Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение   
высшего образования

”Алтайский государственный технический университет им. И. И. Ползунова”

Факультет информационных технологий

наименование подразделения

Кафедра информатики, вычислительной техники

и информационной безопасности

наименование кафедры

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.Ю. Тырышкин

(подпись руководителя от вуза) (инициалы, фамилия)

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

ОТЧЕТ

о производственной\_практике (научно-исследовательской работе)

(вид практики)

в АлтГТУ им. И.И. Ползунова

(название предприятия, организации, учреждения)

ПрП\_09.04.01.04.000\_ОТ

(обозначение документа)

Студент группы \_\_8ИВТ-71\_\_\_С. В. Горских

(инициалы, фамилия)

Руководитель практики

от предприятия АлтГТУ им. И. И. Ползунова д.т.н. профессор Л.И. Сучкова

(должность, ученое звание) (инициалы, фамилия)

Руководитель практики от вуза доцент каф. ИВТ и ИБ С.Ю. Тырышкин

(должность, ученое звание) (инициалы, фамилия)

Барнаул 2019

ФГБОУ ВО «Алтайский государственный технический университет

им. И. И. Ползунова»

Кафедра «Информатика, вычислительная техника и информационная безопасность»

**Индивидуальное задание**

на производственную практику

(научно-исследовательская работа) студенту 2 курса С.В. Горских группы 8ИВТ-71

Профильная организация: АлтГТУ им. И.И. Ползунова

Сроки практики: 14.01.2019 г. - 23.05.2019 г.

Тема: Разработка синтаксически ориентированной программной среды для клеточно-автоматного моделирования.

**Рабочий график (план) проведения практики:**

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **п/п** | **Содержание раздела (этапа) практики** | **Сроки выполнения** | **Планируемые результаты практики** |
| 1 | Получить индивидуальное задание на практику.  Пройти инструктаж по охране труда и технике безопасности.  Завершить аналитический обзор аналогичных отечественных и зарубежных разработок в области информатики и вычислительной техники по теме магистерской диссертации.  Сформулировать цель и задачи исследований в диссертации  Описать методы, применяемые для проектирования, разработки и исследования объектов профессиональной деятельности в рамках магистерской диссертации. Проанализировать существующие алгоритмы нечеткого вывода.  Написание 1 главы диссертации | 1-4 недели | Формирование части «уметь» компетенции ОК-5: использование результатов освоения фундаментальных и прикладных дисциплин магистерской программы для решения задач в своей предметной области, самостоятельное решение научных, исследовательских, проектных и технологических задач в области информатики и вычислительной техники, составление плана проведения практики  Формирование частей «знать» и «уметь» компетенции ОК-7: поиск информации по теоретическим и экспериментальным исследованиям по выбранной теме, применение знаний и умений в решении конкретных практических задач.  Базовое формирование компетенции ПК-2: применение навыков проведения теоретического и экспериментального исследования к выбранным объектам профессиональной деятельности, подготовка научной публикации к печати |
| 2 | Обосновать и описать программно-технические средства, используемые для решения задач магистерской диссертации  Написание 2 главы диссертации | 5-10 недели | Базовое формирование компетенции  ОК-8: закрепить навыки инсталляции, диагностики и тестирования оборудования и программного обеспечения при выполнении задания по НИР, закрепить навыки работы на современном оборудовании и в средах разработки, применяемых для разработок, исследований и производства программно-технических разработок в выбранной предметной области |
| 3 | Доработка программно-аппаратного обеспечения согласно требованиям эксперимента и технического задания, работы по настройке, тестированию и опытной проверке программно-технических средств, требуемых для выполнения исследований по теме диссертации: язык программирования C++, среда разработки Qt Creator.  Написание 3 главы диссертации | 11-15 недели | Базовое формирование компетенций: - ПК-7: применение перспективных методов исследования и решения профессиональных задач на основе знания мировых тенденций развития вычислительной техники и информационных технологий;  - ПК-10: способность разрабатывать и реализовывать планы информатизации предприятий и их подразделений на основе современных технологий (для внедрения и опытной эксплуатации результатов диссертации);  - ПК-11:участвовать в разработке аппаратных и/ или программных средств вычислительной техники, необходимых для достижения цели магистерской диссертации |
| 4 | Оформить и сдать отчет по НИР. В отчете отразить полученные результаты | 16-18 недели | Базовое формирование компетенции  ОК-9: оформление отчета по результатам проведенной научно -исследовательской работы и подготовка публикаций по ре-зультатам исследования |
| |  |  | | --- | --- | | Руководитель практики от вуза \_\_\_\_\_\_\_\_\_\_\_\_\_Тырышкин С.Ю.,­ к.т.н., доцент каф. ИВТиИБ  (подпись) (Ф.И.О., должность)  Руководитель практики от  профильной организации Сучкова Л.И­. д.т.н., профессор, проректор  (подпись) (Ф.И.О., должность)  Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горских С.В.  (подпись) (Ф.И.О.) |  | | | | |  |

**Инструктаж по ОТ, ТБ, ПБ, ПВТР**

Инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового распорядка проведен «14» января 2019 г.

Руководитель практики от

профильной организации\_\_\_\_\_\_\_\_\_\_\_ Сучкова Л.И.\_,проректор, д.т.н., профессор (подпись)

**Содержание**

[**Введение** 7](#_Toc9879673)

[**1** **Обзор систем клеточно-автоматного моделирования** 8](#_Toc9879674)

[**1.1** **Понятие клеточно-автоматной модели** 8](#_Toc9879675)

[1.2 Существующие системы клеточно-автоматного моделирования 9](#_Toc9879676)

[1.2.1 Система имитационного моделирования WinALT 9](#_Toc9879677)

[1.2.2 CAME&L 9](#_Toc9879678)

[1.2.3 SoftCAM 9](#_Toc9879679)

[1.2.4 Golly 9](#_Toc9879680)

[**2** **Проектирование синтаксически-ориентированной среды клеточного-автоматного моделирования** 10](#_Toc9879681)

[2.1 Разработка синтаксиса описания клеточно-автоматной модели 10](#_Toc9879682)

[2.1.1 Описание клеточных автоматов 10](#_Toc9879683)

[2.1.2 Системы параллельных подстановок 15](#_Toc9879684)

[2.1.3 Параллельные подстановки 16](#_Toc9879685)

[2.1.4 Контекстные подстановки 18](#_Toc9879686)

[2.1.5 Функции переходов 18](#_Toc9879687)

[2.1.6 Внешние воздействия 18](#_Toc9879688)

[2.2 Проектирование среды моделирования 18](#_Toc9879689)

[2.2.1 Проектирование интерпретатора 18](#_Toc9879690)

[2.2.2 Проектирование UI - 18](#_Toc9879691)

[**3** **Программная реализация алгоритма отсечения вариантов при нечетком выводе** 19](#_Toc9879692)

[3.1 Структура интерпретатора 19](#_Toc9879693)

[3.1.1 Лексический анализатор 19](#_Toc9879694)

[3.1.2 Синтаксический анализатор 19](#_Toc9879695)

[3.1.3 Модуль семантического анализа 19](#_Toc9879696)

[3.1.4 Подпрограммы контроля интерпретации 19](#_Toc9879697)

[3.2 Компонент визуализации 19](#_Toc9879698)

[**Заключение** 20](#_Toc9879699)

[**Список использованных источников** 21](#_Toc9879700)

**Введение**

Типа введение.

Промежуточные результаты данной работы были представлены на конференции «Наука и молодёжь» в 2019 году.

# **Обзор систем клеточно-автоматного моделирования**

* 1. **Понятие клеточно-автоматной модели**

Влупим часть с старой диссертации часть с моей практики.

* 1. Существующие системы клеточно-автоматного моделирования
     1. Система имитационного моделирования WinALT

Лол какая система.

* + 1. CAME&L

О даааааа, кэмел

* + 1. SoftCAM

Лол кек

* + 1. Golly

Алгоритм.

# **Проектирование синтаксически-ориентированной среды клеточного-автоматного моделирования**

Среда клеточно-автоматного моделирования состоит из следующих основных компонентов:

* Клиентский интерфейс для описания клеточно-автоматных моделей на спроектированном синтаксисе;
* Модуль визуализации процесса моделирования;
* Интерпретатор для спроектированного языка;
  1. Разработка синтаксиса описания клеточно-автоматной модели
     1. Описание клеточных автоматов

Предполагаемый язык является Си-подобным интерпретируемым языком и основан на математической модели «Алгоритм Параллельных Подстановок» АПП, позволяющий описывать формально КА-модели. Условно, синтаксис языка можно разделить на две составляющие – синтаксические конструкции языка Си и синтаксис описаний КА, основанный на модели АПП.

Текст программы пользователя может состоять из объявлений или определений данных, описаний функций, описаний клеточных автоматов. Также допускаются описания пользовательских типов, таких как структуры. В качестве данных могут выступать переменные, константы, статические и динамические массивы, которые могут быть стандартного или пользовательского типа, а также специализированные КА-типы.

Язык допускает описание идентификаторов, которые могут состоять из букв английского алфавита, цифр, символов подчеркивания и всегда начинаются с буквы. Комментарии могут быть однострочными и многострочными.

Типы данных, поддержваемые данным языком представлены в таблице 2.

Таблица 2.1 – Типы данных

| **Тип** | **Краткое описание** |
| --- | --- |
| Стандартные типы данных | |
| int | Целый (4 байта) |
| long | Целый (8 байт ) |
| bool | Булев (1 байт) |
| double | Вещественный с двойной точностью (8 байт) |
| void | - |
| string | Строковый (переменный размер) |
| Специализированные типы данных | |
| cell | Указатель на клетку (4 байта) |
| classArray | Клеточный массив (переменный размер) |
| neig | Шаблон соседства (переменный размер) |

В таблице 2.2 приведен список поддерживаемых операторов языка Си. Помимо представленных операторов язык содержит и некоторые собственные операторы.

Таблица 2.2 – Список поддерживаемых операторов языка Си

|  |  |
| --- | --- |
| **Оператор** | **Краткое описание** |
| [выражение]; | Оператор «выражение» |
| while | Цикл с предусловием |
| for | Цикл с параметром (for) |
| if/else | Условный оператор if |
| break | Немедленное завершение цикла |
| continue | Переход к следующей итерации цикла |
| return | Возврат из функции |

В таблице 2.3 представлен список поддерживаемых унарных, бинарных и тернарных операций языка Си.

Таблица 2.3 – Список поддерживаемых операций языка Си

| **Операция** | **Краткое описание** |
| --- | --- |
| Унарные операции | |
| ++ | Инкремент |
| -- | Декремент |
| ! | Логическое отрицание |
| + | Унарный плюс |
| - | Унарный минус (арифметическое отрицание) |
| new | Выделение памяти |
| delete | Освобождение памяти |
| Бинарные и тернарные операции | |
| \* | Умножение |
| / | Деление |
| % | Остаток от деления |
| + | Сложение |
| - | Вычитание |
| < | Меньше |
| > | Больше |
| == | Равно |
| <= | Меньше или равно |
| >= | Больше или равно |
| != | Не равно |
| && | Логическое И |
| || | Логическое ИЛИ |
| = | Присваивание |
| . | Операция доступа |

Рассмотрим более подробнее реализацию функциональных возможностей языка по описанию клеточных автоматов.

Для клеточного автомата требуется клеточное пространство, на котором он моделируется. Клеточные пространства описываются с помощью специальных классов клеточных массивов, имеющих тип classArray. Ниже представлен синтаксис описания класса клеточных массивов:

*classArray имя <тип\_решетки, форма\_пространства>*

*{*

*поля\_данных*

*шаблоны соседства*

*};*

*тип\_решетки* – структура решетки, на которой описывается класс. Тип решетки задает количество измерений пространства и структуру соседства;

*форма\_пространства* – задает граничные условия. К примеру, клеточное пространство может задаваться формой тора;

*поля\_данных* – элементы данных, расположенные в узлах решетки. Другими словами, это описание переменных (частный случай), которые будут хранить значения клеток, связанных с ними;

*шаблоны соседства* – объявление шаблона соседства, определяющего набор соседних клеток, которые будут задействованы в вычислении очередного значения клетки.

Ниже представлен листинг кода описания класса клеточного массива:

*classArray SimpleClassArray <CA.Lattice.HEXAGONAL\_2D,CA.Surface.FLAT>*

*{*

*bool isAlive;*

*neig template = {#[0], #[1], #[2], #[1, 2], #[1, 3]};*

*};*

В классе SimpleClassArray, который представляет двумерную квадратную сетку с граничными условиями типа FLAT ( противоположные грани соединяются друг с другом), определено поле клетки типа bool, хранящее определенную информацию, а также шаблон соседства template, состоящий из пяти элементов, причем первый элемент указывает на центровую клетку, а последний указывает на соседа через одну клетку (сосед соседа).

Для создания клеточного массива определенного класса клеточных массивов используется следующий синтаксис:

*имя\_класса имя = new имя\_класса(размеры\_клеточного\_массива);*

Где *имя\_класса* – имя класса клеточных массивов, описанного пользователем, *размеры\_клеточного\_массива* – численное значение размеров измерений создаваемого клеточного массива.

Ниже представлен пример кода объявления и инициализации клеточного массива на 2D пространстве с размером 500 на 500 клеток:

*SimpleClassArray simpleArray = new SimpleClassArray(500, 500);*

Синтаксис описания непосредственно клеточного автомата представляет собой описание функции, которая по заданным правилам преобразует данные в указанных клеточных массивах. Она содержит в себе описание систем параллельных подстановок, а также может содержать описание контекстных подстановок и функций переходов. Клеточный автомат может принимать параметры, а также возвращает значение типа integer ­– число завершенных глобальных итераций эволюции. Описание клеточного автомата должно содержать, по крайне мере, одну систему подстановок, которая всегда идет в описание первой. Клеточные автоматы работают только с теми клеточными массивами, которые были переданы в функцию в качестве параметра. Ниже представлен синтаксис описания клеточного автомата:

*CellAuto имя (список\_параметров)*

*{*

*вектор\_внешних\_воздействий*

*системы\_подстановок*

*[контекстные\_подстановки]*

*[функции\_переходов]*

*}*

*список\_параметров* – список клеточных массивов, с которыми будет работать клеточный автомат;

*системы\_подстановок* – системы параллельных подстановок. См. «2.1.2 Системы параллельных подстановок»;

*контекстные подстановки* – см. «2.1.3 Контекстные подстановки»;

*функции\_переходов* – см. «2.1.4 Функции переходов»;

*вектор\_внешних\_воздействий* – см. «2.1.5 Внешние воздействия»

Ниже представлен листинг кода, демонстрирующий описание клеточного автомата с именем SimpleCAЮ, который имеет единственный параметр – клеточный массив класса SimpleClassArray. Клеточный автомат состоит из системы параллельных подстановок S1 с единственной подстановкой PP1, при помощи которой каждая клетка массива SCA сможет обменяться состоянием (переменной isAlive) с левой клеткой-соседом.

*CellAuto SimpleCA(SimpleClassArray SCA)*

*{*

*System : S1*

*{*

*PP1: (var x1, SCA.isAlive), (var x2, SCA.template[0].isAlive)* ->

*(x2, SCA.isAlive), (x1, SCA.template[0].isAlive);*

*}*

*}*

* + 1. Системы параллельных подстановок

Система параллельных подстановок представляет собой множество правил переходов клеточного автомата и может состоять из одной или нескольких параллельных подстановок. Подстановки из одной системы постановок применяются последовательно к каждой клетке указанного клеточного множества и, в конечном счете, после применения всех подстановок ко всем клеткам, образуют выполнения глобального оператора.

Каждая система подстановок допускает описание подстановок, которые могут применяться к клеточным массивам различных классов. При этом стоит учитывать, что система подстановок может выполняться только на единственной области значений имен клеток, которая указывается при объявлении системы подстановок. По этой причине, клеточные массивы, к которым применяются подстановки из одной системы параллельных подстановок, должны быть одной мерности и одного размера (это не относится к контексту). В противном случае, корректная работа клеточного автомата не гарантирована, и может привести к критическим ошибкам памяти. Ниже представлен синтаксис описания системы параллельных подстановок:

System [имя\_системы] : указатель\_области\_значений [определяющий\_шаблон]

{

параллельные\_подстановки

}

Рассмотрим составные части подробнее:

*имя\_системы* – имя описываемой системы параллельных подстановок, которое допускается не указывать, если оно не требуется;

*указатель\_области\_значений* – указывает область значений имен клеток, которая будет использоваться в подстановках данной системы. В качестве указателя может выступать клеточный массив или клеточная область.

*параллельные\_подстановки* – список параллельных подстановок системы (см. «2.1.3 Параллельные подстановки»);

*определяющий шаблон* – необязательный параметр, который используется при задании блочно-синхронных клеточных автоматов.

* + 1. Параллельные подстановки

Каждая параллельная подстановка состоит из базы и правой части подстановки, а также может содержать контекст. В свою очередь база, правая часть подстановки и контекст состоят из описаний клеток базы, правой части и контекста соответственно.

Подстановки также могут выполняться в циклах, при этом все итерации цикла выполняются на одном подтакте последовательно и не считаются по отношению друг к другу параллельными. Контекст подстановки вычисляется на каждой итерации цикла. Следует также помнить, что при синхронном режиме работы клеточного автомата, значения клеток будет изменено только после применения всей системы подстановок ко всем клеткам. По этой причине на каждой итерации цикла клетки будут иметь свое первоначальное значение.

Каждая подстановка заканчиваются точкой с запятой и должна быть описана раньше функций переходов и контекстных подстановок, применяемых в данной подстановке. Контекст в подстановках выполняется в первую очередь. Если хоть один предикат в контексте ложен, то данная подстановка не будет выполнена. Ниже представлен синтаксис описания параллельной подстановки:

имя: база [ & контекст ] –> правая\_часть [~ циклы]

Рассмотрим подробнее описание параллельной подстановки:

*база* – определяет локальную конфигурацию, клетки которой учавствуют в подстановке. Локальная конфигурация задается клетками базы, перечисленными через запятую;

*контекст* – обычно исполняет роль предиката, но также может быть использован для вычисления конкретного индекса из области значений индексов клеток (пространственный контекст), или играть роль вероятностного срабатывания подстановки (вероятностный контекст). Контекстом могут быть как контекстные клетки (вызовы контекстных подстановок), так и клетки клеточных массивов, которые были переданы в клеточный автомат в качестве параметров. Элементы контекста разделяются запятыми;

*правая\_часть* – должна соответствовать локальной конфигурации базы.

Синтаксис описания клетки базы представлен ниже:

([ var ] имя, поле\_клетки\_базы)

*поле\_клетки\_базы –* указатель на поле клеточного массива. Нельзя использовать клетки клеточного массива, который не был явно указан в принимаемых клеточным автоматом параметрах.

*[var] имя* – указание модификатора var при описании означает, что будет объявлена локальная переменная с именем имя, тип которой выводится в соответствии с типом указанного поля клетки. При выполнении подстановки данная переменная будет содержать значение соответствующего поля. Если ключевое слово var отсутствует , то клетка базы играет роль предиката, а в качестве имени можно указать выражение, локально описанную функцию, константу или переменную, значения которых будут сопоставляться со значением исползуемого поля клетки. Предикаты базы проверяются после анализа контекста, и в случае, если хоть один из них ложен – подстановка не будет выполнена.

Синтаксис описания клетки контекста:

([ var ] имя, поле\_клетки\_контекста)

Рассмотрим составные части описания клетки контекста подробней:

*поле\_клетки\_контекста –* поле клетки клеточного массива, имя переменной, либо имя контекстной подстановки (см «2.1.4 Контекстные подстановки»). Нельзя указывать клетки клеточного массива, который не был явно передан в клеточный автомат в качестве параметра. В отличие от базы, здесь не допускается указывать поле клетки, которое является структурой или массивом (а не их конкретным элементом);

*[ var ] имя –* аналогично описанию базы.

Синтаксис описание клетки правой части:

*(значение, поле\_клетки\_правой\_части)*

Рассмотрим составные части описания клетки правой части подробнее:

поле\_клетки\_правой\_части – локальная конфигурация правой части подстановки должна полностью соответствовать локальной конфигурации базы;

значение – переменная, значение которой при выполнении подстановки будет присвоено указанному полю клетки. Также допускается указывать выражения или вызовы функций переходов. Тип поля клетки и тип значения должны совпадать. В отличие от базы и контекста, здесь не допускается указывать модификатор var. Если полем клетки является массив или структура (а не их конкретные элементы), то в качестве значения может быть только функция переходов, которая возвращает значение типа void. В этом случае изменять поле клетки следует непосредственно в теле указанной функции переходов, корректность чего полностью возлагается на плечи пользователя.

Синтаксис описания циклов:

*for (инициализация; выражение; модификация)*

Описание циклов аналогично описанию циклов for в языке С/С++. Указание оператора цикла означает, что подстановка будет выполняться в цикле по всем правилам языка С/С++. Подстановка может иметь несколько операторов цикла, разделенных запятыми. В этом случае циклы стоящие правее являются вложенными по отношению циклов стоящих левее.

Ниже представлен листинг кода, демонстрирующий описание параллельной подстановки с циклом:

CellAuto SimpleAuto(SimpleClassArray ARR, SimpleClassArray OTH)

{

System: sys

{

PP1: (var x1, ARR.#[i].f), (var y1, OTH.#[i].v) &

(true, cnt) –>

(y1, ARR.#[i].f), (x1, OTH.#[i].v) ~

for(int i = 0; i<8; i++);

}

(bool, cnt)

{

if (randf() > 0.5) return true;

else return false;

}

}

Подстановка с именем PP1 имеет базу, локальная конфигурация которой состоит из двух клеток, а также контекст (в виде вызова контекстной подстановки cnt), который выполняет роль предиката. Подстановка выполняется в цикле, который позволяет клеткам из массивов ARR и OTH обменяться значениями клеток из своих локальных конфигураций.

* + 1. Контекстные подстановки

Ранее уже было отмечено, контекст в подстановках может играть роль предигата, служить для вероятностного срабатывания подстановки, а также использоваться для вычислений имен клеток из области значений. Для того чтобы не усложнять этими операциями тело контекста, вычисления выносятся в контекстные подстановки, которые аналогичны функциям, за исключением того, что они не могут принимать параметров. Контекстые подстановки, как и обычные функции, должны возвращать значение (за исключением типа void), но в физическом смысле не явлются функциями, а их код непосредственно интерпретируется в точке вызова. Также в контекстные подстановки скрыто передается указатель на текущую клетку клеточной области, что дает возможность неявного обращения к данным. Вызов контекстной подстановки допускается до ее описания, но в любом случае она должна быть описана в теле клеточного автомата.

Синтаксис описания контекстной подстановки:

(тип, имя)

{

тело\_контекстной\_клетки

}

тип – тип возвращаемого контекстной подстановкой значения. В качестве типа могут использоваться любые стандартные типы (в том числе и void), кроме массивов и структур. Тип в описании контекстной подстановки, и тип переменной, которая принимает значение этой подстановки в контексте, должны совпадать. Это относится и к случаю, когда контекст играет роль предиката – контекстная подстановка и выражение в контексте должны быть одного типа.

тело\_контекстной\_клетки – аналогично телу обычной функции языка С/С++.

Ниже представлен листинг кода, демонстрирующий применение контекстной подстановки:

CellAuto NDiff(SimpleClassArray A)

{

System: A

{

Q1: (var x1, A.f), ( var x2, A.T[k].f) &

(var k, calc) –>

(x2, A.f), (x1, A.T.[k].f);

}

(int, calc)

{

if (0 < randf() <= 0.25) return 1;

else if (0.25 < randf() <= 0.5) return 2;

else if (0.5 < randf() <= 0.75) return 3;

else if (0.75 < randf() <= 1) return 4;

else return 0;

}

}

* + 1. Функции переходов

Функции переходов используются в правой части подстановок для вычисления новых значений клеток. Отличие функций переходов от обычных функций заключается в том, что в функцию переходов скрыто передается указатель на текущую клетку клеточной области, что позволяет обращаться к клеткам неявным способом.

Синтаксис описания функции переходов:

тип\_возвращаемого\_значения имя ([ параметры ])

{

тело\_функции

}

Ниже представлен листинг кода, демонстрирующий применение функции переходов:

CellAuto Live(TArray1 A)

{

System: A

{

Q1: ($, A.f) –>(calc(), A.f);

}

bool calc()

{

int sum = 0;

for (int i = 1; i < A.T.Size; i++) sum += A.T[i].f;

if (A.f == true)

if (sum == 2 || sum == 3) return true;

else return false;

else

if (sum == 3) return true;

else return false;

}

}

Пример описания клеточного автомата, реализующего игру Конуэна «Жизнь». Функция переходов рассчитывает новое состояние, которое будет присвоено клетке A.f.

* + 1. Механизм внешних воздействий

Механизм.

* 1. Проектирование среды моделирования

Основными требованиями для среды клеточно-автоматного моделирования являются:

* наличие пользовательского интерфейса;
* возможность описания клеточно-автоматной модели на спроектированном языке;
* наличие виджета в интерфейсе, отвечающего за визуализацию описанной модели;
* наличие интерпретатора, который проверяет корректность описания модели и осуществляет выполнение моделирования.

На рисунке 2.1 представлена структурная схема среды моделирования.

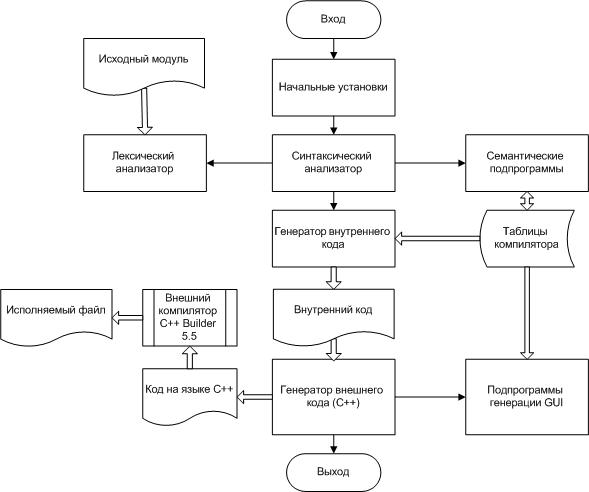


Рисунок 2.1 – Структурная схема среды моделирования

* + 1. Проектирование интерпретатора

Предполагаемый интерпретатор спроектированного языка является однопроходным, и его главной задачей является анализ и незамедлительная трансляция написанного пользователем кода описания моделируемого клеточного автомата.

На рисунке 2.2 представлена схема интерпретатора.

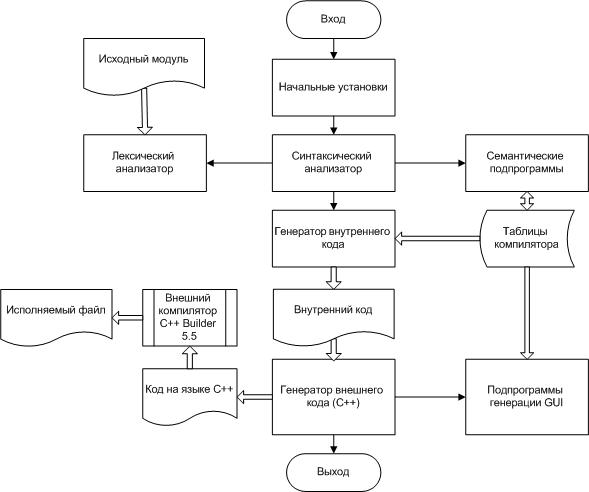


Рисунок 2.2 – Схема интерпретатора

Исходным модулем называется код, который пользователь написал при описании своей клеточно-автоматной модели. Он передается для анализа лексическому анализатору.

Лексический анализатор в свою очередь последовательно считывает лексемы текста исходного модуля. Если встречается лексема, не описанная в таблице лексем, то пользователю выдается лексическая ошибка, указывающая на неизвестную лексическую единицу.

Управление лексическим анализатором осуществляет синтаксический анализатор, который проверяет корректность синтаксиса введенного пользователем кода, анализируя набор считываемых лексическим анализатором лексем.

Семантические подпрограммы, внедряемые в синтаксический анализатор следят за корректным сопоставлением типов, за области видимости(здесь необходимо сказать про external переменные, подстановки и тд) и еще что то, а также тд пд иллд.

Подпрограммы интерпретации???

* + 1. Проектирование пользовательского интерфейса

Пользовательский интерфейс должен содержать в себе следующие компоненты:

* удобное поле ввода кода описания клеточно-автоматной модели с возможностью работы с несколькими вкладками для описания различных моделей;
* виджет визуализации работы клеточного автомата;
* инструментарий контроля над внешними воздействиями;
* панель контроля за процессом визуализации.

Поле для ввода кода должно иметь изменяемые размеры, а также иметь возможность изменять размер шрифта, что является удобной функцией для пользователя.

Виджет визуализации должен иметь достаточные размеры для четкого отображения процесса моделирования клеточного автомата. При необходимости, для более детального наблюдения за процессом визуализации, требуется возможность открытия виджета в новом окне с изменяемыми размерами.

Для установки и изменения значений вектора внешних воздействий, а также специальных переменных внешних воздействий должно быть предусмотрено поле ввода данных и кнопка подтверждения действия над контролируемыми параметрами. В случаях, когда требуется тщательный пошаговый контроль над влиянием внешних воздействий, можно использовать данный инструментарий вместе функциями контроля за процессом визуализации.

Панель контроля за процессом визуализации должна предоставлять пользователю возможность запуска процесса, его остановку, продолжение и завершение.

# **Программная реализация синтаксически-ориентированной среды клеточно-автоматного моделирования**

Программа имеет модульную структуру, которая изображена на рисунке 3.1. Точки взаимодействия обозначены с помощью стрелок.

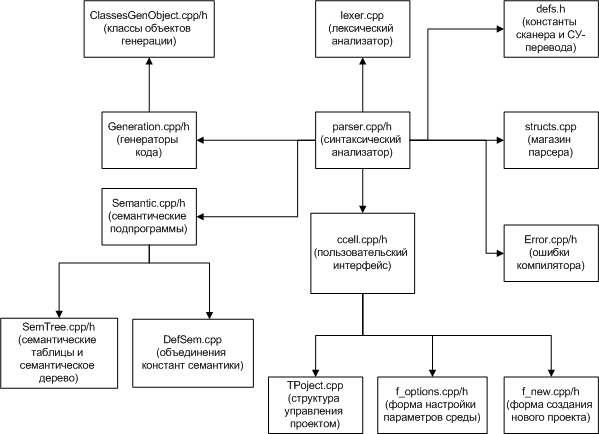


Рисунок 3.1 – Общая структура программы

* 1. Структура интерпретатора

При проектировании интерпретатора были определены основные его компоненты, такие как лексический анализатор и таблица лексем, синтаксический анализатор, процедуры семантического контроля, подпрограммы интерпретации. На рисунке 3.1 программные модули интерпретатора выделены отдельно.

* + 1. Лексический анализатор

Лексический анализатор за один свой проход считывает в исходном тексте одну лексему и выполняет поиск соответствия в таблице лексем, если соответствие найдено, он отдает все данные о лексеме другим модулям, в противном случае возвращается признак ошибочной лексемы.

В программной реализации модуль представлен основным классом *Scaner*, а также файлом, хранящим список лексем *defs.h*.

Класс *Scaner* содержит следующие поля и методы:

*char[] text* – объект для хранения исходного кода описания клеточно-автоматной модели;

*unsigned int pointer* – переменная для хранения номера очередного прочитанного символа исходного кода, является неким указателем на текущую точку анализа текста;

*int column* – переменная, хранящая номер столбца текущего символа анализируемого исходного кода;

*int row* – переменная, хранящая номер строки текущего символа анализируемого текста;

*Scaner(char\*), Scaner(const char\*)* – конструкторы класса с входным параметром в виде указателя на массив символов. Предполагает ипользование в двух случаях: считывание исходного кода из файла, либо получение исходного кода из текстового поля пользовательского интерфейса программной среды;

У полей *pointer, column* и *row* имеются методы получения и установки их значений.

*void loadFromFile(char\* path)* – метод считывания исходного кода в переменную text;

*void showError(char\* err, char\* lex)* – метод вывода ошибки при анализе очередной лексемы;

*int doScan(LEX lex)* – метод анализа очередной лексемы. Возвращает соответствующий идентификатор из таблицы лексем;

Файл *defs.h* содержит список поддерживаемых синтаксисом языка лексем, который представлен в таблице 3.1

Таблица 3.1 – Таблица лексических единиц спроектированного языка

| **Изображение лексемы** | **Тип лексемы** | |
| --- | --- | --- |
| Ключевые слова | | |
| <идентификатор> | t\_id = 1 | |
| var | t\_var = 2 | |
| int | t\_int = 3 | |
| long | t\_long = 4 | |
| double | t\_double = 5 | |
| bool | t\_bool = 6 | |
| neig | t\_neig = 7 | |
| classArray | t\_class\_array = 8 | |
| CellAuto | t\_cell\_auto = 9 | |
| System | t\_system = 10 | |
| const | t\_const = 11 | |
| new | t\_new = 12 | |
| <целая константа> | t\_int = 13 | |
| <целая длинная константа> | t\_long = 14 | |
| <вещественная константа с двойной точностью> | t\_double = 15 | |
| <константа bool> | t\_bool = 16 | |
| for | t\_for = 17 | |
| if | t\_if = 18 | |
| else | t\_else = 19 | |
| delete | t\_delete = 20 | |
| Специальные символы | | |
| { | | t\_op\_brace = 21 |
| } | | t\_cl\_brace = 22 |
| ( | | t\_op\_rnd\_bkt = 23 |
| ) | | t\_cl\_rnd\_bkt = 24 |
| [ | | t\_op\_sc\_bkt = 25 |
| ] | | t\_cl\_sc\_bkt = 26 |
| ; | | t\_semicolon = 27 |
| = | | t\_assignment = 28 |
| # | | t\_scharp = 29 |
| : | | t\_colon = 30 |
| . | | t\_dot = 31 |
| , | | t\_comma = 32 |
| –> | | t\_lambda = 33 |
| \* | | t\_star = 34 |
| + | | t\_plus = 35 |
| - | | t\_minus = 36 |
| / | | t\_div = 37 |
| > | | t\_more = 38 |
| >= | | t\_more\_equal = 39 |
| < | | t\_less = 40 |
| <= | | t\_less\_equal = 41 |
| == | | t\_equal = 42 |
| != | | t\_not\_equal = 43 |
| && | | t\_and = 44 |
| || | | t\_or = 45 |
| $ | | t\_not\_use = 46 |
| @ | | t\_end = 47 |

* + 1. Синтаксический анализатор
    2. Подпрограммы семантического контроля

Компонент

* 1. Компонент визуализации

Кек.

**Заключение**

В ходе выполнения данной практики в рамках взаимодействия с сотрудниками компании ООО «ЛинксСофт» были выполнены в полном объеме все задачи для достижения поставленной цели практики.

На первом этапе были проанализированы уже существующие алгоритмы классические и гибридные нечеткого вывода, а также выделены их основные проблемы.

На следующем этапе был разработан собственный алгоритм отсечения вариантов при нечетком выводе.

На третьем этапе осуществлялось проектирование алгоритма, разбиение его на модули и определение подхода к его программной реализации, а также выбраны средства и технологии разработки модуля.

В результате выполнения практики спроектированы следующие компоненты:

1. Архитектура базы данных;
2. Компонент инициализации переменных и выражений;
3. Компонент грамматического разбора выражений;
4. Компонент лексического анализа выражений;
5. Компонент по работе с ПОЛИЗ;
6. Компонент по работе с подвыражениями.

**Список использованных источников**

1. Zadeh, L. A.: Fuzzy Sets. Information and Control, vol. 8, pp. 338–353 (1965).

2. Бессмертный И.А. Методы, модели и программные средства для построения интеллектуальных систем на продукционной модели знаний: дисс... докт. техн. наук. – Санкт-Петербург, 2014.