АВТОМАТИЗАЦИЯ И УПРАВЛЕНИЕ ТЕХНОЛОГИЧЕСКИМИ ПРОЦЕССАМИ И ПРОИЗВОДСТВАМИ

УДК 62-50:004.057.8

Р.Л. Пушков, С.В. Евстафиева, П.И. Милославский
R.L. Pushkov, S.V. Evstafieva, P.I. Miloslavskiy

ПОСТРОЕНИЕ КОНФИГУРИРУЕМОГО СТАТИЧЕСКОГО АНАЛИЗАТОРА УПРАВЛЯЮЩИХ ПРОГРАММ СИСТЕМ ЧПУ

BUILDING A CONFIGURABLE STATIC ANALYZER FOR CNC PART-PROGRAMS

В работе изложено исследование типов ошибок в управляющих программ современных систем ЧПУ, выявлена возможность возникновения не только синтаксических ошибок, но и потенциальных логических ошибок. Разработана архитектурная модель конфигурируемого статического анализатора управляющих программ, приведен пример его применения для СЧПУ «АксиОМА Контрол».

The paper describes a research of the types of errors in part-programs of modern CNC systems, reveals the possibility of not only syntax errors, but also potential logical errors. An architectural model of a configurable part-program static analyzer has been developed, an example of its application for the «AxiOMA Control» control system is given.

Ключевые слова: управляющая программа, система ЧПУ, синтаксический анализ, статический анализ, ошибки, станок, РБНФ, язык высокого уровня.

Keywords: part-program, CNC, syntax analysis, static analysis, errors, machine-tool, EBNF, high-level language.

Введение

Статический анализ кода — это технология поиска ошибок в программах путем разбора исходного кода и поиска в нем паттернов известных, наиболее распространенных ошибок. Результат работы статического анализатора — это список обнаруженных в коде потенциальных проблем с указанием имени файла и конкретной строки. В большинстве случаев анализ производится над какой-либо версией исходного кода.

Статический анализ актуален не только для языков программирования, но и для управляющих программ. Это становится возможным с развитием языков управляющих программ, появлением в них новых конструкций. В течение большого времени стандартом языка описания траектории движения инструмента и команд управления электроавтоматикой является язык ISO-7bit. Структура языка не предусматривает создания параметрических циклов. Повторное использование участков управляющей программы посредством вызова из другой управляющей программы достаточно ограничено, это часто решается посредством многократного повторения одних и тех же фрагментов программы. Разработчики системы ЧПУ стали включать в свои системы поддержку отдельных синтаксических конструкций, присущих алгоритмическим языкам высокого уровня [1-4]. В результате современные языки управляющих программ стали алгоритмическими языками высокого уровня, а это может требовать дополнительных инструментов анализа.

Еще одна возможность использовать статический анализ для управляющих программ заключается в способности нахождения логических ошибок. Например, в управляющей программе существуют модальные функции, которые могут быть ошибочно введены в одном месте. Использование статического анализатора позволит получить список обнаруженных в программе ошибок с указанием конкретной строки.

Ошибки управляющих программ в СЧПУ

При разработке управляющих программ могут возникать разного рода ошибки.

Группа ошибок перемещений оси. Причина следующей группы ошибок может возникнуть, если на этапе интерполяции конечная точка кадра выходит за предел допустимых перемещений оси. Группа ошибок команд движения. Причина следующей группы ошибок может возникнуть при верификации команды движения на входе интерполятора. Если подача в команде движения задана нулем. Группа ошибок настройки функций. Причина следующей группы ошибок может возникнуть, при программировании в одном кадре двух несовместимых функций, а также при попытке задать постоянный цикл в кадре, содержащем нелинейное движение. Группа ошибок задания контура эквидистанты. Причина следующей группы ошибок может возникнуть, если запрограммирован выход из эквидистантного контура без входа в него. Группы знаковых ошибок. Причина следующей группы ошибок может возникнуть, если знак (-) был задан в команде ЧПУ или в системной переменной, где задание знака минус не разрешено. Так же к потенциальным (логическим) ошибкам относятся: несоответствие типов, выход за пределы массива, выход за пределы размерности типа, использование неинициализированных элементов массива.

Так как современные языки управляющих программ стали алгоритмическими языками высокого уровня, то статический анализ становится для них крайне актуален. Он поможет отследить возникающие группы ошибок, характерных для управляющих программ современных систем ЧПУ, вследствие чего сэкономит большое количество времени оператору для их дальнейшего устранения. В современных языках программирования есть все нужные инструменты, чтобы построить статический анализатор управляющих программ (УП). Вследствие этого можно разработать конфигурированный программный модуль, который позволит применять данный анализатор кода для разных языков систем ЧПУ.

Разработка архитектурной модели статического анализатора кода УП для СЧПУ

В архитектурной модели (рис. 1) статического анализатора кода УП для СЧПУ [5] показаны компоненты статического анализатора и способы их взаимодействия друг с другом. Конфигурационный модуль содержит описание правил языка управляющих программ. Математическим аппаратом в данном модуле является расширенная форма Бекуса-Наура (РБНФ). Модуль лексического анализа предназначен для разбиения входных управляющих программ на описанные токены (минимальная лексическая единица языка УП). Состоит из класса GrammarLexer, который наследует и переопределяет необходимые методы из класса Lexer. Для обеспечения возможности работы с разными языками УП модуль может быть расширен соответствующими правилами. Модуль синтаксического анализа состоит из класса GrammarParser, который наследует и переопределяет необходимые методы из класса Parser. Использует полученные токены и строит синтаксическое дерево разбора для проверки заданной структуры кода управляющих программ. Модуль семантического анализа состоит из класса SemanticAnalysis и интерфейса RegularExpressionConstants, в котором находятся необходимые константы. В нем описываются правила поиска логических ошибок, которые основываются на регулярных выражениях.

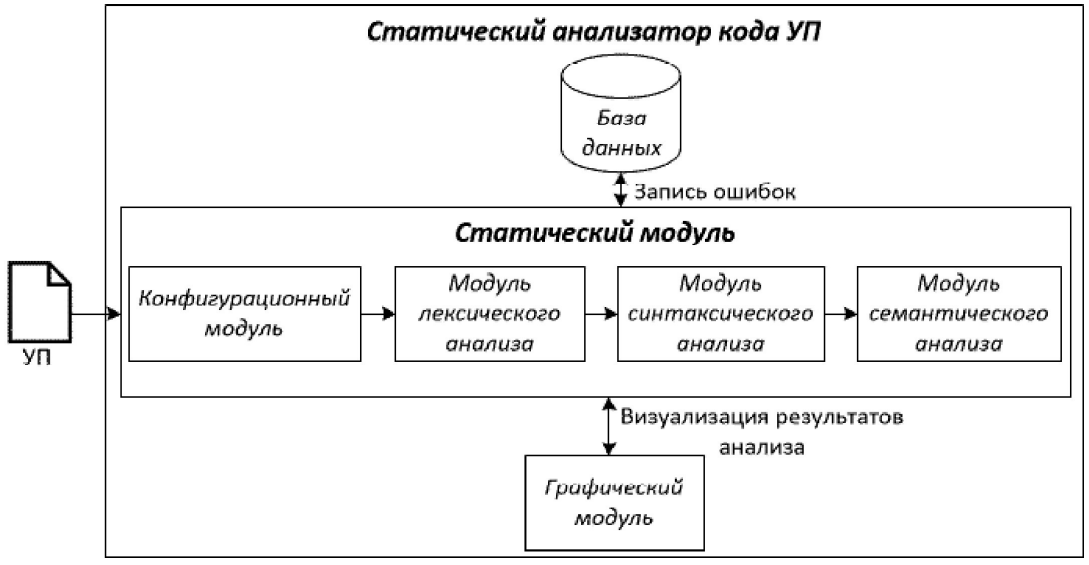

Рис. 1. Архитектурная модель статического анализатора кода

Графический модуль состоит из класса AppFrame. В нем находится главный фрейм (jFrame) на котором размещаются необходимые компоненты, такие как кнопки, текстовые поля, меню-бар (jMenuBar), лейблы (jLabel). База данных в разрабатываемом приложении нужна для хранения различного рода ошибок (логов).

Построение конфигурации для выявления ошибок на примере СЧПУ «АксиОМА Контрол»

Для определения групп потенциальных ошибок необходимо выделить комбинацию слов и кадров языка управляющих программ «АксиОМА Контрол», являющихся допустимыми. Один из возможных вариантов — использование регулярных выражений. Они позволяют произвести поиск возможных ошибок. Далее шаблоны регулярных выражений используются для поиска недопустимых конструкций в управляющей программе.

РБНФ — формальная система определения синтаксиса, в которой одни синтаксические конструкции последовательно определяются через другие. Расширенная форма Бэкуса-Наура необходима для описания контекстно-свободных формальных грамматик [6].

В расширенной форме Бэкуса-Наура для описания грамматики необходим набор правил, определяющих отношения между терминальными символами и нетерминальными символами. Терминальные символы представляют собой минимальные элементы грамматики, не имеющие собственной структуры. Нетерминальные символы представляют собой элементы грамматики, у которых есть собственные имена и структура. Любой нетерминальный символ состоит из одного или более терминальных и (или) нетерминальных символов, представление которых определяется правилами грамматики.

В расширенной форме Бэкуса-Наура нет никаких специальных предписаний относительно порядка записи правил, но такие предписания могут вводиться при использовании РБНФ программными средствами с автоматической генерацией программ синтаксического разбора по описанию грамматики.

Используя анализ синтаксических конструкции языка управляющих программ системы «АксиОМА Контрол» и применяя расширенную форму Бэкуса-Наура, получаем следующий набор формальных правил (показано для нескольких алгоритмических циклов):

<оператор> = (<цикл_for> / <цикл_while> / <ветвление> / <переход> / <присваивание> / <кадр_ISO_7bit> «;»

<присваивание> = <идентификатор> «=» <выражение>

<цикл_for> = «for» «(« <присваивание> «;» <условие> «;» <выражение> «)» «{« *<оператор> «}»

<условие> = (<идентификатор> / <десятичное_число> / <строка> / <символ>) * (<логическая_операция> (<идентификатор> / <десятичное_число> / <строка> / <символ>))

<цикл_while> = «while» «(« <условие> «)» «{« *<оператор> «}»

<ветвление> = «if» «(« <условие> «)» «{« <оператор> «}» [«else» «{« *<оператор> «}»]

<переход> = «goto» <идентификатор>

Далее составленную расширенную форму Бэкуса-Наура, можно реализовать программно. Для этого необходимо использовать фреймворк ANTLR [7], который позволяет использовать расширенную форму в специальном формате .g4 для автоматизации построения программных классов, позволяющих работать с описанными синтаксическими конструкциями.

В результате перевода РБНФ в формат .g4 получается фрагмент программной реализации, представленный на рисунке 2.

Тестирование работы статического анализатора

Для тестирования работы статического анализатора необходимо загрузить в него управляющую программу. Загрузка управляющей программы происходит в главном меню статического анализатора.

После нажатия кнопки «Structure» в статическом анализаторе произойдет проверка структуры, и полученный результат отобразится в нижнем окне вывода (Рис. 3).

После проверки структуры управляющей программы и исправления возможных ошибок можно приступить к поиску потенциально возможных

```
<operator> = (<cycle_for> | <cycle_while> | <branching> | <transition> | <assignment> | <frame_ISO_7bit>).
<assignment> = <identifer> <'='> <expression>.

<cycle_for> = <'for'> <'('> <assignment> <','> <condition> <','> <expression> <')'> <'{'> <operator> <'}'>.
<condition> = (<identifer> | <decimal_number> | <string> | <char>) (<logic_operation> (<identifer> |
              <decimal_number> | <string> | <char>)).
<cycle_while> = <'while'> <'('> <expression> <')'> <'{'> <operator> <'}'>.
<branching> = <'if'> <'('> <condition> <')'> <'{'> <operator> <'}'> <'else'> <'{'> <operator> <'}'>.
<transition> = <'goto'> <identifer>.
```

Рис. 2. Фрагмент программной реализации РБНФ в формате фреймворка ANTLR

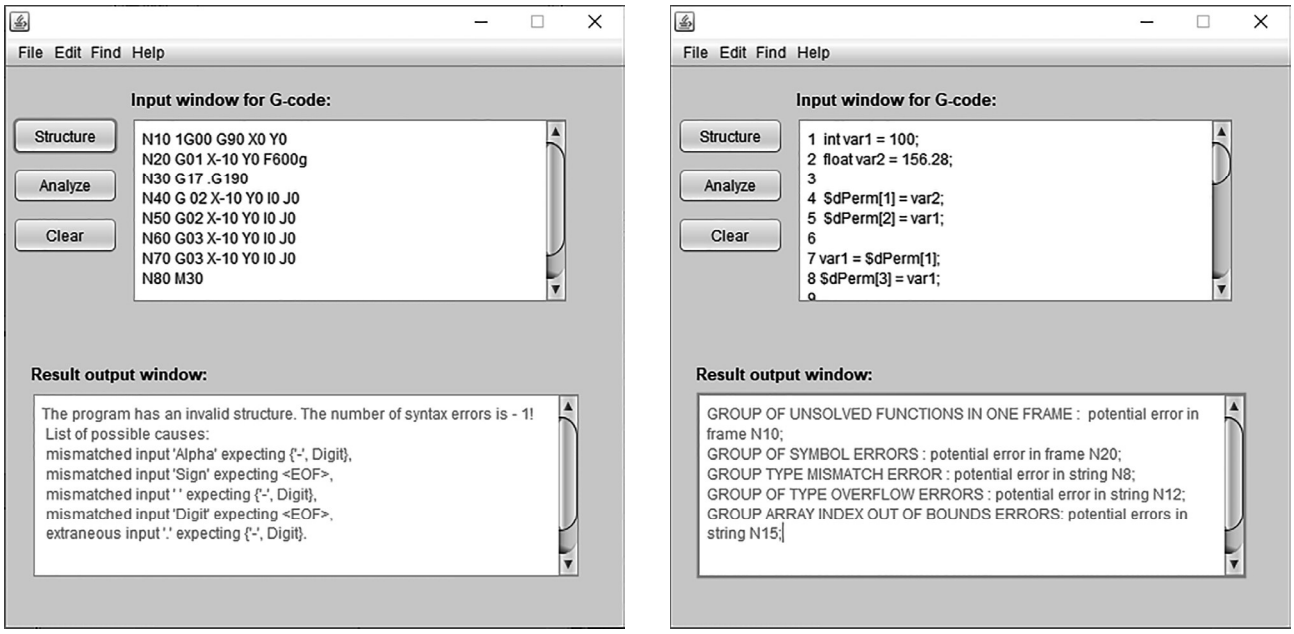

Рис. 3. Результаты анализа при наличии ошибки и статического анализа расширенной программы

ошибок. Они описаны в модуле логического анализа. Для демонстрации возможностей статического анализатора необходимо расширить управляющую программ, добавив в нее переменные различных типов, обращение к системным переменным.

Вследствие анализа в окне вывода отображаются потенциальные ошибки, которые необходимо устранить. В окне вывода приводятся типы ошибок и указание на конкретную строку УП их возникновения.

Выводы

Полученные результаты тестирования позволяют сделать вывод, что примененный к управляющей программе системы ЧПУ метод статического анализа позволяет получить в результате анализа список потенциальных логических ошибок в управляющих программах, на которые оператору или программисту имеет смысл обратить внимание при разработке. На ранних этапах разработки управляющих программ исправление подобных ошибок позволяет ускорить процесс внедрения программы в производство и избежать в дальнейшем финансовых потерь из-за получения брака и простоя оборудования в процессе поиска ошибок в управляющей программе, ставших причиной брака. При расширении системы ЧПУ для управления оборудованием с динамически изменяющейся кинематикой набор необходимых правил для статического анализа УП также может быть дополнен и расширен [8-11].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 20-07-00305/20.

Библиографический список

1. **Евстафиева С.В.**, **Червоннова Н.Ю.**, **Кудинов О.А.** Особенности применения параметрического программирования при создании управляющих программ для системы ЧПУ «АксиОМА Контрол». Труды XVI-ой международной молодёжной конференции «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2016). Под ред. А.В. Толока. М.: ООО «Аналитик». –2016. С. 57–60.

2. **Мартинов Г.М.**, **Пушков Р.Л.** Построение инструментария отладки управляющих программ систем ЧПУ на языках высокого уровня // Приборы и системы. Управление, контроль, диагностика. 2008. № 11. С. 19–24.

3. **Пушков Р.Л.**, **Саламатин Е.В.**, **Евстафиева С.В.** Практические аспекты применения языка высокого уровня в системе ЧПУ для реализации групповой обработки // Автоматизация в промышленности, № 5. 2018. С.31–34.

4. **Пушков Р.Л.**, **Саламатин С.В.**, **Евстафиева С.В.** Применение языка высокого уровня СЧПУ «АксиОМА Контрол» для реализации цикла групповой обработки. Труды XVII-ой международной научно-практической конференции «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2017). Под общ. ред. А.В. Толока. Ин-т проблем упр. им. В.А. Трапезникова. – Электрон. текстовые дан. – М. : ИПУ РАН, 2017. С. 94–98

5. **Мартинов Г.М.**, **Пушков Р.Л.**, **Евстафиева С.В.** Основы построения однокомпьютерной системы ЧПУ с программно реализованным ядром и открытой модульной архитектурой // Вестник МГТУ «Станкин». 2008. № 4. С. 82–93.

6. **Crocker D.**, **Ed.**, **Overall P.** Augmented BNF for Syntax Specifications: ABNF. RFC 5234 (http://tools.ietf.org/html/rfc5234).

7. ANTLR [Электронный ресурс]: офиц. сайт. // ANTLR 4 Documentation: https://www.antlr.org/about.html (дата обращения 27.04.2021).

8. **Мартинов Г.М.**, **Пушков Р.Л.**, **Соколов С.В.**, **Обухов А.И.**, **Евстафиева С.В.** Числовое программное управление станками с динамически изменяющейся кинематикой // Автоматизация в промышленности, № 5. 2020. С. 12-17. DOI: 10.25728/avtprom.2020.05.02

9. **Любимов А.Б.**, **Мартинов Г.М.**, **Мартинова Л.И.**, **Пушков Р.Л.** Построение цифровой управляющей платформы для технологического оборудования с динамически изменяющейся кинематикой // Автоматизация в промышленности, № 5. 2021. С. 3–7. DOI: 10.25728/avtprom.2021.05.01

10. **Martinov**, **G.M.**, **Sokolov**, **S.V.**, **Pushkov**, **R.L.**, **Obukvov**, **A.I.**, **Evstafieva**, **S.V.** Control of the machine tools with variable kinematics. Int J Adv Manuf Technol (2021). https://doi.org/10.1007/s00170-021-07339-1

11. **Martinov G.**, **Kozak N.** and **Evstafieva S.** Implementation of Dynamic Changes in Machine Kinematics in the Electroautomatic Subsystem of the CNC System. 2021 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2021, pp. 596–601, doi: 10.1109/ICIEAM51226.2021.9446459.

Пушков Роман Львович – канд. техн. наук, доцент кафедры компьютерных систем управления ФГБОУ ВО «МГТУ «СТАНКИН»
pushkov@ncsystems.ru

Евстафиева Светлана Владимировна – старший преподаватель кафедры компьютерных систем управления ФГБОУ ВО «МГТУ «СТАНКИН»
svetlana.evstafieva@gmail.com

Милославский Павел Игоревич – магистрант группы АДМ-19-04 кафедры компьютерных систем управления ФГБОУ ВО «МГТУ «СТАНКИН»
swon-more@mail.ru

Pushkov Roman Lvovich – candidate Sc. of Engineering, assistant professor of Department of Computer-Architecture Control Systems of MSUT «STANKIN»
pushkov@ncsystems.ru

Evstafieva Svetlana Vladimirovna – lecturer of Department of Computer-Architecture Control Systems of MSUT «STANKIN»
svetlana.evstafieva@gmail.com

Miloslavskiy Pavel Igorevich – master degree student group ADM-19-04 of Department of Computer-Architecture Control Systems of MSUT «STANKIN»
swon-more@mail.ru