論理式表現を媒介としたPLCコードデータセットの自動構築

田中宏治[1]　金井健一郎[1]　斉藤辰彦[1]
[1]三菱電機株式会社
{Tanaka.Koji@bc, Kanai.Kenichiro@dy, Saito.Tatsuhiko@db}
.MitsubishiElectric.co.jp

概要

本論文では、産業用制御システムにおける PLC（Programmable Logic Controller）の命令リスト（IL: Instruction List）コード生成を対象に、ローカル環境で動作可能な中小規模言語モデルのファインチューニングに用いる学習データセットを自動構築する手法を提案する。具体的には、文脈自由文法に基づき論理式を生成し、それらを対応する IL コードにルールベースで変換するとともに、その論理式が表す機器動作を LLM を用いて自然言語の動作記述として生成することで、IL コードと動作記述の対からなるデータセットを自動構築する。構築したデータセットを用いて言語モデルをファインチューニングした結果、構文正解率及び動作整合性評価において性能向上を確認した。これにより、複雑なプロンプト設計を必要とせずにローカル環境で高精度な IL コード生成が可能となることを示した。今後は、より多様な IL の命令語に対応し実用性向上を目指す。

1　はじめに

産業用制御システムにおいて、シーケンサ（PLC: Programmable Logic Controller）は製造現場の自動化を支える重要な要素である。シーケンサの動作は主にラダー図や命令リスト（IL: Instruction List）、ST（Structured Text）といったプログラミング言語によって制御されるが、これらのコーディングには専門的な知識を要する。近年、自然言語処理技術の発展に伴い、言語モデルを用いたコード自動生成の研究が進展している。しかしながら、工場現場の多くはインターネット接続が制限されており、クラウド上の大規模言語モデル（LLM）を利用することが困難である。このため、ローカル環境で動作可能な中小規模の言語モデルを用いて、PLC コードを自動生成する技術が求められている。既存のクラウド LLM を用いた IL コード生成の試みでは、三菱電機独自の IL 命令をプロンプトエンジニアリングによりモデルに学習させる方法が報告されている [1]。しかしながら、良好な生成精度を得るためには高度なプロンプト設計が必要であり、その整備には多大な労力がかかる。ローカルモデルにおいても同様に、単にプロンプトを工夫するだけでは限界があると考えられる。そこで本研究では、ローカル言語モデルを IL コード生成に特化してファインチューニングすることを提案する。ファインチューニングには大量の学習データが必要であるが、PLC コードは専門性が高くクローズドな分野であり大規模なデータセットの収集が困難である。これを解決するため、本研究では IL コードの学習用データセットを自動構築する手法を提案する。具体的には、文脈自由文法に基づいて自動生成した論理式を、ルールベースで IL コードに変換する。同時に、同じ論理式から制御機器の動作を表すテキストを LLM を用いて生成する。これにより、IL コードと動作記述テキストの対を自動生成し、ローカル言語モデルのファインチューニングに活用可能なデータセットを構築する。本論文では、提案手法により構築したデータセットを用いてローカル言語モデルを学習し、未学習のモデルとの IL コード生成精度を比較評価する。実験により、提案データセットを用いたモデルが高精度な IL コードを生成することを示す。

2　関連研究

ルールベースによる PLC コード生成　言語モデルを用いないルールベースでの PLC コード生成に関する研究が存在する。[2] では、線形時相論理（Linear Temporal Logic: LTL）による動作記述を自動的に ST へ変換する手法が示されている。[3] では、SysML（System Modeling Language）から形式的仕様と要素間のマッピングルールを定義し、ST へ変換する手法が示されている。これらの手法はルールベースによる正確なコード生成が可能である一方

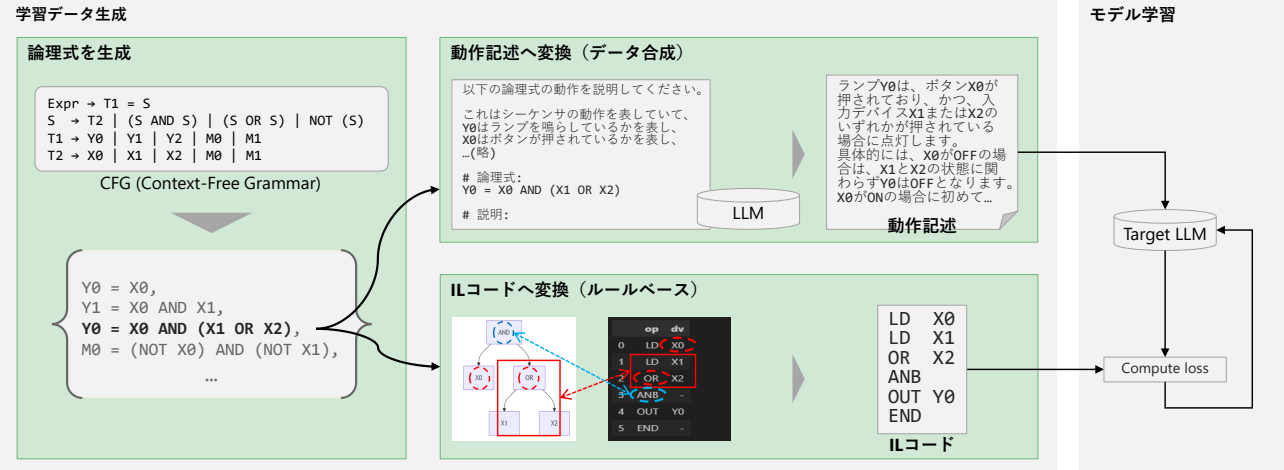

図 1　提案手法の全体像

で、動作仕様を所定の形式表現で詳細に記述する必要があり、記法の学習コストや記述負荷が高い。

LLM による PLC コード生成　近年の LLM によるコード生成のように、動作仕様を自然言語により指示することができれば、前述の負荷が抑えられる。LLM を用いた PLC コード生成の研究として、[1, 4] では、マルチエージェント手法を用いて、自然言語の動作仕様から ST や IL を生成する手法が示されており、生成されたコードの検証エージェントを導入するなどして精度向上を図っている。特に [1] ではメーカ独自の IL 命令やその構文を扱うために、専用命令の情報を In-Context Learning としてプロンプトに与えている。しかし、こうしたプロンプトエンジニアリングは煩雑になりやすくメンテナンスの負荷が高い。こういったドメイン知識は LLM 自身が初めから身に着けていた方が望ましい。また、エージェントシステムはしばしばクラウドベースの LLM で構成されるが、工場現場ではインターネット接続が制限されるケースも多く、クラウド LLM をそのまま運用することが困難であるという実践的な課題が存在する。

ローカルの中小規模の言語モデルを PLC コード生成に向けてファインチューニングする研究も存在し [5, 6]、SFT や DPO により ST 生成タスクの学習を行っているが、小規模な PLC 関連データを元に学習データを作成しており、PLC というクローズドな領域における大規模データセットの構築に課題が残る。

3　提案手法

本研究では、IL コード生成モデルを訓練するための学習データを自動構築する手法を提案する。提案手法では、元手となるデータを全く必要とせずに、動作を表す自然言語の記述と対応する IL コードの対を自動で生成することができる。提案手法の全体像を図 1 に示す。ランダムな論理式を多数生成し、それぞれの論理式を、対応する IL コード及び動作記述へ変換する。同一の論理式から変換された IL コードと動作記述の対を用いてモデル学習することで、入力された動作記述から正しく IL コードを生成するように訓練することができる。論理式の生成は文法さえ定義すればいくらでも可能であるため、現実のデータを全く収集することなしに学習データを作ることができる。以下にそれぞれの詳細を説明する。

論理式生成　文脈自由文法（CFG: Context Free Grammar）により機械的に多数の論理式を生成する。CFG は生成規則に基づいて文を段階的に構成する形式的枠組みであり、命題記号や論理演算子の規則を文法として定義することで、論理式を表現・生成することが可能である。得られた論理式を制御機器の何らかの動作を表現したものであると解釈し、後段で学習データとして用いる表現へと変換する。

IL コードへ変換　論理式を、その論理構造に対応する動作を表した IL コードに変換する。変換は、論理式の木構造を解析するルールベースにより行う。例えば、論理式中に ”(...) AND ((...) OR (...))” という部分がある場合、図 1 中央下段に示すように、対応する IL の記法と部分木の表現を組み合わ

せることで、最終的に論理木全体を IL コードとして表現することができる。

動作記述へ変換　論理式を、その論理構造を制御機器の動作の様子と解釈した場合の自然言語による記述に変換する。変換は、LLM に論理式の動作を説明するよう指示を与えることで動作記述を生成する。このとき、論理式中に含まれる変数が表す機器を仮想的に割り当てて指示に加えることで、機器制御らしい動作記述を得る。

このようにして、同一の論理式を元にして得られた IL コードと動作記述の対は、互いに同じ動作内容をコードと自然言語という別の形式で表現したデータとなる。したがって、動作記述を入力、IL コードを教師ラベルとして言語モデルを学習することができる。

4　実験

提案手法により構築したデータセットで言語モデルを学習し、IL コード生成性能の評価を行う。

4.1　データセット

提案手法において、次の条件でデータセットを構築する。論理式は、`Y0,Y1,M0,M1,X0,X1,X2` の 5 つの原子命題と、`AND,OR,NOT` の 3 つの論理演算からなる論理式を深さの上限が 6 である範囲で CFG により生成する。動作記述への変換では、Gemma3[1] を利用する。IL コードへの変換では、三菱電機の IL 命令のうち、シンプルな命題論理表現との変換が容易な `LD,AND,OR,LDI,ANI,ORI,ANB,ORB,OUT,END` の 10 種の命令から構成される IL コードへ変換する。`LDI,ANI,ORI` はそれぞれ `LD,AND,OR` 演算の後にビット反転（論理否定）を行う命令、`ANB,ORB` は複数行の命令ブロックに対し `AND,OR` 演算を行う命令であり、三菱電機の独自命令である。

このようにして構築された 27,085 件、約 10M トークンの動作記述と IL コードからなるデータセットを、学習・検証・評価用に 8:1:1 の比率で分割して利用する。

4.2　モデル及び学習条件

Phi-4-mini-instruct[2] をベースモデルとして、本データセットを追加学習する。以下の 3 種の条件で学習を行う。

- 継続事前学習（CPT: Continued Pre-Training）：動作記述と対応する IL コードを結合したデータにより因果言語モデリング学習を行う
- 教師ありファインチューニング（SFT: Supervised Fine-Tuning）：動作記述を入力、IL コードを出力とする教師あり学習を行う
- CPT+SFT：CPT を施した後、SFT を行う

学習負荷の軽減のため、いずれも QLoRA[7] によるパラメータ効率学習を行う。学習設定の詳細を付録 A に示す。

4.3　評価指標

性能評価には以下の 2 つの指標を用いる。

- 構文正解率：生成された IL コードの構文の正しさを、構文解析器により評価する。全評価サンプルのうち構文エラーを含まないサンプルの割合として算出する
- LLM 判定スコア：判定用 LLM により、生成された IL コードが動作記述と一致しているかを 10 点満点で評価する。各 IL 命令語の説明と正解の IL コードを参照として与え、動作の整合性の観点で評価する。評価者 LLM として GPT-4.1[3] を利用する

前述の 3 つの学習条件により学習したモデルと、学習を施す前のモデルについて、評価セットの動作記述を入力して生成された IL コードを、これらの指標で評価する。IL コード生成は 0-shot で行うが、未学習モデルについては 2-shot で生成した場合も評価する。2-shot では学習セットの中から論理否定やブロック表現を含むサンプルを使用する。

5　結果と考察

各学習条件における構文正解率と LLM 判定スコアの比較結果を表 1 に示す。

表 1　実験結果

学習条件	構文正解率	LLM 判定スコア
学習なし	0.000 (0-shot)	- (0-shot)
	0.769 (2-shot)	2.767 (2-shot)
CPT のみ	0.969 (0-shot)	9.864 (0-shot)
SFT のみ	1.000 (0-shot)	9.887 (0-shot)
CPT+SFT	1.000 (0-shot)	9.979 (0-shot)

学習を施すことで精度が向上し、特に CPT と SFT の両方を施したモデルは最も高い精度となり、学習

1）`https://ollama.com/library/gemma3:27b`
2）`https://huggingface.co/microsoft/Phi-4-mini-instruct`
3）`https://openai.com/index/gpt-4-1/`

前に比べて構文正解率が 0.231pt、LLM 判定スコアが 7.212 点向上した。2-shot での学習前モデルよりも 0-shot での学習後モデルの方が高精度であったことから、提案手法により、プロンプトエンジニアリングなしに高精度な IL コード生成が可能であることが示唆された。

CPT のみを施したモデルでは構文正解率 0.969pt、LLM 判定スコア 9.864 点と高い値を得たが、SFT のみのモデルではさらに高い値となった。特に構文正解率は 100%であり、これは、SFT が応答部分に絞って学習したことで出力の形式がより強力に学習されたためと考えられる。CPT+SFT の条件ではさらに精度が向上した。CPT により動作記述の部分も学習したことで動作に関する考え方を獲得したためとも考えられるが、すでに上限に近いスコアに至っていて上昇幅は僅かであり、この考えを結論付けるためにはさらなる検証が必要である。

CPT+SFT 条件のモデルの生成内容については、多くが LLM 判定スコアが満点である 10 点であり、正解の IL コードと完全一致しているものがほとんどであった。LLM 判定結果が低スコアであった生成例（図 2）では、否定系の命令（LDI）が誤って挿入されてしまっており、独自命令を正確に扱うのが難しい場合が確認された。これは、アラインメント学習や強化学習など、本データセットを用いてさらに事後学習を施すことで改善が期待できる。また、この例のように、自動生成した論理式に冗長な表現（X2 AND X2 など）が含まれる場合が多く、これを変換して得られた IL コードや動作記述にも冗長で不自然な表現がしばしば含まれていた。このようなデータを学習したことがモデルに悪影響を与えていないかについて検証が必要である。悪影響が確認された場合は、論理式内のこのような表現を事前に修正してから学習データへ変換することでデータの品質を高めていくことができる。

提案手法の制約として、タイマーによるタイミング制御や実数値の比較など、シンプルな論理式で表現しきれない IL 命令のデータは生成できないことが挙げられる。例えば、時間情報を扱う形式表現を導入する、実データも混ぜて学習する、などのアプローチによって本手法を拡張することでこの課題に対応していく。

元となった論理式：

```
M0 = ( ( X2 AND X2 ) OR ( X2 OR X0 ) )
```

入力した動作記述：

この論理式は、シーケンサにおける内部デバイスM0の状態を決定するものです。

式を分解すると、`X2 AND X2` は、センサX2が物体を検出しているかどうかを繰り返し確認する役割を持ちます。論理積なので、X2がオフ（物体未検出）であれば全体もオフとなります。

次に、`X2 OR X0` は、センサX2またはX0のどちらか一方でも物体を検出すればオンとなります。

そして、これらの結果を論理和で組み合わせることで、以下の動作を実現します。

* **X2がオフの場合:** `X2 AND X2` はオフ、`X2 OR X0` はX0の状態に依存しますが、結果的にM0はX0の状態に依存します。
* **X2がオンの場合:** `X2 AND X2` はオンとなり、`X2 OR X0` は常にオンとなります。論理和の特性上、M0は必ずオンとなります。

つまり、センサX2が物体を検出すると、常に内部デバイスM0がオンになり、X2がオフの場合はセンサX0の状態に依存してM0の状態が変化します。`X2 AND X2` は冗長に見えますが、センサX2の信号の安定性を高める、あるいは特定のシーケンサの仕様を満たすために意図的に組み込まれている可能性があります。

正解のILコード：

```
ステップ番号,命令,I/O(デバイス)
0,LD,X2
1,AND,X2
2,LD,X2
3,OR,X0
4,ORB,-
5,OUT,M0
6,END,-
```

生成されたILコード：

```
ステップ番号,命令,I/O(デバイス)
0,LDI,X2
1,LD,X2
2,AND,X2
3,LD,X2
4,OR,X0
5,ORB,-
6,OUT,M0
7,END,-
```

図 2　低スコアであった生成例

6　おわりに

本研究では、産業用制御システムにおける PLC の IL コード生成を目的として、論理式から自動的に IL コードと動作記述を生成する手法を提案した。具体的には、文脈自由文法に基づき論理式を生成し、それをルールベースにより IL コードに変換するとともに、LLM を用いて動作記述を生成することで、大規模な学習データセットを自動構築した。このデータセットを用いてローカルで動作する中小規模の言語モデルを追加学習し、IL コード生成性能を評価した。

実験の結果、提案したデータセットを学習することで、生成された IL コードの構文正解率と動作の正しさを示す LLM 評価スコアが大きく向上した。特に、継続事前学習と教師あり学習を併用したモデルは、学習前と比較して最も高い精度を実現し、プロンプトエンジニアリングに依存しない高精度な IL コード生成の可能性を示した。

これらの結果は、合成データを活用した学習によって、現場での実用性を考慮したローカル環境における PLC コード自動生成の一つの有効なアプローチを提示するものである。今後は、より多様な命令セットや実データの導入を進め、より実用的な自動生成モデルの構築を目指す。

参考文献

[1] 青木隆尚, 安部夏樹, 加羽澤優, 茂呂田美弥, 菅原直樹, 羽藤淳平, 乙村浩太郎. 大規模言語モデルによる plc 向けコード生成. 情報処理学会 第 87 回全国大会. 情報処理学会, 2025.

[2] Chih-Hong Cheng, Chung-Hao Huang, Harald Ruess, and Stefan Stattelmann. G4ltl-st: Automatic generation of plc programs. In **Proceedings of the 16th International Conference on Computer Aided Verification - Volume 8559**, pp. 541–549, Berlin, Heidelberg, 2014. Springer-Verlag.

[3] Bo Ling, Changyong Chu, and Chuan Xu. Automatic plc control logic generation method based on sysml system design model. **Actuators**, Vol. 14, No. 5, 2025.

[4] Zihan Liu, Ruinan Zeng, Dongxia Wang, Gengyun Peng, Jingyi Wang, Qiang Liu, Peiyu Liu, and Wenhai Wang. Agents4plc: Automating closed-loop plc code generation and verification in industrial control systems using llm-based agents, 2024.

[5] Mohamad Fakih, Rahul Dharmaji, Yasamin Moghaddas, Gustavo Quiros, Oluwatosin Ogundare, and Mohammad Abdullah Al Faruque. Llm4plc: Harnessing large language models for verifiable programming of plcs in industrial control systems. In **Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice**, ICSE-SEIP '24, pp. 192–203, New York, NY, USA, 2024. Association for Computing Machinery.

[6] Aaron Haag, Bertram Fuchs, Altay Kacan, and Oliver Lohse. Training llms for generating iec 61131-3 structured text with online feedback, 2024.

[7] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: efficient finetuning of quantized llms. In **Proceedings of the 37th International Conference on Neural Information Processing Systems**, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

A　モデル学習設定の詳細

実験におけるモデル学習の詳細について記す。CPT、SFT ともに QLoRA を表 2 の設定で実施した。

表 2　QLoRA の設定

項目	値
量子化	NormalFloat4
r	16
α	32
lora_dropout	0.05
target_modules	{o, qkv, gate_up, down}_proj
per_device_train_batch_size	8
gradient_accumulation_steps	4
num_train_epochs	10
learning_rate	1e-4
lr_scheduler_type	cosine
warmup_ratio	0.05
optim	paged_adamw_32bit

学習は NVIDIA RTX A6000 (VRAM 48GB) 1 枚で行い、CPT の実行では約 31 時間を要した。

学習データのプロンプトテンプレートを以下に示す。SFT では ”### IL:” 以降の部分のみを学習した。

```
以下の動作を行う IL プログラムを生成します。

### 動作:
{動作記述}

### IL:
{IL コード}
```