1J-1

FA コントローラプログラミングの複数 DSL 間の相互変換

藤井　春香†　　塩見　彰睦†
静岡大学大学院　情報学研究科†

1. 背景・目的

本研究室では株式会社エヌエスティーとの共同研究として，“スタンドアロン型汎用 FA コントローラ NX-2000” (以下 NX2KS と呼ぶ)の開発に取り組んでいる．

FA (Factory Automation)分野では電気や機械分野のエンジニアにとって理解しやすい，特定分野に特化したドメイン特化言語 (Domain Specific Language 以下 DSL と呼ぶ)と呼ばれるラダー言語や NC コードが長く利用されている．しかし，構造化プログラミング言語と比較してそれら DSL の表現要素は少なく単純なため，近年のソフトウェアの複雑化に伴い可読性が低く巨大なプログラムとなっている．そのため保守性，再利用性が低い問題を抱えている．現在 NX2KS のプログラミングはフローチャートを採用しているが，前述した DSL で記述された多くの設計資産を活用したいという声も寄せられている．本研究では，それら設計資産の再利用を目指して，ラダー言語などの DSL とフローチャートの相互変換の提案と試行を目的とする．

2. 対象 DSL

本研究で対象とする現状の NX2K のフローチャートは処理ノードと分岐ノードで構成される．処理ノードには代入，関数呼び出しなどの処理を記述し，次に処理するノードへ線を記述する．分岐ノードには条件式を記述し，条件を満たす場合と満たさない場合の 2 本の線を次のノードへ記述する．

次にラダー言語について述べる．ラダープログラムは図 1 に示すように左側に入力点を，右に出力点を配置し処理を表現する．入力点と出力点は基本的に BOOL 値である．図 1 の例では，次のように動作する．

(1)母線から ON 信号が供給される．
(2)入力点で構成された論理式(IN1 OR !IN2)AND IN3 が評価される．
(3)論理式の評価結果である，真または偽が OUT1 に出力される．
(4) 論理式の評価結果が真であれば，プログラム内の SUB_ROUTINE ラベルへ処理を移動する．
(5)同様に下に続くラダープログラムを順次実行する．入力点は基本的に BOOL 値であるが，算術演算やデバイスの処理など複雑な演算や処理は FB (Function Block)を用いて表現する．本研究で対象とするラダー言語は PLC プログラミングの世界標準である IEC61131-3[1]の規格を対象とする．

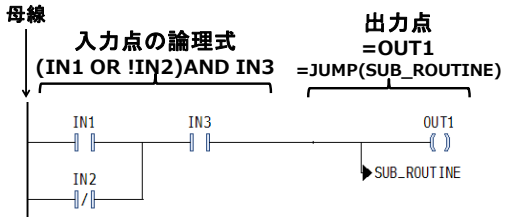

図 1 ラダープログラム

3. 課題

ラダープログラムは図 1 のような単純な処理を繰返し記述するため，単純に変換すると同じ論理式を繰返し用いた冗長なコードとなり得る．例えば図 1 の処理を単純に変換すると表 1 Prog1 のコードだが，表 1 Prog2 の 2 行目の条件式を置き換えて可読性を確保することが望ましい．しかし，置き換える場合は評価が変化しないことを保証する場合のみに適応できる．例えば表 2 の 2 つのプログラムを比較する．開始時に変数 IN が真だとすると，Prog1 は 3 行目のジャンプ命令を実行するが，Prog2 は実行しない．このように置き換えには，置き換えた後に論理式の評価が変化しないことが条件となる．

表 1 ラダープログラム変換結果

	Prog1	Prog2
1	OUT1=(IN1 OR !IN2)AND IN3;	OUT1=(IN1 OR !IN2)AND IN3;
2	if((IN1 OR !IN2)AND IN3)	if(OUT1)
3	{	{
4	GOTO SUB_ROUTINE;	GOTO SUB_ROUTINE;
5	}	}

表 2 置き換えで結果が異なるプログラムの比較

	Prog1	Prog2
1	IN = !IN ;	IN = !IN;
2	if(!IN)	if(IN)
3	{	{
4	GOTO SUB ;	GOTO SUB;
5	}	}

ラダープログラムはリレーシーケンス制御を元とした言語であるため，論理回路に似た表現を得意とする．しかし，構造化プログラミングにおける分岐・繰返しに対応する明確な構文はなく，ジャンプ命令とラベルを用いてそれらを表現する．図 2 にその対応を示す．図 2 のうち，while 構文のラダー言語評価・実行の流れを以下に示す．

(1)条件式を評価し，偽であれば Label2 へジャンプせずに反復処理を実行する．
(2)Label1 へジャンプし再び条件式を評価する．

Program Transformation in FA Controller Programming
†Fujii Haruka　†Shiomi Akichika
†Graduate School of Informatics, Shizuoka University

(3)条件式の評価が真であれば，Label2 へジャンプして while 構文を終了し，次の命令を処理する．

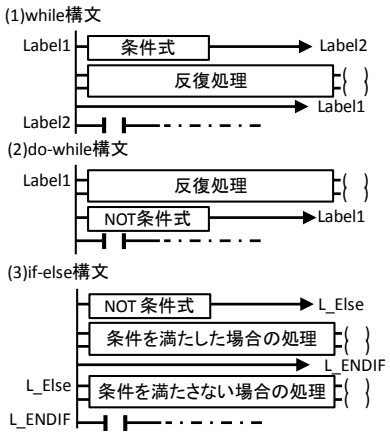

図 2 構造化プログラムとラダー言語での表現

4. 方針

変換時の課題として，プログラムの意味解析を行い，コードの変形を行う点があげられる．そのためには各プログラムを構文解析して AST (Abstract Syntax Tree)を作成し，それに対して意味解析とコード変形を行う必要があるが，この方法を言語ごと適応すると開発コストが大きい．

この問題に対して，先行研究であるソースコードの処理フレームワークである UNICOEN[2]を利用した方法を提案する．UNICOEN は，対応する 7 つの言語の構文表現をすべて持つ統合コードモデル(AST)と，統合コードモデル上の要素を抽出・追加・変更・削除する機能を提供する．UNICOEN を含めた変換システム全体のモジュールとデータフローの構成を図 3 に示す．

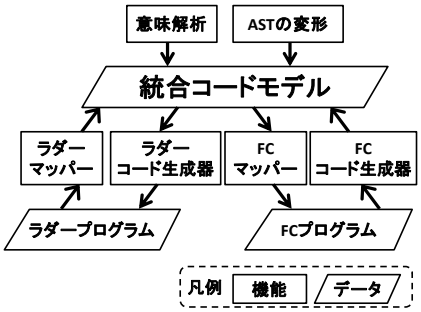

図 3 システムの全体構成

変換の手順は以下の通りである．

(1)プログラム言語のマッパーでプログラムのソースコードを UNICOEN の統合コードモデルの AST にマッピングし，AST を作成する．
(2)AST のみに対して意味解析と変形を行う．
(3)ターゲットコードを生成するコード生成器に AST を入力し，ターゲットコードを出力する．

構文解析後の AST を統合コードモデルにまとめることで，AST に対する意味解析，AST の変形のモジュールが 1 つで済む．

5. 評価

提案したシステムのうち，ラダープログラムからフローチャートへ変換する機能を実装した．図 4 に変換前のラダープログラム，図 5 に変換後のフローチャートを示す．

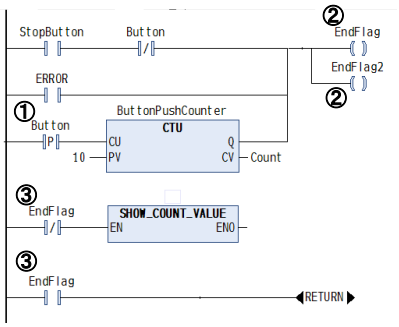

図 4 変換前のラダープログラム

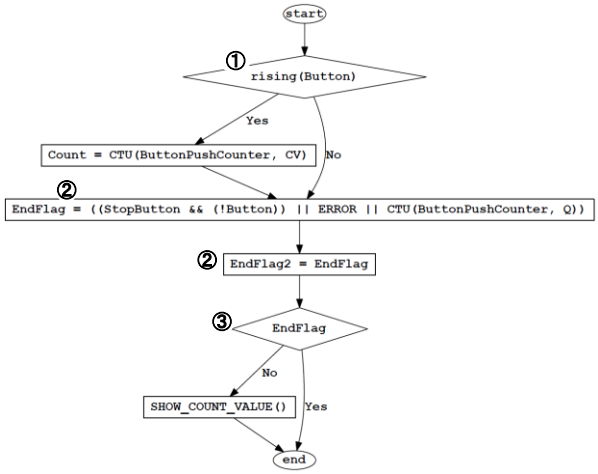

図 5 変換後フローチャート

ラダーマッパーでは，ラダーの各要素と同等の機能を持つ AST にマッピングした．例えば図 4 の①は続く処理が CTU FB であることから，図 5 の① if 文に変換している．

意味解析と AST の変形は以下の点を実装し，変換後の結果に反映されていることを確認した．

(1)重複している論理式を置き換える．図 4 の②から図 5 の②への変換に該当する．
(2)同等の条件式の条件分岐構文を一つにまとめる．図 4 の③から図 5 の③への変換が該当する．

6. おわりに

本研究では UNICOEN の統一コードモデルを利用した DSL 間の変換を提案し，ラダープログラムからフローチャートへ変換できることを確認した．構造化されたプログラムへの変換手法の実現が今後の課題である．

参考文献

[1] K. -H. John, M. Tiegelkamp ほか: “IEC61131-3 を用いた PLC プログラミング”, 丸善, 2012
[2] 坂本 一憲ほか: “UNICOEN:複数プログラミング言語対応のソースコード処理フレームワーク”, 情報処理学会論文誌, pp. 1234-1249, 2008