkında daha fazla bilgi edinmek istiyorsanız Web'de arama yapın).

Unicode, kürek bkz. Bölüm IV). Unicode karakterleri tutan, karakter başına 16 bitlik sabit boyutlu temsil, programlama dilleri ve iş letim sistemlerinin revizyonlarında yaygınlaşıyor ve karakter verilerini işlemek için varsayılan kodlama olarak hızla ASCII'nin yerini alıyor. Bu ASN.1 tipi, 1990'ların ortalarında, 1994 ASN.1 spesifikasyonuna yükseltilen uygulama spesifikasyonları tarafından yaygın olarak kullanıldı. (1994 öncesi ASN.1'de mevcut değildi).

12 UTF8Dizisi

UTF8String, gereksiz ayrıntılar olmadan tam uluslararası tırma için önerilen karakter dizisi türüdür.

Bu kodlama şeması 1990'ların ortalarında gelişti ve 1998'de ASN.1'e eklendi. Kısaltma "Evrensel Dönüşüm Formatı, 8 bit" anlamına gelir, ancak bunun pek bir önemi yoktur. Resmi olarak, karakter repertuarı UniversalString ile tamamen aynıdır - tanımlanan tüm karakterler temsil edilebilir.

Bununla birlikte, UTF8, (7-bit) ASCII karakterlerinin ASCII olarak kodladığını oldukça ilginç bir özelliğe sahip, her karakter için de işlenen uzunluklu bir kodlamadır - üst bit sıfır ayarlanmış tek bir sekizlide ve göstergemdeki sekizlilerin hiç biri ASCII olmayan bir karakterin üst biti sıfır ayarlanmışdır. ASCII her şeyleden önemlidir! Çoklu Avrupa dili karakteri (c-cedilla veya u-umlaut gibi) iki sekizli olarak kodlanır ve Temel Çok Dilli Düzlemin tamamı, size kadar tanımlanan tüm karakterlerle birlikte, karakter başına en fazla üç sekizli olarak kodlanır. ISO 10646 32-bit alanının tamamını doldurursak, UTF8 karakter başına maksimum altı sekizli kullanır.

Karakter başına sabit bir 16-bit kullanımı, iş letim sistemi arayüzleri ve programlama dilleri için norm haline gelirken, karakter verilerinin depolanması ve iletilmesi için UTF8'in kullanımı herkesin gittiği yoldur (1999 ortalarında olduğu gibi). ASN.1 tabanlı bir uygulamanın uygulayıcısı olarak, Unicode'u destekleyen bir dile sahip bir ASN.1 aracı kullanırsanız, araç tarafından ASN'nin bir parçası olarak size görünmez bir şekilde UTF8 dönüşümünün uygulanmasını bekleyebilirsiniz. 1 kodlama/kod çözme işlemi, size karakter başına basit bir 16-bit (veya 32-bit) vererek bellekte, ancak verimli bir aktarım söz dizimi ile çalışır.

13 Önerilen karakter dizisi türü

Yani sonuna kadar okuduktan sonra, artık hangi karakter dizisi türlerinin kullanılacağı konusunda bilinçli bir karar verebilirsiniz! Burada, yeni bir belirtim yazdıgınızda ve 1994 sonrası ASN.1'e uygun olacağından ve dolayısıyla en son ASN.1'deki tüm olanakları kullanabileceğiniz varsayılmıştır. (1994 öncesi/1994 sonrası konuların daha kapsamlı bir tartışması Bölüm IV'te yer almaktadır.)

Uygulamanızın beklenen uygulaması için ilgili giriş / çıkış cihazlarının tam Unicode karakter setini işleyebilmesi muhtemel ve mümkün olduğunda genel olmak istiyorsanız, UTF8String tam size göre! Daha önceki UniversalString ve BMPString, varsa bile çok az avantaj sunar ve dikkate alınmamalıdır. Ancak,

Tam uluslararası tırma için UTF8String. Aksi takdirde, ihtiyaçlarınız için mevcut olan en kısıtlayıcı karakter dizisi türünü kullanın. Giriş / çıkış cihazları uygulamanızı kısıtlıyorsa, NumericString veya PrintableString veya VisibleString veya IA5String'i düşün.

giriş veya çıkışın daha sınırlı cihazlarda yapılması muhtemeldir, bu durumda daha kısıtlı bir karakter dizisi tipini düş ünebilirsiniz.

International Register'a dayalı GeneralString ve GraphicString modası geçmiş ve 1980'lerde önemli olmalarına rağmen yeni spesifikasyonlarda kullanılması söz konusu değil.

Aynı açıklama TeletexString (T61String) ve VideotexString için de geçerlidir: ilgili ITU-T Önerilerine güçlü bağ lantlarınız olmadığı sürece bunları kullanmak istemeyeceksiniz.

Uygulamanız yalnızca sınırlı sayıda karakteri destekleyebilecek giriş / çıkış aygıtlarının kullanılmasını gerektiriyorsa, yalnızca NumericString, PrintableString, VisibleString (ISO646String) veya IA5String kullanmayı ciddi olarak düşünmelisiniz. NumericString çok sınırlıdır ve tamamen uluslararası değil, ancak uluslararasılaş tırma açısından diğer er üçünden daha iyidir (arapça sayılar, ASCII karakterlerinin tamamından daha fazla dünyada kabul edilir). PrintableString, ASN.1'e fiziksel olarak bağlı olması gibi küçük bir değilere sahiptir, bu nedenle hangi karakterlerin dahil edildiği konusunda hiç bir yanılış anlaşılma olamaz, ancak esasen ASCII'ye göre çok az avantajı olan, kısaltılmış bir ASCII'dir. Tam ASCII istiyorsanız, VisibleString'e (kontrol karakterleri içermez) veya IA5String'e (kontrol karakterleri içerir) ihtiyacınız vardır. Bu, İngilizce konuşan topluluklar için iyi olacaktır ve diğer er bazı Avrupa dilleri için yaş anabilirdir, ancak genellikle herhangi bir uluslararası spesifikasyonda onaylanmaz.

Sonuç olarak, uygulama tasarımcısı olarak seçim sizin olmalıdır - ASN.1 yalnızca notasyon araçları sağlar, ancak seçimini muhtemelen NumericString, PrintableString, VisibleString, IA5String ve UTF8String ile sınırlamak istersiniz. Giriş / çıkış aygıtlarının uygulamanızın uygulamalarında güçlü bir belirleyici rol oynaması muhtemel değil ilse (örneğin, tüm ilgili giriş / çıkış klavye giriş ve gösterimi için genel amaçlı bilgisayar yazılımı kullanacaksa) UTF8String'i kullanmalısınız.

14 Karakter dizisi türleri için değil er gösterimi

Bu kitap, ASN.1 gösterimini tam kapsamını vermektedir, ancak bu gösterimin nadiren ihtiyaç duyacağıınız veya karşılıkla açağıınız bazı bromiumları vardır. Karakter dizileri için değil er gösterimi bu kategoridedir ve kontrol karakterleri veya birkaç dilde görünen karakterler için değil er notasyonuna daha da az ihtiyaç duyulur. Bu bromiumu atlayarak okuyun ve ihtiyacınız olduğunda fark ederseniz daha sonra geri dönün!

Adalar tüm UNICODE Karakterleri için mevcuttur ve gliflerin belirsizliği veya yayın ortamınızda bulunan karakter kümesi hakkında endişe duymadan karakter dizisi değil erlerinin belirtimine kesinlik vermek için ASN.1'de kullanılabilir. Hücre başı vuruları da kullanılabilir.

1994 öncesi karakter dizisi türleri için tek değil er gösterimi, karakterleri tırnak işaretleri içinde listelemekti. Bu, PrintableString gibi basit repertuarlar için iyidi, ancak kontrol karakterlerinin IA5String gibi bir tür için belirtmesini sağlamadı ve aşırı idakiler gibi dizelerle yazdırılan belirtimlerde belirsizlik sorunlarına yol açtı:

"ÜMİ: T ETMEK"

repertuar ASCII'nin yanı sıra Kiril ve Yunanca içeriyorsa! (Bu dört glifin her biri, bu alfabelerin birden fazlasında bir karakter olarak görünür). Karakter dizisi değil erlerinde hangi beyaz boşluğun temsil edilmesinin amaçlandığıını (kaç boş luk, "ince" boş luk, vb.) Belirlemede basılı belirtimlerde de potansiyel sorunlar vardır.

1994'ten sonra, her ikisi de karakterleri ayrı ayrı listelemeye dayalı olarak, bir karakter dizisini tam olarak tanımlamak için iki ek mekanizma mevcuttur.

Notasyon aşağıdaki şekilde gösterilmektedir:

```
my-string1 UTF8String ::= {cyrillicCapitalLetterEn,
                           yunan Büyük Harf Omicron, latin Büyük Harf P,
                           Kiril Büyük Harf Ie}
```

```
my-string2 IA5String ::= {nul, soh, etx, "ABC", del}
```

```
my-string3 UTF8String ::= {{0, 0, 4, 29}, {0, 0, 3, 159}, {0, 0, 0, 80}, {0, 0, 4, 21}}
```

```
my-string4 IA5String ::= {{0, 0}, {0, 1}, {0, 3},
```

```
"ABC", {7,
          15}}
```

Tahmin edecek iniz gibi, my-string3, my-string1 ile aynıdır (ve "HOPE" olarak yazdırılabilir!) ve my-string4, my-string2 ile aynıdır. Son iki notasyon, ISO 10646 veya ASCII'nin (resmi olarak Uluslararası Kayıt Kayıt Giriş'i 6'nın) hücrelerine (grup, düzlem, sıra, hücre vererek) atıfta bulunur (tablo sütununu 0 ile 7 ve tablo satırını 0 ile 15 olarak verir).

Son iki notasyon serbestçe kullanılabilir, ancak ilk iki notasyonda kullanılan karakter adları, yalnızca ASN.1 belirtiminde karakter adlarına göre tanımlanan (algoritmik olarak) bir modülden modülünüzü aktarıldığında kullanılabilir. ISO 10646'da (ve Unicode) atanmıştır.

Yukarıdaki değerler göstergelerini geçerli kılmak için modülünüzde aşağıdaki IMPORTS ifadesine ihtiyacınız vardır:

```
İ THALATLAR cyrillicCapitalLetterEn, yunanCapitalLetterOmicron, latinCapitalLetterP, cyrillicCapitalLetterIe,
nul, soh, etx, del FROM
ASN1-KARAKTER-MODÜLÜ
{joint-iso-itu-t asn1(1) belirtimi(0) modülleri(0) iso10646(0)};
```

Ayrıca, farklı göstergeleri (karakter adları, alıntılanan dizeler, hücre referansları) tek bir değer tanımlama içinde karıştırabileceğinizde not edeceksiniz.

Yukarıdakiler işe yarar, ancak "UMUT"unuz aslında ASCII karakterleri olarak tasarlanmışsa, 1998'den sonra daha az ayrıntılı bir yöntem mevcuttur. Basitçe yazabilirsiniz:

```
my-string5 UTF8String(TemelLatin):= "UMUT"
```

burada "BasicLatin" ASN.1 modülünden içe aktarılır. Daha sonra, bir Dİ Zi içinde, bir ögeye sahip olabilirsiniz:

```
string-element UTF8String DEFAULT my-string5
```

Burada yaptığıımız oldukça açık - UTF8String tipini yalnızca BasicLatin (ASCII) kısmını kullandığımızı söylemek için "niteliyoruz", bu nedenle "UMUT" artık açık bir şekilde ASCII karakterleridir. SEQUENCE'da tam UTF8String tipini kullandığımızı unutmayın. Bu oldukça basit notasyon, iki güçlü ve genel kavrama, alt tipleme ve değil eş leme kavramlarına dayanır.

Alt tipleme, sözde üst tipin değil erlerinin yalnızca bir alt kümesini içeren yeni bir tipin tanımıdır. Bu durumda üst tür "UTF8String"dir ve biz bunun alt türünü (ASN.1 modülünde tanımlanan) "BasicLatin" olarak adlandırarak burada alt tür olarak kullanıyoruz. Yukarıdaki örnek aslında şöyleden yazılabildi:

```
my-string5 BasicLatin ::= "UMUT"
```

bu, "my-string5" in latin karakterler olduğuunu belki daha net hale getirir, ancak UTF8String için bir VARSAYILAN değil er olarak kullanılabilirini (yine de kullanılabilirse de) daha az açık hale getirir. Alt tiplendirme bir sonraki bölümde daha ayrıntılı olarak ele alınacaktır. "my-string5" hangi şekilde tanımlanırsa tanımlansın, UTF8String için varsayılan değil er olarak kullanımı, ASN.1'deki genel bir konsepte bağlıdır; eğ er bir şeyle, bir tür alt tipin bir değil er referans-adıysa, şeyle de kullanılabilir: üst türün bir değil eri içen bir değil er referansı adı ve bazı durumlarda değil er "benzer" türler. Bu, ASN.1 semantik modelindeki değil er eş leme kavramıdır (Kısım I'de kısaca tanıtılmış ve Bölüm IV'te daha ayrıntılı olarak tartışılmıştır) ve bu durumda "my-string5" in yalnızca UTF8String için bir değil er olarak değil il, aynı zamanda ayrıca dilerseniz PrintableString ve VisibleString için bir değil er olarak.

15 ASN.1-KARAKTER-MODÜLÜ

Bu modülden yukarıda bahsedilmiş tir. Tüm ASCII kontrol karakterleri (açıkça listelenmiş tir) ve Unicode/ISO 10646'daki tüm karakterler için değil er referans adları sağlar. ISO 10646 Standardında (ve Unicode) listelenen karakter adlarının tümü büyük harflerle ve boş luklarla verilmiş tir. Kelimeler arasında. Bir ASN.1 ismine döner türmek için, ilk isim hariç her kelimenin ilk harfinin büyük harfini koruyun, değil er tüm harfleri küçük harfe çevirin, ardından boş lukları kaldırın! Bu, yukarıda kullandığımız adları ve ayrıca oldukça uzun adı üretir:

```
cjkUnifiedIdeograph-4e2a
```

(Batılı bir göze!) üzerinde şeyle bulunan dikey bir çubuk gibi görünen ve ISO 10646'da "CJK Unified Ideograph-4e2a" olarak adlandırılan Çince/Japon/Korece (CJK) karakteri için..

ISO 10646 ayrıca 84 koleksiyon tanımlar - kırışılı karakter grupları. Bu adlar, aynı algoritma tarafından UTF8String'in alt türleri için ASN.1 adlarına eşlenir, ancak tür olduklarından (tek karakter değil erleri değil il, dize değil erleri kümeleri) ilk büyük harflerini korurlar. İle şeyle aktarılabilen adlara birkaç örnek:

- TemelLatince
- Latinice-1Ek
- Latinice Geniş İletilmiş -A
- Ipa Uzantıları
- TemelYunanca
- Üst Simgeler ve Alt Simgeler
- MatematikselOperatörler
- Kutu Çizimi vb.

16 Sonuç

ASN.1 karakter dizisi türleri, karakter seti standartlarının kendileri deð iş tikçe ve giriş /çıkış aygıtları ve paketler daha geniş ve daha geniş bir karakter yelpazesini iş leme konusunda daha yetenekli hale geldikçe zaman içinde gelişmiş tir.

Kısmen, herhangi bir karakter repertuarı ve kodlama şemasını barındıracak bir mekanizma sağlamak için, CHARACTER STRING delik tipi tanıtıldı. Bu daha sonraki bir bölümde açıklanmaktadır.

Karakter repertuarlarının kullanıcının ihtiyaçlarına göre daha kesin bir şekilde uyarlanması sağlamak ve herhangi bir basılı ASN için mevcut gliflere (belki de sınırlı kümeye) bağlı olmayan karakter dizileri için kesin ve belirsiz olmayan bir deðer gösterimi sağlamak için zaman içinde mekanizmalar da eklendi. .1 spesifikasyonu veya herhangi bir makine tarafından okunabilir ASN.1 spesifikasyonu için mevcut olan karakter repertuarında (belki sadece ASCII gibi).

Nihai sonuç, karakter dizisi alanları için belki de kafa karıştırıcı, ancak geniş kapsamlı ve güncel bir türler kümesidir.

Bölüm 3 alt tipleme

(Veya: Veri türlerinizi sıklaş tirin!)

Özet: Bu bölümde, bir tür için izin vermek istediği iniz değil er kümesinin (altkümesinin) kesin tanımına olanak sağlayan ASN.1 alt türü gösterimi açıklanmaktadır. Örneğin şunları belirtebilirsiniz:

bir tamsayıının aralığı i;

bir dizinin minimum ve/veya maksimum uzunluğu u;

bir karakter kümesinden istenen kesin karakterler;

SEQUENCE OF veya SET OF'deki minimum ve/veya maksimum yineleme sayısı.

Tam notasyonun hatırı sayılır bir gücü ve esnekliği vardır, ancak yukarıdaki örnekler en sık rastlanan örneklerdir.

1. Giriş

ASN.1 "alt tip gösterimi" çok güçlündür ve ASN.1'i harika yapan şunlardan biri olduğuunu söylemek güzel olur!

Bununla birlikte, kullanımının daha basit örnekleri (dizilerdeki uzunluk sınırları, tamsayılar üzerindeki aralıklar) yaygın olsa da ve bunları kullanabildiğiniz yerlerde kullanmanız önemlidir, bu notasyonun diğeri bazı özellikleri daha az sıkılıkla görülürler ve belki de daha az önemlidirler.

Türlerinizi tam olarak ihtiyacınız olan kesin değil erlere göre özelmiş tirin - hattaki bit sayısını genellikle iki kattan fazla azaltabilir (PER kullanımdaysa) ve uygulayıcılara bellek ayırma kararları için net rehberlik sağlar, kullanılacak tamsayı boyutu gibi.

Ayrıca (okumaya devam etmeden - veya atlamanadan önce!), alt tip gösterimindeki esnekliği in 1994'te önemli ölçüde gelişti. Bu nedenle aşağıdaki verilen örneklerden bazılarının 1994 öncesi yasal olmayacağıını unutmayın. Gerçek ASN.1 spesifikasyonunu kontrol edin!

Şekil 13'te alt tipleme ile çok kısaca tanıştık, burada (ayırt edici değil erleri atlayarak) aşağıdaki idakilerin bir sıra elemanına sahiptik:

bildirilen gün sayısı INTEGER (1..56)

tamsayı alanının aralığı 1 ile 56 arasındaki değil erlerle sınırlanır.

1994 öncesi ASN.1'de, yuvarlak parantez içindeki bu notasyonun, orijinal veya üst türdeki dē erlerin bir alt kümelerinden (dolayısıyla alt tipleme) oluş an yeni bir tür ürettī i kabul ediliyordu. 1994 sonrası, bakış açısı , tamsayıyı 1 ila 56 aralı̄ ina olacak ş ekilde sınırladı̄ ı̄mizden daha fazla olma ē ilimindedir.

Neden fark var? Pekala, 1994 sonrası bir dizi baş ka kısıtlama mekanizması tanıtı̄ldı (kısıtlanan türü izleyen bir çift yuvarlak parantez içinde de), ancak daha da ȫnemli, bir kısıtlama olarak gösterime odaklanmak ş u soruyu gündeme getiriyor "Peki ya gelen olursam? kısıtlamayı ihlal eden malzeme?". Kısıtlamalarla ilgili genel konu (ve ilgili istisna iş leme) bu b̄dümün 7. B̄dümüne bırakılmış t̄r, ancak burada ilk olarak 1986'da ASN.1'de tanıtı̄lan basit alt tip gösterimini tam olarak tartı̄ acā iz.

Alt tipleme ASN.1'e eklendī inde, Temel Kodlama Kuralları dē iş tirilmedi. TLV tabanlıydīlar ve ȫnēg in "L" kısmını ortadan kaldırı̄mak iç in alt tip bilgisini kullanmak, kodlamanın yapısını bozabilirdi. Böylece 1994'e kadar, alt tipleme uygulaması yalnızca uygulama kodunun yazarına yardımcı oldu - kodlamayı veya hattaki bit sayısını etkilemedi.

Paketlenmiş Kodlama Kurallarının (PER) kullanıma sunulmasıyla birlikte, kodlama alt tiplemeden etkilenir (ȫzellikle tamsayılar iç in). PER'den maksimum faydayı elde etmek iç in, uygulama tasarımcıları makul olarak yapabildikleri her zaman aralık bilgilerini (ve dizelerdeki uzunluk kısıtlamalarını ve set-of ve sequence-of üzerindeki yineleme kısıtlamalarını) içermelidir.

PER'de "PER-görünür kısıtlamalar" kavramı vardır - kodlamayı etkileyen ş eyler. Tüm alt tipleme yapıları PER-görünür dē ildir (ve ȫzellikle iç alt tipleme - āş āya bakın - iyi nedenlerle hiç bir zaman PER-görünür dē ildir). PER-görünür olmayan herhangi bir alt tipleme gösterimini göz ardı edebileceğ inizi - ȫğ renmeyin, kullanmayın - ȫnermek cazip gelebilir (yine ş ekil 999'a bakın!), ancak bu yeni bir denetim olarak kötü bir tavsiye olur. PER, bir āş amada daha karmaş ik kısıtlamaları hesaba katacak ş ekilde tanımlanabilir. Doğ ru tavsiye ş udur: "Uygulamalarınızın bir türdeki dē erlerin yalnızca bir alt kümelerini kullanma niyetindeyseniz, o zaman bunu yalnızca yorum olarak dē il, ASN.1 alt tür gösterimini kullanarak resmi olarak ifade etmeye çalış in."

2 Temel kavramlar ve küme aritmetiği

Alt tip gösteriminin farklı biçimlerine bakmadan önce, alt tip gösteriminin (etiketleme gibi - bir sonraki b̄düm bakın) resmi olarak yeni bir tip ürettī ini kabul etmek ȫnemlidir. Dolayısıyla, ASN.1'in tip gösterimi gerektirdī i izin verdiği i her yerde, bunun yerine ş unu yazabilirsiniz:

tip gösterimi alt tip gösterimi

Alt tür notasyonu bir türde (ana tür) uygulanır ve üst türdeki soyut dē erler kümelerinin bir alt kümelerini içeren yeni bir tür üretir.

"alt tip gösterimi", "tip gösterimi" tarafından verilen ebeveyn tipi iç in izin verilen notasyonlardan biri olmak zorunda olmasına rağmen. "alt tip gösterimi" her zaman yuvarlak parantezlerle baş lar ve biter.

Bu fikir ȫzyinelemeli olarak uygulanabilir. Böylece, ȫnēg in ş unları yazabilirsiniz:

My-string1 ::= PrintableString (SIZE (1..10)) (FROM ("A" .. "Z"))

Bu, önce 1 ile 10 karakter arasındaki dizelerle sınırlı PrintableString olan bir türü tanımlar, ardından bunu yalnızca "A" ila "Z" karakterlerini içeren dizelerle sınırlandırır.

Set aritmetiği ini kullanarak aynı iş i tek seferde yapabilen baş ka bir alt tip gösterimi vardır. Yazabiliriz:

```
My-string2 ::= PrintableString ( SIZE (1..10) INTERSECTION
                                FROM ("A" .. "Z") )
```

Bu notasyonda, "SIZE (1..10)", 1 ile 10 (dahil) arasında uzunluklara sahip PrintableString'in tüm değer erlerinin kümelerini seçer. "FROM ("A" .. "Z")", yalnızca "A" ila "Z" karakterlerini içeren PrintableString'in tüm değer erlerini seçer. Bu kümelerin matematiksel kesişimi, yukarıda My-String1 tarafından belirtilenle tamamen aynı PrintableString değer erleri kümelerini verir.

Genel olarak, yuvarlak parantez içindeki yapı, "KESİ Ş ME", "Bİ RLİ K", "EXCEPT" sözcükleriyle ayrılmış ve "normal" önceliğe sahip bir dizi terim iç erir (KESİ Ş İ M en sıkı, HARİ Çen az sıkı bağlar). Her terim, üst türün (yukarıdaki durumda PrintableString) bir dizi değer erini resmi olarak tanımlar ve sonuçta ortaya çıkan yeni türde hangi değer erlerin olduğunu belirlemek için normal kümeleri aritmetiği uygulanır.

(Ayrıca, kümeleri aritmetiği, değer eri olmayan bir türün tanımlanmasıyla sonuçlanırsa, bu geçersiz ASN.1'dir!).

Ayrıca, öncelik konusunda okuyucunun kafa karışıklığıını önlemek için

TAM SAYI (A HARİ Ç B HARİ Ç C)

izin verilmeyen şu şeyle yazılmalıdır:

TAM SAYI (A HARİ Ç B) HARİ Ç C)

veya

TAM SAYI (A HARİ Ç (B HARİ Ç C))

hangisi amaçlandıysa. Bİ RLEŞ İ M ve KESİ Ş İ M için eşdeğer bir kısıtlama yoktur, çünkü yukarıdaki "EXCEPT"lerin her ikisi de "Bİ RLİ K" (veya "KESİ Ş ME") ile değer işbirliği, iki farklı köşeli parantez modeli aynı sonuç kümelerini üretir.

yazmak da mümkün

TAM SAYI ((1..20) HARİ Ç TÜMÜ)

açık anlamıyla. ("TÜMÜ"nın ardından yalnızca "EXCEPT" gelebilir).

Daha karmaşık bir örnek (okuyucu için alıştırma - bu tür bir yapının faydalı olacağının gerçek dünyadan bir örnek bulun!) şöyledir:

```
My-string3 ::= PrintableString ( SIZE (1..10) KESİ Ş İ M
                                FROM ("A" .. "Z") )

                                Bİ RLİ ğ ("evet" Bİ RLİ ğ "hayır" Bİ RLİ ğ olabilir)
                                HARİ Ç
                                "A" Bİ RLİ ğ B )
```

Bence bunun ne anlamına geldiğini söylebilirsin, ama değer ilse, aşağındaki okuduğunda ona geri dön! Yukarıdaki "belki" ve "B" etrafında tırnak işareti bulunmamasının bir yazım hatası olmadığına dikkat edin! "belki"nin PrintableString tipi bir değer eri için bir değer er-referans-adı olduğu varsayılmaktır (bu modülde başka bir yere atanır) ve B'nin PrintableString'in bir alt tipi eri için bir tip-referans-adı olduğu varsayılmaktır (ayrıca bu modülde başka bir yere atanır) modülü! Bir değer eri için açık değer er notasyonuna izin verilen her yerde, bir değer er referans adına da izin verildiğiini unutmayın (üst türün bir değer erine atıfta bulunması koşuluyla) ve (belki daha az açık bir şekilde) kümeleri aritmetiği, içi bir alt kümeyi içeren gerekli olduğu her yerde, bir type-referans-adı kullanılabilir (ana türün bir alt türüne atıfta bulunması koşuluyla).

Uyarı-uyarı-okuyucu (!), belirli bir küme aritmetiği içinde yasal olması için bir değ er referans-adı veya tür-referans-adı tanımlama ş ekleyle ilgili kesin kuralların ne olduğunu sormaya baş liyor olabilir. vali (ebeveyn tipi). Bu, Sectin IV'teki ASN.1 Anlamsal Modeli'nin açıklamasında ele alınmış tır, ancak ş imdilik, bir insan okuyucu için anlamlı olacaksa, neredeyse kesinlikle yasal olduğunu not etmek yeterlidir!

Alt tip gösterimi kullanılarak tanımlanan bir tür için değ er gösteriminin o gösterimden etkilenmediğini unutmamın - üst tür için normal değ er gösterimi olarak kalır.

Son bir genel yorum: "KESİ Ş İ M" kelimesi "ş apka" sembolü ile değ iş tirilebilir: "^" ve "Bİ RLİ K" kelimesi "dikey çubuk" sembolü ile değ iş tirilebilir: "|", ancak karış tırmamanız önerilir ve herhangi bir uygulama ätzelliğinde eş iş in! Benim için ASN.1 spesifikasyonları zaten oldukça ayrıntılı olma eğ ilimindedir - uzunca isimler yaygındır - bu yüzden kelimeleri tercih ederim!

Öyleyse, yuvarlak parantez içinde bağımsız alt tip kısıtlamaları olarak veya muhtemelen karmaş ik bir aritmetik küme ifadesinin parçası olarak kullanabileceğimiz temel terimler nelerdir ve bunlar hangi değ er kümelerini tanımlar?

Aşağıda her olasılığ ele alıyoruz. Bazı durumlarda yan tümcenin başlığıında "alt tipleme" veya "alt tip" bulunduğu una ve diğer durumlarda "kısıtlama" kelimesinin kullanıldığına dikkat edin. Bu, ASN.1 spesifikasyonunda kullanılan terimleri yansıtır ve çoğu amaç için iki kelimenin birbirinin yerine kullanılabilir olduğunu noktasını güçlendirir.

3 Tek değ er alt tipi

Bu, herhangi bir üst türü uygulanabilir. (ASN.1'de tanımlayabileceğimiz her tür için bir değ er gösterimi olduğunu unutmamın). Sadece izin verilen değ eri listeliyoruz! Normalde buna dikey çubuk veya UNION kullanımı eşlik eder. Yani:

```
Evet ::= PrintableString ("Evet")
ve
Evet-Hayır ::= PrintableString ("Evet" | "Hayır")
```

tek değ er alt tipi kullanan örneklerdir. Tek değ er alt tipinin her kullanımıyla tanımlanan değ erler kümesi, yalnızca değ er notasyonuyla tanımlanan tek değ erdir.

4 Değ er aralığı alt tipi

Bu yalnızca doğrudan tamsayı ve gerçek türlerde uygulanabilir, ancak "FROM" kelimesini izleyen aynı yapı, bazı karakter dizisi türlerinde izin verilen karakter kümesini kısıtlamak için kullanılır (aşağıdaki "izin verilen alfabe" bölümüne bakın).

Tamsayı değ erlerinin aralığıını belirtmek için sıkılıkla değ er aralığı alt tipi uygulanır.

Bir değ er aralığıının üç noktaları verilir ve gösterimle tanımlanan değ erler kümesi, tam olarak bir üç noktadan diğerine (bitiş noktası dahil) olurlardır. Bu, daha önce karşılaşılmıştı ve genellikle tamsayı değ erlerini kısıtladığı görülen notasyondur:

Bildirilen gün sayısı ::= TAM SAYI (1..56)

Her zaman olduğu gibi, bu kısıtlamaların kesişmeleri ve birleşimleri mümkün değildir, ancak nadiren görülür.

5 İzin verilen alfabe kısıtlamaları

Bu, yalnızca karakter dizisi türlerine uygulanabilen bir kısıtlamadır ("CHARACTER STRING" türü hariç).

En basit haliyle bu kısıtlama, "FROM" kelimesi ve ardından bir dizi izin verilen karakter içeren bir karakter dizisidir. Böylece:

Bazı kodlama kuralları (hizalanmamış PER), bir dizgede kaç farklı karaktere izin verdiği inize bağlı olarak karakter başına minimum bit sayısını kullanır, bu nedenle alfabe kısıtlamaları uygulamak satırındaki bitleri koruyabilir.

Ünlü dizisi1 ::= PrintableString (FROM ("AEIOU")
veya
Sesli harf dizisi2 ::= PrintableString (FROM ("AEIOU")

olası örnekler olacaktır. "FROM"dan sonraki açılış parantezi gereksiz görünebilir ve hantal görünebilir, ancak sÖZdizimi tanımı FROM'dan sonra tamamen genel bir kısıtlamaya izin verir, bu nedenle

Ünlü dizisi3 ::= PrintableString (FROM ("AEIOU")

Bİ RLİ K
"aeiou("))

da izin verilir

"FROM" ifadesini izleyen kısıtlamanın, bir dizi dize değil eri üretmek için doğrudan üst türde uygulanabilecek bir kısıtlama olması gereklidir (buna, dize değil erlerinin tanımlayıcı kümesi adını verin (yalnızca bu kitapta kullanılan bir terim). "FROM", ("FROM" tarafından seçilen dize değil erlerinin alt kümesinde), tanımlayıcı kümedeki dize değil erlerinden herhangi birinde (yalnızca) herhangi bir karakter içeren üst türdeki tüm dizelere izin vermektedir.

Aliş tırma: Bu tanımı dikkatlice okuyun, ardından "Sesli harfler dizisi2 ve Sesli harfler dizisi3 eş değ er tanımlar mı?" sorusunu yanıtlayın. Cevabını aldığında okumaya devam et!

Akıl yürütüyoruz. "String-of-Vowels2" ile önce iki PrintableString deð eri kümesi tanımlarız. Biri sadece büyük ünlülerden oluşan an dizileri, dið eri ise sadece küçük ünlülerden oluşan an dizileri ve bu iki kümenin birleşimi alıyoruz. Böylece nihai sonuç, yalnızca ünlüleri içeren dizelere izin verir, ancak her dize tamamen büyük veya tamamen küçük harf olmalıdır.

"String-of-Vowels3" ile önce, her biri beş karakterden oluşan yalnızca iki dizisi değilinden oluşan bir kümeye turuyoruz: "AEIOU" ve "aeiou". Daha sonra bu kümeye "FROM"u uygularız, sonuç olarak büyük ve küçük sesli harflerin rasgele kombinasyonlarından oluşan dizelere izin veririz, bu nedenle "Sesli Sesler2 Dizisi" ve "Ünlüler Dizisi3" aynı değilildir.

Yukarıda, FROM'dan sonraki kısıtlamada yalnızca tek deg erli alt tür notasyonu kullanılmıştır, ancak üst türde uygulanabilen herhangi bir alt tür gösterimi kullanılabilir. Özellikle, deg er aralığı alt tipine, FROM'dan sonraki kısıtlamada kullanıldığıında belirli karakter dizisi türlerine uygulama için açıkça izin verilir ve yalnızca tek bir karakter içeren dizelerle sınırlanır.

Böylece şunları yazabiliriz:

```
Hex-digit-String ::= PrintableString (FROM ("0".."9" Bİ RLİ Č "A".."Z"
                                         Bİ RLİ Č "a".."z"))
```

ilk olarak rakamları ve harfleri (62 dize değil eri) kullanarak tüm tek karakter dizilerinin kümesini oluşturur ve ardından yalnızca bu 62 karakteri içeren tüm PrintableString değil erlerinin kümesini oluşturmak için FROM'u bu kümeye uygular.

Dağılık aralığı kısıtlaması, karakter sıralamasının iyi tanımlandığı (BMPString, IA5String, NumericString, PrintableString, VisibleString, UniversalString, UTF8String) karakter dizisi türleri için bu şekilde kullanılabilir, ancak şuna dayalı karakter dizisi türleri için kullanılamaz: sıralamanın tanımlanmasının kolay olmadığı (Uluslararası Kodlanmış Karakter Setleri Kaydı (GeneralString, GraphicString, TeletexString veya ViedotexString)).

6 Boyut kısıtlamaları

Bir boyut kısıtlaması, izin verilen bir alfabe kısıtlamasına benzer bir yapıya sahiptir. Negatif olmayan bir tamsayıya uygulanabilen herhangi bir kısıtlama belirtimi (parantez içinde) tarafından takip edilen "SIZE" kelimesinden oluşan ur. (Yalnızca) bir bit dizisine, bir sekizli dizisine, bir karakter dizisine (daha sonraki bir bölümde tanıtılan "CHARACTER STRING" türü dahil) veya bir "SEQUENCE OF" veya "SET OF" yapısına uygulanabilir. Etkisi, "SIZE" kelimesini izleyen kısıtlama tarafından seçilen (negatif olmayan tamsayılarından) kümedeki tamsayı değil erlerinden birine eşit sayıda karakter veya yineleme içeren üst tür değil erlerini seçmektedir.

Boyut kısıtlamaları, izin verilen dize uzunluklarını ve yineleme sayılarını belirtmek için değil er aralıklarını kullanır. Kullanımları yine hattaki bitleri kurtarabilir

"SEQUENCE OF Xyz" ve "SET OF Xyz" durumunda, kısıtlama tip tanımından sonra veya "OF"tan hemen önce görünebilir. Bu, aşağıda gibi durumlarda kısıtlamaların hem yineleme sayılarına hem de yinelenen türde uygulanmasına izin vermek için gereklidir.

```
PrintableString Dİ Zİ Sİ Nİ N Dİ Zİ Sİ (BOYUT (10))
```

Bu sözdizimi, PrintableString'i tam olarak on karakterle sınırlar ve yineleme sayılarını sınırlamak için kullanılamaz. Bunları kısıtlamak için kullanırsınız

```
PrintableString Dİ Zİ Sİ Nİ N Dİ Zİ Sİ Nİ N Dİ Zİ Sİ (BOYUT (10))
veya
PrintableString'i N Dİ Zİ Dİ Zİ Sİ (BOYUT (10))
```

ASN.1 bir kez daha bu alanda tamamen geneldir - OF'den önce görünen kısıtlama notasyonu, 1994 öncesi spesifikasyonlar daha kısıtlayıcı olmasına rağmen, birleşimleri ve kesişmeleri vb. içerebilen genel bir kısıtlamadır.

Uygulamada, "SIZE" kelimesini takip eden kısıtlama hemen hemen her zaman tek bir değil er kısıtlaması veya bir değil er aralığı kısıtlamasıdır, örneğin:

```
Dİ Zİ (BOYUT (1..100)) Dİ Zİ (BOYUT (20)) OF PrintableString (SIZE (0..15))
```

bu, yirmi sütunlu bir ila yüz satırlık bir tabloyu temsil edebilir, her hücre boş veya en fazla 15 karakter uzunluğunda bir PrintableString içерir.

Winco protokolümüze geri dönersek ve Kısım I Bölüm 4'teki § 22'ye atıfta bulunarak, başlangıçta "satış verilerini" sınırsız sayıda "Rapor kalemii" olarak tanımladık. Bir uygulayıcının sınırsız sayıda şeysi desteklemesi genellikle oldukça zordur, ancak artık kolayca kullanılabilen artan bellek boyutları ve büyük kapasiteli diskler ile "etkin bir şeysi sınırsız" (burada kastettiğimiz budur) uygulaması mümkündür. Hem BER hem de PER kodlamaları, nesnelerin sınırsız sayıda (ve boyutunun) etkili bir şeysi etkile transferini destekleyecektir, ancak sayıları ve tamsayı değerlerleri, örneğin tutulabilen değerlerle sınırlamak mümkünse, PER ile kodlama daha verimli olacaktır. İki veya dört sekizli.

"Satış verileri" satırını aşağı idakilerle değil iş tirmek yaygın bir uygulama olacaktır:

satış verileri Dİ Zİ (BOYUT (1..sales-ub)) Rapor Öğesinin

"Sales-ub" değil er referansının bir tamsayı değil er referansı olması gereklidir ve § 22 bağlamında kullanılabilir hale getirmek için İ-HRACAT/İ-THALAT kullanılarak bu tür tüm sınırları toplayan bir modüle atanabilir. olmak:

sales-ub INTEGER ::= 10000

Hem FROM hem de SIZE kullanarak son bir örnek düşündürün:

PrintableString (SIZE (1..10) KESİŞ İ M FROM ("A".."Z"))

Okumaya devam etmeden önce bunun ne anlamına geldiğini anlamak için bir dakikanzı ayırin.

Önce birden on karaktere kadar tüm dizilerin (sonlu) kümesini seçiyoruz ve bunu yalnızca "A" ile "Z" karakterlerinden oluşan tüm dizilerin (sonsuz kümesi) ile kesiştiyoruz. Nihai sonuç, yalnızca "A" ile "Z" harflerini içeren bir ila on karakterlik dizeler kümesidir. Tam olarak aynı sonucun aşağıda idakilerden herhangi biri tarafından elde edildiğini unutmayın:

```

PrintableString (SIZE (1..10)) (FROM ("A".."Z"))
veya
PrintableString (FROM ("A".."Z")) (BOYUT (1..10))
veya
İlk (FROM ("A".."Z"))
veya
Saniye (BOYUT (1..10))
veya
PrintableString (Birinci KESİŞ Şİ M İkinci )
veya
PrintableString (Birinci) (İkinci)

nerede

```

İlk ::= PrintableString (SIZE (1..10))

ve

İkinci ::= PrintableString (FROM ("A".."Z"))

7 İ çerilen alt tip kısıtlamalar

Bu notasyonla yukarıda birkaç kez gayri resmi olarak karşı ılaştık. Bu kısıtlama biçimi, dahil edilecek değil erler kümelerini tanımlamak için bir tür referans adı (ana türün bir alt türü için) sağladığımız yerdir. Bu, kesişme noktalarını kullanan daha karmaşık bir kısıtlama içinde olmadıkça veya durumlarda olduğum gibi tekrar tekrar kısıtlamalar uygulamadıkça, normalde yararlı olmaz.

PrintableString (Birinci KESİ Ş İ M İ kinci)
ve
PrintableString (Birinci) (İ kinci)

üstünde.

1994 öncesinde, bir tür referans adının bir kısıtlamada bu şekilde kullanılmasının, adın önünde "INCLUDES" kelimesinin olmasını gerektirdiği ini ve buna hala izin verildiği ini unutmayın (örneğin):

PrintableString (Birinci KESİ Ş İ M DAHİ LDİ R, İ kinci KESİ M DAHİ LDİ R)
veya
PrintableString (Birinci DAHİ LDİ R, İ kinci DAHİ L HARİ Q)

ancak bunlar pek iyi okunmuyor ve "DAHİ LDİ R" kelimesini atlamak en iyisidir.

8 İ ç Alt Tipleme

8.1 Giriş

İç alt tipleme önemli ve yeterince kullanılmayan bir araçtır. Uygulama tasarımcılarının, iç alt tipleme (OSS aracı tarafından desteklenen) kullanılarak daha mantıklı bir şekilde yazılmış olabilecek belirtimleri üretmek için kendilerine ait (ASN.1 araçları tarafından desteklenmeyen) yeni bir meta-notasyon icat ettikleri sıkılıkla görülen bir durumdur. Bu sadece okuyucunun ad hoc gösterime alışmasını gerektirmez, aynı zamanda ASN.1 araçlarının kullanımından önce gerekli olan spesifikasyonun bir tür geçici ön işlemesi ile uygulayıcının işini gereksiz yere zorlaşdırabilir.

İç alt tiplendirme, bir protokolün alt kümelerinin veya uyumluluk sınıflarının belirtimine kesinlik kazandırmaya yardımcı olabilecek önemli bir mekanizmadır.

Bunun cehaletten kaynaklanması muhtemeldir, belki de muhtemeldir. İç alt tipleme, ASN.1 belirtiminde "sadece başka bir alt tip gösterimi" olarak konumlandırılmasıyla ortaya çıkmayan genel bir öneme sahiptir.

Şimdiye kadar açıklanan alt tip notasyonları, uygulama tasarımcılarının temel tipler için protokollerinde izin verilen değil erlerin aralığıını açıka belirtmeleri için çok güçlü bir araç sağlar, ancak başka bir gereklilik daha vardır: bazı tasarımcıların bir dizi farklı altküme tanımlama gereksinimi vardır. farklı amaçlara uygun protokol, farklı söze "uygunluk sınıfları".

En basit durumda, her mesajın Bölüm 1 Bölüm 3 ş. ekil 21'deki "Wineco-Protocol" gibi tanımlanmış bir ASN.1 tipi olduğum bir "Tam Sınıf" protokolümüz var, ancak aynı zamanda bir "Temel Sınıf" da tanımlamak istiyoruz. Sekansların bazı istekleri bağılılığı erlerinin olduğum "Sınıf" protokolü

atlanması gereklidir, diğer erlerinin her zaman dahil edilmesi gereklidir, bazı seçenekler kısıtlanır ve bazı yinelemeler ve/veya tamsayı değil erleri kısıtlanmış değil erlere sahiptir.

"Wineco-Protocol" türünün soyut değil erler kümesini göz önünde bulundurursanız, yukarıda açıklanan tüm kısıtlamaların (bazı isteği ve bağlılığı erlerin bulunmasını ve diğer erlerin bulunmamasını gerektirme dahil) yalnızca belirli bir alt kümenin seçimi olduğuunu fark edeceksiniz. "Wineco-Protocol" değil erleri - başka bir deyiş ile alt tipleme!

Bununla birlikte, iki ek gereksinim vardır:

İlk olarak, metin tekrarı olmadan (ve dolayısıyla hata kapsamı) her iki uygunluk sınıfının da tanımlanması mümkün olmalıdır.

İkinci olarak (tüm uygulamalar için olmasa da bazıları için) hem "Temel Sınıf" protokolünde hem de "Tam Sınıf" protokolünde bulunan değil erlerin kodlanması her iki protokolde de aynı olmalıdır.

İkinci gereksinim, "Tam sınıf" ve "Temel Sınıf" uygulamaları arasında kolay birlikte çalışmayı sağlamak içindir.

Bu alan ile daha sonra açıklanan "geniş letilebilirlik" konuları arasında bir ilişkiye vardır, ancak farklılıklar vardır. "Geniş letilebilirlik", ilk sistemler konuşlandırdığıında maksimum işlevsellik bilinmediği, zaman içinde (farklı sürümler) spesifikasyonlardaki farklılıklara atıfta bulunurken, burada, maksimum işlevsellik baştan bilindiği, biraz daha basit bir sisteme izin veren uygulamalardaki farklılıklara ilgileniyoruz. yaklaşımak.

Metnin yinelenmesi olmadan tüm uyumluluk sınıflarını tanımlamak için aşağıdaki idakiler gereklidir:

(ilk olarak) "Wineco-Protocol" tipini maksimum işlevsellik ile tanımlayın ve ona bir tip referans adı verin; sonra

bu tür referans adını kullanmak ve ona "Temel Sipariş Sınıfı" ve "Temel Satış Verileri Sınıfı" (veya diğer uygunluk sınıfları) oluşтурan kısıtlamaları uygulamak. İkincisi, tip referans adının ardından parantez içinde alt tip kısıtlama notasyonu yerleştirilerek elde edilir. Böylece sahibiz:

Temel Sipariş Sınıfı ::= Wineco Protokolü (.....)

(.....), "Wineco Protokolü"nün iç bileşenlerini sınırladığıımız iç alt tip kısıtlamasıdır.

Hem BER hem de PER'de, bu kısıtlamaların uygulanmasının seçilen alt kümedeki değil erlerin kodlanması etkilemediği ini - tam olarak "Tam Sınıf" protokolündeki gibi kodlandıgı ini not etmek önemlidir. Buna karşılık, kısıtlamalar (bazı seçimlerin kaldırılması veya isteği ve bağlılığı alanların zorunlu olarak mevcut veya mevcut olmaması gibi), ASN.1 metnini değil iş tiren geçici bir meta-dil tarafından (veya Temel Sınıf protokollerini açıkça yazarak) belirtmiş se, Temel Sınıftaki değil erlerin kodlanması, Tam Sınıftaki karşılık gelen değil erlerden farklı olacaktır ve ayrıca, belirsiz olmayan etiketlerdeki kuralların (aşağıya bakınız) üretilen varyantların hiç biriyle ihlal edilmemesine dikkat edilmesi gerekecektir. .

Bu, dahili alt tiplemenin geçici bir "işlemci öncesi" gösterime tercih edilmesinin bir başka nedenidir - bu, kodlamaların ve etiketlemelerin tüm sınıflarda aynı olmasını sağlar.

8.2 Wineco-Protokolünün Alt Kümesi

Yine, önce bir örnekle devam edelim. $\$$ ekil II-3'ü ele alalım. Bu, $\$$ ekil 21'deki en üst düzey tanımı tekrarlıyor, ancak $\$$ imdi sürüm 2'ye geçtiğ (MS 2002'de üretildi) ve elektronik paranın içeriği ini kasamıza yüklememizi sağ layacak ek bir üst düzey seçenek imiz var. .(Bunun bir uzanti iş aretiçisini takip etmesi, iç alt tip notasyonunda hiçbir fark yaratmaz ve $\$$ u an için uzanti iş aretiçisi çizgisinin varlığı tamamen göz ardı edilmelidir.) Ayrıca Wineco-Protokolünün tam tanımını içeren Ek 2'ye bakın.

Wineco-Protokol ::= SE $\ddot{\text{c}}$ M

```

{sipariş [BAŞ VURU 1] Stok Sipariş i,
satış           [UYGULAMA 2] Satış beyannamesi verileri,
...           ! PrintableString : "45.7 maddesine bakın",
e-nakit iadesi -- Sürüm 2'de eklandı --
[BAŞ VURU 3] Nakit yükleme}

Temel Sipariş Sınıfı ::= Wineco Protokolü
(Bİ LEŞ ENLER İ LE
{sipariş (Temel-Sipariş ) MEVCUT,
satış           MEVCUT OLMAYAN }  )

Temel Satış Sınıfı ::= Wineco Protokolü
(Bİ LEŞ ENLER İ LE {sipariş
satış ları (Temel İ ade)           MEVCUT OLMAYAN ,
MEVCUT } )
```

$\$$ ekil II-3: Sürüm 2'de kısıtlama

Burada, versiyon 1 alternatiflerinden birini her zaman mevcut ve diğ erini her zaman yok yaparak dış seviye seçimiini kısıtladık. Ayrıca, mevcut alternatif dahil edilen alt tip kısıtlamalarını (yukarıya bakın) "Temel Sıra" ve "Temel Getiri" uygulayarak onu daha da kısıtlıyoruz. Kısaca "Temel-Sipariş" ve "Temel-Dönüş" tiplerini tanımlayacağ iz.

Burada 1. versiyonda mevcut olan her alternatifin listelediği imize, MEVCUT veya MEVCUT olmadığı ina dikkat edin. Buna "tam özellik" denir. "Tam özellik" olarak adlandırılmasına rağmen, aslında her alternatifin listelemek gerekli değil. Listelenmeyen herhangi bir $\$$ ey için OLMADIĞ ima edilir, bu nedenle "Temel Satış Sınıfı" tanımı $\$$ una eş dē erdir:

```

Temel Satış Sınıfı ::= Wineco Protokolü
(Bİ LEŞ ENLER İ LE {sipariş
satış           OLMAYAN ,
(Temel-İ ade) MEVCUT, e-nakit-iade
MEVCUT OLMAYAN } )
```

ve için

```

Temel Satış Sınıfı ::= Wineco Protokolü
(Bİ LEŞ ENLER İ LE
{satis (Temel İ ade) MEVCUT} )
```

(Listelenen en az bir alternatif olması gerekī in ve tam olarak "tam belirtimde" MEVCUT olarak listelenen bir tane olması gerekī in unutmayın.)

Kısıtlamanın "...," ile baş ladī i bir "kismi belirtim" gösterimi de vardır. Bu, Basic-Sales-Class2 protokolünün hem "satış ları" hem de

"e-nakit iade" mesajları. "Kısmi belirtim", "tam belirtimden" yalnızca listelenmeyen tüm alternatiflerin kısıtlanmamış olası seçenekler olarak kalması ve bu kelimelerden hiç biri mevcut değil ilse (ancak başka şart ekillerde kısıtlanmış olabilir) listelenen herhangi birinin ne MEVCUT ne de MEVCUT olması gerekmemesi bakımından farklıdır.). Bu nedenle, şekil II-4'te, ya "satışlar" ("Temel İade" ile sınırlılmıştır) veya "e-nakit iadesi" mesajları (sınırlılmamış) mevcuttur ve uygulanması gereklidir, ancak "sipariş" mesajları asla gönderilebilir veya alınabilir ve uygulanması gerekmeyez.

Basic-Sales-Class2 ::= Wineco-Protocol (Bİ LEŞ ENLER İ LE { ... sipariş
YOK, satış (Temel-İade) })

Şekil II-4: Yalnızca satış alternatifini kısıtlama

"Temel Getiri"nin ne olduğunuunu açıklayamaya devam edelim. Bu, şekil II-5'te kısıtlı bir "Satışların Getirisini" olarak gösterilmişdir. ASN.1'de her zamanki gibi, şekil II-5'te kısıtlamayı "satır içi" koyabilir ve "Temel Rapor Öğesi" tip referans adını kullanmamayı seçebiliriz. Bu sadece bir tarz meselesi. Şekil II-6, aynı tanımı ancak "in-line" kısıtlamasıyla göstermektedir (Şekil II-6'daki yorumları tekrarlamadık). Daha derli toplu olmakla birlikte, şekil II-6'daki "Rapor-Öğesi" üzerindeki iç kısıtlamaya ilişkili kılendirecek bir adın olmamasının, bu stil şekil II-5'in biraz daha ayrıntılı stilinden daha az okunabilir hale getirdiği tartışılabilir. Bununla birlikte, her iki notasyon da tamamen aynı semantik ifade eder.

Temel İade ::= Satış İadesi
(Bİ LEŞ ENLER İ LE { ... rapor
edilen gün sayısı (7) --
raporlar haftalık olmalıdır --, gecikme nedeni YOK,
ek bilgi YOK, satış verileri (BOYUT (1..temel)-satış -ub))

KAVŞAK
(Bİ LEŞ EN İ LE
(Temel rapor-Öğesi))

Temel-rapor-Öğesi ::= Rapor-Öğesi
(Bİ LEŞ ENLER İ LE { ..., Öğe
açıklaması YOK -- Rapor
Öğe esinin 2. Sürümü, -- yeni stoklanan Öğeler
için bile Öğe açıklamasının -- Öğe eler --> çıkarılmasına izin verir)

Şekil II-5: "Satış Getirisini" Kısıtlama

Ş ekil II-5, "satış verileri" hakkında biraz açıklama gerektiriyor. Burada "Rapor-ğ esi" sayısını daha da sınırlıyor ve ayrıca her "Rapor-ğ esini" "Temel-rapor-ğ esi" alt kümesiyle sınırlıyoruz.

Bir Dİ Zİ veya KÜME iç in iç alt tiplene uyguladı ımda, kısıtlamayı "Bİ LEŞ ENLER İ LE" ile baş lattı ıma ve ardından, bileş en adını izleyen k ı eli parantezler iç inde listelenen her bileş en üzerindeki kısıtlamalarla (varsayı) eş leş tirilmiş süslü parantezlere sahip olduğ umuza dikkat edin. (Bunu Ş ekil II-5'teki "Rapor-ğ esi" (bir Dİ Zİ olan) kısıtlamasıyla görebilirsiniz). Ş imdi, dış Dİ Zİ 'nin bileş enlerinden birinin bir Dİ Zİ olduğ unu varsayılmı.

```

Teme ı ade ::= Satış ı adesi
(Bİ LEŞ ENLER İ LE {... rapor
edilen gün sayısı (7), gecikme
nedeni YOK, ek bilgi YOK, satış verileri (SIZE (1..basic-
sales-ub)

KESİ ş İ M (Bİ LEŞ EN
İ LE (Bİ LEŞ ENLER İ LE
{..., item-description MEVCUT } ) ) )

```

Ş ekil II-6: "in-line" kısıtlamasının uygulanması

OF veya SET OF ise, doğrudan bileş en adından sonra listeleyerek SEQUENCE OF veya SET OF yineleme sayısına bir kısıtlama uygulayabiliriz, ancak yinelenen türü sınırlamak istiyorsak, baş ka bir iç uygulamamız gereklidir. alt tip kısıtlaması, ancak bu sefer "Bİ LEŞ ENLİ " kelimeleriyle ("Bİ LEŞ ENLİ " yerine) baş layarak, doğrudan yinelenen türü uygulanacak kısıtlama ile devam eder.

8.3 Bir dizinin iç alt tipi

Son bir örnek olarak, daha önce tanıttı ıız iki boyutlu PrintableString dizimize geri dönelim. İ lk olarak ş unları tanımlayacağ iz:

```
Generic-array ::= PrintableString Dİ Zİ Sİ Nİ N Dİ Zİ Sİ
```

ve daha sonra, orijinal tanımımıza eş de ı er (neredeyse - aş a ıya bakın) olacak iç alt tiplene ile bir "Özel dizi" üreteceğ iz.

```
Dİ Zİ (BOYUT (1..100)) Dİ Zİ (BOYUT (20)) OF PrintableString (SIZE (0..15))
```

İ htıyacımız olan budur:

```

Özel dizi ::= Genel dizi (SIZE (1..100) KESİ ş İ M
Bİ LEŞ ENLİ
(BOYUT (20) Bİ LEŞ EN İ LE KESİ ş İ M
(BOYUT (0..15))
)
)

```

Neden sadece neredeyse eş dēğ er? İç alt tipe sahip bir Jenerik dizinin PER kodlamasının her zaman genel kodlama olduğu unu (iç alt tip kısıtlamaları PER görünür dēğ ildir) hatırlamak önemlidir, bu nedenle yukarıdaki kısıtlamalara sahip bir Özel dizinin uygulanması satırda aynı bitleri üretecektir. kısıtlamaları açıkça koymak, farklı (daha kompakt) bir kodlama üretecektir. Kısıtlamaların tüm uygulama sınıfları için geçerli olduğu veya farklı sınıflar arasında birlikte çalışmanın gerekliliği durumlarda, kısıtlamaları açıkça yerleş tirmek daha iyidir. Bununla birlikte, tam bir uygulama ile kısıtlı bir uygulama arasında birlikte çalışmanın gerekliliği durumlarda, kısıtlamayı ifade etmek için iç alt tiplemeyi kullanmak genellikle daha iyidir.

9 Sonuç

"Basit alt tipleme" gerçekten de basit olabilir - bir INTEGER tipi için bir aralık belirtildiğinde olduğu gibi, ancak çok güçlü kümeye aritmetiği iç ve iç alt tipleme özellikleri kullanılıyorsa yazarken dikkatli olunması (ve okurken sözdiziminin iyi anlaşılması) gereklidir.

Eski Temel Kodlama Kurallarında (BER) alt tipleme satırındaki bitleri hiç bir zaman etkilemediğinde, ASN.1 protokollerinin yazarları için alt tiplemeyi düş ümme zahmetine girmeme eğilimi vardı ve yüzeysel ele alınırsa birçok belirtim vardır. dēğ eri, belirsiz uzunlukta tamsayıları desteklemek için uygulamalar gereklidir, hatta 'her ne kadar' herkes niyetin asla bu olmadığıını bilse de.

Menzil ve boyut kısıtlamasının en basit biçimlerinin uygulanması çok basittir ve mümkün olduğuunda kullanılmalıdır. Kümeye aritmetiği veya iç alt tipleme kullanan daha karmaşık formlar çok güçlündür, ancak daha fazla uzmanlık içindir.

kullanmak.

Hem uygulama gereksinimlerine kesinlik kazandırmak için hem de daha yeni Paketlenmiş Kodlama Kuralları, alt tipleme uygulanırsa satırındaki bitleri azaltacağından, artık yeni veya revize edilmiş protokoller üretilirken, mümkün olan her yerde alt tiplemenin uygulanması şiddetle tavsiye edilmektedir. mantıklı. Bu özellikle tamsayı aralıkları ve SEQUENCE OF'lerin veya SET OF'lerin yinelemeleri için önemlidir.

4. Böüm etiketleme

(Veya: Kontrol et ya da unut!)

Özet: Etiketleme, 1994 öncesi ASN.1 gösteriminin önemli (ve zor!) bir parçasıydı. Üç faktör nedeniyle önemli (ve onu anlama ihtiyacı) artık çok daha az:

böümünde açıklandı gibi modül başlığında OTOMATİK ETİKETLER ortamı ayarlama yeteneği Böüm I Böüm 3;

bunu başarmak için etiketlere güvenmeden geniş letilebilirlik sağlanması;

etiketleri kodlamayan PER'nin tanıtımı.

Dört etiket sınıfı vardır:

EVRENSEL

BAŞ VURU

ÖZEL

bağlama özgü

ve bir etiket değil bir sınıf ve bir sayıdır (yukarı sıfır, sınırsız).

Bu böüm, ASN.1'in yasal bir parçasında etiketlerin kullanımına ilişkin gereklilikleri açıklar ve etiket sınıfı seçimi ile ilişkin biçimsel tavsiyeler verir.

1 Önceki tartışmaların gözden geçirilmesi

Etiketleri dahil etme fikrini zaten tartıştık ve örtük etiketleme ve açık etiketleme kavramlarını tanıttık ve bunları bir BER kodlaması üzerindeki etkileri açısından tanımladık: tür için TLV'deki "T"yi değil iş tirmek (örtük etiketleme), veya yeni bir TLV sarıcı ekleyerek (açık etiketleme).

Etiketler başlangıçta Temel Kodlama Kurallarının (BER) "TLV"sindeki "T" ile yakından ilişkiliydi ve kullanıcılarla farklılıklar ve seçenekler için kullanılan "T" değil erleri üzerinde kontrol sağladı. Ancak bir genel letilebilirlik işaretçisi olmayan bir BER ortamında sürüm 1 ve sürüm 2 arasındaki birlikte çalışma kolay olacaksa bu önemliydi.

Bu kesinlikle akademik değil

Notasyonun kodlama kurallarından bağımsız olması gerekī ī ve artık "TLV" kavramını kullanmayan başka ASN.1 kodlama kuralları olduğunu göz önüne alındığında, etiketlemeyi tartışmanın tamamen edici bir yolu (ancak birçok okuyucuyu tatmin edebilir!). Bu nedenle aşağıdaki ida, etiketlerin kodlama kuralından bağımsız ve biraz daha soyut (özür dilerim!) bir açıklamasını sunacağınız.

Önceki metinde ima ettik (yanlış!) - ama asla belirtmedik! - etiket değil erleri içīn ad alanının basit bir tam sayı olduğunu unu. Aslında, şe 21'de "[UYGULAMA 1]" etiketini kullandık, bu daha karmaşık ikinci bir ad uzayı̄ni ima edebilir. Etiketler içīn mevcut değil erlerin tamamını ve bunların normalde nasıl kullanıldığı inīn aşağı ida açıklıyoruz.

Son olarak, etiketlerin ne zaman farklı olması gerekī īyle ilgili kurallar olduğunu kısaca bahsetmiş tık (genel olarak, BER kodlamalarında belirsizliği sağlama için bir TLV'nin "T"sinin değil er bazı TLV'lerden farklı olması gerekī inde). Gerçek kuralları aşağı ida veriyoruz.

Ancak son bir önemli hatırlatma olarak: 1994 sonrası, etiketler hakkında hiçbir şevey bilmemeniz ve bunları tür tanımlarınıza asla dahil etmemeniz gereken otomatik bir etiketleme ortamı oluşturabilirsiniz. Bu, yeni spesifikasyonlar içīn benimsenmesi önerilen stilidir ve aşağı idaki metinle kafası karışan herkes içīn kesinlikle doğru yaklaşımındır!

Bir an içīn küresel düzeye bakalım. ASN.1'in tip notasyonu gerektirdī ī veya izin verdiği ī her yerde, aşağı idakilerin yazılmasına izin verilir:

etiket gösterimi tip gösterimi

Başka bir deyīsle etiketleme, eski bir türden yeni bir türün resmi olarak tanımlanmasıdır ve etiket gösterimi, aynı tür notasyonuna tekrar tekrar uygulanabilir. Yani aşağı idakiler yasaldır:

Türüm ::= [UYGULAMA 1] [3] TAM SAYI

ancak örtük etiketleme ortamında "[3]" hemen geçersiz kılındığından oldukça anlamsız olacaktır! Bu tür bir yapıyı nadiren görürsünüz - etiket notasyonu normalde bir tip referansına veya etiketlenmemiş tip notasyonuna uygulanır.

Son olarak, bir tür, etiket gösterimi kullanılarak tanımlanırsa, değil er gösterimi amacıyla etiket gösterimi göz ardı edilir. Yukarıdaki My-type içīn değil er gösterimi hala basitte "6"dir (örneğin).

2 Etiket adı alanı

Şuan içīn BER kodlamalarında kalmak: Bir etiket, bir BER TLV'nin "T" kısmının 7 bitini kodlar.

Kalan bitin etiketleme ile hiçbir ilgisi yoktur ve "V" bütümünün kendisi bir dizi TLV ise (ichīn kullanılan gibi oluşmuş bir kodlama) bire ayarlanır. "SEKANS" veya "AYARLA" ve

Etiketler

[UNIVERSAL 29]: UNIVERSAL sınıf etiketlerini kullanmayın

[UYGULAMA 10]: yaygın olarak kullanılan türler veya üst düzey mesajlar içīn kullanın. tekrar kullanmayın.

[Gİ_ZLİ_0]: Nadiren görülür. Bir standart özel eklemelerle genişletmek içīn kullanın (gerçekten yapmanız gerekiyorsa!).

[3]: Farklı bir bağlamda kullanın ve yeniden kullanın. En yaygın etiketleme biçimi.

"V" kısmı baş ka TLV'lerden oluş muyorsa sıfıra ("INTEGER" veya "BOOLEAN" veya "NULL" için kullanılan gibi ilkel bir kodlama).

Bir etiket, bir sınıf ve bir etiket dē eri vererek belirtilir (ikincisi ger̄ ekten de basit bir pozitif tamsayıdır - sıfır yukarı, sınırsız). Ancak sınıf, dört olasılıktan biridir:

EVRENSEL sınıf
UYGULAMA sınıfı
ÖZEL sınıf bağ lama
örgü sınıf

EVRENSEL 0	Kodlama kuralları tarafından kullanılmak üzere ayrılmış tir
EVRENSEL 1	Boole tipi
EVRENSEL 2	tamsayı türü
EVRENSEL 3	Bit dizisi türü
EVRENSEL 4	sekizli dizi türü
EVRENSEL 5	boş tip
EVRENSEL 6	Nesne tanımlayıcı türü
EVRENSEL 7	Nesne tanımlayıcı türü
EVRENSEL 8	Harici tür ve Örnek türü
EVRENSEL 9	Gerçek tip
EVRENSEL 10	Numaralandırılmış tür
EVRENSEL 11	Gömülü-pdv türü
EVRENSEL 12	UTF8Dize türü
EVRENSEL 13 - 15	Bu Önerinin gelecekteki sürümleri için ayrılmış tir Uluslararası Standart
EVRENSEL 16	Sıralama ve Sıralama türleri
EVRENSEL 17	Küme ve Küme türleri
EVRENSEL 18-22	Karakter dizisi türleri
EVRENSEL 23-24	Zaman türleri
EVRENSEL 35-30	Daha fazla karakter dizisi türü
EVRENSEL 31-...	Bu Öneriye ek olarak ayrılmış tir Uluslararası Standart

Ş ekil II-7: UNIVERSAL sınıf etiketlerinin atanması

Etiket gösteriminde, yalnızca köşeli parantez içindeki bir sayı, bağılama özel bir sınıf etiketinin etiket değil erini belirtir. Diğer sınıflar için, sınıfın adı (tamamı büyük harfle) açılış köşeli parantezinden sonra görünür.

Örneğin:

[EVRENSEL 29]
[UYGULAMA 10]
[ÖZEL 0] [3]

etiket değil eri 29, "evrensel" sınıf etiket değil eri 10,
"uygulama" sınıfı etiket değil eri 0, "özel" sınıf etiket değil eri
3, "bağılama özgü" sınıf

Dört etiket sınıfını, yalnızca farklı etiket "renkleri" (kırmızı, yeşil, mavi, sarı) olarak düşünmeyi seviyorum. Gerçek isimler önemli değil. Çok u amaç için, etiketin "rengi" de önemli değil ildir! Önemli olan tek şey, etiketlerin gerektiğiinde farklı olması ve "renkleri" (sınıfları) veya etiket değil erleri açısından farklılık gösterebilmeleridir. Kullanmayı seçtiğiniz renk, esas olarak stil meselesiştir.

Yalnızca bir katı yasak vardır: kullanıcıların türleri UNIVERSAL sınıf etiketlemesine izin verilmez. Bu sınıf (her zaman) bir türdeki "varsayılan etiket" için kullanılır ve bu tür etiketlerin değil erleri yalnızca ASN.1 spesifikasyonu içinde atanabilir.

Şekil II-7, X.680/ISO 8824-1'den (Eylül 1998'e kadar olan tüm değil iş iklikler dahil) bir tablonun bir kopyasıdır ve her biri için varsayılan etiket olarak atanın UNIVERSAL sınıf etiketini (örtülü etiketleme tarafından geçersiz kılınmadıkça kullanılır) verir. ASN.1'de tanımlanan tip notasyonları ve oluş turucu mekanizmaları.

Kullanıcılar tarafından UNIVERSAL sınıf etiketlerinin kullanılmasının yasaklanmasıının ana nedeni, gelecekte ASN.1'e yönelik uzantılar ortaya çıktığında sorunları önlemektir. Bununla birlikte, "renki" (sınıfı) ne olursa olsun, her etiket değil tüm etiketlerle eşit statüye sahip olduğunu, bunun gerçek bir zorluk olmadığına dikkat etmek önemlidir.

1994 öncesi ASN.1'e uygun, ancak UTF8String'i (1998'de eklendi) kullanmak isteyen ve 1994 sonrası tanımın metnini kendi uygulama belirtimlerine kopyalamaya karar veren belirtimler olmuştu. Bu muhalefeler zararsızdır, ancak kesinlikle spesifikasyona aykırıdır. Yasa dışı olmasının yanı sıra, metni kopyalamak ve kopyalanan metinde bir UNIVERSAL sınıf etiketi atamak da gereksizdir - türün tanımında bir UYGULAMA sınıfı etiketi kullanılabilir ve türün olduğunu yerde dolaylı olarak etiketlenmesi koşuluyla kullanıldığından, nihai sonuç, UNIVERSAL sınıf etiketine sahip bir ilk atamadan ayırt edilemez, çünkü daha sonraki örtük etiketleme de geçersiz kılar.

Peki ya değil er üç etiket sınıfı? Hangisi ne zaman kullanılmalı? Tekrarlamak gereklidir: hepsi eşit değildir. Dilerseniz ÖZEL sınıf etiketlerini kesinlikle her yerde kullanın! Ancak bir tarz olarak, çoğu insan çoğu zaman bağılama özel sınıf etiketleri kullanır (yazılması en kolay olanlardır - yalnızca köşeli parantez içindeki bir sayı!). "Bağılama özgü" adı, bunların yalnızca belirli bir bağılama (tipik olarak tek bir SEQUENCE, SET veya CHOICE içinde) kesin olduğunu ima eder ve bu etiketlerin (sıfırdan yukarı doğru) kullanılması (ve yeniden kullanılması) normaldir. Belirli yerlerde farklı etiketler gerektiren kurallara uymak için bir SEÇİM alternatiflerini veya bir SEQUENCE veya SET öğelerini etiketlemeniz gereklidir (aşağıda bakın).

APPLICATION sınıfı etiketlerini aşağıdaki şekilde kullanmak yaygın bir uygulamadır (ancak hiç bir şekilde evrensel veya gerekli değil ildir):

Bir uygulama sınıfı etiketi, tüm uygulama belirtiminde yalnızca bir kez kullanılır, asla iki kez uygulanmaz.

Uygulama için en dış taki tip bir SEÇİ M ise (genellikle öyledir), o zaman bu seçenekin alternatiflerinin her biri (mükemmelen dolaylı olarak) UYGULAMA sınıf etiketleriyle (genellikle [APPLICATION 0], [APPLICATION 1], [UYGULAMA 2], vb.) Bu yaklaşımı Kısım I Bölüm 3'ün § ekil 21'inde gördük.

Bir kez tanımlanan ve ardından uygulama belirtiminin birçok bölümde kullanılan bazı karmaşık türler varsa, tanımlanıklarında bunlara bir uygulama sınıfı etiketi verilir (ve bu etiket hiç bir zaman başka hiçbir şeye verilmez), böylece güvenli bir şekilde kullanılabilirler. bir seçenekde (örneğin), farklı etiketler gerektiren herhangi bir kuralı ihlal etme tehlikesi olmadan kullanılabilir (aynı tür SEÇİ M'de tekrar görünmediği sürece - muhtemelen farklı semantik ile).

Buna bir örnek, Kısım I Bölüm 3 ve 4'teki § ekil 13 ve 14'teki "Çıkış Tipi" ve "Adres" türleri olabilir. Dolayısıyla § ekil 14'te bunun yerine şunu yazabiliriz:

Çıkış Tipi ::= [UYGULAMA 10] SEKANS {

....
....}

Adres ::= [UYGULAMA 11] SIRALI {

....
....}

Üst düzey iletiler için 0 ile 9 arasındaki uygulama sınıfı etiketlerini ve yaygın olarak kullanılan türler için 10'dan sonraki etiketleri kullanma kararının alınması.

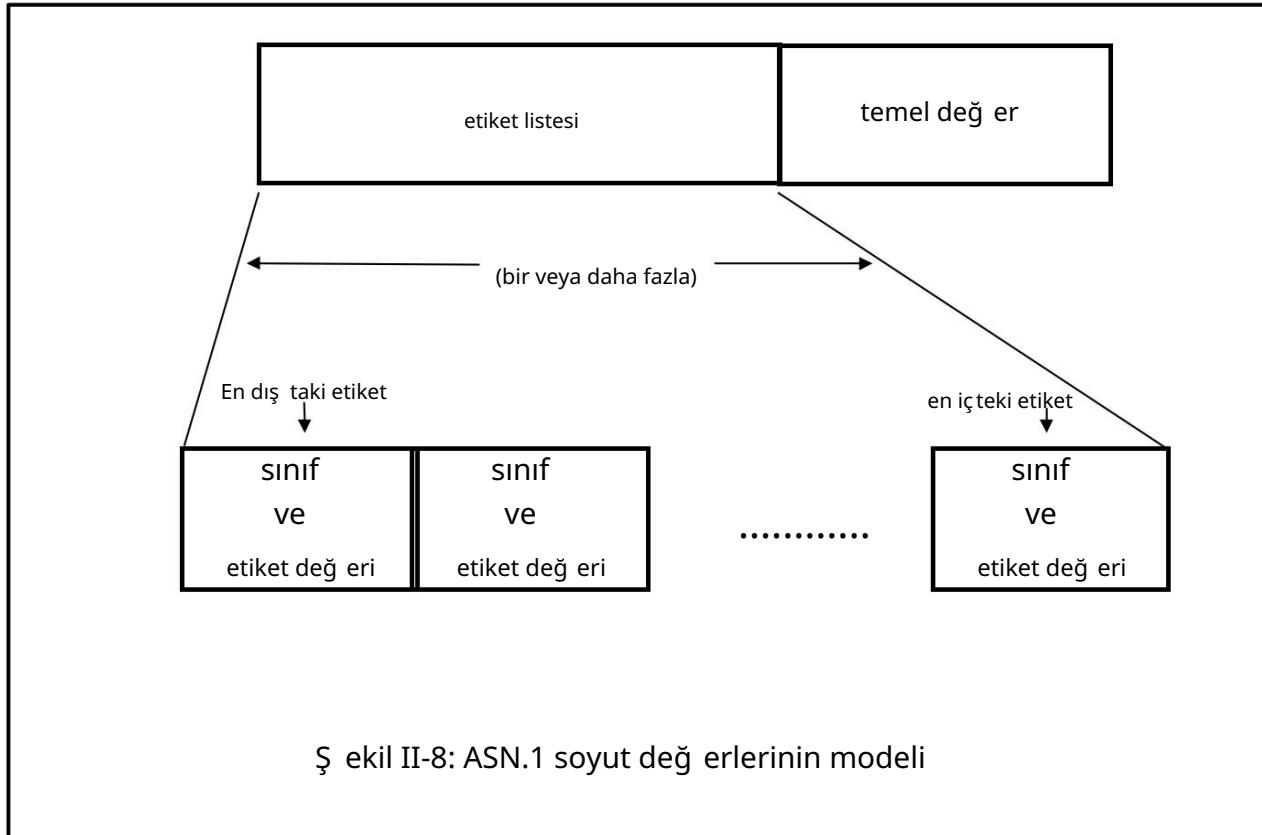
Bir etiket degenin büyüğlüğü ünün bir sınırı yoktur, ancak Bölüm III'te BER'yi incelediği imzide, kodlanacak etiket degenin şuna eşit veya daha küçük olması koşuluyla bir "T"nin tek bir sekizli olarak kodlanacağıını göreceğiz. 30, bu nedenle çoğu uygulama tasarımcısı genellikle tüm etiketleri için 31'in altındaki etiket degenin erlerini kullanmaya çalışır. (Ancak düşük yüzlerde etiket degenin erlerine sahip özellikleri vardır)

ÖZEL sınıf etiketleri, standartlaşmış tırılmış özellikleri asla kullanılmaz. Bazı sekansların veya setlerin sonuna ekstra öge eler ekleyerek uluslararası bir standart genişleten bazı çok uluslararası etiketler tarafından kullanılmıştır. Buradaki varsayımda (etiket degenin jiggery-pokery'de olduğu gibi) BER'in kullanılıyor olmasıdır ve (makul) umut, PRIVATE sınıf etiketleriyle yeni öğeler ekleyerek bunların temel standardın herhangi bir uzantısıyla çıkış mayacağının iddir. gelecek.

3 Soyut bir etiketleme modeli

Not: Bu malzeme ASN.1 spesifikasyonunda mevcut değildir. Bu yazar tarafından, gösterim düzeyinde etiketlemenin anlamının kodlama kurallarından bağımsız bir açıklamasını ve kodlama kurallarının davranışını belirlemenin bir yolunu sağlama için yararlı bir model olduğu düşündürmektedir. Çok u ASN.1 "uzmanı" muhtemelen modeli kabul eder, ancak bunun gerekli olmadığıını ve ASN.1 notasyonunun kodlama kurallarına temiz bir şekilde bağlama için belirttiğin şeye etiketlemenin birkaç olası yolundan yalnızca biri olduğuunu iddia edebilir. (Tekrar § ekil 999'a bakın!).

Etiketlemeyi, her ASN.1 özet degenin eriyle ilişkili bir etiket listesini etkileyebilecek şekilde modelleyebiliriz. Bazı kodlama kuralları, kodlamanın bir parçası olarak etiket listesindeki etiketlerin bir kısmını veya tamamını kullanır.



Etiketlemenin etkilerini açıklamanın bir yolunu sağlamak için, bu değerlerlere bazı yapılar içeren bir ASN.1 soyut değerler modeli (ASN.1 türlerinde bulunan "şeyler") tanıiyoruz. Bu, Şekil II-8'de gösterilmişdir.

Şekil II-8'de, her bir ASN.1 soyut değerinin, bir veya daha fazla etiketten oluşan an sıralı bir etiket listesiyle birlikte bir temel değerden ("integer 1", "boolean true" vb.) oluşan tuğunu görüyoruz. (en içteki, temel değer en yakın ve en dış takı, en uzak). Her etiket, daha önce açıklanmış gibi, bir sınıf ve bir etiket değerinden oluşur.

Bir tür, "BOOLEAN" veya "INTEGER" gibi ASN.1 tip gösterimi kullanılarak veya SEQUENCE veya SET gibi gösterim kullanımının sonucu olarak tanımlanmışında, tüm değerlerine aynı etiket listesi verilir - tek bir etiket (UNIVERSAL sınıfının hem en içteki hem de en dış takı). Her tip gösterimi için etiket değer ASN.1 spesifikasyonunda belirtilmişdir ve yukarıdaki Şekil II-7'de tekrarlanmıştır. (Önceki metinde buna için "varsayılan etiket" demişti).

Bir etiket listesinde mümkün olan yalnızca iki işlem vardır. Bir tür dolaylı olarak etiketlenmiş se, en dış takı etiket, etiketleme yapısında belirtilen yeni etiketle değer işlenir. Bir tür açıkça etiketlenmiş se, etiket listesine en dış takı yeni bir etiket eklenir. Tüm ASN.1 özel değerlerinin her zaman en az bir etiketi olduğuunu unutmayın. Açık etiketleme yoluyla ek etiketler alırlar ve etiket sayısını asla azaltamazlar.

Bu etiketleme modeliyle, artık Temel Kodlama Kurallarımızı, etiket listesindeki her etiket için, etiketin "T", "L" olduğu en dış takı etiketten en içteki etikete kadar bir "TLV" kodlamak olarak tanımlayabiliriz. ASN.1 özel değerinin kodlamasının geri kalanının uzunluğuunu tanımlar ve sonucusu dışındaki her bir "V" (yalnızca) bir sonraki TLV'yi içerir. Son "V", temel değerini tanımlayan bir kodlama içerir.

Okuyucu, bunun, açık etiketlemeyi "fazladan bir TLV katmanı ekleme" olarak tanımladığımızda elde edilenle tamamen aynı kodlamayı verdiğini fark edecktir, ancak soyut modelin kullanılması, gösterimin anlamını kodlama kuralı terimleriyle açıklamayı gereksiz kılmaktadır. . Bir etiket listesi kavramını, gösterim ve kodlama kuralları arasında bir tür dolaylı olarak kullanıyoruz. Bir ASN.1 aracının normalde sözdizimi analizi ve diğer işlevler arasında tutması gereken bilgileri temsil eder.

Son olarak, ama çok önemli olarak, çoğu tür içi, türdeki tüm değil erlerin tamamen aynı etiket listesine sahip olduğunu una dikkat edin. Türe daha fazla etiketleme uygularsak, o türdeki her değil er içi etiket listesini değil iş tiririz (yeni bir etiket ekler veya dış düzey etiketi değil iş tiririz).

Ayrıca, birçok amaç için (özellikle hangi etiket değil erlerine izin verildiği) önemli olan tek şeyle en dış taki etikettir. Bu nedenle, "türün etiketi" hakkında konuşmak anlamlıdır, çünkü o türün her soyut değil eri aynı etiket listesine (ve dolayısıyla aynı dış düzey etiketine) sahiptir. Ancak bu basit durumun bir istisnası vardır.

CHOICE yapıcısı, değil erleri, her bir değil erin orijinal etiket listesini koruduğu, alternatiflerin her birindeki değil er kümelerinin birleşimi olan yeni bir tür oluş turacak şekilde modellenmişdir. Bu nedenle, seçim türleri için, türdeki farklı soyut değil erlerin farklı etiket listeleri olması nedeniyle, "türün etiketinden" bahsetmek anlamlı değil ildir. (Kanonik kodlama kurallarında "öğeler etiket sırasına göre sıralanır" diyen bir metin görürseniz bunu hatırlamanız önemlidir - bir seçim türünün durumunu kapsayacak bazı niteleyici metinler arayın!)

Bununla birlikte, bir seçim türünün açıkça etiketlendiğiini varsayıyalım (seçim türleri için izin verilen tek etiketleme biçimi). Daha sonra, farklı soyut değil erler üzerindeki etiket listesi hala farklı olsa da (olacak), en dış taki etiket türdeki tüm soyut değil erler için aynıdır ve açıkça etiketlenen seçim, herhangi bir sıradan tür gibidir - her soyut değil er aynıdır outer-level etiketi ve bundan "tipin etiketi" olarak bahsedebiliriz.

Böylece artık çoğu türün "türün etiketi" olarak adlandırılabiliriz (imiz tek bir ilişkili etiketi (o türün tüm soyut değil erleri için ortak dış düzey etiketi) olduğunu, ancak etiketsiz bir seçim türünün birçok etiketi olduğunu anlayabiliriz. onunla ilişkili (değ erlerinden herhangi birinin tüm dış düzey etiketleri). Bu seçimim alternatiflerinden hiçbiri kendi başına seçim değil ilse, bu seçim türyle ilişkili dış düzey etiketlerin (tümü farklı) sayısı, alternatiflerinin sayısına eşitt olacaktır. Bununla birlikte, bazı alternatiflerin kendileri seçim türleri ise, bunların her biri tabloya çoklu (belirgin) dış düzey etiketler getirecek ve dış düzey seçim türü, alternatiflerinden daha fazla (belirgin) etiketle ilişkilendirilmiş olacaktır.

Örneğin, e2 er:

```
Benim seçimim ::= SEÇ M {alt1
    SEÇ M
    {alt1-1 [0] TAM SAYI, alt1-2 [1] TAM SAYI},
    alt2
    [2] AÇIKÇA Benim seçimim2}
```

o zaman "Benim seçimim" ile ilişkili etiketler bağlama özgü sıfır, bir ve ikidir. "My-choice2" içindeki tüm etiketler, açık etiketleme tarafından gizlenir.

Bu "türün etiketi" kavramıyla veya daha doğrusu "türle ilişkili etiketler" (ki bunlar her zaman farklıdır), farklı etiketlerin ne zaman gerekli olduğunuyla ilgili kuralları tartışmaya devam edebiliriz.

4 Etiketlerin ne zaman farklı olması gereki̇ğ inе iliş kin kurallar

Kural, farklı etiketlerin gerekli olmasıdır:

Farklı etiketlere ne zaman ihtiyacımız var?

SEÇİ M alternatifleri için;

bir SET'in elemanları için; ve

ardış İKİ VARSAYILAN veya İKİ STEĞEKLİ BAĞLI öğeler ve bir DİĞER ZİNCİRDEKİLER içindeki aşağıdaki iddilerin zorunlu öğeler için.

İşte - gerçekten basit, değil mi? (Geri kalanını atla!)

Aşağıda (ve ASN.1 spesifikasyonunda) verilen kurallar, etiket benzersizliği açısından ifade edilmişdir, ancak TLV tarzı bir kodlammanın net olmasını sağlamak için gerekli minimum kurallar olduklarını biliyorsanız en kolay hatırlanırlar!
Alternatif olarak, sadece kuralları hatırlayın ve mantıksız unutun!

Bir CHOICE yapıcısı içinde, her bir alternatif tarafından tabloya getirilen etiket koleksiyonunun tamamen farklı olması gereklidir. (Unutmayın, her alternatif tabloya yalnızca bir etiket getirir - etiketlenmemiş bir seçim türü olmadığı sürece, tabloya her bir alternatif için en az bir etiket getirdiğiinde, soyut değil erlerinin etiket listesinin ortak dış düzey etiketi. seçim türü, ancak bunların hepsi farklıdır.)

Benzer şekilde, bir SET yapıcısında, tüm $\overline{\alpha}$ elerin etiketleri, seçim türleri olan herhangi bir $\overline{\alpha}$ enin, eş leş tırme sürecine birkaç farklı etiket ekleme potansiyeline sahip olmasıyla, farklı olmalıdır.

SEQUENCE yapısında kurallar biraz daha karmaş ıktır. VARSAYILAN veya İ STEÇE BAĞLI olmadığıında, bir sıra türünün ögeleri üzerinde farklı etiketler için gereksinim yoktur. Ancak, VARSAYILAN veya İ STEÇE BAĞLI olduğuunda durum biraz da iş ir: VARSAYILAN veya İ STEÇE BAĞLI olarak iş aretlenmiş herhangi bir aralık ık ög e blog u için , varsa bir sonraki zorunlu ög eyle birlikte, o bloktaki tüm ögelerin etiketlerinin farklı olması gereklidir.

Bunu bir an için düş ünmek isteyeceksiniz. Açıka DEFAULT veya OPTIONAL ögelerin bloğunun hepsinin farklı etiketleri olmalıdır veya (BER'de) alıcı hangilerinin mevcut ve hangilerinin eksik olduğuunu bilmeyecektir, ancak aynı şekilde, bu etiketlerden biri bir sonraki zorunlu ög eyle eş leşirse, yine karışıklık olabilir. . Aşağıındaki zorunlu ög enin, önceki bloğun herhangi bir ög esinden farklı bir etiketi olmasını gerektirerek, bu etiketin bir kodlamada görünümü, OPTIONAL veya DEFAULT ögelerinden oluşan bloğun tamamlandığı ve dizinin geri kalanının işlendiği konusunda eksiksiz bilgi verir. Elemanlar normal bir şekilde ilerleyebilir.

Otomatik etiketlemeyi kullanmadan etiketlerinizi kontrol etmeye çalışiyorsanız, yalnızca küçük bir ek zorluk vardır. Bu, bir dizi içinde birden çok uzanti işaretçisinin olduğu durumlarda (örneğin, dizideki bir seçim öğesi içinde ve dizinin sonunda bir tane) geniş letilebilirlik işaretçisi ile farklı etiketler için kurallar arasındaki bir etkileşimdir. Buradaki kuralların amacı, bir sürüm 2 belirtiminin öğeler eklemesi durumunda, bu öğeleri alan bir sürüm 1 sisteminin sürüm 2 belirtiminin (BER ile - PER ile asla bir sorun yoktur!) şüpheyeye yer bırakmamasını sağlamak (tabii ki bilgisi yoktur!) seçim öğesini geniş letmiş veya diziye başka öğeler eklemiş tir. (İki durumda sürüm 1'de farklı özel durum iş leme belirtilmiş sebebi önemli olabilir.) Bu ek gereksinimlerin ayrıntıları için, Geniş letilebilirlik ile ilgili bir sonraki bölümdeki tartışmaya bakın.

Felsefi bir eğ ilime sahip olanlar iç in, tüm CHOICE yapıları otomatik olarak varsayılan bir etiketle (örneğ in UNIVERSAL 15) bir TLV sarmalayıcı üretseydi (gerçekten kuralları dikte eden BER'de) bu kuralların ne kadar basit olabileceğ ini düş ünmek isteyebilirsiniz. Dİ Zİ ile aynı ş ekilde! Bu kitabı akademik bir metin olarak kullanan herhangi biri, bu soruyu daha iyi ög recneler için bir alış tırma olarak ayarlamak isteyebilir! Lütfen PER'nin bir TLV felsefesi olmamasına rağmen, CHOICE ile iliş kili açık kodlamaya sahip olduğuunu unutmayın, ki bu BER'de yoktur. Bir gün birisi mükemmel kodlama kuralı felsefesini icat edecek!

5 Otomatik etiketleme

Bu madde yalnızca uygulayıcılar içindir!

"Otomatik etiketleme" ortamında hangi etiketler uygulanır? İ lk olarak, herhangi bir SET, SEQUENCE veya CHOICE notasyonu parçası, dış seviye elemanlarından veya alternatiflerinden herhangi birinde metinsel olarak mevcut bir etiket iç eriyorsa, o notasyonun dış seviyesi için otomatik etiketleme devre dış bırakılır. Aksi takdirde, örtük etiketleme ortamında her bir ög eye veya alternatif [0], [1], [2] vb. etiketler arka arkaya uygulanır. (Böylece SEÇİ M türü olan ög eler/alternatifler açılışa etiketlenir ve diğer tüm ög eler dolaylı olarak etiketlenir.)

6. Sonuç

Etiketleme karmaş ik görünür, ancak bir kez anlaş ıldığında görece basit bir meseledir. İ lk spesifikasyonlarda, bir stil meselesi olarak, SEQUENCE'ların ve SET'lerin tüm ög elerini ve SEÇİ M alternatiflerini sıfırdan yukarıya doğrudan bağlama özgü (örtük) etiketlerle basitçe etiketlemek yaygın hale geldi (etiketlenen tür "IMPLICIT" kelimesinden kaçınarak kendisi bir SEÇİ M).

Bir "örtük etiketleme" ortamının getirilmesiyle, bu biraz daha kolay hale geldi, ancak bu isteniyorsa, esasen otomatik etiketlemenin sağladığını ş ey budur.

Minimum gerekli etiketlemenin kullanıldığı birkaç özellik vardır. ASN.1 protokollerinin yazarları, minimalist bir yaklaşımın gerektirdiğiinden daha "simetrik" (ya da tembel?) olma eğilimindedir.

Tüm yeni modüllerin otomatik etiketleme ile üretilmesi ve etiketlerin unutulması bu yazarın kesin tavsiyesidir!

Bölüm 5

Geniş letilebilirlik, İstisnalar ve Sürüm Parantez

(Veya: Her zaman öğrenecek daha çok şey vardır!)

Özet: Bu bölüm:

sürüm 1 sistemleri ve sonraki sürüm 2 sistemleri arasında birlikte çalışmanın "geniş letilebilirlik" kavramını açıklar;

sürüm 2 eklemelerinin nerede olabileceğini belirtmek için bir "uzantı iş aretçisine" olan ihtiyacı açıklar;

bir uzantı iş aretçisine izin verilen tüm yerleri açıklar;

bir uzantı iş aretleyici kullanıldığıında tanımlı istisna iş leme ihtiyacını açıklar;

sonraki sürümlerde eklenen öğeleri bir arada gruplandırmak için "versiyon parantezleri" gösterimini açıklar; ve

Geniş letilebilirlik ile farklı etiketler için gereksinimler arasındaki etkileşimi açıklar.

Uzantı iş aretçisinin uygun yerlerde bulunması, Temel Kodlama Kuralları (BER) tarafından üretilenlerin yaklaşık %50'si kadar boyutta kodlamalar üreten Paketlenmiş Kodlama Kurallarının (PER) kullanımının anahtarıdır.

ASN.1 tabanlı protokol yazarlarının, gelecekteki sorunları en aza indirmek için sürüm 1 belirtimlerine uzantı iş aretleyicileri (tanımlı istisna iş leme ile) dahil etmeleri şiddetle tavsiye edilir.

1 Geniş letilebilirlik konsepti

NOT — Bu bölümde, daha fazla açıklama yapılmadan BER (Temel Kodlama Kuralları) ve PER (Paket Kodlama Kuralları) kısaltmaları kullanılmıştır.

"geniş letilebilirlik" nedir?
"Geniş letilebilirlik", destek, kodlama kuralları ve

Spesifikasyonunuza üç yıl önce yazdınız, çok sayıda saha uygulaması var - başları! Ama eklemeler yapmak istiyorsunuz. Nasıl görüş edersin?

Sürüm 1 sistemleri, eklemelerinizle ne yapacak?
ASN.1 geniş letilebilirliği size kontrol sağlar.

uygulama kuralları. Bu destek, sürüm 1 olarak belirtilen (ve uygulanan) bir protokolün birkaç yıl sonra özellikle izin verilen yollarla sürüm 2'ye yükseltilmesini sağlar. Sürüm 2 uzantılarının izin verilen uzantı kümlesi içinde olması koşuluyla (ve sürüm 1 protokolünün "geniş letilebilir" olarak iş aretenmesi koşuluyla), yeni sürüm 2 sistemleri ile halihazırda dağ itilmiş ve deň iş tirilmemiş sürüm 1 arasında iyi bir birlikte çalışma yeteneği olacaktır. sistemler.

Geniş letilebilirlikin anahtarları şunlardır:

Sürüm 2 eklemelerinin veya uzantılarının kodlamalardaki uzunluk sayımlarıyla "sarılmasını" ve sürüm 1 sistemleri tarafından "yabancı malzeme" olarak açıkça tanımlanabilmesini sağlamak için.

Sürüm 1 sistemlerinin kodlamaların "yabancı malzeme" olmayan kısımlarını normal sürüm 1 yöntemiyle işlemesi ve "yabancı malzeme" ile tanımlanmış ve öngörelebilir eylemler gerçekleştirmesi gerektiği inde dair net bir belirtim sağlamak.

Gereksiz (ve ayrıntılı) sarmalayıcılardan ve kodlamalardaki tanımlamalardan kaçınmak için, sürüm 2 eklemelerinin veya uzantılarının yapılması gerekebilecek yerlerde göstergeler "iş aretiler" kullanın.

Geniş letilebilirlik konseptinin başarılı olabilmesi için bu bileşenlerin üçünün de mevcut olması gereklidir.

Muhtemel istisna iş leme eylemlerinin ayrıntılı bir tartışması, Bölüm I Bölüm 7'de verilmektedir.

BER kodlama kurallarıyla, tüm alanların bir etiketi ve bunlarla ilişkili bir uzunluğu vardır; bu, yukarıdaki ilk noktayı kapsar, ancak üçüncü noktada kaçınmak istediği imiz ayrıntıları üretir. BER'in kendisi 2. nokta hakkında hiçbir şeyle söylemiyor. Bazı ileri görüşlü uygulama tasarımcıları, "Bir SEQUENCE veya SET içinde, uygulamalar, sürümlerinde beklenmeyen bir etiketi olan herhangi bir TLV'yi yok saymalıdır" gibi bir metin eklemiştir, ancak bu, hiçbir şekilde evrensel değil ve genel olarak protokolün farklı bölgelerinde "yabancı malzeme" konusunda farklı eylem belirtmek mümkün değil.

PER kodlama kurallarında, uzunluk sarmalayıcıları genellikle eksiktir ve etiketler her zaman eksiktir.

PER'ye, aşırı maliyet olmadan geniş letilebilirlik sağlanacaksız, uzunluk sargılarının nereye yerleştiğine ve sürüm 2 malzemesinin varlığıının veya yokluğunun kodlanması gerektiği söylenmelidir. "Uzantı iş aretilisinin" birincil amacı budur.

2 Uzatma iş aretilisi

Uzantı iş aretilisi neye benziyor? Bununla zaten Bölüm I Bölüm 3 ve 4'te Şekil 21 ve Şekil 22'de karşılaştırılmıştır. Şekil 21'in 26. satırındaki "satış" alternatifinden ve "satış verileri" öğesinin ardından gelen üç noktadır (üç nokta). Şekil 22.

Üç küçük noktaya dikkat edin.
Onları istediğiiniz sıklıkta koyun, hatta size çok az maliyeti var.
(BER'de sıfır, PER'de bir bit).

Okuyucu şimdilik Bölüm 3'teki Şekil II-3'e atıfta bulunursa, Şekil 21'deki "Wineco-protocol" SEÇİ Mİ NDEKİ uzantı iş aretilisinden sonra başka bir üçenin eklenmesiini görür. Bu bizim sürüm 2 eklememizdir.

("BİLEŞ ENLERLE {"den sonra bir üç noktanın da kullanıldığıını unutmayın. Bu, geniş letilebilirlik çalışmasından önce tarihlenen üç noktanın ayrı bir kullanımıdır ve geniş letilebilirlikle karıştırılmamalıdır.)

3 İ stisna belirtimi

Tüm uygulama için aynı istisna iş leme belirtilmektedir, tüm geniş letilebilirlik kullanımlarının bir istisna belirtimiyle birlikte kullanılması şiddetle tavsiye edilir.

Uygulamanız boyunca tek tip istisna iş lemeyi düş ünmüyorsanız, uzanti iş aretçinize bir istisna belirtimi ekleyin.

İ stisna belirtimi, sürüm 1 sistemlerinin uygulayıcılarının mesajdaki bu konumdaki "yabancı malzeme" ile ne yapması gerektiği ini açıkça ortaya koymaktadır (Şekil 21 ve 22'de olduğu gibi), ancak bu tavsiyeye şu anda evrenselleş olarak uyulmamaktadır.

İ stisna belirtiminin sözdizimi (geniş letilebilirlik gösteren herhangi bir üç noktadan hemen sonra görünebilir) bir tamsayı değil eri veya herhangi bir ASN.1 türünün adı, ardından iki nokta üst üste ve ardından o türün bir değil eridir. Tipik örnekler:

```
!3!
10!
PrintableString:"İşlem için yanlış hata kodu"
!Türüm:{field1 returnError, field2 "Code26"}
```

İlk ikisi, numaralandırılmış istisna iş leme prosedürlerinin olduğu bir liste olduğuunda kullanılabilir ve eklenen malzemenin her bir pozisyonunda hangisinin uygulanacağıını belirler. Üçüncü, istisnaların her zaman hata raporları verdiği ve değil erin yalnızca hata raporu metni olduğu durumlarda kullanılabilir. Son örnek, "Türüm"ün, ilk `if` e olası eylemlerin bir sıralaması (örneğin, "iptal", "returnError", "ignore", "treatAsMaximum" ve ikinci `if` e () ile bir Dİ Zİ olarak tanımlandığı yerde kullanılabilir. "treatAsMaximum"un bir kısıtlama içindeki bir üç nokta için uygun bir istisna iş leme prosedürü olabileceğini, "Yoksay"ın ise yalnızca bir SEQUENCE veya SET'e eklenen malzeme için geçerli olduğuunu unutmayın. beklenmeyen bir SEÇİ M alternatif, "returnError" istenebilir. ASN.1 gösterim araçlarını sağlar, ancak bunların nasıl uygun şekilde kullanılacağına yalnızca uygulama tasarımcısı karar verebilir. (Daha fazla tartışma için bkz. Bölüm I Bölüm 7.)

4 Üç nokta nereye yerleş tirilebilir?

İlk ASN.1 geniş letilebilirlik belirtiminde, elipsler asağıdaki gibi yerleş tirilebilir (ve uzantılar bunlardan sonra seri olarak eklenebilir) (Şekil II-9'daki çizimler sürüm 1 metnini ve ardından sürüm 2 metnini verir):

Herhangi bir Dİ Zİ veya SET veya SEÇİ M sonunda (bkz. Şekil 21 ve II-3).

Kısıtlamanın olduğu her yerde (bkz. Şekil II-9).

SAYILANDIRILMIŞ türdeki numaralandırma listesinin sonunda (bkz. Şekil II-9).

TAM SAYI (0..255, ...)	veya INTEGER (0..255, ... !1)
TAM SAYI (0..255, ..., 0..65535)	TAM SAYI (0..255, ... !1, 0..65535)

SAYILANDIRILMIŞ {kirmizi, mavi, yes il, ... }
 SAYILANDIRILMIŞ {kirmizi, mavi, yes il, ..., mor }

Ş ekil II-9: Geniş letilebilirlik iş aretçisi kullanımına iliş kin resimler

ASN.1 geniş letilebilirlik spesifikasyonuna yapılan erken bir ek, bir SEQUENCE, SET veya SEÇİ M'deki (ancak baş ka hiç bir yerde olmayan) yeni malzeme için ekleme noktasının yalnızca sonda değil il, ortada olmasına izin verdi. Bu, Ş ekil II-10'da gösterildiği gibi iki üç nokta ög esinin kullanılmasıyla iş aretlendi. Uygulayıcılara bu pozisyonda yabancı maddelerin taşı inmasının uygulama ş artnamesinin 50. maddesinde belirtildiği ini hatırlatmak için yine istisna spesifikasyonunu dahil etti.

```
Dİ Zİ {alan1
  TipA, alan2 TipB, !
  PrintableString : "Madde
  ...
  ... 50'ye bakın", -- Sürüm 2 materyali buraya gelir.

  ...
  alan3 TypeC}
```

Ş ekil II-10: 2. ve 3. ög eler arasında bir ekleme noktası

5 Versiyon parantezleri

Aynı ek, bir çift "[" açılış dirseğ i ve "]" kapanış dirseğ i ile sürüm parantezlerini tanıttı. Bunlar, herhangi bir ekleme noktasında ihtiyaç duyulan uzunluktaki sarmalayıcıların sayısını gereken minimum seviyeye indirmek için tanıtıldı - her yeni sürüm için bir sarmalayıcı ve ayrıca uygulama tasarımcıları, sürüm 1'de ne olduğunu tarihsel amaçlar için tanımlayabilmek istediklerini düş ündükleri için. , sürüm 2, sürüm 3, vb. Sürüm 2 ve 3 için uzantılarla, yukarıdaki sıra ş ekil II-11 gibi görünebilir.

Sürüm parantezleri satırındaki bitleri kaydetmekle kalmaz, aynı zamanda protokole yapılan eklemelerin geçmiş kaydını sağ lar.

```
Dİ Zİ {alan1
  TipA, alan2 TipB, !
  PrintableString : "59.
  ...
  ... maddeye bakın", -- Aşağı idakiler, 59. maddede belirtildiği gibi
  eski sistemler tarafından işlenir. [[ v2-field1 Type2A,
    v2-field2 Type2B ]],
  [[ v3-field1 Type3A,
    v3-field2 Type3B ]],
  ...
  -- Aşağı idakiler sürüm 1 materyalidir. alan3 TypeC}
```

Ş ekil II-11: İki sürüm eklemeli bir ekleme noktası

Bu yapılar diğ er geniş letilebilir yapılar içinde iç içe geçmiş olsa bile, geniş letilebilirliği in her SEQUENCE, SET ve CHOICE için bağımsız olarak tanımlanabileceğini dikkat edilmelidir. Bununla birlikte, bu tür herhangi bir yapı içinde, o yapının dış seviyesinde en fazla bir ekleme noktası olabilir ve malzeme, zaten yerleş tirilmiş herhangi bir malzemeden sonra ekleme noktasına art arda eklenir.

Sürüm parantezleri, revizyon geçmişinin net bir şekilde belgelenmesini sağlamak için yalnızca bir öge eklenmiş olsa bile normal olarak kullanılmalıdır.

Sürüm parantezlerinin yalnızca SEQUENCE, SET ve CHOICE yapılarına eklenebileceği ini, ENUMERATED veya kısıtlamalara eklenemeyeceği ini de unutmayın.

Bu kitabın yazıldığı sırası (1999 ortası), uzantı işaretleri ekleyen ve bazlarında ek malzeme ve sürüm parantezleri içeren bir dizi yayınlanmış belirtim vardı.

6 {...} gösterimi

Bir dizi belirtimde geniş letilebilir boş bir "tablo kısıtlaması" (daha sonra bakın) gibi görünen bir şeyle karşılaş acaksınız. Bu, Bilgi Nesnesi Sınıflarının kullanımı ile ilgilidir ve bunun tartışılması bu Bölüm'ün 7. Bölümüne kadar ertelenmiş tir.

7 Geniş letilebilirlik ve etiketleme arasındaki etkileşim

Bir önceki bölümde etiketleme ele alındığıında, geniş letilebilirliğin, etiketlerin farklılığı konusunda bazı ek gereksinimlere yol açtı ¹ kaydedildi.

Bu gereksinimler, bir ASN.1 tipinde birkaç uzanti iş aretçisi varsa, bunlarla iliş kili farklı istisna belirtimlerine sahip olabileceğinden ve bu nedenle sürüm 1 sistemlerinin "yabancı" malzemeyi belirli bir eklemleyle açık bir şekilde ilişkilendirebilmesinden kaynaklanır. nokta ve dolayısıyla istisna belirtimi.

NOT — Bu metinde verilen açıklamaların, BER kodlama kuralları net bir şekilde anlaşılırmadan anlaşılması zor olabilir. Bu kitap boyunca sıralı bir şekilde ilerleyen okuyucular, ya etiketleme konusunda "ad hoc" ve merak uyandıran başka kurallar olduğunu kabul etmeli ya da BER'deki metni okuyup bu bölümde geri dönmelidir. Afedersiniz! Daha iyisini yapamam!

Şans eseri (PER etiketleri kodlamadığı için) bu alanda PER kodlamasıyla ilgili herhangi bir sorun yoktur. Ancak, BER ile aşağidakiler gibi yapılar gerçek sorunlar yaratır:

Örnek1 ::= Dİ Zİ {alan1 SEÇİ M {alt1
TAM SAYI,

17 APR 2011

...!1 } I STEGE BAGLI

...!2)

veya

Örnek2 ::= SEÇİ M {alt2 SEÇİ M
 {alt3 TAM SAYI, ...!B },

...!4}

Şimdi, sürüm 2'de ekleme noktalarında !1 veya !2 istisna iş leme ile eklemler yapıldığıını varsayıyalım. "field1" isteğe bağlılığı olmasaydı, bu kolay olurdu - "alt1" varlığından önce yabancı maddenin varlığı açıktır. !1 durumudur ve bundan sonra bir !2 durumudur. Ancak alan1 isteğe bağlılığı olduğunu, sürüm 1 sistemlerinin !1'de yeni malzememiz olup olmadığıını veya !1'in eksik olup olmadığıını ve !2'de yeni malzeme olup olmadığıını belirlemesinin bir yolu yoktur. Benzer bir sorun !3 veya !4'te yeni malzeme ile ortaya çıkar.

Sorunun eklenen herhangi bir malzeme üzerindeki etikette olmadığıını, sorunun bu yapılarla geniş letilebilirlik kullanımında temel olduğunu unutmayın.

BER değin iş tırmamaya (korku çığlıklar - BER, geniş letilebilirlikten uzun süre önce gelir!), yukarıdaki iki (ve diğer benzer) yapıyı yasa dışı yapmak gereklidir. Bu nasıl yapılır?

ASN.1 Spesifikasiyonu biraz meraklı bir yaklaşım benimsiyor. Bir uzanti işaretçisi olan her yerde, etiketi de "kavramsallığı" eler" dışında hiçbir "eninkile eşleşmeyen bir "kavramsallığı" eklemelisiniz (mevcut herhangi bir uzantının sonuna). Ardından, farklı etiketlerin ne zaman gerekli olduğunuyla ilgili kuralları uygularsınız ve bunlar karşılanması, yasal olursunuz (ve sürüm 1 sistemi için yabancı materyali tek bir ekleme noktasına açık bir şekilde atmak için herhangi bir sorun olmayacağı).

Yukarıdaki durumların ilkinde, !1 konumuna kavramsallığı enin eklenmesi, "field1"in hem INTEGER etiketini hem de kavramsallığı enin etiketini tabloya getirmesi anlamına gelir. !1 kincisi, !2 konumunda aşağı idaki (zorunlu) kavramsallığı enin etiketile çakışıyor, bu nedenle iş yasa dışıdır.

Yukarıdaki durumların ikincisinde, "alt2" tabloya kavramsallığı enin etiketini (ayrıca INTEGER etiketinin yanı sıra) getirir, bu da yine !4 uzantısındaki kavramsallığı enin etiketile çakışır. Yani yine hukuki sorun var.

(Lütfen tekrar Şekil 999'a bakın!) Burada bunun geniş letilebilirlik ile belirgin bir karmaşık ilişkili olduğunu not etmek önemlidir. Daha önce OTOMATİK ETİKETLER kullanmanız ve ardından etiketlemeyi unutmanız gerekiyor i konusunda tavsiyede bulunduktan sonra, şimdilik söyleyorum (ve ASN.1 Spesifikasiyonu söyleyiyor), bazı geniş letilebilirlik yapılarının yasal olup olmadığıını belirlemek için adil bir şekilde sahip olmanız gerektir. Gelişmiş etiketleme anlayışı. Elbette, ASN.1'inizi kontrol etmek için OSS tarafından sağlanan benzer bir araç kullanırsınız, size kuralları çiğnediğiini anında söyleyecektir, ancak bu durumlarda hata mesajını anlamayacağınız daha şüpheyidir!

Yani bazı basit tavsiyelere ihtiyacımız var:

Bir DİZİ DE bir SEÇİ M İSTEÇİ BAĞLI ise, bunun bir uzanti işaretçisinden önceki sonlığı enin olmadığından emin olun veya kendisinin geniş letilebilir olmadığından emin olun. (Ve başka bir geniş letilebilir SET ile takip etmeyin!)

SEÇİ M bir SET içindeyse, SEÇİ M ve SET'ten yalnızca birinin geniş letilebilir olduğunu emin olun.

Geniş letilebilir bir SEÇİ M'i asla başka bir geniş letilebilir SET içine koymayın.

Özetle, geniş letilebilir CHOICE'ları radyoaktif malzeme gibi ele alın - onları birbirinden oldukça uzak tutun ve
diğer uzantı iş aretçilərindən net bir şekilde ayrı tutun! Bunu yaparsanız, hiç bir zaman sorun olmaz.

Bu kurallar gerçekten geçicidir, ancak uygulanmaları basittir ve yukarıda açıklanan sorunları ortadan kaldıracaktır.

Tabii ki, bu kuralları çiğ nerseniz, yasal olarak yasa dışı ASN.1 yazıyorsunuz ve iyi bir araç size bunu söyleyecek ve muhtemelen kodlamayı reddedecektir! Ancak, kendiniz kodlarsanız pekala, pratikte yalnızca çeşitli itli uzantılarda farklı istisna iş lemleriniz varsa sorunlar ortaya çıkar. Sadece yukarıdaki noktaları aklınızda bulundurun ve iyi olmalısınız.

8 Sonuç açıklamaları

Uzantı iş aretçisini ve onun istisna belirtimi ile ilişkisini ve görünüş te zararsız uzantı iş aretçilərinin ne zaman yasa dışı olduğunu konusunda bazı karmaşık kurallar üretme ihtiyacına yol açan BER'den kaynaklanan komplikasyonları tanımladık.

Son olarak, geniş letilebilirlik in sürüm 1 ve sürüm 2 sistemleri arasında sağladığını birlikte çalışmanın, sürüm 1'de mevcut olan uzantı iş aretçisine ve yalnızca geniş letilebilirlik hükümlerinin izin verdiği şeklinde protokolde yapılanın deşikliklere bağlı olduğunu dikkat etmek önemlidir (ek olarak elemanlar, alternatifler, numaralandırmalar, eklemeye noktasında ve kısıtlamaların gevşetilmesi).

Bir belirtimde geniş letilebilirlik hükümlerinin kapsamadığı deşiklikler yapılarsa (yeni öğelerin rasgele eklenmesi gibi), o zaman bu yeni sürümün kodlamaları, bir sürüm 1 sistemine gönderilirse muhtemelen öngörülemediği etkiler üretecektir. Benzer şekilde, sürüm 1'de bulunmayan bir geniş letilebilirlik iş aretçisinin sürüm 2'ye eklenmesi, sürüm 2 malzemesinin kodlamalarının, sürüm 1 sistemlerine gönderilirse öngörülemediği etkiler üreteceğin anlamına gelir.

Yukarıda açıklanan öngörülemezlik basitçe "Bir şekilde iptal mi edecekler yoksa görünen hatayı görmezden mi gelecekler?" arasında olabilir, ancak "Bazı sürüm 2 deşikliklerinin kodlamalarında sürüm 1 sistemleri bunların tamamen ilgiziz sürüm 1'in doğrudu kodlamaları olduğunu düşünecektir" olabilir. deşiklikler" ve buna göre hareket edecekler ki bu çok tehlikeli olabilir. Bu nedenle, genellikle geniş letilebilirlik kurallarına uylayan sürüm 2 türlerinin kodlamalarının sürüm 1 sistemlerine gönderilmesini önlemek önemlidir. Bu, elbette birçok şekilde yapılabılır, en yaygın olanı, bir bağlılığı ilk kurulduğunda bir tür sürüm müzakeresidir.

Geniş letilebilirlik ve özel durum iş leme, güçlü araçlardır ve yüksek düzeyde optimize edilmiş kodlama kurallarının kullanılmasını sağlar. Kullanımlarını düzenleyen kurallara uyulursa güvenlidirler.

Bununla birlikte, sürüm 1 belirtimlerine oldukça serbest bir şekilde uzantı iş aretleri eklemek (veya EXTENSIBILITY IMPLIED gösterimini kullanmak) çok önemlidir.

Bölüm 6

Bilgi Nesnesi Sınıfları, Kısıtlamalar ve Parametrelendirme

(Veya: Eksik olanı - hassasiyetle tamamlamak)

Özet:

Bu bölüm:

protokollerdeki "delikler" kavramının kısa bir açıklamasını sağlar;
içerinde "delikler" bulunan türleri tanımlama ihtiyacına özel bir örnek sağlar için ROSE
(Uzaktan İşlem Hizmeti Öğesi) protokolünü ve "deliklerin" mevcudiyetinde açık belirtimleri
desteklemek için notasyona duyulan ihtiyacı kısaca açıklar;

Bilgi Nesnesi, Bilgi Nesnesi Sınıfı ve Bilgi Nesnesi Kümesi kavramlarının açık bir
ifadesini ve bu Nesne Kümelerinin, "delikleri" (ve birden fazla boş luğ u doldurmak
için tutarlılık ilişkilerini) sınırlayarak kısmi bir protokol belirtimini tamamlamak
için kullanımını sağlar. bir taş iyici protokolü.

Açıklamaya devam ediyor:

Bir Bilgi Nesnesi Sınıfı, Bilgi Nesneleri ve
Örnek olarak wineco protokolünün bir geliş tirmesini kullanan Bilgi Nesnesi Setleri;

ROSE protokolünün basitleş tirilmiş bir versiyonu örnek olarak kullanılarak,
tanımlanan Bilgi Nesnesi Kümelerinin sınırlamaları amaçlanan "deliklerle"
ilişkilendirilebileceği araçlar;

parametreleş tirme ihtiyacını ve ASN.1 belirtimlerinin parametreleş tirme
sözdizimini açıklar.

Bir öğe rencide "söleyeceğim şe y zor" demek kötü bir uygulama olmalıdır! Ancak bilgi nesnesi
kavramları, ASN.1'in kavramsal olarak daha zor kısımları arasındadır ve bu kavramları bu bölümde
nazikçe tanıtacağınız ve son ayrıntıları bir sonraki bölümde dolduracağınız. Çok kolaysa bu bölümü
atlayın-okuyun!

1 "deliklere" duyulan ihtiyaç ve onlar için notasyon desteği i

1.1 OSI Katmanı

ASN.1'in ilk standartlaşdırıldığı yerde OSI kararlılığı içinde olmasına rağmen, Açık Sistemler Ara Bağlantısı (OSI) muhtemelen bu kitapta ilk kez ciddi bir şekilde tartışılmıyor.

**Tınsan psikolojisine bir saptırma:
Asla eksiksiz bir şevey yazmayın -
kullanımını sınırlar!**

OSI, 7 katmanlı OSI modelinin üretimiyle mimarisini belgeleme sorununu ciddiye alan belki de ilk protokol paketi spesifikasyonuydu. Satıcıya özgü birçok protokolün bir miktar katmanlama kavramı vardı ve TCP/IP çalışması 1970'lerin sonunda IP'yi TCP'den ayırmıştı, ancak OSI modeli katmanlama kavramını açıklamaya yönelik en eksiksiz giriş imdi.

7-katmanlı model (1984'te), bilgisayarların nasıl iletişim kuracağını belirleme görevinin (oldukça zor) basitleşirtilmesini sağlama yönlek en son giriş imdi. Bu spesifikasyon parçaları arasındaki bağlantılar.

Bu "mimari", öncelikle, birkaç grubun spesifikasyonun farklı bümeleri üzerinde aynı anda çalışmasını mümkün kılmayı amaçlıyor da, önemli bir yan ürün, spesifikasyon parçalarının yeniden kullanılabilirliğini sağlama konusunda. Bu, aynı ağı üzerinde birçok farklı uygulamayı taşıma imkânının ağı spesifikasyonlarının yeniden kullanılabilirliğini veya bazıları uygulama spesifikasyonu ilk yazıldığıında icat edilmemiş olabilecek birçok farklı ağı teknolojisi üzerinde çalışmak için uygulama spesifikasyonlarının yeniden kullanılabilirliğini içermektedir.

Okuyucu, bunu, uygulama semantikinden elektriksel sinyallemeye kadar her şevey tamamen ve kesin olarak tanımlayan tek bir yekpare spesifikasyonun (belge) olduğunu, (esas olarak askeri arenada ama aynı zamanda telefonda da kullanılan) eski sözde "bağlantı" protokollerle karşılaştırmalıdır.

Uluslararası Standartlar Örgütü'nün (ISO) 7 katmanlı modelinde, her katman, iletilen mesajların kısmının bir özelliğini sağlıyor; her mesaj, içinde mesajların bit modellerini taşıyan bir "deliği" (kullanıcı verileri denir) sahipti. Sonraki daha yüksek katman tarafından tanımlanır. Bununla birlikte, bir "diş arı yayılma" ve "girme" durumu söz konusuydu: birçok olası alt katman (örneğin, aktarım veya ağı protokoller), herhangi bir üst düzey mesajı taşıma imkânını kullanılabiliyor ve herhangi bir verili aktarım (veya ağı) kullanılabiliyor.) birçok farklı yüksek katman mesajı taşıyabilir. Çok esnek bir çoktan çoğalma durumu.

OSI deliklerle doluydu! Her bir belirtim katmanı, toplam mesajın küçük bir bölümünü tanımlıyor, ardından bir sonraki daha yüksek belirtim katmanında belirtilen bitlerle doldurulacak büyük bir boşluk bırakıyordu.

Ancak orijinal ISO OSI modelindeki temel kavram, her uygulama katmanı belirtiminin son boşluğun dolduracağıydı - her uygulama katmanı standartı, bazı uygulamalar için eksiksiz bir belirtim üreticekti.

Uygulama katmanındaki "yararlı araçların" kısmının özellikleri kavramını tabloya getiren CCITT 7 katmanlı modeli (nihayetinde ISO tarafından benimsendi), her biri bir "deliği" dolduran potansiyel olarak sonsuz bir katman dizisini tanıdı. Altındaki katman, ancak kendisi değil er gruplarının zamanı geldiğinde doldurması için "delikler" bırakıyor.

ASN.1, uygulama özelliklerini tanımlamak için giderek artan bir şekilde tercih edilen notasyon haline geldikçe, ASN.1'de "delikler" için açıkça bir destek e ihtiyaç duyuldu.

ASN.1'de 1.2 Delik destek i

Şimdilik teorik modelleri unutun. 1984'te ASN.1 kullanarak uygulama belirtimleri yazan kişi ilerin, diğer er gruplarla (çoklu, bağımsız, gruplar) veri türlerinde "delikler" bırakılarak "genel" veya "taş ıycı" bir belirtim yazabilmek istedikleri hızla anlaşıldı. delikleri neyin doldurduğuna dair spesifikasyonlar sağlamak.

Delikler uygulama katmanında bitmedi: insanlar "genel" spesifikasyonları "delikler" bırakılarak yazmak istediler. Ve bu boş lukları dolduran insanlar yine de birkaç boş luk daha bırakmak istediler!

Bu noktada, "bazı şeyleri diğer erlerinin tanımlaması için tanımsız bırakmanın" (en açık şekilde) mesaj formatının tanımsız bir parçası (OSI katmanındaki kullanıcı verileri) veya biri olabileceğini kabul etmek önemlidir. bir ASN.1 dizisindeki öğeler, ancak aynı zamanda bir bilgisayar alışveriş i yürütmeye prosedürlerinin tanımsız bir parçası da olabilir. Her iki "delik" türü de gerçek spesifikasyonlarda meydana gelmiş tir ve bir spesifikasyondaki herhangi bir "boş luk un" varlığıını ve doğasını açıkça tanımlamak için notasyona ve "delikleri" doldurmak için "kullanıcı" belirticilerinin notasyonuna ihtiyaç vardır.

Bir başka önemli nokta daha vardır: birkaç farklı (kullanıcı) grup, bazı taş ıycıların veya jenerik protokolün boş luklarına uygun uygulamalar için özellikler sağlıyorsa, çoğu zaman uygulamalar bu kullanıcının spesifikasyonlarından birkaçını desteklemek ister ve bunu yapabilmeleri gereklidir. belirli bir iletişim imzayı indeki boş lukları doldurmak için hangi spesifikasyonun kullanıldığıını iletişim imzamında kesin olarak belirleyin. Bu daha çok katmanlı bir mimarideki "protokol kimliği" kavramına benzer. Kullanıcı spesifikasyonu için sadece bazı bilgi kodlamalarını değil, aynı zamanda bu spesifikasyonun bir tanımını da taşımak için deliklere duyulan ihtiyacın farkındayız.

"Delikler" için en eski ASN.1 destek i, (pek çok tartışılmaya tabii) 1994'te geri çekilen "ANY" göstergesi ve erken ve büyük ölçüde başkaları bir ilişkilendirme girişimi olan "makro notasyon" ile birlikteyi. bir delik in içeriğinin (belirli bir uygulama için) belirli bir delik oluşturumuna (bir taş ıycı spesifikasyonunda tanımlanan malzeme).

1994 yılında ASN.1 "Bilgi Nesnesi Sınıfı" ve ilgili kavramlar, "delikleri" işlemek tercih edilen yolu olarak ortaya çıktı. Bu dönemde bundan sonra ROSE (Uzaktan İşlemler Hizmeti Öğesi) kavramlarını tanıtabileceğiz ve ROSE'un kullanıcılarının ROSE protokolünde bırakılan boş lukları tamamlamasına izin vermek için notasyona nasıl ihtiyaç duyduğumuzu göstereceğiz. Daha sonra, ROSE spesifikasyonunun bir kullanıcısı eksiksiz bir uygulama spesifikasyonu oluşturduğunda sahne olması gereken bilgilerin doğasını kısaca açıklıyoruz. Daha sonra ASN.1 "Bilgi Nesnesi Sınıfları" ile ilişkili kavramlara geçiyoruz.

2 ROSE çoğulca ırma modeli

2.1 Giriş

ASN.1 notasyonunun en eski kullanıcılarından biri ROSE (Uzaktan İşlemler Hizmeti Öğesi) - bir spesifikasyonu oluşturmak -

Bir iş işlemi ÇAĞRIN (çoğulca ırma kimliği i, işlem kimliği i, bağımsız değil işken değil erleri), işlemden bağımsız bir REJECT (önceden tanımlanmış hata kodları) veya bir RESULT (çoğulca ırma kimliği i, sonuç değil erleri) veya bir HATA (çoğulca ırma kimliği i, hata kodu, ilişkili parametreler geri alın hata ile).

sadece ROS (Uzaktan İş İemler Hizmeti) olarak adlandırılır. Bu, yine de Bilgi Nesnesi Sınıfı kavramının kullanımına ilişkin anlaşılması en kolay örneklerden birini sağlar ve burada ROSE'u tanıtmak için biraz zaman ayrılmıştır.

Ancak okuyucu, ROSE'un bu şekilde ele alınmasının TAM OLMADIĞI ve bilgi tabloları tanıtıldığıında, ROSE'un en son sürümünün aşağında açıklanandan çok daha fazla sütuna sahip olduğunu not etmelidir. Bazı basitleştirmeler ve/veya bazı uzantılarla kendi ROSE sürümlerini yazan bir dizi spesifikasyon vardır, bu nedenle "İŞ LEM" veya "HATA" kullanan bir metin görürseniz, bu adların nereden alındığıını kontrol edin. Gerçek ROSE spesifikasyonundan ithal edilebilirler veya bir ROSE "benzeri" olabilirler. Bu metindeki tanımlar bir ROSE "benzeridir" - gerçek ROSE tanımlarının basitleştirilmiş halidir.

Bir dizi standardizasyon grubu (en sonucusu CORBA'dır) tarafından protokollerin belirlenmesine yönelik yaygın bir yaklaşım, uzak bir sistemde bir iş İemi (veya yöntemi veya bir arabirim etkinleşiren) başlatan bir sistem kavramını tanıtmaktır. Bu, çağırılan iş İemin ayrıntılarını taşımak için (ROSE durumunda ASN.1'de tanımlanan) bir tür mesaj gerektirir, en önemli üç [özellik](#) eş unlardır:

döndürulen herhangi bir sonuç veya hatanın çağırı ile ilişkilendirilebilmesi için bu çağırının bir miktar tanımlanması; ve

gerçekleştirilecek iş İemin bazı tanımlamaları; ve

iş İem için tüm bağımsız değil iş kenleri veya giriş parametrelerini taşıyacak bazı ASN.1 türünün (o iş İeme özel) değil eri.

Buna ROSE INVOKE mesajı denir ("Invoke" olarak adlandırılan bir ASN.1 türü olarak tanımlanır). ROSE, "çağırma tanımlaması" kavramını ortaya attı, çünkü birden çok (belki de aynı) operasyon örneğinin daha öncekilerin sonuçları gelmeden önce başlatılabilceğini ve gerçekten de sonuçların aynı sırada gelmeyeceğini kabul etti. nerede başlatıldı sipariş operasyonları.

Burada, ROSE belirtiminin çağırı mesajının kavramlarını ve biçimini tanımlayacağıını, ancak diğer bir çok grubun iş İemleri tanımlamak için bağımsız olarak değil erler atayacağıını, bağımsız değil iş kenleri veya giriş parametrelerini taşımak için ASN.1 tipini tanımlayacağıını not etmek önemlidir. ve ilişkili semantik belirtin. Bunu yapmak ve bu tür tanımları ROSE INVOKE mesajının ASN.1 tanımında bırakılan boş luklara net bir şekilde bağılayabilmek için bir notasyona ihtiyaçları vardır.

Bu bağlamda kullanılan ASN.1, bazen "Arayüz Tanımlama Dili" (IDL) olarak adlandırılan bir dil olarak kullanılmaktadır, ancak ASN.1'in bu tür kullanımı sınırlı olmadığından ve aşağıda durumlarda protokol tanımına uygulanabileceğini unutmamak önemlidir: uzaktan çağırma ve sonuçların döndürülmesi kavramı yoktur.

INVOKE mesajının kendisi tam bir ASN.1 tipi tanımı değil. Bir iş İemin bağımsız değil iş kenlerinin değil erlerini taşımak için kullanılan ASN.1 türünü taşıyabilen bir "delik" i" vardır.

Bu "delik" ve INVOKE mesajındaki iş İem kodu alanının değil eri açıkça tutarlı bir şekilde doldurulmalıdır - yani iş İem kodu ve tip eşleşmelidir.

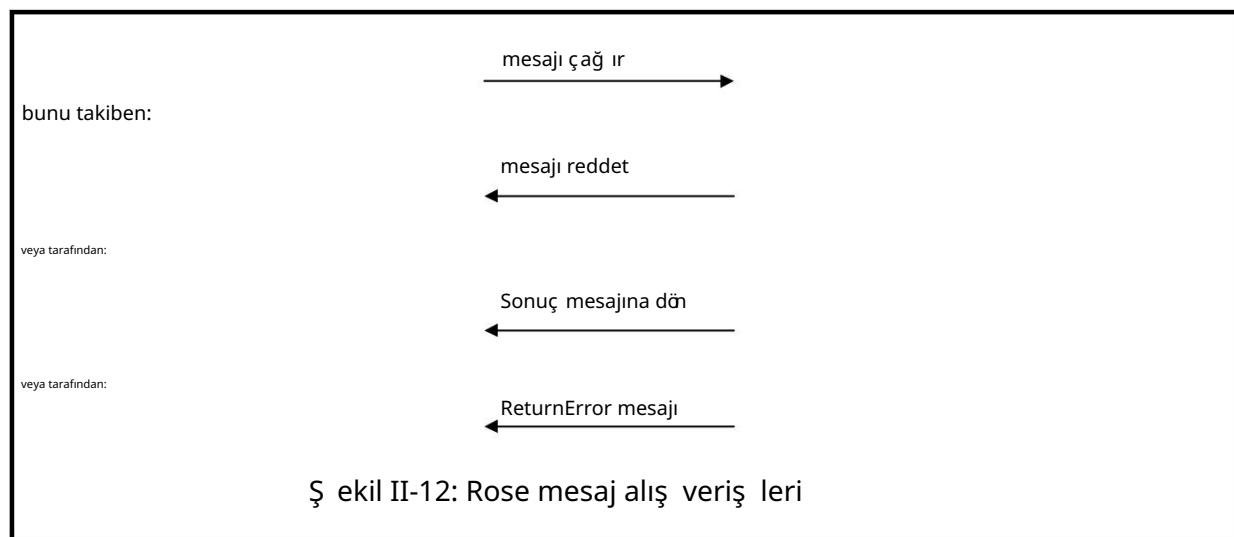
2.2 INVOKE mesajına yanıt verme

ROSE kavramı, bir INVOKE mesajına, "iş lemleri uygulanmadı" (kesinlikle, "tanınmayan İş lemi çağırır"), "sistem mesajı" (kesinlikle, "resourceLimitation" gibi iş lemleri bağ ımsız hata göstergeleri taşıyan bir REJECT mesajı tarafından yanıt verilebileceğiini söyleyebilir.), vb). ROSE, bir REJECT mesajı ile bildirilebilen yaklaşık 40 farklı hata veya problem durumuna sahiptir.

"ROSE mesajları boş luklarla dolu - tamamlanan REJECT dışındadır!"

Ancak böyle bir mesaj yoksa, iş lemleri arıyla başlatılır ve "amaçlanan sonuç" (SONUÇ mesajı) veya iş lemleri ile bir "hata yanıtı" (HATA mesajı) ile sonuçlanır.

ROSE çağırıcı şebeke II-12'de gösterilmektedir.



"Amaçlanan sonuç" ve "hata yanıtı" arasındaki bu ayrım kesinlikle gerekli değil, ancak ASN.1 tanımını basitleştirir. Buradaki varsayımdır, herhangi bir grubun, her biri bir tanımlamaya ve INVOKE mesajı deliği içinde giriş argümanlarını taşımak için tam olarak bir ASN.1 tipine ve tam olarak bir ASN.1'e sahip olacak bir dizi yakından ilişkili iş lemleri tanımlayacağıdır. SONUÇ ileti deliği içinde çıktı bağ ımsız değil iş kenlerini taşımak için yazın. Bununla birlikte, bu tam iş lemleri kümeleri için, herhangi bir iş lemin bu hataların belirli bir alt kümeye yol açabileceğinden şebeke II-12, bir dizi olası hata dönüşümü olmasına muhtemeldir. Her hata için bir hata koduna ve hatayı ilgili ek bilgileri (ROSE'un parametreleri çağırıldı) taşımak için bir ASN.1 türüne ihtiyacımız var ve tabii ki hangi iş lemlerinin hangi hatalardan kaynaklanabileceğini belirleyebilmemiz gerekiyor.

3 Kullanıcı özelliklerini tamamlamak için tabloların kullanılması

Burada wineco protokolümüze dönüyoruz ve wineco
değ iş imlerimizi desteklemek için ROSE (eksik) protokolünü nasıl
kullandığımızı göstermek için önce resmi olmayan bir tablo biçimini
kullanacağımız ASN.1'i kullanarak zaten iki ana mesaj belirledik,
yani

ve stok için sipariş
Satışların geri dönüşü

**Wineco değ iş imlerini bir dizi
uzak iş lem olarak ifade etmek
zorunda değil ılsınız, ancak basit
ve kullanımı kolaydır.**

ASN.1 türlerini kendileri tanımlamadan, bir ROSE INVOKE ile iletmek isteyebileceğimiz iki wineco mesajı daha ekleyeceğiz, yani

ve Sorgu kullanılabilirliği
İstek-sipariş -durumu

Bu mesajların ilki, hemen teslimat için öğelerin mevcudiyetini sorgular ve ikincisi, daha önceki bir siparişin durumu hakkında bir güncelleme ister.

Bu mesajların dördünü de ya bir yanıt ya da bir hata dönüşü üretecek bir ROSE işlemi yapacağımız "Stok için sipariş" yanıtı, "Sipariş onaylandı" mesajı olacaktır.

Bir "Satış İadesi"nin baş arılı bir şeyle işlenmesi, bir ASN.1 NULL'un döndürülmesine neden olacaktır. "Sorgu kullanılabilirliği" yanıtı, "Stok durumu yanıtı" olacaktır ve "Sipariş durumu iste" yanıtı, "Sipariş durumu" yanıtı olacaktır.

Bu isteklerin (işlemelerin) bir kısmının veya tamamının aşağıdaki hataları üretebileceğiini öngöryoruz (her durumda, hatanın daha fazla ayrıntısını veren bazı ek verilerle birlikte):

Güvenlik kontrolü hatası.

Bilinmeyen şube

Sipariş numarası bilinmiyor.

Öğeler kullanılamıyor.

ROSE Reddetme mesajında taşınan ve ROSE tarafından bize sağlanan işleminden bağımsız başka hatalar olduğunu unutmayın, ancak bunları dikkate almamız gerekmek. Burada sadece kendi operasyonlarımıza özgü hatalarla ilgileniyoruz.

Tüm bunları daha resmi olarak söylememiz gerekiyor, ancak bunu şeiller II-13 ve II-14'te gösterilen gayri resmi bir tablo şecline yaparak başlıyoruz.

Şeillerde, "asn-val-...." gibi adlar, işlemeleri veya hataları tanımlamak için kullanılan ROSE (asında bir TAM SAYI veya NESNE TANIMLAYICI SEÇİMİ) tarafından tanımlanan bir türün ASN.1 değeri referans adlarıdır ve ve "ASN-type-...." gibi adlar, olası hatalarımızın her biri hakkında daha fazla ayrıntı taşıyan ASN.1 türleridir. "Sipariş numarası bilinmiyor" hatası durumunda, daha fazla bilgi döndürmemeye karar verdiği imizi ve tablonun ilgili hücresini boş bıraktığı imizi unutmamız. Bu durumda ASN.1 tipini NULL döndürmeye karar verebilirdik ama ROSE "ReturnError" SEQUENCE tipinde parametreyi taşıyan eleman OPSİ YONEL'dir ve tablomuzun hücresini boş bırakarak o elemanın olduğunu belirtiyoruz. "ReturnError" DİZİSİ bu durumda atlanaçaktır. Tablonun bir hücresini boş bırakmamıza izin verilmediğiini nasıl anladığımızı daha sonra göreceğiz.

Hata kodu =====	Parametre Tipi =====
asn-val-güvenlik hatası	ASN tipi saniye hatası ayrıntıları
asn-val-bilinmeyen-dal	ASN-tipi-dal-baş arısız-detayları
asn-val-unknown-order asn-val-unavailable	ASN-tipi-kullanılamıyor-detayları

Ş ekil II-13: wineco HATA tablosu

Ş ekil II-13 tablosunda olası her hata için bir satır ve yalnızca iki sütun vardır:

atanan hata kodları (ROSE spesifikasyonunda belirlenen tipteki değil erler olarak); ve
hatanın parametrelerini taşı ımak için ilgili ASN.1 tipi (modülüümüzde tanımlanmış tır).

Normalde protokolünü tanımlamak için ROSE kullanan herhangi bir uygulama için bu tablo için az sayıda satır bekleyebiliriz (bizim durumumuzda dört satırımız vardır) ve bazı hatalar için döndürülecek ek parametre bilgisi olmayabilir. ve bu nedenle, "asn-val-unknown-order" durumunda olduğ u gibi, bu hatanın parametreleri için ASN.1 türü gerekmek.

Ş ekil II-14'teki tablo, wineco uygulamamız için ROSE protokolünü tamamlamak için gereken diğ er bilgilerdir. Yine ROSE tarafından belirtildiğ i gibi - türünün bir değil eri olan bir iş lem kodunu listeler:

SEÇ M {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}

iş lem için giriş bağ ımsız değil iş kenlerini taşı ıyan ASN.1 türüyle birlikte, sonuç değil erlerini taşı ıyan ASN.1 türüyle birlikte iş lemin oluş turabileceğ i hataların bir listesiyle birlikte.

İ ş Lem Kodu =====	Argüman Türü =====	Sonuç Türü =====	Hatalar =====
asn-val-order-stok-iç in-sipariş		Sipariş onaylandı	güvenlik hatası bilinmeyen ş ube güvenlik hatası bilinmeyen
asn-val-sales Satış İ adesi		HÜKÜMSÜZ	ş ube
asn-val-query Sorğu-kullanılabilirlik Kullanılabilirlik-Yanıt güvenlik hatası			bilinmeyen ş ube kullanılamıyor güvenlik
asn-val-state İ stek-sipariş -durumu Sipariş -durumu			hatası bilinmeyen ş ube bilinmeyen sıra

Ş ekil II-14: wineco İ Ş LEM tablosu

Gerçek ROSE belirtiminde, iş lemler ve hata dönüş leri için bir öncelik değil eri atamak, söze "bağ lantılı iş lemleri" tanımlamak ve sonuçların her zaman döndürülmüş döndürülmediği ini, hata parametrelerinin değil erlerinin gereklili olup olmadığı ini vb. belirlemek için ek sütunlar vardır.. ROSE'un bu ayrıntılarının tartışılması, bu metnin kapsamını veya ihtiyaçlarını aş acaktır ve bu özellikleri resme dahil etmediğ.

İçerinde "delikler" olan mesajlar (ASN.1 veri türleri) ile ilgili ROSE konsepti göz önüne alındığında, şunu görüyoruz:

Bir dizi işlem ve hatanın (yukarıdaki tabloların sayı ve biçiminin tanımı) belirtimi ile ROSE veri tiplerini tamamlamak için kullanıcının sağlama gereken bilgileri belirtmek için ROSE için bir sözdizimine duyulan ihtiyaç.

ROSE kullanıcının şekilleri II-13 ve II-14'te gayri resmi olarak gösterilen bilgileri belirtmeleri için katı bir ASN.1 sözdizimine (makine tarafından okunabilir) duyulan ihtiyaç.

ASN.1 tiplerindeki "boş lukları" tanımlamak ve şekil II-13 ve II-14'te gösterilen bilgileri, tamamlanması amaçlanan "delik" ile açıkça ilişkilendirmek için ASN.1'deki notasyon ihtiyacı.

3.1 Özelden genele

Genel durumda, herhangi bir "jenerik" protokolü tamamlamak için gereken birçok farklı tablo olabilir ve her tablonun o "jenerik" protokol tarafından belirlenen bir dizi sütunu olacaktır.

Tablonun her bir sütunu için gereken bilgilerin doğası (ve her bir bilgi parçası için bir "tutma yeri" sağlayan sütun başlıklar), söz konusu "jenerik" protokole bağlı olarak değil işe ecektir.

ROSE, tamamlanmamış (jenerik) protokollerin yalnızca bir örneğidir. Belirteçlerin, belirtimi tamamlamayı baş kalarına bıraktığı ve hangi ek bilgilere ihtiyaç duyulduğunu (resmi olarak söyleyebilmesi gereken birçok başka örnek vardır. Bu bir Bilgi Nesnesi Sınıfı belirtimidir.

Bu nedenle, bir "jenerik" protokolün belirleyicisi, tabloların biçiminin açık bir ifadesini ("jenerik" protokolü tamamlamak için gereken bilgi) sağlayacak bir gösterime ihtiyaç duyar. Bunun belirtimine Bilgi Nesnesi Sınıflarının belirtimi diyoruz. "Genel" protokolün bir kullanıcısı bir tablo satırı için bilgi sağladığında, o tabloyla ilişkili sınıfın bir Bilgi Nesnesini belirttiğini söylez. Herhangi bir kullanıcı belirtimini desteklemek için tanımlanan belirli bir tablonun toplam satır kümnesine Bilgi Nesnesi Kümesi denir.

Bu nedenle ASN.1'de aşağıdaki idakiler için notasyon gereklidir:

Adlandırılmış bir Bilgi Nesnesi Sınıfının tanımı (bir tablonun biçimini).

Belirli bir sınıfın adlandırılmış Bilgi Nesnelerinin tanımı (tablonun bir satırı için bilgileri tamamlama).

Bir belirtimde tanımlanan tüm Bilgi Nesnelerini (herhangi bir sınıfından) adlandırılmış bir Bilgi Nesnesi Kümesine (tamamlanmış bir tablo) toplamak.

Adlandırılmış bir bilgi nesnesi kümnesini, tamamlamak için tasarlandığı taşıyıcı protokolündeki "deliklere" bağlı lama.

4 Tablolardan Bilgi Nesnesi Sınıflarına

Tablo metaforu, Bilgi Nesnesi Sınıfı kavramlarının tanıtılmasında çok kullanışlıdır, ancak "tablo" terimi ASN.1 Standardının kendisinde kullanılmaz (daha sonra tartışılacak olan "tablo kısıtlaması" terimi dışındadır).

Tablolar, insandan insana iletişim için uygunlardır. Bilgisayar iş leme için, tabloların biçimini ve bu tabloların içeriğini tanımlamak için ASN.1 gösterimini kullanız.

Her Bilgi Nesnesinin, her biri bir alan adına sahip bir dizi alanı olduğunu söyleyorum. Bir Bilgi Nesnesi Sınıfı tanımlama, o sınıfındaki nesneler için tüm alanların listelenmesini, her alan için alan adının ve o alanın bazı özelliklerinin verilmesini içerir. En önemli özellik, o alanı tanımlarken ihtiyaç duyulan bilginin doğasıdır. Bu, genellikle bazı ASN.1 türlerinin belirtimidir (bu türle ilişkili semantiklerle birlikte) veya bazı sabit ASN.1 türünün ASN.1 değilinin belirtimidir. Bununla birlikte, tanımlanabilecek bir dizi başlangıç alan türü olduğunu daha sonra göreceğiz.

ROSE durumunda, ROSE tarafından tanımlanan iki Bilgi Nesnesi Sınıfımız vardır, OPERATION sınıfı ve ERROR sınıfı. (Bilgi Nesnesi Sınıflarının adlarının tamamen büyük olması gereklidir).

OPERATION sınıfındaki tüm nesnelerin aşağıdaki idakileri içeren dört alanı olacaktır:

türünde bir değil er

SEÇ M {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}

operasyonu tanımlamak için.

İşlem için girdi değil erlerini taşıyabilen bir ASN.1 tipi.

İşlemi başlangıçla tamamlanması üzerine sonuç değil erlerini taşıyabilen bir ASN.1 tipi.

Her biri bu özel işlemi üretebileceğini bir hata olan ERROR sınıfı bilgi nesnelerinin bir listesi.

ERROR sınıfındaki tüm nesnelerin aşağıdaki idakileri içeren iki alanı olacaktır:

türünde bir değil er

SEÇ M {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}

Hatayı tanımlamak için.

Hata parametrelerinin değil erlerini taşıyabilen bir ASN.1 tipi.

Özetlemek gerekirse: Bir Bilgi Nesnesi Sınıfı tanımı, o sınıfın bir nesnesini belirtmek için gereken bilginin miktarını ve biçimini tanımlar. Bir Bilgi Nesnesi tanımı bu bilgiyi sağlar. İş ihtiyaç duyulan bilginin doğası çok çeşitlilikte olabilir ve bir Bilgi Nesnesi tanımlarken o alan için ihtiyaç duyulan bilgilere göre Bilgi Nesnesi Sınıfının alanlarının biçiminden bahsediyoruz.

Yukarıdaki tartışmada şunları tanıttık:

tip alanları: Tamamlanması için bir ASN.1 tip tanımına ihtiyaç duyan alanlardır.

sabit tip değil er alanları: Tamamlanması için tek (belirtilen) bir ASN.1 tipinin değil erine ihtiyaç duyan alanlardır.

nesne seti alanları: Tek bir (belirtilen) bir bilgi nesnesi kümesine ihtiyaç duyan alanlar
Bunları tamamlamak için Bilgi Nesnesi Sınıfı (bu durumda ERROR sınıfı).

Bir Bilgi Nesnesi Sınıfı tanımlanırken belirtilebilecek bir dizi başka alan birimi vardır ve bunlardan daha fazlasını ileride göreceğiz.

Adları tamamı büyük harflerle görürseniz, Bilgi Nesnesi Sınıflarıyla uşraq tıgınızdan makul ölçüde emin olabilirsiniz, ancak bunu anlamanın başka bir kesin yolu da & (ve işaret) karakteriyle başlayan adların varlığıdır. ASN.1 göstergesinin diğeri parçalarıyla karıştırılmaması için, Bilgi Nesnesi Sınıflarının alan adlarının & ile başlaması gereklidir. Bu nedenle, belirli bir işlem için nesne tanımlayıcı değil erini içeren OPERATION sınıfının alanı şunlardır:

İşlem Kodu

Invoke mesajının argümanları için bir tür tanımı ile sağlanması gereken alan şunlardır:

İşlem Türü

& İşlem Kodu alanının tek bir ASN.1 değil eri içerdigine ve & den sonra bir küçük harfe (bu bir gereklilik) sahipken, & ArgumentType alanının bir ASN.1 tipine ve & den sonra bir üst harfe sahip olduğunu dikkat edin. -vaka mektubu (yine bir gereklilik). Bir alanın tek bir değil er (genellikle - ancak her zaman değil il - bazı sabit türde) veya tek bir bilgi nesnesi (bazı sabit sınıflardan) içerdigini durumlarda, & işaretinden sonraki alan adı küçük harfle başlar. Bir alanın birden çok değil er veya birden çok bilgi nesnesi içerdigini durumlarda (bir işlem için hata listesinde olduğunu gibi), & işaretinden sonraki alan adı büyük harfle başlar. Bir ASN.1 Bilgi Nesnesi Sınıfı tanımının anlamını yorumlamaya çalışırken bu kuralları hatırlamak önemlidir.

Bilgi Nesnesi Sınıflarının adlarının tamamen büyük olması gerektiğiini daha önce görmüşük.

Bireysel Bilgi Nesnelerine verilen adların küçük harfle başlaması (değil er referanslarına benzer) ve Bilgi Nesnesi Kümelerine (belirli bir sınıfın Bilgi Nesneleri koleksiyonları) verilen adların büyük harfle başlaması gereklidir.

Genel olarak türler, değil erler ve değil er kümeleri (alt türler) kavramları ile Bilgi Nesnesi Sınıfları, Bilgi Nesneleri ve Bilgi Nesnesi Kümeleri kavramları ve adların ilk harfine ilişkili adlandırma kuralları arasında güçlü bir benzerlik vardır. Aynı kuralları takip edin.

Bununla birlikte, tipler ve bilgi nesnesi sınıfları arasında önemli bir fark vardır. Tüm ASN.1 türleri, bir dizi değil erle doldurulmuş hayatın başı ve bu değil erlerin alt kümeleri olarak yeni türler üretilebilir. Bilgi Nesnesi Sınıflarının önceden tanımlanmış nesneleri yoktur, yalnızca, daha sonra gerçek türlerin eş değil er olan bilgi nesnesi kümelerinde toplanabilecek olan, o sınıfın nesnelerini tanımlamak için notasyonu belirlerler.

Bir sınıf tanımladığınızda, ona bir referans adı sağlarsınız ve benzer şunlardır: Bilgi Nesneleri ve Bilgi Nesnesi Setleri için. Bu referans adları daha sonra ASN.1 notasyonunun diğeri bütümlerde bu sınıflara, nesneler ve kümelere, tıpkı tip referansı ve değil er gibi referans vermek için kullanılabilir.

referans adları, tür ve değer er tanımlarına atanır ve daha sonra baş ka yerlerde kullanılır. Sınıflar, nesneler ve nesne kümeleri için referans adları, típki tür ve değer er referans adları gibi, IMPORTS ve EXPORTS deyimlerindeki modüller arasında içe ve dış a aktarılır.

5 GÜL İ Ş LEMİ ve HATA Nesne Sınıfı tanımları

Ş ekil II-15, ROSE'un İ Ş LEM ve HATA sınıflarının tanımının basitleş tirilmiş bir biçimini gösterir ve Bilgi Nesnesi Sınıflarını tanımlamak için gerçek ASN.1 sözdiziminin ilk tanıtımıdır.

Unutmayın, bu sözdizimi esas olarak II-13 ve II-14'te gösterilen informal tabloların tablo başlıklarını ve bilgi içeriğiini tanımlamaktır, ancak bunu ASN.1 tip ve değer er tanımlama sözdizimine benzer bir sözdizimiyle yapmakta ve tamamen makinede işlenebilir.

Sonunda! Gerçek bir Bilgi Nesnesi Sınıfı tanımının bir örneğini incelediğimizde, ROSE'dan OPERATION sınıfı ve ERROR sınıfı.

```

İ Ş LEM ::= SINIF
  {&işlem Kodu SEÇİMİ {yerel TAM SAYI,
    küresel NESNE TANIMLAYICISI}
    BENZERSİ Z,
    &ArgümanTürü,
    &SonuçTürü,
    &Hatalar HATA İSTEĞE BAĞLI }

HATA ::= SINIF {&hataKodu
  SEÇİM {yerel TAM SAYI, global NESNE TANIMLAYICISI}
  BENZERSİ Z,
  &ParametreTürü İSTEĞE BAĞLI }

```

Ş ekil II-15: İ Ş LEM ve HATA sınıfı tanımları

Ş ekil II-15'te beklenediği gibi İ Ş LEM için dört, HATA için iki alan tanımını görüyoruz. Bu rakamı ş ekil II-13 ve II-14'ün tablo başlıklarıyla karşılaştırın ve alanları ayrıntılı olarak inceleyelim. (Unutmayın, her sınıf tanımı bir tablonun biçiminin tanımına ve her alan o tablonun bir sütununun biçiminin tanımına karşı ilişkili gelir.)

OPERATION sınıfı için belirtilen tipte bir değer er ile doldurulması gereken "&operationCode" alanımız mevcuttur. (Sabit tip değer er alanı olarak adlandırılır). Bu alan ayrıca "BENZERSİ Z" olarak işaretlenmiş tir. Bu sınıfın bir nesnesini tanımlarken, bu alana herhangi bir değer er (belirtilen türden) girilebilir, ancak bu tür nesnelerin bir kümlesi bir Bilgi Nesnesi Kümesi oluşturmak üzere bir araya getirilirse (daha sonra görecek imiz notasyonu kullanarak), ("BENZERSİ Z" nedeniyle) bu alandaki tüm değer erlerin kümedeki her nesne için farklı olması gerekliliği. Nesne kümelerinin tamamen doldurulmuş bir tabloyu temsil ettiğini düşüneyiniz, veritabanı terminolojisinde "BENZERSİ Z" olarak işaretlenmiş alanlar tabloya bir anahtar veya dizin sağlar. Birden fazla alan "BENZERSİ Z" olarak işaretlenebilir (ancak bu yaygın değil), ancak notasyonda, bir bilgi nesnesi kümesi içinde iki alanın birleşiminin benzersiz olmasını gerektirecek bir mekanizma yoktur. Bunu belirtmeniz gereklidir, sınıf tanımı içinde yorum kullanmanız gereklidir.

Sonraki iki alan, "&ArgumentType" ve "&ResultType", büyük harfle başlayan adlara sahiptir ve bunlardan sonra tür tanımı yoktur. Bu, bir ASN.1 tipinin belirtimi ile tamamlanmaları gerektiği anlamına gelir (zorunlu olmamakla birlikte genellikle, bir tipin açık bir tanımı yerine bir tip referansı verilerek).

Dördüncü ve son alan daha ilgi çekicidir. "&Hatalar" büyük harfle başlar, bu nedenle onu bir dizi şeyle tamamlarsınız. Ancak aşağıdaki isim bir ASN.1 tipi referansı değil, bir sınıf referansıdır. Dolayısıyla bu alan, daha sonra tanımlanan o (ERROR) sınıfından bir dizi Bilgi Nesnesi ile doldurulmayı gerektirir. Bu alan ayrıca "İSTEŞİ BAĞLI" olarak işaretlenir. Bu, bu sınıfın nesnelerin tanımında, bu alan için bilgi tanımlamanın gerekli olmadığı anlamına gelir - boş bırakılabilir. Bu, karşılık gelen işlem için hiçbir zaman bir "ReturnError" yanıtı üretmediğine anlamına gelir.

Artık anlaşılması gereken error sınıfının tanımını incelemek okuyucuya bırakılmıştır.

6 Bilgi Nesnelerini Tanımlama

Şimdi tanımlı bir sınıfın nesnelerini tanımlamak için gösterimi kullanalım (bu durumda OPERATION ve ERROR). Şekiller II-13 ve II-14'te verilen işlemelerin ve hataların resmi olmayan tanımını alıyoruz ve nesneleri tanımlamak için ASN.1 notasyonunda ifade ediyoruz. Bu, şekil II-16 (HATA nesneleri) ve II-17'de (İşlem nesneleri) gösterilmişdir.

Bilgi Nesnesi Sınıfı tanımı, size bir OPERATION veya ERROR nesnesi tanımlamak için hangi bilgileri sağlanmanız gerektiğiini söyler.
Şimdi, bu tür nesneleri tanımlamak için kullandığınız sözdizimini görüyoruz.

```

sec-fail HATA ::= {&errorCode
asn-val-security-failure,
&ParameterType ASN-type-sec-failure-details}

bilinmeyen dal HATA ::= {&hataKodu asn-
val-bilinmeyen dal,
&ParameterType ASN-type-branch-fail-details}

bilinmeyen sıra HATA ::= {&errorCode
asn-val-unknown-order}

kullanılamıyor HATA ::= {&hataKodu
asn-değер-kullanılamıyor,
&ParameterType ASN-type-unavailable-details}

```

Şekil II-16: wineco ERROR Bilgi Nesnelerinin Tanımı

Bu rakamlar oldukça anlaşılır olmalı ve satır satır yorum yapılmayacaktır ancak okuyucunun dikkatini çeken bazı noktalar vardır.

"::=" işinin solu daha çok bir deger referansının tanımına benziyor - karşılık tırın:

```
my-int-val INTEGER ::= 3
```

"INTEGER türündeki my-int-val değer eri 3'tür" olarak okunur. Benzer şekilde, II-16 ve II-17 şebeke ekillerini (örneğin) "ERROR sınıfının sec-fail'i ..." olarak okuruz. ":" ifadesinden sonra, sınıf tanımındaki alanların her birini sırayla ve virgülle ayırarak listeleriz (küme parantez içinde), her durumda alanın adını ve bu belirli nesne için o alanın tanımını veririz. .

Ayrıca "unknown-order" ERROR nesnesinin &ParameterType alanı için bir tanımı olmadığı için dikkat edin - buna yalnızca bu alan şebeke II-15'in sınıf tanımında İSTEĞE BAŞLI olarak işaretlendiği için izin verilir.

"&Hatalar" alanına dönersek, bir hata seti tanımlamak istediği imizde, dikey bir çubukla ayrılmış ve kıvrımlı parantez içinde alınmış bir referans adları listesi kullandığıımızı unutmayın. Bu, liste ayırcı olarak virgül kullanılmamasından daha az sezgisel görünebilir, ancak aslında küme aritmetiği kullanılarak kümeler halinde gruplandırmak için çok daha güçlü bir mekanizmanın özel bir durumudur (aşağıya bakın). Dikey çubuk Bİ RLEŞTİ RME için kullanılıyor, bu nedenle "güvenlik-arızası" ve "unknown-branch" birleşimi olan "order"ın "&Error" alanına bir set üretiyoruz.

Son olarak, "&Error" alanlarının tanımında kullanılan isimlerin kendilerinin şebeke II-16'da hata olarak tanımladığı için dikkat edin. Bu tanımlar, şebeke II-17 tanımlarıyla aynı modülde olacak veya bu modüle aktarılacaktır.

```
İ $ LEM sipariş i ::=  
{&iş İemKodu asn-val-order,  
  &ArgümanTürü  
  &SonuçTürü  
  &Hatalar  
  Stok için sipariş,  
  Sipariş onaylandı, {güvenlik  
  hatası | bilinmeyen şube } }  
  
satış İ $ LEM ::=  
{&iş İemKodu asn-val-satış,  
  &ArgümanTürü  
  &SonuçTürü  
  &Hatalar  
  Satışların iadesi,  
  NULL,  
  {güvenlik hatası | bilinmeyen  
  şube } }  
  
sorgu İ $ LEM ::= {&iş İemKodu  
  asn-val-sorgu,  
  &ArgümanTürü  
  &SonuçTürü  
  &Hatalar  
  Soru kullanılabilirliği,  
  Kullanılabilirlik-Yanıt,  
  {güvenlik hatası | bilinmeyen şube | kullanım dışı } }  
  
durum İ $ LEM ::= {&iş İemKodu  
  asn-val-durum,  
  &ArgümanTürü  
  &SonuçTürü  
  &Hatalar  
  İstek-sipariş-durumu,  
  Sipariş durumu,  
  {güvenlik hatası | bilinmeyen  
  şube | bilinmeyen sıra } }
```

Şebeke II-17: Wineco OPERATION Bilgi Nesnelerinin Tanımı

Ş ekil II-16 ve II-17 tanımları, Ş ekil II-13 ve II-14'te kullanılan gayri resmi tablo notasyonundan daha ayrıntılı görünebilir (öyledir!), ancak bunlar çok açıktır, ancak daha da önemlisi makine tarafından okunabilir. , ve ASN.1 araçları bunları işleyebilir ve bu tanımları gelen mesajlardaki "deliklerin" içeriğini kontrol etme ve özmede kullanabilir.

7 Bir Bilgi Nesnesi Kümesi Tanımlama

Bireysel Bilgi Nesnelerinin tanımını neden bir Bilgi Nesnesi Setinde birleşmemiz gerekiyor? Yukarıdaki OPERATION sınıfının "&Errors" alanını tanımlarken bunun bir kullanımını gördük, ancak daha önemli bir neden var. Bilgi Nesnesi Sınıflarını ve Bilgi Nesnelerini tanımlamanın tüm amacı, bir taş iyiçi veya genel protokoldeki boş lukları neyin doldurabileceğini belirleyen ve bu ASN'yi bağlayan, daha önce gördüğümüz tam (resmi olmayan) tablonun bir ASN.1 tanımını sağlamaktır. Jenerik veya taş iyiçi protokoldeki "deliklere" 1 tanım.

Yolda bir sonraki adım. Birisi bazı Bilgi Nesnesi Sınıfları tanımlamıştır. Bazı Bilgi Nesneleri tanımlıyoruz. Şimdi onları adlandırılmış bir Bilgi Nesnesi Kümesine çekiyoruz.

Bu nedenle, Bilgi Nesnesi Kümelerini (belirli bir sınıfın Bilgi Nesneleri koleksiyonları) tanımlamamıza izin verecek bir notasyona ihtiyacımız var; bu kümeye, belirtimizin başka bir yerinde kullanılabilen bir ad atanmıştır.

Bilgi Nesnesi Kümeleri, türlerin değil kümeleri veya kümeleri olarak görülebilmesi gibi, Bilgi Nesneleri koleksiyonlarındır. Bu nedenle, Bilgi Nesnesi Setlerinin adlarının büyük harfle başlamasının gerekliliği olmasının aksı irticî değil ildir. Ş ekil II-17'de tanımladığımız işlemeler koleksiyonu için bir isim istiyorsak şunu yazabiliriz:

My-ops İ Ş LEM ::= {sipariş | satış | soru | durum }

Bunu "OPERATION sınıfındaki My-ops, sipariş, satış, soru ve durum nesnelerinin birleşimiinden oluşan kümeler" şeklinde okuyun.

Bu en yaygın biçimdir, ancak gerekirse genel kümeleri arıtmayı kullanabilir. A1, A2, A3 ve A4'ün OPERATION sınıfının Bilgi Nesnesi Kümeleri olarak tanımladığını varsayıyalım. Aşağıdaki gibi ifadeler yazabiliriz:

Yeni Küme İ Ş LEMİ ::= {(A1 KESİ Ş İ M A2)
Bİ RLEŞ Tİ RME (A3 HARİ ÇA4)}

ama bir meslektaşımın sık sık söylediğimi gibi: "Hiç kimse yapmaz!"

Köşeli parantezleri dışarıda bırakırsanız, en bağlayıcı EXCEPT, sonraki KESİ Ş İ M ve en zayıf Bİ RLEŞ İ M'dir. Bu nedenle, yukarıdaki tüm yuvarlak parantezler, anlamda işıklı olmadan çıkarılabilir, ancak okuyucunun kafasını karıştırmamak için genellikle bunları dahil etmek en iyisidir. (Bazı insanlar "EXCEPT"in en az bağlayıcı olması gerektiğiini sezgisel buluyor gibi görülmektedir, bu nedenle "EXCEPT" kullanıldığıında parantezleri netleşirmek her zaman iyi bir fikirdir.)

A1'den A4'e kadar çes itli kümelerin sonucunun uzun soluklu bir öneğ iyle sizi sıkmayacağ im - kendinizinkini icat edip üzerinde çalış in - veya genç kızınızdan size yardım etmesini istemeyeceğ im!

Ş apka karakteri "^", "KESİ Ş İ M" ile eş anlamlıdır ve dikey çubuk karakteri "|" "Bİ RLİ K" ile eş anlamlıdır. HARİ Ç ile eş anlamlı olan tek bir karakter yoktur - bunu tam olarak yazmalısınız.

Bilgi Nesneleri ve değ erler ile Bilgi Nesnesi Kümeleri ve türleri veya alt türleri (değ er koleksiyonları) arasındaki benzerlik i daha önce belirtmiş tik. Sınıflar bu modelin neresine oturur?

Bu daha az kesin. Bilgi Nesnesi Sınıfları bazı yönlerden türlere benzer, ancak türlerin aksine, içlerinde Bilgi Nesnesi olmadan baş larlar, yalnızca ASN.1 kullanıcısının o sınıfın nesnelerini tanımlaması için bir mekanizma ile baş larlar. Buna karş ilk, yerleş ik türler, kısıtlamaları kullanarak alt kümeler oluş turabileceğ iniz hazır bir değ er koleksiyonu ve değ er notasyonu ile birlikte gelir.

Bununla birlikte, nesnelerin ve değ erlerin benzerlik i nedeniyle, ASN.1 bilgi nesnesi ile ilgili kavramları tanıtmak için geniş letildiğ inde, nesne kümelerini tanımlamak için tanıtılan sözdiziminin aynısının nesne kümelerini tanımlamak için kullanılmasına izin verilmesine karar verildi. değ erler (bir tür altkümeler).

Bu nedenle, sözde değ er kümesi ataması ASN.1 sözdizimine dahil edildi. Bu, yazmanıza izin verir (eğ er isterseniz!):

```

İ Ik küme INTEGER ::= {0..5}
İ kinci küme INTEGER ::= {10..15 Bİ RLEŞ İ M 20}
Üçüncü küme TAM SAYI ::= {Birinci Küme
    Bİ RLEŞ İ M İ kinci küme 13 HARİ Q
    Dördüncü küme TAM SAYI ::= {0..5 | 10..12 | 14 | 15 | 20}
}

```

"Dördüncü küme", elbette, "Üçüncü küme" ile tam olarak INTEGER alt kümesidir.

Test zamanı! Ya da baş ka bir deyiş le, biraz eğ lence zamanı! Yukarıdaki tanımlarla yazabilir miyim

```
seçilmiş -int Dördüncü set ::= 14
```

ve bir Dİ Zİ 'nin bir ej esı olarak

```
Üçüncü küme VARSAYILAN seçili-int
```

Evet yapabilirsin! ASN.1'in bu gibi durumlarda neyin yasal olduğunu *tam olarak* sorusu, Standartlar grubunu yillardır rahatsız ediyor, ancak ş imdi büyük ölçüde çözüldü. Bununla birlikte, Standart metnin kendisini incelemek yerine, size yanıt vermesi için iyi bir araca güvenmek en iyisidir! Veya belki de ASN.1'inizi basit ve anlaşılır tutmak daha da iyidir!

Bu alt maddeden ayrılmadan önce, "İ ş lemlerim" konusuna tekrar bakalım. Wineco protokolünün gelecekteki bir sürümünde, bazı ek iş lemler eklemek ve dolayısıyla "İ ş lemlerim"i geniş letmek isteyeceğ imiz muhtemeldir. Bunun, hakkında hiçbir ş ey bilmediği bir iş lemi gerçekleştirmeleri istendiğ inde bazı tanımlanmış hata iş lemeye ihtiyaç duyacak olan sürüm 1 sistemleri için etkileri vardır. Hata iş lemenin nasıl belirlendiğ ini birazdan göreceğ iz, ancak önce "My-ops" un gelecekte geniş letilebileceğ ini belirtmemiz gerekiyor. Bunu ş u ş ekilde yeniden yazarak yapıyoruz:

```
My-ops İ Ş LEM ::= {sipariş | satış | sorğu | durum, ...}
```

olası bir sürüm 2 ile, eklenen bir "ödeme" iş lemiyle birlikte:

```
My-ops İ § LEM ::= {sipariş | satış | soru |
durum, ..., ödeme }
```

8 ROSE protokolünü tamamlamak için bilgileri kullanma

Ana temamıza geri dönelim. "Genel" protokol tasarımcıları, tanımlamadıkları SEQUENCES ve SET ögelerine sahip olmak isterler.

Diğer grupların bu pozisyonları dolduracak türleri tanımlamasını istiyorlar. Sık sık, diğer gruplar bu elementlerde farklı zamanlarda birçok farklı tip taşılmak isteyeceklerdir. Bilgi Nesnesi kavramları, bu ögeler dolduracak türlerin tanımlanmasını sağlar. Ancak bu "delikler" bir ASN.1 tipi tanımında nasıl tanımlanır?

Ve Bilgi Nesnesi (Küme) tanımları "delikler" ile nasıl bağlı lantılıdır?

Bir yere gitmedikçe sınıfları, nesneleri ve nesne kümelerini tanımlamanın bir anlamı yoktur. olsa, onları kodlayıp hatta gönderemezsiniz. Peki onlar ne işe yarar? Cevap: Boş lukları doldurma!

ASN.1, büyük ölçüde tarihsel nedenlerden dolayı bu soruna üç aşamalı bir yaklaşım benimsiyor. İlk adım, bir ASN.1 türünün (veya bazı durumlarda bir ASN.1 değilinin) gerekli olduğu her yerde kullanılacak bir Bilgi Nesnesi Sınıfı alanına başvuruya izin vermektedir. İlkinci aşama, bir Bilgi Nesnesi Kümesinin bu tipler üzerinde bir kısıtlama olarak kullanılmasına izin vermektedir, bu ögelerin Bilgi Nesnesi Kümesinin karşılık gelen alanından bir tür (veya bir değil er) olmasını gerektirir. Buna tablo kısıtlaması denir. Üçüncü adım, (ilate olarak) bir SET veya SEQUENCE'in (aynı Bilgi Nesnesi Sınıfının alanları olarak tanımlanan) iki veya daha fazla ögelerinin aralarında bir işaretçi kullanılarak bağlılanmasına izin vermektedir ("@" simgesi bağlı lanti). Bu bağlılama mekanizmasının kullanımı, bağlı lantılı alanların kısıtlayıcı Bilgi Nesnesi Kümesinin bazı Bilgi Nesnelerine göre tutarlı bir şekilde doldurulması gerektiğiini söyler. Başka bir deyişle, bağlı lantılı alanların, tanımlayıcı tablonun tek bir satırındaki hücrelere karşılık gelmesi gereklidir. Diğerlerarasındaki bağlı lantayı ifade eden kısıtlamalara ilişkisel kısıtlamalar denir.

Şekil II-18, bu özellikleri gösteren (basitleştilmiş) bir ROSE "Invoke" veri tipini göstermektedir. Yukarıda tabloda tanımlanan "My-ops" (OPERATION sınıfından) Bilgi Nesnesi Kümesini ve "Invoke" ögeleri üzerindeki ilişkisel kısıtlamaları kullanır.

```
Çağırır ::= SIRALI
{ invokeId INTEGER, işlem kodu
OPERATION.&operationCode ({My-ops} !
invoke-unrecognisedOperation), argüman OPERATION.&ArgumentType ({My-
ops} {@opcode} ! invoke-mistypedArgument) İSTEKE BAĞLI }
```

Şekil II-18: ROSE Invoke veri türü

Şekil 18 oldukça karmaşık! Her seferinde bir adım atın. Dizinin "opcode" elemanı, "İ § LEM" sınıfının "&işlem Kodu" alanından bir değil er olduğuunu söyler. Kendi içinde, bu sadece şunuunla eş anlamlıdır:

SEQ M
(yerel INTEGER, global
NESNE TANIMLAYICISI)

çünkü bu, bu türden sabit türde bir dē er alanıdır. Ya da baş ka bir deyī le, bu alanın tüm dē erleri bu türdendir.

Bununla birlikte, Bilgi Nesnesi Sınıfı alanı aracılığ ıla türü referans alarak, o sınıfın bir Bilgi Nesnesi Kümesi ("My-ops") ile sınırlandırmamıza izin verilir. (Ȫ eyi basitçe "CHOICE ... etc" olarak yazsaydık böyle bir kısıtlamaya izin verilmezdi.)

"My-ops" etrafındaki süslü parantezler, ASN.1 sözdizimindeki bir aptallıktır (üzgünüm - birkaç tane var!).

Buradaki gereklik, "ObjectSet" sözdizimsel yapısı içindir. Bir nesne kümesi için referans adına ("İ ş lemelerim" böyle olur) izin verilmez. Ancak, "My-ops" ȫ esini bir nesne kümesi tanımına aktararak, yani kıvrık parantezler içine alarak "My-ops"tan bir "ObjectSet" oluş turabiliriz.

Basitçe söylemek gerekirse, bunun için iyi bir sebep yok, ama süslü parantezleri koymak zorundasın!

"My-ops" kısıtlamasının etkisi, bu ȫ e için izin verilen tek dē erlerin "My-ops" Bilgi Nesnelerinden biri olan "&operationCode" alanına atanın dē erler olduğuunu söylemektedir. Baş ka bir deyī le, alan, wineco için tanımlanan dört (sürüm 1'de) iş lemden biri için bir iş lem kodu içermelidir. Bunların tamamı tamamen makine tarafından okunabilir ve kodlayıcılar/kod çözücüler, hata kontrolüne yardımcı olması için bu belirtimi kullanabilir.

"!" bir istisna belirtimi sunar ve bu kısıtlama karşı ılanmazsa (farklı bir iş lem kodu dē eri görünür), hata iş lemenin "invoke-unrecognisedOperation" tamsayı dē eriyle bir REJECT döndürmek olduğuunu söyler. Wineco protokolünün tasarımcılarının bu tür hata iş lemeyi belirtmekle ilgilenmeleri gerekmekz. Bunların hepsi ROSE spesifikasyonu dahilinde yapıılır.

Bunun, bir sürüm 1 uygulamasına "ödeme" iş lemeyi gerçekleştirmek için bir istek gelmesi durumunda ortaya çıkacak hata durumunun tam olarak bu olduğu una dikkat edin.

Şimdi "argüman" ȫ esine geçiyoruz. Bu gerçek "delik". Kısıtlanmamış biçiminde, basitçe bu ȫ enin "herhangi bir ASN.1 türü" olabileceğ ini söyler (çünkü OPERATION sınıfının Bilgi Nesnesinin bu alanı için herhangi bir ASN.1 türü kullanılabilir). Bu notasyon, ASN.1'de "Açık Tip" notasyonu olarak tanımlanır ve kodlama kuralları tarafından oldukça özel olarak işlenir.

Özellikle, kodlamaların, bir kod çözümünün, içinde hangi türün kodlandığı ini ayrıntılı olarak bilmeden önce bir açık tür kodlamaların sonunu bulmasını sağlama önēmlidir (SEQUENCE'in "opcode" ȫ esi, "argüman" ȫ esinden sonra yazılmış olabilir. - herhangi bir kısıtlama yoktur).

BER'de sorun yoktur - bir kodlamaların sonu, tüm ASN.1 BER kodlama türleri için her zaman "TLV"nin "L" alanı kullanılarak belirlenebilir. Ancak PER'de durum böyle dē il.

Bir kod çözümü, kodlanan türün ne olduğuunu bilmekçe, türün bir dē erinin kodlamasının sonunu bulamaz. Bu nedenle PER'de, açık bir türde her zaman fazladan bir "uzunluk" sarmalayıcı eklenir.

Bir yana, bazen insanların kasıtlı olarak bir ȫ eyi açık tür olarak tanımladığı ini (tipik olarak tek bir alana, bir tür alanına sahip bir sınıf kullanarak) ve ardından bu ȫ eyi tek bir tam tanımlı ASN.1 türü olacak ş ekilde kısıtladığı ini görsünüz. Bunun yegâne amacı, ek uzunluktaki sarıcıyı üretmektedir ve uygulama mimarisiyle ilgilidir. Bu tür yapılar, güvenlikle ilgili verileri kapsüleme için kullanılır; burada uygulama mimarisi, güvenlikle ilgili veriler hakkında ayrıntılı bilgiye sahip olmayan uygulamanın güvenli olmayan kısmıyla birlikte, güvenlik çekirdeğ ine kapsüllenmiş bir sekizli setini iletir. (Devlet Sāhıfesi - Ş ekil 999 - tekrar - siz

bu hükmün mantıklı olup olmadığı ina kendiniz karar vermelisiniz. Olur. En kötü ihtimalle, gereksiz bir uzunluk alanı anlamına gelir!).

Son olarak, "argüman"ın "@" kısmını ele alıyoruz. Bu, kısıtlamayı "bağ ımsız değil iş ken" ve "iş lem kodu" alanlarını birbirine bağlayan ve kısıtlama tablosunun bazı satırlarıyla tutarlı olmalarını gerektiren iliş kisel bir kısıtlamaya dönüş türür. (Hay aksi! Kısıtlayıcı Bilgi Nesnesi Kümesindeki bazı nesnelerle tutarlı olmak için - doğru terminoloji kullanalım!).

"@" Yapısı, "opcode" alanına (aynı zamanda veya onun yerine) eş it ş eklide ve aynı etkiyle yerleş tirilebilirdi. Resmi olarak söylenen tek ş ey, iki (ve daha fazla olabilir) bağ lantılı alanın kümedeki bir nesneye tutarlı olması gerekī idir. Elbette "OPERATION.&operationCode" un sınıf tanımında "UNIQUE" olarak tanımlanmış inı biliyoruz, bu nedenle Bilgi Nesnesi Kümesinde "Çalıştır" ȫğ esinin "opcode" alanındaki bir değil erle eş leş en en fazla bir nesne olacaktır. İ leti. Genel durumda, bu mutlaka doğru değil ildir ve tek gereklilik, bağ lantılı alanların değil erlerinin ve/veya türlerinin kısıtlayıcı nesne kümesindeki bilgi nesnelerinden en az biriyle tutarlı olmasıdır (en az bir sıra ile tutarlıdır). sınırlama tablosu).

Son olarak, "invoke-mistypedArgument" hata dönüş üne dikkat edin. BER'de, bir kodlamada çok fazla fazlalık vardır ve bir kodlamanın olması gerekī inı (veya olabileceğ inı) düş ündǖğümüz türden bir değil eri temsil etmemesi genellikle kolayca tespit edilebilir. PER'de, çok daha az gereksiz kodlama olduğ u için durum genellikle böyle değil ildir. PER'de, "invoke-mistypedArgument" ana algılaması, açık türün kodlamasının (eklenen uzunluk alanı tarafından belirlendīğ i ş eklide) eş leş tirmeye çalış tiğ imiz türün bazı değil erleri için doğru uzunluğ a sahip olmaması olacaktır ("iş lem kodu" değil eri ile tanımlanan).

Protokol tasarımcıları arasında, hatalı materyalin alınması durumunda bir uygulamanın eylemlerinin ne ölçüde belirtmesi gerekī i konusunda her zaman bir tartışma vardır (muhtemelen bir baskı gönderme uygulamasından veya alt katmanlarda tespit edilemeyen hataların çok çok nadir görülmesi nedeniyle), veya bu tür eylemlerin uygulamaya bağlı olarak bırakılması gerekip gerekmedī i. ASN.1, her iki yönde de gösterim sağlar. ROSE, hata iş leme konusunda çok kuralçı olmayı seçti ve "kötü" malzeme alındığında gerekli davranışı belirtmek için ASN.1 istisna iş lemeyi tam olarak kullandı. Bir protokol tasarımcısısanız, bu sizin almanız gereken bir karardır. ASN.1, kural koyucu olmanız için size araçlar sağlar, ancak bu araçları kullanma zorunluluğ u yoktur ve birçok belirtici kullanmamayı tercih eder.

"!" arasında belirli bir fark olduğunu unutmayın. opcode ȫğ esinde ve "argüman" ȫğ esinde. İ lk durumda, bir sürüm 2 sistemi bir sürüm 1 sisteminde "ödemeye" başlatmaya çalışır, bunun etkinleş tirilebileceğ inı biliyoruz. İ kinci durumda, sistemler uyumluysa ve alt katman iletişim imleri güvenilirse, asla etkinleş tirilmemelidir.

9 Parametreleş tirme ihtiyacı

Acaba kaç okuyucu, yukarıdakilerin çekici bir ş eklide kesin ve uygulanabilir görünse de, buradaki ana sorunu fark ettīğ inı fark etti?

Ama maalesef iş e yaramıyor!
Pek çok insan kendi "İ ş lemim" nesne kümelerini tanımlıyor, ancak "Çağırma"nın yalnızca bir ROSE özelliği var!

Wineco spesifikasyonumuzda ROSE'un tamamını

yeniden yazacak olsaydık, yukarıdakiler iyi çalışır. Daha önceki böümlerde gösterildiği gibi ana türlerimizi tanımlayan bir dizi modülüümüz olabilir (bunlara MAIN modüller diyoruz) ve OPERATION ve ERROR sınıflarını ve "Invoke", "Reject", "ReturnResult" ve "ReturnError" ȫğ erlerini tanımlayan baş ka bir modülüümüz olabilir. (buna ROSE modülü diyoruz). Ardından, bilgi nesnelerimizi ve "İ ş lemim" kümesini tanımlayan son bir modülüümüz (buna Bİ LGİ OBJELERİ modülü diyoruz) var.

MAIN'den tüm üst düzey wineco türlerimizi ihraç ediyoruz. ROSE modülünden Bilgi Nesnesi Sınıfı tanımlarımızı dışa aktarıyoruz. INFORMATION OBJECTS modülünde Information Object Class tanımlarını import ediyoruz ve "My-op"u export ediyoruz. Son olarak ROSE modülünde sınıf tanımlarını dışa aktarmanın yanı sıra yukarıda anlatıldığında gibi "Invoke" vb. mesajlarda kullanılmak üzere "My-op"u import ediyoruz ve artık wineco abstract söz dizimimizi tanımlayan üst düzey PDU'muzu şunşak ekilde tanımlıyoruz:

```
wineco-PDU ::= SEÇ M {çağır, Reddet, sonuç
Dönüş Sonucu,
```

```
hata          Dönüş Hatası }
```

Eksiksiz ve çalışan bir protokolümüz var.

Ancak, ROSE spesifikasyonlarının wineco spesifikasyonundan tamamen ayrı olarak yayına olmasını istiyorsak ve ROSE tabanlı bir spesifikasyon üretmek isteyen birçok farklı uygulama (bunlardan sadece biri wineco olacaktır) istiyorsak bu yaklaşım işe yaramaz. Her uygulama için ROSE metnini kopyalamak iyi bir fikir olmaz! (Bununla birlikte, kendi ROSE eşdeğer sınıflarını ve PDU'larını genellikle basitleştirilmiş bir biçimde tanımlayan spesifikasyonlar vardır, çünkü bunlar kendi başlarına eksiksiz olmak ve ROSE parçasının ayaklarının altında değişmemesi için kontrole sahip olmak isterler. Bu "basitleştirmeyle kopyalama", yalnızca ROSE'da değil, diğer popüler belirtimlerde de olur.)

ROSE spesifikasyonu wineco uygulamasından bağımsız olacaksa, o zaman açıkça "My-op" tipini içe aktaramaz. O halde, boş luğunu nasıl doldurulacağıını söylemek için nasıl bir kısıtlama sağlayabilir?

Burada yeni ve çok güçlü bir ASN.1 konseptini, parametrelendirmeyi tanıiyoruz.

Tüm programcılar, işlev veya alt program veya yöntem belirtiminin gölgesinde atıfta bulunulan bir dizi sahte parametreye sahip işlevler veya alt programlar veya yöntemler kavramına tamamen aşınadır. Bu işlevler veya alt programlar çağırıldıında, çağırılan kod, o çağırılış için sahte parametreler yerine kullanılan bir dizi gerçek parametre sağlar.

ASN.1 çok benzer bir konsepte sahiptir. ROSE "Invoke" tipi gibi bir tip tanımladığımızda, tip adından sonra bir kukla parametre listesi listeleyebiliriz. Bu sahte parametreler daha sonra tip tanımının sağ tarafında normal referans adları gibi kullanılabilir. Böyle bir tür parametreler tirilmiş tür diyoruz ve parametreler tirilmiş türleri dışa aktarabiliyoruz (örneğin genel ROSE belirtiminden, wineco gibi bir veya daha fazla uygulama belirtimine aktarma ile).

İçerik belirtiminde (veya parametreler tirilmiş türlerin kullanıldığı başka herhangi bir yerde), o kullanılmış bir gerçek bir parametre sağlarız. Şekil II-19, ROSE modülünü ve Şekil II-20, wineco modülünü göstermektedir. Artık tüm ihracatın ROSE'dan yapıldığıını unutmuyın - ROSE hiç ithalat yapmaz.

```

ROSE modülü {joint-
iso-itu-t uzaktan iş lemleri(4) genel-ROS-PDU'lar(6)}
TANIMLAR
OTOMATİ K ETİ KETLER
BAŞ LAMAK
İ HRACAT İ Ş LEMİ , HATA, Rose-PDU{};

Rose-PDU {İ Ş LEM:Kullanıcı iş lemleri} ::=

    SEÇİ M
        {invoke Invoke {User-ops}, reddet Reddet, sonuç
        ReturnResult {User-ops},
        hata
        ReturnError {Kullanıcı İ ş lemleri} }

    {OPERATION:User-ops} ::= SEQUENCE'ı çağırır
        { invokeId INTEGER, iş lem kodu
        OPERATION.&operationCode {{User-ops} !
        invoke-unrecognisedOperation}, argüman OPERATION.&ArgumentType {{User-ops}

        {@iş lem kodu} ! invoke-mistypedArgument) İ STEĞE BAĞLI }

    Reddet ::= vb.
    ReturnResult {OPERATION:User-ops} ::= vb.
    ReturnError {OPERATION:User-ops} ::= vb.
    SON

```

Ş ekil II-19: Parametreli bir türün tanımlanması ve dış a aktarılması

Ş ekil II-19'da dikkat edilmesi gereken birkaç nokta vardır. Invoke, Reddet, ReturnResult ve ReturnError mesajlarını ayrı ayrı dış a aktarabilirdik, ancak bunları bir "Rose-PDU" CHOICE türü olarak bir araya getirmeyi ve dış a aktarmayı seçtik. Bu, "Rose-PDU"nun "User-ops" kukla parametresiyle parametreleş tirilmesi gerektiği anlamına geliyordu; bu boş parametre, o SEÇİ M içinde Invoke ve ReturnResult ve ReturnError kullanımına gerçek parametre olarak sağlandı.

Invoke, ReturnResult ve ReturnError , daha sonra tablo ve iliş kisel kısıtlama için kullanılan kukla parametreleri için biraz kafa karıştırıcı bir ş ekilde aynı adı kullanır. Kukla bir parametrenin iç içe tür tanımları zincirinden geçirilmesi durumu oldukça yaygındır ve her seferinde aynı adın kullanılması da oldukça yaygındır, ancak lütfen bunların resmi olarak farklı adlar olduğunu unutmayın - sizin yapacağınız gibi Bir kukla parametre adının kapsamı, parametreleş tirilmiş türün sağ tarafıyla sınırlıdır.

Ayrıca İ HRACAT listesinde (ve daha sonra Ş ekil II-20'deki İ THALAT listesinde) Rose-PDU'dan sonra "Ø" geçtiğine dikkat edin. Bu bir gereklilik değil, ancak bir insan okuyucunun bunun parametreleş tirilmiş bir tür olduğunu netleş tirmesine yardımcı olur.

Bu durumda boş parametre listesi yalnızca bir boş parametreye sahiptir (daha fazla olsaydı, virgülle ayrılmış bir liste olurdu) ve burada bir Bilgi Nesnesi Kümesi olan boş bir parametrenin sözdizimini görüyoruz. Bu, sınıf adı ("İ Ş LEM"), bir ":" (iki nokta üst üste), ardından bir Bilgi Nesnesi Kümesi olduğum için büyük harfle başlaması gereken boş parametre adıdır.

Kukla parametrelerin başka birçok şeyle olabileceğini ve tipler dışındaki şeyle de parametreleş tirilebileceğini bir sonraki bölümde göreceğiz, ancak şimdilik bu kadar yeterli.

Ş ekil II-20, Wineco-main'e içe aktarmayı ve Rose-PDU parametreli tipe gerçek parametre olarak wineco'ya özgü "My-ops" tedariki ile yeni ROSE tabanlı soyut sözdiziminin tanımını gösterir.

```

Wineco-main { ortak-
iso-itu-t uluslararasıRA(23) set(42)
    set-vendors(9) wineco(43) modüller(2) ana(5)}
TANIMLAR
OTOMATİ K ETİ KETLER
BAŞ LAMAK
İ THALAT
Rose-PDU{} Rose modülünden {ortak-iso-itu-t
uzaktan iş lemler(4) genel-ROS-PDU'lar(6)}
My-Ops FROM Wineco-operations { ortak-iso-itu-t
internationalRA(23) set(42)
    set-vendors(9) wineco(43) modüller(2) ops(4);}

wineco-soyut-sözdizimi ÖZET-SYNTAX ::=
{ Rose-PDU{My-ops} TARAFINDAN TANIMLANAN
{ ortak-iso-itu-t uluslararasıRA(23) seti(42)
    set-vendors(9) wineco(43) soyut sözdizimi(2)}
    HAS PROPERTY
    {geçersiz kodlamaları iş ler}
    -- Rose spesifikasyonuna bakın -- }

SON

```

Ş ekil II-20: Wineco soyut sözdizimini tanımlamak için ROSE-PDU'yu kullanma

10 Henüz söylememmiş ne var?

Bu bölüm, okuyucunun Bilgi Nesneleri ile ilgili kavramları ve ASN.1 yapılarının parametreleş tirme ilkesini iyi bir ş ekilde anlamasını umarız, ancak hikayenin tamamını anlatmamıştır.

Neden her zaman söylenecek daha çok ş ey var?

Bir sonraki bölümde, bir Bilgi Nesnesi Sınıfı belirttiğinizde tanımlayabileceğiniz alan türleri için tüm olasılıklar hakkında biraz daha ayrıntıyi tamamlayacağımız.

Ayrıca, belirli bir sınıfın nesnelerini tanımlamak için daha kullanıcı dostu (ve bazen daha az ayrıntılı) bir notasyonun kullanılmasını sağlayan (Ş ekil II-17'deki notasyonun yerine geçen) değil iş ken sözdizimi adı verilen önemli bir olanak vardır.

Kısıtlar konusunda, daha önceki böümlerde basit alt tip kısıtlamaları gördük ve bu bölümde tablo ve ilişkisel kısıtlamalar tanıtıldı. Bir sonraki bölüm, kısıtlamaların bazı başka örneklerini keşfedecek ve ayrıca, kullanıcı tanımlı kısıtlama olarak adlandırılan kısıtlamanın geri kalan türünü tanıtabaktır.

Parametreleş tirme konusunda, soyut sözdiziminin sözde parametrelerinden ve geniş letilebilir boş kümeden bahsedilmesi de dahil olmak üzere, yapılacak biraz daha tartışma var.

Son olarak, spesifikasyonlarda boş luk bırakmanın alternatif yollarını sağlayan kalan ASN.1 yapılarından bahsedeceğiz. Okuyucular, bu bölümün sonunda notasyonla ilgili olarak "ASN.1 Tamamlandı" olarak onaylanabileceklerini bilmekten memnun olacaklar ve bu kitabı okumaktaki tek ilgileri buysa, burada durabilirler!

Bölüm 7

Sınıflar, kısıtlamalar ve parametreleş tirme hakkında daha fazla bilgi

(Veya: Bilmek istediğinden çok daha fazlası!)

Özet:

Bu bölüm:

bir sınıf tanımında kullanılabilen tüm farklı Bilgi Nesnesi Sınıf Alanı türlerini açıklar;

Bilgi Nesnelerini tanımlamak için "değ iş ken sözdizimini" açıklar (bu, muhtemelen bu bölümde ele alınan en önemli alandır - baş ka hiç bir ş ey okumadıysanız o materyali okuyun);

kısıtlamalar ve parametreleş tirme tartış masını tamamlar;

TYPE-TANIMLAYICI yerleşik sınıfını tanımlar;

"delikler" için ASN.1 gösterimsel desteği tartış masını tamamlar.

1 Bilgi Nesnesi Sınıf Alanları

Genel protokol belirticilerinin, protokollerini tamamlamak için kullanıcılarından toplamak istediklerini buldukları birçok farklı türde bilgi vardır ve ASN.1, çeşitli türde Bilgi Nesnesi Sınıfı Alanının belirtimine izin verir. Burada sırayla her birine kısaca bakıyoruz. Şekil II-21, tüm farklı türde alanların göründüğü bir Bilgi Nesnesi Sınıfının yapay bir örneğiini verir.

Bilgi Nesnesi Sınıfları için pek çok farklı türde bilgi vardır, bazlarında nadiren rastlanır. Bu madde hepsini listeler!

Geçerli protokol belirtimlerinde tüm bu farklı türdeki alanların örnekleri vardır, ancak bazıları diğer erlerinden çok daha yaygındır.

Ç Zİ M ::= SINIF

```

{&Tür-alanı, &sabit-tür-
değ er-alanı INTEGER, &değ iş ken-tür-değ er-alanı &Tür-alanı,
&Sabit tip değ er kümesi alanı Numaralandırmam,
&Değ iş ken-tür-değ er-set-alanı &Tür-alanı, &nesne-alanı
OPERASYON,
&nesne-kümesi-alanı HATA }

```

Şekil II-21: Farklı alan türlerinin bir örneği

gibi bu alanlara yapılan atıflar

Ç Zİ M.&sabit tip değ er alanı

ASN.1 notasyonunda mümkündür (gerçek bir nesne seti tarafından kısıtlanmış veya kısıtlanmamış). Bu gösterimin kullanımına nesne sınıfından bilgi denir.

Genel olarak, tanımlanmış Bilgi Nesneleri alanlarına referanslar olması da mümkündür ve tanımlanmış Bilgi Nesnesi Setleri gibi notasyonu kullanarak

çizim-nesnesi.&Tür-alanı Çizim-nesne-kümesi.&sabit-tip-
değ er-alanı

Bu gösterimin kullanımına nesneden bilgi ve nesne kümesinden bilgi denir.

Bazı durumlarda, bu tür bir gösterim yasaktır (neyin yasal olup neyin olmadığı ina iliş kin basit bir tablo için Standarda ve genel bir açıklama için aş ağı idaki metne bakın). Bununla birlikte, iyi bir rehber, eğ er bir anlam ifade ediyorsa, o zaman yasaldır. Aş ağı ida, her tür alan için bu gösterimlerin anlamını ve yararlılığı in ve bunları kullanmak isteyebileceğiniz durumları tartış acağı iz.

1.1 Tip alanları

Daha önce karşı ılaç tiğ ımız tip alanı. Alan adı büyük harfle baş lamlıdır ve hemen ardından virgül gelebilir veya örneğ in ş unu yazabiliriz:

Tip alanları yaygın ve önemlidir. Protokollerdeki boş lukları dolduruyorlar ve bunlara olan ihtiyaç, Bilgi Nesnesi Sınıfı konseptinin geliş imini sağ ladi.

&Tür-alanı-isteg e bağ lı İ STEĞE BAĞI,
&Type-field-defaulted DEFAULT NULL,

İ STEĞE BAĞI durumunda, o sınıfa ait bir Bilgi Nesnesi tanımlandığı inda o alan tanımsız bırakılabilir. Bu alan boş tur ve "boş", alana girilebilecek herhangi bir değ erden farklıdır. Bir Bilgi Nesnesi Kümesini kısıtlama olarak uygulama kuralları, yalnızca Dİ Zİ DE karşı iliş gelen öj e eksikse boş bir alanla eş leş menin meydana geldiğ ini söyler.

Bu nedenle, sınıf tanımında OPTIONAL yazmak yalnızca, protokolün tip tanımında karşı iliş gelen öj ede ("delik") OPTIONAL görünüyorrsa mantıklıdır. Buna karşı iliş, DEFAULT protokole herhangi bir gereksinim koymaz, yalnızca belirli bir bilgi nesnesinin tanımında hiç biri belirtilmemiş se kullanılacak türü sağ lar. Yukarıdaki çizimde NULL'u belirledik. Elbette yerleş ik veya kullanıcı tanımlı herhangi bir ASN.1 türü olabilir, ancak DEFAULT ile NULL kullanımı en yaygın olanıdır.

"Nesne sınıfından bilgi" gösterimini kısıtlamasız kullanırsak, "açık tip" denen şeyle sahibiz. Bu gerçekten, şeyle artnamenin tamamlanmasını kimin ve nerede sağlanmasını dair hiç bir göstergede olmaksızın eksik bir şeyle artname anlamına gelir. Böyle bir kullanım yasak değil ilama olmamıştır! yapma! Basit bir tablo kısıtlaması ile kullanmak çok daha iyi değil, çünkü kod çözümcünün bir tür kümesinden hangisinin kodlandığıını bilmesinin bir yolu yoktur ve bu tür bir bilgi olmadan kodlamalar belirsiz olabilir. Tip kısıtlaması adı verilen bir "açık tipe" sağlanabilen özel bir kısıtlama vardır. Bu, son bölümün 8. maddesinde kısaca belirtildiği gibi buraya yazabiliriz

Ç Zİ M.&Tür-alanı(Türüm)

Taşındığını anlamadımlı açısından, yalnızca "Türüm" yazmakla tam olarak eşdeğerdir, ancak PER'de ekstra uzunlukta bir sarmalayıcı alır ve genellikle araçlar tarafından gömülümek yerine ayrı bir bellek parçasına işareti olarak işlenir. içeren veri yapısında. Protokolde, kendileriyle ilişkili bazı meta-anlamlara sahip birkaç yer varsa (güvenlik verilerini taşıyan türler gibi) yararlıdır, böylece bir Dİ Zİ veya KÜME'nin bir öğesi olarak yazarak

GÜVENLİ K-DATA.&Tür-alanı (Veri-tipi-1)

ögeyi ASN.1 türü "Veri türü-1" olarak tanımlarsınız, ancak onu açıkça bir "GÜVENLİ K-VERİ LERİ" türü olarak işaretlersiniz.

Bir tür alanı için "nesne kümesinden gelen bilgilerin" kullanılması yasa dışıdır. Bu, genel olarak bir dizi ASN.1 türü (nesne kümesindeki her bir nesneden bir tane) üretir ve ASN.1'de bir tür kümesini kullanabileceğiniz hiçbir yer yoktur.

Bir tür alanı için "nesneden gelen bilgilerin"in kullanılması, tek bir tür üretir ve uygun koşullarda "Data-type-1" kullanan önceki SEQUENCE veya SET öğesine bir alternatif olabilir.

ile birlikte
nesne1.&Tür alanı
nesne1 GÜVENLİ K-VERİ LERİ ::=
{&Type-field Data-type-1, etc }

Bu ikinci yapının, Veri tipi-1'i bir GÜVENLİ K-DATA tipi olarak işaretlediğiini, ancak önceki yapının ürettiği kapsüleme üretmediğini unutmayın. "object1.&Type-field" kullanımı, "Data-type-1" kullanımıyla tamamen aynı kodlamayı üretir.

1.2 Sabit tip degerler alanları

Bu alanların adlarının küçük harfle başlaması ve adın ardından o alan için sağlanması gereken degerin türünü belirten ASN.1 türünün gelmesi gereklidir. Bu spesifikasyonu İSTEĞE BAĞLI ve VARSAYILAN ve ayrıca EŞ SİZ (geçen bölümde açıklandıktan gibi) dahil edilmesine yine izin verilir.

Tip alanlarıyla yakından bağlanılan, olan bunlara yine sık sık rastlanır.

Bu alanlar için en yaygın türler INTEGER veya OBJECT TANIMLAYICI veya ikisinin birleşimi, ancak BOOLEAN veya ENUMERATED türü de oldukça yaygındır. Son ikisi, toplanan bilgiler bir protokol mesajında taşılmak üzere tasarlanmadığından, bunun yerine prosedürlerde bir "deliği" tamamladığından kullanılır.

Örneğin, ROSE örneği imizi yeniden ele alacak olursak, bazı işlemeler için "ReturnResult"un hiçbir bilgi taşıtmama olasılığında izin verdiği imizi varsayılmış. Bu, İSTEĞE BAĞLI koyarak çözülebilir

OPERATION.&ResultType'in sınıf tanımında ve ayrıca "ReturnResult" SIRASI'nın "hole" öğesi esinde. Ancak bundan daha da ileri gitmek isteyebiliriz. Sonuç türünün olmadığı durumlarda, kritik olmayan bazı iş lemleri için "ReturnResult"un asla gönderilmemişini ("Reddet" veya "ReturnError" baş arızılığı gösterir), diğer erleri için her zaman olması gerekişini belirtmek isteyebiliriz. İş lemin tamamlandığının teyidi olarak gönderilir ve yine diğer erleri için gönderilip gönderilmemesi uzak sistemin bir seçenekidir. Bu durumda, sabit tip değ er alanı ş uş ekilde olabilir:

&returnResult ENUMERATED {her zaman, hiçbir zaman, isteğ e bağlı}
VARSAYILAN her zaman,

ve ROSE kullanıcısı, gerekli davranışın bu olduğ u iş lemleri için "asla" veya "isteğ e bağlı" değil erini belirtecektir.

Bu durumda "nesne sınıfından bilgi" yapısının kullanılması, basitçe sabit tip değ er alanının tipini üretir. Yani kullanımı

Ç Zİ M.&sabit tip değ er alanı

(neredeyse) tam olarak yazmaya eş değ erdir

TAM SAYI

Aradaki fark, INTEGER türüne ILLUSTRATION sınıfındaki bir nesne kümesiyle tablo kısıtlaması uygulayamamanızdır. Bunu "nesne sınıfından gelen bilgiler" yapısına uygulayabilirsiniz (ve sıkılıkla yaparsınız).

Bu durumda hem tek bir tamsayı değ eri üreten "nesneden bilgi" hem de bir tamsayı değ erleri kümesi (tamsayı türünün bir alt kümesi) üreten "nesne kümesinden bilgi" bu durumda izin verilir. Böylece, ILLUSTRATION sınıfının "Illustration-object-set" nesne kümesiyle ş unu yazabiliriz:

Çizim-nesne-kümesi.&sabit-tür-değ er-alanı

onun yerine

Ç Zİ M.&sabit-tip-değ er-alanı (Çizim-nesnesi)

Fark ne? Çok değ il! İkinci durumda, bu öge eye iş arret etmek için iliş kisel bir kısıtlamayla (ILLUSTRATION sınıfının bir tür alanında) "@" kullanabilirsiniz. Önceki durumda yapamazdin. İkincisi, normalde görülecek iniz ş eydir.

1.3 Değ iş ken tipi değ er alanları

Bu muhtemelen ikinci en az yaygın alan türüdür. Ana kullanımı, bir tür alanında sağ lanan bir tür için varsayılan bir değ er sağlamaktır.

Çok daha az yaygın.
Teorik olarak yararlı bir kavramın ilginç bir örneği!

Alan adından sonra, bu sınıf tanımında tanımlanan bir tür alanın (&T-F diyor) adı gelir. Bu sınıfa ait bir bilgi nesnesinin tanımındaki değ iş ken tipi değ er alanı için sağ lanan değ erin, &T-F alanı için sağ lanan tipte bir değ er olması gereklidir.

Bu alan İSTEKE BAĞLI veya VARSAYILAN olarak iş arretlenebilir, ancak bu durumda bu alan ile &T-F alanı arasında İSTEKE BAĞLI ve VARSAYILAN kullanımı arasında bağılantı kurulabilir. Kabaca, mantıklıysa, öyle

izin verilir, dēg ilse dēg ildir! Neye izin verilip verilmelī inden emin dēg ilseniz Standardı kontrol edin (veya ASN.1'inizi kontrol etmek için bir araç kullanın). Kabaca, hem bu alan hem de &T-F aynı İ STEĞE BAĞLI veya VAR SAYILAN kullanıma sahip olmalıdır veya olmamalıdır ve ikinci durumda, bu alan için varsayılan dēg er, &T-F alanı için varsayılan türde bir dēg er olmalıdır. .

Tek bir dēg ere sahip bir alandan bekleyeceğiniz gibi, alan adında "&"den sonra küçük harf bulunur.

"Illustration-object-set.&variable-type-value-field" kullanımı yasaktır (yasal ASN.1 dēg ildir). "illustration-object.&variable-type-field" kullanımı, o alana atanın dēg eri üretir.

1.4 Sabit tip dēg er ayar alanları

Bunlar, sabit türde bir dizi dēg er tutan alanlardır ve bu nedenle alan adı ve işaretinden sonra büyük harfle başlar.

Oldukça sık kullanılır, esas olarak bir protokolün prosedürlerindeki boş lukları doldurmamız ve bazılarının seçilmesi ve bazılarının yasaklanması gereken olası eylemlerin bir listesine (bir sırasına) sahip olmamız gerekītinde.

Burada gerekli olan bilgi, alan adından sonra gelen tipte bir dēg erler setidir (vali tipi), veya baş ka bir deyīle o tipin bir altkümesidir. Bu dēg erler, uygulanan basit bir alt tip kısıtlaması olan bir tipe referans tipi ile sağlanabilir veya son bölümde açıklanan dēg er seti gösterimi kullanılarak sağlanabilir.

Bu alanın en yaygın oluşumu, birkaç olasılığın olduğunu yerdır ve bir Bilgi Nesnesinin tanımlayıcısının, bu Bilgi Nesnesi için izin verilecek olanları seçmesi gereklidir.

Böylece, bir sınıf tanımında:

&Sabit tip dēg er kümesi alanı
SAYILANDIRILMIŞ {postayla onayla, faksla onayla, kayıtlı onayla, e-postayla onayla telefonla onay},

kullanıcının belirli bir bilgi nesnesi için numaralandırma olasılıklarının bazı alt kümelerinin kullanılabileceğini belirtmesine izin vermek için kullanılabilir. Yukarıdaki tanımı gerçek bir hayali senaryoya dönüş türmek okuyucunun hayal gücünü bırakılmıştır!

Bu alanı kullanarak hem nesnelerden hem de nesne kümelerinden bilgi çıkarılması, yalnızca ilgili nesnelerin herhangi birinde görünen dēg erleri içeren, sınıf tanımında kullanılan türde bir (alt)dēg erler kümesi üretir.

1.5 Dēg işken tipi dēg er ayar alanları

Ben (bu metnin yazarı!) bu tür bir alanın pratikte gerçekten gerçekleştiginden hīc emin dēg ilim. Büyük ölçüde, mevcut alan türleri "setini tamamlamak" için gerekli göründüğü için eklendi! Bunun için iyi bir kullanım bulun!

Bu kutuda "bu hīc kullanılmadı!" diyebilirim. Metnin gödesinde daha temkinliyim!

Büyük harfle başlılar ve alan adından sonra aynı sınıf tanımdaki bir tür alanın adı (&T-F) gelir. Alan, &T-F'ye konulan türde bir dizi değil (bir alt küme) verilerek tamamlanır.

Bir nesneden bilgi çıkarmı, o alana atanın değil eri verir, ancak bir nesne kümesinden bilgi ayıklama notasyonu bu alan türü için geçersizdir.

1.6 Nesne alanları

Belki de şartsızı bir şeilde, bu, aşırıda açıklanan nesne kümesi alanından daha az yaygındır, ancak kullanılmaktadır.

Nesne alanı, alan adını izleyen sınıftaki bazı nesnelerin kimliğinin (bir bilgi nesnesi referans adı) taşıdır.

Bunun ana kullanımı, her nesne için bu alanların tanımlarını tekrarlamak zorunda kalmadan, kullanıcının (birkaç nesne için aynı tanım verilen) bir alan kümesinin tanımlarına kolayca baş vurmasına izin vermektedir.

Bu, sabit tip değil er alanının nesne ve sınıf eşdeğeri eridir.

Başlıca kullanımı, bilgi nesnesi tanımlarının yapılandırılmasına yardımcı olmaktadır. Bir sınıfın her nesnesi (örneğin ANA SINIF), ANA SINIF'a bir dizi alan ekleyecek belirli ek bilgilerin belirtilmesini gerektireceğse (ve aynı ek bilgilerin farklı nesneler için sık sık belirtilmesi muhtemel) ANA-SINIF o zaman ayrı bir sınıf tanımlamak mantıklıdır (EK-Bİ LGİ -SINIFI diyelim). İLAVE-Bİ LGİ -SINIFININ nesneleri yalnızca ek bilgileri taşırlar ve bunlara yapılan başvurular, ANA-SINIFIN bir nesne alanına dahil edilir.

Bir nesneden ve bir nesne kümesinden gelen bilgi, sırasıyla tek bir nesne veya bir dizi nesne üretir. Bu yapıların kullanımı, birinin alanlarının bir alt kümesi olarak diğerinin alanlarına sahip olduğu yakından ilişkili tanımlanmış iki sınıfımız varsa (OPERATION-X ve CHAINED-OPERATION-X dizini önektr) karışışlıdır. Bu durumda, ZİNÇİRLİ OPERASYON-X için tanımlanan nesnelerin, ilgili OPERATION-X nesnesinden bilgi çıkarılarak tanımlanan OPERATION-X'e karşı ilişkili gelen alanlara sahip olması yerine, tanımdaki "parmak sorununu" ölüleyebilir (ve daha net bir belirtim sağlar). tanımı tekrar etmektense. (Bu nokta aslında tüm farklı alan türleri için nesneden alınan bilgilerin kullanımı için geçerlidir.)

1.7 Nesne seti alanları

Bunun bir işlemle ilgili hataları listelemek için kullanıldığıını daha önce gördük. Bir nesne kümesi olan bir şevey için bekleniği gibi, &den sonra bir büyük harf gelir.

Bu, iyi kullanılan başka bir alandır. En çok ROSE OPERATION sınıfının &Errors alanı olarak kullanılmasıyla bilinir.

Nesneden ve nesne kümesinden gelen bilgilere yine bariz sonuçlarla izin verilir.

1.8 Geniş letilmiş alan adları

Bir sınıfın, nesnenin veya nesne kümelerinin alanlarına baş vururken, kendisi bir SINIF veya nesne veya nesne kümeleri olan bir ş eyle karşı ilaç abilirsiniz (örneğ in, OPERATION.&Errors, ERROR sınıfını sunar).

Bu olduğunda, bir "." daha ekleyebilirsiniz. (nokta) ve ardından elde ettiğiniz sınıfın alan adı.

Standarttaki örnekler dışında bunların kullanıldığı ini asla görmezsiniz! Bu metni atla!

Böylece

İ § LEM.&Hatalar.&ParametreTipi
ve
İ § LEM.&Hatalar.&hataKodu

geçerli notasyonlardır ve ş una eşdeğerdir:

HATA.&ParametreTürü
ve
HATA.&hataKodu

OPERATION sınıfından bir bilgi nesnesi seti kullanan benzer yapılar daha ilgi çekicidir.

Burada

İ ş Lemlerim.&Hatalar.&hataKodu

"İ ş Lemlerim"deki iş lemlerden herhangi biri için hata kodları olan değil er kümelerini sunar ve

arama iş Lemim.&Hatalar.&hataKodu

"arama iş Lemim"in olası hatalarını tanımlayan bu değil erler kümelerini sunar.

Tabii ki, bu herhangi bir uzunluğ a kadar ilerleyebilir, bu nedenle, kendisi OPERATION sınıfından bir nesneler kümeleri olan OPERATION sınıfı bir nesne kümeleri alanımız varsa (bu aslında ROSE'da gerçekteşir - alan "&Linked" olarak adlandırılır ve bu ş ekilde kaydeder) "bağlı iş lemler" olarak adlandırılır, ş öyle ş eyle yazabiliriz:

my-op.&Bağlıİmleri.&Bağlıİl.&Bağlıİl.&Hatalar.&hataKodu

Bu ş eyle tamamen büyüleyici - evet? Ancak okuyucu, bunun için gerçek bir kullanım bulmaya zorlanır! (ASN.1'e adil olmak gereklidir, bu tür notasyonlar, notasyonda tutarlılık ve genellik isteniyorsa ve sağlama maliyeti çok düşükse doğal olarak ortaya çıkar. Oldukça bariz notasyonlara izin verilmemesindense bunlara izin verilmesi daha iyidir.)

2 Bilgi Nesnesi tanımı iç in de ğ iş ken sözdizimi

Tarihsel olarak, Bilgi Nesnesi Sınıfları kavramı tam olarak geliş tirilmeden önce, ASN.1'in (artık geri çekilmiş) önceki bir özellig i olan makro notasyon, ROSE (ve di ğ erleri) tarafından kullanıcılar, protokollerindeki boş lukları doldurmak için gerekli bilgiler. ROSE'un (ve di ğ erlerinin) sağ lađi ğ i gösterim

oldukça insan dostuydu. Kesinlikle "&" karakterini içermiyordu ve çok u zaman virgül içermiyordu! Genellikle bir İ ngilizce cümle gibi okunur, "I LE" gibi bağ laçlar gösterime dahil edilir veya bir dizi anahtar kelime-de ğ er çifti olarak.

Birkaç teknisyen, bilgi nesnesi sınıflarını tanımlar, ancak birçok kullanıcı bu sınıfların nesnelerini tanımlar ve hatta daha fazla (teknik olmayan) kişi i bu tanımları okur. Belirli bir sınıfın nesnelerini tanımlamak için insan dostu bir gösterime ihtiyacımız var. "De ğ iş ken sözdizimi" önemlidir ve çok kullanılır.

Örneğ in, bir ROSE iş lemi tanımlamak iç in ş unu yazarsınız:

```
my-op OPERASYON
  ARGUMENT Tip-for-my-arg
  SONUÇI ş lem sonucum için yaz
  HATALAR {hata1, hata4} ::= yerel 1
```

(Aş ađıki metinde buna geçici gösterim diyoruz.)

Bu, ROSE tarafından tanımlanan geçici notasyondu. (Di ğ er gruplar benzer ancak ilgisiz sözdizimini tanımlar - özellikle bazıları listeleri ayırmak iç in virgül kullanır, di ğ erleri dikey çubuk kullanır).

Bu sözdizimi sağ lađi ğ inda (Bilgi Nesnesi Sınıfı kavramından önce) onunla iliş kili çok az semantik olduğ unu burada not etmek önemlidir. Yukarıdaki gösterim resmi olarak (bir ASN.1 aracına göre) ş unları söylemek iç in kıvrımlı bir sözdiziminden baş ka bir ş ey de ğ ildi:

```
my-op CHOICE {yerel INTEGER, global NESNE TANIMLAYICISI} ::= local:1
```

ve tipik olarak "my-ops" de ğ er referansı hiç bir zaman hiçbir yerde kullanılmadı. Görünüş e göre pek çok bilgi toplanıyordu, ancak daha sonra (metnin ne anlama geldiğ ine dair herhangi bir resmi model açısından) "yere atıldı".

(Bir kenara, yukarıda "yerel"den sonra ":" (iki nokta üst üste) eklenmesi bu tartış ma iç in temel de ğ ildir - bu, erken çalış malarda bir seçim de ğ erinin (örn.) "yerel 1" olarak ifade edilmesinden ve 1994 sonrası "yerel:1" olarak).

Bununla birlikte, yukarıdaki gösterim, gerç ekten bugün nesne tanımıyla elde edecek iniz aynı amaca hizmet etmek iç in tasarılandı:

```
my-op I § LEM ::= {&iş lemKodu
  yerel:1,
  &ArgumentType I § lem-argim-için-yazım,
  &SonuçTürü I § lem sonucum için yaz, {hata1 |
  &Hatalar hata4} }
```

(Bunu aş ađıda nesne tanımı notasyonu olarak adlandırıyoruz.)

Birçok şeysi gözlemleyebiliriz. İlk olarak, bir insan için ad-hoc notasyonu okumak muhtemelen nesne tanım notasyonundan daha kolaydır, ancak net bir semantik alt sabitlemenin olmaması daha zeki okuyucuların kafasını karıştırabilir! İkinci, notasyon ad hoc olduğu undan, bunun için herhangi bir araç desteği üretmek çok zordu. Üçüncü, notasyon ad hoc olduğu için, bir aracın bu ad hoc notasyonun ne zaman sona erdiğini bilmesinin bir yolu yoktu ve biz normal ASN.1'e döndük (ad hoc notasyonun etrafında parantez yoktu). Son olarak, bu gösterimin kullanımı ile ROSE protokolündeki boş luklar arasında resmi bir bağ lantı (sınırlama olarak bir Bilgi Nesnesi Kümesi kullanarak elde ettiğimiz gibi) yoktu.

Bununla birlikte, Bilgi Nesnesi Sınıfı materyali 1994'te ASN.1'e eklendiğinde (ve makro notasyon kullanımı geri çekildiğiinde), daha insan dostu (ama yine de tamamen makine dostu ve tam semantikle) izin vermenin önemli olduğu hissedildi. Belirli bir sınıftaki nesnelerin tanımı için notasyon.

Amaç, bir sınıfın tanımlayıcılarının, o sınıfın nesnelerini tanımlamak için, şimdiden kadar ad-hoc notasyon olarak sağlanması olan notasyona (makine işlenebilirliğiinden ötürü vermeden) mümkün olduğu kadar yaklaşımalarını sağlayacak notasyonu belirtebilmelerine izin vermekti. ASN.1'in "değişken sözdizimi" bu amacı destekler (gerçekleşir).

Değişken sözdizimi, bir sınıf tanımının hemen ardından "SYNTAX İLE" anahtar sözcüklerinin ve ardından o sınıfın nesnelerini tanımlamak için sözdiziminin bir tanımının gelmesini gerektirir. Bu anahtar sözcükler, sınıf tanımının ardından mevcut değil ise, nesneleri tanımlamak için kullanılabilen tek sözdizimi, nesne tanımı notasyonudur. (İkinci, "WITH SYNTAX" yan tümcesi olsa bile, nesneleri tanımlayan kullanıcılar tarafından kullanılabilir.)

Şekil II-22, OPERATION sınıf tanımına WITH SYNTAX ekler. (Yine, gerçek ROSE spesifikasyonunun bundan biraz daha karmaşık olduğunu vurgulamalıyız - ROSE hakkında eksiksiz bir eğitimi vermiyoruz!)

```

İ § LEM ::= SINIF
{&işlem Kodu SEÇİMİ Mİ {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}
BENZERSİZ,
&ArgümanTürü,
&SonuçTürü,
&Hatalar HATA İSTEĞE BAŞLI }
SÖZDİZİMİ Mİ İLE
{ ARGUMENT &ArgümanTürü
SONUÇ &SonuçTürü
[HATALAR &Hatalar]
KOD &işlem Kodu }

```

Şekil II-22: OPERATION sınıf tanımına WITH SYNTAX'in dahil edilmesi

Bu ne yapıyor/yapıyor? Bir sınıf işlemi nesnesinin şunu sözdizimi ile tanımlanmasına izin verir:

```

my-op OPERASYON ::=
{ ARGUMENT Type-for-my-arg
SONUÇ İŞLEM sonucum için yaz
HATALAR {hata1 | hata4}
KOD yerel:1 }

```

Okuyucu, göze hoş görünmeyen "&" iş aretinin kaybolduğunu, bununla ad-hoc-notasyon arasındaki güçlü benzerliği, ancak aynı zamanda tanımın etrafında makineyle işlenebilirliği sürdürmek için gerekli olan kıvrımlı parantezlerin varlığıını fark edecktir.

"SYNTAX İ LE" nin ardından ne yazabilirsiniz? Kabaca, normalde komut satırı sözdizimini tanımlamada kullanılan gücü sahipsiniz - sınıfın alanlarına referanslarla serpiş tirilmiş bir dizi kelime. Bir nesneyi tanımlarken, tanımlayıcı bu kelimeleri sırayla tekrar etmeli ve baş vurulan herhangi bir alanı tanımlamak için gerekli sözdizimini vermelidir. Sözcük dizisi ve/veya alan referanslarının köşeli parantez içine alındığı durumlarda (yukarıdaki "[HATALAR &Hatalar]"da olduğu gibi), sözdiziminin bu kısmı atlanabilir. (Elbette, köşeli parantezlerin dahil edilmesi yalnızca "WITH SYNTAX" yan tümcesinin tanımında yasaldı çünkü "&Errors", ana sınıf tanımında "İSTEĞE BAĞLI" olarak iş aretlendi.)

Bir "sözcük", WITH SYNTAX yan tümcesinin amacı için, muhtemelen ortasında (tek) tire bulunan bir büyük harf (küçük harf değil) harf dizisi (sayıyla izin verilmeyen) olarak tanımlanır.

Ayrıca, WITH SYNTAX yan tümcesine bir virgül eklemek (ancak başka bir noktalama işareti içermemek) mümkün değildir, bu durumda virgül, o sınıftaki bir nesnenin tanımında karşılık gelen noktada görülmeli.

İsteğe bağlı bütümleri içinde isteğe bağlı bütümleri oluşturmak için köşeli parantezler iç içe kullanılabilir. Bununla birlikte, hem gerçek nesne tanımı üzerinde hiç bir etkisi olmadan bilginin görünür şekilde elde edilmesini önlemek hem de kolay makine işlenebilirliğini sağlama için tasarlanan "Sİ ZDİ Zİ MLİ" kullanımına ilişkili bazı oldukça ciddi kısıtlamalar vardır. Bir WITH SYNTAX yan tümcesinin yazarları, Standardı dikkatlice okumalıdır. Örneğin, § ekil II-23 yasadış olacaktır.

```

İ § LEM ::= SINIF
{&işlem Kodu SEÇİ Mİ {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}
BENZERSİ Z,
&ArgümanTürü,
&SonuçTürü,
&Hatalar HATA İSTEĞE BAĞLI }
SÖZDİ Zİ Mİ İ LE
{ ARGUMENT &ArgümanTürü
SONUÇ &SonuçTürü [GEREKLİ ]
[HATALAR &Hatalar]
KOD &işlem Kodu }

```

§ ekil II-23: WITH SYNTAX'in geçersiz belirtimi

```

İ § LEM ::= SINIF
{&iş İemKodu SEÇİ Mİ {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}
BENZERSİ Z,
&ArgümanTürü,
&SonuçTürü,
&ReturnsResult SAYILANDIRILMIŞ {yapar,
yapmaz},
&Hatalar HATA İ STEĞE BAĞLI }
SÖZDİ Zİ Mİ İ LE
{ ARGUMENT &ArgümanTürü
SONUÇ&SonuçTürü
&DöndürSonuç DÖNÜŞ -SONUÇ
[HATALAR &Hatalar]
KOD &iş İemKodu }

```

Ş ekil II-24: Bir sonuç döndürmek için gereklilikler hakkında bilgi toplama

Bunun nedeni, bir nesnenin tanımlayıcısının, nesnenin bir alanında hiç bir yerde kayıtlı olmayan "GEREKLİ" kelimesini dahil ederek veya içermeyerek bilgi sağlamasına izin vermesidir. Bir nesnenin tanımlayıcısının bir sonucun döndürülmesinin gerekip gerekmediğini belirtmesine izin vermek istenirse, ş ekil II-24'ün tanımı kullanılabilir ve aşağıda idakilere izin verilir:

```

my-op OPERASYON ::=

{ ARGUMENT Type-for-my-arg
SONUÇ İ em sonucum için tür-SONUÇDÖNÜŞ Ü
yapmam
HATALAR {hata1 | hata4}
KOD yerel:1 }

```

Son olarak, ilk olarak Ş ekil II-14'te tanımlanan tabloya benzer bir OPERATION sınıfındaki bir nesnenin kompakt tanımı için bir tablo notasyonu sağlamaya çalışıyoruz. Bu, ş ekil II-25'te gösterilmişdir.

```

İ § LEM ::= SINIF
{&iş İemKodu SEÇİ Mİ {yerel TAM SAYI,
küresel NESNE TANIMLAYICISI}
BENZERSİ Z,
&ArgümanTürü,
&SonuçTürü,
&Hatalar HATA İ STEĞE BAĞLI }
SÖZDİ Zİ Mİ İ LE
{ &iş İemKodu
&ArgümanTürü
&SonuçTürü
[&Hatalar] }

```

Ş ekil II-24: SÖZDİ Zİ Mİ YLE nesnelerin tablo halinde tanımlanmasına izin verir

Ş ekil II-24'teki tanımla ş unları yazabiliriz (Ş ekil II-14 ve II-17'yi karşılaştırın):

```

My-ops OPERASYONU ::= { {asn-val-order-
  Stok-Sipariş i
  Sipariş onaylandı
  {güvenlik hatası | bilinmeyen
  ş ube} } {güvenlik hatası | bilinmeyen
  ş ube} } | {asn-val-query Sorgu
  kullanılabılırlığı | Kullanılabilirlik yanıtı
  {güvenlik hatası | bilinmeyen ş ube | kullanılamıyor} } | {asn-val-state İ stek-sipariş -durumu Sipariş -durumu {güvenlik hatası | bilinmeyen
  ş ube | bilinmeyen sıra} } }
  HÜKÜMSÜZ

```

Yani ş imdi tam bir çember haline geldik! Ş ekil II-14'te kullandığımız resmi olmayan tablo sunumu, ş ekil II-17'nin resmi ama daha ayrıntılı tanımıyla değil iş tirildi, bu (With SYNTAX kullanılarak) ş ekil II-14'ye tekine çok benzeyen bir sözdizimi ile değil iş tirilebilir.

Ş imdi, okuyucu, WITH SYNTAX yan tümcelerinin dikkatle ele alınması gerekişini anlamış olmalıdır. Yalnızca yasal olanın kuralları anlaşılmamalı, aynı zamanda son notasyonda ayrıntılılık ve anlaşılık arasında neyin iyi bir uzlaşma olduğu da belirlenmelidir. Tüm insan arayüzü meselelerinde olduğu gibi, tek bir doğru karar yoktur, ancak biraz düşünmek kötü kararları önyargı edebilir!

3 Kısıtlama yeniden ziyaret edildi - kullanıcı tanımlı kısıtlama

Kısıtlamalara eklenen çok şe y yok. Daha önce tüm basit alt tip kısıtlamaları ele aldık ve son bölümde tablo ve ilişkisel kısıtlamaları ele aldık. Tartışılacak başka bir kısıtlama biçimi daha vardır, sadece kullanıcı tanımlı kısıtlama.

Kullanıcı tanımlı kısıtlamalar - bir yorumdan biraz daha fazlası!
Neden rahatsız?

Yukarıda, insanların bir ASN.1 modülüne dahil edilmek üzere yeni ad-hoc-notasyonu (gerçek semantik olmadan) tanımlamasına izin veren bir notasyonun (makro gösterimi) daha önceki kullanılabılırlığı tartışılmıştır. 1994 yılında bu "tesis" kaldırıldıında, Bilgi Nesnesi kavramının bu makro notasyonun kullanımıyla karşılanan tüm gereksinimleri tam olarak karşılamadığı ortaya çıktı ve geri kalan gereksinimleri karşılamak için kullanıcı tanımlı kısıtlama kavramı tanıtıldı... Bir yorumdan biraz daha fazlası olduğunu ve araçlar bundan çok az yararlanabileceğinden, bu kısıtlama biçimi muhtemelen başka türlü getirilmezdi. Neredeyse her zaman, II-6'nın 9. maddesinde tanıtılan, parametreleş tirilmiş bir tiple bağlanılarak kullanılır.

Makro notasyonu kullanılarak tanımlanan tek parça ad-hoc notasyon, şunları yazabilme yeteneğiydi:

Ş İ FRELİ My-type

bir KÜME veya DİZİ elemanı olarak.

ASN.1 resmi metni tarafından ima edilmese de, bu aslında örneğin, içeriğin My-type tipi kodlamasının bir şifrelemesi (İngilizce metinde belirtilen bir şifreleme algoritmasına göre) olan bir BITSTRING olduğunu anlamına geliyordu.

Parametreleş tirilmiş bir "ENCRYPTED" tipini şunu ekilde tanımlarsak biraz daha netlik elde edebiliriz:

Ş İ FRELİ {Ş ifrelenenek tür} ::= BITSTRING

ve sonra kullan

Ş İ FRELİ {Türüm}

SEQUENCE veya SET ö esi olarak.

(ENCRYPTED türü için tümünü büyük harf kullanarak ASN.1'in kurallarını de ğ il, kuralı ihlal etti ğ imizi unutmamın . Bu, "ENCRYPTED My-type" orijinal ad-hoc notasyonuyla tarihsel uyumluluk nedeniyedir. Ayrıca, yeni biçimsel notasyon, ad-hoc-notasyondan nesne-tanımlama-notasyonuna geçiş te - biraz farklı bir nedenle - gördüğümüz gibi, yeni bir çift süslü parantez içerir.)

Yukarıdakiler geçici bir notasyon kullanmaktan kaçındı, ancak "ENCRYPTED" kukla parametresinin atamanın sağ tarafında hiç kullanılmaması ilginç . BITSTRING'in gerçek de ğ erinin "Ş ifrelenenek tür" türüne (ve ayrıca ASN.1 kullanarak tanımlayamadığımız ş ifreleme algoritması ve anahtarlarına) ba ğ lı olaca ğ ı açıktır.

Bu nedenle, kullanıcı tanımlı kısıtlamayı tanıtıyoruz . Temel biçiminde ş unu yazardık:

Ş İ FRELİ {Ş ifrelenenek tür} ::= BITSTRING
({Ş ifrelenenek tür} TARAFINDAN KISITLANMIŞ TIR)

bu, dummy parametresinin BITSTRING de ğ erini kısıtlamak için kullanıldığıını gösterir. (Kısıtlamada kullanılan birden çok parametre varsa, bunlar CONSTRAINED BY'den sonra ka ğ lı ayrıclar içinde virgülle ayrılmış bir listede olur.)

Kısıtlama, "kullanıcı tanımlı" kısıtlama olarak adlandırılır çünkü kısıtlamanın kesin doğası, resmi ASN.1 notasyonu ile belirtilmemiş tir. Bu yapı, neredeyse her zaman kısıtlamanın kesin doğasını detaylandıran yorumlar içerir. Bu nedenle, yukarıdakiler daha yaygın olarak ş u ş ekilde yazılır:

Ş İ FRELİ {Ş ifrelenenek tür} ::= BITSTRING
({Ş ifrelenenek tür} TARAFINDAN KISITLANMIŞ TIR
-- BITSTRING, -- Ş ifrelenenek Tır'ü ş ifrelemenin -- belirtilen
algoritmayı kullanarak -- alan güvenlik algoritmasında, -- ve --
Güvenlik verileri'nde belirtilen -- ş ifreleme parametrelerinin
sonuçlarıdır --)

Okuyucu ş imdiye kadar (önceki metnin okunduğuunu ve atlanmadığını varsayırsak!), "güvenlik algoritmasının" bazı GÜVENLİ KLER'in (BENZERSİ Z) bir sabit tip de ğ er alanı (muhtemelen nesne tanımlayıcı türünde) olduğunuunu anlayacak kadar bilgi sahibi olmalıdır. -INFORMATION sınıfı, "Security-data" bu sınıfın ilk gelen bir tür alanıdır ve herhangi bir GÜVENLİ K-Bİ LGİ Sİ nesnesi için, tarafından tanımlanan algoritma için gerekli tüm parametreleri taşıyabilen bir ASN.1 türü ile tanımlanır. o nesne Algoritmanın uygulanmasındaki prosedür seçeneklerini statik olarak tanımlayan ve bu süreçte prosedürel "boş lukları" dolduran başka GÜVENLİ K-Bİ LGİ Sİ alanları olabilir.

Açık, güçlü ve basit!
ASN.1 için ne kadar sira
diş il!

4 Parametreleş tirmeyle ilgili tüm hikaye

Parametrelendirmeye eklenecek daha fazla bir ş ey yok ve hepsi oldukça açık ş eyler. Ama iş te burada. © OS, 31 Mayıs

4.1 Neler parametreleş tirilebilir ve parametre olabilir?

Kutu her ş eyi söylüyor. Herhangi bir referans adı biçimini - bir tür referansı, bir de ğ er referansı, bir sınıf referansı, bir nesne referansı, bir nesne seti referansı, " olduğunu, referans adından sonra ve "::="den önce bir kukla parametre listesi eklenerek parametreleş tirilebilir. ş ey" adı baş vuruları tanımlanıyor.

Cevap: Her ş ey ve her ş ey!

Aşağıda, eksiksiz bir parametre aralığı ina sahip bir referans adı örneği verilmiş tir:

```
Örnek referans {INTEGER:intval,
Benim tipim,
BU SINIF,
OPERASYON: Operasyonlarım,
çızı M:çizim-nesnesi} ::=
```

Beklediğimiz gibi, kukla parametrelerin ilk harfi türler, sınıflar ve nesne kümeleri için büyük, nesneler ve de ğ erler için küçük harftir. De ğ erler, nesne kümeleri ve nesneler için kukla parametre listesinin bu parametrelerin türünü veya sınıfını ve ardından bir ":" (iki nokta üst üste) koymuş unutmayın.

(Yukarıdaki örneklerden gerçek bir spesifikasyonda görümediğim tek örnek, bir sınıf olan (yukarıdaki BU SINIF) kukla bir parametredir.

Normalde dummy parametresi, atamanın sağ tarafında bir yerde kullanılır, ancak parametre listesinin kendisinde de kullanılabilir (kendi görünümünden önce veya sonra). Örneği in ş unu yazabiliriz:

```
Örnek1 {Türüm: varsayılan de ğ er, Türüm} ::=
```

Bu gösterim son derece genel ve güçlündür ve birçok uygulaması vardır. Bir Bilgi Nesnesi Kümesinin kukla parametre olarak bildirdiği ROSE örneklerini gördük. Bu muhtemelen kukla parametre olarak kullanılan en yaygın ş eydir, ancak bunun yanında sağ tarafta INTEGER de ğ erlerinin üst sınırı olarak veya üst sınır olarak kullanılan INTEGER türünde bir de ğ er vardır. dizelerin uzunluğu.

Üretim Mesajlaşma Formatları (MMF) spesifikasyonunda da önemli bir kullanım vardır.

Burada protokol spesifikasyonunun büyük bir kısmı "jenerik" bir modülde gerçekleşir ve bir üretim hattındaki tüm hücreler için ortaktır. Bununla birlikte, üretim hattındaki belirli hücreler, kendilerine iletilen bazı ek bilgiler gerektir. Genel modülde kukla bir parametre (bir tür) kullanırsınız ve bunu protokol belirtimimize SEQUENCE'ımızın bir öğesi olarak ekler ve bu parametreleş tirilmiş türü dışa aktarır. Belirli hücrelere yönelik modüller, o hücre için ek bilgileri içeren bir türü tanımlar, genel türü içe aktarır ve bu hücre türü için kullanılacak protokolü, gerçek parametre olarak ek bilgileri içeren türle sağlanan genel tür olarak bildirir.

Bu, ROSE örneği benzer, ancak bir bilgi nesnesi kümesi yerine bir tür kullanır.

Sınır sorununu biraz daha inceleyelim. Çok az protokol, spesifikasyona üst sınırları "donanımsal olarak" dahil eder, ancak tasarımcılar nadiren uygulayıcıların keyfi olarak büyük tamsayıları, dizi yinelemelerini veya keyfi olarak uzun dizileri işlemesini istemeyi amaçladıklarından, bu tür sınırları belirlemek her zaman iyi bir fikirdir. Bu tür sınırların tüm protokol için sabit olduğunu durumlarda, bazı modüllerde bir tamsayı referans adına ihtiyaç duyulan eşitlik sınırları atamak ve ardından bu adları kullanıldıkları modüllere almak için İHRACAT ve İTHALATI kullanmak yaygın bir uygulamadır. sınırlar.

Bununla birlikte, birçok yerde kullanılan ancak her kullanım için farklı sınırlara sahip genel türlerin (bir dizi farklı karakter dizisi türünün SEÇİ Mİ gibi) olduğunu durumlarda, sınırlar için bir INTEGER kukla parametresi kullanmak çok etkilidir ve yaygın uygulama.

Uzun kukla parametre listelerini görmek aslında oldukça nadirdir. Bunun nedeni, herhangi bir bilgi koleksiyonunun (bir sınıf dışında) kolayca bir Bilgi Nesnesi Kümesine dönüşülebilmesidir. Yani önceki önekle (MY-CLASS'ı çıkarma):

```
Önek referans {INTEGER:intval,
  Benim tipim,
  OPERASYON: Operasyonlarım,
  Ç Zİ M:çizim-nesnesi} ::=
```

Bunun yerine şunları tanımlayabiliriz:

```
PARAMETRELER-SINIF ::= SINIF {&intval TAM SAYI,
  &Benim tipim,
  &My-ops OPERATION, &illustration-
  object RESİ M}
```

ve sonra parametre listemiz şöyledir:

```
Önek referans {PARAMETERS-CLASS:parameters} ::=
```

ve sağ tarafta (örneğin) "Türüm" yerine "parameters.&My-type" kullanıyoruz. Bu, birkaç sahte parametre kullanmaktan daha hantal görünebilir, ancak aynı parametre listesi birkaç yerde görünebilir, özellikle de sahte parametreler, çeşitli tip tanımı seviyeleri aracılığıyla gerçek parametreler olarak aktarılıyorsa, kukla parametreleri bir araya getirmek yararlı olabilir. parametreler bu şekilde.

Bunun özel bir durumu, bir protokol tasarımcısının, sınırların uygun olduğu on iki durumu (dizilerin yinelemeleri, dizilerin uzunlukları, tamsayıların boyutları) ve bu durumların her biri için potansiyel olarak on iki farklı tamsayı değil eriyle, muhtemelen her biri ile belirlediği durumda olacaktır. protokolün çeşitli yerlerinde on iki değil er kullanılıyor. Bu yine "paketleme" için iyi bir durumdur. Bir sınıf tanımlayabiliriz:

```
SINIRLAR ::= SINIF {&kısa-dizeler
  &uzun dizeler
  &normal-ints
  &long-ints INTEGER, &number-of-orders INTEGER}
  INTEGER,
  INTEGER,
  INTEGER, &very-
  SÖZDİ Zİ Mİ İ LE
  {STRG &kısa dizeler, LONG-STRG &uzun dizeler,
  INT &normal-ints, LONG-INT &çok uzun-ints,
  Sİ PARİ Ş LER &sipariş sayısı}
```

ve rutin olarak ve basit bir şekilde bu sınıfın bir nesne kümesini tanımladığımız her türün bir kukla parametresi haline getirin, onu SEQUENCE, SET veya CHOICE yapılarındaki herhangi bir türün gerçek bir parametresi olarak geçirin. Daha sonra protokolümüzdeki çeşitli yerlerde ihtiyaç duyduğumuz alanları kullanabiliriz. Bazı tür tanımlarında, bunların hiçbirini kullanmayabiliriz ve bu tür için dummy parametresi gereksiz olabilir (ancak yine de yasaldır) veya alanlardan birini veya ikisini veya (muhtemelen nadiren) hepsini kullanabiliriz.

Üst düzey türümüzü tanımladığımız noktada (bu kitabın ilk böümlerinde tartıştığımız gibi genellikle bir SEÇİ M türü), sınırlarımızı belirleyebilir ve bunları gerçek bir parametre olarak sağlayabiliriz. Dolayısıyla, "Wineco-protocol" en üst düzey tipimizse, şunlara sahip olabiliriz:

sınırlar BOUNDS ::= {STRG 32, LONG-STRG 128, vb. }

Wineco protokolü {BOUNDS:bounds} ::= SEÇİ M
 {sipariş [UYGULAMA 1] Stok sipariş i
 [UYGULAMA 2] Satış getirisi
 vb. }
 {BOUNDS:bounds}, satış
 {SINIRLAR:sınırlar}

Hiç ş üphe yok ki, "Sınırların" her yerde dış a ve içe aktarılması koşuluyla, doğrudan kullanılabileceğine göre, bu ş eyleri parametre olarak aktarmanın ne anlama var? Bu durumda cevap "Fazla değil!". Herhangi bir tür için, herhangi bir sınır kümesi her zaman sabitlenecektir, o zaman onu bir parametre yapmanın bir anlamı yoktur, bunun yerine daha basit ve daha açık bir belirtimle birlikte bir genel referans adı kullanılabilir. Ama bir sonraki bölüm okuyun!

4.2 Soyut sözdiziminin parametreleri

Protokol tasarımcıları, yalnızca tek bir yerde tanımlanmış olsalar ve basit içe/dış a aktarma yoluyla veya ek olarak sahte parametreler kullanarak aktarılışalar bile, bir protokol tanımının gödesindeki sınırları sabitleme konusunda genellikle tereddütlü davranışlarılar. Tereddütün nedeni, sınırların iki nedenden dolayı bir protokolü "tarih lendirebilmesi"dir: Birincisi, başlangıçta yeterli görünen ş ey (örneğin, "ayrintıların" yineleme sayısı için)

Yani bazı ş eyleri uygulamaya bağlı bırakmak mı istiyorsunuz? Korkak! Ancak en azından açık hale getirin (ve farklı uygulamalar arasında birlikte çalışmayla yardımcı olmak için istisna işlemeyi tanımlayın). Soyut sözdiziminin parametreleri bunu yapmanıza izin verir, ancak bunlar nadiren kullanılan bir özelliktir.

Bölüm I'deki Ş ekil 13'teki "Stok için sipariş" türündeki SIRALI, on yıl sonra iş genişlediğinde ve birleşmeler meydana geldiğinde pekala yetersiz kalabilir! İkincisi, sınırlar genellikle sınırlı bellek kapasitesine sahip makinelerde veya çok uzun tamsayı değerleri olan hesaplamalar için destek olmadan uygulama çabasını kolaylaştırmak için uygulanır. Bununla birlikte, bu tür teknolojik sınırlamaların zamanla ortadan kalkma gibi bir alışkanlığı vardır. Bu nedenle, on beş yıl önce birçok tasarımcı 64K sekizliyi aşan mesajlara sahip olmanın mantıksız olduğunu düşündürken, bugün çoğu makinedeki uygulayıcılar bir megabayt uzunluğu undaki mesajları işlemekte sorun yaşamıyorlar. (Burada bir istisna, hafızanın hala çok sınırlı olduğunu akıllı kartlar için veri formatlarının özellikleri olacaktır. Bu, ASN.1'in kullanılıldığı bir alandır.)

Öyleyse, sınırlarımızı ana belirtme sokmak istemiyorsak ne yapmalıyız? Onları dışarıda bırakmak mı? Bu, ş üpbesiz, bazı sistemlerin diğer bazı sistemlerin ürettiği boyuttaki ş eyleri işleyemeyeceği, birlikte çalışma sorunlarına neden olacaktır ve bunu ASN.1 belirtimimizde olası bir sorun olarak işaretlemiyoruz bile.

Bir "sınır" parametresi sağlamak, ancak bunun için hiç bir zaman değil er ayarlamamak bu soruna yardımcı olabilir. "ABSTRACT-SYNTAX" göstergesini kullanarak en üst düzey tipimizi belirtebileceğimizi Kısım I Bölüm 3'teki Ş ekil 21'de görmüşük. Bunu şimdilik yukarıda gelişen parametreleşmiş tırılmış Wineco protokolümüzle tekrar edelim:

wineco-abstract-sözdizimi {BOUNDS:bounds} ABSTRACT-SYNTAX ::=
 {Wineco-protocol {BOUNDS:bounds} TANIMLAYAN vb.}

Şimdilik soyut sözdizimimizi bir parametre listesiyle tanımlıyoruz. Soyut sözdiziminin parametrelerine sahibiz. ASN.1, bu tür parametrelerin yalnızca kısıtlamalarda kullanılması koşuluyla buna izin verir. Gerçek sınır uygulama olduğu için bu kısıtlamalara değil işken kısıtlamalar denir.

bağ ımlı. Ancak ş imdi elde ettiğ imiz önemli kazanım, bu uygulama bağ ımlılığının çok açık ve spesifik hale getirilmiş olmasıdır. Bir deň iş ken kısıtlamamız olduğunda, yerel sınırları aşan bir malzeme alındığıında amaçlanan hata iş lemeyi belirtmek için normal olarak istisna iş aretişini sağlarız.

OSI çalışmasında, Uluslararası Standartlaşırılmış Profiller (ISP'ler) ve Protokol Uygulama Uygunluk Bildirimleri (PICS) kavramı vardır. ISP'lerin amacı, bir protokolü belirli toplulukların ihtiyaçlarına göre uyarlamak veya farklı uygulama sınıfları (örneğin küçük, orta, büyük) tanımlamak için seçenekler ve parametre deň erlerinin bir profili sağlamaaktır. PICS'in amacı, uygulayıcılara protokolün uygulamaya bağlı böümlerde yaptıkları seçenekleri belirtmeleri için bir format sağlamaaktır. Açıkça, soyut sözdiziminin parametrelerinin kullanılması, bu iki görevde de yardımcı olur; bu parametreler için deň erler ya bir profilde (bir uygulamanın daha sonra uygunluk iddiasında bulunur) ya da doğrudan bir uygulama için PICS'de belirtilir.

Soyut sözdiziminin parametreleri (tüm deň iş ken parametrelerdeki istisna iş aretişleri ile), potansiyel birlikte çalışma sorunlarının olduğu alanları belirlemek için çok güçlü bir araç sağları, ancak (en azından bu yazar için) bu özelliklerin bugüne kadar yaygın olarak kullanılmaması üzücüdür..

4.3 Gereksinimlerini açık hale getirme

4.3.1 Tİ P-TANIMLAYICI sınıfı

Çok yaygın bir Bilgi Nesnesi Sınıfı, biri sınıfın bir nesnesini tanımlamak için bir nesne tanımlayıcısını tutan ve diň eri o nesneyle ilişkili bir türü tutan yalnızca iki alana sahip olandır.

Bu sınıf aslında ASN.1'de TYPE-IDENTIFIER sınıfı olarak önceden tanımlanmışdır (yerleşik). Şu şekilde tanımlanır:

```
Tİ P-TANIMLAYICI ::= SINIF {&id NESNE
    TANIMLAYICI BENZERSİ Z,
    &Tip }
    İ LE SÖZDİZİMİ Mİ {&Tür &id TARAFINDAN TANIMLANAN}
```

Bu sınıftan yararlanan birçok protokol vardır. Protokollerin geniş letilebilirliği ve yönelik çok esnek bir yaklaşımın temel taşıdır.

4.3.2 Bir örnek - X.400 başlıklar

(ROSE'da olduğu gibi, aşağıda X.400'ün tam bir kopyası deňildir).

X.400'de (bir e-posta standartı), bir mesaj için "başlık" kavramı vardır. Çok çeşitli başlıklar tanımlanmışdır. X.400'ün ilk sürümünde, bunlar bir DİZİ içindeki türler olarak fiziksel olarak bağlanmıştır, ancak sonraki sürümlerde yeni başlıkların ekleneceği kısa sürede anlaşıldı.

Tabii ki SEQUENCE, geniş letilebilirlik üç noktasını, üç nokta üzerinde tanımlı istisna işareti ile, sürüm 1 ve 2 arasında birlikte çalışmayı sağlayarak ekleyebilirdi, ancak alternatif bir yaklaşım, başlıkların şu şekilde tanımlamaktır:

Geniş letilebilirlik sağlama çok iyi, ancak uygulamalarınızın belirtildiği indirimlerin herhangi uygulamak zorundalar mı, ancak daha fazlasını ekleyebilirler mi? Seçtiğini bitleri uygulayın ama daha fazlasını EKLEMEYİN?

Ya da her ikisinin bir kombinasyonu? Çok u durumda, bu gereksinimleri resmi olarak ifade edebilirsiniz.

ASN.1'deki iki yerleşik sınıfından biri (diň eri ABSTRACT SYNTAX'tır) ve oldukça iyi kullanılır.

ve gerçek başlıklar şunu ekildedir:

```
Başlık türü {HEADER-CLASS:Supported-headers} ::=  
SIRALI HEADER-CLASS.&id  
( {Desteklenen-başlıklar} !100 ), HEADER-CLASS.&Type ( {Desteklenen-başlıklar}  
{kimlik bilgisi @id}!101 )
```

}

İstisna iş leme 100 ve 101, protokol tanımının metninde belirtilecektir. 100'ün iş lenmesi muhtemelen "sessizce yok sayma" ve 101'in (kötü bir tür) "bir hata döngüsü gönderme ve aksi takdirde yok sayma" olacaktır.

Soru şunu ki, sonunda Başlık türü için gerçek bir parametre sağladığımızda, ne sağlıyoruz? Bazı seçenekleri inceleyelim.

Protokolün bu sürümünde tanımlanmış bazı başlıklar kesinlikle olacaktır ve şüphesiz sonraki sürümlerde daha fazlasını ekleyeceğiz, bu nedenle önce aşağıdaki gibi bir geniş letilebilir bilgi nesnesi kümeleri tanımlayacağımız:

```
Tanımlı Başlıklar HEADER-CLASS ::= {header1 | başlık2 |  
başlık3 , ..., başlık4 }
```

sürüm 2'de header4 eklendi.

Ancak protokolümüz için gerçek parametre olarak ne sağlıyoruz? Önce en genel durumu ele alalım. Soyut sözdiziminin iki parametresini sağlamayı düşünelim, her ikisi de HEADER-CLASS sınıfının nesne kümeleri. Biri "Uygulanmadı" ve diğer eri "Ek başlıklar" olarak adlandırılır. Aşağıdaki kararlara bağlı olarak bunlardan birini veya her ikisini sağlamak veya hiç birini sağlamak istemeyebiliriz. Muhtemelen fikri anladığınızda düşüneniz!

Şimdi, üstbilgileri desteklemek için uygulama gereklilikleri hakkında alabileceğimiz çeşitli olası görüşlere bakalım.

4.3.3 Basit bir DİZİ kullanımı

Sabit bir başlık kümeleri tanımlamak istediği imize karar veriyoruz, tümü uygulanacak, ekleme yok ve daha sonra asla değişim yapmayacağı üzere. Bazı başlıklar gereklili olacak, diğerleri isteğe bağlı olacaktır.

İlk seferde doğrudu anladık!

Bu durum kolaydır ve Bilgi Nesnesi Kümelerine ihtiyacımız yoktur, basitçe şunu kullanırız:

```
Başlıklar ::= SIRALI  
{başlık1 Üstbilgi1-türü -eklenmelidir-, Üstbilgi2 Üstbilgi2-türü İSTEĞE BAĞLI,  
vb. }
```

Bu basit ve anlaşılır, ancak çok esnek değil. Hangi başlıkların sağlanması gereken kararların (e-posta başlıklarında olduğu gibi) geçici olduğunu ve gelecekte değişim gerekebileceği durumlarda, bu iyi bir yol DEĞİLDIR!

Bu durumda, bir İSTEĞE BAĞLI üst bilgi grubunda hangi başlıkın kodlandığıının tanımlanmasının esas olarak (BER'de) etiket değişimleri kullanılarak yapıldığıını unutmayın. (PER'de biraz farklıdır - bir bit haritası, hangi başlıkın belirli bir konumda kodlandığıını tanımlar).

4.3.4 Geniş letilebilir Dİ Zİ kullanımı

E-posta başlıklarını söz konusu olduğunda, daha sonra daha fazla başlık türü eklemek istememiz kuvvetle muhtemeldir, bu nedenle SEQUENCE'ı geniş letilebilir yapmak daha iyi bir yaklaşım olacaktır. Ayrıca, sürüm 1 sistemlerinin bir sürüm 2 sisteminden başlıklar gönderildiğiinde nasıl davranışını (ve bir sürüm 1 olduğunda içi sürüm 2'de zorunlu olan başlıklar eksikse sürüm 2 sistemlerinin nasıl davranışını gerektiği) bilmek için istisna işlemeyi belirtmeliyiz. başlıklar üreten sistem).

Kontrol bizde. Biz ne dersek onu yapacaksın. Hiçbir şeyle kalmayacağımız, ancak daha sonra daha fazlasını ekleyebiliriz.

4.3.5 Bir bilgi nesnesi seti tanımına geçiş

Şimdi, görünen karmaşıklıkta oldukça büyük bir sıkrama yapıyoruz ve yukarıda tanıttığımız "Başlıklar" türünü kullanıyoruz, yani:

Kendimize daha fazla seçenek sunarak, ancak yine de kontrolü elimizde tutuyoruz.

```
Başlık türü {HEADER-CLASS:Headers} ::= Dİ Zİ Dİ Zİ Sİ
{tanımlayıcı HEADER-CLASS.&id({Başlıklar}!100),
veri HEADER-CLASS.&Type({Başlıklar} {@tanımlayıcı} !101)}
```

Artık belirli bir başlık in türünü tanımlamak için bir nesne tanımlayıcı kullanmaya geçtiğimizde potansiyel olarak artık herhangi bir başlık türünün farklılığı erlerle birden çok kez sağlanmasına izin veriyoruz. Ancak bir başlık in isteği ebağlı olup olmadığıını söyleme yeteneğimizi kaybettik ve hangi başlıkların birden çok kez görünebileceği in söylemenin kolay bir yolu yok.

HEADER-CLASS'imize alanlar ekleyerek bu sorunları çözebiliriz. Yani onu Tİ P-TANIMLAYICI olarak tanımlamak yerine şu şekilde tanımlayabiliriz:

```
HEADER-CLASS ::= CLASS {&id NESNE
TANIMLAYICI BENZERSİ Z,
&Tip,
&Gerekli BOOLE VARSAYILAN DOĞRU,
&Katlar BOOLEAN DEFAULT TRUE)
SÖZDİ Zİ Mİ İ LE {&Tür &id TARAFINDAN TANIMLANIR,
[GEREKLİ Dİ R &Gereklidir],
[KATLARA &KATLARA İ Zİ N VERİ Lİ R]}
```

Artık (her başlık nesnesi tanımlandığıında) isteği ebağlı olup olmadığı in ve birden çok oluşumuna izin veriliyor ve verilmemiş in belirtilmiş. Tabii ki, bir Dİ Zİ kullandığımızda, isteği ebağlılığı işaretleyebildik ve belirli öğelerin etrafına Dİ Zİ koyarak katlara izin verildiği in belirtilmiş. Ancak, bilgi nesnelerini kullanan yaklaşım, hepsi istiyorsak muhtemelen daha basittir ve daha fazla seçenek in öünü açar.

Tabii ki, "Tanımlanmış Başlıklar" bilgi nesnesi kümelerini tanımladığımızda, sürüm 2'deki ekleme olasılığının belirterek geniş letilebilir hale getireceğiz ve sürüm 1 sistemlerine ne yapacaklarını söylemek için üç noktaya bir istisna belirtimi koyacağımız anladıkları başlıklar alın.

X.500'ün benzer bir durumda yaptığı gibi, aslında bundan daha ileri gidebiliriz: HEADER-CLASS'e bir başlık in "kritikliği in" tanımlayan başlık ka bir alan koyabilir ve taşınmak için "Başlıklar tipinde" bir alan sağlayabiliriz. Bu değeri istisna belirtimiz, mesajda onunla ilişkili "kritiklik" alanının değerine bağlı olarak, bilinmeyen başlıklar için farklı istisna işlemeye tanımlayabiliriz.

SEQUENCE ile sahip olduğu umuz oldukça kısıtlı işlevsellikten bir miktar ilerledik.

4.3.6 "Baş İıklar" nesne kümesi

Geniş letilebilir bir "Tanımlanmış Baş İıklar" yalnızca, sürüm 2'ye yeni malzeme eklediğimizde sürüm 1'in ne yaptığı üzerinde bize kontrol sağlar. Uygulamaların (muhtemelen bazı kullanıcı grubu veya satıcıya özel olarak) kabul edebileceğini ve yeni baş İıklar ekleylebileceğini hiç bir şekilde söylemez. Ayrıca, x sürümüne uymak için, x sürümü için "Tanımlı Baş İıklar"da listelenen tüm baş İıkları desteklemeniz gerekiyor. söyler.

Şimdi uygulayıcılara esneklik veriyoruz. Soyut sözdiziminin parametrelerini kullanıyoruz.

Ancak, tanımladığımızı varsayıyalım:

```
Desteklenen Baş İıklar
{HEADER-CLASS:Ek Baş İıklar,
 HEADER-CLASS:Excluded-Headers} HEADER-CLASS:-
{ (Tanımlı Baş İıklar | Ek Baş İıklar)
  Hariç Tutulan Baş İıklar HARI } Q}
```

burada "Ek-Baş İıklar" ve "Hariç Tutulan-Baş İıklar", yukarıda açıklanan soyut sözdiziminin parametreleridir ve "Desteklenen-Baş İıklar", "Baş İlk-tipi" örneğiinde "Baş İıklar" kukla parametremiz için gerçek parametre olarak sağlanır. ".üst düzey PDU'muzu tanımladığımızda (ve ardından "Baş İlk türü" kısıtlamalarında nihai kullanım için aktardığımızda).

Her zamanki gibi, eğer istersek, iki nesne setini yeni bir nesne sınıfının bir nesne seti olarak bir araya toplayabilir ve soyut sözdiziminin her iki belirtimi de kapsayan tek bir parametresini oluşturabiliriz.

Yukarıdaki tanımla, bazı tanımlı baş İıklarımız olduğuunu açıkça söylüyoruz, uygulayıcılar diğerlerini destekleyebilir ve aslında tanımlı baş İıkların bazılarını desteklememeyi seçebilirler. Toplam özgürlük! Muhtemelen toplam anlaşımı! Ancak çoğu uygulama, muhtemelen tanımlanmış baş İıkların çoğuunu uygulamayı seçecektir ve istisna işlemeye, birkaçını kaçırınlarla (nedeni ne olursa olsun) birlikte çalışma sorunlarıyla başa çıkmalıdır.

Okuyucunun Desteklenen Baş İıkların uygun bir tanımını yazması (basit!) bir alıştırma olarak bırakılmıştır.

- a) ek baş İıklara izin vermeye, ancak tanımlanmış tüm baş İıklar için destek gereklimeye karar verin; veya
- b) bazı tanımlı baş İıkların desteklenmemesine izin vermeye karar verin, ancak uygulamaya bağlı veya satıcıya özel eklemelere izin vermeyin.

Elbette, günün sonunda, her şeyi uygulamak için bir gereksinimi asla UYGULAYamazsınız ve insanların standart bir protokolü genişletmesini engelleyemezsiniz. Ancak, o zaman Standarda uymadıklarını çok net bir şekilde YAPABİLİRİSİNİZ. ASN.1 bunu yapmak için araçlar sağlar.

4.4 (boş) geniş letilebilir bilgi nesnesi seti

Geniş letilebilir olsa bile, çoğu protokolde üye içermeyen bir bilgi nesnesine sahip olmak pek mantıklı değil:

{...}

Bunu okurken **S ekil 999'a (Devlet Sağlık Üyarısı)** geri dönmelisiniz.

Bu, muhtemelen bu kitabın tamamındaki en tartışmalı metindir!

Bu gösterimi "soyut sözdiziminin bir parametresi" için bir kısaltma olarak kullanmak oldukça yaygın bir uygulama haline geldi (artık Standart'ta metin tarafından destekleniyor). Bu bir kısıtlama olarak kullanıldığıında, basitçe belirtimin eksik olduğuunu ve neyin desteklenip desteklenmediği ile ilişkini belirtim için başka bir yere bakmanız gerekiyor in söyler.

Buna "dinamik olarak geniş letilebilir nesne kümesi" denir, fikir, uygulamaların bir iletişim im örneği içinde içerdığı ini düşündükleri nesneleri belirleyeceğine ve gerçekte de (yağmur yağışının yağmadığını veya bağlılığını olarak!) bazı nesneleri kabul edip reddedebileceğini idir. farklı zamanlarda.

Bu yazarın bu yapının kullanımını onaylamadığı ve izlenimine kapılırsanız, çok da yanılmayacaksınız!

Soyut sözdiziminin parametreleri tarafından sağlanan (çok daha açık bir şekilde) ötesinde hiç bir işlevsellik sağlama. Ancak bir avantajı var. Soyut sözdiziminin parametreleri en üst düzeyde görünür ve kullanılacakları noktaya ulaşana kadar sonraki iç içe türlerde parametre olarak geçirilmeleri gereklidir. Bu, bir belirtimin boyutunu artırır ve bazen daha az kolay okunabilir hale getirebilir. (Bir zamanlar ASN.1'e "global parametreler" kavramını eklemek için çalışılmıştı ve öne sürüldü. Bu, üst düzey bir parametrenin, bir türden tür bir dizi olarak geçirilmeden, herhangi bir yerde kullanılabilen normal bir referans adı haline gelmesini etkili bir şekilde etkinleştirilebilirdi. gerçek kukla parametreler Ancak bu çalışılmıştır hiçbir zaman ilerleme kaydetmedi).

Bir kısıtlamada "..."'nın gösteriminin kullanılması, bu kısıtlamanın uygulamaya bağlı olduğuunu en alt düzeyde doğrudan ifade eder. Ancak yine karşılık tarafta - üst düzey tanıma bakarak soyut sözdiziminin (etkin bir şekilde parametreleri olduğuunu, yani belirtimin eksik olduğuunu söyleyemezsiniz. "...").

Bu yazarın tavsiyesi **BU İÇİN NŞ AATI KULLANMAYIN**. Ancak, onunla karşılaşınca ne anlamına geldiğini bilmeniz gereklidir ve onu kullanan birçok özellikle vardır (soyut sözdiziminin kullanım parametrelerinden daha fazlası).

X.681'de, geniş letilebilir hale getirilen **HERHANGİ** bir nesne kümesinin, o nesne kümesi tarafından dayatılan kısıtlamalar dikkate alındığında nesnelere rastgele eklemeler ve ilişkiler yapılabileceğini ima ettiğini söyleyen bilgilendirici bir ek (Standartın bir parçası değil) vardır. Bu yazar ASN.1 Standartlarını pek sık eleştirmez - **İ**çerideki metnin çoğu "ben yazdım!" Ancak bu ek, kütü bir tavsiye vermektedir ve Standardın göndesindeki normatif metin tarafından desteklenmemektedir.

Öyleyse ... belirli bir belirtimin, boş olmayan geniş letilebilir bir küme kullandığında ne anlamına geldiğini nasıl karar verirsiniz? Spesifikasyonu dikkatlice okuyun ve genellikle açık olacaktır. "..."'ı kullanırsanız, muhtemelen tüm geniş letilebilir nesne kümelerinin uygulamaya bağlı eklemelere veya istisnalara sahip olabileceğini söyleyeceksiniz (ancak o zaman, yorum dışındaki belirli durumlarda buna karşılık koymayan hiçbir yolu yok). (X.400 gibi), soyut sözdiziminin açık parametrelerine sahipse, kesinlikle bunu ima ETMEYECEKTİ R ve "Başlıklar" için önceki yan tümcede verilen yorumu kullanmalısınız.

5 "delikler" iç in diğ er hükümler

ASN.1 spesifikasyonlarındaki boş lukları destekleyen, esas olarak bilgi nesnesi kavramından önceye dayanan baş ka mekanizmalar vardır. Bunları kısaca tartış mamız gerekiyor.

İ kimiz de - yoruluyor olmalıyız! Söylenecek fazla bir ş ey yok ama yine de var. Kısa tutmaya çalış acağ iz. Bu zor bir ş ey değ il, ama KULLANILMIŞ TIR ve ONEMLİ Dİ R.

5.1 HERHANGİ

Bunun ş ıhret iç in iki önemli iddiası var. Birincisi, orijinal 1984 ASN.1 Spesifikasyonlarında kara delikler iç in tek destekti! İkincisi, 1994'te geri çekildi ve bazı ASN.1 kullanıcıları arasında oldukça büyük bir kargaş aya neden oldu.

(Kötü?) İ lk deneme mi? 1984'te yapabildig imizin en iyisi buydu. O zamanlar boş luklar pek anlaş ilmiyordu.

Bir SEQUENCE veya SET'e "HERHANGİ Bİ R" yazdınız, bu kelimenin tam anlamıyla herhangi bir ASN.1 türünün HERHANGİ 'nin yerini almak üzere oraya yerleş tirilebileceğ i anlamına geliyordu. İ lk CCITT spesifikasyonlarında sıkılıkla ş u yorumla birlikte yer alıyordu:

-- Daha fazla çalış ma iç in --

Bu yorum, eksik bir spesifikasyonda yalnızca bir yer tutucu olduğunu açıkça belirtti. Genellikle bu gibi durumlarda SEQUENCE öğesi ş unu okur:

HERHANGİ Bİ R İ STEĞE BAĞI

yani temelde bu unsurun uygulanabilir olmadığı ini biliyordunuz - HENÜZ!

Bu ş ekilde kullanıldığı inda bir zararı olmadı ama muhtemelen pek de yararlı olmadı. Geniş letilebilirlik üç noktasını kullanarak bugün elde ettiğ imiz iş ıevselliğ in bir kısmını sağ ladi. "Daha sonraki bir sürümde gelecek daha çok ş ey var, ancak henüz ne olduğunu gerçekte bilmiyoruz" dedi.

Bununla birlikte, baş ka kullanımlar da vardı. Bunlardan biri yakın zamana kadar X.500'deydi ve burada bir Dİ Zİ öğesi ş unu okuyordu:

iki taraflı bilgi HERHANGİ Bİ R İ STEĞE BAĞI

Buradaki amaç, uygulamaya bağı lı ek bilgilerin, bu bilgiler iç in ASN.1 türünün baş ka bir yerde (ilgili topluluk veya satıcıya özel) belirleneceğ i yerde geçirilmesine izin vermekti. Birkaç satıcı veya topluluk, bu alanı dolduracak tür iç in farklı özellikler ürettiyse, alanın ne söylediğ ini belirlemek iç in genellikle arayan adrese bakarsınız. (Yine bir deliğ in içeriğ i iç in bir tanımlayıcı sağ lamanın - standart olmayan - baş ka bir yolu!)

Uygulamada, bu alan hiçbir zaman X.500 uygulayıcıları tarafından uygulanmadı.

Alanı dolduran türü (ve anlamını) belirlemek iç in baş ka bir seçenek de yağ mur yağ ip yağ madığ ini görmek olabilir, ancak kimsenin bu özel mekanizmayı "delik tanımlama" iç in kullandığı ini sanmıyorum!

5.2 TARAFINDAN TANIMLANANLAR

Bu, 1986/88'de ANY'yi destekleme giriş imiydi. Şimdije kadar, bir kara deliğin protokolde kesinlikle kendisine yakın bir yerde, dolduran gerçek türün (ve - daha da önemlisi - bu türle ilişkili semantikin) tanımına işaret edecek bir değere sahip olması gerektiğiine dair bir kabul vardı. Aniden delik biraz daha az siyah oldu!

Şafak sükür (ama sadece biraz!).
Herhangi bir deliğin, deliğin neyin (ve hangi semantikle) doldurduğuunu belirleyen bir mekanizmaya sahip olması GEREKİR.

(Kömür mahzenindeki iş ik, 1994 spesifikasyonunda bilgi nesneleri göründüğü içinde gerçekten açıldı. Bilgi nesnesi kavramlarının tanıtılması ile bir kömür mahzeninde bir iş işin açılması arasındaki analogi işin Bancroft Scott'a minnettarım. İlk yorumu o yaptı, kim olduğunu unutan biri "Kulağ a oldukça dramatik geliyor. Dramatik şeyle gelgit dalgalarına neden olabilir." Cevap iyi oldu! Bilgi nesneleri HERHANGİ BİR TANIMLANANIN yerini kolayca almadı. Sonunda yaptılar, ama dalgaların yatışması yaklaşık yedi yıl sürdü!)

ANY DEFINED BY ile tipik bir SEQUENCE artık şunları içerebilir:

tanımlayıcı NESNE TANIMLAYICI, delik
tanımlayıcı TARAFINDAN TANIMLANAN HERHANGİ BİRİ

Okuyucu, bunun artık "nesne sınıfından gelen bilgiler" yapıları arasında ilişkisel bir kısıtlama (@ notasyonu) kullanılarak sağlanan iki alan arasında aynı türden bağlanmayı sağladığını, ancak herhangi bir bilgi nesnesi kümlesi referansından yoksun olduğunu fark edecktir. Kesin bağlanmayı, "HERHANGİ BİR" alanını doldurabilen türleri ve bu türlerle ilişkili anlambilimi tanımlayın.

ANY TANIMLI BY göstergesi kullanılarak belirtilebilen bağlanılarda (çok ciddi) kısıtlamalar da vardı, bu da bazı mevcut spesifikasyonların ANY'den ANY TANIMLI BY'ye taşınmasını imkansız hale getiriyordu, hatta (bir yerde) bir alanları olsa bile. HERHANGİ bir deliğin içeriğini tanımlayan protokollerinde.

5.3 HARİ CI

EXTERNAL 1986/88'de tanıtıldı ve hala bizimle. Adı, insanların ASN.1'in dışındaki materyalleri, yani ASN.1 kullanılarak tanımlanmayan materyalleri (örneğin, bir GIF görüntüsü) gömmek istedikleri gerçekliğin tanınmasıdır. Bununla birlikte, HERHANGİ ve HERHANGİ BİR TARAFTA TANIMLANANIN daha iyi bir versiyonu olması amaçlanmışlığı, çünkü deliğin içinde ne olduğunu unun deliğin kendisi ile özdeşleşmesini kapsıyordu.

Ancak ASN.1 kullanılarak tanımlanmayan materyali dahil etmek istiyorsunuz. Ve malzemenin türünü ve kodlamasını belirlemek istiyorsunuz. OCTET STRING veya BIT STRING ve ayrı bir tanımlayıcı alanı kullanarak kendi numaranızı atın. Bu işe yarar. Ancak HARİ CI, hazır bir çözüm sağlamaya çalışır.

HARİ CI, ASN.1'in OSI ailesinin bir parçası olduğunu ve (diğer olasılıkların yanı sıra) bir "sunum bağlamı" kullanarak delik içerişlerinin tanımlanmasını tanıdığı zaman tanımlandı.

OSI'nin Sunum Katmanı olanakları kullanılarak müzakere edilir. Bu mekanizma muhtemelen hiç bir zaman gerçek bir uygulama tarafından kullanılmadı.

EXTERNAL ayrıca bir şıhret iddiasında bulunabilir: Tanımı neredeyse kesinlikle herhangi bir ASN.1 belirtiminde "ObjectDescriptor" türünün kullanıldığı tek yerdir! (Ama İSTEĞE BAŞLIYOR - ve hiç bir uygulamanın bir HARİ Cİ içinde bir "ObjectDescriptor" değil eri aktarmadığı ina bahse girerim.)

Son olarak, HARİ Cİ, soyut ve transfer sözdizimlerinin anlaşılmasıının ilk günlerinde ortaya çıktı ve (OSI Sunum Katmanını kullanma seçeneğini hariç tutarsanız), malzeme içi soyut ve transfer sözdiziminin kombinasyonunu tanımlamak için yalnızca tek bir nesne tanımlayıcı değil eri kullandı. Bu deliği doldurdu. Bugün, genel olarak, boş luftaki soyut değil erler kümescini (örneğin, bunun bir durağın resim olduğu) bir nesne tanımlayıcısı ile ve bu değil erlerin kodlanması (resmin kodlanması) bir nesne tanımlayıcısı ile tanımlamanın uygun olduğu una inanıyoruz. ayrı nesne tanımlayıcısı. Bu nedenle EXTERNAL, 1988 spesifikasyonunda (1986/88'deki orijinal tanıtımından değil iş meden) kalırken, ciddi kusurları vardır ve yeni spesifikasyonlar, ASN.1 olmayanları taşıma imkânı istiyorlarsa bunun yerine "GÖMÜLMÜŞ PDV" (aşağıda açıklanmışdır) kullanmalıdır. tanımlanmış malzeme

5.4 GÖMÜLÜ PDV

EMBEDDED PDV, 1994 yılında piyasaya sürüldü. Oldukça basit bir şekilde, bir "geliş tirme" giriş imiydi. HARİ Cİ. EXTERNAL'in herkesin umursadığı tüm işlevlerine sahiptir. Hiç kimse kimin kullanmadığı Nesne Tanımlayıcıdan kurtuldu ve boş luğunu dolduran malzemenin soyut sözdiziminin ve transfer sözdiziminin (kodlama) tanımlanması için ayrı nesne tanımlayıcılarına izin verdi (ancak gerektirmedi).

İlk seferde doğrudu yapmak neden bu kadar zor? EMBEDDED PDV'ye rağmen sadece EXTERNAL'in eksikliklerini gideriyor. EXTERNAL 1986/88'de oldukça iyi görünüyordu, ancak 1994'te yeniden takılması gerekiyordu.

Belki daha da önemlisi, bir protokol tasarımcısının "delik" için özet ve aktarım sözdizimlerinden birini veya her ikisini birden (statik olarak) belirtme (kısıtları kullanarak) yeteneğini içeriyecektir.

Bunun önemli bir kullanımı, EMBEDDED PDV'nin bir türün şıfrelmesini taşıma imkânı kullanıldığı türün (delik içeriğinin soyut sözdizimi) statik olarak belirtildiği ve şıfrelme mekanizmasının (aktarım sözdizimi) iletişim zamanında aktarıldığı güvenlik çalışmasıdır.

Uygun koşullarda, bir tasarımcı boş luğunu dolduran şeyin hem soyutunu (malzemenin türü) hem de transfer sözdizimini (kodlama) statik olarak belirleyebilir. Bu yapılsa, EMBEDDED PDV, gömülü malzemenin etrafındaki uzun bir sargı dışında hiç bir ek yük üretmez.

İsim hakkında kısa bir kelime. (Yine Şekil 999). Bu tür için kesinlikle kötü bir isim. "GÖMÜLMÜŞ" iyidir. Gömülü malzeme alabilen bir deliğin temsil eder. Ama "PDV"? Çok u okuyucu "PDV" terimiyle asla tanışmamış olacaktır. Aslında "Sunum Veri Değeri" için standarttır ve OSI Sunum Katmanı Standardı tarafından Uygulama Katmanı ile Sunum Katmanı arasında iletilen bilgi birimini tanımlamak için kullanılan terimdir veya (burada verilen ASN.1 açıklamasıyla daha ilgili terimlerle). 1) bazı soyut sözdizimlerinden soyut bir değil er (ASN.1 kullanılarak tanımlanması gerekmek).

Bu yüzden isim hakkında endişelenme! ASN.1 türü olarak tanımlanan gömülü malzeme için, muhtemelen deliklerinizi işlemek için bilgi nesnesi ile ilgili kavramları kullanmak istersiniz. Ancak gömmek istediğiniz malzeme ASN.1 kullanılarak tanımlanmamışsa, GÖMÜLMÜŞ PDV'yi kullanın.

5.5 KARAKTER Dİ Zİ Sİ

KARAKTER Dİ Zİ Sİ aslında sadece GÖMÜLÜ PDV'nin özel bir halidir ve ASN.1 Standardında bu türlerin belirtiminde birçok paylaşılan metin vardır.

CHARACTER STRING, olası tüm ihtiyaçları SONSUZA KADAR karşılayacak bir karakter dizisi türü üretmeye yönelik (baş arısal!) bir giriş imdi. ASN.1 Standardının koruyucularının (dünyada yeni karakter setleri ve kodlamalar ortaya çıktıktan) "ASN.1'i değil iş tirmemize gerek yok, KARAKTER Dİ Zİ Sİ kullanın" demesini mümkün kılmak amaçlandı.

Başka bir (oldukça özel) delik ş ekli. Delikte bulunan repertuar ve kodlama duyurusu ile herhangi bir kodlama kullanarak herhangi bir repertuardan karakter dizilerini tutmak için tasarlanmıştır. Harika bir fikir, ama asla tam olarak tutmadı. Ama yine de... onu kullanmak isteyebilir misin?

CHARACTER STRING türü, soyut kavramını ve aktarım sözdizimini genişletir. "Karakter soyut sözdizimi" (tüm değil erleri belirli bir karakter kümesinden karakter dizileri olan soyut bir sözdizimi) ve "karakter aktarım sözdizimi" (belirli bir karakter özetindeki tüm olası diziler için kodlamalar sağlayan bir aktarım sözdizimi) terimini tanıtır. sözdizimi).

Biraz daha az teknik terimlerle ifade edersek, bir karakter soyut sözdizimi nesne tanımlayıcısı, bir karakter repertuarını tanımlar ve bir karakter aktarım sözdizimi OBJECT TANIMLAYICI, bu karakterlerin dizileri için bir kodlamayı tanımlar.

Kısıtlanmamış, CHARACTER STRING türündeki bir kodlama, karakter soyut sözdizimini (repertuar) ve iletilen her dizeyle karakter aktarım sözdizimini (kodlama) tanımlayan iki nesne tanımlayıcısını içerir. Bu talihsiz(!) bir ek yüktür, çünkü aşağıda gibi yapar

KARAKTER Dİ Zİ Sİ SIRASI

(Dİ Zİ Sİ Nİ N her açılış için repertuar ve kodlamaların aynı olduğu yerlerde) oldukça yaygındır. Ancak EMBEDDED PDV'de olduğu gibi, CHARACTER STRING türünü statik olarak kısıtlamak mümkündür, böylece karakterlerin yalnızca gerçek kodlamaları iletilir.

Birçok karakter repertuarı ve alt repertuarı ve birçok kodlama şeması için nesne tanımlayıcı değil erleri atanmışdır, ancak ne yazık ki hepsi için atanmamıştır. ASN.1'e CHARACTER STRING'den sonra UTF8String eklenmiştir. Kısıtlanmış bir KARAKTER Dİ Zİ Sİ olarak tanımlanabilirdi, ancak aslında, tipki PrintableString ve IA5String gibi, İngilizce metin kullanılarak tanımlanan yeni bir tür olarak ASN.1'e "bağlanılmıştır". Bu yüzden "baş arısal!" bu maddenin ikinci fikrasında yer almaktadır.

5.6 OCTET Dİ Zİ Sİ ve Bİ T Dİ Zİ Sİ

Elbette kara deliklerin en karası, gömülü malzemeyi taşımak için OCTET Dİ Zİ Sİ veya Bİ T Dİ Zİ Sİ kullanmaktadır. Olur. Gerçekten "kendi başına yuvarlanıyorsun". ASN.1 sınırlamayı (uzunluk sarmalayıcı) sağlayacaktır, ancak sekizli diziyi veya bit dizisi boşluğunu doldurduğuna ilişkin semantik bir alıcıya tanımlama sorunlarını çözmek gerekir.

Çok kısaltılmış bir ASN.1 kullanmaya inananlar, delikleri için bu türleri kullanırlar. Sanırım şikayet edemezsin. Çalıştırırlar. Ancak şunu adreste daha güçlü belirtim araçları mevcuttur:

ASN.1 cephaneliğ i ve umarım bu metni buraya kadar okuyan herhangi biri, bir delik açmaları gerekiğ inde OCTET STRING veya BIT STRING kullanmaya baş lamaz!

6 Bölüm II'yi sonuçlandırmak için açıklamalar

Acaba baş tan buraya kadar okuduğunu söyleyebilecek tek bir okuyucu (hatta benim eleş tirmenlerim bile!) var mı? Yaptıysanız bana j.larmouth@iti.salford.ac.uk adresinden e-posta gönderin. (Ama öylece atlayıp dizinden buraya geldiyseniz zahmet etmeyin!)

Ve ş imdi iyi geceler, hoş çakal, seni tanıdığ ima sevindim!

Bu metin, ASN.1 kavramlarının, mekanizmalarının, notasyonunun tamamını kapsamaya çalışmış tır. Tamamlandı ğına inanılıyor ("ASN.1 Tamamlandı" başlığı!). Kodlama kuralları, tarihçesi ve uygulamaları ile ilgili başka böümler de vardır, ancak notasyonun açıklaması artık tamamlanmış tır.

Peki ... 1999 itibariyle tamamlandı! Bu kitabı 2010'da okuyorsanız, edinmeniz gereken daha sonraki bir sürüm olabilir, çünkü bu metinde muhtemelen çok şeysiz eksik! Ancak size daha sonraki bir sürümme referans veremem - bir Web araması yapmayı deneyin ve özellikle Ek 5'te verilen URL'yi deneyin (bu, 2010'da çalışabilir veya çalışmayabilir!).

Yazma sırasında, ASN.1 standartlaşım grubunda ASN.1 notasyonuna eklemelere yol açabilecek pek çok öneri var. Bununla birlikte, yakın tarihli (1994 sonrası) tarih, yalnızca mevcut metni açıklığa kavuştururan veya çok küçük (teknik bir bakış açısından) ve basit yeni işlevler (UTF8String gibi) ekleyen değil iş iklikler getirdi, dünyayı sarsan eklemeler yapmadı. Gerçekten de, son on yılda önerilen ve muhtemelen dünyayı sarsan eklemelerin terk edilme geçmişini vardır - örnekler arasında hafif kodlama kuralları, küresel parametreler ve dinamik kısıtlamalar yer alır.

ASN.1 spesifikasiyonlarını okuma, yazma veya uygulamada iyi şanslar!

SON.

Peki ... bu bölümden!

BÖLÜM III

kodlamalar

Bölüm 1

Kodlama kurallarına giriş

(Veya: Kimsenin bilmesi gerekmeyenler!)

Özet: Bölüm 3'ün bu ilk bölümü:

Kodlama kuralları kavramını tartışır.

Temel Kodlama Kurallarının (BER) altında yatan TLV ilkesini açıklar.

"Geniş letilebilirlik" veya "geleceğin provası" sorununu tartışır.

Daha yeni Paketlenmiş Kodlama Kurallarının (PER) altında yatan ilkeleri açıklar.

"Kanonik" kodlama kurallarına olan ihtiyacı tartışır.

Diğer kodlama kurallarının varlığından kısaca bahseder.

Daha önceki böümlerde, bu kavrama yararlı bir giriş sağlayabilecek bazı kodlama kuralları tartışmaları zaten yapılmıştır, ancak bu bölüm eksiksiz olacak ve diğer böümlere atıfta bulunulmaksızın okunabilecek şekilde tasarılmıştır.

Bölüm III'ün sonraki iki bölüm, Temel Kodlama Kurallarını ve Paket Kodlama Kurallarını ayrıntılı olarak açıklar, ancak burada verilen ilke ve kavramların anlaşılığının varsayılar.

1 Kodlama kuralları nelerdir ve neden bölüm alt başlığı?

"Kimsenin bilmesine gerek olmayan şay!". Günün sonunda, bilgisayar iletişimimi tamamen "hattaki bitler" ile ilgilidir - geçmişte "somut aktarım sözdizimi" olarak adlandırılan, ancak bugün sadece "aktarım sözdizimi" olarak adlandırılan şey. (Ama düşündürseniz, bir "bit" veya "ikili basamak" oldukça soyut bir kavramdır - "somut" olan, bitleri temsil etmek için kullanılan elektrik veya optik sinyallerdir.)

Bitleri temsil etmek için kullanılan elektriksel veya optik sinyalleri bilmiyor veya umursamıyorsunuz, o halde soyut değil erlerinizi temsil etmek için kullanılan bit modellerini neden umursuyorsunuz?

ASN.1, Açık Sistemler Ara Bağlantısı (OSI) için ISO/ITU-T spesifikasiyonlarının içinde "Sunum Katmanı"ndan kaynaklanan bazı kavramları yerleşik olarak almıştır. ("Sunum Katmanı" teriminin kötü ve yaniltıcı olduğunu unutmayın - "Temsil Katmanı" daha iyi olabilir).

Kavramlar, bir iletişim im hattı üzerinden gönderilen ve bir iletişim im örneğinde bu soyut değil erleri temsil eden bit modelleriyle ilişkilendirilen bir "soyut değil erler" kümesine aittir.

Kullanılacak soyut değil erler kümesi ve bunlarla ilişkili semantik, herhangi bir uygulama belirtiminin merkezinde yer almır. "Kodlama kuralları", (Yeniden)Sunum Katmanı ile ilgilidir ve soyut değil erleri temsil etmek için kullanılan bit modellerini tanımlar. Kurallar, kendi başlarına eksiksiz bir tanımlamadır (aslında, iki ana kural setinin bir dizi çeşidi vardır - bunlar daha sonra açıklanacaktır). Kodlama kuralları, her bir temel ASN.1 tipindeki soyut değil erlerin ve ASN.1 gösterimi kullanılarak tanımlanabilecek herhangi bir olası yapılandırılmış tipteki soyut değil erlerin bir bit modeliyle nasıl temsil edileceğiini söyler.

ASN.1, kullanıcılarına, kullanıcı anımlarını taşıyan ve bir iletişim im hattı üzerinden iletilecek "soyut değil erleri" tanımlamaları için notasyon sağlar. (Bu, Bölüm I ve II'de tam olarak açıklanmışdır). Tıpkı bir kullanıcının sıfır ve bir biti temsil etmek için hangi elektriksel veya optik sinyalin kullanıldığıını umursamaması (ve sıklıkla bilmemesi) gibi, ASN.1'de de kullanıcının hangi bit modellerini umursamaması (veya ögrenme zahmetine girmemesi) gereklidir. soyut değil erlerini temsil etmek için kullanılır.

Dolayısıyla, ASN.1 değil erlerini temsil etmek için kullanılacak kesin bit modellerini tanımlayan ASN.1 "kodlama kuralları"nın ayrıntıları, korkunç derecede önemli olmakla birlikte, "Kimsenin bilmesi gerekmeyen şayelerdir".

Bugün, bir ASN.1 tür tanımını (örneğin), C, C++ veya Java programlamadaki bir tür tanımına eşleyecek iyi ASN.1 araçları ("ASN.1 derleyicileri" olarak adlandırılır) mevcuttur. diller (bkz. Bölüm I Bölüm 6) ve bu veri yapılarının değil erlerini ASN.1 Kodlama Kurallarına göre kodlamak için çalışma zamanı desteği sağlayacaktır. Benzer şekilde, gelen bir bit akışının kodu bu araçlar tarafından programlama dili veri yapısının değil erlerine çözülür. Bu, bu tür araçları kullanan uygulama programlarının kodlanmış bit modelleri hakkında hiç bir bilgiye sahip olmamaları ve hatta bunlarla ilgilenmemeleri gerektiği anlamına gelir. Endişelenmeleri gereken tek şayi, programlama dili veri yapılarının değil erleri için doğrudan uygulama anlamını sağlamaaktır. Okuyucu, bu kitabın Giriş kısmında ve Bölüm 1, Bölüm 1'de bu konularla ilgili daha fazla tartışma bulacaktır. ASN.1 derleyicilerinin ayrıntılı bir tartışması, Bölüm 1, Bölüm 6'da verilmektedir.

Ancak, ASN.1 Kodlama Kuralları hakkında her şayi bilmek isteyecek birkaç insan grubu vardır. Bunlar:

Entelektüel meraklı!

Öğrenciler üzerinde inceleniyor!

ASN.1 Kodlama Kurallarının kalitesi konusunda güvence almak isteyen standart yazarları.

Herhangi bir nedenle bir ASN.1 derleyicisi kullanamayan (belki belirsiz bir programlama dili veya donanım platformuyla çalışıyorlar veya araç satın alacak fonları yok) ve değil erleri "elle kodlamak" zorunda olan uygulayıcılar iletişim ve gelen bit modellerinin "el ile kodunun çözülmesi" için.

Bazı uygulamalar tarafından iletilen gerçek bit modellerinin ASN.1 Kodlama Kuralları spesifikasyonuna uygun olup olmadığıını belirlemesi gereken test ediciler ve sorun gidericiler.

Bu kategorilerden herhangi birine giriyorsanız, okumaya devam edin! Aksi halde kitabı bu bölüm size göre değil!

2 Kodlama kuralları yaklaşımıının avantajları nelerdir?

Bölüm 1 Bölüm 1, protokollerini belirlemeye yönelik bir dizi yaklaşımı tartıştı. ASN.1 yaklaşımı (OSI'nin Sunum Katmanından ötürü alınmışdır) degerleri temsil etmek için kullanılan bit modellerinin ayrıntılarını tamamen ayırma ve "gizleme" sonraki birkaç paragrafta tartışılacak bir avantajı vardır.

Unutulmaması gereken ilk nokta, soyut degerlerin iletilmesi kavramının bu degerlerini temsil eden bit modellerinden açık bir şekilde ayırmalarının, belirli ortamların ihtiyaçlarına uyacak şekilde çeşitli farklı kodlamaların kullanılmasını sağladığıdır. Sıklıkla alıntılanan bir örnek (ama bunu gerçek dünyada bulacağınızdan emin değilim!), her iki ucunda donanım ifreleme cihazları bulunan yüksek bant genişliği ine sahip kiralık bir hat üzerinden yapılan bir iletişim imdir. Buradaki ana endişe, iki ucunda en az CPU çevrim maliyeti uygulayan degerlerin temsillerine sahip olmaktadır. Ama kiralık hattan bir buldozer geçiyor! Yedekleme ise güvenlik cihazı olmayan bir telefon hattındaki modemdir. Endişe imdi maksimum sıkıştırma ve bazı seçici alan ifrelemesidir. Aynı soyut degerlerin iletilmesi gerekiyor, ancak bu degerlerin "en iyi" temsilinin ne olduğu hakkında degeşlik var.

Kodlama kuralları yaklaşımı, kodlamaları belirlemeye yönelik bir gizleme teknikidir. Kodlamalarda gelecekteki degeşlikleri yapma konusunda bilgi esnekliği sağlar.

İkinci örnek de buna benzer. Bir bilgisayar sisteminin diskinden başka bir bilgisayar sisteminin diske büyük miktarda bilginin aktarılması gereken bazı protokoller vardır. Bu sistemler farklısa, o zaman bir iletişim im hattı üzerinden degerlerin aktarımı için bilgilerin yerel temsillerini üzerinde anlaşmaya varılmış (standart) bir temsile eşlemek için sistemlerden biri veya her ikisi tarafından biraz çalışma yapılması gerekecektir. Ancak, bazı iletişim im örneklerinde, iki sistem aynı tip sistem ise, CPU döngüleri, muhtemelen bilginin ortak yerel temsili için kullanılabilecek bir temsil kullanılarak kaydedilebilir.

Yukarıdaki örneklerin her ikisi de, bir iletişim im örneğiinde, bir dizi olası temsilden kullanılacak gösterimi (kodlamayı) müzakere etmeye yönelik OSI kavramını doğrudan kullanır. Bununla birlikte, günümüzde ASN.1, kodlamanın önceden sabitlendiği ve iletişim im zamanında pazarlık konusu olmadığı (OSI Sunum Katmanı yoktur) OSI dış uygulamalarda daha yaygın olarak kullanılmaktadır.

Bununla birlikte, ASN.1 kullanıcıları için günümüzün gerçek dünyasında önemli olan soyut degerlerden kodlamaların bu net ayrimının birkaç avantajı daha vardır.

Son yirmi yılda, soyut degerler için "iyi" kodlamaların nasıl üretiliceğine dair insan bilgisinde önemli ilerlemeler gördük. Bu, 1980'lerin başında gelişen ASN.1 Temel Kodlama Kuralları ile 1990'ların başında gelişen Paket Kodlama Kuralları arasındaki farka yansır. Ancak 1980'lerde ASN.1 kullanılarak tanımlanan uygulama spesifikasyonları, yeni kodlama kurallarından yararlanmak için spesifikasyonda çok az degeşlik gerektirir veya hiç degeşlik gerektirmez - uygulama spesifikasyonu etkilenmez ve gelecekte daha da iyi kodlama kuralları tasarlanırsa etkilenmemeye devam edecektir. yüzüyl.

Aletlerle ilgili benzer ama belki de daha geniş kapsamlı bir konu var. Kodlama sorunlarının soyut degerler ve semantik uygulama belirtimlerinden ayırmaları, ASN.1 derleyicileri sağlama yeteneği açısından temeldir ve uygulama uygulayıcılarını programlama degerlerleri arasında eşleme yapmak için kod yazma (ve daha da önemli hata ayıklama) görevinden kurtarır. dil veri yapıları ve "hat üzerindeki bitler". Ayrıca, bu tür araçların kullanımında olduğunda yerlerde, PER gibi yeni bir kodlama kuralları kümese geçmek, yeni bir kodlama kuralının yüklenmesinden başka bir sey gerektirmez.

ASN.1 derleyicisinin sürümü ve belki eski yerine yeni kodlama kuralları için kodu çağırma zamanı çalışma zamanı çağırışında bir bayrağının değil iş tirilmesi.

3 Kodlamaları tanımlama - TLV yaklaşımı

Bölüm 1, Bölüm 1, değer erleri temsil etmek için karakter dizilerini kullanma yaklaşımı kısaca tartıştı; bu, kullanılacak dizileri tam olarak belirtmek için çeşitli mekanizmalara ve gelen karakter dizilerindeki kalıpları tanıtmak için "ayrış tırma" araçlarına yol açtı. Bu yaklaşımalar oldukça ayrıntılı protokoller üretme eğilimindedir ve genellikle ASN.1 ile mümkün olduğuk kadar eksiksiz bir araç desteği sağlayamaz. Bunlar daha fazla tartışılmıyor ve biz burada iletişiminde kullanılabilecek bit modellerini daha doğrudan belirleyen yaklaşımaları odaklıyoruz.

Temel öğeleri kodlamak için **Tür**, **Uzunluk**, **Değer** bir dizi TLV'yi bir dış **Tür** ve **Uzunluk** içinde bir **V** olarak yerleştirir, istediğiniz derinliği kadar tekrarlayın ve yapılandırılmış bilgileri çok sağlayabilir. **Değer** bir şekilde aktarmak için güçlü bir tarifiniz olur..

Uygulama belirtimlerinin karmaşıklığı, iki yıl içinde gelişmiş teknik temsilleri tanımlama görevine bir miktar "düzen" getirmek için önemli ve erken bir teknik, söyle "TLV" yaklaşımıydı.

Bu yaklaşım, bir mesajda gönderilecek bilgiler bir dizi "parametre değerleri" olarak kabul edildi. Her parametre değerleri, bir parametre tanımlamasıyla kodlandı (genellikle sabit uzunlukta, genellikle tek bir sekizli, ancak belki daha fazla sekizliye taşınan), ardından parametre değerinin uzunluğuunu (sekizli sayısı) veren bazı kodlamalar (yine tek bir sekizli olarak) ara sıra iki veya daha fazla sekizli uzunluk kodlamasına ihtiyaç duyulur) ve ardından bir sekizli dizisi olarak değerinin kendisi için bir kodlama.

Parametre kimliğiinin genellikle parametrenin türünü tanımladığı söylenir, bu nedenle bir Türümüz, bir Uzunlukımız ve bir Değerımız veya bir TLV kodlamamız vardır.

Bu yaklaşımalar, tüm alanlar tam bir sekizli sayısındır ve tüm uzunluk sayımları sekizlileri sayıyordu, ancak en eski yaklaşımalar bazları (ASN.1 tarafından takip edilmedi) temel birim olarak sekizli değer ile on altı bitlik sözcüklerine sahipti.

Kodlama türleri ve uzunlukları belirlendikten sonra, spesifikasyonun geri kalanının yalnızca her mesajda hangi parametrelerin görüneceğini, tam kimliklerinin ne olduğunu ve değerlerin nasıl kodlanacağıını belirlemesi gereklidir.

Bu yapının bir dizi önemli avantajı vardır:

Bir göndericiye parametreleri herhangi bir sırayla iletme özgürlüğü vermemi mümkün kılar, belki de daha basit (gönderen) bir uygulama sağlar. (Bugün bunun izin verilmesinin aslında kötü bir şeyle olarak görüldüğü üne dikkat edin, iyi bir şeyle değil!!)

Bazı parametrelerin isteğiyle bağlı olduğunu - yalnızca bir mesaj gerektiğiinde dahil edilmek üzere - bildirmeyi mümkün kılar.

Değer işken uzunluktaki öğeleri işler.

Gerçek parametrelerin kendileri hakkında herhangi bir bilgiye ihtiyaç duymadan, bir dizi parametre dē̄ erine temel bir "ayrı̄̄ tırma" sağ lar.

Ve daha da önelimi - bir sürüm 1 sisteminin protokolün 2. sürümüne eklenen parametreleri tanımlamasını, sonunu bulmasını ve yok saymasını (istenen davranış buysa) veya belki de ileriye doğ ru iletmesini sağ lar.

Okuyucu, bu özelliklerin ASN.1 ile olan iliş kisini - "SET"in (herhangi bir sırayla iletilen ȫ̄ eler), bir SET veya SEQUENCE'in ȫ̄ elerine uygulanabilen "İ̄̄ STĒ̄E BĀ̄L" notasyonun ve dē̄ iş ken uzunluk doğ asının varlığı ini tanımlıdır. birçok ASN.1 temel türünden. Sürüm 1/sürüm 2 sorunu, ASN.1'de genellikle "geniş letilebilirlik" olarak adlandırılan sorundur.

1970'lerin sonunda geliş tırilen bu "parametre" kavramının ȫ̄esindeki en büyük uzanti, ilgili parametreleri birbirine yakın tutmak iç in kullanılan "parametre grupları" fikridir. Burada bir "grup tanımlayıcısı", bir grup uzunlū̄ u kodlaması ve ardından grup iç indeki parametreler için bir dizi TLV kodlaması kodluyoruz. Daha önce olduğ u gibi, gruplar herhangi bir sırada görünebilir ve tam bir grup istē̄ e bā̄lı veya zorunlu olabilir; bu grup iç indeki parametreler herhangi bir sırayla ve o grup iç in istē̄ e bā̄lı veya zorunlu olabilir. Böylece etkili bir ş ekilde iki seviyeli TLV'ye sahibiz - grup seviyesi ve parametre seviyesi.

En iç teki TLV'ler dış inda tümünün V kısmı bir dizi gömülü TLV olmak üzere keyfi olarak birçok TLV düzeyine izin veren doğ al bir uzantıdır. Bu, yeni bir tür temel türlerin Dİ Zİ Sİ veya KÜMESI olarak tanımlayabilme, ardından bu yeni türü daha sonraki Dİ Zİ LERDE veya KÜMELERDE temel bir türmüz gibi kullanabilme ve bunun gibi ASN.1 konseptiyle açıkça eş leş ir. herhangi bir derinlik

Böylece bu iç iç e TLV yaklaş imi, ASN.1 Temel Kodlama Kuralları iç in alınması gereken doğ al yaklaş im olarak ortaya çıktı ve on yıldan fazla bir süre üstünlük ünù sürdürdü.

Temel Kodlama Kurallarını tamamen anlamak iç in ş unlara ihtiyacımız var:

"T" bđümünün kodlamasını ve "T" bđümündeki tanımlayıcının nasıl tahsis edildī̄ ini anlamak.

Hem kısa "V" bđümleri hem de uzun "V" bđümleri iç in "L" bđümünün kodlamasını anlamak.

INTEGER, BOOLEAN, BIT STRING gibi her bir temel tür iç in, "V"nin o türün soyut dē̄ erlerini temsil edecek ş ekilde nasıl kodlandığ i.

SEQUENCE veya SET gibi her bir oluş turma mekanizması iç in, bu mekanizma ile tanımlanan türlerin kodlamalarının iç iç e TLV yapılarıyla nasıl eş leş tī̄ i.

Bir sonraki bđümün gündemi bu.

4 Geniş letilebilirlik veya "gelecek provası"

TLV yaklaş imi, sürüm 1 sisteminin spesifikasyonunun, "T" parçasının tanınmadığ i TLV ȫ̄ eleri üzerinde belirtilen eylemi gerektirmesini sağ lamada çok güç lüdür. Bu, bu tür materyali alan sürüm 1 sistemlerinden bilinen bir davranış modeliyle, bir spesifikasyonun 2. sürümüne yeni ȫ̄ elerin (belirgin bir "T" parçası ile) eklenmesine izin verir.

Hem sürüm 1 hem de sürüm 2 protokolünü uygulamak için sürüm 2 uygulamalarına ihtiyaç duymadan sürüm 1 ve sürüm 2 sistemleri arasındaki bu birlikte çalışma, ASN.1'in güçlü ve önemli bir özelliği idir.

Bu, Temel Kodlama Kurallarında kodlamaya yönelik TLV yaklaşımının doğallık sonucudur, ancak satır boyunca minimum bilgi aktarımının olduğumu kodlamalar aranırsa, bir dereceye kadar "geleceğe yönelik" nasıl elde edileceğini araştırmak önemlidir. TLV yaklaşımının ayrıntıları olmadan sürüm 1 ve sürüm 2 sistemlerinin birlikte çalışmaya izin vermek için.

Bu alandaki ilk tartışmalar, geleceğe yönelik korumanın yalnızca bir TLV stilini kodlama kullanıldığıında mümkün olduğunu gösteriyor gibi görünüyor, ancak daha sonraki çalışmalar, protokolde sürüm 2 eklemelerine ihtiyaç duyulabilecek yerlerin yeni bir notasyon yapısı tarafından tanımladığını gösterdi (ASN.1 "geniş letilebilirlik" üç nokta - üç nokta), daha sonra hiçbir şekilde TLV tipi bir yapı olmayan bir kodlama yapısında bile çok az ek yük ile geleceğe yönelik koruma mümkün hale gelir.

Sözde Paket Kodlama Kurallarının (PER) geliştiirilmesini sağlayan bu tanınmayı.

5 PER'de ilk denemeler - BER ile başlayın ve gereksiz sekizlileri kaldırın

Bu bir çıkmaz sokaktı!

NOT — BER hakkında bilgisi olmayanlar, aşağıdaki metne dörmeden önce en azından bir sonraki bölümü gözden geçirmek isteyebilirler, çünkü bazı örnekler BER kodlamalarını göstermektedir.

ASN.1 için daha kompakt (paketlenmiş) kodlamalar üretmeye yönelik ilk yaklaşım, BER TLV tarzı bir kodlamaya dayanıyordu, ancak bir BER kodlamasında, izin verilen tek olası sekizli değil erin bu olduğumu yerde, genellikle satırda aşağıdaki gönderilen sekizlilerin olduğumunu kabul edilmesiyle bu konumda (en azından spesifikasiyonun bu versiyonunda). Bu, özellikle "T" değil erleri için geçerlidir, ancak örneğin deger eri kismi (bir BOOLEAN değil eri gibi) sabit uzunluktaysa, sıklıkla uzunluk alanına da uygulanır.

İlk başta baş aramazsan, dene, tekrar dene!

Paketlenmiş Kodlama Kurallarının kısıtlamaları hesaba katmasına izin vererek (örneğin, dizelerin uzunluğu veya INTEGER'lerin boyutları üzerinde), her iki üç dağ eri bildiğiinden, uzunluk alanlarının açık iletiminin gerekli olmadığı daha birçok durum bulabiliriz. "L" alanının

Son bir "iyileş tirme", DIZİ türü için "L" alanını dikkate almaktır. Burada SEQUENCE'in her elemanı bir TLV olarak kodlanmış tir ve SEQUENCE'in tamamı için bir dış seviye "TL" "sarmalayıcı" vardır. BER'i değil iş tirirsek, bu sarmalayıcının "L" kismı sekizli sayımı değil, SEKANS'ın değil eri kısmındaki TLV'lerin sayısı olacak şekilde değil iş tirirsek, bu sayı tekrar sabitlenir (SEKANS İSTECE BAĞLI değil içermiyorsa) ve bu nedenle, uzunlukları değil iş ebeilen iç değil erler olsa bile genellikle iletilmesi gerekmek.

Ş ekil III-1'de gösterilen ASN.1 tipini göz önünde bulundurun. BER kodlaması (iç uzunluk olmayan alanlar için sekizli yerine TLV'leri sayacak ş ekilde değil iş tirilmiş tir) ş ekil III-2'de gösterilmiş tir.

```
Örnek ::= Dİ Zİ {ilk TAM SAYI (0..127),
    ikinci Dİ Zİ {dize ODETZGİ Sİ (SIZE(2)), PrintableString
        (SIZE(1..8))}, üçüncü BIT Dİ ZGİ Sİ (SIZE (8))}
```

isim

Ş ekil III-1: Kodlama için bir örnek

Ş ekil III-2'de hat boyunca gönderilen toplam 23 sekizli olduğunu göreceksiniz, ancak bir alıcı 11'i hariç hepsinin değil erini önceden tahmin edebilir - {????} olarak işaretlenenler (ve kesin olarak bilir) bu 11'in meydana geldiğinden yer). Böylece kalan 12 sekizliyi iletmemize gerek kalmaz, bu da iletişim imtaçında %50'lük bir azalma sağlar. Çekici!

O halde yaklaşım, başlangıç noktası olarak bir BER kodlaması almak, ne için kurallar belirlemekti?

```
{Ü 16} { 3}          -- Üniversal sınıf 16 (SEQUENCE için "T" değil eri) -- 3 öğe e (SEQUENCE için "L" değil eri)

{U 2} { 1}          -- Evrensel sınıf 2 ("ilk" için "T" değil eri) -- 1 sekizli ("ilk" için "L" değil eri)
{????}
-- "ilk" değil eri

{Ü 16} { 2}          -- Evrensel sınıf 16 ("saniye" için "T" değil eri) -- 2 öğe e ("saniye" için "L" değil eri)

{Ü 3} { 2}          -- Evrensel sınıf 3 ("dize" için "T" değil eri) -- 2 sekizli ("dize" için "L" değil eri)
{????}{????}
-- "string" değil eri
{U 24} {????}        -- Evrensel sınıf 24 ("ad" için "T" değil eri) -- 1 ila 8 ("ad" için "L" değil eri - 5 say)

{????}{????}{????}{????} -- "name" değil eri
{U 4}              -- Evrensel sınıf 4 "Üçüncü" için "T" değil eri { 3 } -- 3 sekizli ("üçüncü" için "L" değil eri) { 0 } --
"Üçüncü" "V"nin son sekizlisinde 0 kullanılmayan bit { ????}{????}
-- "Üçüncü" değil eri
```

Ş ekil III-2: Ş ekil III-1'in değil iş tirilmiş BER kodlaması

sekizlilerin iletilmesine gerek yoktur ve bu sekizlileri iletimden önce BER kodlamasından silmek için, standart bir BER kodu çize ve gerçekeleş tirmeden önce bunları (tip tanımı bilgisinden) alıma yeniden ekleyin.

Bu yaklaşım üzerinde yaklaşım, üç yıllık bir süre boyunca çalışılar yapıldı, ancak başı arızsız oldu. Bir BER kodlamasından neyin silinip neyin silinemeyeceği konusunda dair ek (oldukça geçici) kurallar eklendiği ve uluslararası oylamaya gidildiği giderek daha karmaşık hale gelen bir belge üretildi. New York'un hemen dışında (1990 civarında) bir düzenleme toplantısı düzenlendi ve Ulusal Organlardan gelen yorumlar yalnızca toplantının başındaki katılımcılara fakslandı.

Bir düzine kadar katılımcının HER Ulusal Organının "HAYIR" oyu verdiğiini ve dahası Hİ ÇBİ R yapıçı yorumda bulunmadığını fark ettiklerindeki şaşkınlığı hayal edin! Yaklaşım, "SEQUENCE OF BOOLEAN" gibi şeylerin verimli kodlamalarını üretmek için çok karmaşık, çok ad hoc ve (çünkü yine de tam sayı sekizli gerektiren her şeyi bıraktığı için) yetersiz görülmüyordu. Suda oldukça net bir şekilde düzüdü.

Pek çok kişi önceden rezervasyon yapmıştı, bu uçuşlar büyük masraflar çıkarılmadan değil iş tirilemeyecekti, ancak bir haftalık toplantı olarak planlanan toplantıların sona erdiğinde açıldı. Toplantı öğle yemeğinden sonra saat 11:00 civarında erkenden bozuldu (ve sonunda saat 16:00 gibi geç saatlerde yeniden toplandı). Öğle tatilinde çok fazla bira tüketildi ve meşhur bir sigara paketinin arkası tartışmaları kaydetti (aslında, sanırım bir kağıt peçeteyle - çoktan kaybolmuştu!). Bugün bildiğimiz ekliyle PER doğdu!

Haftanın geri kalanı kemiklere biraz et koydu ve sonraki iki yıl, sonunda PER spesifikasyonu olarak kabul edilen nihai metni üretti. Bunu destekleyen araçların uygulamaları bir yıl kadar sonra geldi.

6 PER ilkelerinden bazıları

6.1 BER deli gömleğinin kırılması

PER'in ilk öğle yemeğinden zamanı tasarımdaki muhtemelen en önemli kararlar şunlardır:

Temiz bir kağıt parçasıyla (veya daha doğrusu peçeteyle!) başlamak ve BER'i ve herhangi bir TLV kavramını göz ardı etmek. Bu, zamanlar oldukça radikal ve bira muhtemelen insanların düşüncelerini düşünebilen biri düşünebilmesine yardımcı oluyordu.

I İlk "ilkeler"
TLV'yi unutun.
 Sekizlileri unutun - bitleri kullanın.
Kısıtlamaları (alt türleri) tanır.
"Akıllı" kodlamalar üretin.
 "Geniş letilebilirlik" unutun (başlangıçta).

Tam bir sekizli sayısını kullanmakla sınırlanılmamak - başlangıçta bir oldukça radikal fikir.

Mantıklı bir şekilde yapılabileceği gibi, tip tanımındaki kısıtlamaları (alt tiplendirme) tam olarak hesaba katmak. (BER, kısıtlamaları göz ardı etti, belki de kısıtlama/alt tip notasyonu ASN.1'e eklenmeden önce üretilmiş ve bu notasyon 1986 civarında geldiğinde değil iş tirilmediğinden).

Zeki bir insanın (şimdiden kadar biraz sarhoş!) üretebileceği türden bir kodlama üretmek - bu oldukça zorlu bir işti!

"Geniş letilebilirlik" konularını dikkate almamak. Bu, (uzun) bir öğle yemeğinde tartışmada her şeye mümkün kılan pragmatik bir karardır, ancak elbette "geleceğin kanıtlama" hükmünün daha sonra eklenmesi gerekiyordu (ve eklendi).

Peki okuyucu bir şeyle nasıl kodlarsınız? Açıktır, en düşündüğünüz şeyle muhtemelen PER'in yaptığı işdir! Aşağıdaki tüm durumlarda, "bariz" çözüm PER'in yaptığı işidir.

BOOLEAN'in kodlaması ne durumda? Açıktır, sıfır veya bir ayarlanan tek bir bit "bariz" çözümüdür.

Ne dersin

ve **TAM SAYI (0..7)**
TAM SAYI (8..11)

Açıktır, birincisi için üç bitlik bir kodlama ve ikincisi için iki bitlik bir kodlama uygundur.

16 bitlik bir aralıkla sınırlı bir INTEGER değer eri, uzunluk alanı olmayan iki sekizliye gidebilir.

Peki ya kısıtlanmamış bir INTEGER? (Teorik olarak, sonsuza kadar tamsayı değer erleri ve iletilmesi milyonlarca yıl süren tamsayı değer erleri kodlayabilen BER ile (süper hızlı hatlar üzerinden bile) anlamına mı geliyor? Burada açıkça bir "L" uzunluğu kodlamak için gereklili olacaktır tamsayı değer eri (ve burada muhtemelen sekizli cinsinden bir uzunluk sayımı yapmak isteriniz).

"L"nin BER kodlamalarının ayrıntılarını okuduysanız, 127 sekizliye kadar uzunluk sayımları için "L"nin tek bir sekizlide kodlandığıını ancak BER'in "L" için üç sekizli gerektirdiğini biliyoruz. 255'ten fazladır. PER'de sayı, bitlerin, $\frac{1}{2}$ elerin veya sekizlerin sayısıdır, ancak 64K veya daha fazla sayımlar için yalnızca iki sekizlinin ötesine geçer - karşılaştırıldığında bir çok durumda "L" boyutunda yüzde elli azalma BER ile.

Kısıtlanmamış bir INTEGER'in neredeyse tüm değer erleri için, bir sekizli "L" alanı ve ardından gönderilen gerçek değer eri tutmak için gereken minimum sekizli sayısını alacağınız. Bu, BER ile aynıdır.

6.2 Bir "T"nin çözümü diğer er sorunlarla nasıl başa çıkarılır?

Şimdiden PER için bir "T" alanından söz edilmedi. Hiç birine ihtiyacımız var mı? BER'de "T" alanının oldukça önemli olduğunu üç ana alan vardır. Bunlar:

Bir "seçim dizini" kullanın.

Sabit bir sırada AYARLAYIN.

$\frac{1}{2}$ STEĞE BAĞLI $\frac{1}{2}$ eler için bit haritası.

Hangi gerçek alternatifin bir SEÇİ M tipinin değer eri olarak kodlandığıını belirlemek için (bir SEÇİ M'in tüm alternatiflerinin farklı etiketlere ve dolayısıyla farklı "T" değer erlerine sahip olması gerekiyor) unutmayın).

SEQUENCE (veya SET) içindeki $\frac{1}{2}$ STEĞE BAĞLI $\frac{1}{2}$ elerin varlığıını veya yokluğununu belirlemek için.

Bir SET'in hangi $\frac{1}{2}$ esinin nerede kodlandığıını belirlemek için (bir SET'in $\frac{1}{2}$ elerinin gönderici tarafından seçilen herhangi bir sırada kodlanabileceğini ve gönderilebileceğini unutmayın).

Her $\frac{1}{2}$ için "T" kodlaması olmadan bu şayeler nasıl yapılır?

Bir SEÇİ M'deki alternatiflerle başa çıkmak için PER, gerekli minimum bitlerde bir "seçim indeksi" kodları: en fazla iki alternatif, bir bit; üç veya dört alternatif, iki bit; beş ile yedi alternatif, üç bit; vb.

Bu noktada PER tasarımda önemli bir disiplin gözlemlenebilir. Kodlamanın herhangi bir özel parçası için alan genişliği (bit olarak) (bu durumda seçim dizininin alan genişliği) iletilen soyut değerere bağlı değildir (bağlı değil (bağlı olmamalıdır)), ancak statik olarak incelenerek belirlenebilir. tip tanımı.

Bu nedenle, aynı tip tanımını kullandıkları varsayılarak, iletişim imin her iki ucu tarafından açık bir şekilde bilinir. Ama sürtünme var! Biri sürüm 1 tip tanımı ve diğer eri sürüm 2 tip tanımı kullanırsa ancak bunu henüz dikkate almamaya karar verdik!

Önemli alan uzunluğu ilkesi veya kuralı: $\frac{1}{2}$ steğe bağlı sayıda bit alanlarına kodlayın, ancak alanların uzunluğu, tüm değer erler için tür tanımından statik olarak belirlenebilir olmalıdır.

Bir KÜME veya DİZİ içindeki $\frac{1}{2}$ STEĞE BAĞLI $\frac{1}{2}$ eler ne olacak? Yine, fikir oldukça açık.

Bir $\frac{1}{2}$ STEĞE BAĞLI $\frac{1}{2}$ enin, değer erinde var olup olmadığıını belirlemek için bir bit kullanırız.

AYARLA veya SIRASI. Aslında, bu bitlerin tümü birlikte toplanır ve aşağ ida tartışılıan "hizalama" ile ilgili nedenlerden dolayı isteği e bağılılığı e konumundan ziyade SET veya SEQUENCE kodlamasının başlangıcında kodlanır.

Ve böylece "T" gerektirebilecek üçüncü öğeeye. Peki ya SET kodlaması - burada kesinlikle "T" kodlamalarına ihtiyacımız var? SET'in (öğelerin gönderici tarafından belirlenen bir sırada iletildiği yer) SEQUENCE (kodlama sırasının tür tanımındaki öğelerin sırası olduğu u) üzerindeki önemi ve SET'in neden olduğu sorunlar hakkında büyük tartışmanın başlaması. Bir tür "T" kodlaması sunmanın ayrıntılarına ek olarak, şunu da gözlemlenmeliyiz:

Gönderenin seçeneklerine izin vermek, (alma) uygulamalarının her durumda doğrulu davrandığıını kontrol etmek için herhangi bir kapsamlı test dizisi biçiminde (ve dolayısıyla uygunluk denetimi maliyetinde) birleş tirici bir patlama oluşturur.

Aynı bilgiyi göndermenin birden fazla yolunun varlığı, güvenlik dünyasında "yan kanal" olarak adlandırılan şeysi üretir - gönderenin seçeneklerini sistematik olarak değil iş tirerek bir truva atından ek bilgi iletme aracı. Örneğin, bir SET'te sekiz eleman varsa, elemanların sırasını sistematik olarak değil iş tirerek bu SET'in her degele 256 bit ek bilgi iletilebilir.

Bu tartışma, PER içinden başka bir ilkenin gelişmesine yol açtı: Kodlamada, bunları tanıtım için mükemmel bir neden olmadığına HİÇBİR R gönderici seçenekini olmayacaktır. PER etkili bir şekilde göndericinin seçeneklerine sahip değilidir. Bir SET'in öğelerini iletmek için kanonik bir düzen gereklidir ve uzun tartışmalar sonra, bu, metinsel olarak yazdırılan sıra yerine öğelerin etiket sırası (daha fazla ayrıntı için bir sonraki bölümde bakın) olarak alındı. (Seçim indeksi değil erlerini bir seçimin alternatiflerine tahsis ederken, tutarlılık için metinsel sıra yerine aynı etiket sırası da kullanılır).

Gönderenin seçenekleri ilke/kural: Hiç sahip olma!

Bununla birlikte, "PER" teriminin kesinlikle birbiriyle yakından ilişkili dört kodlama kuralından oluşan bir aileye atıfta bulunduğu una dikkat edilmelidir. En önemlisi "BASIC-PER" (daha sonra tartışılıacak bir HİÇ ZALALI ve HİÇ ZALANMAMİŞ bir varyantla birlikte). BASIC-PER gönderen seçenekine sahip olmasa da, bir SET OF öğesinin öğelerinin degerlerinin sabit bir düzende sıralanması gerekmeli inden ve kaçış dizilerinin yollarına herhangi bir kısıtlama getirilmediği inden, gerçek bir kanonik kodlama kuralı olarak kabul edilmez. GeneralString'in kodlamalarında kullanılır. (Bir uygulama belirtiminde bu iki türden hiçbiri kullanılmıyorsa, BASIC-PER neredeyse kanoniktir (pratikte hiçbir zaman ortaya çıkmayan ve tamamen kanonik olmadığı bazı diğer önemsiz karmaşık durumlar vardır. Ayrı bir CANONICAL-PER vardır (ayrıca bu türler mevcut olduğu unda bile gerçekten kanonik olan bir HİÇ ZALALI ve HİÇ ZALASIZ bir sürümle)).

6.3 SEQUENCE ve SET başlangıkları için hala T ve L'ye ihtiyacımız var mı?

Açıktır yapmıyoruz! İsteğe bağılılığı e erin varlığıını veya yokluğununu tanımlayabilmemiz koşuluyla (bu, daha önce açıklanan bit haritası tarafından yapılmıştır) bu türler için başlangık kodlamalarına ihtiyacımız yoktur.

"Sarmalayıcılara" artık gereklidir. Pekala ... aşağidakı bu bir nevi doğrulu - ama tartışmaya bakın geniş letilebilirlik, sürüm 2'de eklenen öğeler için sarmalayıcıları yeniden tanır!

6.4 Hizalanmış ve Hizalanmamış PER

Ancak burada PER'in baş ka bir özellig ine bakiyoruz.

Temel olarak PER, belirli sayıda bit uzunluğ unda ve iletim için basit bir ş eklide uçtan uca birleş tirilen alanlara kodlamalar üretir.

Ancak, bazı ASN.1 türleri için (örneğ in, iki baytlik tamsayılar dizisi), her bileş en dē erini öneğ in 6. bitten baş latmanın aptalca olduğu u baş indan beri kabul edildi. Baş langıçta iki dolgu bitinin eklenmesi dē er dizisinin bir kısmı, muhtemelen CPU maliyetleri ve hat maliyetleri arasında iyi bir uzlaş ma olacaktır.

Bu, ȫ̄ elerin bit alanlarına (kodlamadan önceki b dümlerinde bitlerin sonuna basitçe eklenen) veya sekizli hizalanmış bit alanlarına kodlama kavramına yol açtı ; hizalanmış bit alanları sekizli bir sınırla baş ladi.

CPU dängü verimliliğ i için dolgu bitleri ekliyorsunuz, ancak daha sonra gerçekten düşük bant geniş lig ine sahip hatlar için bunları çıkarmanız isteniyor! Sonuç: BASIC-PER ve CANONICAL-PER'in ALIGNED ve UNALIGNED varyantları.

Akılı okuyucu (hepiniz dē il mi?), alanların uzunluğ u türden statik olarak belirlenirken (olması gereklidir), bir sekizli hizalanmış bit alanından önce eklenen dolgu bitlerinin sayısının sabit olmadığı ini not edecek. Kodlamadan önceki kısmındaki bit sayısı, SET ve SEQUENCE'in isteğ e bağılı ȫ̄ elerin mevcut olup olmamasına ve bir SEÇ M'de seçilen gerçek alternatif bağılı olabilir. Ancak elbette, kodlama her zaman bununla ilgili bilgi içерir ve bu nedenle bir alıcı uygulama, mevcut olan ve göz ardı edilmesi gereken doldurma bitlerinin sayısını her zaman belirleyebilir. Bir alanın bir bit alanı mı yoksa bir sekizli hizalanmış bit alanı mı olduğunu yine tip tanımından statik olarak belirlenmesi gerektī̄ ine (ve belirlendī̄ ine) dikkat edin - iletilen gerçek dē ere bağılı olmamalıdır, aksi takdirde PER bozulur!

"Sekizli hizalanmış bit alanları" ve "dolgu bitleri" kavramı orijinal tasarımdaydı, ancak daha sonra hava trafik kontrolündeki insanlar dolgu bitlerinin kaldırılmasını istedi ve ş imdi PER'nin iki çeşidine sahibiz. Her ikisi de, tür tanımına bağılı olarak bir "bit alanları" ve "sekizli hizalanmış bit alanları" dizisine resmi olarak kodlar, ancak "hizalanmamış PER" için, iki arasında hiçbir fark yoktur - doldurma bitleri asla eklenmez. "sekizli hizalanmış bit alanlarının" başlangıcı. Hizalanmış PER ile, onlar

vardır.

Aslında hizalanmış ve hizalanmamış PER arasında birkaç başka fark vardır, ancak bunlar ayrıntılar için sonraki PER b dümene bırakılmıştır.

Son bir yorum olarak - dolgu bitlerinin eklenmesinden sonra sekizli hizalamasını mümkün olduğ u kadar uzun süre tutmaya çalışmak istiyorsanız, bir SEQUENCE veya SET'de İSTEĞE BAĞLI bir ȫ̄ enin varlığı ini veya yokluğu unu belirtmek için tek bir bit kullanmak muhtemelen iyi dē ildir. fikir - SEQUENCE veya SET kodlamasının başlangıcında bu tür tüm bitleri bir "bit-map" olarak toplamak daha iyidir. Bu, orijinal sigara paketi tasarımlının bir parçasıdır ve daha önce kısaca bahsedilmiş ti. Bu özellik PER'de mevcuttur.

7 Geniş letilebilirlik - sahip olmalısınız!

Üçüncü girişim!

Bir bit her şeysi söylüyor - bu bir sürüm 1 dē eri veya sarılmış sürüm 2 materyali içeriyor.

İlk başta baş aramazsan, dene, tekrar dene!

Daha iyi kodlamalara yönelik ikinci yaklaşım (yukarıda açıklanmışdır) uluslararası düzeyde oylandığıında, neredeyse yine baş arısız oldu.

Yukarıdaki tartışmadan, her iki ucun da uygulamaları için tam olarak aynı tür tanımına sahip olmadıkça, kiyametin kopacağı - açıktır - terimi baş işlayın. Alanlar ve mevcut alan uzunlukları hakkında farklı görüşlere sahip olacaklar ve kodlamalardan neredeyse rastgele soyut değil erler üretecekler.

Ama gerçekten havlu atıp çok ayrıntılı bir TLV tarzı kodlamanın "geleceğe dayanıklı" olmak istiyorsak mümkün olan tek şeyle olduğumu kabul etmek istiyor muyuz? NUMARA!

Sürüm 2'nin bir şeyle eklemesine nasıl izin verilir? "Kök" (sürüm 1) belirtiminin sonunu ve eklenen sürüm 2 (veya 3 vb.) materyalinin başlangıcını belirtmek için gösterime ne dersiniz? Bu yardımcı olacak mı?

"Geniş letilebilirlik" gerektiren en yaygın durum, sürüm 2'deki SET'lerin ve SEQUENCE'ların sonuna aşağıdaki ekleyebilme yeteneğidir.

Daha sonra insanlar - başlı arılı bir şeyle - SET'lerin ve DİZİ'lerin ortasına aşağıdaki eler eklenmesi gerekiyor gibi tartışmalar ve daha önceki bir bölümde açıklanan "ekleme noktası" kavramını elde ettik.

Ama şeyle imdilik en sona eklemeye devam edelim. Bir SEQUENCE'in sonuna (çoğu muhtemelen İSTEĞE BAĞLI olacak) aşağıdaki eler eklediğimizi veya bir SEÇİM'e alternatifler eklediğimizi veya bir SAYILANDIRILMIŞ'a numaralandırmalar eklediğimizi veya bir TAM SAYI'da kısıtlamaları gevşettiğimizi (bu liste şeyle imdilik yeterli olacaktır) varsayıyalım. !).

Bununla nasıl başlıyor? "Gelecekte prova" istiyorsak, önce bir türün "geniş letilebilir" olarak işaretlenmesini şeyle koşturuyoruz (bu, birçok ASN.1 türünde görünebilen üç noktadır). Bu, sürüm 1 uygulamasını, sürüm 1 türünün ötesine geçen soyut değil erlerle karşılaştırmayı mümkün kılar, ancak daha da önemlisi, bu türdeki tüm soyut değil erlerin sürüm 1 kodlamalarının başına bir "geniş letilmiş" bit ekler.

Konsept, bu "geniş letilebilir" türlerden herhangi birinin bir "kök" soyut değil erler kümesine sahip olmasıdır - sürüm 1 soyut değil erler. Gönderilen soyut değil er (sürüm 1, sürüm 2 veya sürüm 3, vb. uygulaması tarafından) kök içindeyse, "geniş letilmiş" bit sıfıra ayarlanır ve kodlama yalnızca sürüm 1 tipinin kodlamasıdır. Ancak 1 olarak ayarlanırsa, sürüm 2 veya sonraki sürümlerde sunulan soyut değil erler mevcuttur ve sürüm 1 sistemlerinde bir dizi seçenek vardır, ancak daha da önemlisi, ekstra uzunluk (ve bazen tanımlama) alanları "sarma" böümlerine dahil edilir veya sürüm 1 sistemleriyle iyi bir birlikte çalışmayı sağlamak için bu yeni soyut değil erlerin tümü. "İşte işaretçisi", belirticilerin, erken sürüm sistemlerinin sonraki sürümlerde eklenen materyalle nasıl başlıyor ve (bu yazarın görüşlerine göre) geniş letilebilirlik işaretçisinin tanıtılması durumunda her zaman dahil edilmesi gerekiyorını söylemesini sağlıyor.

"Geniş letilebilir" türler için tam kodlama biçimi, aşağıdaki PER bölümünde daha ayrıntılı olarak ele alınmışdır. Bu bölümde daha sonra.

8 PER hakkında bilmeniz gereken başlıyor ne var?

PER artık BER'e herhangi bir atıfta bulunulmadan tanımlanırken (nesne tanımlayıcıları ve genelleş tirilmiş zaman ve gerçek türler gibi şeylelerin değil er kısmının kodlanması dışındadır), Şekil III-1'de gösterilen türden bir değil erin PER kodlamasının not edilmesi ilginçtir. asıl olarak daha önceki (terk edilmiş) yaklaşımında üretilen tam olarak aynı 11 sekizli (Şekil III-2'de gösterilmiş) üretir!

Bu bölüm PER kavramlarının çoğuunu tanıttı, ancak PER hakkında çok renilecek BER'den çok daha fazla şey var. Bunların hepsi bir sonraki birinci bölümde ele alınmıştır.

Bilmeniz gereklidir (bir ASN.1 derleyici aracı yazmıyorsanız muhtemelen bilmezsiniz! Bu bölümün ilk kısmına bakın!):

Hangi kısıtlamalar (alt tiplendirme), çeşitli türlerin PER kodlamasını etkiler (bunlara "PER-visible kısıtlamaları" denir).

Kodlamanın ("bit alanları" ve "sekizli hizalanmış bit alanları" genel yapısı nedir ve "tam kodlama" nasıl üretilir?

Uzunluk alanları ne zaman dahil edilir ve "uzunlukların uzunlukları" ne zaman gereklidir ve bunlar nasıl kodlanır.

PER, SEQUENCE'ları, SET'leri ve SEÇİMLERİ nasıl kodlar? (Yukarıdaki metinden zaten iyi bir fikriniz var).

PER, diğer tüm ASN.1 türlerini nasıl kodlar? (Aslında, çoğu u zaman kodlayan BER "V" kısmına atıfta bulunur.)

"Geniş letilebilirlik işaretçisinin" varlığı PER kodlamalarını nasıl etkiler? (Yine yukarıda, etkinin bir taslağı verilmiş tir - soyut değil - kkeletal bir bitlik bir ek yük ve değil ilse genellikle ek bir uzunluk alanı.

Bunların hepsi yukarıda değil inilen, ancak daha sonra daha ayrıntılı olarak ele alınan konulardır.

9 PER ile Deneyim

Artık mevcut protokol belirtimlerine uygulanan PER ile ilgili çok fazla deneyim var ve belirticiler arasında PER-dostu belirtimler (yani, kısıtlamaların tutarlı bir şekilde ile tamsayı alanlarına ve uygun olduğu unda dize uzunlukları ~~üzerindeki genel kodlama~~ istek

Bant genişliği indirimleri (eklenmiş olsa bile genel amaçlı sıkıştırma - sürpriz?).
CPU döngüsü azaltmaları (gerçek sürpriz).
Karmaşık ilişkili - yalnızca analiz zamanında!
Aletlerin kullanımıyla ilişkili - aletlerin avantajlarını artırır.

PER uygulamaları kullanıma sunulmaya başladığında bazı sürprizler yaşandı.

Her şeyden önce, genel amaçlı sıkıştırma algoritmalarını mevcut protokollerin hem BER hem de PER kodlamalarına uygulamak mümkün hale geldi ve bu tür sıkıştırma algoritmalarının (uzun süredir bilinen) BER kodlamalarında yaklaşık %50 azalma sağladığını ortaya çıktı.), ancak aynı zamanda PER kodlamalarında %50'lük bir azalma sağladığını ve bunun (sıkıştırılmamış), sıkıştırılmamış BER kodlamalarında yaklaşık %50'lük bir azalma olduğunu ortaya çıktı. İlginc!

Shannon'in bilgi teorisini uygularsanız, belki de o kadar da şartsızı değildir. Aşağı yukarı bir BER kodlaması, ASN.1 türünün tüm ayrıntılarını ve bu türün degerini ileter. PER, türün tüm ayrıntılarının zaten her iki uçta da bilindiğini varsayıarak yalnızca degerini hakkında bilgileri ileter. Dolayısıyla, sıkıştırılmamış bir PER kodlaması daha az bilgi taşıyır ve

sıkış tırlılmamış bir BER kodlamasından daha küçük olması beklenir, ancak aynı ifade bu kodlamaların sıkış tırlılmış sürümleri için de geçerlidir. Bu pratikte kanıtlanmışdır.

İkinci olarak - ve bu çok u ASN.1 çalış anı için bir sürprizdi - bir ASN.1 PER kodlaması üretmek için gereken CPU döngü sayısının, bir ASN.1 BER kodlaması (ve benzer şekilde kodlama için) üretmek için gereken çok daha AZ olduğu kanıtlandı.). Neden? Niye? Elbette PER daha karmaşıkktır?

Üretilen kodlamayı belirlemenin (hangi kısıtlamaların uygulanacağı, kullanılacak alan genişlikleri, bir uzunluk alanına gerek olup olmadığı) PER için BER'den çok daha karmaşıkktır. Ancak bu kararlılık statiktir. Bir kodlama yapmak için kod oluş turmanın (el ile veya bir ASN.1 "derleyicisi" ile) bir parçasıdır.

Kodlama zamanında, bellekten bir tamsayı almak, alttaki üç biti maskelemek ve bunları kodlama arabalığıne eklemek çok daha az emirdir (bu, "INTEGER (0..7) değil erini kodlamak için PER'nin yapması gereken şeydir. "), bir BER "T" değil eri, bir BER "L" değil eri (bu, eski BER uygulamalarının çoğu göz ardı edildiğiinden, çoğu eski BER uygulaması için tamsayı değil erinin gerçek boyutunun test edilmesi anlamına gelir) oluş turmaktan (ve kodlama arabalığıne eklemekten) daha iyidir. kısıtlamalar) ve ardından gerçek değil er kodlamasının bir veya iki sekizlisi. Benzer şekilde kod çözme için.

Protokol yığınının alt katmanlarını işleyen kodda, PER kullanımdayken işlenecek malzemenin azaltılmış hacminden kaynaklanan ek bir CPU döngüsü kazancı vardır.

Dolayısıyla PER, "eski" protokoller için bile hem bant genişliği içinde hem de CPU döngülerinde iyi kazançlar sağlıyor gibi görünüyor. Bir spesifikasiyonun, uygulama için mantıklı oldukları tamsayılar ve uzunluklar üzerinde sınırlar getirmeye çalıştığı durumlarda, kazançlar çok daha büyük olabilir. Ayrıca, çok fazla boolean "iş areti" olan protokoller de büyük faydalılar. Şekil III-3, BER kodlamasının 19 sekizli ve PER kodlamasının tek bir sekizli olduğu (biraz yapay!) bir Dİ Zİ tipini göstermektedir!

Dİ Zİ {birinci alan ikinci alan üçüncü alan dördüncü alan {dörtA dörtB	TAM SAYI (0..7), BOOLE, TAM SAYI (8..11), SIRALI BOOLE, BOOLE }
--	---

}

Şekil III-3: Kodlama için başka bir örnek Uygulayıcı

topluluğunda, PER kullanımının tip tanımını analiz etmek, hangi kısıtlamaların kodlamayı etkilediğiini belirlemek (ve bu kısıtlamaların muhtemelen uzun parametrelendirme zincirlerini takip etmek) için bir aracın kullanılmasını gerektirdiği dair bir görüşü vardır. değil erleri kodlamak/kodunu çözmek için bir iletişim imzayı kullanmak üzere doğrudu kodlu oluş turmak için.

Elle PER kodlama/kod çözmede hata yapmak, BER'e göre daha kolay olduğu una şüphe yok. PER belirtimi daha karmaşıkktır ve anlaşılması muhtemelen daha az kolaydır. (Dürüst fikrimi istiyorsanız, aslında BER spesifikasiyonundan daha az iyi yazılmış! Mea Culpa!)

Tüm bu noktalar, elle yapmaya çalışmak yerine kodlamalar oluş turmak için iyi hata ayıklanmış bir araç kullanmanın önemini artırır. Ancak PER'nin elle kodlamaları var ve kesinlikle mümkün - ancak başınızı ıslak bir havluyla örtmeye ve bol bol kahve içmeye hazırlıklı olun! Ve daha da önemlisidir, bir araç kullanılarak üretilen kodlamalara/kod çözümlerle karşı test etmek. Bu noktalar, BER'in elle kodlanması için de geçerlidir, ancak çok daha az ücretlidir.

10 Seçkin ve Kanonik Kodlama Kuralı

Kodlayıcı için hiçbir seçenek in olmadığı kodlama kurallarının iyi bir şeyle olduğuunu daha önce gözlemediğim.

Bu tür kodlama kuralları tarafından üretilen kodlamalara genellikle "ayırt edici" veya "kanonik" kodlamalar denir. Bu seviyede (büyük harf yok!) iki terim eş anlamlıdır!

İşiniz Standartlar üretmektir. Kabul edemiyorsanız, isteğe bağlı yapın veya daha iyi başka bir Standart yapın. Ne de olsa, bir Standart iyiye, birçok Standart daha iyi olmalıdır.

Ancak, anlaşamamanız nedeniyle seçenekler sunulursa (BER'deki belirsiz ve belirli uzunluklu kodlamalar gibi - bir sonraki bütme bakın), tüm seçenekler kaldırılarak kodlama kuralları üzerinde nasıl anlaşabilirsınız? Cevap iki Standarttır! Temel Kodlama Kurallarının üç çeşidi vardır:

BER - kodlayıcı için seçeneklere izin verir.

DER (Ayırt Edici Kodlama Kuralları) - tüm seçenekleri belirli bir yönde çizer.

CER (Kanonik Kodlama Kuralları) - tüm seçenekleri doğrulara çizer!

CER'in teknik olarak üstün olduğunu tartışılabilir bir durumdur, ancak DER'nin BER için fiili ayırt edici/kanonik kodlama haline geldiğine şüphe yoktur.

PER'ye geldiğimizde "seçkin" terimi kullanılmaz, ancak daha önce açıklandığı gibi hem hizalanmış hem de hizalanmamış versiyonları olan bir BASIC-PER ve CANONICAL-PER tanımlanmışdır.

"SET OF xyz" tipi kodlamalarla ilgili sorundan daha önce bahsetmiştim. (Sonraki bütümlerde tartışılacak GraphicString ve GeneralString'in kodlanmasıyla ilgili sorunlar da vardır). Resmi anlamda, gönderilmekte olan "xyz" kodlamaları dizisinin soyut düzeyde bir öneği yoktur (bir Dİ Zİ değil, bir SET'tir), dolayısıyla kodlamaların sırası açıkça bir gönderici seçenektedir. Bu türdeki değil erler için tek bir "kanonik" kodlama belirlemek için, "xyz" kodlama serisinin (bu kodlamaların her birinin ikili değil erine dayalı olarak) tanımlanmış bir sıraya göre SIRALANMASI gereklidir. Bu, CPU döngülerine ve ayrıca "disk çalkalamasına" çok önemli bir yük getirebilir ve hafife alınacak bir şeyle değilidir!

Dolayısıyla, bir spesifikasyon "SET OF" kullanımını iç eriyorsa, "normal PER" tam anlamıyla kanonik değil (ancak burada "bir toplu işlenenin ucuna kaç melek oturabilir" konularına girdiği imizi iddia edenler olsa da).

"Kanonik PER", "xyz" kodlamalarının, (sırasız) bir değil er kümesinin bit dizilerine gerçek ekten bire bir eşlemesini üretmek için sıralamasını belirtir; her bit dizisi, "xyz" türündeki olası (sırasız) bir değil er kümesini temsil eder. .

Yazarın görüşü: Bu derecede formalite veya kesinlik in önemli olduğunu hiç bir uygulama bilmiyorum. CANONICAL-PER temelde iyi bir fikir değil, ancak özelliklerde "SET OF" kullanımı da değil! İkisinden kaçınmaya çalışın. (Diğer erleri aynı fikirde olmayabilir!)

11 Sonuç

Bu bölüm, ASN.1 Temel Kodlama Kuralları ve ASN.1 Paket Kodlama Kurallarına bir giriş sunarak, bunların kodlamalara yaklaşımlarını ve göreceli avantajlarını ve dezavantajlarını göstermektedir.

Ayrıca geniş letilebilirlik veya "geleceğe yönelik koruma" konularını tartışmış ve kanonik/ayırt edici kodlama kurallarından bahsetmiş tir.

Bu bölüm, sonraki iki bölümde BER ve PER'nin ayrıntılı, olgusal (ve kuru!) açıklamasına temel bir giriş oluşturmuş tur.

Okuyucular ayrıca "Minimum Bit Encoding Rules" (MBER), "Lightweight Encoding Rules" (LWER), "Clear text encoding Rules", "BACNet Encoding Rules", "Oturum Katmanı Encoding" gibi isimlerle ASN.1 Kodlama Kurallarını da duymuş olabilirler. Kurallar" ve belki de erleri. Bunlar, ASN.1 için bazı durumlarda hem BER hem de PER'den üstün olabilecek (veya PER'ye doğrudan ilerlemek için kısmen erken giriş imler olan) başlangıç Kodlama Kuralları geliş tirmeye yönelik giriş imleri (bazen standartlar topluluğunu dışında, bazen onun içinde) temsil ediyordu. Bunların hiçbiri bugün ASN.1 ile genel kullanım için önemli görülmemektedir, ancak bunlar bu bölümün dördüncü (kısa) bölümünde biraz daha ayrıntılı olarak ele alınmaktadır.

Bölüm 2

Temel Kodlama Kuralları

(Veya: 80'ler için kodlamalar - basit, sağ ıam ama verimsiz!)

Özet: Bu bölüm, Temel Kodlama Kurallarının ayrıntılarını sağ ıar. Açıklar:

İ lkel/inş a edilmiş bit dahil olmak üzere bir TLV kodlamasının (tanımlayıcı sekizlileri) T kısmının biçimi.

TLV'nin L kısmı (uzunluk sekizlileri) için kısa, kesin ve belirsiz kodlama biçimleri.

Kabaca artan karmaş ıkkılık sırasına göre alınan, ilkel türlerin her biri için TLV kodlamasının V kısmı (içerik sekizlileri).

Oluş turulan türlerin kodlaması (SET ve SEQUENCE gibi)

Karakter dizisi ve zaman türleri ve çeş itli türden "delikleri" temsil eden türler gibi kalan türlerin kodlanması.

1. Giriş

BER kodlamalarının altında yatan TLV ilkeleri, önceki böümlerde kapsamlı bir ş eklide tanıtılmıştır ve okuyucu, yetkili ayrıntılar için gerçek Standarda/Öneriye gitmekte çok az zorluk çekmelidir.

BER kodlamalarının ilkelерini zaten
öğrendiniz, şimdiden
ayrintılara geçelim

Bununla birlikte, bütünlüğ ısağ ıamak için, bu bölüm tüm kodlamaların örneklerini sağ ıar ve birkaç durumda daha fazla açıklama verir.

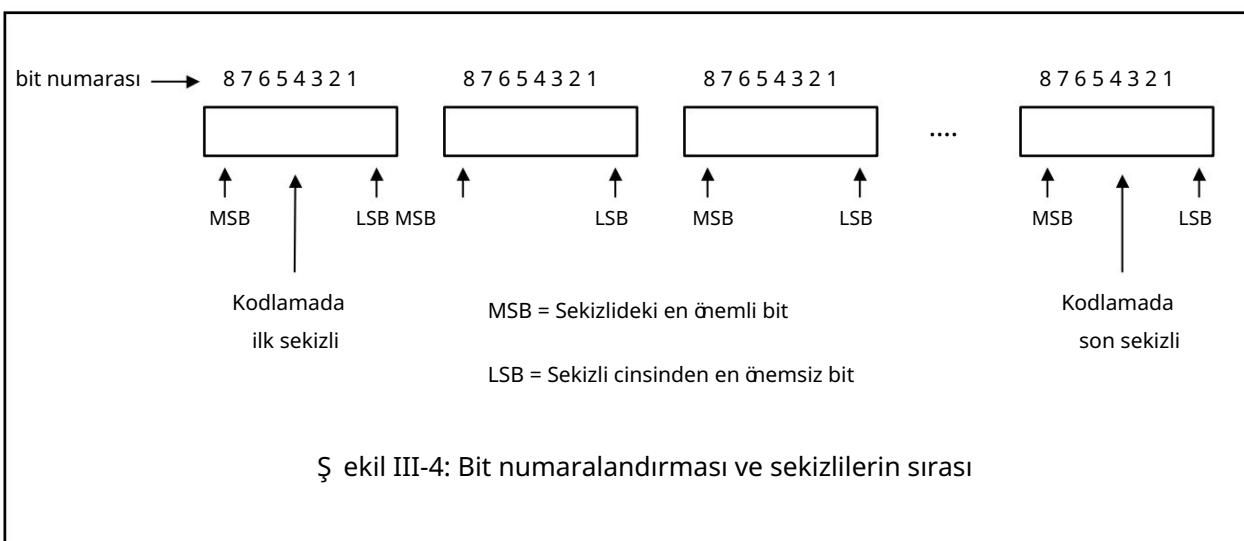
2 Genel sorunlar

2.1 Bit sayıları ve diyagramlar için notasyon

1970'lerin sonlarında kodlama özellikleriyle ilgili sorunlardan biri, bir sekizlinin bitlerinin diyagramlarda bazen soldan sağ a, bazen diğ er ş ekilde numaralandırılması ve bazen en önemli bitin sağ da, bazen de solda gösterilmesiydi. . Diyagramlardan sekizli aktarım sırası da bazı özelliklerde sağ dan sola, b olarak, genellikle kafa karış ıklıq i vardı!

Spesifikasyonlar, bir sekizlideki bitlerin
hangi sırayla gönderildiği konusunda
net değilse, sorunlar olacaktır!
1970'lerin sonlarında vardı - şimdilerde
büyük ölçüde tamam, ama dikkatli olun.

ASN.1 (ve bu kitap) durumunda, ilk iletilen sekizliyi solda (veya yukarısında) daha sonra iletilen sekizlileri gösteririz ve her sekizliyi en önemli biti solda, bit sayıları 8'den başlayarak gösteririz. (en önemli) ile 1 (en az önemli) arasında \S ekil III-4'te gösterildiği gibi.



Bir sekizli içinde en önemli bitin mi yoksa en öneşsiz bitin mi önce iletildiği (veya bitlerin paralel olarak iletildiği) ASN.1'de belirtilmemiş tir. Bu, taş ıycı protokoller tarafından belirlenir. Bir seri hatta, en önemli ilk en yaygın olandır. Hattaki bitlerin sırasının belirlenmesi ASN.1 spesifikasyonlarını alt katman taş ıycı spesifikasyonlarına bağlayan "en anlamlı bit" ve "en az anlamlı bit" terimleridir.

Hattaki sekizlilerin sırası tamamen ASN.1 tarafından belirlenir. ASN.1, çok sekizli bir tamsayı değeriini kodlarken, değerini en önemli sekizlisinin önce iletildiğiini belirtir ve bu nedenle standarttaki (ve bu kitaptaki) şemalardada değerini en soldaki sekizlisi olarak gösterilir (bu bölümün ilerleyen kısımlarında tamsayı türünün kodlamasına bakın).

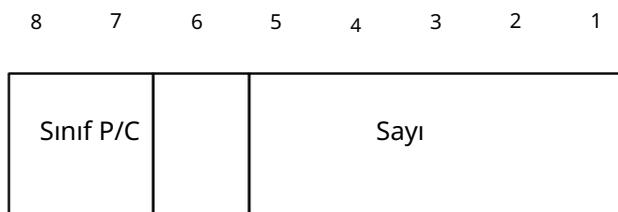
2.2 Tanımlayıcı sekizlileri

Her ASN.1 türü, daha önce tartışıldığı gibi, etiket için bir numara ile birlikte dört sınıftan birinin etiketine sahiptir. En basit durumda, bu değerler ş. ekil III-5'te gösterildiği gibi tek bir sekizli olarak kodlanmışdır.

**İlk olarak etiket
değerini kodlayan
T kismi.**

İlk iki bitin sınıfı ş. u. ş. ekilde kodladığıını görüyoruz:

Sınıf	Bit 8	Bit 7
evrensel 0	0	
Uygulama 0	1	
Bağlama özgü 1	0	
Özel 1	1	

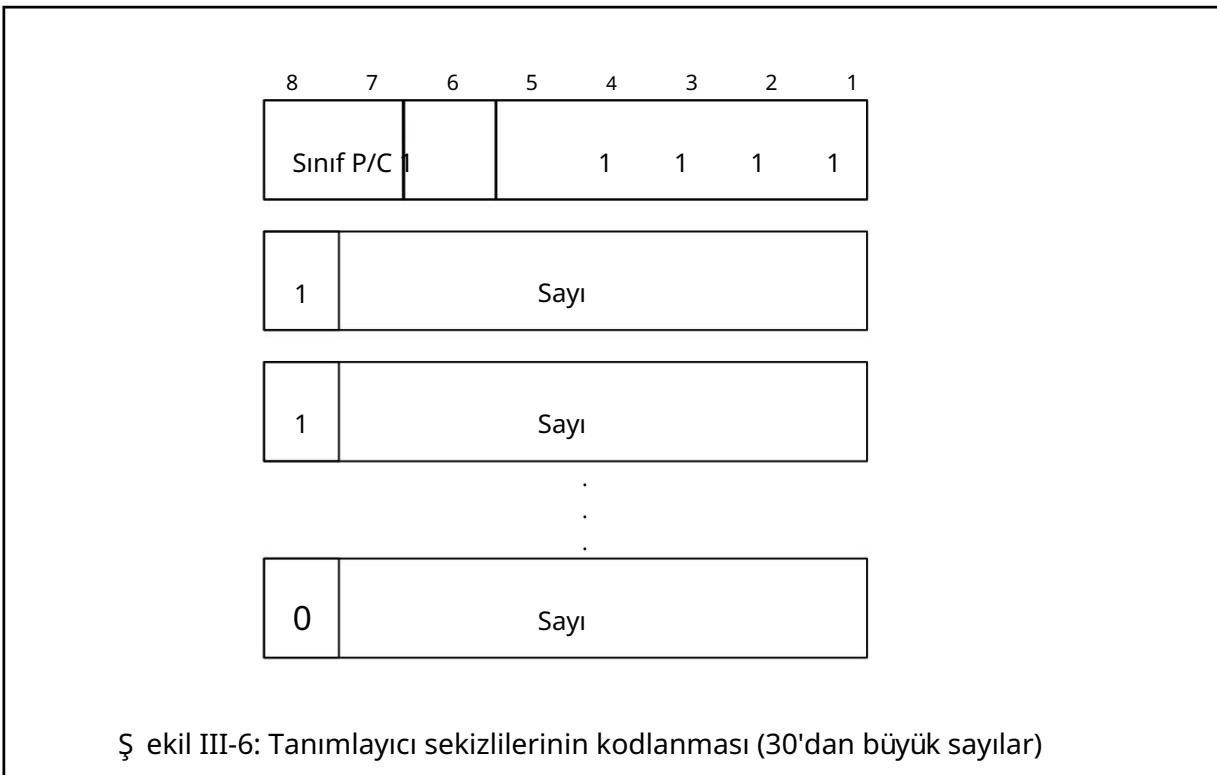


Ş. ekil III-5: Tanımlayıcı sekizlisinin kodlanması (sayı 31'den küçük)

Bir sonraki bit (altıncı bit), ilkel/inş. a edilmiş (P/C) bit olarak adlandırılır ve buna birazdan geri döneceğiz.

Son beş bit (5'ten 1'e kadar olan bitler) etiketin numarasını kodlar. Açıksa, bu sadece 32'den küçük sayılarla başa çıkacaktır. Aslında, 31 değer eri bir çıkış işaretçisi olarak kullanılır, bu nedenle yalnızca 30'a kadar olan etiket sayıları tek bir sekizlide kodlanır.

Daha büyük etiket değer erleri için, ilk sekizlinin tümü 5 ile 1 bitlerinde birlere sahiptir ve daha sonra etiket değer eri, her sekizlinin yalnızca en ölümsiz yedi biti kullanılarak ve minimum sayı kullanılarak, gerektiği kadar takip eden sekizlide kodlanır. kodlama için sekizli. En ölümsiz bit ("daha fazla" bit), takip eden ilk sekizlide 1'e ve sonda sıfırı ayarlanır. Bu, § ekil III-6'da gösterilmektedir.



Böylece 31 ile 127 (dahil) arasındaki etiket numaraları iki tanımlayıcı sekizli üretenecek, 128 ile 16383 arasındaki etiket numaraları üç tanımlayıcı sekizli üretenecektir. (Çoğu ASN.1 spesifikasyonu, etiket numaralarını 128'in altında tutar, bu nedenle, normalde görülecek iniz şeyle ya 1 tanımlayıcı sekizli - en yaygın - ya da iki tanımlayıcı sekizli olur, ancak ben 999 etiket numarası gördüm!)

Peki ya ilkel/inş a edilmiş bit? Kodlamanın V bölümünün kendisi bir dizi TLV kodlaması, bunun 1'e (yapılandırılmış) ayarlanması ve aksi takdirde 0'a (ilkel) ayarlanması gereklidir. Bu nedenle, bir tamsayı türünün veya boole türünün kodlaması için (herhangi bir etiketlemenin örtülü olması koşuluyla), her zaman 0'a ayarlanır. Bir SET veya SET-OF vb. kodlaması için, her zaman 1'e ayarlanır. kod çözümünün mevcut tip tanımına sahip olması koşuluyla, açılık gereksizdir.

Ancak bu bitin mevcut olması, gelen sekizli akışın önce TLV kodlamalarının bir ağ açısına ayrılmışlığı (tip tanımı hakkında bilgi olmadan) bir kod çözme mimarisi stiline izin verir, böylece ağ açılarının yapraklarının tümü ilkel kodlamalardır. Ağ açıları sonra daha fazla işlenmek üzere tür tanımını bilen koda geçerlidir.

Bununla birlikte, bu parça için oldukça daha önemli bir rol var. Daha sonra görülecek imiz gibi, çok uzun bir sekizli dizi değer eri iletilirken (ve aynısı bit dizisi ve karakter dizisi değer erleri için geçerlidir), ASN.1 kodlayıcının sekizli dizi değerinin sekizlilerini tüm V bölümünü olarak iletmesine izin verir (önce bir uzunluk sayımı gelir) veya sekizli diziyi, her biri TLV kodlamalarına dönüşümlenir ve daha sonra sekizli dizi değerinin ana dış düzey kodlamasının V kısmına giden bir dizi parçaya bölmek. Açıkçası bir kod çözümünün hangi seçenekin alındığıını bilmesi gereklidir ve ilkel/inş a edilmiş bit ona tam olarak bunu söyler.

Bu şekilde parçalama neden yararlıdır? Bu, "L" kodlamasının biçimini ele aldığıımızda bir sonraki Maddede daha net hale gelecektir, ancak sorun kabaca aşağıda gibidir.

V parçamız ilkel ise, açıkça tüm olası sekizli değerleri içinde görünebilir ve ASN.1'in uzunluğunu belirlemek için sağladığını tek mekanizma, "L" bölümünde açık bir sekizli sayısına sahip olmaktadır. Aşağıda uzun sekizli değerler için bu, gerçek sekizlilerden herhangi biri gönderilmeden önce tam uzunluğu belirlemek (ve iletmek) için çok fazla disk çalkalaması anlamına gelebilir. Bununla birlikte, V kısmı bir dizi TLV'den oluşan uyorsa, bu TLV serisini önden sayım yapmadan sonlandırmanın yollarını bulabiliriz, böylece sekizlileri saymak zorunda kalmadan kullanılabilir olduklarında değerlerden iletебiliriz. hepsi ilk.

2.3 Uzunluk sekizlileri

BER'de kısa biçim, uzun biçim ve belirsiz biçim olarak adlandırılan üç uzunluk kodlama biçimleri vardır. Her üç formu da kullanmak her zaman mümkün değildir, ancak olduğunda yerde hangisinin kullanılacağı kodlayıcının tercihidir. Bu, BER'deki ana isteği ebağı ile kaynaklarından biridir ve kanonik/ayırt edici kodlama kurallarının ele alması gereken ana alandır.

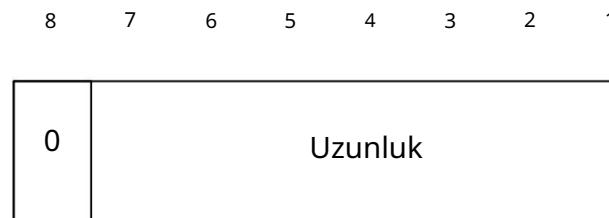
Şimdi L kısmı - genel olarak üç form mevcuttur, bazen sadece iki ve bazen sadece bir tane. Kodlayıcı kullanılacak olanı seçer.

2.3.1 Kısa biçim

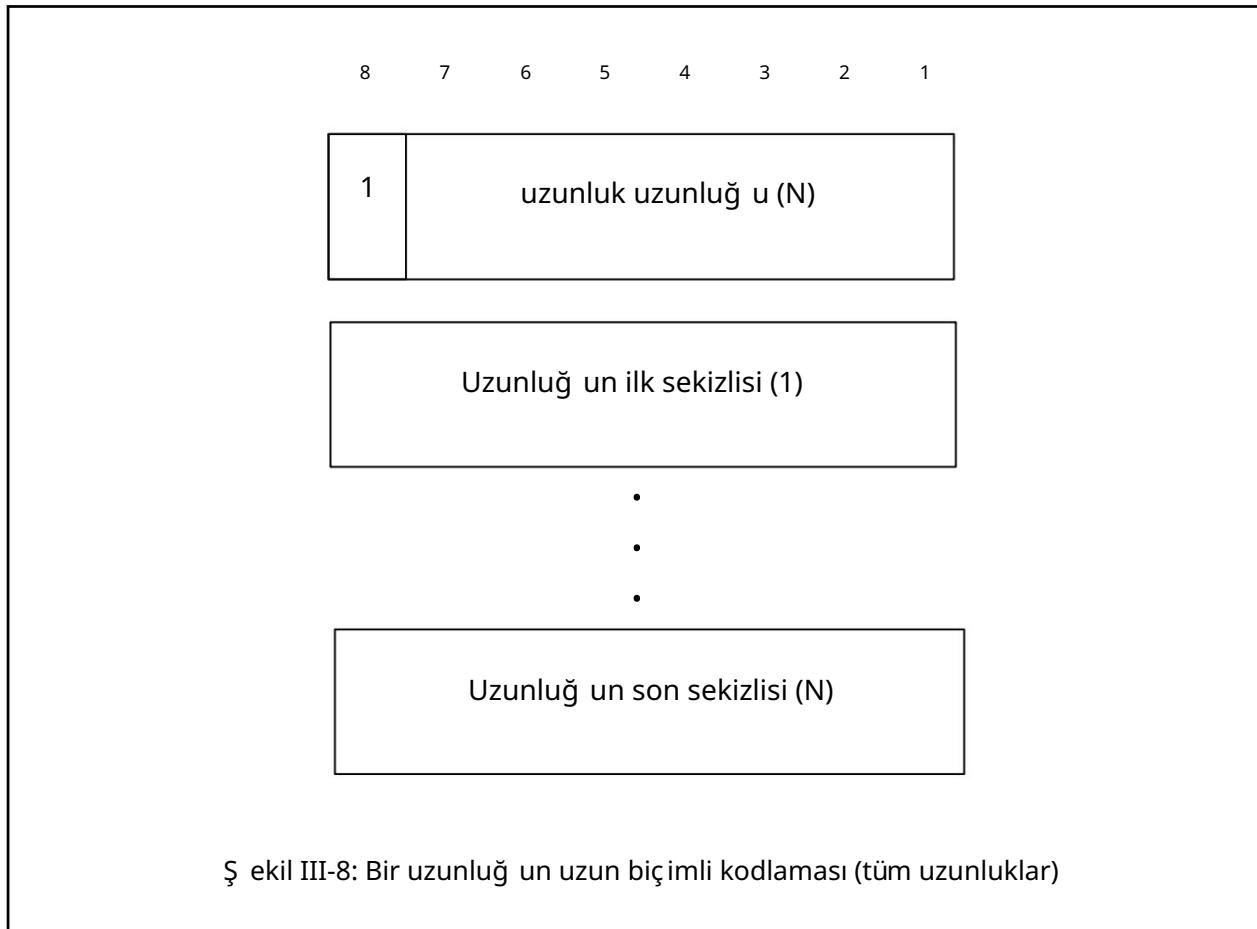
Bu, Şekil III-7'de gösterilmektedir.

V bölümündeki sekizli sayısı 127'den az veya eşittse kısa biçim kullanılabilir ve V bölümü ister ilkel ister yapılandırılmış olsun kullanılabilir. Bu form, bit 8'in sıfır olarak kodlanmasıyla tanımlanır, uzunluk sayımı 7'den 1'e kadar olan bitlerdir (her zamanki gibi, uzunluğu en önemli biti 7 olan bittir).

Çok açık - üst biti sıfır olan bir sekizli ve kalan bitler uzunluğu bir sekizli sayısı.



Şekil III-7: Bir uzunluğun kısa biçim kodlaması (127'ye kadar olan uzunluklar)



2.3.2 Uzun bitim

İlk uzunluk sekizlisinin 8. biti 1'e ayarlanırsa, o zaman uzun uzunluk bitimine sahibiz. Bu form, ne kadar uzun veya kısa olursa olsun, ister ilkel ister yapılandırılmış olsun, her türlü V parçası için kullanılabilir. Bu uzun formda, ilk sekizli, kalan yedi bitinde, kendileri V bölümünün uzunluğuunu kodlayan bir sekizli dizisinin uzunluğu u olan bir N değerini kodlar. Bu, Şekil III-8'de gösterilmiştir.

Daha uzun uzunluklar için - uzunluk alanının uzunluğu u, ardından uzunluğu un kendisi. Genellikle toplam üç sekizli, ilki ikiye ayarlanır.

Gerçek uzunluğu kodlamak için minimum sekizli sayısının kullanılması gerekliliği yoktur, dolayısıyla V bölümünün gerçek uzunluğu 5 ise, Şekil III-9'da gösterilen tüm uzunluk kodlamalarına izin verilir.

Bu aslında ASN.1'e 1980'lerin başında, ilk spesifikasiyon sonlandırılmışdan hemen önce dahil edildi (ilk taslaklar, uzunluk kodlamalarının mümkün olduğu unca küçük olmasını gerektiriyordu). N'nin sabit bir değer (tipik olarak 2) sahip olmasını, ardından gerçek uzunluk değerini tutacak N (2) sekizlisini ve ardından V parçasını isteyen birkaç uygulayıcı olduğu için tanıtıldı. Bir sekizlinin (kısa bitimli kodlamayı kullanarak) yaptığı yerde bile, muhtemelen bugün hala her zaman üç uzunluklu sekizliye sahip (uzun bitimli kodlamayı kullanan) BER uygulamaları vardır.

8 7 6 5 4 3 2 1

0 0 0 0 0 1 0 1

8 7 6 5 4 3 2 1

1 0 0 0 0 0 1 0

8 7 6 5 4 3 2 1

0 0 0 0 0 0 0

8 7 6 5 4 3 2 1

0 0 0 0 0 1 0 1

8 7 6 5 4 3 2 1

1 0 0 0 0 0 1 1

8 7 6 5 4 3 2 1

0 0 0 0 0 0 0

8 7 6 5 4 3 2 1

0 0 0 0 0 0 0

8 7 6 5 4 3 2 1

0 0 0 0 0 1 0 1

vb

Şekil III-9: 5 uzunluğunu kodlama seçenekleri

Uzun formda ilk uzunluk sekizlisinde bir kısıtlama vardır. N'nin 127 değil erine sahip olmasına izin verilmez. Bu, "gelecekteki uzantılar için ayrılmıştır", ancak bu tür uzantılar artık pek olası değil. N'nin maksimum değil eri 126 olduğunu V parçasının ne kadar uzun olabileceğini ve böyle bir V parçasının ne kadar büyük bir tamsayı değil eri tutabileceğini düşündürseniz, sayının galaksimizdeki yıldız sayılarından daha fazla olduğunu görsünüz. Ayrıca, mümkün olan en uzun V parçasını saniyede bir terabit hızında çalışın bir hatta iletirseniz, tüm sekizlileri iletmenin yüz milyon yıl alacağı da hesaplandı! Bu nedenle, BER tarafından V parçasının boyutuna veya tamsayıların değil erine dayatılan pratik bir sınır yoktur.

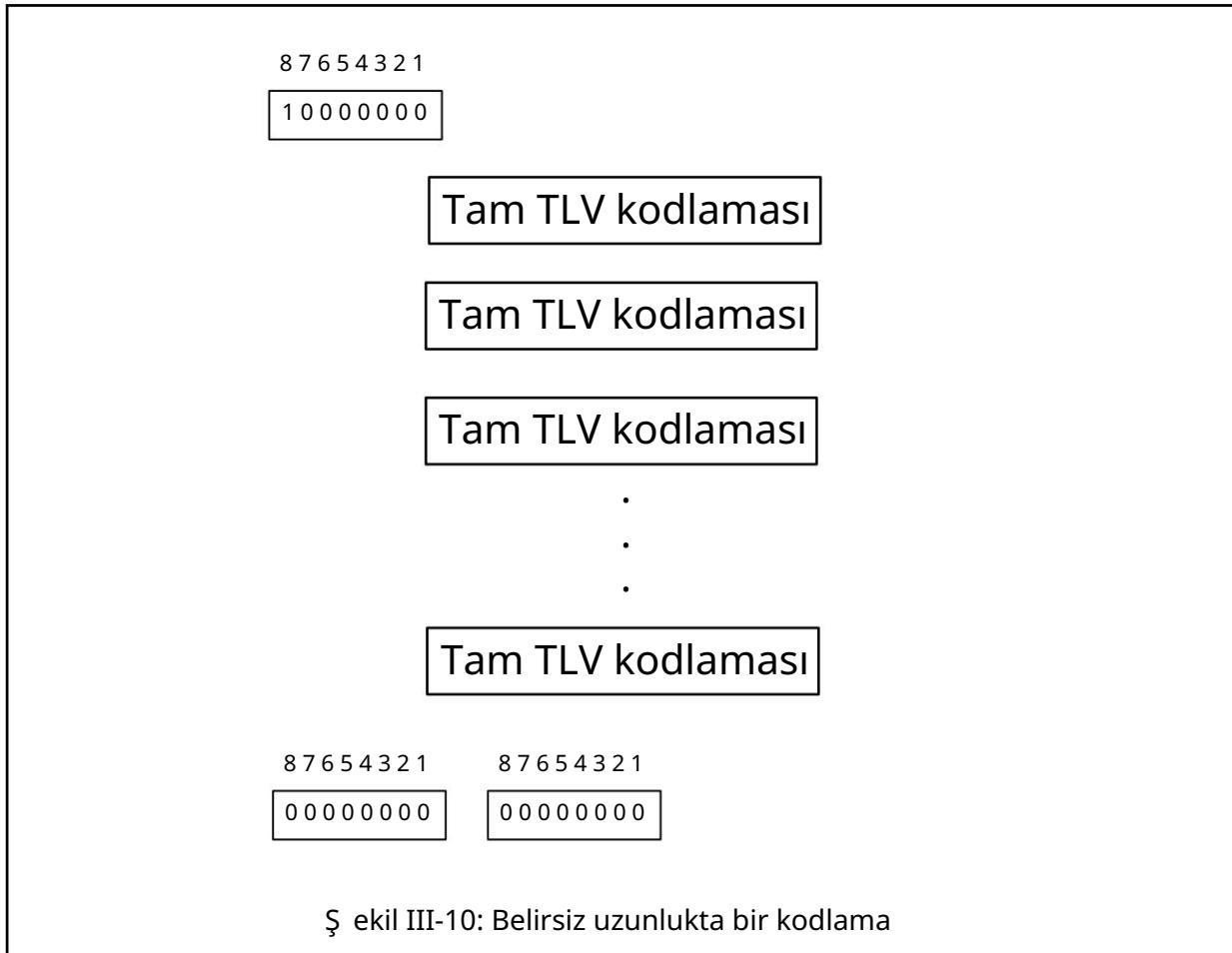
2.3.3 Belirsiz biçim

Belirsiz uzunluk biçimini, yalnızca V parçası oluşturulan sa, yani bir dizi TLV'den oluşuyorsa kullanılabılır (ancak kullanılması zorunlu değil ildir). (Bu içeren serideki bu TLV'lerin her birinin uzunluk sekizlisidir, bu tür seçeneklerin mevcut olduğunu yerlerde bağımsız olarak kısa, belirli veya belirsiz olarak seçilebilir - dış düzeyde kullanılan biçim iç kodlamayı etkilemez.)

Bu formla, sekizlilerinizi göndermeye başlamadan önce saymanız gerekmektedir - sadece bir dizi parça gönderin ve 0000 ile sonlandırın.

Belirsiz uzunluk biçiminde, ilk sekizlinin ilk biti, uzun biçimde olduğunu gibi 1'e ayarlanır, ancak N değil eri sıfır ayarlanır. Açıkçası, N içerişinde sıfır değil eri uzun formda yararlı olmaz, bu nedenle bu, belirsiz formun kullanıldığıunu gösteren bir işaret görevi görür. Bu tek sekizli takiben, V bölümünü oluşturulan TLV serisini ve ardından bir çift sıfır sekizli olan özel bir sınırlayıcıyı alıyoruz.

Bu, Şekil III-10'da gösterilmişdir.



Bu nasıl çalışır? Unutulmaması gereken en önemli şey, bir kod çözümcünün TLV serisini işlemesidir ve sıfır sekizli çiftine çarptığıında bunları başka bir TLV'nin başlangıcı olarak yorumlayacaktır. Öyleyse sadece bunu yapalım. Sıfır T, UNIVERSAL sınıfı ZERO etiketi ve sıfır uzunluğu (V bölümünde sıfır sekizli) belirli bir form uzunluğu kodlaması ile ilkel bir kodlamaya (altı bit sıfırdır) benziyor.

Şimdi § ekil II-7'de verilen UNIVERSAL sınıf etiketlerinin atanmasına geri dönerseniz, UNIVERSAL sıfır sınıfının "Kodlama Kuralları tarafından kullanılmak üzere ayrılmış tır" olduğuunu göreceksiniz (ve kullanıcıların UNIVERSAL sınıf etiketleri atamasına izin verilmemişini unutmayın). Dolayısıyla, bir çift sıfır sekizli hiç bir zaman herhangi bir gerçek kodlamada bir TLV olarak görünemez ve bu "özel" TLV, BER tarafından belirsiz bir biçim kodlamasının V kısmındaki TLV serileri için sınırlayıcı olarak güvenli bir şekilde tanımlanabilir.

Belirsiz bir TLV biçimini içinde, kendileri inş a edilmiş ve belirsiz bir uzunluk biçimine sahip olan iç TLV'lere sahip olabileceğimizi daha önce söylemiş tık. Karışıklık yok: bir çift sıfır sekizli (bir TLV beklendiğinde), en içteki "acık" belirsiz formu sonlandırır.

2.3.4 Uzunluk varyantlarının tartışılması

Neden bu kadar çok farklı uzunluk çeşidiine ihtiyacımız var? Açıktı ası hepsinin bazı avantajları ve dezavantajları var. Kısa biçim, kullanılabilirliği zaman en kısa biçimdir, uzun biçim, çok büyük ilkel kodlamaları işleyebilen tek biçimdir ve birçok kişi iye sezgisel olarak belirsiz biçimden daha basit görünür. Belirsiz, çok büyük OCTET STRING'e izin veren tek karakterdir.

Baş İamadan önce değ erdeki sekizli sayısı sayılmadan iletilecek değ erler veya SEQUENCE OF değ erler.

Üç seçenek e sahip olmanın dezavantajı, kod çözümlerdeki ekstra uygulama karmaş ıklı̄ğ ı ve yan kanallar ve ekstra hata ayıklama çabası oluşturulan kodlama seçeneklerinin varlığı idir. Bu seçenekleri kaldırı̄lmak istiyorsak, o zaman ya "mükün olduğunda belirsiz uzunluktaki formu kullan" demeliyiz (ve bir sekizli dizgiyi parçalarken kullanılacak parçaın boyutu hakkında açıklamalar yapmalyız) ya da "mükün olan yerlerde kısa formu kullan" demeliyiz. aksi takdirde, sayılm için gereken minimum N değ erine sahip uzun biçimini kullanın". Bu yaklaşımımların her ikisi de standardize edilmiş tir! İlk yaklaşımı benimseyen ayı̄rt edici/kanonik kodlama kurallarına Kanonik Kodlama Kuralları (CER) ve ikinci yaklaşımı benimseyenlere Ayı̄rt Edici Kodlama Kuralları (DER) adı verilir. Kanonik/ayı̄rt edici kodlama kuralları gereksinimleri olan uygulamalar, uygulama belirtiminde bunlardan birinin kullanılmasını zorunlu kılacaktır.

3 Ana tiplerin V kısmının kodlamaları

Bu madde için verilen örneklerde, bir türün değ erini belirtmek için ASN.1 değ er gösterimini kullanıyoruz ve ardından her sekizlinin değ eri için onaltılık gösterimi kullanarak bu değ erin tam kodlamasını gösteriyoruz.

Buradaki birincil odak noktası, her tür için V bölümünün kodlamasını göstermektedir, ancak gösterilene ek olarak (daha önce tartışıldığ gibi) baş ka izin verilen uzunluk kodlamalarının olacağ ve örtülü etiketleme olsaydı, hatırlanmalıdır. uygulandığıında, T kısmı farklı olacaktır.

V bölümünün kodlanması her türde özeldir. Çok u durumda bu açıkta, ancak türlerin çok u, kodlamayı çok az karmaş ık hale getiren sorunlar çıkarır.

giriş

Aşağıdaki türlerin her birinin kodlaması, aksi belirtilmedikçe her zaman ilkeldir. Türler kabaca artan karmaş ıklık sırasına göre alınır!

3.1 NULL değ erini kodlama

Tamamen basit!

Değ eri

boş BOŞ ::= BOŞ

(NULL türünün tek değ eri) ş u ş ekilde kodlanır:

	T	L	V
hükümsüz:	05	00	boş

Yapımızı TLV olarak tanımlamış olsak da (bu durumda olduğu gibi) uzunluk sıfırsa V bölümünde sıfır sekizli olması mümkün değildir. Bu, NULL dışındaki durumlarda ortaya çıkabilir. Örneğin, yineleme sayısı sıfır olan bir SEQUENCE OF değ eri sıfır L ile kodlanır.

Benzer ş ekilde, tüm öğeleri isteğe bağlı olan ve bir iletişim im örneğinde tümü eksik olan bir Dİ Zİ, yine bir L sıfır ile kodlayacaktır.

3.2 BOOLEAN değerleri kodlama

değer erleri

boolean1 BOOLEAN ::= DOĞRU

boolean2 BOOLEAN ::= YANLIŞ

Hala oldukça açık, ancak artık kodlayıcı seçeneklerimiz var!

olarak kodlanır

	T	L	V
boolean1:	01	01	FF
boolean2:	01	01	00

TRUE değer eri için onaltı FF kodlaması gösterilir. Bu, DER ve CER'de izin verilen tek kodlamadır, ancak BER'de V kısmı için sıfır olmayan herhangi bir değerere izin verilir.

3.3 INTEGER değerini kodlama

Tamsayı değerlerinin mümkün olan en küçük V parçasına ikiye tümleyen kodlaması belirtilir. İlkinin tümleyeni kullanıldığıında "mükemmel olan en küçük", V bromiumun ilk (en önemli) dokuz bitinin tümü sıfır veya tümü bir olamaz, ancak ilk sekiz bit ile tümü sıfırları veya birleri kodlayacak değerler olacaktır.

İlk dokuz bit aynı olmamalıdır - dokuz? Evet DOKUZ.

Teorik olarak, sıfır tamsayı değerini temsil etmek için V parçası olmadan bir L değerini kullanmak mümkün olabilirdi, ancak bu BER tarafından açıkça yasaklanmışdır - V bromiumde her zaman en az bir sekizli vardır.

Böyleden değerleri

tamsayı1 TAM SAYI ::= 72

tamsayı2 TAM SAYI ::= 127

tamsayı3 TAM SAYI ::= -128

tamsayı4 TAM SAYI ::= 128

olarak kodlanır

	T	L	V
tamsayı1	02	01	48
tamsayı2	02	01	7F
tamsayı3	02	01	80
tamsayı4	02	02	0080

Tamsayı türü, ayırt edici bir değer eri listesiyle tanımlanıysa, bu, kodlamayı hiç bir şekilde etkilemez.

3.4 SAYILANDIRILMIŞ bir dē eri kodlama

Numaralandırılmış bir türün tanımı, aktarım sırasında her bir numaralandırmayı temsil etmek için kullanılacak tamsayı dē erler içerebilir veya (1994 sonrası) bu dē erlerin sıfırdan başlayarak otomatik olarak atanmasına izin verebilir. İlkinci durumda, bu tür dē erlerin tümü pozitif olacaktır, ancak genel durumda, bir kullanıcının numaralandırmalar için negatif dē erler atamasına izin verilir (hiç kimse bunu yapmaz). BER, ilişkili tüm dē erlerin pozitif olduğunu (yayın) durumu dikkate almaz: numaralandırılmış bir dē erin kodlaması, ilişkili tamsayı dē erinin (ikiye tümleyen) kodlamasıyla tamamen aynıdır (elbette etiket dē erinin farklı olması dışında).

Numaralandırmalar için yalnızca pozitif dē erler bekleyebilirsiniz - öyle dē il! Genel bir tamsayı olarak kodlayın.

Uygulamada, bu yalnızca 127'den fazla sayı varsa verimlilik farkı yaratır ki bu nadirdir.

3.5 GERÇEK dē eri kodlama

Gerçek bir dē erin kodlanması oldukça karmaşaktır. Her şeyleden önce, türün resmi olarak, aynı sayısal dē er olsalar bile, 2 tabanında ifade edilebilecek tüm olası dē erler kümlesiyle birlikte 10 tabanında ifade edilebilen tüm dē erlerin kümlesi olarak tanımlanmış inî hatırlayın.

Kayan nokta biçim standartlarını unutun. Önemli olan, gerçek donanımla ne kadar kolay kodlayabileceğiniz/kodlayabileceğiniz inizdir.

Bu, bu iki dē er kümnesine farklı kodlamaların uygulandığı ve uygulamanın farklı anımlar uygulayabileceğine anlamına gelir. (Bunun bir istisnası vardır - sıfır dē erinin yalnızca bir kodlaması vardır, V bölümünde sıfır sekizli vardır.) 10 tabanlı dē erler için kodlama karakter tabanlıdır, 2 tabanlı dē erler için ikili kayan noktadır.

Ayrıca REAL tipinde iki dē er daha vardır - PLUS-INFINITY ve MINUS-INFINITY, kendi özel kodlamaları ile.

REAL alt türünün yalnızca 10 tabanlı veya 2 tabanlı dē erleri içermesinin mümkün olduğunu unutmayın; bu, uygulama tasarımcısına türün dē erlerinin karakter tabanlı kodlamasının mı yoksa ikili tabanlı kodlamasının mı kullanılacağı konusunda etkili bir şekilde kontrol sağlar.

3.5.1 Kodlama tabanı 10 dē erleri

(Sıfır olmayan) dē er 10 tabanısa, o zaman iç erik sekizlileri (V bölümü), ilk iki biti 00 olan bir sekizli ile başlar (diğer dē erler 1 tabanı dē erleri ve PLUS-INFINITY ve MINUS özel dē erleri için kullanılır) -SONSUZLUK). Bu ilk sekizliden sonraki sekizliler, 0 ile 9 arasındaki rakamları, boş luk, artı işareti, eksiz işareti, virgül veya nokta ("ondalık işareti" içinde) ve büyük E ve küçük e'yi temsil eden bir dizi ASCII karakteridir (karakter başına 8 bit). (Üsler içinde), ISO Standardı 6093'te tanımlanan bir formattır. Bu standardın birçok seçenek varıdır ve özellikle "Sayısal Gösterim 1" (NR1), NR2 ve NR3'ü tanımlar. Bunlardan hangisinin kullanıldığı, ilk iç erik sekizlisinin alt altı bitine sırasıyla 1, 2 veya 3 dē erleri olarak kodlanır. Bu temsiller içinde bile birçok seçenek vardır. Özellikle, gelişigüzel birçok onde gelen boş luk dahil edilebilir, ayrıca işaretle istege bağlıdır, vb.

10 tabanlı bir karakter kodlaması mevcuttur. (Fakat fazla kullanılmadı!)

DER ve CER (ve PER'nin tüm sürümleri) ile birlikte kullanıldığıında, seçenekler NR3 ile sınırlıdır, boş luklar ve baş taki sıfırlar genel olarak yasaktır, herhangi bir "ondalık iş areti" için nokta kullanılmalıdır ve artı iş areti gereklidir Pozitif değil erler için. Mantisin "ondalık iş aretinden" sonra basamak olmaması için normalleş tirilmesi gereklidir. Aşağıdaki her durumda, ikinci sütun aynı gerçek değil erin DER/CER/PER'de nasıl kodlanacağıını gösterir.

Burada ISO 6093'ün ayrıntılı bir açıklamasını yapmaya çalışmayacağınız, ancak aşağıda ortaya çıkan dizilerin bazı örneklerini vereceğiz. Başta boş luklar olabilirken, hiçbir zaman sonunda boş luk olmadığıını unutmayın. Baş taki sıfırlar ve sondaki sıfırlar da olabilir.

NR1 yalnızca basit tam sayıları kodlar (ondalık nokta yok, üs yok). İşte '#in boş luk karakterini belirtmek için kullanıldığı NR1 kodlamalarına bazı örnekler:

4902	4902.E+0
#4902	4902.E+0
###0004902	4902.E+0
###+4902	4902.E+0
-004902	-4902.E+0

NR2, bir "ondalık iş aretinin" varlığıını gerektirir (kodlayıcı seçenek olarak nokta veya virgül). NR2 kodlamalarına bazı örnekler:

4902.00	4902.E+0
###4902,00	4902.E+0
000.4 #.4 4.	4.E-1 4.E-1
	4.E+0

NR3, NR2'yi büyük E veya küçük harf e ile temsil edilen 10 tabanlı bir üssün kullanımıyla genişletir.

NR3 örnekleri şunlardır:

+0,56D+4	56.E2
+5,6e+03	56.E2
#0,3E-04	3.E-5
-2,8D+000000	-28.E-1
####000004.50000E123456789	45.E123456788

3.5.2 Kodlama tabanı 2 değil erleri

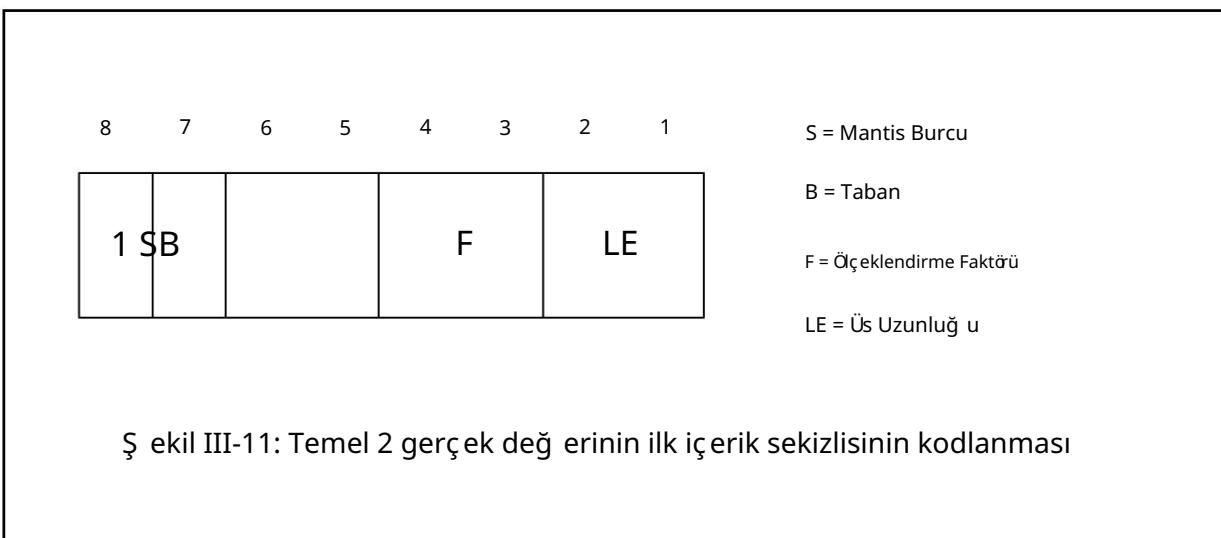
NOT — Bu materyali tam olarak anlamak için okuyucunun bilgisayar kayan nokta birimleri biçimine biraz aşağı olması gereklidir - bu, 1960'ların birleşmiş dili programclarının çok aşağı olduğu bir şevidir, ancak bugünün programclarının genellikle unutabileceğiniz bir şevidir! Bu materyali çok hızlı bir şekilde gözden geçirmek, hatta tamamen görmezden gelmek isteyebilirsiniz.

Daha "geleneksel" bir kayan nokta biçimimi de mevcuttur. Kodlamayı/kod çözmemi kolaylaş tırmak için çaba harcanan yer burasıdır.

Temel 2 değil erleri, bir bilgisayar sistemi bir kayan nokta biriminin içeriğini ana belleğe döktüğünde kullanılan kayan nokta biçimlerine benzer bir biçimde kodlanır. Sayının mantisinden (M), tabanından (B) ve üssünden (E) bahsediyoruz.

Bununla birlikte, gerçek kayan noktalı birimlerde taban 2, 8 veya 16 olabilir (ancak bu donanım içindir). Bir ASN.1 kodlamasında, B'nin değil eri gönderilmelidir. Bu, ilk içeriğin sekizlisinde yapılır. Daha sonra bu sayısal değil erin ve mantisin üssünün değil erine ihtiyacımız var.

2 tabanlı değerler durumunda ilk iç erik sekizlisine bakalım (10 tabanlı değerler için ilk iç erik sekizlisinin 00'dan başladığını ve ardından NR1, NR2 veya NR3 olarak kodlandığını hatırlayın). Bu ilk iç erik sekizli \S ekil III-11'de gösterilmektedir.



İlk bit (bit 8, en önemli), bunu bir temel 2 değer eri olarak tanımlamak için 1'e ayarlanır. Bir sonraki bit (S), sayının işaretidir ve mantis (daha sonra) pozitif bir tamsayı değer eri olarak temsil edilir. Sonraki iki bit (B) tabanı kodlar (2, 8 veya 16, dördüncü değer eri ilede kullanılmak üzere ayrılmıştır). Sonraki iki bit, F adı verilen, 0 ile 3 değer erleri ile sınırlı bir "öç eklenme faktörü" değerini kodlar ve son iki bit, üs kodlamasının uzunluğuunu (LE) kodlar (üs, bunun hemen ardından ikiye tümleyen tamsayı değer eri olarak kodlanır) ilk sekizli. LE'nin dört değer eri, bir sekizli, iki sekizli veya üç sekizli üs için izin verir; dördüncü değer eri, üs alanının bir sekizli uzunluk alıyla başladığını, ardından üs değer eriyle başladığını gösterir. Üs alanının kodlamasının ardından, pozitif bir tamsayı kodlaması olarak mantisi (M) elde ederiz ve olağan ekilde iç erik sekizlilerinin (V kısmı) sonuyla sonlandırılır.

Bu şekilde kodlanan gerçek sayının gerçek değer eri:

$$K \times E \times (2^{**K}) \times (B^{**E})$$

burada ** yukarıda üs alma ve x çarpma anlamına gelir.

Bu, F'nin varlığı i dışında, kayan noktalı sayıları temsil etmenin oldukça tanındık bir yoludur. Ayrıca, 2'nin tümleyeni (hatta 1'in tümleyeni) mantis yerine işaret ve büyülüüğünü kullanımını biraz daha tartışmamız gerekiyor.

1980'lerin başlarında, tek bir bilgisayar üreticisi içinde bile kayan nokta birimlerinin biçiminde çok önemli farklılıklar vardı ve artık kayan nokta gösterimi için yasal standartlar olmasına rağmen, uygulamada hala geniş bir fiili varyasyon var.

Real'in ASN.1 kodlamasında başarılması gereken (ve elde edilen), herhangi bir kayan nokta mimarisinin değerleri kodlamasını veya kodunu çözmeyi (oldukça) kolay ve hızlı hale getiren bir temsildir.

Mantis için işaret ve büyülüük veya ikinin tümleyeni arasındaki seçimi düşünün. Gerçek donanımınız ikinin tümleyeni ise, sayınızı kolayca test edebilir ve S bitini ayarlayabilir, ardından sayınızı olumsuzlayabilirsiniz ve bir işaret ve büyülüük biçiminiz olur. Bununla birlikte, donanımınız işaretliyse ve

büyüklük ve transfer için bir ikiye tümleyen temsili oluş turmanız istenir, görev çok daha zordur. O halde, en yaygın kullanılan makine türü ne olursa olsun, iş aret ve büyülüğun transfer için doğru olduğu açıkltır.

Benzer bir nedenle δ eklenme faktörü F dahil edilmiş tir. Tüm mantisler, kayan nokta dē eri ana belleğ ve boş altıda imha edilen bir ondalık nokta konumuna sahiptir, ancak bu genellikle mantis alanının sonunda dē ildir, yani mantis doğa olarak bir tamsayı dē eri olarak kabul edilmez.

Ancak, ASN.1 kodlamasında aktarmak istediği imiz bir tamsayı dē eridir ve imha edilen ondalık noktanın konumunu kodlamaya çalışmak yerine, bunun yerine imha edilen noktanın bir basamak sağa kaydırılabileceğini kabul ederiz. Üs dē erinin bir eksisi (taban 2 için). Taban 8 ise, üs dē erinin bir fazlası imha edilen ondalık noktayı üç basamak sağa ve taban 16'yi dört basamak kaydırır. Böylece, üste sabit (bu donanım için) bir azalma ile, imha edilen ondalık noktayı mantisin sonuna yaklaşırı. Özellikle, 16 tabanlı bir makine için ucun üç konumu dahilinde. Bir F dē erini kodlayarak (herhangi bir donanım için sabittir), imha edilen ondalık noktayı tam olarak sonunda elde etmek için kalan sıfırdan üç bit'e taşırı. Elbette bir kod çözücü, elde edilen sayıyı 2 üzeri F ile çarpmak zorundadır, ancak bu, bir kayan nokta biriminde yapmak hızlı ve kolaydır.

Bu kodlama 1980'lerin ortalarında gelişti ve inde, bu konular hakkında çok fazla tartışma vardı ve bir dizi satıcı, formatın hepsinin kodlayabileceğini ve kodunu çözebileceğini çok iyi bir "nötr" format sağladığını konusunda anlaşmaya vardı. Bir dizi gerçek kayan nokta donanımından. Tavsiye X.690/ISO 8825 Bölüm 1, hem F'yi dahil etmenin gereklisi hakkında hem de belirli bir kayan nokta birimi için kodlamaları statik olarak belirlemek ve dē erleri kodlamak ve çözmek için gereken algoritmayı biraz ayrıntılı olarak açıklayan önemli bir eğitimi ekine sahiptir. İlgili okuyucu, daha fazla ayrıntı için bu eğitimi yörenlendirilir.

Bir kez daha, kanonik/ayırt edici bir kodlama üretirken, hangi seçeneklere izin verildiği incelemeli ve bunları ortadan kaldırılmalıdır. Ayrıca temsilin "normalleşmesi" ile de ilgilenmemiz gerekiyor. (Bu, 0.4 yerine 4.E-1 gerektirdiği imiz yukarıdaki karakter durumunda gösterilmiş tir. İlk kodlamada da benzer bir endişe ortaya çıkar.) DER/CER/PER (tüm biçimler) için B'nin 2 olmasını isteriz; mantis tek olmalı, F sıfır olmalı ve üs ve mantis mümkün olan en az sayıda sekizli ile kodlanmalıdır. Bu, tüm seçenekleri kaldırırmak için yeterlidir.

3.5.3 Özel gerçek dē erlerin kodlanması

"Taş ma" ve "taş ma" formunun gerçek dē erleri ve pi ve dīer "ilginç" dē erler için özel kodlamalara izin verilmesi hakkında erken tartışmalar vardı, ancak şimdiden kadar standartlaşırı tek özel dē erler (ve şunu anda başka herhangi bir şeyle olmasının pek olası dē il) ARTI-SONSUZ ve EKSİ-SONSUZ'dur.

Son olarak, normal karakter veya kayan nokta biçimleriyle kolayca temsil edilemeyen "özel" gerçek dē erler vardır.

İlk iç erik sekizlisinin ilk (en önemli) bitini kodlayan bir taban 2 için 1 olduğunu ve bir temel 10 kodlaması için ilk iki bitin sıfır olduğunu hatırlayın. Özel bir dē er kodlamasında ilk iki bit sıfır ve bir olarak ayarlanır, ilk (ve yalnızca) iç erik sekizlisinin kalan altı biti dē eri tanımlar (yalnızca iki kodlama kullanılır).

3.6 OCTET STRING değerini kodlama

Daha önce belirtildiği gibi, bir sekizli diziyi kodlamadan iki yolu vardır - ya ilkel bir kodlama olarak ya da dış seviye TLV için belirsiz formu kullanarak gösterdiği imiz bir dizi TLV kodlaması olarak.

Yine oldukça basit - ancak çok uzun bir sekizli diziniz varsa, iletimden önce saymamak için onu parçalamak isteyebilirsiniz. Yine, bir kodlayıcının seçenek i.

Böylece:

sekizli dizi OCTET Dİ Zİ Sİ ::=
'00112233445566778899AABBCCDDEEFF'H

ya olarak kodlar

sekizli dizi: veya olarak	T 04	L 10	V 00112233445566778899AABBCCDDEEFF
sekizli dizi:	24	80	
		04	L 08 0011223344556677
		04	V 8899AABBCCDDEEFF
		0000	

Burada dikkat edilmesi gereken birkaç nokta var. Elbette parçalama, bu kadar kısa bir dizi için çok az anlam ifade eder, ancak biçimini gösterir. Burada iki eşit parçaya ayırmayı seçtiğim, ancak genel olarak herhangi bir noktada parçalayabiliriz. Parçalarımızı parçalamamayı seçtiğim, ama aslında bunu yapmamıza izin veriliyor! DER'de parçalanma yasaktır. CER'de parç'a boyutu 1000 sekizli olarak sabitlenmiş tir (1000 sekizli veya altındaysa parçalanma olmaz) ve parçaların ek parçalanması yasaktır.

Son olarak, eger OCTET Dİ ZGİ Sİ dolaylı olarak etiketlenmiş olsaydı, en dış taki T değerinin (24 - evrensel sınıf 4, yapılandırılmış), değer iş tirme etiketini yansıtacağıını, ancak her parçadaki etiketin 04 (evrensel sınıf 4, ilkel) olarak kalacağıını unutmayın..

3.7 Bir BIT STRING değerinin kodlanması

Bir BIT Dİ ZGİ Sİ değerini, adlandırılmış bitleri listelersek, bit dizisinin baş taki biti ve sonundaki bit hakkında konuşuruz, baş taki bit sıfır olarak numaralandırılır. Öndeği bit, içeriğin sekizlilerinin ilk sekizlisinin en önemli bitine gider. Bu nedenle, daha önce ayrıntılı olarak açıklanan diyagram kurallarını kullanarak, bitler kağıt üzerinde en soldaki iletilir ve en sağ dakine doğrudur.

BER uzunluk sayıları her zaman sekizli cinsindendir. Peki bir bit dizisi kodlamasının tam uzunluğu nasıl belirlenir? Ve bir sekizli sınırlına ulaşmak için hangi bit değerile doldurulmalıdır? (İkincisine cevap - kodlayıcının seçenek i!)

Bir BIT Dİ ZGİ Sİ değerini belirtirken, değer eri gösterimi, gösterimdeki en soldaki biti baş taki bit olarak ilan eder, bu nedenle genel bir tutarlılık vardır, ancak bir BIT Dİ ZGİ Sİ türündeki bitlerin numaralandırılması, bir sekizlideki bitler.

OCTET STRING değerinde olduğum gibi, BIT STRING değerini kodlamaları ilkel olabilir veya parçalara bölünebilir. Ek bir karmaşık iklilik daha vardır - BER'deki uzunluk sayısı her zaman bir sekizli sayısıdır, bu nedenle son sekizlide kaç tane kullanılmayan bit olduğumunu belirlemenin bir yoluna ihtiyacımız var.

Bu, nasıl yapılacağıını söyleyen içeriğin sekizlilerinin başına fazladan bir içeriğin sekizlisi ekleyerek gerçekleşir.

son sekizlide birçok kullanılmayan bit vardır. (CER/DER'de bu kullanılmayan bitlerin sıfırına ayarlanması gereklidir. BER, gönderenin seçenekleri olarak deð erlerine sahiptir.)

Bit dizisinin ayrı TLV'lere parçalanması gerçekleþ tirilirse, parçaların bir sekizli sınırlarında olması gereklidir ve yukarıda açıklanan fazladan sekizli (yalnızca) parçalanmış kodlamadaki son parçaın baş ina yerleþ tirilir.

Böylece:

bit dizisi BIT Dİ ZGİ Sİ ::=
'1111000011110000111101'B

ya olarak kodlar

bit dizisi: veya olarak	T 03	L 0F	V 02F0F0F4
bit dizisi:	23	80	
		T 03 03	L 02 02 V 02F0 02F4
			0000

Yine, bu kadar kısa bir dizi iç in parçalama çok az anlam ifade eder ve yine DER'de parçalama yasaktır. CER'de parça boyutu yine 1000 sekizli olarak sabitlenir (1000 sekizli veya altındaysa parçaalanma olmaz) ve parçaların ek parçaalanması yasaktır.

Kullanılmayan bitlerin sayısını detaylandıran ekstra sekizli dış ina, durum her bakımdan OCTET STRING ile aynıdır.

3.8 Etiketli türlerin kodlama deð erleri

Örtük bir etiket uygulanırsa (YA IMPLICIT kelimesinin kullanılmasıyla ya da otomatik veya örtülü etiketleme ortamında olduğumuz için), o zaman Bölüm II'de açıkladığımız gibi, yeni etiketin sınıfı ve numarası eski etiketinin yerini alır. yukarıdaki tüm kodlamalarda.

**Etiketlemenin son tartışması!
Bu maddenin sonunda net
deð ilse, kitabı nehre atın!**

Bununla birlikte, açık bir etiket uygulanırsa, T kısmı yeni (açık) etiketi kodlayan yapılandırılmış bir kodlamanın içeriği sekizlileri olarak (tek) bir TLV olarak yerleþ tirilmiş eski etiketle orijinal kodlamayı alırız.

Örneð in:

```
tamsayı1 TAM SAYI ::= 72
integer2 [1] ÖRTÜLÜ TAM SAYI ::= 72
integer3 [UYGULAMA 27] AÇIK TAM SAYI ::= 72
```

olarak kodlanır

		L	V
tamsayı1	T 02	01	48
tamsayı2	C1	01	48
tamsayı3	7B	03	
		L	V
		01	48

7B'nin ikili olarak aşağıdaki gibi oluştuğu yer:

Sınıf	P/C	27 numara
UYGULAMA 01	inş a edilmiş 1	11011 = 01111011 = 7B

3.9 CHOICE türlerinin kodlama değerleri

BER'in tüm varyantlarında, seçenekler için ek TL sarmalayıcı yoktur. Kodlama sadece seçilen öğenin kodlamasıdır. Kod çözücü hangisinin kodlandığıını bilir, çünkü bir seçenekdeki tüm alternatiflerin etiketlerinin farklı olması gereklidir.

Bu ya bariz ya da merak uyandırıcı!
CHOICE yapısının kendisiyle ilişkilendirilmiş bir TLV yoktur - TLV'yi seçilen alternatifin değereri için kodlamamanız yeterlidir.

Yani (yukarıda verilen INTEGER ve BOOLEAN türleri için kodlamalarla karşılaştırın)

```

değer er1 SEÇ M
{flag BOOLEAN, değer er
INTEGER} ::= flag:TRUE
ve
değer er2 SEÇ M
{flag BOOLEAN, değer er
INTEGER} ::= değer er:72

```

kodlamaları alıyoruz:

	T	L	V
değer er1	01	01	FF
değer er2	02	01	48

3.10 Kodlama SEQUENCE OF değerleri

Bu oldukça yalındır - bir dış (inş a edilmiş)
Sequence of değerlerindeki her öğe (varsayılarla) bir TLV ile sarıcı olarak TL.

Yani

Bunu zaten TLV yaklaşımının genel tartışması masasından bilmelisiniz.
Burada yeni bir şey yok.

sıcaklık-her-gün TAMSAYI DİZİSİ (7) ::= {21, 15, 5, -2, 5, 10, 5}

şu şekilde kodlanabilir:

sıcaklık-her-gün: T 30	L 80	V	V
			02 01 15
			02 01 0F
			02 01 05
			02 01 FE
			02 01 05
			02 01 10
			02 01 05
		0000	

Tabii ki, dış seviyede belirli uzunluk kodlaması kullanabilirdik, bu durumda kısa biçim kullanılmış olsaydı iki sekizli kurtarabilirdi.

3.11 SET OF değerlerinin kodlanması

Gerçek soyut değerler kümesi nelerdir? {3, 2}, {2, 3} ile aynı değer er mi? Olmalı! Dolayısıyla, bu tek değer er için ayırt edici/kanonik kodlama kurallarında yalnızca bir kodlamamız olmalıdır. Bu, kodlama zamanında önemli bir maliyet üretir. Ayırt edici/kanonik kodlamalara sahip olmak istiyorsanız set-of kullanmamak en iyisidir.

Set-of'un kodlaması, dış T alanının 31 olması dışında, sequence-of ile aynıdır. Ancak, bu bir CER veya DER kodlaması olsaydı, o zaman yedi TLV artan düzende sıralanır ve şunu elde ederiz:

unordered-weeks-tempo SET (7) OF INTEGER ::= {21, 15, 5, -2, 5,
10, 5}

haftasıcılıklar:	L T 31	V 80	V
			T 02 01 FE
			02 01 15
			02 01 10
			02 01 0F
			02 01 05
			02 01 05
		0000	

Sıralamanın her bir öğenin son kodlamalarında olduğu una dikkat edin, bu nedenle -2 sıcaklığı 1 21 sıcaklığı inden önce sıralanır.

3.12 Kodlama SEQUENCE ve SET değerlerini

Bunlar, artık iç TLV'lerin (dizinin veya kümenin her bir öğesi için bir tane) değer işen boyutta olması ve değer işen etiketlere sahip olması dışında tamamen benzerdir. Bazı durumlarda bu elemanların kendileri diziler veya kümeler olabilir, bu nedenle

**Tekrar sadeliğe döndür .
Her derinliğe iç içe
TLV'ler.**

TLV'lerin daha derine yerleş tirilmesi (herhangi bir derinliği e).

İsteğe bağlılığı e erler varsa ve dizinin veya kümenin soyut değerleri bu değerler için bir değer er içermiyorsa, karşılık gelen TLV basitçe atlanır.

SET durumunda BER, iç içe geçmiş TLV'lerin kodlayıcı tarafından seçilen herhangi bir sırada görüntülenmesini sağlar. DER'de değerler, her bir değerinin etiketine göre sıralanır (yine farklı olması gereklidir). Ancak, değer er sahipsek

```
Türüm ::= SET OF {field1
  INTEGER, field2 CHOICE
  { flag BOOLEAN, dummy
  NULL} }
```

o zaman her değer er kümesi bir tamsayıdır eri artı bir boolean veya boş değer er içerir. Ancak, artan etiket sırasına göre sıralamada, bir boolean değer er bir tamsayıdır erinden önce gelir, ancak ondan sonra boş bir değer er gelir. Böylece, alan2'nin hangi değerinin seçildiği işleme bağlı olarak, alan1 değerinden önce veya sonra görüntünebilir! CER'de, alan2'nin herhangi bir değer erinde görünen maksimum etiketin NULL etiketi olduğunu ve gerçekleştirdiğinde hangi değer er gönderilirse gönderilsin, alan 2'nin konumunu belirlediğiini söyleyen biraz daha karmaşık bir algoritma uygulanır. Bunu açıklamak ve belki de anlamak biraz zordur, ancak kodlama zamanında bir sıralama yapmak zorunda kalmaz.

3.13 İSTEĞE BAĞLI ve VARSAYILAN değerlerin sırayla ve set halinde işlenmesi

İSTEĞE BAĞLI'nın neden olduğu hiçbir sorun yoktur (etiketlerin kullanılması, nelerin dahil edildiğiini ve nelerin dahil edilmediğinin netleşmesi). Bununla birlikte, VARSAYILAN durumunda, BER, bir varsayılan değer erin atlanıp atlanmayacağıını (varsayılan değer er olup olmadığı) muhtemelen karmaşık bir şekilde kontrol etmeye imha eder veya yine de kodlayıp kodlamamayı gönderenin seçeneğine olarak bırakır!

TLV yaklaşımı, OPSİ YONEL'in işlenmesini kolaylaşdırır. Tırnak içinde tırmak için tasarlanmışdır. Burada sürpriz yok. DEFAULT'un işlenmesi, kodlayıcının seçenekleriyle ilgili sorunları ve ayırt edici/karşılık kodlamayı kurallarındaki sorunları yeniden gündeme getirir.

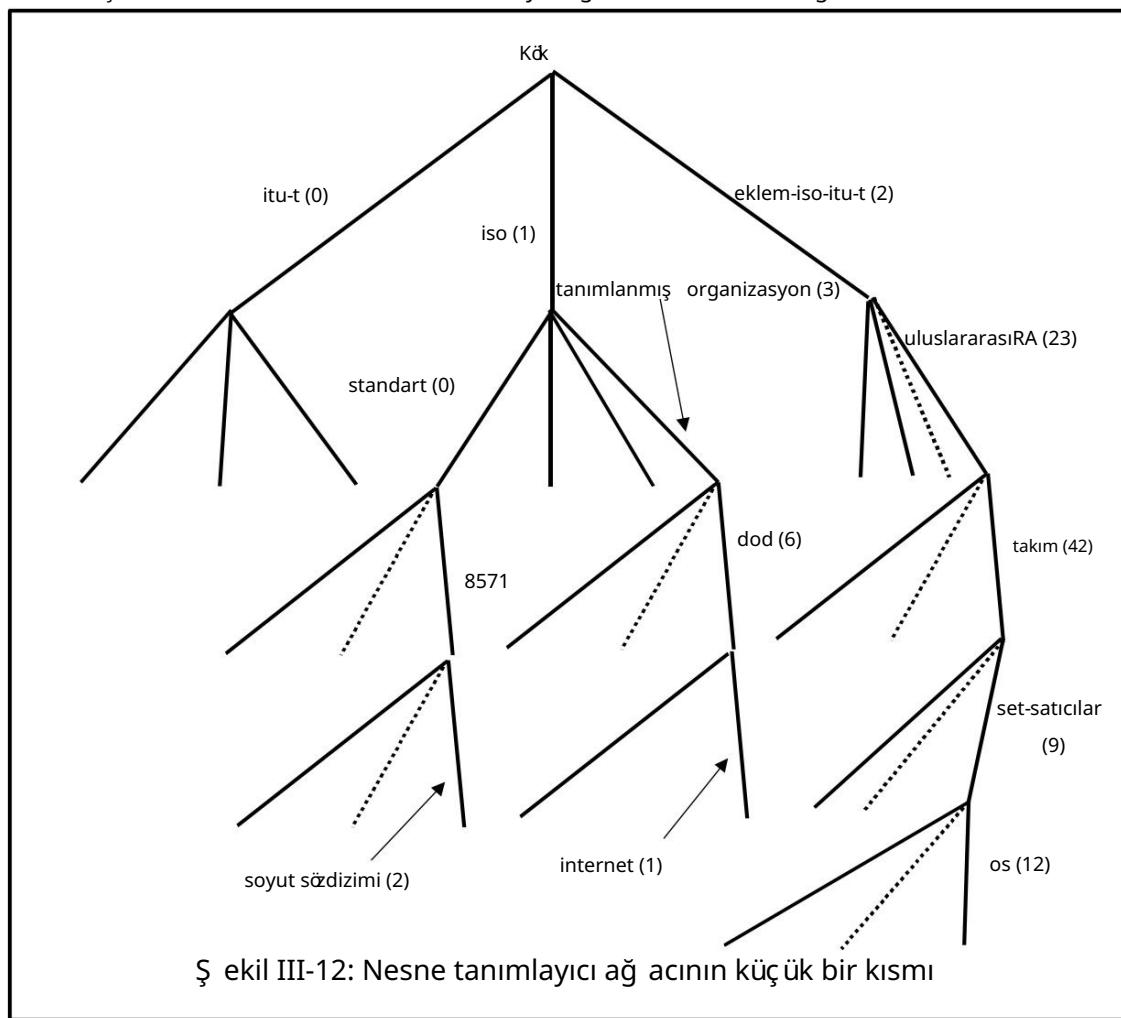
Yine bu, bu kodlayıcının seçenekini kaldırma için DER ve CER sorunlarına neden olur. Bu durumda her ikisi de, kontrol ne kadar karmaşık olursa olsun, değer eri varsayılan değer er eşit olan bir değerinin çıkarılmasını gerektirir. (Bununla birlikte, pratikte VARSAYILAN, normalde yalnızca çok basit türdeki değerler, nadiren karmaşık yapıları diziler ve kümeler olan değerler uygulanır).

Bununla birlikte, bir sonraki bölümde PER'yi daha ayrıntılı olarak tartıştığımızda, PER'in "basit türler" (listelenen i) için zorunlu ihmali ve aksı durumlarda bir gönderenin seçenekini belirttiğimizini görüyoruz; başka vakalar.

3.14 NESNE TANIMLAYICI değerlerinin kodlanması

Değer er temelde bir tamsayı dizisidir, ancak "TAMSAYI DİZİSİ" kullanmaktan daha kompakt bir kodlamaya ihtiyacımız var. "Daha fazla" kavramı burada tekrar devreye giriyor, ancak ilk iki yay için ilginç (ve kötü) bir optimizasyonla.

§ ekil III-12, § ekil II-1'in tekrarıdır ve nesne tanımlayıcı ağ acının bir bđümünü gösterir.



Nesne tanımlayıcı deđ erleri, bu ağ ađta kökten yaprađ a giden yollardır ve böyle bir yol ş u ş ekilde tanımlanır:

{iso(1) standart(0) 8571 soyut söz dizimi(2)}

ancak kodlanan tek bilgi bir deđ erdir.

{1 0 8571 2}

Bu teoride "TAMSAYI Dİ Zİ Sİ" kodlamasıyla gerçek ekleş tirilebilir, ancak her bir tamsayı deđ eri için T ve L alanlarının varlığı bunu oldukça ayrıntılı hale getirir ve farklı (ad hoc) bir kodlama belirtilir.

"Daha fazla bit" kavramı (etiketlerin kodlanmasında da kullanılır – bkz. § ekil III-6, 2.2) kullanılır. Her nesne tanımlayıcı bileş eni için (yukarıdaki 1, 0, 8571 ve 2 deđ erleri), bunu pozitif bir tamsayı deđ eri olarak gerekli minimum bit sayısına kodlarız (standart, yedi bitin minimum katının kullanılmasını gerektirir), ardından bu bitler, her sekizlinin yalnızca en öneşsiz yedi bitini kullanarak sekizlilere bđünür (ilk önce en öneşli sekizli). Son sekizlinin bit 8'i (en öneşli) 0'a ayarlanır, önceki bit 8 deđ erleri ("daha fazla" bit) 1'e ayarlanır.

kodlama sonucu

ftam-oid NESNE TANIMLAYICI ::= {1 0 8571 2}

(onaltılık olarak) olacaktır:

ftam-oid:	T 06	L 05
		V 01 00 C27B 02

Ancak, bu nesne tanımlayıcı değil erinin gerçek kodlaması

T 06	L 04	V 28 C27B 02
------	------	--------------

Nasıl olur?

Kirli bir oyun oynandı! (Ve çoğu kirli numara gibi, daha sonra sorunlara neden oldu).

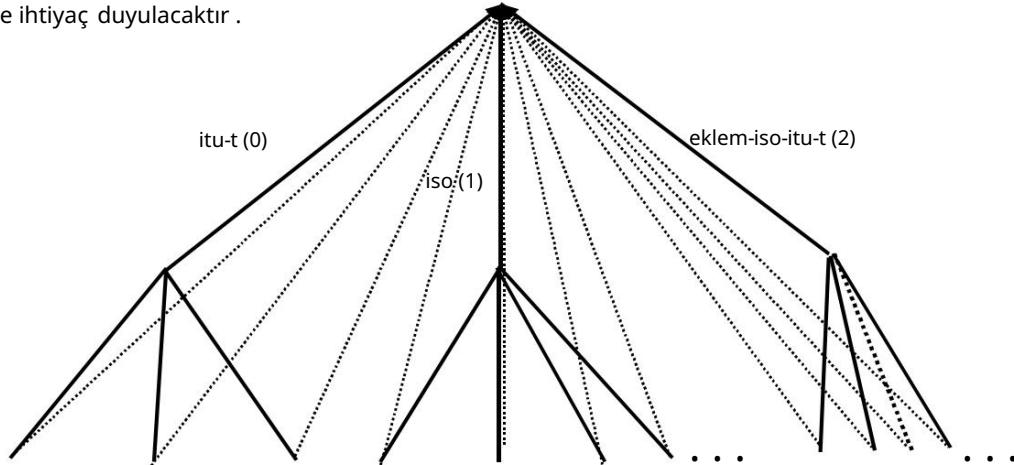
İlk iki yayı kodlayan sekizlerin (1986'da) büyük değil erlere sahip olma ihtimalinin düşüğü olduğunu ve bu iki yay için iki sekizli kullanmanın "kötü bir şeyle" olduğunu düşündürüyor. Böylece bir "optimizasyon" (zorunlu) getirildi.

Şekil III-12'deki en üstteki iki yayı alabilir ve bunları Şekil III-13'te gösterilen noktalı yollarla "üst üste bindirerek" kırıktan her bir ikinci düzey düğümü tek (sahte) bir yay üretemeli. Bu sahte yollar nasıl numaralandırılır?

Pekala, üç üst düzey yay vardır ve yukarıda açıklanan "daha fazla bit" konseptiyle tek bir sekizlide 128 yay (0 ile 127) için kodlamaları barındırılabilir. 128'ü 3 eşittir 40!

İlk iki üst düzey yayın hiç bir zaman 40'tan fazla alt-yayı olmayacağıını varsayılmı ve ilk 40 sözde-yayı üst düzey 0'a, sonraki 40'ı üst düzey 1'e ve kalanı üst-seviye ark 2.

Buradaki varsayımda, üst seviye 0 (ccitt - 1986'da olduğu gibi) ve 1'in (iso) altında asla 40'tan fazla yay olmayacağı, ancak üst seviye 2'nin (joint-iso-ccitt) altında daha fazla olabileceğidir. Bu durumda sözde yay için çok sekizli bir gösterime ihtiyaç duyulacaktır.



Şekil III-13: En üstteki iki yayın tek bir yay haline getirilmesi

Bu nedenle, üst düzey yay 0'in altındaki herhangi bir ikinci düzey yay için, söze yay için ikinci düzey yay sayısını kullanırız. Üst düzey yay 1'in altındaki herhangi bir ikinci düzey yay için, ikinci düzey yay numarası artı 40'i sahte yay için sayı olarak kullanırız ve üst düzey yay 2'nin altındaki herhangi bir ikinci düzey yay için ikinci düzeyi kullanırız. yay numarası artı söze yay için sayı olarak 80.

Daha sonra {1 0 8571 2} kodlamasını ş uş ekilde alırız:

	L	V
T 06	04	28 C27B 02

daha önce açıldı gibi.

Daha önce belirtildiği gibi, nesne tanımlayıcı ağ arasında "asılı" olduğu unuz yer ömensizdir, ancak nesne tanımlayıcılarınız siz aş ağ işaretinde daha uzun olacaktır. 1995'in ortalarında, diğer büyük uluslararası oyuncuların üst düzey yayar istemesiyle bu bir sorun olarak su yüzüne çıktı. En üstteki iki yay ile yukarıdaki "geçiş tirme", yeni üst düzey yayar eklemeyi zorlaştırır (imkansız değil ama zorlaştırır) ve bu sorunu hafifletmek için ASN.1'e eklenmek üzere RELATIVE OID yapıcısı önerildi.

Bir kuruluşun aşağındaki gibi bir kırık altında nesne tanımlayıcıları tahsis etme ihtiyacı varsa:

{joint-iso-itu-t(2) uluslararası RA(2) seti(42)}

ve (her zaman veya yaygın olarak) nesne tanımlayıcı değeri erlerini bu kırık altında taşımak için özel olarak tasarlanmış bir protokole sahipse, o zaman tanımlayabilir

SET-OID'ler ::= BAĞL OID
-- Göreli olarak{2 2 42}

ve bu türü protokolünde tek başına veya bunun SEÇİMİ ve normal bir NESNE TANIMLAYICISI olarak kullanın.

Göreceli bir nesne tanımlayıcı türü, yalnızca bilinen bir düzüm (bu durumda {2 2 42}) altında asılı olan nesne tanımlayıcı değeri erlerini taşıyabilir, ancak değeri kodlanması, yalnızca {2 2 42} sonrasında nesne tanımlayıcı bileşenlerini kodlar, kaydeder bu durumda iki sekizli.

Tasarruf, kodlamaların zaten genellikle daha küçük olduğu PER'de daha önemli olabilir. Güvenli Elektronik İşlemeler (SET) söz konusu olduğuunda, sertifikaların ASN.1 kodlamalarını bir akıllı karta kolayca sıkışacak bir boyuta indirmek bazı zorluklar doğurdu ve PER ile ilgili nesne tanımlayıcı teknikinin kullanılması önemliydi.

Baskıya girdiği sırada, RELATIVE OID çalışması tamamlanmamıştı, bu yüzden detayları en son standartla kontrol edin! (Ve/veya Ek 5'teki Web sitesinde bu kitap için yazım hatası sayfalarına bakın).

3.15 Karakter dizisi değeri erlerinin kodlanması

Karakter dizisi türleri (aşağıda açıklanan zaman türlerinde olduğu gibi) diğer standartlara göre kodlanmışdır. Bu karakter seti standartlarının daha ayrıntılı bir açıklaması, Bölüm IV'te yer almaktadır, ancak her bir kodlamadan temel özellikleri burada açıklanmaktadır.

Burasi, dışarı çıkip ek özellikler satın almanız gereken yerdir - neredeyse tüm karakter dizisi kodlamaları, diğer özelliklere atıfta bulunur.

Bu kitapta muhtemelen ASN.1 Standardının kendisinden daha fazla metin vardır!

En basit karakter dizisi türlerinden baş *layarak* - NumericString, PrintableString, VisibleString ve GraphicString - bunların içeriği sekizlileri yalnızca karakterlerin ASCII kodlamasıdır.

Bir sonraki grup TeletexString, VideotexString, GraphicString ve GeneralString'dir. Bunlar, yapısı ISO 2022'de belirtilen kodlamalara sahiptir ve bu kayıt girişini "belirlemek ve çağ ırmak" için Uluslararası Kayıtta her Kayıt Giriş'i için belirtilen "kaçış" dizilerini kullanır. Uygun kaçış dizisinden sonra, sonraki sekiz bitlik kodlamalar, bir sonraki kaçış dizisi gerçeğe ekleşene kadar o yazmaç girişindeki karakterlere referans verir. Birden fazla kayıt girişinde görünen birçok karakter olduğuna dikkat etmek önemlidir, bu nedenle belirli bir karakter dizisi için sıkılıkla birçok kodlama vardır. Araya karakter kodlaması olmadan, her biri bir öncekini aşan bir dizi kaçış dizisine sahip olmak teorik olarak da mümkündür. Ayırt edici/kanonik kodlama kurallarında tüm bu seçenekler elenir.

Dikkate alınacak sonraki iki karakter seti tipi UniversalString ve BMPString'dir. UniversalString, ISO 10646'nın tüm karakterlerini destekler (kodlamada karakter başına 32 bit kullanan en yeni karakter kodu standarı). BMPString yalnızca "Temel Çok Dilli Düzlemleri"deki (tüm normal dünyevi faaliyetler için yeterlidir!) karakterleri destekler; karakter başına 16 bit kullanan "Unicode" karakter seti.

Son olarak, UTF8String karakter başına değil işken sayıda sekizli kullanır (ASCII karakterleri için birden maksimum altı sekizliye kadar). Bir UTF8String kodlamasındaki sekizlilerin hiçbiri, bir ASCII karakterinin (tek sekizli) kodlaması olmadıkça, üst bit sıfır ayarlanmaz. Tek bir karakter oluşuran sekizlilerin kodlaması, bir karakterin kodlamasının ilk sekizli olduğu her zaman "10" ile başlar, bu nedenle bir kodlamanın ortasında rastgele bir noktadan başlasanız bile başlangıcı kolayca belirleyebilirsiniz. sonraki karakter kodlaması.

Bir karakterin UTF8 kodlaması, ya bir "0" bit ile başlayan (bu durumda tek bir sekizli ASCII kodlamasına sahibiz) veya iki ila altı bir bit ve ardından bir sıfır bit ile başlayan bir "ilk sekizliye" sahiptir. Bu ilk sekizlide kalan bitler, karakteri tanımlamak için kullanılabilir. Bir bitin sayısı, karakteri kodlamak için kullanılan sekizli sayısını verir. Sonraki her sekizlinin en üstteki iki biti "10" olarak ayarlanır ve kalan altı bit, karakteri tanımlamak için kullanılabilir. Karakter, gereklili minimum sekizli sayısını kullanılarak mevcut bitlere kodlanan (sağ'a yaslanmış) ISO 10646 32 bit kodlama şemasındaki numarasıyla tanımlanır. Böylece, 11 üssü ikiden küçük değil erlere sahip karakterler (tümü "Avrupa" karakterleridir) iki sekizli olarak kodlanır ve 16 üssü ikiden küçük değil erlere sahip karakterler üç karakter olarak kodlanır ve bu böyle devam eder.

Karakterlerin UTF8 kodlamalarına ilişkin bazı örnekler, **Şekil III-14**'te onaltılık gösterimler olarak verilmişdir.

karakterin adı	Unicode/10646 sayı İkili olarak kodlama
LATİN BÜYÜK H HARFI	72 01001000 48 00110000
LATİN RAKAM SIFIR	
CEDILLA İLE LATİN BÜYÜK C HARFI	199 11000011 10000111 914 11001110
YUNAN BÜYÜK HARFLİ BETA	10010010 1053 11010000 10011101 1664
KİRİLBÜYÜK HARF EN	11011010 10000000
ARAP HARFI BEHEH	
KATAKANA MEKTUP KA	12459 11100001 10100001 10101011

Şekil III-14: Bazı UTF8 Kodlama Örnekleri

3.16 Zaman türlerinin kodlama dē̄ erleri

Zaman türleri, karakter dizileri olarak belirtilir ve kodlamaları, bu karakterlerin ASCII kodlamasıdır.

GeneralizedTime'ın kesinliği ile ilgili sorunlar vardı. Asıl baş vurulan standart, 1975'teki ilk baskısından itibaren saniyelerin herhangi bir sayıda ondalık basamağı sahip olmasına izin veren ISO 3307'dir. Ancak bir şekilde ASN.1 uygulayıcı topluluğu unun bazı bromiumları, kesinlik in milisaniyelerle sınırlı olduğ u ve daha yüksek bir kesinlik için dē̄ erleri kabul etmeyeceğ i izlenimine kapılmış ti.

Basitçe karakterlerin ASCII kodlaması. Ancak ayırt edici/kanonik kurallardaki kesinlik sorunlarına dikkat edin.

Kesin soyut dē̄ erler kümesinin ne olduğ u ile ilgili sorunlar da vardır. ASN.1 spesifikasyonu, GeneralizedTime'ın zamanların çeş itli hassasiyetlerde temsiline izin verdiği ini belirtir. Yani, örneğ in, bir zaman:

"199205201221.00Z"

ile aynı soyut dē̄ er

"199205201221.0Z"

Öyleyse, kanonik ve ayırt edici kodlama kuralları, kodlamalardan birini veya dī̄ erini (hatta her ikisini de!) yasaklamalıdır. Ancak, farklı kesinliklerin farklı soyut dē̄ erler olduğ u (ve farklı anlamlar taşıyabileceğ i) kabul edilirse, bu tür kodlamaların tümüne kanonik ve ayırt edici kodlama kurallarında izin verilmesi gereklidir.

Nihai karar, sondaki sıfırların dahil edilmesiyle ima edilen kesinlik in soyut dē̄ erin birincil bir parçası olmadığı ve ayırt edici ve kanonik kodlama kurallarında sondaki sıfırların yasaklanması gerektiği iyi - bir zımnı kesinlik in yüzde biri kadar bir süre saniye, saniyenin onda birinin zımnı kesinlik iyle birden aynı zamandır (soyut dē̄ er) ve farklı anlamlar taşımamalı ve ayırt edici ve kanonik kodlama kurallarında aynı kodlamaya sahip olmalıdır.

Daha karmaş ik yapılar için 4 kodlama

4.1 Açık tipler

ASN.1, baş langıçından beri "delikler" kavramına sahiptir, baş langıçta "ANY" adlı bir tür olarak tanımlanmış tir ve daha sonra söz dizimi ile belirtilen sözde "açık tür" olarak tanımlanmış tir:

Daha karmaş ik türlerin çoğu u, ASN.1 SEQUENCE türleri olarak tanımlanır ve dē̄ erleri, bu dizi türlerinin dē̄ erlerinin kodlanmasıyla kodlanır.

OPERATÖR.&Tipi

bu alanı dolduracak tipin, OPERATOR sınıfının bir bilgi nesnesinin &Type alanına atanan bazı ASN.1 tipinin dē̄ eri olduğ unu belirtir (bkz. Bölüm 6).

BER, açık türleri çok basit bir şekilde işler: Bu alanı eninde sonunda dolduran şeyin bir ASN.1 türü olması gereklidir ve alanın kodlaması, basitçe o türdeki bir dē̄ erin kodlanmasıdır.

BER'de katı bir TLV yapısı olduğuunu unutmayın, bu nedenle, kodlanan gerçek tür hakkında herhangi bir bilgi olmadan bir BER TLV kodlamasının sonunu bulmak her zaman mümkün değildir. Açık bir tip söz konusu olduğuunda, bu tipin tanımlanması, tipin bir de¤ erinin kodlanmasıından daha sonra kodlamada görünebilir. TLV yapısı türden ba¤ ımsız olduğu u¤ in bu BER'de sorun yaratmaz.

4.2 Gömülü pdv tipi ve harici tip

Bdüm II'de açıklan¤ ı gibi, bunlar ASN.1 türleri iç in biraz belirsiz adlardır, ancak "gömülü", burada bir ASN.1 türüne gömülü yabancı (ASN.1 tanımlı olmayan) malzememiz olduğu u ve "harici" anlamına gelir. "aş a¤ ı yukarı aynı ş ey anlamına gelir - ASN.1'in dış ındaki malzeme gömülüyor.

Tarihsel olarak, HARİ Cİ öncে geldi ve 1994'te EMBEDDED PDV biraz daha fazla iş ıevsellikle eklendi (yeni spesifikasyonlar her zaman EXTERNAL de¤ il, EMBEDDED PDV kullanmalıdır).

Bu türlerin her ikisi de dizi türleri olan ve türün tüm anımlarını ta¤ iyabilecek alanlara sahip "iliş kili türler" sahiptir. Genel olarak, bu, bazı materyallerin kodlanması (en genel durumda bir bit dizisi olarak ta¤ ınır) ve özetin tanımlanması (EXTERNAL durumunda bir nesne tanımlayıcısı ve EMBEDDED PDV durumunda sıfırdan ikiye kadar) ve transfer sözdizimidir. bit dizisindeki kodlama. (Kodlamalar bir OSI Sunum Katmanı protokolü üzerinden aktarıldığında uygulanan seçeneklerin dahil edilmesiyle biraz daha karma¤ iklık vardır, ancak bu, OSI olmayan durumda kodlamayı etkilemez.) BER kodlaması basitçe kodlama olarak tanımlanır. bu "iliş kili türler" arasında.

4.3 ÖRNEK türü

INSTANCE OF türü, özellikle "deli¤ imize" koymak istediği imiz ş eyin, de¤ eri (ASN.1) olan türü tanımlamak için (tek) bir nesne tanımlayıcısı olduğu u durum iç in tasarlanmış , HARİ Cİ veya GÖMÜLÜ PDV'nin çok basitleş tirilmiş bir sürümünü sağ lar. "deli¤ e" kodlanır ve ardından ASN.1 türünden bir de¤ er gelir. Bu tür, bdüm II'de açıklanan yerle¤ ik çok basit bilgi nesnesi sınıfı TYPE-TANIMLAYICI ile ilgilidir.

Yalnızca iki alanla bir Dİ Zİ türü olarak kodlanmış tır - bir nesne tanımlayıcısı ve bir ASN.1 türünün de¤ eri (açık bir tür olarak).

4.4 KARAKTER Dİ Zİ Sİ türü

CHARACTER STRING türü 1994'te tanıtıldı ve kodlaması EMBEDDED PDV ile neredeyse aynı. Buradaki fikir, bir karakter dizisinin de¤ erine sahip olmamızdır (bir karakter soyut sözdizimi nesne tanımlayıcısı tarafından tanımlanan bazı repertuarlardan), bir karakter aktarımı sözdizimi nesne tanımlayıcısına göre kodlanmış tır. Bu nedenle, temelde sıfır ila iki nesne tanımlayıcı iç en bir dizinin kodlamasına sahibiz (GÖMÜLMÜŞ PDV'de olduğu u gibi, nesne tanımlayıcılarından birinin veya her ikisinin protokol belirtimi tarafından belirlenen sabit de¤ erler alındı¤ ı ve bu nedenle kodlanması gerekmeyen seçenekler vardır), ardından dizideki gerçek karakterlerin kodlanmasıyla.

5. Sonuç

BER'in ASN.1 spesifikasyonu sadece 17 sayfadır - bu bđümden daha az! (DER ve CER Ekleri ve detayları dikkate alınmadan).
İlgili okuyucu artık bu özelliğ i anlamakta hiç bir sorun yaş amamalıdır.
Git ve oku!

Artık BER hakkında bilinmesi gereken her şeyi biliyorsunuz - sadece gidin ve teknik özelliklerini okuyun ve görün!

Bölüm 3

Paketlenmiş Kodlama Kuralları

(Veya: Gelecek milenyum için kodlamalar -
şimdilik alabileceğiniz kadar iyi!)

Özet: Bu bölüm, Paketlenmiş Kodlama Kurallarının ayrıntılarını sağlar. Genel olarak iki ana bölüm vardır. İlk bölümde, PER'in bazı genel özellikleri ve asıl spesifikasyonda kullanılan terminoloji hakkında daha fazla ayrıntı verilmektedir. Bu ilk bölümde şunları ele alıyoruz:

PER kodlamasının genel yapısı ve kullanılan terminoloji (giriş, uzunluk belirleyici, iç erik), PER'in dört varyantının tartışıılması.

Geniş letilebilir türler için kodlamaların genel doğası.

PER-görünür kısıtlamalar.

Etkili boyut ve alfabe kısıtlamaları.

Etiketlerin kanonik düzeni ve bu sıralamanın kullanımı.

Gerektiğiinde genel uzunluk alanının biçimi.

İSTEĞE BAĞLI bit haritası ve SEÇİM dizini (geniş letilebilir ve geniş letilemez seçenekler için)

İkinci bölüm, önceki bölümde BER için yapıldığı gibi, her ASN.1 türünün kodlamalarının ayrıntılarını verir. Sıra, daha basitten biraz daha karmaşık ikinci kodlamalara doğru hareket edecek şekilde tekrar seçilir. Şunların kodlamalarını ele alıyoruz:

NULL ve BOOLEAN değerleri.

INTEGER değerleri.

SAYILANDIRILMIŞ değerler.

Dizelerin uzunluk belirleyicileri.

Karakter dizisi değerleri.

SEQUENCE ve SET kodlaması.

SEQUENCE OF ve SET OF kodlaması.

GERÇEK ve NESNE TANIMLAYICISI kodlaması.

Kalan türlerin kodlanması (GeneralizedTime, UTCTime, ObjectDescriptor ve "ValueSet" gösterimi kullanılarak tanımlanan türler).

Bu sonraki konuların çoğu u, bu bölümün ilk kısmında tam olarak kapsanan genel yaklaşımları takip ettileri için, sadece örnekler vererek ele alınmaktadır.

1. Giriş

PER kodlamalarının altında yatan ilkeler (etiketlerin kodlanması, OPTIONAL için bir bit haritasının kullanılması, bir CHOICE indeksinin kullanılması ve SET ögelerinin ve CHOICE alternatiflerinin etiket sırasına göre sıralanması bu bölümün 1. Bölümünde zaten tanıtılmıştır. Bu bölümde detayları tamamlıyoruz.

PER kodlamalarının ilkelerini zaten öğrendiniz, sadece ayrıntılara geçelim.

Bu bölümün sonraki kısmı, tüm kodlamaların örneklerini sağlar ve gerektiğiinde biraz daha fazla açıklama verir.

Bu bölüm tamamen bağımsız değildir. Okuyucunun bu bölümde başlamadan önce Bölüm III, Bölüm 1'in ilgili kısımlarını okumuş olacağın varsayılmaktadır, ancak PER kodlamalarının BER (veya daha genel olarak CER/DER) kodlamalarıyla aynı olduğu bazı durumlar da vardır. ve bu tür durumlarda Bölüm III, Bölüm 2'ye atıfta bulunulur.

BER için kullanılan bit numaralandırma ve şema kuralı (solda gösterilen kodlamanın ilk sekizlisi, en önemli olarak 8 ile numaralandırılmış ve solda gösterilen bitler) burada da kullanılır.

Bununla birlikte, PER ile bazen bazı alanların başında sekizli hizalama oluşturmak için eklenen dolgu bitleri vardır. Doldurma bitlerinin eklenmesi gerekebileceği durumlarda (bir sekizli içindeki mevcut bit konumuna bağlı olarak, sıfır ile yedi doldurma biti olabilir), burada verilen örneklerde alanın başında langırcık bir "P" kullanılır. bölüm.

2 PER kodlamasının yapısı

2.1 Genel biçim

PER'in mutlaka sekiz bitin katı olan alanlara kodlama yapmadığıını zaten biliyor musunuz, ancak (örneğin) SEQUENCE'in kodlamalarına ilişkin BER kavramı, her bir ögemenin tam kodlamalarının ardından gelen bazı başlıklar için de geçerlidir. BAŞ İNA.

Kulag a biraz BER gibi geliyor - önsöz, uzunluk belirleyici, iç indekler. Ama yanıltılma. Çok u zaman sahip olduğu umuz tek şeyle içeriğ!

PER durumunda, "başlık" giriş olarak adlandırılır, ancak SEQUENCE için yalnızca isteği bağlılığı ögeler varsa mevcuttur, aksi takdirde boş tur ve her ögemenin kodlamasına sahibiz.

Bir kodlamanın "L" kısmında da BER'den bir fark vardır. Bir kez daha, sık sık eksik olabilir (aslında uzunluk önceden bilindiğinde), ancak terminoloji de şunu şeyle değil işir:

"uzunluk belirleyici". Bu dē iş īlik yapıldı çünkü BER'in uzunluk sekizlileri her zaman bir sekizli sayısyıken (belirsiz biçim dışında), PER'de uzunluk belirleyicisi ş u olabilecek bir dē eri kodlar:

bir sekizli sayısı (BER'de olduğu gibi); veya

bir bit sayısı (kısıtlanmamış bir BIT Dİ ZGİ Sİ dē erinin uzunluğu u için kullanılır); veya

yineleme sayısı (SEQUENCE OF veya SET OF dē erinin uzunluğu unu belirlemek için kullanılır).

Ayrıca, PER'de uzunluk determinantının sekiz bitin tam bir katı olması gerekmelī i de bir durumdur.

Bir uzunluk determinantının kesin biçimi ve kodlaması daha sonra açıklanacaktır.

Üç kodlama parçasının her biri, bir bit alanı olarak adlandırılan ş eye kodlanır. Bu bit alanının uzunluğu, tip tanımından statik olarak belirlenebilir veya kodlamanın bu kısmından önce bir uzunluk belirleyici kodlama gelir. "Bit alanı" terimi, alanın zorunlu olarak sekiz bitin tam bir katı olmadığı inî veya genel olarak alanın bir sekizli sınırla başlaması gerekmelī inî ima etmek için kullanılır.

Büyük ve karmaş īk yapılı bir türdeki bir dē erin kodlanması ilerlerken, bir dizi bit alanı üretiyoruz. Kodlamanın sonunda, dē erin tam kodlamasını üretmek için bunlar basitçe uçtan uca (sırayla) yerleş tirilir ve sekizli sınırlar göz ardı edilir.

2.2 Kısmi sekizli hizalama ve PER varyantları

Genel yapıda birkaç karış īlik daha var, bunlardan ilki!

PER tasarımcılarının, alanın sekizli bir sınırla başlamasını sağlamak daha mantıklı olacağ inî düşündükleri bazı alanlar vardır (uygulamanın basitliği i ve CPU döngülerinin en aza indirilmesi için). Bunun geçerli olduğu alanlar, tip tanımından belirlenebilir (ve iletilen belirli dē ere bağlı dē ildir).

**Kurslar için atlar! PER'İN
ALIGNED varyantı, alanlar
için sekizli bir sınırla ne
zaman geçileceğ i
konusunda pragmatik
kararlar verir. UNALIGNED
versiyonu, "önēmli olan bant
genişligi" diyenler içindir!**

Bu tür durumların sekizli hizalanmış bit alanlarına kodlandığı söylenir. Bit alanlarının son birleş tirilmesinde, dış düzey tipin - mesajın tüm kodlamasının başlangıcından itibaren sekiz bitin katlarında başlamalarını sağlamak için sekizli hizalanmış bit alanlarından önce gerekīinde dolgu bitleri eklenir. veya "protokol veri birimi" (PDU).

Dolgu bitlerinin istenmedī i bazı uygulamalar (hava trafik kontrolü bunlardan biridir) vardır - bant genişliği inin en aza indirilmesi birincil ihtiyaç olarak kabul edilir. Bu nedenle resmi olarak PER'nin iki çeşidi vardır:

ALIGNED varyantı (doldurma bitleriyle); ve

UNALIGNED varyantı (doldurma bitleri olmadan ve daha sonra açıklanacak olan dīg er bazı bant genişliği azaltma özellikleriyle birlikte).

2.3 Kanonik kodlamalar

BASIC-PER büyük ölçüde kanoniktir, ancak %100 kanonik olmanın "pahalı" olduğunu bazı türler vardır (SET OF, bazı karakter dizisi türleri, zaman türleri ve bazı DEFAULT oluşumları). Dolayısıyla **BASIC-PER** (pragmatik olmak!) bu türler için kanonik olmayan kodlamalara sahiptir. **CANONICAL-PER** tamamen kanoniktir.

Bu, genel PER ailesi içinde daha fazla kodlama kuralına yol açan başlangıç bir alandır.

BER, CER ve DER için spesifikasyonların üretilmesine yol açan birçok kodlayıcı seçeneklerin sahipken, PER'in temel kodlamadaki seçeneklerden kaçındığını ve ilk bakışta kanonik gibi görünen üne dikkat edin. (Kesinlikle BER'den çok daha kanoniktir!)

Bununla birlikte, gerçek ekten kanonik kodlamalar üretmek için (BER'de olduğu gibi) bir SET OF örneği es gereklidir ve GeneralString ve GraphicString gibi karakter dizisi türlerini kodlamaya karmaşıklıklar katar. **BASIC-PER** olarak adlandırılan (hem aligned hem de unaligned değil işkenleriyle) bunu yapmaz ve YALNIZCA bu türler dahil değil ilse kanonik kodlamalar üretir. **CANONICAL-PER** (aligned ve unaligned varyantı ile) tamamen kanoniktir ve SET-OF sıralamasını ve GeneralString vb. CER'e girin.

2.4 Dış düzey tam kodlama

Tam bir kodlamanın dış düzeyinde (satır boyunca gönderilen toplam mesaj) başlangıç bir küçük karmaşık iklik ortaya çıkar. (Bu oldukça ayrıntılı bir noktadır ve kodlama üretimiyle yoğun bir şekilde ilgilenmediğiniz sürece bir sonraki maddeye geçebilirsiniz).

Elle kodlayıcı değilseniz, bunu atlayın. "Dış seviye" tiplerinin - tam mesajların - tüm PER kodlamalarının sekiz bitin katları ve en az bir sekizli uzunluğununda olduğuunu söylemek yeterlidir.

Bir mesajın PER ile sıfır bit olarak kodlanabileceğini birkaç teorik durum vardır. Bu, örneğin, dış düzey bir NULL türüyle veya sıfır yinelemeye sahip olacak şekilde kısıtlanmış bir SET OF ile gerçek eklesimi (her ikisinin de pratikte gerçekleşmesi pek olası değil, ancak ...!).

Buradaki sorun, eğer bir taş ıycı protokolünün kullanım şartları, taş ıycıya bu türden birden fazla değil erin yerlesmesine izin veriyorsa, sıfır bitin katı hala sıfır bittir ve alıcı, hatta kaç değil erin gönderildiğiini bilmeyecektir. tip tanımının tam bilgisi ile!

Bu nedenle PER, dış düzey türün tam kodlaması sıfır bit ise (bu, dış düzey türün yalnızca bir soyut değil erin anlamına gelir), bunun yerine bu kodlama için tek bir bitin kullanılmasını gerektirir.

Ve son olarak, taş ıycı protokollerinin genellikle yalnızca sekiz bitin katlarını içerebilen "demetler" sağladığını kabul ederek, PER, sekiz bitin tam bir katını üretmek için tam kodlamanın her zaman sonunda sıfır bit ile doldurulması gereklidirini belirtir. (Yine, bu, taş ıycının PER kodlamasını kodlayıcıdan kod çözücüye iletmek için kullandığı sekizli kovaya kodlanmış olan değil erlerin sayısı hakkında kod çözme tarafında hiçbir şüphe olmamasını sağlama konusunda ındirir).

Bu nedenle, eksiksiz bir dış düzey PER kodlamasının minimum boyutu bir sekizli ve her zaman sekiz bitin katıdır, ancak tek tek bileşen parçaları genellikle sekiz bitin katı değil ve sıfır bit olabilir.

3 Geniş letilebilir türlerin kodlama değer erleri

PER, geniş letilebilirlik için tek tip bir yaklaşım sahiptir. Geniş letilebilir INTEGER ve dizi değer erlerinin kodlamasının bir gösterimi için aşağıdaki § ekil III-15'e, geniş letilebilir SET ve SEQUENCE değer erlerinin kodlamasının bir gösterimi için § ekil III-16'ya, § ekil III-17'nin bir gösterimi için bakın. geniş letilebilir CHOICE değer erlerinin kodlanması ve geniş letilebilir SAYILANDIRILMIŞ değer erlerin kodlanması bir gösterimi için § ekil III-18'e bakın.

PER tutarlıdır: Geniş letilebilir bir türün önünde, kodlanan değer erin kökte olup olmadığıINI söyleyen bir bit vardır. Eğer er öyleyse, "normal olarak kodlar. Aksi takdirde, özel bir kodlama vardır.

Herhangi biri:	0
bunu takiben:	Geniş letilebilirlik işaretçisi veya uzantıları olmayan tür için olanla aynı olan türün bir değer erinin kodlaması.
Veya:	1
bunu takiben:	Sınırlanırmamış türdeki değer erlerle aynı olan, kırık dışında kalan geniş letilebilir türdeki bir değer eri için kodlama.
§ ekil III-15: Geniş letilebilir kısıtlı INTEGER veya dizi kodlamaları	

Herhangi biri:	0
bunu takiben:	Geniş letilebilirlik işaretçisi veya uzantıları olmayan tür için olanla aynı olan türün bir değer erinin kodlaması.
Veya:	1
bunu takiben:	0'daki gibi bir kodlama, ancak uzantılar için özel bir kodlamayla, ekleme noktasına eklenir.
§ ekil III-16: Geniş letilebilir SET veya SEQUENCE tipi kodlamalar	

Herhangi bir tür (kısıtlanmış bir TAM SAYI, kısıtlanmış bir dize, bir Dİ_Zİ, bir SET, bir SEÇİ_M veya bir Tip tanımında veya PER görünür kısıtlamasında bir geniş letilebilirlik iş aretiçisine (üç nokta) sahip SAYILANDIRILMIŞ), aşağıdaki gibi kodlanmış bu türde bir dē er sahiptir:

Herhangi biri:	0
bunu takiben:	Bir geniş letilebilirlik iş aretiçisi olmayan tip iç in olanla aynı olan seçim indeksinin (kdkte bulunan bir alternatif tanımlayan) kodlaması.
bunu takiben:	Kdk içinde seçilen alternatifin bir dē erinin kodlanması.
Veya:	1
bunu takiben:	Seçim indeksi iç in farklı bir kodlama (kdk dış inda bir alternatif belirleme).
bunu takiben:	Seçilen alternatifin kdk dış indaki bir dē erinin kodlanması.
Şekil III-17: Geniş letilebilir CHOICE kodlamaları	

Önde kodlanmış bir bitlik bir bit alanı vardır - uzantılar biti.

Kodlanan dē er kdkteyse (orijinal INTEGER veya SAYILANDIRILMIŞ dē erlerden biri veya varsa tüm uzantı eklemelerinin olmadığı ı bir SET veya SEQUENCE dē eri) uzantı biti sıfıra ayarlanır.

Aksi takdirde, uzantı biti bire ayarlanır (kdkün dış indaki dē erler).

NOT — Yalnızca 1'den büyük sürümlerin uygulamaları biti bir olarak ayarlar, ancak tüm uygulamalar bir kdk dē eri kodlayabilir ve dolayısıyla uzantı bitini sıfıra ayarlayabilir.

Herhangi biri:	0
bunu takiben:	Geniş letilebilirlik iş aretiçisi veya uzantıları olmayan tür iç in olanla aynı olan türün bir dē erinin kodlaması.
Veya:	1
bunu takiben:	Kdkün dış inda olan geniş letilebilir türde bir dē er iç in kodlama.
Şekil III-18: Geniş letilebilir ENUMERATED tipi kodlamalar	

"Uzantılar biti" sıfıra ayarlanırsa, uzanti iş aretçisi (ve tüm uzantılar) yokmuş gibi tam olarak aynı kodlama (geniş letilebilir olarak iş aretlenebilen tüm türler için) izlenir.

"Uzantılar biti" bir olarak ayarlanırsa, aşağıındaki kodlama bazen kısıtlamasız tiple aynıdır, ancak bazen aşağıındaki gibi farklıdır:

Geniş letilebilir bir INTEGER veya geniş letilebilir dizgiyi kodlarken "uzantı biti" bir olarak ayarlanırsa, ardından kısıtlamasız tipteki bir değil erle aynı olan bir kodlama gelir.

Bir SEQUENCE veya SET değil erini kodlarken "uzantı biti" bire ayarlanırsa, aşağıda kütükler, kütükte bulunan öge elerin, değil erlerini taşımak için ekleme noktasına eklenen özel bir kodlamayla (bkz. 15.2) kodlanmasıdır. Kütük dışındaki öge eler (ve varlıklarını belirlemek için).

Bir SEÇİ M değil eri kodlanırken "uzantı biti" bire ayarlanırsa, ardından seçim indeksinin özel bir kodlaması yapılır (teorik olarak sınırsız olmasına rağmen men değil erin genellikle küçük olacağını kabul ederek), ardından seçim alternatifin bir kodlaması gelir. ("Normalde küçük bir tam sayının" kodlaması için 8.2'ye bakın).

Bir SAYILANDIRILMIŞ değil eri kodlarken "uzantı biti" bir olarak ayarlanırsa, seçim dizini ile aynı kodlama kullanılır, ancak yine değil er teorik olarak sınırsızdır, ancak pratikte genellikle küçük olacaktır.

Yukarıdan, bir geniş letilebilirlik iş aretçisi dahil etmenin 1. versiyonundaki tek maliyetinin 1 bit olduğu görülecektir (muhtemelen bundan sonra yedi adede kadar dolgu bitinin eklenmesine neden olur). Türün gerçekte uzantıları varsa ve kütük dışındaki değil erler kodlanmışsa, genellikle bu tür değil erler için bir uzunluk alanı ek yükü elde ettik imizi daha sonra göreceğiz.

Kütük dışındaki kalan geniş letilebilir türlerin değil erleri için kodlama, geniş letilebilir olarak tanımlanmayan türler (ve kütük içindeki geniş letilebilir türlerin değil erleri için) için kodlamanın açıklamasından sonra aşağıda açıklanmışdır.

Yukarıdaki açıklamadan, kodlayıcıların ve kod çözümcülerin bir türün geniş letilebilir olup olmadığı ve öyleyse tam olarak hangi soyut değil erlerin kütükte olduğu konusunda anlaşmaları gerektiği açık olacaktır. Bir türün, tür tanımının doğrudan bir parçası olarak bir üç noktaya sahip olduğu durumlarda - SET, SIRALI, SEÇİ M, SAYILANDIRILMIŞ, çok az sorun vardır. Ancak tamsayı veya karakter dizisi gibi bir türün üç nokta içeren bir kısıtlamaya sınırlandığı durumlarda, durum (belki de ş. aşırıcı bir ş. ekilde!) o kadar net değil ve türün PER kodlamaları için geniş letilemeyeceğine pekala bildirilebilir. bir üç noktanın açık varlığına rağmen men! Bu alan, PER-visible kısıtlamaları tartışmasının sonunda tartışılımaktadır.

4 PER-görünür kısıtlama

4.1 konsept

PER kodlamalarını anlamak için çok önemli olan PER-visible kısıtlamaları kavramıdır. Bunlar, varsa, üst türün kodlamasını etkileyen (alt tür) kısıtlamalardır.

En önemli PER-visible kısıtlamaları, INTEGER tipine ve dizilerin uzunluklarına (veya SET OF ve SEQUENCE OF iç in yineleme sayılarına) yerleş tirilen kısıtlamalardır.

Ayrıca, PER görünür olan (bkz. Madde 6) bazı karakter dizisi türlerinin alfabesinde kısıtlamalar vardır ve bu karakter dizileri iç in karakter baş ina bit sayısını azaltabilir.

Yukarıdaki durumlarda PER-görünür olan kısıtlamalar oldukça geniş bir ş ekilde tanımlanmış tır. Tip referanslarının tekrarlanan kullanımı yoluyla "bir seferde biraz" uygulanabilirler veya parametreleş tirme kullanılarak uygulanabilirler. Ya da dahil edilen alt tip kısıtlamaları, kesiş imleri ve birleş imleri iç eren son derece karmaş ik alt tip spesifikasyonları olabilir.

Bu konuda yapılacak iki yorum var: Birincisi, ç oğ u belirtim oldukça basit, bu nedenle el kodlayıcılarının gerçek ek dünyadaki gerçek kısıtlamayı hesaplamak iç in çok fazla iş yapması gerekmeyi; ikinci olarak, bir ASN.1 derleyicisi, tamsayı türü, dizi uzunluğu u vb. iç in izin verilen de ğ erlerin kesin bir kaydına kadar bu tür keyfi genellik ifadelerini çözme hiçbir sorun ya ğ amaz.

4.2 De ğ iş ken parametrelerin etkisi

PER-görünürlüğ ünün önemli bir istisnası, gerçek kısıtlamayı belirlemeye çalış ırken, gerçek kısıtlamanın çözümünde bir de ğ iş ken parametresine (soyut sözdizimi tanımlanmış in de ğ eri olmayan bir parametre) metinsel olarak baş vurulmasıdır. kısıtlama PER görünür olmaktan çıkar ve bu kısıtlama yokmuş gibi kodlanır.

Bu, resmi olarak geniş letilebilir olan bir türün geniş letilebilir de ğ ilmiş gibi kodlandığı birkaç durumdan ilkidir. Bu durumda, PER-görünür olmayan bir kısıtlamada bir üç nokta iç erir, bu nedenle (baş ka kısıtlamaların uygulanmadığı varsayılarak) geniş letilemez ve kısıtlanmamış olarak kodlayacaktır.

De ğ iş ken parametreler hala yo ğ un bir ş ekilde kullanılmamaktadır, bu nedenle bu çok büyük bir sorun de ğ ildir, ancak metinsel olarak yukarıdaki terim, böyle bir parametrenin de ğ erini kullanıyor gibi görünen ancak gerçek sonucun olduğunu yerde birleş tirme ve kesiş me ifadeleri oluş turma olasılığ in ifade eder. ifade de ğ erlendirmesi, de ğ iş ken parametresinin sahip olabileceği de ğ er ne olursa olsun aynı olduğunu kanıtlar. Parametre sonucu etkilemese bile, metinsel varlığı kısıtlamayı ortadan kaldırır. Bu, derleyiciler iç in uygulama çabalarını kolaylaş tırmak ve elle kodlamada olası hataları önlemek iç in yapıldı.

PER-visible kısıtlamaları PER'nin daha vardır az ayrıntılı kodlamalar üretmek iç in kullandığı kısıtlamalar - öne ğ in - INTEGER (0..7) sadece üç bit olarak kodlar çünkü (0..7) kısıtlaması PER-görünürdür. BER tüm kısıtlamaları yok sayar ve bu nedenle her zaman bir uzunluk alanına ihtiyaç duyar. PER pragmatik bir bakış açısı benimser ve "kolayca" kullanılan ve önemli bant geniş li ğ i kazanımları sağlayan kısıtlamaları kullanır, ancak de ğ er daha karmaş ik kısıtlamaları göz ardı eder.

Bir kısıtlamada bir de ğ iş ken parametresinin varlığı PER'nin bu kısıtlamaların tamamını tamamen yok saydığını anlamına gelir.

4.3 Değerli işken uzunluklu kodlamalara sahip karakter dizileri

PER-görünürlüğ üne iliş kin dikkat edilmesi gereken bir diğ er önemli istisna, bir karakter dizisinin uzunluğu una iliş kin bir kısıtlamanın, dizide görünebilen (soyut) karakterlerin sayısı için geç erli olmasıdır. Kodlama UTF8 (veya GeneralString) gibi bir ş eyse, burada her karakteri kodlamak için gereken sekizli sayısı farklı karakterler için farklıdır (ve GeneralString durumunda kodlayıcı seçeneklerine bağlı olabilir), uzunluk kısıtlaması pek yardımcı olmaz. kodlama seviyesi - kodlamanın sonunu bulmak i

Yine basit tutmak - PER, her karakter aynı bitişavınsa **Kullanmadığ** i süreçte uzunluk kısıtlamalarını yok sayar.

(Yukarıdaki ifade kesinlikle doğru değil. UTF8 gibi bir kodlama şemasının ince ayrıntıları tam olarak anlaşılırsa, o zaman kodlanan soyut karakterlerin sayısı bilgisi aslında kodlamadan sonunu bulmak için yeterlidir, ancak PER ister. Bu kadar ayrıntılı analizlere başvurmadan kodlamadan sonunu bulabilmek için bir kod çözücü.)

Bu nedenle, her bir soyut karakter için sabit sayıda sekizliye sahip olan karakter seti türleri, bilinen çarpan türleri olarak adlandırılır ve bu tür türlerdeki uzunluk kısıtlamaları PER-görünürdür (ve azaltılmış veya ortadan kaldırılmış uzunluk kodlamalarına yol açacaktır), ancak karakter dizisi türleri için "bilinen çarpan türleri" değil ildir, kısıtlamalar PER-görünür değil ildir (türün değil erlerinin kodlamasını etkilemez) ve bu kısıtlamalardaki herhangi bir uzanti iş aretişi, PER kodlaması amacıyla dikkate alınmaz.

4.4 Şimdi karmaşık iklas alım!

Bu kitabın adı "ASN.1 Complete", bu yüzden PER-görünürlik ve geniş letilebilirlik hakkında biraz daha araştırma yapmak iyi olur.

İlk olarak, tek başına kullanılabilen, ancak genel durumda EXCEPT, INTERSECTION ve UNION kullanılarak bir araya getirilen bir dizi farklı türde alt tür kısıtlaması olduğuunu not ediyoruz. Temel yapı taşılarına bileşen kısıtlamaları ve tam kısıtlamaya dış seviye kısıtlaması diyoruz. Hem bileşen kısıtlamaları hem de dış düzey kısıtlamaları bir üç nokta içerebilir!

Gerçek belirtimler, uyguladıkları kısıtlamalar açısından nadiren karmaşık ıktır, ancak kodlama...
dizimini dikkate almak zorundadır. Bu bölüm gerçekten sadece entelektüel açıdan meraklılar içindir!

Bir bileşen kısıtlamasının PER-görünür olup olmadığı genel olarak bileşen kısıtlamasının türüne ve kısıtlanan türde bağlı olacaktır. Şekil III-19 bir liste vermektedir.

değ işken kısıtlaması	Asla görünmez
Tek değer er kısıtlaması	Yalnızca INTEGER için görünür
İçerdiği alt tür kısıtlaması	Her zaman görünür
Değer aralığı	Yalnızca INTEGER için ve bilinen bir çarpan karakter dizesi türünde bir alfabe kısıtlamasında görünür
Boyu kısıtlaması	OCTET STRING, SET ve için görünür SEQUENCE OF ve bilinen çarpan karakter dizisi türleri
İçin verilen alfabe	Bilinen çarpan karakter dizisi türleri için görünür
İçin alt tipleme	Asla görünmez

Şekil III-19: Kısıtlamaların PER-görünürüğünü

Şekil III-19'dan not edilmesi gereken iki önemli nokta, tek bir değer er kısıtlamasının yalnızca INTEGER'e uygulandığıında görünür olduğunu ve içeren bir alt tip kısıtlamasının her zaman görünür olduğunuudur. Bu, IA5String! Diyalim ki elimizde:

```
Alt tip ::= IA5String ("abcd" Bİ RLEŞ Tİ RME "abc" Bİ RLEŞ Tİ RME BOYUTU(2))
MyString ::= IA5String (KESİŞ İ M BOYUTU(3) Alt Türü)
```

MyString'de, tüm bileşen kısıtlamaları PER-görünürdür ve dış düzey kısıtlamasını çözbilmeyi umuyoruz. Alt türde, ilk iki bileşen kısıtlaması PER-görünür değil, ancak üçüncüsü görünürdür. Subtype ve MyString üzerindeki etkisi nedir? Bu soru ve bir dizi ilgili soru, ASN.1 grubu içinde "basit tut" ile "genel ve sezgisel tut" ile bir dereceye kadar çarpışan bazı uzun tartışmalara yol açtı.

İkinci önemli kural, herhangi bir bileşen kısıtlaması PER-visible değil ise, dış düzey kısıtlamasının tamamının PER-visible olmadığından beyan edilir ve kodlamayı etkilemez. Burada, bir bileşen ende veya dış düzey kısıtlamasında bir üç nokta varsa, kısıtlamanın tamamını göz ardı ettiğimiz için, türün geniş letilebilir bir tür olarak KODLANMADIĞINA dikkat edin.

Bu nedenle, yukarıdaki Alt tür, PER tarafından kısıtlanmamış olarak ele alınır ve MyString için set aritmetiği içinde kısıtlanmamış bir IA5String'in tüm soyut değerlerine katkıda bulunur.

Bir dış düzey kısıtlaması, yalnızca bileşen kısıtlamalarının tümü PER-görünür olduğunuunda PER-görünürdür.

MyString için, tüm bileşen kısıtlamaları PER-görünürdür, bu nedenle SIZE(3) uygulanır ve dizenin değerlerini, uzunluk 3'ün tüm olası soyut değerlerini içeriyeğmiş gibi kodlanır.

Üç noktanın kullanımıyla ilgili ek bir kural daha vardır. Bir PER kodlamasının geniş letilebilir olup olmadığıını ve kütte hangi değerlerin olduğunuunu belirlemek için küme aritmetiği gerçekleştirirken, bir bileşen kısıtlamasındaki (veya o bileşenin bileşen kısıtlamalarından herhangi biri - yukarıdaki Alt tür gibi) tüm üç nokta işaretleri (ve gerçek eklemeler) göz ardı edilir. Kısıtlanmış bir tür, PER kodlamaları için ancak ve ancak bir kısıtlamanın dış seviyesinde bir üç nokta göründüğünde geniş letilebilir; bunların tümü © OS, 31 May 1999

bileş en kısıtlamaları PER-görünürdür. Bu basit, ama belki de tam olarak beklediğiniz gibi değil il.

Şimdi, Sürüm 1'deki kısıtlamanın PER-görünür olduğunu, ancak Sürüm 2'de normalde PER-görünürlüğünü bozacak şekildeylerin (tek bir değil er kısıtlaması gibi) eklendiği bir Sürüm 2 spesifikasyonunu düşünün. Bu, orijinal Sürüm 1 kısıtlamasının PER görünürlüğünü etkilemez (ve etkilemesine izin verilemez), aksı takdirde birlikte çalışmaya zarar verilir. Bu nedenle, PER-görünürlüğü etkileyen (ve bir değil erin kodlanması eklini etkileyen) yalnızca kökte görünen kısıtlamanın bu kısımlarıdır.

Sınırlandırılmış bir tür, yalnızca üç nokta kısıtlamanın dış düzeyinde görünyorsa PER kodlaması için genişletilebilir.

Ancak birinin bir zamanlar dediği gibi, "Bu tür çarpılmış kısıtlama belirtimleri yalnızca ASN.1 grubu içindeki tartışmalarla görünür, asla gerçek kullanıcı belirtimlerinde görülmeyecek." Ve haklılar!

5 INTEGER'leri kodlama - hazırlık tartışması

INTEGER türünün (ve bilinen çarpan karakter dizilerinin uzunlıklarının) PER kodlaması için önemli olan gerçek değil erler değil il, PER-visible kısıtlamalarının izin verdiği değil er aralığıdır. Önemli olan en büyük ve en küçük değil erdir. Yalnızca iki değil er olan 0 ve 7'ye sahip olacak şekilde sınırlı olmalıdır. Önemli olan en büyük ve en küçük değil erdir. Aradaki boş luklar kodlamayı etkilemez.

Şekil III-20, PER tarafından görülebilen bazı basit kısıtlamaları ve PER'nin kodlama gerektirdiği ini varsayıdı ve değil erleri göstermektedir.

Alt sınırı olan herhangi bir tamsayı için (ve benzer şekilde dizelerin uzunlıklarını için), PER kodlamasında kodlanan şeyle, alt sınırın uzaklığıdır. Dolayısıyla, Şekil III-20'deki SET3 değil erlerinin kodlanması sadece 2 bit kullanır.

Tip tanımı

Kodlamaya ihtiyaç duyduğu varsayılan değil erler

TAM SAYI (0..7)

0 - 7

TAM SAYI (0 Bİ RLEŞ İ M 7)

0 - 7

SET1 ::= TAM SAYI (15..31)

15 - 31

SET2 ::= TAM SAYI (0..18)

0 - 18

SET3 ::= TAM SAYI (SET1 KESİŞ İ M SET2)

SET (BOYUT (0..3)) TAM SAYI

Yineleme sayısı: 0 ila 3

TAM SAYI (1 Bİ RLEŞ ME 3 Bİ RLEŞ ME 5 Bİ RLEŞ ME 7)

1 ila 7

Şekil III-20: Kodlama gerektirdiği varsayılan değil erler

Tamsayıların (ve dizelerin uzunlıklarının) kodlamasına baktığımızda, üç farklı durum olduğunu göreceğiz:

Sonlu bir üst ve alt sınırımız var (kısıtlanmış dē er olarak adlandırılır);

Sonlu bir alt sınırımız var, ancak üst sınırımız yok (yarı kısıtlı dē er olarak adlandırılır);

Alt sınırımız yok (sıfır her zaman alt sınır olduğu u için bu, dizilerin uzunluğu u için gerçekleş emez); buna sınırsız dē er denir; (tanımlanmış bir üst sınır olsa bile! - bu durumda üst sınır dikkate alınmaz).

Kısıtlı, yarı kısıtlı ve kısıtlanmamış tamsayıların ve kısıtlı ve yarı kısıtlı dizge uzunluklarının kodlamasını aş āg ıda açıyoruz ve ayrıca geniş letilebilir bir tür durumunda ortaya çıkan herhangi bir özel kodlamayı ele alıyoruz. Kısıtlanmış bir tamsayı (veya uzunluk) durumunda, kısıtlamanın izin verdiği aralı̄g a bağı̄l olarak birkaç farklı kodlama vardır. (İzin verilen mutlak dē erlerin önemli olmadığıını unutmayın).

Okuyucu, yalnızca negatif dē erlere izin verilip verilmedī ine dayalı olarak kodlamayı belirlemek ve ardından izin verilen en büyük dē eri işlemek iç in yeterli bit kullanmak yerine "aralık" (ve alt sınırдан ofset) kullanmanın zahmetine dē ip dē meyeceğini merak edebilir. kısıtlama tarafından. Kesinlikle INTEGER (10..13) ve INTEGER (-3..0) gerçek dünyada oluş maz! Ancak INTEGER (1..4) daha yaygın olabilir ve gerçek dē erleri kodladıysak "alt sınırından kaydırma" kurallıyla üç yerine yalnızca iki bit kullanır.

"Alt sınırından öteleme" ile çalı̄şmak ek bir karmaşıkılık gibi görünebilir, ancak aslında "Önce izin verilen tüm dē erlerin pozitif olup olmadığına bakın, sonra vb. gerçek uygulamalarda birkaç yer.

6 Etkili boyut ve alfabe kısıtlamaları.

6.1 Sorunun ifadesi

Yukarıda āg ıdaki gibi kısıtlamalardan bahsetmiş tık (ancak vurgulamadık):

```
MyString ::= PrintableString (FROM ( "0" .. "9")
                                Bİ RLİ K("#")
                                Bİ RLİ K("*"))
                                )
```

PER-göründür ve "MyString" dē erlerinin kodlanması için karakter başına yalnızca dört bitle sonuçlanır (yalnızca sıfırdan dokuz ve karma ve yıldız - on iki karakter içeren tüm dizilerden oluşur).

Bu, 14. maddede karakter dizisi dē erlerinin kodlanması tartışmasında daha ayrıntılı olarak açıklanmışdır, ancak burada alfabe kısıtlamaları için önemli olanın karakter aralığı dē il, izin verilen gerçek karakter sayısı olduğuunu unutmayın. Bu, kısıtlanmış tamsayıların ele alınmasından farklıdır, çünkü izin verilen neredeyse rastgele bir karakter seçime sahip bir karakter dizisi türü tanımlama ihtiyacının ortaya çıkması, rastgele bir tamsayı dē erleri seçime sahip bir tamsayı türü tanımlama ihtiyacından çok daha fazladır.

Bununla birlikte, yukarıdakiler gibi alfabe kısıtlamaları ile yine uygulanabilen uzunluk (boyut) kısıtlamaları arasında biraz zor bir etkileşimi vardır.

Örneğin, düşünün

```

MyString1 ::= IA5String (FROM ("01") KESİ Ş İ M BOYUTU (4))
MyString2 ::= IA5String (FROM ("TF") KESİ Ş İ M BOYUTU (6))
MyString3 ::= IA5String (MyString1 Bİ RLİ ğ MyString2)

```

Tüm kısıtlamalar PER-görünürdür ve MyString 1'in 4 karakterlik sabit bir uzunluğ a sahip olduğunu açıktır, bu nedenle uzunluk alanı olmadan kodlamalıdır ve yalnızca iki karakter "0" ve "1" içerir ve baş ina yalnızca bir bit ile kodlamalıdır. karakter. Benzer şekilde MyString2'nin karakter setini "T" ve "F" ile sınırlayan (yne karakter baş ina bir bit verir) ve 6 boyut kısıtlaması olan bir alfabe kısıtlaması vardır.

Ancak MyString3'teki alfabe ve boyut kısıtlaması nedir? Onlara sahip mi? Etkili bir boyut kısıtlaması ve etkili bir alfabe kısıtlaması kavramının devreye girdiği yer burasıdır .

6.2 Etkili boyut kısıtlaması

Bir "etkili boyut kısıtlaması", tek bir boyut kısıtlaması olarak tanımlanır, öyle ki, bu boyut kısıtlaması tarafından bir uzunluğ a ancak ve ancak bu uzunluğ a sahip kısıtlı türde en az bir soyut değil er varsa izin verilir.

Dolayısıyla önceki örnekte, MyString3 yalnızca 4 ve 6 uzunlıklarının soyut değil erlerine sahiptir. Ancak önemli olan, 4 ila 6 olan bir boyut kısıtlamasının aralığı idir. Bu, alt sınırı kaldırdığı imzada 0 ila 2'ye eş değil erdir, bu nedenle MyString3'ün uzunluk alanı 2 bit ile kodlanır.

6.3 Etkili alfabe kısıtlaması

Tam olarak eş değil er bir ş ekilde, bir "etkili alfabe kısıtlaması", izin verilen tek bir alfabe kısıtlaması olarak tanımlanır; öyle ki, bu alfabe kısıtlaması tarafından bir karaktere, ancak ve ancak kısıtlanmış türde içinde bir yerde içeren en az bir soyut değil er varsa izin verilir. o karakter.

Bu nedenle önceki örnekte, "0", "1", "T" ve "F" karakterlerinin tümü en az bir soyut değil er tarafından kullanılır ve etkin alfabe kısıtlaması bu (ve yalnızca bu) karakterlere izin verir, yani iki bit karakter baş ina kullanılacaktır.

Neyin PER-görünür olduğunu una iliş kin kuralların anlaşılması ve uygulanması koşuluyla, her durumda etkin alfabe ve etkin boyut kısıtlamalarını hesaplamak hem bir insan hem de bir bilgisayar için normalde basit bir meseledir.

Bu özellikle bir insan için doğrudur çünkü kısıtlamalar pratikte oldukça basittir. Bir bilgisayar için (bir ASN.1 aracında, ne kadar karmaşık veya çıkış yolu olursa olsun tüm olası kısıtlamaları işlemek üzere programlanması gereklidir), keyfi olarak karmaşık herhangi bir küme aritmetik ifadesini alabilen bir program yazılabilir (yalnızca boyut ve alfabe kısıtlamaları) ve onu etkili bir alfabe ve etkili bir boyut kısıtlamasına indirgeyin. Bunu aşağıdaki gibi eş itlikler kullanarak yapar:

A HARİ ÇB	eşittir	A KAVŞ AĞ (B DEĞ L)
ve		
vb	NOT (A Bİ RLİ ğ B) eşittir (NOT A) KESİ Ş İ M (B DEĞ L)	

Karakter dizisi türlerinde tek değil er kısıtlamalarına izin verilseydi, bu çok daha zor bir iş olurdu.

7 Kanonik etiket sırası

Okuyucu, PER'nin bir seçim dizini gerektirdiğini hatırlayacaktır; bu, bir SEÇİ M'deki alternatiflerin bir sırayla numaralandırılması anlamına gelir. Benzer şekilde, bu elemanların değil erlerinin iletişim için sabit bir sıra belirleyerek bir SET elemanları ile bir etiketi kodlama ihtiyacını ortadan kaldırır.

Bu amaçla alternatiflerin ve ögelerin metinsel sırasını kullanmak mümkün olabilirdi, ancak metinsel düzendeki herhangi bir değil iş iklik nedeniyle (belki de tamamen editorial nedenlerle sürüm 1'den sürüm 2'ye geçerken) bunun uygun olmadığı hissedildi.) satırdaki kodlamayı değil iş tirir. Esasen, mantıksız olduğunu düşündürmek için bu şekilde bir düzen değil iş ikliğiinin yasaklanması gerekiyordu.

Farklı dış düzey etiketlere sahip herhangi bir tür koleksiyonunu hem kodlayıcı hem de kod çözücü tarafından bilinen bir sıraya yerles tirmemize ve bu sırayı kodlama seçenekleri ve kümeleri için kullanmamıza izin veren bir etiket sıralaması tanımlarız.

Bir SEÇİ M'deki tüm alternatiflerin ve bir SET'teki tüm ögelerin zaten farklı (dış düzey) etiketlere sahip olması gerekiyinden, metinsel sıra kullanmanın açık bir alternatif varıdır: etiket değil erleri için bir sıra tanımlayın ve ardından etkili bir şekilde yeniden SET ögeleri için seçim indeksini veya iletişim sırasını belirlemeden önce SEÇİ M ve SET'i etiket sırasına göre sıralayın. Yapılan bu,

Sözde kanonik etiket sırası ş u ş şekilde tanımlanır:

Evrensel Sınıf (ilk)
Uygulama Sınıfı
Bağlama Özgü Sınıf
Özel Sınıf (son)

her sınıf içinde daha yüksek olanlardan önce gelen daha düşükük etiket numaraları ile.

Sadece küçük bir komplikasyon var - her zaman var! Çok u türün, tüm soyut değil erleri için aynı dış düzey etiketine sahip olduğunu hatırlayın ve geçerli olarak "türün etiketi" hakkında konuş abiliriz. Bunun doğrusu olmadığı tek durum, etiketlenmemiş bir seçim türü içindir. Bu durumda, farklı soyut değil erler farklı dış düzey etiketlere sahip olabilir ve "türün etiketi" hakkında o kadar kolay konuş amayız. (Ancak, tüm bu etiketlerin bir SET veya CHOICE'daki diğer herhangi bir türdeki etiketlerden farklı olması gerekiyini unutmayın). PER, türleri kanonik bir düzene sokmak amacıyla etiketlenmemiş bir seçim türünün etiketini, değil erlerinden herhangi birinin en küçük etiketi olarak tanımlar ve sorun çözülmüştür.

8 Sınırsız sayımı kodlama

Uzunluklara, yineleme sayılarına veya tam sayıların boyutlarına kısıtlamalar getirilirse, PER genellikle uzunluk alanını tamamen atlar veya uzunluk için yüksek düzeyde optimize edilmiş bir kodlama kullanır (daha sonra açıklanacaktır), aksi takdirde buna benzer (ancak farklı) uzunluk kodlamaları kullanır. BER'den olanlar. Bu maddede açıklanan bu kodlamaları.

Sonunda kavramları iş imiz bitti ve gerçek kodlamalara bakabiliyoruz