論文96-33A-3-4

개방 시스템 응용을 위한 개선된 ASN.1 컴파일러 설계 및 구현

(The Design and Implementation of an Enhanced ASN.1 Compiler for Open System Application)

金 弘 烈 *, 林 濟 鐸 *

(Hong Ryul Kim and Chae Tak Lim)

요 약

ITU-T 와 ISO 에서 권고하는 ASN.1 은 분산 개방 시스템들의 상호접속을 위해 널리 사용되는 국제표준 형식언어이다. 분산 개방 시스템의 상호접속을 위해서는 ASN.1 으로 정의된 데이타들을 BER 옥테트 스트림으로 변환하기위한 잘 정의된 부호화 및 복호화 모듈 구성이 필수적이다. 본 논문에서는 사용자가 정의한 ASN.1 데이타 타입들의 명세를 간단하고 간결한 사용자 인터페이스를 갖는 C-언어 부호화 함수 및 복호화 함수들로 자동 변환하는 새로운 기능을 갖는 ASN.1 컴파일러인 HYASNC 와 국제표준 BER 전송구문으로 변환하기 위한 HY BER 부호화/복호화 라이브러리 및 다양한 유틸리티 함수들을 설계 구현하였으며, 기존의 ASN.1 컴파일러들과의 상호 연동시험 및 성능 평가를 수행하였다.

Abstract

Abstract Syntax Notation One(ASN.1), defined by ITU-T and ISO, is a formal abstract specification language which has been widely used in international standards specification to inteconnect distributed open systems. It is necessary to have well defined encoder/decoder modules which translate ASN.1 datum to BER octets stream to interconnect disributed open systems. In this paper, we designed and implemented a new ASN.1-to-C compiler, called HYASNC(Hanyang ASN.1-to-C), which automatically translates an ASN.1 specification into C-language BER encoders and decoders with simple and neat I/F for the defined ASN.1 data types, and enhanced BER(Basic Encoding Rules) encoding/decoding libraries, called HY(Hanyang) BER library, and useful utility functions. And this paper discusses HYASNC compiler, HY BER runtime library's design and implementation principles, and also evaluates the performance of HY BER library and the interoperability with other ASN.1 compilers.

I. 서 론

최근 정보 산업의 발전과 더불어 이기종 컴퓨터 시스템들간의 상호 접속을 위한 요구가 증대되어 왔으나, 컴퓨터 시스템들 간의 정보 표현의 방법이 상이하여 분산 이기종 시스템의 상호접속에 많은 문제가 발생되었다.

분산 이기종 시스템내 응용 시스템들 간의 원활한 상호접속을 위한 외부 데이타 표현형식으로 ASN.1 (Abstract Syntax Notation One)[2]과 이들 데이타를 상호 교환하기위한 부호화 규칙으로 BER (Basic Encoding Rules)[3], CER(Cannonical Encoding Rules)[4] 및 DER(Distin- guished Encoding Rules)[4]등이 ISO 와 ITU-T 에 의해 국제표준화 되었다.

* 正會員, 漢陽大學校 電子工學科

(Dept. of Elec., Eng, Hanyang Univ.)

接受日字: 1995年12月28日, 수정완료일: 1996年2月13日

분산 개방 시스템의 상호접속을 위한 응용 시스템 개발에 있어서는 ASN.1 으로 정의된 프로토콜 데이타들에 대한 부호화 및 복호화 모듈들이 잘 구성되어야 하나, 이러한 작업들은 시간이 많이 소요되고, 귀찮으며, 수 작업시 많은 오류를 발생시킨다[1].

따라서 이러한 작업을 자동으로 수행하는 소프트웨어 도구가 요구되었고, 현재까지 많은 ASN.1 컴파일러[1][4][5] 들이 구현되었으나, 이들은 ASN.1 서브타입(subtype)들과 값(value) 정의에 대한 자동 코드 생성이 불가능하고, 자동 생성된 코드들의 사용이 복잡하였으며, 새로운 기능의 확장성이 어려운 단점이 있다.

본 논문에서는 국내 최초로 1993년 버젼 X.208 권고안[2]에 정의된 모든 ASN.1 기본타입과 기존의 컴파일러에서 처리하지 못한 기본값(default value), 서브 타입(subtype) 및 값(value) 정의에 대한 부호화/복호화를 위한 C 언어 스텁(stub)화일 및 헤드 화일들을 자동으로 생성하는 새로운 HYASNC 컴파일러(Hanyang ASN.1-to-C Compiler) 와 X.209 권고안[3]에 정의된 BER 부호화 규칙에 따른 새로운 BER 라이브러리 루틴들을 설계 및 구현 하였으며, 기존의 ASN.1 컴파일러 및 BER 라이브러리들과 호환성(interoperability)시험 및 성능 평가를 수행하였다.

서론에 이어 II 장에서는 ASN.1 추상구문 및 BER 부호화 규칙에 대해 설명하고, III 장에서는 구현된 컴파일러 및 라이브러리에서 적용된 ASN.1 과 C 자료구조 간의 변환 규칙을 제시하고, IV 장에서는 구현된 HYASNC 컴파일러의 구조, 기능 및 HY BER 라이브러리에 대해 기술하고, V 장에서는 성능 분석 결과를 제시하고, VI 장에서는 본 논문에 대한 요약 및 결론을 정리한다.

II. ASN.1 추상구문 및 부호화 규칙

1. ASN.1 추상구문

ASN.1 추상구문은 이기종 시스템들의 상호접속을 추상 데이타 타입(types) 및 추상 데이타 값(values)들을 기술하기 위한 하나의 표기법이며, 응용구문(예, 전자우편)이나 특정 프로토콜 엔티티와 관련된 프로토콜 데이타(PDUs: Protocol Data Units) 들의 구조를 정의하기 위해 사용되며 일반적인 고 수준 프로그래밍 언어와 유사하게 모든 변수들이 선언되었을때 그들과 관련된 ASN.1 데이타 타입들 역시 정의된다. 그리고 어떤 값이 그 변수에 할당될때 그 구문은 정의된 타입이 된다.

1) ASN.1 타입

ASN.1 에서 어떤 타입은 무한한 크기의 값들의 집합이며 주어진 ASN.1 타입의 값은 그 타입 값들의 집합에서 하나의 요소가 된다. 컴퓨터 시스템에서의 데이타 값들에 대한 해석은 먼저 어떤 타입인지를 파악하여야 한다. ASN.1 에서는 지원되는 데이타 타입은 표 1과 같다.

표 1. ASN.1 타입
Table 1. ASN.1 types.

타 입	비 고
단순 타입 (simple types)	구성요소가 없음(atomic)
구조적 타입 (structured types)	여러 구성요소 포함
태그 타입 (tagged type)	다른 타입에서 유도됨
기타 타입	CHOICE 나 ANY 타입을 포함하는 타입들

단순 타입은 X.208 에서만 정의되는 기본 타입으로 크게 스트링 타입(string type) 과 비스트링 타입(non-string type)들로 구분된다. 스트링 타입에는 OCTET STRING, BIT STRING, NumericString, PrintableString, TelexString, VideoString, VisibleString, IA5String, GraphicString, GeneralString 타입등이 있으며, 비 스트링 타입에는 BOOLEAN, INTEGER, NULL, REAL, UTCTime, GernalizedTime, OBJECT IDENTIFIER, ENUMERATED 타입등이 정의된다.

구조적 타입은 여러가지 타입들을 구성 요소로 갖는 새롭게 정의된 타입이며 모든 타입들과 값들은 ASN.1 할당 연산자(즉, ::=)를 갖는 이름(name)을 갖으며, 이들 이름들은 다른 타입이나 값들을 정의 할 때 사용될 수 있다. 구조적 타입의 구성요소는 표 1에 정의된 모든 타입들을 포함한다. 구조적 타입에는 SET, SET OF, SEQUENCE, SEQUENCE OF 타입등이 있다.

태그 타입은 암시적 태그타입(implicitly tagged type)과 명시적 태그타입(explicitly tagged type)으로 구분되며 기본 타입의 태그 값들을 변형하여 다른 타입으로 부터 얻어진다.

2) 태그 클래스 및 태그

CHOICE 와 ANY 타입 이외의 모든 ASN.1 타입들은 태그 클래스와 양수값을 갖는 태그(tag)를 갖는다. 태그 번호가 같은 ASN.1 타입들은 추상적으로 동일하다. 즉, ASN.1 타입의 이름은 추상적 의미에 영향을 미치지 못하고 오직 태그 값(tag value) 만이 영향을 미친다. ASN.1 에서는 다음 4 가지 태그 클래스를 정의한다.

표 2. ASN.1 태그 클래스
Table 2. Tag classes of the ASN.1.

태그 클래스	의 미
UNIVERSAL	모든 응용에서 동일한 의미를 갖는 타입들. X.208(ISO 8824)에서만 정의됨
APPLICATION	특정 응용에서만 의미를 갖는 타입들. (예: X.400 MHS, X.500 디렉토리서비스) 서로 다른 응용에서의 타입들은 동일한 application -specific 태그를 갖으나 다른 의미를 갖는다.
CONTEXT-SPECIFIC	특정 구조적 타입 내에서만 특정의미를 갖는 타입
PRIVATE	사용자 정의 타입

ASN.1 타입에 대한 태그와 클래스 값은 X.208 (ISO 8824) 권고안에 정의되어 있고, 이외의 태그들을 갖는 타입들은 여러 곳에서 정의되며 그들은 항상 암시적 또는 명시적 태깅으로 구해진다. X.208 에 정의된 유니버셜(Universal) 태그를 갖는 타입 및 태그 값들은 표 3과 같다.

표 3. 유니버셜 타입 및 태그 값
Table 3. Universal types and their's tag values.

ASN.1 타입	태그 번호(10 진수)	태그 번호(16 진수)
INTEGER	2	02
BIT STRING	3	03
OCTET STRING	4	04
NULL	5	05
OBJECT IDENTIFIER	6	06
SEQUENCE & SEQUENCE OF	16	10
SET & SET OF	17	11
PrintableString	19	13
IA5String	22	16
UTCtime	23	17

3) ASN.1 추상구문 규칙

ASN.1 타입 및 값들을 표현할때는 다음과 같은 특별한 규칙을 갖으며 다른 프로그래밍 언어 표기법과 같이 표현할 수 있는 유연성을 갖는다.

(1) 다수의 공란(space) 및 라인브레이커(line breaker)들은 단일 공란으로 간주한다.

(2) 주석(comment)은 하이픈 쌍들로 범위가 제한되거나, 하나의 하이픈 쌍과 라인 브레이커로 제한된다.

(3) 식별자(값들이나 필드들의 이름)와 타입 참조자(타입의 이름)는 소문자, 숫자, 하이픈으로 구성된다. 식별자는 반드시 소문자로 시작하고, 타입 참조자는 대문자로 시작된다.

(4) 키워드는 모든 문자들이 대문자로 구성된다.

ASN.1 추상 구문으로 기술되는 프로토콜 명세의 형식은 다음과 같다.

```
모듈명  DEFINITIONS ::=
BEGIN
      타입참조자-1::= 〈type definition〉
      타입참조자-2::= 〈type definition〉
      타입참조자-3::= 〈type definition〉
END
```

모듈명은 다른 ASN.1 추상 구문들과의 구분을 위한 것이고, 키워드(keyword) "DEFINITIONS" 는 "BEGIN" 과 "END" 사이에 프로토콜 데이타들에 대한 ASN.1 타입들이 정의되어 있음을 나타낸다. 타입 참조자(typereference)-1,-2,-3 은 기본 타입 또는 기본 타입들의 조합으로 생성된 새로운 타입들을 나타낸다. 후자의 경우를 정의된 타입(defined type)이라 한다.

정의된 타입의 예는 다음과 같다.

```
PersonnelRecord ::= SEQUENCE {
                                Name,
      title          [0]    VisibleString,
      dateOfHire     [1]    Date
}
```

2. BER 부호화 규칙

추상표현 구문인 ASN.1 에 대한 기본 부호화 규칙 (BER: Basic Encoding Rules)은 ASN.1 타입 및 값들에 대한 부호화 규칙 및 데이타 전송을 위한 정형화된 기본 포맷을 제공한다. 이들은 X.209(ISO 8825) 권고안에 정의되며, ASN.1 값들을 옥테트

(octets) 스트링으로 표현하기 위한 몇가지의 방법을 제공한다.

BER 부호화에서는 그림 1과 같은 4 가지 옥테트들을 포함한다.

Identifier octets (T)	Length octets (L)	Contents octets (V)	End-of-contents octets (EOC)

그림 1. BER 기본 포맷
Fig. 1. Basic format for BER.

(1) 식별자 옥테트(identifier octets)

ASN.1 값의 태그 클래스 및 태그 번호를 식별하여 타입들을 서로 구별 가능하게하며, 부호화 방법이 프리미티브(primitive)타입인지 구조적(constructed) 타입인지를 가르킨다. 식별자 옥테트의 구성은 다음과 같다.

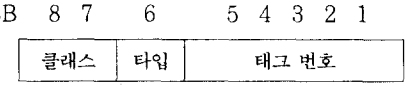

그림 2. BER 식별자 옥테트 포맷
Fig. 2. Identifier octet's format of BER.

BER 식별자 옥테트의 상위 2 비트는 ASN.1 타입 값의 태그 클래스에 따라 표 4와 같이 세트되며, 비트 6 은 ASN.1 타입이 프리미티브 형이면 "0" 으로, 구조적 타입이면 "1" 로 세트 되며 나머지 비트 5 ~ 1 은 태그 값을 세팅하기 위해 사용된다.

표 4. BER 식별자 옥테트의 클래스 비트 값
Table 4. Class bits values of BER identifier octet.

클래스	비트 8	비트 7
universal	0	0
application	0	1
context-specific	1	0
private	1	1

유니버설 타입은 태그 번호가 1 ~ 30 까지 이므로 1 개의 옥테트로 표현(low tag number form) 가능하나, 태그 타입(암시적 또는 명시적 태그)인 경우 태그 번호가 32 이상이므로 2 개 이상의 옥테트를 사용하여(high tag number form) 태그번호를 부호화 한다.

(2) 길이 옥테트(length octets)

한정길이(definite-length) 부호화 방법에서는 내용 옥테트의 길이를 나타내며, 1 개의 옥테트로 표시할때를 단 길이형(short form), 2 옥테트 이상으로 표현할 때를 장 길이형(long form) 이라하며, 구조적 무한길이(constructed indefinite-length) 부호화 방법에서는 "0x80" 으로 세트되며 길이가 무한함을 가르킨다.

(3) 내용 옥테트(contents octets)

식별자 옥테트(T)의 비트 6 이 "0" 으로 세트되면 기본 타입에 대한 값을 가르키며, "1" 로 세트되면 하나 또는 그 이상의 타입 구성 요소 값들의 연결을 가르킨다.

(4) 내용-끝 옥테트(end-of-contents octets)

길이 옥테트(L)가 "0x80" 으로 세트되었을 경우 내용 옥테트의 끝(EOC)에 "0x00 0x00" 을 덧붙여 내용 옥테트의 끝을 가르킨다. 그 외의 방법에서는 사용되지 않는다.

ASN.1 값의 BER 부호화 규칙에는 값의 타입과 길이를 아는지 여부에 따라 다음 3 가지 방법으로 부호화 된다.

1) 프리미티브 한정길이 부호화 방법

단순 비스트링 타입과 암시적 태깅에 의한 단순 타입들을 부호화 한다. 이 방법은 미리 값의 길이를 알아야 한다. 하나의 타입은 반드시 "타입 길이 값"(Type Length Value: 이하 TLV) 형식으로 부호화 된다.

T	L	V

그림 3. 프리미티브 한정길이 부호화 포맷
Fig. 3. Definite-length encoding format for primitive types.

2) 구조적 한정길이 부호화 방법

이 방법은 단순 스트링 타입, 구조적 타입, 단순 스트링 타입에서 유도되는 타입, 암시적 태깅에 의한 구조적 타입 및 명시적 태깅에 의해 어떠한 것으로부터 유도되는 타입들의 부호화에 적용된다. 이 방법은 미리 값의 길이를 알아야 한다. 구조적 한정길이 부호화 포맷은 그림 3과 동일하나 T 옥테트의 6 번째 비트가 "1"로 세트된다.

3) 구조적 무한길이 부호화 방법

이 방법은 2.2.2 에 명시된 타입들의 부호화에 적용되며, 미리 값의 길이를 알아야할 필요가 없다. 구조적

무한길이 부호화 포맷은 EOC (End Of Contents) 옥테트를 부가한다.

T	L	V	EOC

그림 4. 구조적 무한정길이 부호화 포맷
Fig. 4. Indefinite-length encoding format for constructed types.

Ⅲ. ASN.1 과 C 타입간의 변환 규칙

X.208 에 정의된 모든 ASN.1 타입은 C 언어의 자료형으로 변환이 가능하다. 단순 타입은 C 기본 자료형으로 직접 변환이 가능하고, 구조적 타입은 사용자 정의 C 자료형으로 변환 가능하다. 본 논문에서는 다음과 같은 변환 규칙을 제안하였으며, 이들 변환 규칙은 HYASNC 컴파일러가 입력 ASN.1 타입들에 대한 헤드 화일 및 BER 부호화 및 복호화 스텁 화일들을 생성할 때 C 자료 구조로의 변환시 적용된다.

1. 단순 타입 변환 규칙

BOOLEAN, INTEGER, OCTET STRING 등의 단순 ASN.1 타입들은 C 기본 자료형으로의 직접 변환이 가능하나, OBJECT IDENTIFIER 타입은 객체를 나타내는 정수들의 나열이므로 정수형에 대한 포인터로 변환되고, BIT STRING 타입은 전송 비트 수를 포함하는 사용자 정의 구조체 BITdata 로 변환된다. 그 이외의 단순 타입에 대한 변환 관계는 표 5와 같다.

표 5. ASN 과 C 간의 타입 변환규칙
Table 5. Type conversion rules between ASN.1 and C.

ASN.1 타입	C 자료형
BOOLEAN	char
NULL	char
OCTET STRING	char *
ObjectDescriptor	char *
GeneralizedTime	char *
OBJECT IDENTIFIER	HYOID *
INTEGER	long int
ENUMERATED	long int
REAL	double
PrintableString	unsigned char *
IA5String	unsigned char *
BIT STRING	BITdata *

2. 구조적 타입 변환 규칙

SET, SEQUENCE, CHOICE, SET OF, SEQUENCE OF 등과 같은 구조적 ASN.1 타입들은 기본적으로 struct { ..... } 를 이용한 사용자 정의 구조체로 변환되나 구조체 멤버의 구성이 약간 상의하다.

1) SET 타입 및 SEQUENCE 타입

SET 및 SEQUENCE 타입은 단순 타입 및 구조적 타입들을 구성 인자로 한다. 즉, 여러가지의 타입들이 모여 하나의 새로운 타입들을 형성한다. 따라서 SET 및 SEQUENCE 타입에 대응 되는 C 자료형은 사용자 정의 구조체로 변환된다. 이들의 구성인자가 단순 타입이면 표 5에 기술된 C 기본 자료형을 형성하고, 구조적 타입이면 이에 대응하는 사용자 정의 구조체에 대한 포인터를 생성한다. SET 및 SEQUENCE 타입에 대한 변환 규칙은 다음과 같다.

ASN.1 타입 정의	변환된 C 자료형

-- 모듈명 HYU
Member ::= SET {
name VisibleString,
id INTEGER,
info Information
}
 	
Information ::= SEQUENCE
{ date OCTET STRING,
married BOOLEAN
}
	struct HYU_Member {
	unsigned char *name;
	long int id;
	struct HYU_Information *info;
	};
	
	struct HYU_Information {
	char *date;
	char married;
	};

2) SET OF 및 SEQUENCE OF 타입

SET OF 및 SEQUENCE OF 타입은 오직 하나의 구성요소를 갖으며, 하나의 임의의 타입을 반복적으로 사용하고자 할 때 사용된다. 이들 타입의 구성 요소는 단순 타입 및 구조적 타입으로 구성된다. 따라서 이들 타입들은 링크드 리스트(linked-list) 구조를 이용한 C 구조체로 변환이 가능하다. 만일 구성요소가 단순 타입이면 표 5에 정의된 타입들의 링크드 리스트 연결 구조를 생성하고 구조적 타입이면 사용자 정의 구조체에 대한 포인터들에 대한 링크드 리스트 연결 구조를 생성한다. 다음 예는 SET OF 타입에 대한 변환이다.

3) CHOICE 타입

CHOICE 타입은 여러 구성요소 중 하나의 타입만을 선택하고자 할 때 사용된다. 단순 타입 및 구조적 타입 모두 구성 요소가 될 수 있다.

ASN.1 타입 정의	변환된 C 자료형

-- 모듈명 HY
SetOfExample ::= SET
{ a BIT STRING,
b SET OF Member
}
	struct HY_SetOfExample {
	BITdata *a;
	struct HY_Default0 {
	struct HY_Member *parm;
	struct HY_Default0 *next;
	} *b;
	};

따라서 CHOICE 타입은 구성인자의 상대적 위치를 지정하는 옵셋(offset)을 포함하는 공용체(union)로 이루어진 사용자 정의 C 구조체로 대응될 수 있다. 그 변환의 예는 다음과 같다.

ASN.1 타입 정의	변환된 C 자료형

-- 모듈명 HEI
Selection ::= CHOICE
{ a INTEGER,
b REAL,
c Member
}
	struct HEI_Selection {
	int offset;
	union {
	int a;
	double b;
	struct HEI_Member *c;
	} un;
	};

3. 디폴트 식별자 생성규칙

구조적 타입들의 구성 인자로 정의된 타입들은 식별자(identifier)를 가져야만 타입 서로간의 모호함을 완전히 배제 할 수 있고, 식별자가 생략된 경우에는 완전하지는 않지만 타입 만으로도 타입 상호간의 식별이 가능하다. 하지만, 식별자가 존재하지 않는 ASN.1 타입에 대해서는, 이에 대응하는 C 자료형은 반드시 식별자를 가져야 하므로 자동적으로 식별자를 생성해주는 메카니즘을 제공한다.

ASN.1 타입 정의	변환된 C 자료형

-- 모듈명 RHK
Sample ::= SET
{ INTEGER,
OBJECT IDENTIFIE
}
	struct RHK_Sample {
	long int RHK_HY0;
	int *RHK_HY1;
	};

하나의 모듈 내에서 구조적 타입의 수는 제한이 없으며, 임의의 구조적 타입내에서 식별자가 없는 구성요소의 수 또한 제한이 없다. 그러나 모듈 내에 존재하는 디폴트 생성자들은 반드시 유일(unique) 하여야 한다. 따라서 HYASNC 컴파일러에서는 생성 디폴트 생성자의 생성 규칙은 모듈명과 문자열 "HY" 와 0 부터 1 씩 증가하는 정수를 합성하여 생성한다.

Ⅳ. HYASNC 및 HY BER 라이브러리 설계 및 구현

HYASNC 컴파일러는 1993년 버젼 X.208 에 정의된 모든 ASN.1 기본 타입들, 서브 타입들 및 모든 ASN.1 값에 대한 부호화/복호화를 위한 C 언어 스텁 화일 및 헤드 화일들을 자동으로 생성하고, X.209 에 정의된 BER 에 따른 부호화/복호화를 위한 HY BER 라이브러리 루틴들을 지원한다. 본 연구에서 설계 구현된 HYASNC 컴파일러 및 HY BER 라이브러리 루틴들은 약 30,000 라인의 C 언어로 구현되었으며, 현재 Unix 환경(System V 및 BSD)에서 운용 가능하며, 추후 PC Windows 95 환경에 이식 할 예정이다.

1. HYASNC 컴파일러

본 연구에서 설계, 구현한 HYASNC 컴파일러는 향후 확장을 고려하여 기능 별로 모듈화 하였으며, 탑-다운(top-down) 형식으로 기술된 ASN.1 데이타 명세를 별도의 중간 화일 생성 없이 내부적으로 처리 하도록 하고, 출력 화일도 헤드, 부호화 스텁 및 복호화 스텁 화일만 생성하도록 설계 구현 하였다.

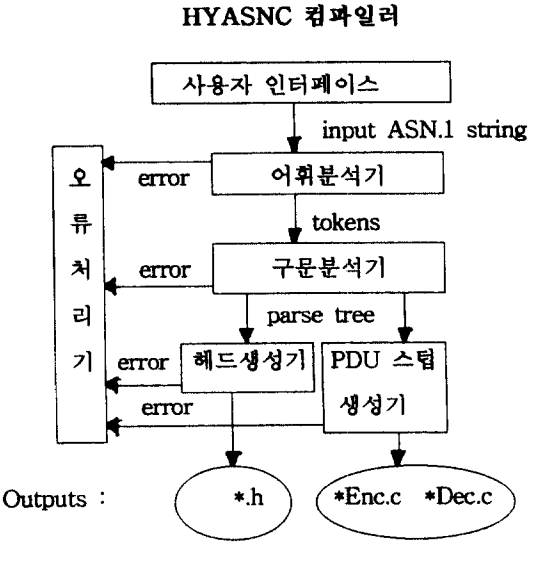

그림 5. HYASNC 컴파일러의 구조도
Fig. 5. Structure of HYASNC compiler.

HYASNC 컴파일러는 그림 5와 같이 사용자 인터페

이스, 어휘분석기(lexical analyzer), 구문분석기(syntax analyzer), 각종 ASN.1 타입들에 대한 헤드 및 C 언어 부호화/복호화 스텁 화일을 생성하는 PDU 스텁 생성기 및 오류 처리기 모듈로 구성된다.

각 모듈들의 처리 기능은 다음과 같다.

1) 사용자 인터페이스

사용자 인터페이스는 사용자에게 컴파일러 입력 및 옵션 입력을 위한 대화형 인터페이스를 제공한다.

2) 어휘분석기

어휘분석기는 X.208, X.219 권고안에 정의된 ASN.1 키워드와 식별자에 대한 어휘분석 및 토큰 발생 기능을 수행한다. 어휘분석기는 Unix 유틸리티인 flex 를 사용하여 구현하였다.

3) 구문분석기

구문분석기는 X.208, X.219 권고안에 정의된 ASN.1 구문에 적합한 문장이 입력되었는지를 검사하는 기능을 수행한다. 구문분석기는 Unix 유틸리티인 bison 을 사용하여 구현하였다.

4) 헤드생성기

ASN.1 입력에 대한 C 타입 헤드화일을 생성한다.

5) PDU 스텁생성기

ASN.1 입력 데이타 타입을 X.209 에 따른 BER 부호화/복호화를 수행하기 위한 HY BER 라이브러리 함수의 C 언어 스텁 화일을 생성한다.

6) 오류처리기

입력 ASN.1 스트링 내에 존재하는 어휘 오류, 구문상 오류와 헤드 및 스텁 생성시 발생되는 의미상(semantic) 오류들에 대한 에러 메시지 출력 및 처리 기능을 수행한다.

HYASNC 컴파일러는 하나의 ASN.1 모듈을 입력받아 다음 화일들을 자동적으로 생성한다 : *Enc.c, *Dec.c, *.h (여기서 * 는 ASN.1 모듈 이름으로부터 유일하게 생성된 화일 이름이다.)

*.h 화일은 입력 ASN.1 에 대한 C 타입 자료구조들로 구성된 헤드 화일이며, *Enc.c 화일은 C 언어 부호화 루틴들에 대한 스텁들을 포함하고, *Dec.c 화일은 C 언어 복호화 루틴들에 대한 스텁들을 포함한다.

2. HY BER 라이브러리

HY BER 라이브러리는 ASN.1 으로 기술된 데이타들을 개방 시스템들간에 상호 전송하기위한 국제표준 전송구문인 BER(Basic Encoding Rules) 옥테트 스트림으로 부호화하고 복호화 하는 함수들 및 다양한 유틸리티 함수들을 포함한다.

HY BER 부호화 라이브러리들은 옥테트 복잡도가 최소가 되도록하고, 복호화 시간 단축을 위해 2 가지 한정 길이 부호화 방법을 채택했으며, 복호화 라이브러리들은 2.2.1 - 2.2.3 절에 기술된 3 가지 부호화 방법으로 부호화된 모든 옥테트들을 모두 처리 할 수 있도록 설계 구현하였으며, 유틸리티 함수들도 응용 프로그래머의 활용이 용의하도록 설계 구현하였다.

HY BER 라이브러리 루틴들은 단순 타입 및 내용 부호화/복호화 함수 라이브러리, 부호화/복호화 유틸리티 루틴들로 구성되며 구조적 타입 및 태그 타입은 이들 라이브러리의 조합으로 처리된다.

1) 부호화 함수 라이브러리

HY BER 부호화 함수 라이브러리 루틴들은 C 타입의 데이타 값을 BER 전달구문의 옥테트 스트림으로 번역하는 기능을 제공한다.

표 6. HY BER 부호화 함수 루틴
Table 6. Encoding function routines of the HY BER.

단순 타입 부호화 루틴	내용 부호화 함수 루틴
EncBoolean(),	
EncInteger ()	Int2Str ()
EncBitString(),	Bit2Str ()
EncOctetString ()	Oct2Str ()
EncNull ()	
EncReal ()	Real2Str ()
EncNumericString ()	Numeric2Str()
EncPrintableString ()	Print2Str ()
EncTeletexString ()	Teletex2Str ()
EncVideotexString ()	Videotex2Str()
EncIA5String ()	IA52Str ()
EncGraphicString ()	Graphic2Str ()
EncVisibleString ()	Visible2Str ()
EncGeneralString ()	General2Str ()
EncUTCTime ()	UTC2Str ()
EncGeneralizedTime ()	GTime2Str ()
EncObject Identifier ()	OID2Str ()

2) 복호화 함수 라이브러리

HY BER 복호화 함수 라이브러리 루틴들은 BER 전달구문의 옥테트 스트림들을 C 타입의 데이타 값들로 번역하는 기능을 제공한다.

3) 유틸리티 루틴

유틸리티 루틴들은 부호화 및 복호화 함수 루틴에서 다양한 기능을 제공하며, 사용자들이 응용 프로그램 작성시에 자주 요구되는 기능을 라이브러리화 하였다.

표 7. HY BER 부호화 루틴
Table 7. Decoding function routines of the HY BER.

단순 타입 복호화 루틴	내용 복호화 함수 루틴
DecBoolean ()	
DecInteger ()	Str2Int ()
DecBitString ()	Str2Bit ()
DecOctetString ()	Str2Oct ()
DecNull ()	
DecReal ()	Str2Real ()
DecNumericString ()	Str2Numeric ()
DecPrintableString ()	Str2Print ()
DecTeletexString ()	Str2Teletex ()
DecVideotexString ()	Str2Videotex ()
DecIA5String ()	Str2IA5 ()
DecGraphicString ()	Str2Graphic ()
DecVisibleString ()	Str2Visible ()
DecGeneralString ()	Str2General ()
DecUTCTime ()	Str2UTC ()
DecGeneralizedTime ()	Str2GTime ()
DecObject Identifier ()	Str2OID ()

표 8. 유틸리티 함수 루틴
Table 8. Utility function routines of the HY BER.

유티리티 루틴	기 능
HYalloc ()	HY 동적 메모리 할당
HYfree ()	HY 동적 메모리 해제
Octet2HY ()	옥테트 스트림을 HY 리스트로 변환
Octet2Stream ()	HY 리스트를 옥테트 스트림으로 변환
HYerror ()	에러 메시지 출력 함수
HYdebugTest ()	입력 데이터의 화면 출력
HYOidCmp ()	Object Identifier 비교

V. 성능 평가

본 연구에서 설계 구현된 HYASNC 컴파일러 및 HY BER 라이브러리의 성능 평가를 위해 현재 가장 널리 사용되고 있는 ASN.1 컴파일러인 "ISODE 8.0 컴파일러"와 비교 평가 하였다.

1. HYASNC 성능평가

HYASNC 컴파일러는 X.208의 모든 단순 및 구조적 타입의 처리 기능을 갖는 기존의 ASN.1 컴파일러[1] [5] [6] 와 달리 다음과 같은 기능이 추가되었다.

(1) X.208 에 정의된 서브 타입 및 값 범위(value range) 처리기능

(2) X.208 에 정의된 모든 값에 대한 처리기능

(3) 기본 값(default value) 처리 기능

서브 타입 및 값 범위 처리 기능은 사용자 입력 데이타의 유효성(validity) 자동 검증을 위한 함수 생성에 요구되는 기능이며, 값에 대한 처리 기능은 응용 프로토콜 명세에 정의된 값들에 대한 헤드 화일을 자동 생성하고, 기본 값은 "DEFAULT" 필드에 대해 복호화 시에 반드시 요구되는 기본 값 처리 루틴의 자동 생성 기능을 갖는다. 이러한 기능은 기존의 ASN.1 컴파일러[1] [5] [6]에서는 전무한 기능이다.

표 9. ASN.1 컴파일러 처리 기능 비교표
Table 9. Functionality fo ASN.1 compilers.

ASN.1 타입 처리 기능	HYASNC	ISODE 8.0
단순 타입	○	○
구조적 타입	○	○
서브 타입 및 값 범위	○	×
ASN.1 값	○	×
기본 값	○	×

표 9는 HYASNC 컴파일러와 ISODE 8.0 컴파일러의 ASN.1 타입 처리기능성을 비교하였다. HYASNC 컴파일러의 기능이 개선되었음을 알수 있다.

2. BER 라이브러리 성능평가

부호화 및 복호화 루틴의 저장공간 복잡도(storage complexity)는 부호화될 데이타 값의 타입 정보를 제공하는 부가 옥테트들의 수로 측정되는 옥테트 복잡도(octet complexity)로 해석된다[1].

모든 BER 구현은 옥테트 복잡도 오버헤드를 갖으나, 서로 다른 BER 구현은 어떤 데이타 값을 부호화 또는 복호화 하는데 소요되는 시간은 서로 다르다.

HY BER 라이브러리의 성능평가를 위해 다음 2 가지 ASN.1 타입을 선택한다[1].

(1) 가장 단순하며 자주 사용되는 단순타입인 옥테트스트링

(2) X.208 에 정의된 구조적 타입인 Personnel Record 타입

HY BER 라이브러리 성능평가를 위한 시험환경은 Solaris 1.0/SUN Sparc-20 호환 기종에서 수행하였다.

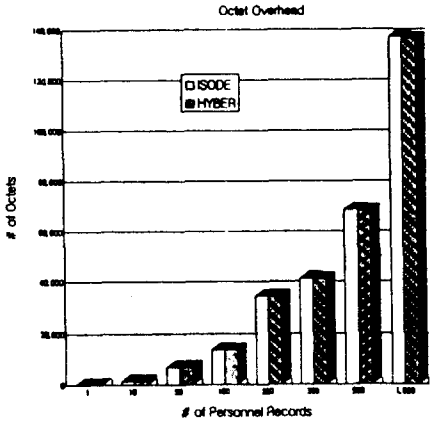

그림 6. 부호화 옥테트 복잡도
Fig. 6. Octet overhead complexity of encoding.

그림 6은 Personnel Record 를 부호화 하였을때의 옥테트 복잡도를 비교하였다. HY BER 부호화 방법이 ISODE 부호화 방법보다 옥테트 복잡도가 개선되었음을 확인 하였다.

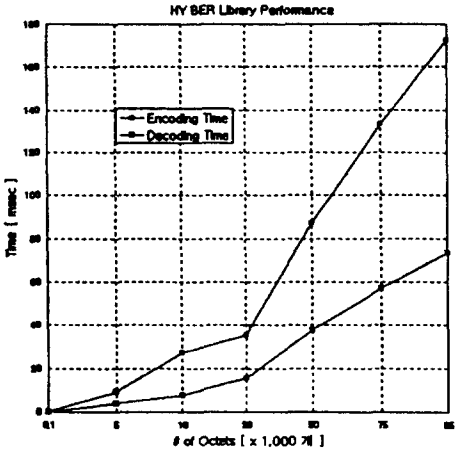

그림 7. 옥테트 스트링 부호화/복호화 시간
Fig. 7. Encoding/decoding time for Octet strings.

그림 7은 임의의 옥테트 스트링에 대한 HY BER 라이브러리의 부호화/복호화 시간을 나타낸다. ISODE 8.0 에서는 옥테트 스트링을 2 진 스트링(binary string) 또는 16 진수 스트링(hexadecimal string) 이 아닌 문자 스트링으로 처리하는 오류가 발견되어 상대적 평가가 불 가능 하였다. ISODE 에서는 이외에도 ASN.1 표준에 어긋나는 다수의 오류를 포함하고 있다.

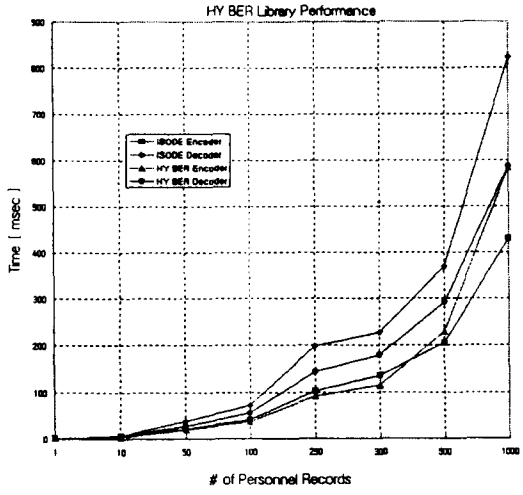

그림 8. Personnel Records 부호화/복호화 시간
Fig. 8. Encoding/decoding time for Personnel Records.

그림 8은 Personnel Record 를 부호화하고 복호화 할 때의 소요시간을 비교 하였다. 부호화 시간은 Personnel Record 가 500 개 이하에서는 HY BER 라이브러리가 우수하였으며, 복호화 시간은 성능이 현저히 개선되었음을 확인하였다.

3. BER 라이브러리 호환성 시험

설계 구현된 HYASNC 컴파일러와 HY BER 라이브러리에 대한 무결성을 검증하기 위해서 다음과 같은 시험을 수행하였다. 임의의 문자 스트링 및 Personnel Record 타입 데이타에 대해 HYASNC 와 ISODE 8.0 을 이용하여 BER 부호화 및 복호화 함수를 자동 생성한 후, 각각의 부호화 함수에서 생성된 부호화된 옥테트들을 각각의 복호화 함수들로 복호화 한 결과와 다른 복호화 함수를 이용한 상호 복호화(cross decoding)한 결과를 비교 검토 하였다.

표 10. HY BER 라이브러리 호환성 시험 결과
Table 10. Interoperability test result of HY BER library.

부호화 \ 복호화	HYASNC's Decoder	ISODE 8.0's Decoder
HYASNC's Encoder		
ISODE 8.0's Encoder		

표 10은 HYASNC 및 ISODE 8.0 컴파일러에서 생성된 부호화 및 복호화 함수들의 호환성 시험한 결과이다. 본 시험을 통해 HYASNC 컴파일러 및 HY BER 라이브러리의 무결성을 확인 하였다.

VI. 결 론

본 연구에서는 확장이 용의한 새로운 처리 기능이 추가된 ASN.1 컴파일러와 성능이 향상된 BER 라이브러리를 설계 구현하여 성능 평가를 수행하였다. HYASNC 컴파일러에서 추가 구현된 ASN.1 서브타입 처리기능, 값의 처리기능 및 기본값에 대한 처리 기능성을 비교 하였으며, 또한 구현된 BER 라이브러리가 기존의 ISODE 8.0 BER 라이브러리와 완벽한 호환성(interoperability)이 보장되고 성능이 개선되었음을 확인하였다.

향후 과제로는 X.690 에 정의된 DER/CER 및 X.691 에 정의된 PER(Packed Encoding Rules) 라이브러리의 구현과 각 라이브러리들의 성능 평가, 다양한 ASN.1 마크로 처리 기능의 구현이 요구된다.

참 고 문 헌

[1] G.W. Neufeld and Y. Yang, "The Design and Implimentation of an ASN.1-C Compiler", *IEEE Trans. on Software Engineering*, Vol. 16, No. 10, Oct. 1990.

[2] ITU-T, *Recommendation X.208 : Specification of Abstract Syntax Notation One*, 1993 (see also ISO 7498).

[3] ITU-T, *Recommendation X.209 : Specification of Basic Encoding Rules for Abstract Syntax Notation One(ASN.1)*, 1993 (see also ISO 8825).

[4] M. T. Rose, *The ISO Development Environment : User's Manual Vol 1. - 5.*, Performance Systems International, Inc., July 18, 1991.

[5] M. Sample and G. Neufeld, *Snacc 1.0 : A High Perfomance ASN.1 to C/C++ Compiler*, Feb. 1993.

[6] B. S. Kaliski, *A Layman's Guide to a Subset of ASN.1, BER and DER*, RSA Data Security, Inc., June 3, 1991.

[7] A. S. Tanenbaum, *Computer Networks*, Prentice-Hall, 1989.

[8] 김홍렬, 김계완, 임제탁, "개방 시스템을 위한 ASN.1 컴파일러의 설계 및 구현", *'95 추계 전자공학회 학술대회 논문집*, 95 제 18권 2호, pp. 425-428, Dec. 1995

저 자 소 개

金 弘 烈(正會員)
1964년 3월 2일생. 1986년 2월 한양대 전자공학과 학사학위 취득. 1988년 2월 한양대 전자공학과 석사학위 취득. 1988년 1월 ~ 현재 현대전자산업(주) S/W 연구소 선임연구원. 1993년 8월 ~ 현재 한양대 전자공학과 박사과정 재학중. 주관심 분야는 OSI, Multimedia Communication, ATM-based Protocol Interworking 등

林 濟 鐸(正會員) 第 30 卷 B 編 第 6 號 參照
현재 한양대학교 전자공학과 교수