УДК 621.865.8

АВТОМАТИЧЕСКОЕ СОЗДАНИЕ ПОСТПРОЦЕССОРА НА ОСНОВЕ ПРИМЕРОВ КОДА

Германович А. П., студент,
Данильчик А. А., студент,
Щеклеина В. П., студент
Белорусский национальный технический университет
Минск, Республика Беларусь
Научный руководитель: старший преподаватель Воюш Н. В.

Аннотация. Рассматривается метод автоматизированного создания управляющих программ для промышленных роботов от различных производителей на основе CAM-системы SprutCAM при помощи автоматической генерации постпроцессоров. Показана практическая значимость архитектуры программного модуля для анализа примеров кода управления конкретным роботом и формирования шаблона постпроцессора SprutCAM.

С целью повышения эффективности разработки управляющих программ для роботизированных комплексов в настоящее время широкое применение получили CAM-системы, позволяющие строить траектории инструмента, моделировать работу робота в симуляции, выбирать режимы и стратегию обработки деталей [1]. Ключевым этапом разработки является постпроцессирование, то есть преобразование внутреннего представления движения (Intermediate Robot Motion Format) в формат непосредственно конкретного производителя. Так как на современных предприятиях часто применяется продукция разных производителей, а также сами производители периодически обновляют свои стандарты, возникает проблема наладки выполнения технологического процесса на каждом роботе в отдельности. Большие затраты времени, а также необходимость изучения документации для создания новых постпроцессоров являются актуальной проблемой, так как несут за собой значительные издержки при организации производства.

Для работы с популярными производителями в пакете SprutCAM есть встроенная система шаблонных постпроцессоров в формате .inp, а также .net, что значительно снижает порог входа для применения программы [2]. Однако, несмотря на развитые возможности программы, разработка нового шаблона требует глубоких знаний о синтаксисе языка программирования конкретного робота.

В качестве решения этой проблемы можно рассмотреть автоматическое создание постпроцессоров на основе примеров кода от производителя, это направление называется Grammar induction [3]. Для этого необходимо создать модуль, задачей которого будет автоматическое извлечение структуры языка примера, формирование шаблона постпроцессора и экспорт файла в формат .inp и .net, таким образом будет упрощена настройка программы под любой тип робота.

Для начала работы необходимо импортировать пример кода со всеми основными командами в виде текстового файла. Далее происходит его синтаксический анализ, для чего каждая строка демонстрационного кода разбивается на токены. Применяем распространенный токенизатор BPE (Byte-Pair Encoding), для него используем открытую библиотеку tokenizers от Hugging Face [4] (рисунок 1). В результате пример разбивается на часто повторяющиеся субслова, которые далее можно обрабатывать. Особенностью токенизаторов BPE является их гибкость, так как при возникновении ранее неизвестных слов программа дробит их на уже известные элементы, тем самым определяет смысл текста.

```python
from tokenizers import Tokenizer
from tokenizers.models import BPE
from tokenizers.trainers import BpeTrainer
from tokenizers.pre_tokenizers import Whitespace
from pathlib import Path

tokenizer = Tokenizer(BPE())     # создаём токенизатор

tokenizer.pre_tokenizer = Whitespace()

trainer = BpeTrainer(
    vocab_size=50,
    special_tokens=["[PAD]", "[UNK]", "[CLS]", "[SEP]", "[MASK]"]
)


corpus_file = Path(r"D:\уник\СТАТЬИ\постпроцессор\corpus.txt")  # полный путь к файлу корпуса


tokenizer.train([str(corpus_file)], trainer)    # обучаем на файле

tokenizer.save("my_tokenizer.json")   # сохраняем
```

Рис. 1. Создание токенизатора

Таким образом, при анализе строки (1) определяются ключевые команды по частотной таблице первых токенов в строках, определяются команды перемещения (линейные, дуги и суставные), а также декларативные команды (frame, tool) и управляющие (GOTO, WAIT).

$$\text{LIN } \{\text{X } 200.0, \text{ Y } 100.0, \text{ Z } 150.0, \text{ A } 0.0, \text{ B } 90.0, \text{ C } 0.0\}. \quad (1)$$

Впоследствии выделяется структура их аргументов (2), строится абстрактная грамматика, которую можно автоматически превратить в правила постпроцессора (2).

$$\text{LIN} \mid \{ \mid \text{X} \mid \text{number} \mid , \mid \text{Y} \mid \text{number} \mid \ldots \}. \quad (2)$$

Таким образом, на основе полученных структур с помощью встроенного шаблонизатора SprutCAM INP можно автоматически построить формат вывода координат, скорости, блоков программы, заголовка. Благодаря этому, создание постпроцессора происходит в полуавтоматическом режиме, что упрощает процесс переноса проектов между роботами и позволяет более эффективно применять CAM-системы.

Список использованных источников

1. What is CAM (Computer Aided Manufacturing)? // GeeksforGeeks. – URL: https://www.geeksforgeeks.org/software-engineering/what-is-cam-computer-aided-manufacturing/ (дата обращения: 15.09.2025).

2. Postprocessor Generator User manual 17 // spruteam. – URL: https://kb.sprutcam.com/docs/Inp/17/en/10001.html (дата обращения: 27.10.2025).

3. Introduction to the Special Topic on Grammar Induction, Representation of Language and Language Learning // Journal of Machine Learning Research. – URL: https://www.jmlr.org/papers/volume12/glowacka11a/glowacka11a.pdf (дата обращения: 28.10.2025).

4. Byte-Pair Encoding (BPE) in NLP // GeeksforGeeks. – URL: https://www.geeksforgeeks.org/nlp/byte-pair-encoding-bpe-in-nlp (дата обращения: 28.10.2025).