Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального 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

“ ” 2018 г.

Руководитель от вуза

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Горячих А.И. /

подпись Ф. И. О.

ОТЧЕТ

по производственной практике

Проектирование языка Cell-C и его компилятора для клеточно-автоматного моделирования

общая формулировка задания

в ФГБОУ ВО «Алтайский государственный технический университет им. И.И.Ползунова»

наименование организации

Студент гр. 8ИВТ-71 Горских С.В.

индекс группы подпись Ф. И. О.

Руководитель от организации Тырышкин С.Ю.

подпись Ф. И. О.

Руководитель от университета Горячих А.И.

подпись Ф. И. О.

БАРНАУЛ 2018

ФГБОУ ВО «Алтайский государственный технический университет

им. И. И. Ползунова»

Кафедра «Информатика, вычислительная техника и информационная безопасность»

**Индивидуальное задание**

на производственную практику

(практика по получению профессиональных умений и опыта профессиональной деятельности) студенту 1 курса Горских С. В. группы 8ИВТ-71

Профильная организация: ФГБОУ ВО АлтГТУ Лаборатория информационно-измерительных систем

г.Барнаул (стационарная) (наименование)

Сроки практики: 30.04.2018 г. - 08.07.2018 г.

Тема: Проектирование языка Cell-C и его компилятора 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 ТБ. Описать методы, применяемые для проектирования, разработки и исследования объектов профессиональной деятельности в рамках магистерской диссертации | 1-2 недели | * Формирование части 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 части компетенции ОК-7: поиск информации по теоретическим и экспериментальным исследованиям по выбранной теме, применение знаний и умений в решении конкретных практических задач. | |
| 2 | Обосновать и описать программно-технические средства, необходимые для решения задач магистерской диссертации | 3-4 недели | Формирование части компетенции  ОК-8: способность проводить инсталляцию, диагностику и тестирование оборудования и программного обеспечения при выполнении задания практики, получить навыки работы на современном оборудовании, применяемом для разработок, исследований и производства изделий и иных продуктов в выбранной предметной области | |
| 3 | Изучить техническую документацию и техническую литературу, относящуюся к проектированию, разработке и исследованию объектов профессиональной деятельности в рамках магистерской диссертации | 5-6 недели | Формирование части компетенции ОПК-6: работать со специальной иностранной технической литературой и технической документацией | |
| 4 | Спроектировать и частично реализовать программно-техническое обеспечение для решения поставленных задач | 7-9 недели | Формирование части компетенции ПК-11:  участвовать в разработке аппаратных и/ или программных средств вычислительной техники, необходимых для достижения цели магистерской диссертации | |
| 5 | Оформить и сдать отчет по практике. | 10 неделя | Формирование части компетенции  ОПК-6: оформление отчета по результатам проектно-исследовательской деятельности | |
| |  |  | | --- | --- | | Руководитель практики от университета Горячих Анна Игоревна, к.п.н., доцент каф. ИВТиИБ (подпись) (Ф.И.О., должность) |  | | Руководитель практики от  профильной организации Тырышкин Сергей Юрьевич, к.т.н., доцент каф.ИВТиИБ  (подпись) (Ф.И.О., должность) |  |   Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Горских Сергей Вадимович  (подпись) (Ф.И.О.) | | | |  | | |
|  | | | | |  |
|  | | | | |  |

**Оглавление**

[Оглавление 4](#_Toc527928268)

[Перечень условных обозначений 5](#_Toc527928269)

[Введение 6](#_Toc527928270)

[Клеточно-автоматное моделирование 7](#_Toc527928271)

[1.1 Применение клеточных автоматов 7](#_Toc527928272)

[1.2 Анализ существующих решений для КА-моделирования 9](#_Toc527928273)

[2 Проектирование среды КА-моделирования 17](#_Toc527928274)

[2.1 Проектирование языка Cell-C 17](#_Toc527928275)

[2.2 Проектирование компилятора для языка Cell-C 23](#_Toc527928276)

[2.3 Постановка задачи на магистерскую диссертацию 31](#_Toc527928277)

[Список использованных источников 33](#_Toc527928278)

# Перечень условных обозначений

|  |  |
| --- | --- |
| КА | клеточный автомат |
| МКА | машина клеточных автоматов |
| ИТ | информационные технологии |
| ПО  АПП | программное обеспечение  алгоритм параллельных подстановок |

# Введение

Моделирование событий реального мира может производиться многими способами. Явления макромира достаточно хорошо описываются моделями, построенными на математике бесконечного и непрерывного. События же, происходящие в микромире, плохо поддаются описанию подобным способом и требуют применения других принципов моделирования.

Одним из способов описания событий микромира является мелкозернистый параллелизм. Основной отличительной особенностью систем с мелкозернистым параллелизмом является возможность одновременного (параллельного) изменения состояния всей системы, в то время как каждый участок системы взаимодействует только со своими непосредственными соседями. Это свойство позволяет при моделировании связать события, происходящие на микроуровне, с изменениями макроуровневого моделируемого объекта. Классическим примером мелкозернистого параллелизма является клеточный автомат.

Клеточно-автоматное моделирование находит себе применение во многих сферах. Для моделирования различных процессов было создано множество МКА. С развитием вычислительной мощности компьютеров упор на аппаратную архитектуру при разработке МКА существенно уменьшается. На данный момент существует множество программных продуктов для обычных компьютеров, осуществляющих моделирование с помощью клеточных автоматов. Большинство из них имеют функционал для демонстрации множества стандартных шаблонов клеточных автоматов и не предоставляют инструментарий для разработки пользовательской клеточно-автоматной модели.

Программный продукт, предоставляющий пользователю функционал для создания собственных клеточно-автоматных моделей, позволяет моделировать самые разнообразные процессы нашего мира. Для описания модели используется специальный язык программирования Cell-C, имеющий низкий порог входа, Си-подобный синтаксис и широкие возможности описания клеточных автоматов и их поведения.

# Клеточно-автоматное моделирование

## Применение клеточных автоматов

Клеточные автоматы нашли широкое применение и имеют высокое фундаментальное значение для всей науки. Начиная с работ Тоффоли и Марголоуса клеточные автоматы стали использоваться в моделях физико-химических процессов. К середине 90-х гг. клеточно-автоматное моделирование проникло в гуманитарные науки при изучении мультиагентных систем в урбанистике (толпа, транспортная пробка) [8]. Обзорная статья Ванага 1999 года по вероятностным КА еще раз легитимизировала для отечественных исследователей клеточные автоматы как метод математического моделирования.

Еще одно широко рассматриваемое направление применения клеточных автоматов — это математическое моделирование динамических процессов [9].

В настоящее время клеточные автоматы используются, как вычислительный инструмент для большого круга различных задач. Они могут упрощать расчеты в тех случаях, когда традиционные подходы приводят к сложным и требующим большого времени вычислениям.

Вероятно, это послужило основанием для того, чтобы применить решеточные газы - один из классов клеточных автоматов - для решения задач газодинамики.

Одной из первых удачных попыток такого рода был "НРР-газ" (названный по первым буквам фамилий своих создателей). Поле этого клеточного автомата представляет собой ортогональную решетку (2-х или 3-х мерную). Возможные состояния клетки соответствуют наличию в ней частиц, движущихся параллельно осям координат (не более одной частицы на каждое направление). На каждом временном шаге частица перемещается на одну клетку. Столкновения частиц считаются абсолютно упругими.

Несмотря на имеющуюся ярко выраженную анизотропию модели (скорости частиц строго параллельны осям координат), макроскопическая картина поведения автомата является изотропной.

Тем не менее, двумерный вариант этого, автомата имеет один недостаток, который в некоторых случаях является существенным: его макродинамическое поведение не удовлетворяет уравнению Навье-Стокса.

Этого недостатка лишен автомат «ТИР-газ», поле которого - гексагональная решетка, образованная равносторонними треугольниками. Более высокий порядок симметрии обеспечивает выполнение уравнения Навье-Стокса для этого клеточного автомата. С другой стороны, особая структура поля несколько усложняет его реализацию на компьютере и замедляют вычисления.

Газ, описываемый данным клеточным автоматом, естественно, является идеальным, т. е. взаимодействие между частицами сводится к упругим столкновениям. Последнее исключает возможность моделирования газодинамических процессов, в которых вещество существует в различных фазах, в частности, процессов, происходящих на границе раздела сред. Между тем, при решении подобных задач с помощью разностных методов возникают трудности, подчас непреодолимые, и использование в этом случае клеточных автоматов могло бы быть вполне уместным[10].

Наиболее эффективно клеточные автоматы используются для описания различных фазовых и бифуркационных переходов, где коллективное поведение системы определяется локальным поведением составляющих элементов. Например, они с успехом применяются в таких задачах как движение ансамблей живых организмов, моделирование различных физических явлений, начиная с элементарных явлений диффузии вещества и тепловых процессов и заканчивая явлениями, описываемыми уравнениями Навье-Стокса и Кортевега де Фриза, для расчета напряженности материалов, моделирования разрывов, деформаций и электрических явлений.

В настоящее время наибольшее применение клеточные автоматы нашли в задачах моделирования гидро- и газодинамических, эволюционных, поведенческих, колебательных и различных вероятностных процессов, что обусловлено сравнительной простотой их реализации, предрасположенностью к распараллеливанию и большими перспективами дальнейшего использования. Вопросам применения клеточных автоматов посвящены исследования многих отечественных и зарубежных учёных. Среди них Малинецкий Г.Г., Шакаева М.С., Лобанов А.И., Биндера К., von Neumann J., Martin O.,Toffoli T., Margolus N., Wolfram S., Moore F., Cipra B., Gacs P., Gardner M.,Gutowitz H. и др.

Однако известные клеточные автоматы не обладают достаточным быстродействием для моделирования неоднородных динамических систем задач в больших масштабах и на подробных сетках. Поэтому совершенствование данного численного метода является весьма перспективным.

Также клеточные автоматы активно применяются в криптографии[11], для генерации псевдослучайных чисел[12], в моделировании автотранспортных потоков[13]

## Анализ существующих решений для КА-моделирования

Резюмируя информацию об использовании клеточных автоматов, можно сказать, что КА используются практически везде. А для работы с ними требуются машины клеточных автоматов ­ – МКА. Данный термин введен Тоффоли. МКА является системой автоматизации проектирования клеточных автоматов; задавая правила функционирования клеточного автомата, можно реализовывать ту или иную математическую модель (или мультиагентную систему).

Рассмотрим «легковесные» и свободно распространяемые МКА. Информация по ним в кратком формате представлена в таблице 1.1.

Таблица 1.1 – Сравнение программного обеспечения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристики | Наименование программы | | | |
| Fam Life | MCell | Life32 | Golly |
| Год издания | 1998/2002 | 2001/2005 | 2002 | 2005/2009 |
| Возможность изменять правила | Да | Да | Да | Да |
| Встроенные конфигурации | Да | Да | Да | Да |
| Язык программирования | Не документировано | Delphi 5.0 (+Java) | Delphi 3.02 | Delphi |
| Быстродействие, с  (за 10 000 ходов) | 18 | 8 | 3 | 16  (Р-м HyperSpeed–1) |

Долгое время, начиная с 70-х годов, когда игра «Жизнь» получила известность в университетской среде США и Европы, МКА создавались безызвестными энтузиастами. До сих пор создаются простые симуляторы игры «Жизнь» на основе флэш-анимации, Java-апплетов. Однако если раньше такие МКА создавались исследователями-любителями, ищущими новые конфигурации, то теперь здесь пробуют силы новички в программировании. Примером служить МКА Life3D, где обычная 2D – игра визуализирована в 3D-пространстве с помощью API OpenGL. Также разработчики пакета MatLab включили в целях обучения в состав демонстрационных M-файлов небольшой симулятор игры «Жизнь».

Важным шагом в развитии МКА было появление симулятора Life 1.05, вместе с которым появился стандарт LIF для записи конфигураций и упрощенную запись правил перехода.

Наибольший интерес привлекает МКА Golly. Помимо открытого кода и кроссплатформенности она обладает следующими достоинствами:

* + размер поля ограничен только физической памятью;
  + число состояний ячейки до 256;
  + использует быстрый и эффективный по занимаемой оперативной памяти алгоритм расчета QuickLife;
  + возможность замены алгоритма расчета, в частности, алгоритм HashLife, основанный на хранении и хэшировании уже вычисленных фрагментов конфигураций и эффективный при симуляции на длительных временах больших по размеру конфигураций;
  + наличие библиотек, содержащих многие классические варианты: ID – игра Стивена Вольфрама, «Мир - проволока», «Поколения», автомат фон Неймана с 29 состояниями, самовоспроизводящийся автомат Фредкина;
  + интерфейс для создания собственных правил перехода;
  + экспорт/импорт конфигураций и паттернов для LIF-, RLE-, MCL-файлов, а также форматов macrocell, dblife;

Для Golly отметим также хороший интерфейс для задания исходных конфигураций с возможностью масштабирования поля и импорта библиотечных паттернов.

Рассмотрим более «тяжелые» и соответственно закрытые МКА, многие из которых являются коммерческой разработкой. Н. Марголус и Т. Тоффоли [14], начиная с середины 80-х годов XX-ого века, проводили в Массачусетском технологическом институте работы по симуляции клеточных автоматов для нужд биологии, кристаллографии, кристаллографии и других прикладных дисциплин. Последней моделью указанной серии является CAM-8, технически представляющая собой систолический массив процессоров, симулирующий параллельную архитектуру SIMD-типа. Собственно говоря, авторы термина «машина клеточных автоматов» изначально придавали ему более узкое значение, чем используется на данный момент. Они неявно полагали дополнительно, что физическая реализация вычислителя предполагает параллелизм при симуляции КА. И действительно, клеточный автомат может сам моделировать многопроцессорную систему, но, вместе с тем, его симуляция действительно эффективна при проведении распределенных вычислений, а не последовательных, выполняемых на компьютере с архитектурой фон Неймана. В настоящее время создан ряд компьютеров, специализированных для таких вычислений. Одними из пионерских и серьезных практических разработок в данном направлении можно отметить клеточные процессоры Легенди [15] и ML-процессоры. Итальянскими специалистами [16] в 1995 году создана вычислительная среда CAMEL, использующая КА-модель в качестве теоретической основы и успешно применяемая для моделирования.

Л. Наумов и А. Шалыто разработали собственную среду моделирования КА, называемую CAME&L [17]. Данная среда разрабатывалась, как простое, расширяемое рабочее пространство для сложных клеточных вычислений. На рисунке 1 представлен интерфейс приложения.

В данном приложении функциональность моделируемого автомата реализуется набором взаимодействующих компонентов.

* Grid – визуализация состояния решетки;
* Metrics ­– определяет координаты клеток, расстояние между ними и отношение соседства;
* Data – обеспечивает хранение состояния решетки в памяти и сохранение его в файл;
* Rules – реализует правила автомата и другие преобразования состояния решетки.

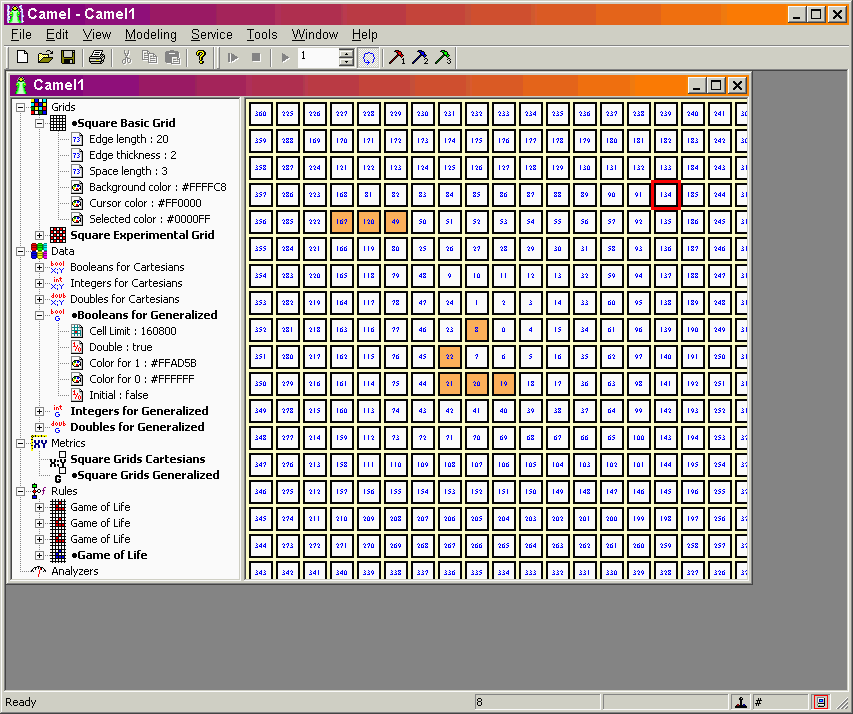


Рисунок 1 – Интерфейс программы Came&l

Достоинства данного решения:

* Обширный набор правил, позволяющих строить разнообразные модели клеточных автоматов;
* Богатый инструментарий для наблюдения за экспериментом;
* Сохранение состояния эксперимента в XML-файл.

Недостатки:

* Реализован в 2007 году, следовательно, используются устаревшие технологии вычислений;
* Интерфейс, не соответствующий современным требованиям;
* Достаточно сложный язык описания модели КА;

И. Матюшкин предложил свой вариант программы для моделирования КА, называемый SoftCAM [18]. На рисунке 2 представлена её архитектура в виде UML-диаграммы.

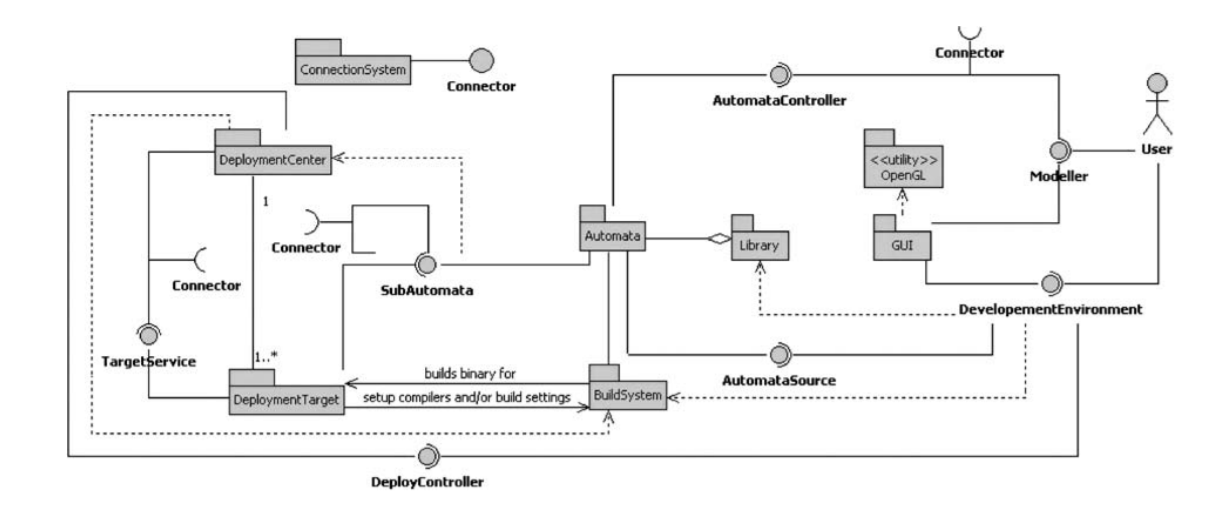


Рисунок 2 – Архитектура SoftCAM

Данное решение расширяет функционал моделирования такой МКА как Golly. Сделан упор на дружественность и открытость к пользователю, удобный и понятный интерфейс, легкий язык моделирования КА. К недостаткам относится низкое быстродействие, являющееся результатом выбора скриптового языка Python как средство разработки. Легкий язык моделирования КА является недостатком в случае проектирования сложных автоматов, когда код становится громоздким. Данная проблема частично решается созданием готовых шаблонов моделирования и библиотек.

Программный комплекс WinALT, разработанный на кафедре параллельных вычислений НГУ и в Институте вычислительной математики и математики геофизики СО РАН, обеспечивает конструирование мелкозернистых параллельных алгоритмов (МЗП) и структур в самой широкой трактовке термина. Данная система базируется на алгоритме параллельных подстановок – алгоритмической системе, являющейся пространственной моделью для представления параллельных мелкозернистых алгоритмов и структур [19]. Алгоритм параллельных подстановок (АПП) концептуально объединяет в себе подстановочный характер алгоритма Маркова с пространственной параллельностью клеточного автомата и основывается на общем для них ассоциативном механизме выполнения операций [20]. АПП позволяет описать вычисления в максимально параллельном виде, когда на каждом шаге выполняются все допустимые действия над всеми имеющимися данными. На рисунке 3 представлен графический интерфейс системы.

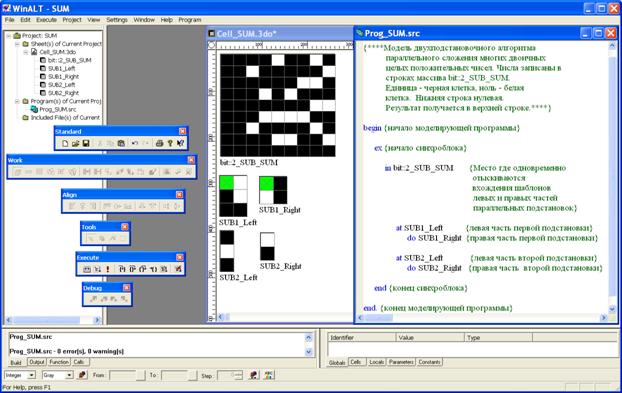


Рисунок 3 – Графическая среда системы WinALT

Ключевые достоинства WinALT:

* Возможность конструирования алгоритмов и структур с разнообразными видами МЗП;
* Полностью открытая система;
* Конструирование и модификация как данных модели, так и моделирующих программ, позволяет следить за динамикой изменения преобразуемых в модели данных;
* Многофункциональный графический интерфейс;
* Работа с объектами данных больших размеров.

К недостаткам можно отнести язык для описания работы клеточно-автоматных моделей, используемый в рассмотренной среде WinALT, имеющий не тривиальный синтаксис и не предоставляющий возможности использовать весь потенциал клеточных автоматов. Из-за ограниченных возможностей внутреннего языка, приходится прибегать к языкам высокого уровня, таким С++, что, в свою очередь, не решит поставленных задач.

Е. Баранов, А. Мирозян и А. Петров из РГАТУ имени П.А. Соловьева разработали аналог системы WinALT для реализации на ПЛИС. Их разработка названа QuickPSA[21].

Функционал приложения разработан на скриптовом языке Python. В целом данное решение во многом повторяет систему WinALT, учитывая, что в производительности QuickPSA уступает. На рисунке 4 представлен графический интерфейс.

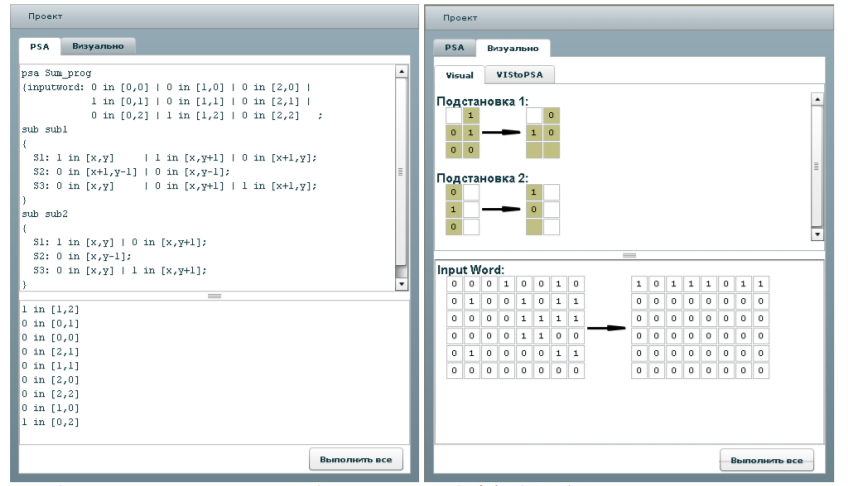


Рисунок 3 – Графический интерфейс программы QuickPSA

У программы QuickPSA стоит лишь отметить удобный и легкий в понимании язык моделирования КА.

# Проектирование среды КА-моделирования

Среда для клеточно-автоматного моделирования состоит из следующих частей:

* Клиентский интерфейс для описания клеточно-автоматных моделей на языке Cell-C
* Модуль визуализации процесса моделирования
* Компилятор языка Cell-C

В рамках данной практики проектируется язык Cell-C и его компилятор.

## Проектирование языка Cell-C

Предполагаемый язык является Си-подобным компилируемым 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 директив препроцессора, объявлений или определений данных, описаний функций, описаний клеточных автоматов. Также допускаются описания пользовательских типов, таких как классы и 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, поддерживаемые данным языком представлены в таблице 2.1.

Таблица 2.1 ­– Типы данных

|  |  |
| --- | --- |
| ***Тип*** | ***Краткое описание*** |
| *Стандартные типы данных* | |
| int | Целый (4 байта) |
| bool | Булев (1 байт) |
| float | Вещественный (4 байта) |
| double | Вещественный с двойной точностью (8 байт) |
| char | Символьный (1 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) |
| area | Клеточная область (переменный размер) |

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

Таблица 2.2 – Список поддерживаемых операторов языка Си

|  |  |
| --- | --- |
| ***Оператор*** | ***Краткое описание*** |
| [выражение]; | Оператор «выражение» |
| while | Цикл с предусловием |
| do while | Цикл с постусловием |
| 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 |
| += | Сложение с присваиванием |
| -= | Вычитание с присваиванием |
| \*= | Умножение с присваиванием |
| /= | Деление с присваиванием |
| . | Операция доступа |

Рассмотрим 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 соседние клетки, которые будут задействованы в очередной итерации автомата.

Ниже представлен листинг кода описания класса клеточного массива:

**classArray** TArray1 <CA.Lattice.HEXAGONAL\_2D, CA.Surface.TOR>

{

**bool** data;

**neig[]** T = {#[0], #[1], #[2], #[3], #[4], #[5], #[6],

#[7], #[1,2]};

};

//В классе TArray1 определен шаблон соседства T, состоящий

//из 9 элементов,

//причем первый элемент шаблона указывает на центровую клетку,

//а последний

//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, *размеры\_клеточного\_массива* – численное значение размеров создаваемого клеточного массива.

Ниже представлен листинг кода определения клеточного массива:

TArray1 A1 = **new** TArray1(500, 500);

//Определение клеточного массива A1 класса TArray1, который задан

//на 2D пространстве

//Размеры клеточного массива 500 на 500 клеток.

Клеточный автомат представляет собой функцию, которая по заданным правилам преобразует указанные клеточные массивы. Описание клеточного автомата состоит из система 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 крайней 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)

{

системы\_подстановок

[ контекстные\_подстановки ]

[ функции\_переходов ]

}

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

*системы\_подстановок* – системы параллельных подстановок. Представляет собой множество правил переходов клеточного автомата и может состоять из одной, или нескольких параллельных подстановок.

Синтаксис описания системы параллельных подстановок:

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

{

параллельные\_подстановки

}

*указатель\_области\_значений* – указывается клеточный массив, над которым будет применены параллельные подстановки данной системы.

*параллельные\_подстановки* – список параллельных подстановок системы.

Синтаксис параллельной подстановки:

имя: база [ \* контекст ] -> правая\_часть [ : циклы ];

*база* – определяет локальную конфигурацию, клетки которой участвуют 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.

Ниже представлен листинг кода, демонстрирующий простую функцию определения клеточного автомата с системой параллельных подстановок, состоящей из одной простой подстановки:

**CellAuto** Test1(TArray2 B)

{

//Описание системы подстановок, состоящей из одной подстановки Q1:

**System** : B  
 {

Q1: (**out** x1, B.#[i].v), ( **out** y1, A.#[i].f) \* //База

(**true**, ma) -> //Контекст

(y1, B.#[i].v), (x1, A.#[i].f) : //Правая часть

**for**(**int** i=0; i<8; i++); //Цикл по подстановке

}

//Описание контекстной подстановки ma:

(**bool**, ma)

{

**if** (randf() > 0.5) **return true**;

**else** **return** **false**;

}

};

Чтобы клеточный автомат работал, его необходимо запустить. Запуск клеточного автомата аналогичен вызову функции. При запуске клеточному автомату необходимо передать фактические параметры, а также указать опции запуска.

Синтаксис запуска клеточного автомата:

клеточный\_автомат[.имя\_системы] ([ параметры ] :

режим\_работы, размер\_эволюции);

клеточный\_автомат – имя запускаемого клеточного автомата.

имя\_системы – является необязательной частью запуска и позволяет запускать не весь клеточный автомат, а только указанную систему параллельных подстановок. Если у системы подстановок не было задано имя, то такую систему невозможно запустить отдельно.

параметры – фактические параметры клеточного автомата.

режим\_работы – режим работы клеточного автомата. Может принимать значения, которые предопределены в стандартном наборе языка. Такими основными режимами являются: синхронный, асинхронный, асинхронно-упоряоченный, блочно-синхронный.

размер\_эволюции – задает количество итераций в эволюции. Может принимать в качестве значения целочисленные переменные, константы и выражения.

Листинг кода, демонстрирующий запуск клеточного автомата:

NDiff(A1 : CA.Mode.ASYNCH, 50);

//Запуск клеточного автомата NDiff моделирующего «наивную диф-

//фузию» Клеточный автомат будет работать в асинхронном режиме //на клеточном массиве A1 и совершит 50 итераций эволюции.

## Проектирование компилятора для языка Cell-C

Проектируемый компилятор языка Cell-C является многопроходным и использует внутренний код, который в дальнейшем может быть транслирован во внешний код. Логическая схема компилятора представлена на рисунке 2.1. Сгенерированный таким образом внешний код может быть собран в исполняемый файл при помощи внешнего компилятора.

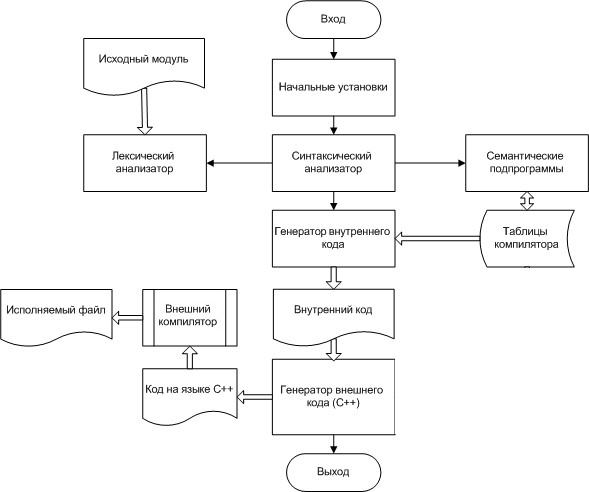


Рисунок 2.1 – Схема компилятора языка Cell-C

В качестве синтаксического анализатора был выбран LL(1)-анализатор, поэтому синтаксис языка Cell-C описан с помощью LL(1)-грамматики. Ниже представлена грамматика языка.

<программа> → <конструкции> <интерфейс>

<конструкции> → <конструкции> <элемент> | e

<интерфейс> → GUInterface <ид-р> { <опис.интерфейса> }; | e

<опис.интерфейса> → <опис.интерфейса> <ид-р> <ид-р>;

| <опис.интерфейса> <ид-р> :: <ид-р> <тело ф-и> | e

<элемент> → # <директива> | <КА> | <данные>; | const <данные>; | <>

| class <ид-р> { <тело класса> };

|classArray <ид-р> <<выр1>, <выр1>> { <тело кл.массива> };

<директива> → include <конст.строков.> | pragma <ид-р>

<данные> → <тип> <объект>

<тип> → int | float | double | char | bool | string | cell | area

| <ид-р>

<объект> → [ <измерения> ] <мас.список> | <список1>

<измерения> → <измерения>, <измерение> |<измерение> | e

<измерение> → ???

<список1> → <список1>, <переменная1> | <переменная1>

<переменная1> → <ид-р> = <переменная2> | <ид-р>

<переменная2> → <выр.1> | new <тип> ( <фак.параметры1> )

<фак.параметры1> → <фак.параметры1>, <выр.1> | <выр.1>

<мас.список> → <мас.список>, <эл.массива> | <эл.массива>

<эл.массива> → <ид-р> = <тип.иниц.> | <ид-р>

<тип инициал.> → { <иниц.массива.1> } | new <тип> [ <иниц.массива.1> ] | <выр.1>

<иниц.массива.1> → <иниц.массива1>, <выр.1> | <выр.1>

<ф-я> → <тип ф-и> <ид-р> ( <форм.пар-ры1> ) <тело функции>

<тип ф-и> → void | tran void | <тип> <фун.мас.> | tran <тип> <фун.мас.>

<фун.мас.> → [ <измерения> ] | e

<форм.пар-ры1> → <форм.пар-ры1>, <пар-р> | <пар-р1> |e

<пар-р1> → <тип> <пар-р2> | const <тип> <пар-р2>

<пар-р2> → [ <измерения> ] <ид-р> | <ид-р> <иниц.пар-ра>

<иниц.параметра> → = <выр1> | e

<тело функции> → { <опер-ры> } | ;

<тело класса> → <тело класса> <эл-т класса> | e

<эл-т класса> → <поле>; | <ф-я> | ~ <ид-р> <тело ф-и>

| <ид-р> (<форм.пар-ры1>) <тело ф-и>

<поле> → <тип> <элемент поля>

<элемент поля> → [ <измерения> ] <п.список1> | <п.список1>

<п.список1> → <п.список1>, <ид-р> | <ид-р>

<тело кл.массива> → <тело кл.массива> <поле>;

| <тело кл.массива> neig [] <ид-р> = {<эл.шаблона1>}; | e

<эл.шаблона1> → <эл.шаблона1>, #[ <ук.эл.шаблона> ] | #[ <ук.эл.шаблона> ]

<ук.эл.шаблона> → <ук.эл.шаблона>, <конст.целая> | <конст.целая>

<КА> → CellAuto <ид-р> (<форм.параметры1>){<тело КА1>};

<тело КА1> → <тело КА1> <сист.подстан.> | <тело КА1> <ф-я> | e

<сист.подстан.> → System <ид-р> : <имя1> <опр.шаблон> {<парал.подст.1>}

<опр.шаблон> → < <имя1> > | e

<парал.подст.1> → <парал.подст.1> <подстановка>; | <подстановка>;

<подстановка> → <ид-р>:<база1><контекст> -> <пр.часть><циклы>

| ( <тип> <фун.мас.> , <ид-р> ) {<опер-ры>}

<база> → <база>, <эл-т базы> | <эл-т базы>

<эл-т базы> → ( <out> <выр.базы> , <имя1> )

<выр.базы> → <выр1> | $

<out> → out | e

<контекст> → <контекст> <эл-т контекста>| \* <эл-т контекста> | e

<эл-т контекста> → ( <out> <выр1>, <имя1> )

<пр.часть> → <пр.часть>, <эл-т пр.части> | <эл-т пр.части>

<эл-т пр.части> → ( <выр1> , <имя1> )

<циклы> → <циклы>, <for КА> | : <for КА>

<for КА> → for (<п.цикла1> ; < п.цикла2> ; < п.цикла3>)

<сост.оператор> → { <опер-ры> }

<опер-ры> → <опер-ры> <оператор> | e

<оператор> → <присваивание>; | <сост.оператор> | <if>; | <for>; | ; |

return <возв.значение>; | delete <имя1>; | <данные>; |

break; | continue; | abreak; | acontinue; |

Parallel ( <фак.параметры1> ) { <композиция КА> }

<if> → if ( <выр1> ) <оператор> <else>

<else> → else <оператор> | e

<for> → for (<п.цикла1> ; < п.цикла2> ; < п.цикла3>) <оператор>

<п.цикла1> → <данные> | <присваивание> | e

<п.цикла2> → <выр1> | e

<п.цикла3> → <присваивание> | e

<возв.значение> → <выр1> | e

<композиция КА> → <композиция КА> <эл-т композиции КА> | e

<эл-т композиции КА> → <ид-р> <в.системы>

(<фак.параметры1> : <выр1> , <выр1>) <домен>;

<в.системы> → . <ид-р> | e

<домен> → domain <конст.целая> | e

<присваивание> → <имя1> <присваивание1> | ++ <имя1> | -- <имя1>

<присваивание1> → = <выр1> <new> | ++ | -- | e

<new> → new <new type> | e

<new type> → ( <фак.параметры1> ) | [ <фак.параметры1> ]

<имя1> → <имя2> ( <фак.параметры1> <запуск КА> ) | <имя2>

<имя2> → <имя2> . <имя3> <эл. массив> | <ид-р> <эл.массив>

<имя3> → <ид-р> | #

<эл.массив> → [ <иниц.массива.1> ] | e

<запуск КА> → : <выр1> , <выр1> | e

<выр1> → + <сравнение> | - <сравнение> | <сравнение>

<сравнение> → <сравнение> > <слагаемое> | <сравнение> >= <слагаемое> |

<сравнение> < <слагаемое> | <сравнение> <= <слагаемое> |

<сравнение> == <слагаемое> | <сравнение> != <слагаемое> |

<сравнение> & <слагаемое> | <сравнение> && <слагаемое> |

<сравнение> | <слагаемое> | <сравнение> || <слагаемое>

| <слагаемое>

<слагаемое> → <слагаемое> + <множитель> | <слагаемое> - <множитель>

| <множитель>

<множитель> → <множитель> \* <эл.выр> | <множитель> / <эл.выр>

| <множитель> % <эл.выр> | <эл.выр>

<эл.выр> → <имя1> | <константа> | ( <выр1> )

<константа> → <конст.целая> | <конст.веществ.> | <конст.символьн.> |

<конст.строков.> | true | false | NULL

Лексические единицы языка представлены в таблице 2.2, в левой части которой находятся изображения лексем, а в правой – их типы, обозначенные в виде именованных констант.

Таблица 2.2 – Таблица лексических единиц языка Cell-C

|  |  |
| --- | --- |
| **Изображение лексемы** | **Тип лексемы** |
| ***1*** | ***2*** |
| *Ключевые слова* | |
| GUI | T\_gui = 100 |
| include | T\_include = 101 |
| pragma | T\_pragma = 102 |
| const | T\_const = 103 |
| class | T\_class = 104 |
| classArray | T\_carray = 105 |
| int | T\_int = 106 |
| float | T\_float = 107 |
| double | T\_double = 108 |
| char | T\_char = 109 |
| bool | T\_bool = 110 |
| string | T\_string = 111 |
| cell | T\_cell = 112 |
| area | T\_area = 113 |
| new | T\_new = 114 |
| void | T\_void = 115 |
| tran | T\_tran = 116 |
| neig | T\_neig = 117 |
| CellAuto | T\_cauto = 118 |
| System | T\_syst = 119 |
| out | T\_out = 120 |
| for | T\_for = 121 |
| if | T\_if = 122 |
| else | T\_else = 123 |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| ***1*** | ***2*** |
| delete | T\_delete = 124 |
| break | T\_break = 125 |
| continue | T\_cont = 126 |
| return | T\_ret = 127 |
| abreak | T\_abreak = 128 |
| acontinue | T\_acont = 129 |
| Parallel | T\_paral = 130 |
| domain | T\_dom = 131 |
| <Идентификатор> | T\_id = 200 |
| <Константа целая> | T\_coni = 201 |
| <Константа вещественная> | T\_conf = 202 |
| <Константа символьная> | T\_conch = 203 |
| <Константа строковая> | T\_cons = 204 |
| true | T\_true = 205 |
| false | T\_false = 206 |
| NULL | T\_null = 207 |
| *Специальные знаки* | |
| { | T\_fso = 220 |
| } | T\_fsz = 221 |
| ( | T\_so = 222 |
| ) | T\_sz = 223 |
| [ | T\_kso = 224 |
| ] | T\_ksz = 225 |
| < | T\_uso = 226 |
| > | T\_usz = 227 |
| . | T\_tch = 228 |
| , | T\_zp = 229 |
| ; | T\_tzp = 230 |
| : | T\_dv = 231 |
| -> | T\_ska = 248 |
| ~ | T\_tld = 270 |
| $ | T\_usa = 272 |
| *Знаки операций* | |
| = | T\_prisv = 232 |
| == | T\_rav = 233 |
| != | T\_nrav = 234 |
| <= | T\_mrav = 235 |
| >= | T\_brav = 236 |
| + | T\_pls = 237 |
| - | T\_mns = 238 |
| ++ | T\_inc = 239 |
| -- | T\_dec = 240 |
| & | T\_and = 241 |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| ***1*** | ***2*** |
| && | T\_land = 242 |
| | | T\_or = 243 |
| || | T\_lor = 244 |
| \* | T\_umn = 245 |
| / | T\_razd = 246 |
| % | T\_ost = 247 |
| ! | T\_inv = 250 |
| # | T\_sharp = 271 |
| *Служебные символы* | |
| *Конец файла* | T\_endf = 333 |
| *Ошибка* | T\_error = 666 |

На рисунке 2.2 представлены конечные автоматы, реализующие следующие лексические конструкции языка:

* Идентификатор
* Константа целая
* Константа вещественная
* Константа символьная
* Константа строковая

Символами *a* и *c* обозначены буквы латинского алфавита (A – Z, a – z) и цифры (0 – 9) соответственно.

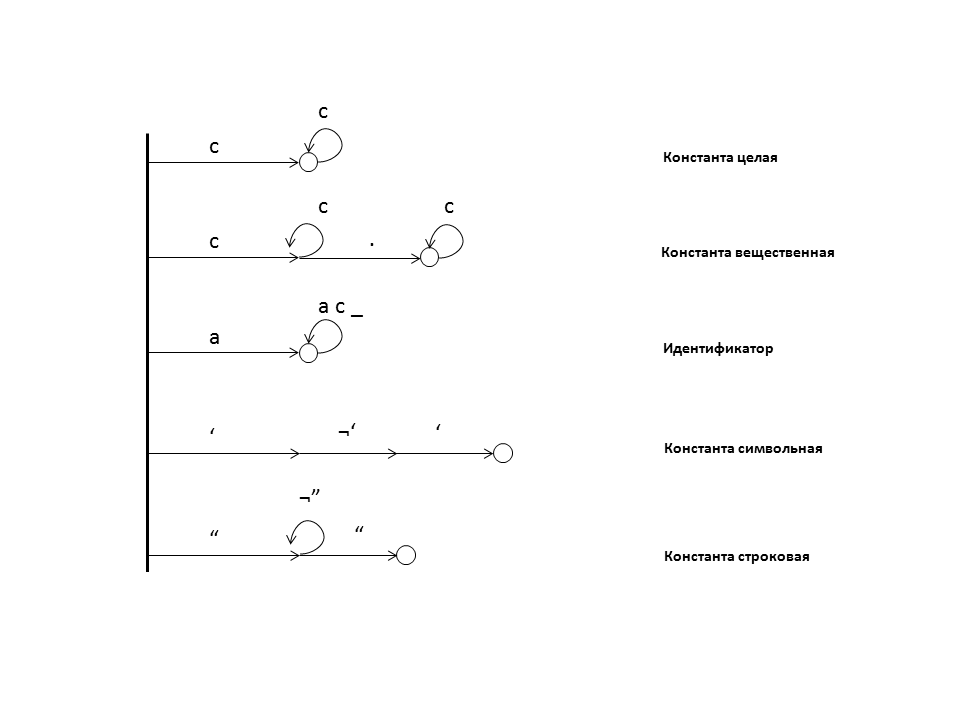


Рисунок 2.2 – Конечные автоматы

В качестве синтаксического анализатора языка Cell-C был выбран LL(1)-анализатор. LL-анализатором называется нисходящий синтаксический анализатор для LL-грамматики, который анализирует входную цепочку слева направо и строит левый вывод грамматики. LL-анализатор является LL(1)-анализатором, если при разборе входной цепочки он просматривает не более 1 символа этой цепочки. Структурная схема анализатора представлена на рисунке 2.3

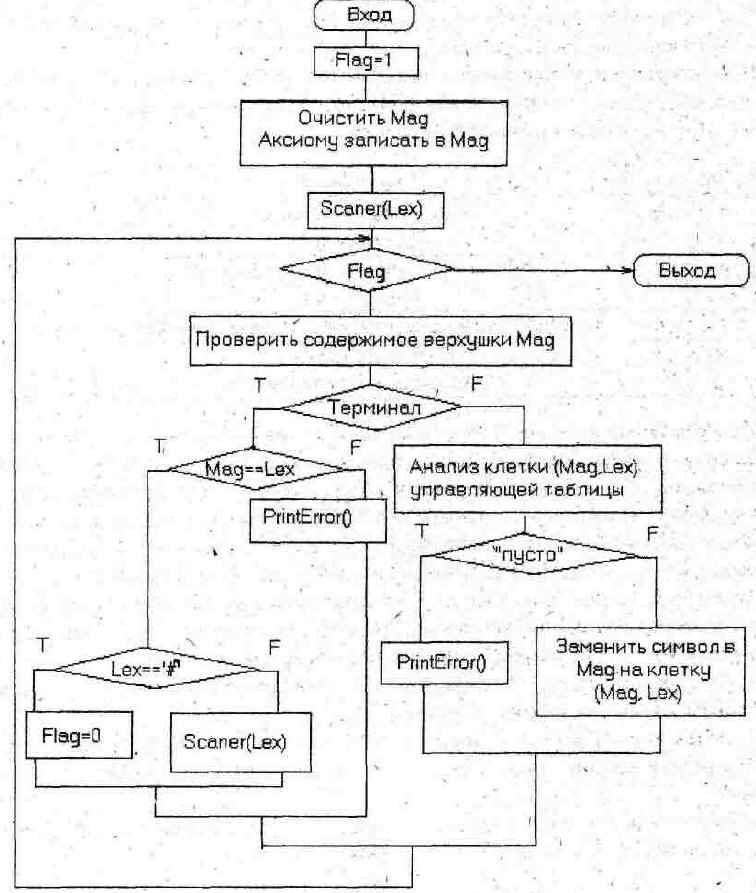


Рисунок 2.3 – Структурная схема LL(1)-анализатора

На рисунке 2.4 представлена структура модулей проектируемой программы.

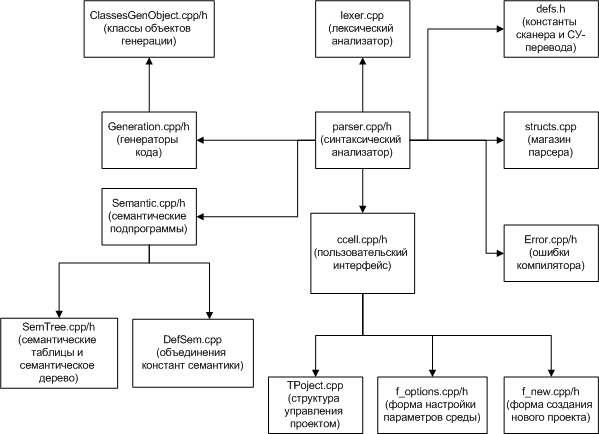


Рисунок 2.4 – Структура программы

Модуль «Константы сканера и СУ-перевода» содержит описания групп именованных констант: лексемы и служебные символы сканера, нетерминалы грамматики, дельта-правила семантики, дельтра-правила генерации.

Модуль «Лексический анализатор» содержит описание класса TLexer, который реализует лексический анализатор компилятора.

Модуль «Синтаксический анализатор» содержит описание класса TLLParser, который реализует синтаксический анализатор LL(1) компилятора

Модуль «Семантические таблицы и семантическое дерево» содержит функционал для создания семантических таблиц конструкций языка и построения семантического дерева.

Модуль «Семантические подпрограммы» содержит функции, реализующие расширение синтаксических процедур логикой семантики языка.

Модуль «Объекты генерации кода». Данный модуль содержит классы объектов генерации, которые используются в генераторах кода и представляют собой конструкции внутреннего кода, используемого компилятором.

## Постановка задачи на магистерскую диссертацию

Анализ существующих решений для моделирования клеточных автоматов определяет направления для совершенствования в данной области. Каждый из рассмотренных МКА обладает достоинствами и недостатками, сущности которых определились в зависимости от предположенных при разработке критериев использования. Другими словами, МКА, акцентированные на визуализации, проигрывают в остальных критериях: эффективности распараллеливания, вариациях моделирования и т.д, а решения, в которых сделан упор на охват функциональности при моделировании, проигрывают в простоте языка.

При разработке собственного решения для клеточно-автоматного моделирования необходимо определиться с направлениями, которые наиболее важны. Это должна быть качественная среда, обладающая такими свойствами, как доступность, простота, надежность и эффективность. Необходимо использовать быстрые алгоритмы распараллеливания, разработать язык моделирования, позволяющий создавать модели клеточных автоматов разной степени сложности, сохраняя легкость в понимании и простоту при его трансляции для выполнения. Удобный функционал визуализации и наблюдения за процессом моделирования также являются важными аспектами при разработке решения.

Для достижения определенных критериев разрабатываемого решения необходимо выполнить следующие задачи:

1. Разработать каждый из описанных модулей: синтаксический анализатор, семантические подпрограммы и остальные.
2. При разработке модуля генерации внутреннего кода необходимо использовать различный функционал оптимизации исходного кода перед его трансляцией.
3. Рассмотреть различные алгоритмы распараллеливания работы клеточных автоматов и реализовать наиболее эффективный.
4. Провести профилирование каждого модуля и определить наиболее трудоемкие процедуры, и осуществить их оптимизацию.
5. Провести тестирование разработанного решения, осуществить анализ результатов тестирования, сделать выводы.
6. По результатам аналитического анализа работы продукта осуществить оценку эффективности разработанного продукта.

# Заключение

В ходе прохождения производственной практики были выполнены в полном объеме некоторые задачи, определенные для достижения поставленной цели дипломного проекта.

На начальном этапе разработки были проанализированы существующие программные продукты для клеточно-автоматного моделирования, описаны их основные достоинства и недостатки.

В ходе последующих этапов осуществлялось проектирование языка Cell-C и его компилятора, был определен общий подход к программной реализации, выбраны средства и технологии разработки программного обеспечения, осуществлялась его реализация.

В результате прохождения производственной практики был спроектирован язык моделирования Cell-C, а также его компилятор и описаны основные конструкции языка и модули компилятора.

# Список использованных источников

1. Toffolli T. Cellular Automata Machine [Текст] / Toffolli T., Margolus N., USA: MIT Press, 1987. – 280c.
2. Achasova S. Parallel Substitution Algorithm. Theory and Application [Текст] / Achasova S., Bandman O., Markova V., Piskunov S., Singapore: World Scientific, 1994. – 296c.
3. Бандман О.Л. Метод построения клеточно-автоматных моделей процессов формирования устойчивых структур [Текст] / Бандман О.Л., ПДМ, 2010, № 4, 91–99.
4. Бандман О.Л. Параллельная реализация клеточно-автоматных алгоритмов моделирования пространственной динамики [Текст] / Бандман О.Л., Сиб. журн. вычисл. математики., РАН. Сиб. отд-ние. – Новосибирск, 2007. – Т. 10, №4. – С.335-348.
5. Бандман О.Л. Клеточно-автоматные модели пространственной динамики [Текст] / Бандман О.Л., Системная информатика, -Новосибирск, 2006, №10, С59.-113.
6. Бандман О.Л. Мелкозернистый параллелизм в вычислительной математике [Текст] / Бандман О.Л., Программирование, -Новосибирск, 2001, №4, 1-18.
7. Бандман О.Л. Методы композиции клеточных автоматов для моделирования пространственной динамики [Текст] / Бандман О.Л., Вестник Томского университета, -Томск, 2002, №9(1), 188-192.
8. Bandman O.L. Cellular Automata composition techniques for spatial dynamics simulation [Текст] / Bandman O.L., Bull. Nov. Comp. Center, Comp. Science., 2008. – V.27. P.1-39.
9. Bandman O.L. Parallel Simulation of Asynchronous Cellular Automata Evolution [Текст] / Bandman O.L., Lect. Notes in Comp. Sci., 2006. – Vol. 4173. P.41-48.
10. Медведев Ю.Г. Расширение клеточно-автоматной модели потока FHP-I до многочастичного варианта FHP-MP [Текст] / Медведев Ю.Г., Новые информационные технологии в исследовании сложных структур, -Томск, 2008. – С73.
11. Cellular Automata [Электронный ресурс] / Режим доступа: http://www.cell-auto.com
12. Крючкова Е.Н. Теория языков программирования и методы трансляций [Текст] / Крючкова Е.Н. - Алт. гос. техн. унив. Им. И.И.Ползунова -Барнаул: АлтГТУ им. И.И. Ползунова, 2003. – 240 с..
13. Конспект лекций «Методы построения компиляторов» [Электронный ресурс] / Режим доступа: <http://it.mmcs.rsu.ru/wiki/Конспект_лекций_> «Методы\_построения\_компиляторов»
14. Элементы теории перевода [Электронный ресурс] / Режим доступа: http://citforum.ru/programming/theory/serebryakov/5.shtml
15. Павловская Т.А. С/C++ Программирование на языках высокого уровня [Текст] / Павловская Т.А. - СПб.: Питер, 2004. – 461с.
16. H. Gutowitz., Cellular automata: theory and experiment // Mit Press. Cambridge, 1991
17. В. К. Ванаг, “Исследование пространственно распределенных динамических систем методами вероятностного клеточного автомата”, УФН, 169:5 (1999), 481–505; Phys. Usp., 42:5 (1999), 413–434
18. Wolfram S. A New Kind of Science. N. Y.: Wolfram Media, 2002. 1197 с.
19. Culik K., Yu S. Undecidability of CA classification schemes // Complex Systems. 1988. Р. 177-190.
20. Li W., Packard N. The structure of the elementary cellular automata rule space // Complex Systems, 1990. Р. 281-297.
21. G. Braga [et al] Pattern growth in elementary cellular automata // Theoret. Comput. 1995. Р. 1-26.
22. Fates N., Morvan M., Remila E. Experimental study of elementary cellular automata dynamics using the density parameter // Discrete Mathematics Theoretical Computer Science. 2003. Р. 155-166.
23. Введение в математическое моделирование транспортных потоков / А.В. Гасников[и др.]М.: МФТИ, 2010.
24. Степанцов М.Е. Применение клеточных автоматов для математического моделирования динамических процессов: Автореф. дис. …канд. тех. наук. –М., 1999. – 18 с.
25. Малинецкий Г. Г., Степанцов М. Е. Применение моделей класса решеточных газов для решения задач газодинамики // Известия Высших учебных заведений. Прикладная нелинейная динамика, 1996, том 4, N 4,5, с.59-04.
26. Жуков А.Е., Клеточные автоматы в криптографии //Вопросы кибербезопасности. – 2017. – №4(22). – C. 47-66.
27. Разработка генераторов псевдослучайных двоичных последовательностей на основе клеточных автоматов Сухинин Б.М. [Электронный ресурс] / Науки и образование. Режим доступа: <http://technomag.edu.ru/doc/159714.html>. свободный. – Загл. с экрана.
28. Долгушин Д.Ю. Многофакторное моделирование транспортных потоков на основе клеточных автоматов: Автореф. дис. …канд. тех. наук. –М., 1999. – 18 с
29. Тоффоли Т., Марголус Н., Машины клеточных автоматов. М.: Мир, 1991. – 280 с.
30. Legendi T. Cellprocessors in Computer Architecture // Comp. Linguist. and Comp. Languages. 1976. Vol. 11. N 2. P. 147—167.
31. Cannataro M. et al. A parallel cellular automata environment on multicomputers for computational science // Parallel Computing. 1995. Vol. 21. P. 803—823.
32. Наумов Л. Метод введения обобщенных координат и инструментальное средство для автоматизации проектирования программного обеспечения вычислительных экпериментов с использованием клеточных автоматов //Дисс. на соиск. уч. ст. канд. техн. наук. СПбГУ ИТМО. 2007. 283 с.
33. Матюшкин И. В. Перспективы развития современных средств проектирования клеточных автоматов // Информационные технологии. – 2011. – №4. С. 8 – 12.
34. Остапкевич М. Б., Пискунов С.В. Система имитационного моделирования алгоритмов с мелкозернистым параллелизмом WinALT // Вестник НГУ. Сер. Информационные технологии. 2012. Том 10. Выпуск 3. – С. 34 – 45.
35. Остапкевич М. Б. Параллельная реализация мелкозернистых алгоритмов в системе WinALT // Вестник ЮУрГУ. Сер. Вычислительная математика и Информатика. 2012. Том 2. Выпуск 1. –С. 80 – 89.
36. Баранов Е.Ю., Мирозян А.С., Петров А.В. Программа моделирования алгоритма параллельных подстановок и пример его реализации на ПЛИС [Электронный ресурс] // Конструкторское бюро «Информационные технологии», 2015. URL: <http://kbit.rsatu.ru/skbit/images/articles/plis/1.pdf> (дата обращения: 01.03.2018).