Consult (コンパイル編) のシステム化について

3N-8

斉藤　哲*1, 佐藤眞木彦*1, 岡本匡人*1, 小林正和*1, 垂井良平*1,
大西　淳*2,　　島崎眞昭*2
(　*1富士通株式会社, *2京都大学　)

1. はじめに

我々は、FORTRAN のコンパイルエラーに対するプログラム相談システムのプロトタイプを作成中であり、それについて概説する。

なお、プロトタイプは汎用大型機のFACOM M780/30 上で、Prolog, FORTRAN, COBOL、及び画面制御用ユーティリティを用いて実現している。

前回の発表時〔1〕と比較して、対処可能なエラーの数が増えていると共に、利用者インターフェイスが格段に向上している。

2. 処理の手順

相談システムの処理の遷移を、図1に示す。

① 相談システムを起動すると、相談対象のソースプログラムを指定する画面がでる。ここでデータセット名を入力する。区分データセットのメンバ名を指定しないとメンバ名のリストが出て、選択が出来る。選択後、相談システムはコンパイラを起動することによってコンパイルリストを得る。そこからエラーコード、クロスリファレンス、ラベルのリストを読み込み、さらにソースプログラムを読み込む。これらから4章に示す内部データが作成される。

② 次に、エラーコードとエラーメッセージが表示される。相談は、「エラーコードについての質問」というかたちでおこなわれ、利用者は相談したいエラーコードを選択する。

③ 相談システムは、選択されたエラーコードについて原因の推論を行い、その回答と関連したソースプログラムの一部を表示し、利用者の確認を求める。この時、エラーの原因に関係すると判断したソースプログラムのステートメントについては、高輝度で表示し利用者の注意を促す。この部分が2箇所あってかつ一つの画面に納まらない場合は、自動的に画面が分割される。回答の正しさの判断が必要な場合、判断は利用者にまかされている。利用者が「Y」と入力した場合はそのエラーについての相談を終了し、処理②に戻る。「N」と入力した場合は、他の原因について解析を進める。

④ 利用者はコマンドを選択することによって、エディタを呼出し修正を行う。この場合、外部のファイルに収納された相談結果を見ながらエラーの修正を行う。エディタを終了すると処理①に戻り、再びコンパイラを起動し、内部データを全て更新して相談を進める。このようなサイクルを通じてエラーが全て無くなったら、相談システムを終了する。

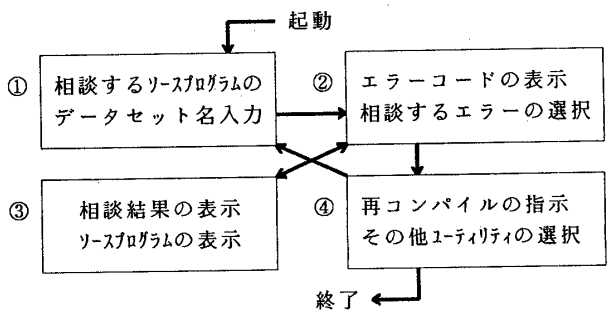

図1　相談システムの処理の遷移

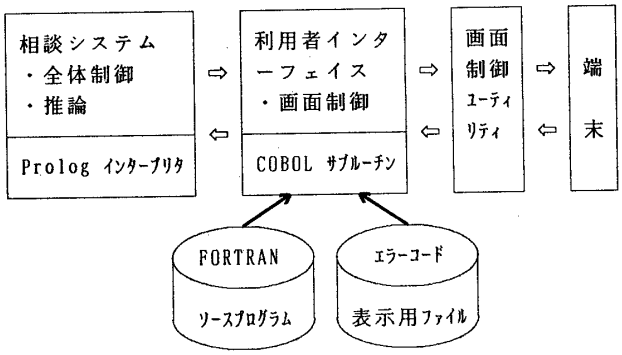

図2　システムと利用者インターフェイス

3. 利用者インターフェイス

前回の発表と大きく異なる点は、利用者インターフェイスの充実である。インターフェイスの実現にあたっては、Prologプログラムから呼び出されるCOBOL による画面の制御プログラムと共に画面制御用ユーティリティを用いている。(図2)

相談システムの全体制御部と推論部は、Prologのインタープリタ上で動作しており、利用者インターフェイスはPrologからサブルーチンとして呼ばれる。利用者インターフェイスは仮想空間に常駐し、充実したインターフェイスを提供するために、相談済のエラーコードを低輝度で表示する、必要な情報を画面を分割して表示する、といった処理を行っている。また、推論部とは別に、表示用としてFORTRAN のソースとエラーコードのファイルを参照している。

4. エラーの推論

エラーを推論する為の内部データは、次のものがある。

CONSULT : PROGRAM CONSULTATION SYSTEM FOR COMPILE-TIME ERRORS

Akira SAITO*1, Makihiko SATO*1, Masato OKAMOTO*1, Masakazu KOBAYASHI*1, Ryouhei TARUI*1,
Atsushi OHNISHI*2, Masaaki SIMASAKI*2
(*1FUJITSU LTD. , *2KYOTO UNIV.)

ⅰ）解析可能な形式にしたソースプログラム
ⅱ）エラーコード，ISN, NAME, LABEL等
ⅲ）変数名のリスト，属性，変数の参照関係
ⅳ）ラベルのリスト，ラベルの参照関係
ⅴ）IF, DO 文のネストに関するテーブル

エラーの推論を行う場合、相談システムは、上記のデータをもとに解析を行い、エラーのパターンにマッチしたものを選び出し、利用者に確認を求める。

一つのエラーコードの解析処理を例に挙げる（図3）。これは、「JZK3001-S ISN NAME: このDO変数はそのDOの入れ子で二重に指定されている」というエラーの推論である。それぞれの項目について、内部に展開した各データを参照して推論を行い、条件を満たすものが選ばれる。

このエラーの推論をもとに、ソースプログラムを解析することで、図4、図5のような画面が端末に表示される。

図4は、ISN=11のDO文に対応する端末文がないために生じたエラーの相談例である。この場合、相談システムはエラーコードからDO変数を抽出し、それとDO文のネストを手掛かりに、エラーの原因となった他のDO文を探す。さらにその端末文についてエラーの原因を探し、図の回答を表示する。この時、関連するDO文とエラーの検出されたDO文は、強調のため高輝度表示される。

図5は、DOループの内部で同一のDO変数を用いたDO型並びを使った時の例である。この場合、相談システムは、図4の場合と同様にDO変数とネストについてチェックする。そして、正しいループの中でDO型並びが使われていることを見出し、図の回答を表示する。

5．おわりに

一般に会話型のシステムでは、利用者インターフェイスが重要である。本システムもさらに充実させて行く予定である。

現在（88年11月末）、対処できるエラーの数は３０であり、この数を増やすと共に質的な向上もはかりながらプロトタイプの評価を進めている。具体的には、初心者が作成したソースプログラムを収集し、テストデータとしている。

懸案であったエディタとの連携は、今回も実現できていない。利用者モデルの研究と実用化も併せて今後の課題である。

謝辞

共同研究関係者、並びに資料収集にあたり協力いただいた方々に感謝致します。

参考文献

〔１〕佐藤他，「FORTRAN のコンパイルエラーに対するプログラム相談の自動化」情報処理学会 第37回全国大会　２Ｋ－３(1988)
〔２〕佐藤他，「プログラム相談プロジェクトConsultの全体構想」情報処理学会第38回全国大会(1989)
〔３〕大西・島崎，「コンパイル時におけるプログラム相談の自動化」電子情報通信学会　技術研究報告 ＡＩ８８－３２(1988)

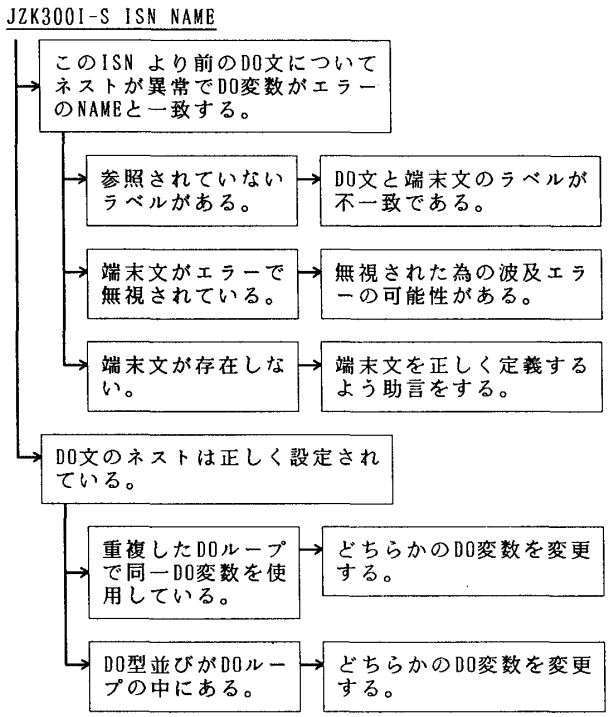

図３　エラー推論の例

```
DO文の端末部( LABEL=70 )がありません。
 10        DO  10  WHILE(INPUT(NEN).NE.EOF)
 11        DO  70  I=1,5
 12        DO  70  J=0,3
 13        MNEN(I,J)=' '
 14        N=NEN
 15        DO  20  I=3,0,-1
 16        IF(NEN.GT.1000) THEN
 17            J=1
 18            DO  30  UNTIL(N.LT.J*10**I)
 43     50 CONTINUE
 44        WRITE(6,300)   (MNEN(I,J),J=3,0,-1)
 45        DO 60 I=1,9,4
 46        WRITE(6,400)   (TSUKI(I+M),M=0,3)
 47        WRITE(6,500)   (CAL(K,J,I+M),J=0,6)
```

図４　JZK3001-S ISN:45 NAME:I の回答例
▒▒ は高輝度表示部分、画面は分割されている

```
DO型並びの変数I が,DO ループの変数と同一です。
 34          J=J+1
 35        ENDIF
 36     30 CONTINUE
 37        IYOBI=J
 38        N(K)=N(K)+1
 39     20 CONTINUE
 40        WRITE(6,100) NEN
 41        DO  40  I=1,3
 42        WRITE(6,200) (N(K),K=1,12)
 43        DO 50 L=1,6
 44        WRITE(6,400) CAL(J,I,K),I=0,6,J=1,6
 45     50 CONTINUE
 46        WRITE(6,500) N(K),K=1,12
 47     40 CONTINUE
```

図５　JZK3001-S ISN:44 NAME:I の回答例
▒▒ は高輝度表示部分