

解説

CCITT における形式記述技法の標準化
動向[†]丸 山 勝 己^{††}

1. はじめに

CCITT での形式記述技法の動向が小生に与えられたテーマであるが CCITT の形式記述技法である仕様記述言語 SDL に関しては本特集号で若原氏が詳しく説明される。(OSI モデルの仕様記述を主眼とした記述法を形式記述技法 FDT: Formal Description Technique と呼ぶ。なお SDL は OSI モデルに限定されない仕様記述言語であり FDT の機能も備えている。)したがって、本稿では SDL のみならず CCITT でのソフトウェア言語関連の検討概要を述べることにしたい。

CCITT (国際電信電話諮問委員会) は、その名の通り電信電話に関する国際標準を研究・勧告する機関である。CCITT の活動は 4 年ごとの会期で区切られており総会が開催されてその会期の勧告の承認と次会期の研究課題の設定が行われる。現在は以下の研究委員会 (Study Group: SG と略す) からなっている。

- SG-I: 電信, データ伝送及びテレマティックサービス定義、運用及びサービス品質
- SG-II: 電話網と ISDN の運用
- SG-III: 一般料金原則
- SG-IV: 国際回線及び自動半自動網の保守
- SG-V: 電気磁気的妨害などからの施設防護
- SG-VI: 局外施設
- SG-VII: データ通信網
- SG-VIII: テレマティックサービス端末装置
- SG-IX: 電信網と端末装置
- SG-X: 電気通信用語と手法
- SG-XI: ISDN と電話網の交換と信号方式
- SG-XII: 電話網と端末の伝送品質
- SG-XV: 伝送方式

[†] Standardization of Formal Description Techniques in CCITT
by Katsumi MARUYAMA (NTT Communication Switching
Laboratories).

^{††} NTT 交換システム研究所

SG-XVI: 電話網によるデータ通信
SG-XVII: ISDN を含むデジタル網

以前の CCITT 活動の中心は、網間インタフェース、信号方式、接続品質、サービスなどの研究・勧告であった。しかし、交換機が蓄積プログラム制御化され (Stored Program Controlled Exchange System: 以下 SPC 交換機と呼ぶ) ソフトウェアの重要性とソフトウェア開発ネックの問題が強く認識されるに従い、ソフトウェアに関する標準化まで担当するようになった。

2. CCITT とソフトウェアの標準化

2.1 発端

CCITT でのソフトウェア研究の発端は古く、1968 年の CCITT 総会にスウェーデンが新しい研究課題として提案したことに始まるとのことである。(SPC 交換機の記念碑ともいうべき AT&T の No. 1 ESS 交換機が商用化されたのは 1965 年であることを考えると、大変に早いスタートではあった。)

この提案に基づき、1969~72 年会期には第 XI 研究委員会 (電話交換の信号方式などの検討を担当) の研究課題の一つとして “SPC 交換機のプログラムロジックの仕様化の検討” が設定された。しかし、交換機用ソフトウェアの具体的な検討には至らずに標準化に関する各国のコンセンサス作りが行われただけで、実際の検討活動は 1973~76 年会期に延期された。このように CCITT におけるソフトウェアの標準化の出だしは遅々としたものであった。

2.2 ソフトウェアの具体的な検討開始

ようやく 1973~1976 年会期に入って、第 XI 研究委員会に第 3 作業部会が設けられ以下の 3 課題の詳細な検討が開始された。

- ① SPC 電話交換機の機能仕様の表現法および内部論理の表現法: SPC 交換機的设计に必要な要求仕様ならびに実現されたシステムの仕様の正確・簡潔な

記述法の標準化を目的とする。

② SPC 電話交換機のための高水準プログラミング言語：SPC 交換機のプログラム記述に使われる高水準プログラミング言語の標準化を目的とする。

③ SPC 電話交換機のためのマンマシン言語：SPC 交換機の運転保守インタフェースの標準化を目的とする。

この研究会期は実り多く、3 課題ともにその骨格と今後の展開の基盤が構築された。ついで、1977～1980 年会期で各言語とも基本内容的にはほとんど完成し、それぞれ以下の名称で勧告化された。

① CCITT 仕様記述言語 SDL (Specification and Description Language)

② CCITT 高水準言語 Chill (CCITT High Level Language)

③ CCITT マンマシン言語 MML (Man-Machine Language)

これらの課題は現在まで継続課題として機能追加や内容のリファインが続けられてきているが、内容的に大幅な変更はない。

2.3 ソフトウェア専門の研究委員会の設置

ソフトウェアの重要性が認識 (ソフトウェア開発は金食い虫で、どうにかしないと大変だというのが各企業トップの本音?) され、1985 年からソフトウェア関連の標準化の検討を専門とする “第 X 研究委員会：電気通信用語と手法” が設定された。この研究委員会には、従来第 XI 研究委員会 (電話交換)、第 VII 研究委員会 (データ通信)、CMBD (信頼性) などで検討されていたソフトウェア関連課題が集められ、以下の標準化活動を行ってきている。

(1) 仕様記述言語関係

(a) SDL の改良と普及活動

(b) 形式記述技法 (FDT: Formal Description Technique) に関する ISO との共同検討：FDT に関して ISO と CCITT とはお互いに情報交流をしている。また LOTOS の図式記述版である Graphical-LOTOS も協力して設計を進めている。

(2) 高水準言語 Chill

Chill の言語仕様の保守と普及活動を進めている。

(3) 電気通信システムの開発保守支援環境

電気通信システムの開発と保守を、ライフサイクル全体をとおして支援する環境について、特に各支援機能に対する要求条件やそれら相互間のインタフェースを検討している。

(4) マンマシン言語関係

電気通信システムの保守運用に関して、グラフィカル端末の使い方、保守運転機能の整理、センサ管理方式などの検討を進め、既存勧告書の機能追加を行っている。

2.4 CCITT の検討の特色

(1) 初期検討の重視

新規方式の標準化を行う場合には、要求条件整理、概念整理、共通認識化など初期検討に十分な時間をかけて検討をスタートする。このため時間はかかるが検討深度を深めることができる。

(2) Committee 設計

ソフトウェア分野をみると、天才的個人の設計によるものに魅惑的・エレガントな製品がある。Committee 設計による製品には、多数の意見を適切に採り入れて有効性や影響力があるものと、多数の要求を採り入れてすぎて肥大化した使いにくいものがある。CCITT の勧告は典型的な Committee 設計ではあるが、徹底的に議論して理想解を求めるので、時間はかかるが比較的出来がよいと思われる。

(3) 通信網運用会社と製造メーカーの整合

通信網運用会社の立場からは、できるだけ厳密に標準化されてどこの交換機も統一的に扱えることが望ましい。これに対し、製造メーカーの立場からは自社の技術・独自性を活かせるように必要以上の規制は望まない。このような二つの立場を整合する必要性から、たとえばマンマシン言語ではパラメータまで含めて異種交換機のコマンドを統一することは不可能であるので、コマンドを設計する上でのガイドラインを示した勧告となっている。

(4) CCITT と ISO との差異

CCITT は通信分野の標準化、ISO は情報処理分野の標準化、という傾向が従来からある。しかし、最近では両者の共通分野が増えリエゾンが進んでいる。たとえば X シリーズプロトコルのように CCITT と ISO でまったく同一内容とするものが増えてきた。

3. 仕様記述法

3.1 仕様記述法標準化の研究経緯

CCITT の仕様記述言語 SDL は、以下の要求に沿って設計された。(ISD の FDT の検討は当初から厳密性を目的に発足しているのに対し、SDL は簡明性を重視していた。)

① 発注や入札の際に標準化された仕様記述法があ

SERVICE A_subscriber_actions

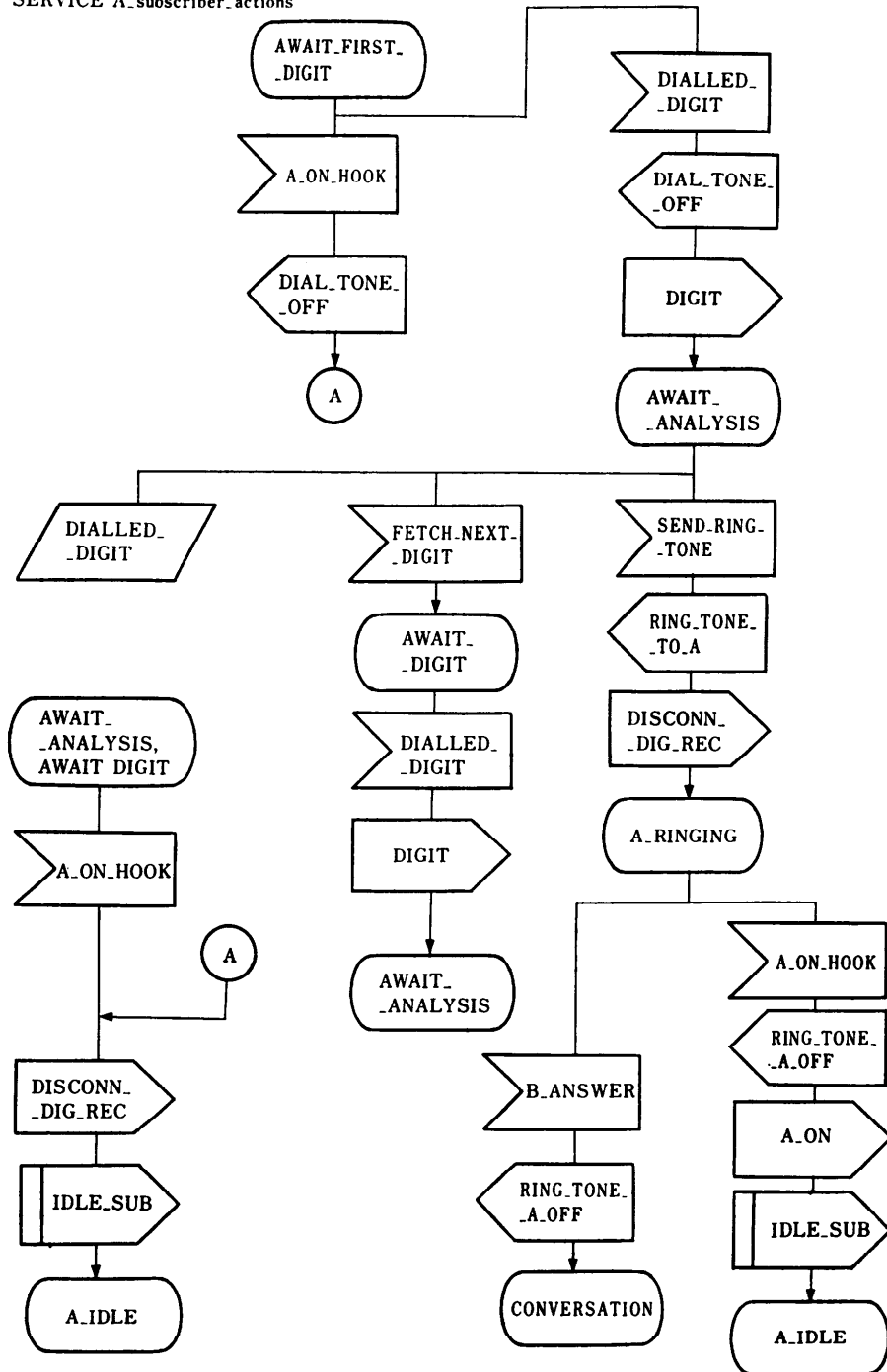


図-1 SDL による仕様記述例

ると便利である。通信網運用側の立場からは要求仕様を曖昧性なく正確簡潔に指定する必要がある。製造メーカー側の立場からは、要求仕様の正確な把握、さらに製品の仕様や内部論理を正確簡明に説明する必要がある。

② 競合する SPC 交換機の比較をするための手法が望まれる。

③ ソフトウェア開発には正確な仕様記述が非常に重要である。

仕様記述法の具体的な検討は、1973～76年会期にさかのぼる。最初は国際標準化検討のベースとなる仕様記述法の募集が行われ、以下の手法が提案された。

(a) 日本案：電子交換機 DEX (Denden Kosha Exchange System) 研究の中で産み出された手法で、交換機能を外部信号に対するシステム状態の変化としてとらえて状態遷移図として表現する手法である。数字受信状態、話中音送出状態などといった“状態”の内容を図式記述して直観的理解をねらっているのも特徴である。

(b) オーストラリア案：日本案をベースにシステム構成に依存した内容をマシン独立にするなどの改良をしたものを提案した。

(c) 英国案：State Diagram, State/Channel Chart, Flow Chart の3種からなる。

(d) スウェーデン案：システムを複数のサブシステムに分解し、各サブシステムの機能をフローチャートと信号/状態マトリックスで表現する手法。

このようにいずれの提案も状態遷移モデルをベースにしている。これは、実時間システムは一般に有限状態マシンとしてモデル化しやすいことから当然の方向といえる。なかでも日本案は使用実績もあり、SDL の検討に大きな影響を与えた。

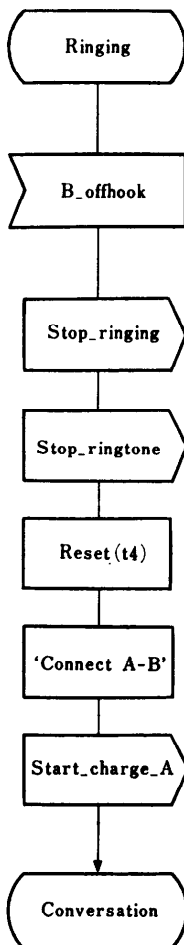
3.2 SDL の初期

1976年には、状態遷移モデルに基づいた最初の(原始的)SDLが勧告化された。現在の詳細厳密な記述能力を備えたSDLに比べると、状態

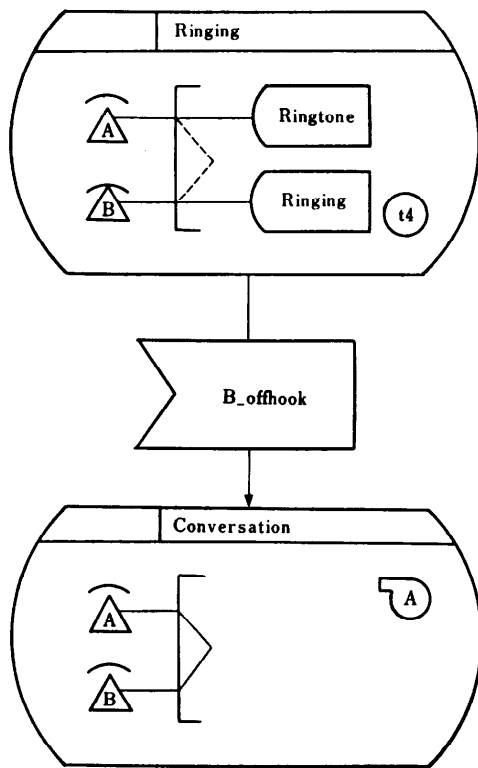
遷移概念のみの原始的でシンプルな仕様記述法である。

SDL記述の例を図-1に示す。有限状態マシンの二大要素は“状態”と“遷移”であるが、この例では状態の内容は状態名のみが書かれており、遷移の中で行うべき処理を手順を踏んで詳細に定義した内容となっている。つまり遷移内容指向の記述法となっている。

これに対し、状態定義指向の記述法もできるようにになっている。状態定義指向の記述法の場合は、状態ボックス中の状態定義を図式表現(Pictorial Elements という)を用いて詳細に記述する代わりに、遷移の内容は簡略化できる。電話交換機の被呼者呼出状態から通話状態までを遷移内容指向と状態定義指向で書いた例を図-2に示す。状態定義指向の記述法は、日本とオーストラリアが愛用している。



(a) 遷移内容指向



(b) 状態定義指向

図-2 同一内容の遷移内容指向及び状態定義指向によるSDL記述

ここで面白いのは、われわれの経験からは状態定義を図式記述することは直観的に分かりやすいのであるが、欧米人にいわせると Pictorial Elements は象形文字に馴染んでいる者には直観的かもしれないが、表音文字文明にはそうでなく、曖昧さが気掛かりだとのことである。

3.3 現在の SDL

SDL は 1976 年の最初の勧告から現在まで CCITT で継続検討されており、機能追加や改良が行われてきている。現在の SDL は、以下のように非常に厳密な記述まで可能な（ただし複雑でもある）仕様記述言語になっている。

- 図式記述法 (SDL/GR) とプログラミング言語形式の記述法 (SDL/PR) の両方が用意されている。両者は意味的にはまったく同一で相互変換が可能である。

- 厳密な動作モデルが定義されている。

- 抽象データタイプによる仕様記述法 ACT-1 を含んでいる。ACT-1 は ISO との検討協力により LOTOS のデータ記述法と共通内容になっている。

- インプリメンテーション言語並の詳細も書ける。

- 直観的な Informal な仕様記述も行える。

なお、SDL が評判のよい理由の第一は図式記述法 (SDL/GR) の直観的な理解しやすさにある。

詳細は、本特集号にて若原氏が説明される。

3.4 SDL の実用化状況

SDL は、交換機の仕様記述、プロトコルの仕様記述などに世界的に広く使われている。CCITT の勧告書の記述にはもちろん SDL が用いられている。ただし、Formal 仕様記述法としてよりも、図式記述による直観的理解をねらったサブドキュメントとして使われている。また SDL 用ツールも各種開発されている。

3.5 ISO とのリエゾン

以前はどちらかというと CCITT は通信関係、ISO は情報処理関係ということで独立に標準化を行ってきた。しかし、最近はプロトコル技術のように通信と情報の区別がなくなり、OSI 通信プロトコルの設計は ISO と CCITT (第 VII 研究委員会) の緊密な連携のもと検討して統一した仕様としている。

また、プロトコル仕様を曖昧性を避けられない自然言語でなく formal な形式記述技法 FDT で記述したいという要求もでてきた。形式記述技法としては、CCITT 勧告言語 SDL の他に ISO の標準化言語 LOTOS, Estelle などがあり、ユーザからは一本化が望まれた。そこで、ISO と CCITT の第 X 研究委員会

が協力してプロトコル仕様記述法一本化の検討を行ったが、3 言語ともにそれぞれの動作モデルが異なり、かつ長所・特徴をもっていることから、適用分野に応じて 3 言語を使い分けるべきであると認識された。このため、ISO と CCITT とが協力してマニュアル：“Estelle, LOTOS 及び SDL の使用ガイドライン”を作成した。本マニュアルには 3 言語の使用法、対応関係、具体的な使用例などが書かれており、形式記述技法の手頃な参考書といえる。このマニュアルは、CCITT 及び ISO から出版される予定である。

ISO と CCITT は、現在も LOTOS の図式記述版である Graphical-LOTOS の設計で協力をしている。

4. プログラミング言語 Chill

4.1 本課題の目的

CCITT では仕様記述言語 SDL の対ともいえるインプリメンテーション言語 Chill (CCITT High Level Language) も勧告している。両者は関連があるので、簡単に Chill の紹介もしておきたい。

交換機のプログラムは、以下の厳しい要求を満たさなければならない。

- 超多重処理（同時に数千の通話が行われる）
- 実時間処理（数 ms の処理遅延でも誤処理となる）
- 超信頼性（システムダウンの許容値は 20 年間に 1 時間）
- 連続運転のまま増設・機能追加を行う
- 厳しい経済性（省メモリで高処理能力）
- ハードウェアを頻繁に制御する
- 数万の加入者回線や中継回線を制御する

このため、交換プログラムの記述言語には一般のプログラム言語よりも一段と厳しい記述能力、省メモリ性、高処理能力性が要求される。

Chill の研究がスタートした 1975 年当時は磁気コアメモリの時代でメモリは非常に高価、プロセッサの処理能力も低かった。このため、交換プログラムはアセンブラ言語を用いて書かれていた。しかし、交換プログラムは頻繁に機能追加され長期にわたって維持管理されることから、プログラムの生産性、保守性、読解性の向上のためにアセンブラ言語に替わる交換プログラム記述用高水準言語が望まれていた。また、マシン依存しない高水準言語を CCITT で標準化しておけば電気通信運用会社・通信機製造メーカーの両者を益することから、CCITT で研究が開始された。

4.2 設計経緯

1975年にChill設計専門家チーム(日本を含む8カ国からなる)が結成され検討が開始された。(当初スイスの働きによりPascalの設計者であるZurich大学のN. Wirth教授が設計チームの顧問になる予定であったが、残念なことに多忙のため参加していただくことはできなかった。Wirth教授はその後Modula, Modula-IIという簡潔で優れたシステム記述言語(大規模・実時間システム開発には機能不足ではあるが)を設計されており、Chillの設計にWirth教授が参加していたら大規模・実時間システム記述言語の決定打として世界のシステム記述言語一本化に寄与したかもしれない。)こうして少数チームによる集約的な検討により、1980年に“Z. 200 Chill: CCITT High Level Language”として勧告化された。

4.3 Chillの概要

Chillは、交換プログラムなどの実時間多重処理システム記述用言語であり、データタイプ概念などはPascalから(構文的にはPL/Iからも)影響を受けている。本言語は、以下のような特色をもつ。

(1) 強力なデータ記述能力

Pascalのデータタイプ概念を拡張して、データ構造記述能力を強化した。また定数記述能力(可変フィールドを含む構造体や配列の定数値など)も強化している。

(2) コンパイル時チェックの強化

Pascal同様のモード種別(Chillではデータタイプのことをモードと呼んでいる)とモードチェックをもつ強タイプ付言語でありモード規則に反する演算は許されないが、明示的モード変換機能を用いれば拡張した演算も可能である。

(3) 大規模システムの開発に適したモジュール化機能

データ宣言と手続き定義のカプセル化機能として、ModulaのMODULE~END同様のモジュール機能をもつ。モジュール外への参照許可はGRANT文、他モジュールへの参照はSEIZE文で明示指定する(それぞれModulaのExport, Importに相当)。これはSDLのもつ構造化機構であるブロックの実現にも適している。

(4) 構造化プログラム向き実行文

適切なフロー制御文をもつ。

(5) 効率的で融通性に富むプロシージャ

融通性のあるパラメータ引継ぎ機構などをもつ。

(6) 並行プロセス機能

交換プログラムは典型的な超多重・実時間処理であるので高水準言語レベルで並列処理記述機能を機自決できることが望ましいこと、コンパイラでエラーチェックをできること、OSに依存しないで並列処理を実現できることなどから、Chillは言語自体に並列処理機能をもたせている。

SDLは、並行動作する実体をプロセスと呼び、プロセスの内部論理を拡張有限状態マシン、システムをシグナルを交信しあうプロセスの集まりとしてモデル化している。Chillの並行プロセス機能は、このSDLモデルの実現に適した以下の特徴を有している。

(a) 軽量・効率的な並行プロセス機能である。いわゆるLight weight processあるいはThreadと呼ばれるものに相当し、効率がよい。SDLのプロセスと同一概念である。

(b) プロセス間の通信機能としては、直接宛先プロセスを指定してメッセージを送るシグナル機能をもっている。本機能は、メッセージの送り元はメッセージを宛先プロセスが待つメッセージ待ち行列に入れて自分はそのまま処理を続行する非同期型通信なので、実時間システムでのプロセス間通信機構として使いやすい。本シグナル機構は、SDLのプロセス間通信のシグナルと大筋では同一であるが、SDLでは受信側プロセスが要求していないシグナルは自動的に破棄されるのに対し、Chillでは破棄されないという点が異なっている。

(c) C.A. Hoareが提案したMonitorの機能(データ定義と手続き定義をキーワード“REGION~END”で囲んだモジュールであり、排他的アクセスが保証される)をもつ。これにより条件付クリティカルリージョンなどが容易に実現できる。

(d) 交換処理では、たとえば発端側イベント、着端末側イベント及びタイムオーバーのうちからいずれかのイベントが発生したら対応する処理を行うといった並列事象待ちの処理が多い。これはSDL仕様記述では複数シグナル待ちの状態として表現される。Chillの並行プロセス機能は、複数事象を待つ最初に着した事象に応答するための並列事象待ち機構をもつので、SDLの複数シグナル待ち状態を素直に記述できる。

(e) 各種の交換機構(シングルプロセッサ構成、共通メモリをもつマルチプロセッサ構成及び共通メモリをもたないマルチプロセッサ構成)を考慮した

```

stack: MODULE;
  GRANT element;
  NEWMODE element = STRUCT(a INT, b BOOL);
  SYN max = 4095, min = 0;
  DCL stack ARRAY(min: max) element;
  DCL index INT INIT := min;
push: PROC(e element) EXCEPTIONS(overflow);
  IF index > max THEN CAUSE overflow; FI;
  stack(index) := e;
  index += 1;
  RETURN;
END push;
pop: PROC( ) RETURNS(element)
  EXCEPTIONS(overflow);
  IF index = min THEN CAUSE underflow; FI;
  index -= 1;
  RETURN stack(index);
END pop;
elem: PROC(i INT) RETURNS(element)
  EXCEPTIONS(bounds);
  IF i < min OR i > index THEN CAUSE bounds; FI;
  RETURN stack(i);
END elem;
END stack;

```

図-3 Chill のプログラム例

機能をもっている。

(7) 使いやすい例外処理機構

使いやすい例外処理機能を持ち、リソース捕捉失敗時などの処理を簡明に記述できる。

Chill のプログラム例を図-3 に示す。

4.4 Chill の実用化状況

Chill は、CCITT しか言語仕様書を発行しておらず一般には名前が知られていない。しかし交換分野では確実に地歩を固めており広く使われている。(日本では D-10, D-60, D-70 等局用交換機など、国外では Siemens の EWSD, ETS, BIGFON, EMS, HICOM 交換機, ITT (現在は Alcatel) の 1240 交換機, Alcatel の E 10 交換機, 5200 BCS などに適用されている。また Chill コンパイラを開発済の機関は数十に上る。) また SDL のモデルとも類似しているため SDL 仕様のインプリメンテーション言語としても適切な機能を備えている。なお、Chill は ISO の標準言語としても承認される予定である。

5. SG-VII での仕様記述関連の検討

CCITT 第 VII 研究委員会 (データ通信を担当) で検討されている仕様記述に関連した話題についても簡単に述べておきたい。

(1) 抽象構文記法 “ASN. 1” (勧告 X. 208: Specification of Abstract Syntax Notation One と勧告

X. 209: Specification of Basic Encoding Rules for Abstract Syntax Notation One; これらは ISO の 8824, 8825 と同一内容の勧告である。)

OSI 環境下で応用プログラムがいろいろのデータを転送しあうには、設計仕様としてのデータの表現方法および実際に転送されるデータのコーディング規則を決めておく必要がある。これは、プログラミング言語のデータタイプ定義に相当するものである。ASN. 1 は従来 X. 409 と呼ばれていたものを X. 208 と X. 209 に分けて規定したものである。X. 208 では、データ構造定義に使用する Abstract Syntax Notation One “ASN. 1” の仕様を (Formal language のシンタックス記述に用いられる Backus-Naur 記法に似た形式で) 規定している。X. 209 では ASN. 1 の転送データでのコーディング規則を述べている。

ASN. 1 に従ったデータ構造記述の一例を以下に示す。

【ASN. 1 によるデータ構造記述の例】

```

PersonnelRecord ::= [APPLICATION 0]
    IMPLICIT SET
    {
        Name,
        title      [0] VisibleString,
        number     EmployeeNumber,
        dateOfHire  Date
        nameOfSpouse Name }
Name ::= [APPLICATION 1] IMPLICIT
    SEQUENCE
    {
        givenName VisibleString,
        familyName VisibleString }
EmployeeNumber ::= [APPLICATION 0]
    IMPLICIT INTEGER
Date ::= [APPLICATION 3] IMPLICIT
    VisibleString
    -- YYYYMMDD

```

【ASN. 1 による上記データ構造の値記述例】

```

{ {givenName "John", familyName "Smith"},
  title "Director",
  number 51,
  dateOfHire "19890401",
  nameOfSpouse
    {givenName "Mary", familyName "Smith"}
}

```

これを以下のように Chill のデータ定義と対比させてみると面白い。

【Chill でのタイプ定義と値の記述】

```

NEWMODE PersonelRecord = STRUCT
  (name STRUCT (givenName, familyName
                                     CHAR (5)),
   title SET (director, manager, supervisor),
   number INT,
   dateOfHire STRUCT (year, month, date INT),
   nameOfSpouse STRUCT (givenName, family
                          Name CHAR (5)
  );
DCL man1 PersonelRecord;
man1 := [ ['John', 'Smith'],
          director,
          51,
          [1989, 4, 1],
          ['Mary', 'Smith'] ];

```

(2) 抽象サービス定義記法 (X. 407)

従来 OSI 各層の仕様は、Peer-to-peer で交換しあうメッセージのフォーマットと手順を規定した『プロトコル』と、レイア間インタフェースを規定した『サービスプリミティブ』の組で規定されていた。しかし、MHS やディレクトリシステムのように分散されている機能要素全体が提供するサービスの規定には、プロトコル+サービスプリミティブによる規定では必ずしも適切でない。そこで MHS 勧告の一環として分散アプリケーションが提供するサービス記述のため『X. 407 抽象サービス定義記法』を規定し、MHS やディレクトリシステムのサービス記述を行っている。

(3) Framework for Support of Distributed Applications (DAF)

通信サービス (分散アプリケーション) は各種サービスが有機的に結合した複合型システムである。このようなシステムの全体の動作を記述するには、個々のアプリケーション要素の動作の規定とともに、その間の相互関係を記述する必要がある。上記 X. 407 はこの目的のために勧告されたものであるが、それを一般化する目的で検討が開始されたのが“Framework for Support of Distributed Applications (DAF)”と呼ばれるオブジェクト指向モデルに基づくモデル化技法である。DAF はまだ検討が開始されたばかりであるが、以下のような方向で検討されており、今後の進展が期待される。

- オブジェクト指向のモデル
- 数学的基盤に基づいた厳密な記述

- オブジェクトの動作は LOTOS で記述
- データ構造は ASN. 1 で記述
- データ構造のセマンティクスは Act-1 で記述
- DAF のモデルのインプリメントを容易にするための基盤技術の整備

6. あ と が き

ソフトウェアの重要性をだれもが唱えているが、その本音は『複雑な要求仕様の実現がすべてソフトウェアに皺寄せされるのだから大変』という見方と、『ソフトウェアの開発は人と時間ばかり食うからどうにかせよ』という見方とがある。

交換・通信関係プログラムは確かに大規模で複雑でありその開発・保守には多大の労力を要しているが、これにはプログラムが実現すべき対象問題が複雑であるという本質がある。つまり、

- 要求仕様がはっきりしていない
- 要求自体が膨大・複雑である

ソフトウェア問題の第一の本質はここにあり、対象問題が複雑な以上それを実現するソフトウェアが複雑になるのはやむをえないと思われる。たとえば交換機のサービスの仕様を日本語で正確完全無欠に記述することは、どれだけ膨大で難しいかを考えてみれば分かる。

仕様記述言語は、この仕様を正確に記述する手段であり、その後の正確なプログラム開発の基本となるものであり、それだけに非常に重要である。またプログラムエラーのかなりの部分は要求仕様のエラーである (本来要求仕様のエラーはプログラムエラーではないが、プログラム開発者が後始末をさせられる) ので仕様書設計段階で完全に仕様チェックを行えばプログラム開発の効率化が計れる。しかし、仕様記述法にも決定打はなく、厳密なモデルに基づいた手法は使いにくく、直観的で分かりやすいものは正確性に欠ける傾向がある。LOTOS が厳密な数学モデルに基づきながら、“比較的” 分かりやすく (といっても LOTOS を使いこなすには相当の数学的素養が必要) 今後の発展に興味が引かれる。

仕様記述法はソフトウェアをアートから工学にするための鍵であり (もっとも、今後も魅惑的なソフトウェアはアートから誕生する) CCITT においても非常に強い関心が注がれている。交換プログラムやプロトコル処理は有限状態マシンとしてモデル化しやすいので、有限状態マシン型仕様記述である SDL はそれな

りに成功を収めてきた。しかし, formal で直観的で自動検証も可能な仕様記述法の決定打はまだ見えておらず, 今後も地道に検討が継続されるであろう。

謝辞 CCITT SG-VI の研究状況に関しては, NTT ネットワーク開発センタの千田昇一氏にいろいろ教えていただきました。ここに感謝いたします。

参 考 文 献

- 1) CCITT 勧告書: Z. 100 SDL: Specification and Description Language, CCITT (1988).
- 2) CCITT: SDL Users Guidelines, CCITT (1988).
- 3) CCITT 勧告書: Z. 200 CHILL: CCITT High

Level Language, CCITT (1988).

- 4) Guidelines for the Application of Estelle, LOTOS and SDL, ISO および CCITT よりマニュアル資料として発行される予定.
- 5) CCITT マニュアル: Z. 200 CHILL Users Manual, CCITT (1988).
- 6) 松尾, 丸山: 交換用プログラミング言語 CHILL, 電気通信協会 (1985).
- 7) CCITT 勧告: X. 208: Specification of Abstract Syntax Notation One, CCITT (1988).
- 8) CCITT 勧告: X. 209: Specification of Basic Encoding Rules for Abstract Syntax Notation One, CCITT (1988).

(平成元年9月1日受付)