DOI 10.37882/2223-2966.2026.02-2.22

ЭКСПЕРИМЕНТ ДАННЕНБЕРГА-ЛЬЮИС: КЛЕТОЧНЫЕ АВТОМАТЫ И АЛГОРИТМИЧЕСКО-КОМБИНАТОРНАЯ СТРУКТУРА АУДИОДАННЫХ (НАУЧНЫЙ АНАЛИЗ, АУДИОПРОГРАММИРОВАНИЕ, ВОСПРОИЗВЕДЕНИЕ ЭКСПЕРИМЕНТА И ФАКТОРЫ ЕГО УТОЧНЕНИЯ)

Таран Василий Васильевич
кандидат культурологии, заведующий Лабораторией компьютерного дизайна и прикладной информатики «SPLASHLab»
allscience@lenta.ru

Аннотация. В статье рассматриваются вопросы практического применения принципа клеточных автоматов для формирования элементарных и сложных алгоритмическо-комбинаторных структур аудиоданных. Научно обоснована и технически аргументирована концепция использования структурно-алгоритмического функционала клеточных автоматов, подразумевающая прямую генерацию многочастотного спектра средствами функционального программирования. В качестве основного функционального средства программирования экспериментальных возможностей клеточных автоматов обозначен диалект LISP — Nyquist. Проведён научно-технический анализ функционирования клеточных автоматов при генерации различных алгоритмическо-комбинаторных структур аудиоданных в соответствии с правилами Вольфрама — 30 и 90. Даны характеристики топологий поведения клеток элементарных клеточных автоматов в условиях получения новых аудиоматериалов. Воспроизведён эксперимент Данненберга-Льюис, характеризующий применение инструментальной базы LISP-ориентированных языков программирования в контексте получения новой хаотичной и фрактальной поведенческой фактуры аудиоматериалов. Уточнены технологические факторы воспроизведения эксперимента с учётом ускоренного развития вычислительной техники и стремительного роста научно-практической базы аудиоинженерии и компьютерных наук. Актуализированы аспекты внедрения полученных прикладных результатов с перспективой дальнейшего приращения научных знаний в научно-экспериментальный оборот современной аудиоинформатики.

Ключевые слова: эксперимент Данненберга-Льюис, клеточные автоматы, алгоритмическо-комбинаторные структуры аудиоданных, аудиопрограммирование, компьютерный аудиосинтез, хаотическое поведение, топологии фракталов, диалекты LISP, LISP-ориентированные языки, язык программирования Nyquist, Audacity®, правила Вольфрама, научный анализ, компьютерные науки, аудиоинженерия, аудиоинформатика.

DANNENBERG-LEWIS EXPERIMENT: CELLULAR AUTOMATA AND ALGORITHMIC-COMBINATORIAL STRUCTURE OF AUDIO DATA (SCIENTIFIC ANALYSIS, AUDIO PROGRAMMING, EXPERIMENT REPRODUCTION, AND FACTORS OF ITS REFINEMENT)

V. Taran

Summary. This paper addresses the practical application of the cellular automata principle for the formation of elementary and complex algorithmic-combinatorial structures of audio data. The concept of utilizing the structural-algorithmic functionality of cellular automata is scientifically substantiated and technically justified, implying the direct generation of a multiband spectrum through functional programming. The dialect of LISP—Nyquist—is identified as the primary functional programming tool for experimental capabilities of cellular automata. A scientific and technical analysis of the functioning of cellular automata in the generation of various algorithmic-combinatorial structures of audio data according to Wolfram's rules 30 and 90 is conducted. Characteristics of the topologies of elementary cellular automata behaviors in the context of obtaining new audio materials are provided. The Dannenberg-Lewis experiment is reproduced, illustrating the application of the instrumental base of LISP-oriented programming languages in the context of generating new chaotic and fractal textures of audio materials. Technological factors influencing the reproduction of the experiment are refined considering the accelerated development of computing technology and the rapid growth of the scientific-practical base of audio engineering and computer science. Aspects of implementing the obtained applied results are updated, with a perspective on further advancing scientific knowledge within the scientific-experimental framework of modern audio informatics.

Keywords: Dannenberg-Lewis experiment, cellular automata, algorithmic-combinatorial structures of audio data, audio programming, computer audio synthesis, chaotic behavior, fractal topologies, LISP dialects, LISP-oriented languages, Nyquist programming language, Audacity®, Wolfram's rules, scientific analysis, computer science, audio engineering, audio informatics.

Компьютерный аудиосинтез в системе редактирования цифрового звука (постановка проблемы, научный анализ и актуальность исследования)

Дизайн аудиокомпозиций — это довольно уникальная и сложная область инженерно-технической деятельности, затрагивающая все аспекты совершенствования образа аудиоматериала с целью его адекватного восприятия потенциальными слушателями.

Адекватность восприятия слушателем аудиокомпозиций определяется качеством записи звуковых дорожек[1], спектрографическим анализом многоком-

[1] Прим. автора. Качество записи аудиотреков напрямую влияет на адекватность понимания звуковой панорамы, поскольку ***математические*** и ***спектрометрические*** показатели определяют «***чёткость***»* представления каждого аудиофрагмента, формируемой инженером сведения аудиокомпозиции. *Понятие чёткости в аудиоинформатике имеет особый смысл и подразумевает, прежде всего, «***адекватность***» восприятия слушателем всех нюансов ***психоакустических*** свойств аудиоматериала, воспроизводимого цифровым аудиооборудованием. Понятие «***адекватность***» определяет первоначально закладываемые ***технические параметры звучания*** аудиоматериала в процессе его создания автором (композитором, саундпродюсером, звукорежиссёром, инженером мастеринга и сведения), говоря иначе, как автор подразумевает восприятие образа своего музыкального произведения, нацеленного на сторонних слушателей и своей потенциальной аудитории. Прежде всего, здесь стоит обратить внимание на массовое тиражирование аудиоматериалов. Потенциально верным считается тот вариант ***мастер***-трека, который при условно равных спектроакустических характеристиках будет звучать на различных системах воспроизведения одинаково. То есть ***спектроакустическая формула**** аудиоматериала после его инженерного редактирования в редакторах звукозаписи не потеряет своих ***спектральных*** свойств. Именно для этих целей в прикладной аудиоинформатике предусмотрены следующие понятия: «***разрушающее***» редактирование ***(предполагающее снижение битности и частоты дискретизации проекта в процессе его обработки внутри программы редактирования звука, как правило при манипуляциях с плагинами, имеющими разную разрядность 64-32 бит)*** и «***неразрушающее***» редактирование звуковых данных ***(не затрагивающее оригинальные проектные нормы****** структуры аудиоматериала, ***когда проектные спектрономические характеристики по разрядности и ‹ ч/д › совпадают с инструментарием для манипуляции над аудиоданными, например, виртуальный компрессор, поддерживающий обработку данных с глубиной 32 бит, обрабатывающий аудио-проект с теми же характеристиками)***. Важно помнить, что термины «***разрушающее***» и «***неразрушающее***» редактирование аудиоматериалов также подразумевают, ***в первом случае***:

— Обработку звука применительно к оригинальной звуковой форме — редактируемый аудиоматериал преобразуется как «оригинальный» ***образ*** без создания дополнительной ***мастер***-копии. При этом методе редактирования акустические свойства звуковой волны, представленной в виде отсчётов, трансформируются и видоизменяются сразу же после манипуляции заданной оператором (компрессия, эквализация, работа с фазой, панорамирование, изменение громкости и т.д.) и происходит ***первона-чальное*** изменение акустических свойств редактируемого аудиоматериала.

Во втором случае:

— Оригинальная звуковая форма и структура волны остаются ***неизменными***. Процесс редактирования аудиоматериала строится на концепции предварительного сохранения команд (операций) в неявном виде описываемых программой как сценарии, которые впоследствии будут интерпретированы процессором воспроизведения волновой формы. Например:

в первом случае — применяемые параметры компрессии изменяют спектральную структуру аудиосодержимого. Изменения затрагивают динамический диапазон, квантование амплитуды, дискретизацию во времени;

во втором случае — аналогичная операция выполняется последовательностью команд (характеризуемых техническими операциями), которые представляют собой события в виде сценариев, интерпретируемые алгоритмами программы в структуру звуковой формы. В случае с ***компрессией*** — события подразумевают описание сценариев по нормировке сигнала, учёта квантования и сжатию динамического диапазона.

*****Проектные нормы*** являются неотъемлемой частью процедур (***запись, редактирование, сведение, финализация, мастеринг аудиоматериала***) цифровой обработки аудиосигналов. Именно ***они*** во многом определяют ***качество*** потенциальной аудиокомпозиции. Сегодня в среде аудиоинженеров условно приняты следующие ***математические*** и ***спектрометрические*** показатели проектных норм: < ***16*** >, < ***24*** >, < 32 > — ***бит*** = ‹ ч/д › → < 8000, 11025, 16000, 22050, ***44100***, ***48000***, 88200, 96000 Гц … >. Оптимальными являются спектрометрические ***характеристики*** в соотношении: < ***16*** > = ***44100;*** < ***16*** > = ***48000*** — < ***24*** > = ***44100;*** < ***24*** > = ***48000.*** Баланс указанных ***характеристик*** наиболее предпочтителен для цифровых проектов, создаваемых в различных секвенсорах и программах цифровой обработки звука. Однако, что касается сложных проектов, где требуется идти по пути ступенчатого (обратного) снижения частоты дискретизации (при ***неразрушающем*** либо ***минимально*** разрушающем редактировании), предпочтительны: < 32 > — ***бит и*** < ***176400, 192000, 352800, 384000*** Гц … >. В последнем случае высокие ***физико-математические*** показатели сэмплирования фонограммы позволяют расширить глубину ***спектрального*** пространства, что, в свою очередь, преображает ***аудиообраз*** и делает более разборчивыми различные сегменты сведённой композиции.

** ***Спектроакустическая формула аудиоматериала (формула преобразования аудиосигналов) в аудиоинформатике — обобщённый термин, обозначающий*** совокупность ***физико-математических*** характеристик аудиоматериала, выражающихся в виде кластерной структуры, ячейки которой последовательно распределены. Каждая ячейка хранит запись (последовательность блоков, заголовки и т.п.) о фрагменте аудиоматериала. Математические описания, полученные в результате первичного аналого-цифрового преобразования, снабжаются ***метками***, участки которых присутствуют в начальных заголовках «сырой» структуры файла (двоичные таблицы, RAW-массивы), такие описания нужны для последующего ***внутреннего*** прочтения программой редактирования аудиозаписей. На основе подобных описаний строится формула многократных изменений волновой структуры звука в процессе его редактирования, а также ведётся статистический учёт видоизменения спектральных показателей. ***Метки*** позволяют ориентироваться программе-аудиоредактору при многократных запросах оператора на ***кодирование*** и ***перекодирование***

понентной *моно* и *стерео* базы, редактируемой аудиомультисессии[2], сведением[3], грамотным подбором спецэффектов[4], точной финализацией (премастерингом)[5] предобработанной аудиокомпозиции.

Дизайн (особенно в сфере цифровой обработки звука) как практико-ориентированный технико-технологический процесс, безусловно, должен включать в себя все *основные* вышеперечисленные *этапы* обработки звука и учитывать *индивидуальные* акустические свойства каждого элемента, присутствующего в общей акустической панораме аудиоматериала.

На сегодняшний день большинство аудиокомпозиций создаются при помощи традиционных компьютерных технологий[6] и даже в некоторых случаях задействуют информационно-коммуникационные технологии, по средствам телекоммуникационных опций, образуя специальные нейронные вычислительные сети (включая облачные вычисления, обслуживаемые алгоритмами искусственного интеллекта)[7] для достижения максимальных результатов в сфере композитинга аудиоматериала. Немаловажную роль в подобных процессах играет компьютерный аудиосинтез[8], способствующий созданию

в соответствующий формат (к примеру, Wave PCM, WMA, MP2, MP3, OGG и т.п., ***по определению*** автора, Таран В.В., 2026 г.).

[2] Прим. автора. Спектрографический анализ звука — это важное направление исследований аудиоинформатики, отвечающее за представление звука в виде синхронизированных слоёв, каждый из которых соответствует определённой частоте. Генерируемая аудиоредактором спектрограмма основывается на рядах Фурье, её калькуляция зависит от структуры алгоритма и программного кода используемого редактора, в стандартном виде она имеет следующие функции:

Спектральные характеристики аудиоматериала.

Волновые характеристики аудиоматериала.

[3] Прим. автора. Сведение — важный этап на пути создания ***оригинальной*** аудиокомпозиции. Именно этот процедурный этап обработки аудиоматериала отвечает за качество синхронности воспроизведения формируемой оператором сонограммы.

[4] Прим. автора. Спецэффекты — неотъемлемая часть формирования психоакустических, аудиальных образов, нацеленная на искусственное воспроизведение физических особенностей звучания различных ***органических*** и ***неорганических***, а также специфических — электромеханических источников, для придания достоверности тому или иному событию и воссоздания у слушателя атмосферы собственного присутствия и эмоциональной аутентичности в контексте прослушиваемого аудиопроизведения.

[5] Прим. автора. Финализация — предзавершающий аудиосессию процесс обработки аудиоматериала, направленный на усовершенствование звучания его спектральных характеристик для придания универсальности и точности спектро-частотного отображения на различных типах звукового оборудования. В практике аудиорежиссуры финализация является предзавершающим этапом обработки аудиоматериала, предшествующим процедурам мастеринга. Отличие ***финализации*** от ***мастеринга*** имеет ***тонкую*** грань, уходящую корнями в ***историю*** звукозаписи. Если не вдаваться в ***исторические*** подробности, то суть отличия этих ***взаимосвязанных*** процессов можно изложить ***следующим образом***:

Финализация (***премастеринг, миксдаун***) — набор технологических процедур, направленный на универсализацию звучания аудиоматериала на различных типах аудиооборудования. Данная процедура базируется на эквализации, компрессии и выравнивании громкости звука. Параметры универсализации будут неконечными, то есть, эквализация и компрессия будут иметь неконечные параметры. Например, порог атаки компрессора будет ниже, чем при мастеринге. Это обуславливается тем, что каналы реализации ***мастер***-трека (***мастер***-копии) могут быть разными. В радиовещании есть свои стандарты, на телевидении свои, у интернет-релизов свои. Поэтому необходимо, чтобы ***мастер-***трек можно было гибко перенастроить, без изменения ***характера*** звучания выполнить ***мастеринг*** (***по определению автора,*** Таран В.В., 2026 г.).

Мастеринг — ***конечная стадия обработки аудиоматериала,*** имеющая ***фиксированные*** значения приборов его обработки. Вносить изменения в ***мастер***-трек после её процедуры не рекомендуется, так как это может считаться ***разрушающим*** редактированием. Данная процедура характерна для создания полноценных ***мастер***-треков различных релизов, а также для записи и тиражирования ***мастер***-копий CD и DVD-аудио и проектов, связанных с нарезкой виниловых дисков (***по определению автора,*** Таран В.В., 2026 г.).

[6] Прим. автора. Под традиционными компьютерными технологиями понимаются классические алгоритмы современных программных средств нелинейного редактирования аудиоматериалов.

[7] Прим. автора. Нейронные сети, облачные технологии и искусственный интеллект в настоящее время открывают широкие возможности для манипуляции звуковыми данными, однако во многих случаях автоматизированная обработка аудиоматериалов не всегда даёт качественные результаты. Особенно это касается аудиоматериалов, имеющих сложную стилистическую направленность и нестандартные спектро-частотные характеристики*. Современные нейросетевые средства обработки звука, основывающиеся на передовых алгоритмах искусственного интеллекта, пока не способны в полной мере заменить ***человеческий слух***, поскольку многие частотные артефакты носят сугубо индивидуальный характер. Это затрудняет выполнение реструктуризации панорамы частотно-балансного анализа в автоматическом режиме. Уникальность частот аудиокомпозиции зависит от многих факторов: это и тембральность, и насыщенность гармониками, и баланс тоново-***формантных*** значений и т.д. Многое из этого перечня обрабатывается именно на слух и зависит от индивидуальных предпочтений инженера сведения. Процесс распознавания частот человеческим слухом в естественной среде существенно отличается от алгоритмов имитации «искусственного прослушивания» и анализа тех или иных акустических аспектов редактируемого звука. В силу вышеперечисленных причин можно утверждать, что оператор звука по-прежнему остаётся ключевым звеном в системе ***человеко-машинного*** взаимодействия, по крайней мере, до тех пор, пока не будут получены ***достоверные*** результаты научных исследований о полноте работы человеческого мозга.

* Под нестандартными ***спектро-частотными характеристиками*** автор имеет в виду аудиоматериал со смешанным содержимым, сведение которого подразумевает от 30 и более звуковых дорожек.

[8] Прим. автора. Выбор категории компьютерного аудиосинтеза для определённых задач, связанных с комбинаторикой звуковых данных, напрямую зависит от конкретных ситуаций. К примеру, аналаговый аудиосинтез может использоваться при моделировании аналого-электретных цепей для генерации звука и воссоздания классических (аппаратных) синтезаторных звуков — синтезированных музыкальных инструментов, различных

уникальных *звуков* и *мелодий,* позволяющих наполнять традиционные звукозаписи новыми *композиционными элементами,* придающими оригинальность характеру звучания аудиопроизведения.

Аудиосинтез является неотъемлемо важным элементом системы компьютерного дизайна аудиокомпозиций и составления *алгоритмическо-комбинаторных* волновых структур, поэтому алгоритмы его совершенствования в *обозримой перспективе* будут оказывать серьёзное влияние на аудиопроизводство. Синтез преобразованных[9] звуковых волн также существенно расширяет арсенал технических средств, предназначенных для *ремастеринга* аудиоматериалов[10]. Существуют разные методы, позволяющие получить синтезированные звуковые формы (образцы), которые впоследствии можно использовать для формирования электронных аудиокомпозиций. В данной рукописи будет проанализирован один из довольно *редких методов*[11], который позволит *расширить* традиционное представление о современном аудиосинтезе. Речь пойдёт о клеточных автоматах[12]. Клеточные автоматы (**C**ellular **A**utomata, сокращённое наименование на английском языке и по тексту статьи CA)[13] — это дискретные абстрактно-логические вычис-

аудиальных эффектов. В то время как ***интервальный*** (фазовый) аудиосинтез может использоваться для генерации звуковых комбинаций, контролируя фазу и амплитуду аудиосигналов, что позволяет создавать сложные (***неоднородные***) звуковые массивы. Область их применения может быть разной: от создания динамических аудиоподкладов, до формирования целостных полифонико-синтетических аудиопроизведений.

[9] Прим. автора. Имеется в виду волновая ***синтез***-структура, преобразованная в мультичастотный аудиосигнал посредством аналого-цифрового преобразования, АЦП. Сегодня науке известны различные виды компьютерного синтеза для работы с аудиоданными, к примеру, можно выделить следующие категории: аддитивный синтез, субтрактивный синтез, таблично-волновой синтез, гранулярный синтез, спектральный синтез и т.п.

[10] Прим. автора. В последнее время в кругу учёных, занимающихся теоретическими и прикладными проблемами аудиоинформатики, наметилась интересная тенденция, демонстрирующая поиск новых более совершенных методов обработки и представления аудиосигналов. В первую очередь, эта тенденция затрагивает те аспекты аудиоинформатики, которые касаются дизайна звуковых форм. Тенденция во многом сформировалась под влиянием совершенствования различных видов вычислений и введения в практический оборот ***полифункциональных*** инструментов обработки данных, основывающихся на нейронных технологиях и искусственном интеллекте (более точно — ***интеллектуальной автоматизации***). В этой связи все большее число исследователей стали обращать внимание на ***альтернативные*** способы обработки аудиоданных. Не осталась без внимания и часть аудиоинформатики, отвечающая за прикладные средства обработки звука, такие как: мастеринг, ремастеринг, реконструкция спектроакустических форм аудиоматериалов, спектральная реструктуризация однородной файловой структуры с неоднородным содержимым. На все перечисленные средства в современных условиях влияют технологии звукового синтеза и ресинтеза.

[11] Прим. автора. Само понятие ***клеточных автоматов, их принцип и структура*** известны мировой науке довольно давно. Однако, что касается использования правил клеточных автоматов в качестве инструмента для производства аудиосинтеза, предполагаемого различные комбинации поведения сигналов при формировании ***новых*** импульсно-волновых структур, а также их физического применения средствами программно-технической реализации (***программирование***, ***разработка функционально-алгоритмической схемотехники***) для аппаратных вычислительных систем (ABC), то в этом аспекте ***научный анализ*** практически ***отсутствует***, за исключением некоторых отдельных работ, находящихся в сфере ***корпоративной науки***.

[12] Прим. автора. Как ***принцип*** (организационная структура, алгоритм, включая правила поведения) — «***клеточные автоматы***», как набор элементарных действий — «***клеточный автомат***».

[13] ***Справочно:***

Первооткрывателем структуры клеточных автоматов принято считать математика Джона фон Неймана (John von Neumann) в своё время (1951 г.) начавшего работать над теорией «самовоспроизведения». Он не оставлял попыток представить научной общественности редукционистскую теорию (упрощённое понимание изучения различных объектов в биологии с точки зрения физико-химических свойств и процессов) биологического развития, которая, по его задумке, должна была воспроизводить точные копии самой себя. Его коллега Станислав Улам (Stanisław Marcin Ulam) предложил Джону фон Нейману сфокусировать своё внимание на дискретной двумерной системе, которая и привела учёного к открытию клеточных автоматов. Фактически Джону фон Нейману удалось создать первую в истории дискретно-параллельную вычислительную модель, способную эмулировать универсальную машину Алана Тьюринга, которая была призвана производить различные рекурсивные вычисления. Позднее над проблемами клеточных автоматов работали следующие видные исследователи: Э́двард Фо́рест Мур (Edward Forrest Moore), Джон Р. Майхилл-старший (John R. Myhill Sr), Анил Нерод (Anil Nerode), Густав А. Хедлунд (Gustav Arnold Hedlund), Джон Хо́ртон Ко́нвей (John Horton Conway), Томмазо Тоффоли (Tommaso Toffoli). Работы Стивена Вольфрама (Stephen Wolfram)* в 1980-х годах способствовали тому, что растущее сообщество последователей CA появилось на научной карте. В серии работ Вольфрам подробно исследовал одномерные CA, предоставив первую качественную систематику их поведения и заложив основу для дальнейших исследований. Вольфрам предположил, что конкретное ***правило перехода*** для одномерного CA, известное как «***правило*** 110», является ***универсальным***. Через некоторое время после высказывания данной гипотезы Мэтью Кук (Matthew Cook) доказал, что ***правило*** 110 поддаётся ***универсальным*** вычислениям: ***см***. Cook M. Universality in Elementary Cellular Automata // Complex Systems 15.2004, pp. 1-40. и Wolfram S. A New Kind of Science. — Champaign, IL: Wolfram Media, 2002. — 1197 p.

* Джон фон Нейман (John von Neumann) — венгерский и американский математик, физик, философ науки и техники. Работы Джона Неймана во многом определили структуру современной науки, поскольку его научно-теоретические изыскания и аргументированные обоснования нашли серьёзное отражение в полидисциплинарных исследованиях нашего времени. Его пионерские работы оказали значительное влияние на такие отрасли науки, как биология (нейробиология — представления о сходстве человеческого мышления и логике компьютерных программ, гипотезы о строении человеческой памяти), физика (алгебраические обоснования квантовой механики), математика (разработка теории алгебр операторов), информатика (участие в разработке компьютеров ENIAC и EDVAC, численные методы) и экономика

лительные системы, которые потенциально полезны как в качестве общих моделей *математической сложности*, так и в качестве более конкретных представлений *нелинейной* динамики в *различных* научных областях[14].

Во-первых, клеточные автоматы (как правило) пространственно и временно дискретны. Они состоят из конечного или неисчислимого набора однородных простых единиц, *атомов* или *клеток*. В каждой единице времени, ячейки создают один экземпляр из конечного набора состояний. Они развиваются параллельно и выражаются в дискретных временных шагах, следуя функциям обновления состояний или *правилам* динамического перехода[15]. Обновление состояния ячейки происходит с учётом положения ячеек в её *локальной* окрестности (следовательно, никаких действий на расстоянии не предусматривается).

Во-вторых, клеточные автоматы *абстрактны*, они могут быть заданы в чисто математических и символьно-алгебраических выражениях, а физические структуры могут их реализовать.

В-третьих, клеточные автоматы — это вычислительные системы, которые способны находить функции и решать сложные алгоритмические задачи. Клеточные автоматы могут имитировать универсальную машину Алана Тьюринга (Alan Turing)[16]. Для этого необходимо подобрать для них наиболее подходящее правило, которое может выполнять *прямое* и *параллельное* вычисления в соответствии с критерием *тезиса* Чёрча-Тьюринга[17].

(модель общего экономического равновесия).

** Стивен Вольфрам (Stephen Wolfram) — британский учёный, физик-математик, изобретатель, специалист в области теоретической информатики. Стивен Вольфрам конкретизировал многие теоретические наработки Джона фон Неймана и других уважаемых деятелей науки инженерии в области рекурсивных вычислений (***самовычислений***), переведя многие вопросы теоретической математики в русло практической плоскости. Стивен Вольфрам представляет математику как некий метаязык, описывающий и устанавливающий различные физические и метафизические закономерности с помощью определённых алгебраических методов. Его работы в области клеточных автоматов подтверждают вышеуказанную характеристику. Клеточные автоматы одно из направлений, которое активно развивает Стивен Вольфрам. Элементарные клеточные автоматы, особенно правило 30, стали для учёного визитной карточкой в области информатики, поскольку клеточные нейротехнологии хорошо вписываются в современный научно-практический уклад развития киберфизических систем, опорным базисом для которых являются в том числе нейронные клеточные автоматы, представляющие собой гибридные вычислительные модели, в которых каждая ячейка автомата является нейроном и способна изменять своё состояние на основе состояний соседних клеток по определённым правилам.

*** Мэтью Кук (Matthew Cook) — американский логик и математик, специалист в области информатики. Известен своими публикациями по проблемам нейрокомпьютинга, аргументировал гипотезу Стивена Вольфрама о том, что клеточный автомат ***правило*** 110 является полным по А. Тьюрингу.

[14] Прим. автора. Имеется в виду принцип клеточных автоматов в нелинейных динамических системах, где под динамическими системами подразумевается абстрактная структура любой природы: техническая, информационная, физическая, химическая, биологическая, социальная, экономическая и т. п. В качестве описания моделей нелинейной динамики выступают дифференциальные уравнения и дискретные вычисления. В отличие от линейной системы, в которой небольшое изменение одной переменной приводит к небольшому систематическому изменению, нелинейная система демонстрирует чувствительную зависимость от начальных условий — небольшие различия в начальных условиях могут привести к совершенно разным результатам.

[15] Прим. автора. Правило динамического перехода в структуре работы клеточных автоматов представляет собой заранее заданное правило, которое определяет изменение состояния каждой клетки в следующий момент времени в зависимости от локации клетки и расположения её ближайших соседей. Правила динамического перехода могут быть детерминированными, вероятностными и обобщающими. ***Детерминированные*** — состояние ячейки в последующий момент времени однозначно определяется состоянием этой ячейки и её ближайших соседей в предыдущий момент времени. ***Вероятностные*** — задаётся вероятность, что на следующем шаге клетка сменит свой цвет на другой. Обычно правила перехода одинаковы для всех клеток, что обеспечивает однородность системы. ***Обобщающие*** — правила, зависящие только от общего числа значений соседних ячеек.

[16] Универсальная машина Алана Тьюринга* (universal Turing machine, UTM) — это унифицированная абстрактно-вычислительная машина обработки данных, концептуально предназначенная для калькуляции различных ***вычислимых*** последовательностей. Гипотетически универсальная машина Алана Тьюринга базируется на постулате о том, что если поставленная оператором задача может быть решена алгоритмическим путём, то такую задачу может решить и универсальная машина Алана Тьюринга, получив сценарий, либо программу, в виде символьно-числовых комбинаций для решения этой задачи в качестве входных (ввод → структура ↔ ввод) данных. Алан Тьюринг предложил концепцию универсальной вычислительной машины в 1936 году, его подход во многом определил и формализовал понятия «алгоритм» и «структура данных», применяемые в современной информатике (и более широком понимании — компьютерных науках). Универсальная машина Алана Тьюринга также объединяет следующие понятия: переменная машина А.Тьюринга, нейронная машина А.Тьюринга, недетерминированная машина А.Тьюринга, квантовая машина А.Тьюринга, вычислительная машина ***пост***-Тьюринга, вероятностная машина А.Тьюринга, многодорожечная машина А.Тьюринга, многоленточная машина А.Тьюринга, однозначная машина А.Тьюринга, универсально-вычислительная машина А.Тьюринга, Машина Zeno (ускоренная машина А. Тьюринга).

* Alan Mathison Turing (Алан Мэтисон Тьюринг) — всемирно известный британский учёный, математик, криптоаналитик и логик, кавалер ордена Британской империи. С именем А. Тьюринга связана целая эпоха становления мировой теоретической информатики (компьютерных наук), его работы в области алгоритмизации и криптографии на долгое время определят вектор развития мультифункциональной вычислительной техники. В честь Алана Тьюринга учреждена ***самая престижная*** по значимости премия в области информатики (***компьютерных наук***) — премия Тьюринга.

[17] Тезис Чёрча*-Тьюринга (***гипотеза вычислимости***) — фундаментальное логико-математическое утверждение, гипотети-

Это обстоятельство позволит компьютерным аппаратам производить абстрактно-логические вычисления с целью «*вычислить — вычислимое*»[18]. Вычислительные возможности клеточных автоматов имеют большой потенциал в сфере экспериментальной информатики. Они функционально обогащают сразу несколько её базовых разделов — *алгоритмизацию структур данных*, *объектную символьно-числовую комбинаторику*, *топологию структурных ветвлений,* выводя область *профессионального программирования* на более *продвинутый уровень*. Важно обратить внимание также на то, что принцип клеточных автоматов является устойчивым мостиком, соединяющим две дисциплины: *информатику* и *математику*, открывая дальнейшие перспективы по приращению полидисциплинарных научных знаний в таких областях исследований, как алгебраическая комбинаторика и логика, теория множеств и математическая лингвистика.

Оригинальный принцип клеточных автоматов заключается в том, что они демонстрируют сложное эмерджентное[19] поведение, начиная с *простых* автоматов, подчиняясь *простым* локальным правилам (правилам перехода)[20]. Именно поэтому СА привлекают все большее число исследователей, занимающихся *когнитивными, техническими* и *естественнонаучными* дисциплинами, желающих изучать формирование различных топологий структур во всей их сложности в чисто абстрактной атмосфере.

Клеточные автоматы и аудиоинформатика: пути взаимодействия и современное состояние практик СА-программирования

В *аудиоинформатике* клеточные автоматы рассматриваются как *отдельный вид аудиосинтеза*, с помощью которого можно создавать новые *органические*[21] и *неор-*

чески обусловливающее тождественность между терминами алгоритмическая вычислимость и такими формализованными понятиями, как функция машины А.Тьюринга и частично-рекурсивная механическая функция.

*Алонзо Чёрч (Alonzo Church) — американский учёный-математик и логик, оказавший огромное влияние на развитие функциональной парадигмы программирования благодаря разработке теории ***лямбда***-вычислений. Ввёл понятие «β-редукции» (***каррирования***[**]) рекурсивных вызовов функций с критериями «α-конверсия» и «η-конверсия», а также доказал неразрешимость проблемы для исчисления предикатов. ***Лямбда***-вычисления благодаря операциям ***абстракция/аппликация*** стали основой для развития систем интеллектуальной автоматизации и машинного искусственного интеллекта.

Каррирование в информатике — это преобразование ***функции*** с ***множеством аргументов*** в набор ***вложенных функций*** с ***одним*** аргументом.

[18] Прим. автора. Имеется в виду повышение эффективности расчётов с применением ЭВМ (в многозадачном аспекте — компьютера), основанных на передовых математических алгоритмах и символьных операциях, способных значительно снизить обращения к аппаратным ресурсам компьютера в процессе сложных вычислений. Иными словами организовать аппаратно-независимые разветвлённые вычисления, для того чтобы обойти аппаратные ограничения компьютера. Это та самая ситуация, когда грамотно составленный, чётко выстроенный и логически правильно изложенный аппаратно-независимый алгоритм* программы (особенно неявного типа) способен продемонстрировать более эффективные результаты по сравнению с аппаратно-зависимыми исчислениями.

*В теории информатики (компьютерных наук) особенно в англо-говорящих странах такие алгоритмы принято называть «квалифицированными алгоритмами», в России вместо этого понятия в некоторой научно-технической литературе можно встретить аналогичный термин «экспертные алгоритмы». Для эффективности функционирования таких алгоритмов важны два условия:

1. Алгоритм должен иметь чёткую иерархию и разветвлённо-соподчинённую структуру.

2. Описание такого алгоритма и его точная интерпретация компьютером должны основываться на продуманном выборе языка программирования с учётом его класса и функциональных возможностей, а также на краткости изложения неявных событий его программного кода. Кстати, здесь уместно вспомнить о знаменитом четырёхтомнике (фактически пятитомнике 4А, 4В) Д.Кнута (Donald Ervin Knuth) «Искусство компьютерного программирования» «The Art of Computer Programming». В русскоязычном официальном переводе «Искусство программирования», который сегодня явно недооценивают многие современные IT-специалисты, высказывая критику, что данная рукопись очень запутана и не даёт читателю представления о современном программировании. Можно много дискутировать на эту тему, но факт есть факт, что основной посыл данной книги —это искусство грамотного изложения кода в соответствии с семиотической топологией, логикой и алгебраическими представлениями. Также профессор Д.Кнут уделяет особое внимание анализу кода перед его дальнейшей ***компиляцией***, считая эту процедуру наиболее важной для эффективной интерпретации алгоритма и раскрытия всего его ***функционального*** потенциала.

[19] Прим. автора. Эмерджентность — качество, свойства системы, которые не присущи её элементам в отдельности, а возникающие благодаря объединению этих элементов в совокупную, целостную систему.

[20] Прим. автора. Простые локальные правила (***правила перехода***) — это алгоритмические схемы, которые идентифицируют обновлённое состояние ячейки (клетки) в следующий момент времени в зависимости от состояния ячейки и окружающих её ближайших соседей. Такие правила обычно описываются специальной картой* переходов, на которой клетка определяет свою новую локацию на каждом минимальном отрезке времени, основываясь на своем местоположении и состоянии положения соседей. Карта переходов содержит информацию о функциях перехода клеточного автомата, в соответствии с которыми клетка имеет возможность вычислить своё новое состояние, основываясь на собственной топологии перемещения и состоянии соседних клеток.

[21] В аудиоинформатике (***технический аспект***) и теории компьютерной музыки ***органические*** звуки —это отдельные акустические импульсы либо сигнально-ритмические комбинации естественного происхождения. Примечательно, что источником звука может выступать любой биологический организм или любой объект естественной природной среды, окружающей его. Интересно, что к числу таких звуков относятся практически любые ана-

ганические[22] звуки и даже целые оригинальные электронные аудиокомпозиции. Переходя от *общего* к *частному,* следует отметить, что применение *принципа* клеточных автоматов очень многогранно, и оно отнюдь не очерчивается профилем *естественных* и *технических* наук, клеточные автоматы можно и нужно изучать с *мультидисциплинарной* точки зрения. Важность изучения клеточных автоматов диктуется и современной технологической повесткой, которая предполагает перевод теоретических наработок из сферы информационно-кибернетических нейровзаимосвязей в *практическую область* их применения компьютерными системами обработки данных. В нашем случае практическая плоскость, предусмотренная подобным подходом, ограничивается *инструментальными средствами информатики*, а именно *языками программирования*. Полноценной описательной практики СА-*программирования*[23] на сегодняшний день крайне мало, практического её применения в области *аудиоинформатики* ещё меньше. Размышляя о причинах такого неудовлетворительного положения дел в сфере СА-*программирования* как в России, так и за рубежом, автор настоящей рукописи полагает, что причины могут здесь крыться в *следующем*:

— В отсутствии научных школ в области информатики (компьютерных наук — computer science), формирующих практико-ориентированные представления о программировании как в искусстве составления высокоэффективных компьютерных программ. Такие школы, безусловно, обеспечат вертикальную преемственность при смене поколений исследователей и инженеров-программистов, формирующих фундаментальные теоретико-прикладные основы новейших (уникальных) знаний в таких разделах информатики, как общая алгоритмизация и алгоритмические структуры данных, принципы построения логико-вычислительных систем, разработка программных структур на основе строковых и графовых алгоритмов с применением динамического программирования.

— В крайне невысокой степени взаимоинтеграции исследовательской базы корпоративной науки (корпоративной информатики) и информатики общего профиля (гражданские научно-технические разработки и исследования).

— В интеллектуальном разрыве и искусственной демаркации научного потенциала между информатикой и кибернетикой, а также WEB-наукой. Бессистемным и хаотичным обменом исследовательского опыта теории и практики в междисциплинарных разделах данных наук.

— В исключительности и оригинальности характера исследований в области прикладной информатики (компьютерных наук), инструментально-вычислительных наук, заключающейся в особой сложности, которая выражается в уникальной специфике работы с аналоговыми и цифровыми данными.

— В закрытости подходов в области коммерческого программирования (компьютерная тайна), а также передовых техник составления сложно-структурированных алгоритмов.

Вышеперечисленные проблемы и низкая степень интеграции различных научных дисциплин в информатику (компьютерные науки) делают заявленную автором тему *особо* актуальной.

Клеточные автоматы как вид экспериментального аудиосинтеза, методы реализации и определение инструментальных средств составления программы

Область авторского исследования — компьютерная *аудиоинженерия* и *аудиоинформатика*. Они, безусловно,

логовые электромагнитные колебания в естественном виде (т.е. не прошедшие процедуры цифро-аналогового преобразования). Таким образом, к органическим звукам можно отнести не только шум дождя, пение птиц и т.д., но и традиционные музыкальные инструменты: духовые, клавишные, струнные и т.п.

[22] Прим. автора. К ***неорганическим*** звукам (***технический аспект***) относятся звуки искусственного (неживого) происхождения. Как правило, это синтетические звуки, получаемые в результате компьютерного синтеза, либо образцы для такого синтеза, прошедшие процедуры аналого-цифрового преобразования. К таким звукам можно отнести генеративные музыкальные импульсы синтезатора (на аналоговой и цифровой схемотехнике), фоновые шумы компьютерной схемотехники, различные механические звуки нерукотворного происхождения. Примечательно, что любые шумы стационарного и нестационарного типа могут быть как ***органическими***, так и ***неорганическими***. В теории музыки иногда такие звуки называются ***натуральными*** и ***ненатуральными***.

[23] Как и в любом разделе информатики (***компьютерных наук***), область которого выходит за рамки фундаментальных теоретических исследований и логических обоснований предусмотрена некая ***инструментальная база,*** являющаяся ответвлением от ***классических*** представлений о закономерностях обработки информации. Как правило, это прикладные разработки и практико-ориентированные исследования. Эти сферы всегда имеют определённую специфику, выражающуюся в конкретном предназначении разрабатываемых исследователем объектов, впоследствии существенно влияющих на развитие и приращение знаний в предметной области. В информатике такой базой служат ***языки программирования***, они играют роль ***посредника*** в технической коммуникации между машиной и оператором, позволяя эффективно решать поставленные задачи. Инструментальная база разделов информатики (***информатика по отраслям деятельности***) также имеет свою внутреннюю специфику, подразумевающую определённый характер действий с оттенком на предметные задачи. Аудиоинформатика как раздел информатики (***компьютерных наук***), а по логике автора уже давно претендующая на научно-отраслевую автономию, здесь не исключение. Традиционно виды алгоритмов, используемых для обработки данных, имеют разные ответвления, поэтому отдельные широкопрофильные технико-применительные практики программирования обычно именуются базовым алгоритмом. В нашем случае это клеточные автоматы (***c***ellular ***a***utomata, CA), поэтому СА-***программирование***.

являются наглядными примерами разветвлённых[24] *полидисциплинарных* исследований.

Компьютерный синтез аудиоматериала целиком зависим от различных методов и алгоритмов, применяемых для его создания, клеточные автоматы могут быть органично встроены в его структуру. В теоретическом плане имеется достаточное количество научных работ, посвящённых данному направлению. Что же касается практики, то здесь дела обстоят хуже. Теоретические наработки в области СА требуют тщательного подбора программно-инструментальных средств, которые, в свою очередь, должны сочетаться с алгебраической комбинаторикой, что в настоящее время является редкостью. Изобрести механизм *трудно*, заставить его работать *ещё труднее*. Исходя из этой *формулы,* автор настоящего исследования сосредоточил своё внимание *исключительно* на работах, описывающих организационно-функциональную структуру клеточных автоматов. К сожалению, таких работ *крайне мало*, несмотря на то, что подобные исследования *существенно* обогащают *прикладную* информатику[25]. Одним из пионеров СА-*программирования* в области аудиоданных, успешно реализовавших подобный метод синтеза аудиоматериала на практике, является Энн Льюис[26] (Ann Lewis). Она одна из немногих экспериментаторов, кто в 2002 году под руководством профессора Р. Данненберга[27] (Roger B. Dannenberg) в школе компьютерных наук (School of Computer Science) университета Карнеги-Меллон (Carnegie Mellon University, CMU/USA) поставила компьютерный эксперимент и применила LISP-модель, описывающую метод клеточных автоматов для производства новой фактуры аудиоматериала. Этот практический эксперимент очень важен для развития прикладной аудиоинформатики, поскольку он даёт возможность формировать новые уникальные аудиоструктуры.

Рис. 1. Логотип частного американского университета Карнеги-Меллон, в котором работает одна из сильнейших мировых школ в области компьютерных наук[28]

Источник: Коллаж составлен автором на основе *оригинального* логотипа*, Таран В.В. — 2026 г.

**Логотип является собственностью университета Карнеги-Меллон.*

[24] Прим. автора. Под разветвлёнными полидисциплинарными исследованиями подразумевается анализ, основанный на широком спектре научных дисциплин, тематически связанных между собой. Разветвлённые полидисциплинарные исследования характеризуются предметными связями трёх и более дисциплин различной классификации. Установка взаимозависимости дисциплин формируется под воздействием тематики исследования. Хочется отметить, что ключевым отличием полидисциплинарных (в некоторых случаях трансдисциплинарных научных исследований) от разветвлённых полидисциплинарных исследований является взаимоинтеграция гуманитарных (общественных), естественнонаучных и технических дисциплин. В то время как традиционные полидисциплинарные исследования классификационно привержены к близким по профилю дисциплинам. В мировой научной периодике такой научно-исследовательский подход именуется как «интегративный». Хорошим примером здесь служит аудиоинформатика, которая, в прикладном понимании зиждется на точных и технических науках, но в теоретическом плане черпает знания также из естественнонаучных и гуманитарных направлений.

[25] Прим. автора. Речь идёт об исследованиях в области информатики, раскрывающих принципы работы того или иного интеллектуального механизма с расшифровкой функциональных операций, характеризующих структуру программного кода.

[26] Прим. автора. Энн Льюис (Ann L. Lewis) — преподаватель школы-факультета компьютерных наук университета Карнеги-Меллон. Энн Льюис начинала свои научные изыскания как студентка факультета компьютерных наук университета Карнеги-Меллон, одновременно получая знания в области музыки при школе музыки Карнеги-Меллон (School of Music — Carnegie Mellon University). Круг её научных интересов очерчивается знаниями в области создания синтезированных ритмических аудиопартий, обработки звука, преобразования текстовых массивов в синтезированную речь. Принимала активное участие в различных конференциях по компьютерной обработке звука. Совместно с Роджером Данненбергом являлась участницей конференции по международной компьютерной музыке (International Computer Music Conference, ICMC — 2002).

[27] Прим. автора. Роджер Данненберг (Roger B. Dannenberg) — американский учёный и преподаватель в области компьютерных наук (информатики), занимающийся разработкой перспективных направлений в сфере обработки звука программно-техническими средствами на базе школы-факультета (School of Computer Science — Computer Science Department) компьютерных наук университета Карнеги-Меллон (Carnegie Mellon University). Роджер Данненберг специализируется на разработке алгоритмов распознавания различных сигнальных комбинаций, включая человеческую речь, музыкальные партии, звуки различной природы. Отдельное внимание учёный уделяет системам обработки аудиоданных, в частности, языкам программирования. Является идейным вдохновителем и создателем языка программирования Nyquist, одним из проектировщиков знаменитого аудиоредактора Audacity®, участвует в различных проектах по созданию компьютерной музыки. Имеет учёную степень PhD в области компьютерных наук (информатики). Является членом института инженеров электротехники и электроники (Institute of Electrical and Electronics Engineers, IEEE), ассоциации вычислительной техники (Association for Computing Machinery, ACM), международной ассоциации компьютерной музыки (International Computer Music Association, ICMA).

[28] Прим. автора. Под технико-манипулятивными проектами в аудиоинформатике понимаются те проекты, при обработке которых необходимо динамически изменять структуры аудиосигнала. Это проекты, на обработку которых требуются значительные ресурсы с точки зрения техник составления алгоритмов, динамически обрабатывающих аудиосигнал. К примеру, таким проектом может служить аудиоморфинг, когда одна часть аудиосигнала постепенно видоизменяется, превращаясь в другую по спектро-частотным характеристикам, но при этом не затрагивает величину сдвига фазы сигнала, девиация фазы, то количество радиан, которое является оптимальным для общего аудиосигнала в не зависимости от участка перехода. Интерполяция параметров такого проекта основывается на трёх манипуляционных техниках: синусоидальное моделирование, оконное преобразование Фурье и кодирование с линейным предсказанием.

Применение данного метода убедительно показало насколько оригинальной и полифоничной может быть фактура мелодии, получаемая в результате соединения фрактальных структур.

Энн Льюис реализовала синтез фрактальных структур как учебный проект на языке программирования Nyquist. И тому есть несколько причин:

1) Nyquist является специализированным языком аудиопрограммирования, позволяющим реализовывать сложно-функциональные высоконагруженные технико-манипулятивные проекты в области звука[29].

2) Nyquist успешно применим в качестве демонстрационного языка программирования.

3) Nyquist обладает развитым техническим потенциалом и использует мощный арсенал операционных средств и технологий, унаследованных от его функциональных прародителей, прежде всего, Arctic[30], Canon[31], Fugue[32], LISP[33]/XLISP[34] и SAL[35]. Вспомогательной исполни-

[29] Прим. автора. Единственным ограничением Nyquist является его работа с нагруженными алгоритмами обработки звука в реальном времени. Обусловлено это тем, что архитектура языка программирования Nyquist разрабатывалась для обработки аудиоматериалов в оффлайн режиме, не стоит забывать, что Nyquist это диалект LISP, поэтому интерпретация фрагментов его кода в режиме настоящего времени вносит задержки, которые могут сказываться на характере обработки сигнала. Задержки могут быть связаны с длительностью интерпретации выражений, а также непредсказуемости времени, затрачиваемого на обработку аудиоматериала из-за сборки мусора и динамической типизации. Этот язык ориентирован на пост обработку аудиоданных и силён именно в инженерных практиках анализа, синтеза сведения и мастеринга аудиотреков. К сожалению, в Audacity® для Nyquist отсутствует низкоуровневый доступ к аудиопотоку, в результате чего язык не имеет доступа к буферу обрабатываемого трека, что, в свою очередь, затрудняет поддержку ***callback***-процессинга для фрагментарной обработки сигнала. Однако эту проблему можно обойти с помощью ограничения инженерных сценариев простыми операциями (фильтрация, работа с огибающими) и использовать буферы малого размера. Эта тема будет обязательно раскрыта в следующих публикациях автора.

[30] Прим. автора. Arctic — это специализированный, декларативный язык программирования, разработанный в Университете Карнеги-Меллона (США) для описания систем реального времени с параллельными процессами. Язык разработан Дином Рубином (Dean Rubine) и Роджером Данненбергом (Roger B. Dannenberg). Целевое назначение языка — это управление звуковым синтезом (генерация музыки), сетевое синхронное создание звуков тревоги и оповещения на различных промышленных объектах, в том числе атомных электростанциях, синхронизации звука по времени в климатических системах и станциях слежения за погодой. Сейчас по факту является историческим языком программирования, давшим питательную почву для различных узкоспециализированных языков программирования обработки аудиоматерилов.

[31] Прим. автора. Canon — это полноценный Lisp-ориентированный, абстрактно-декларативный, специализированный язык программирования и одновременно нотация для музыкальных партитур. Язык изначально создавался как инструмент музыкального программирования для экспериментов и обучения студентов, однако впоследствии зарекомендовал себя как мощное средство по управлению данными сигнальных массивов в аппаратных синтезаторах и вокодерах, а также для работы в MIDI-секвенсорах для создания и корректировки партитур, управляемых MIDI-сигналами. Язык не имеет отдельного интерпретатора, а партитуры оцениваются в ***окружении***, которое влияет на ноты и может изменяться преобразованиями. Для этого используются глобальные переменные для имитации динамической области видимости ***окружения*** (например, *time*, *dur*, *velocity*).

[32] Прим. автора. Fugue — специализированный функциональный Lisp-ориентированный язык программирования для формирования состава музыкальной композиции, синтеза и управления звуком. Язык позволяет описывать алгоритмы обработки сигналов, музыкальные партитуры и высокоуровневые музыкальные процедуры в едином составе. Поддерживает поведенческую абстракцию, выражающуюся в управлении сложными музыкальными структурами. Реализован на комбинации C и XLisp для Unix-рабочих станций. Необходим для описания сложных музыкальных структур и алгоритмов синтеза звука, а также создания и манипулирования звуковыми объектами как абстрактными, неизменяемыми сущностями.

[33] Прим. автора. LISP (LISt Processing language, LISP, Lisp) — легендарный (один из старейших языков программирования широкого профиля, унаследовавших IPL*-структуру) язык программирования, основанный на принципе обработки списков. На языке LISP программы, сценарии, а также данные представляются как списки, состоящие из элементов любой сущности. LISP базируется на ***лямбда***-исчислении А.Чёрча соответственно функции — это полноправные объекты этого языка. LISP поддерживает макросистему, что позволяет вводить нестандартные синтаксические конструкции и манипулировать символьными комбинациями, а также создавать программы, которые неявно пишут другие программы. LISP ***гомоиконичен,*** тождественен представлению кода и данным его программы в виде ***s***-выражений (символьных выражений), позволяет модифицировать код как данные. Сегодня LISP это целое семейство языков программирования, видоизменяющихся в соответствии со временем, породившее большое количество диалектов, предназначенных для различных специальных нужд.

[*] IPL (Information Processing Language) — один из первых высокоуровневых языков программирования, разработанный в конце 1950-х — начале 1960-х годов прошлого века специально для задач искусственного интеллекта (ИИ). Язык проектировался для обработки символьных нечисловых данных, а также для операций со сложными структурами данных (списки, деревья). IPL — пионер в области языков для ИИ. Его идеи (списки, рекурсия, символьная обработка, интерпретируемость, динамическое управление памятью) стали фундаментом для современных технологий, хотя сам язык сегодня не используется в продуктивной разработке.

[34] Прим. автора. XLISP — это специализированный язык процедурно-функционального программирования, созданный с целью экспериментов с объектно-ориентированными событиями на компьютерах с ограниченными ресурсами (малой памятью и производительностью). XLISP является диалектом LISP, и имеет различные усовершенствованные варианты такие как: XLISP-PLUS, XLISP-STAT. Реализации XLISP могут функционировать практически на всех известных операционных системах, что позволяет говорить о его кроссплатформенности.

[35] Прим. автора. SAL (Simple Algorithmic Language*) — это специальный алгоритмический язык программирования для со-

тельной средой служат языки C[36] и Java[37].

Аргументом *первой причины* служит тот факт, что Nyquist — это универсально-экспериментальное средство программирования, удовлетворяющее все мировые стандарты и современные технологические практики в сфере аудиоинженерии, а также редактирования цифрового звука [1].

Аргументом *второй причины* является возможность языка Nyquist интегрировать свои командные функции и микросценарии в один из наиболее известных компьютерных редакторов аудиоматериала — Audacity® [2,3]. Это позволяет задействовать сценарно-программные механизмы, составленные на данном языке, одновременно с интерфейсно-ориентированными опциями данного редактора.

К примеру, процесс демонстрации синтезируемых комбинаций, создаваемых с помощью фрактальных структур, можно визуализировать через оконный интерфейс редактора Audacity® [4]. Это дает возможность в аудиторных условиях эффективнее воспроизводить подобные процессы, наглядно показывая правила и алгоритмы исполнения нотаций инструкций.

В пользу *третьего аргумента* говорит устойчивая взаимосвязь языка Nyquist с одним из самых *мощных* и *высокоразвитых* языков программирования современности — LISP [5]. Язык программирования LISP уникален прежде всего потому, что он создавался как язык, способный производить *интеллектуальные* операции над *программируемыми массивами* данных, а также *включать* такие *операции*[38] в структуру собственного кода [6]. Помимо прочего, языки Arctic, Canon и Fugue в сочетании с языком SAL и кроссплатформенностью дают Nyquist неоспоримые *ультраспециализированные* преимущества, выражающиеся в специфике описания *мультиблоковой* нотации опциональных алгоритмов работы с аудиоданными, составлением сценариев для аудиоэффектов и композиций через комбинации функций [7]. Инкорпорирование в операционный пакет Nyquist таких LISP-диалектов как XLISP делают данный язык лидером в области статистической обработки аудиосигналов.

Nyquist в этом смысле является специализированным декларативным языком программирования, который фактически сочетает в себе все функции, которые могли бы быть использованы в обработке аудиоданных на LISP с особыми предписаниями языка SAL.

Поддержка Nyquist-*программирования* в Audacity®, реализованная через специальное приглашение, делает данный язык мощнейшим средством, способным превратить компьютер *средней мощности* в полноценную *лабораторию* по работе со звуком[39].

ставления компьютерных композиций, разработанный для эффективного взаимодействия со средой Common Music (CM). Common Music (CM) это объектно-ориентированная среда для составления алгоритмической музыкальной композиции, разработанная Генрихом Таубе (Heinrich Taube), почётным профессором композиции и теории музыки в Школе музыки Университета Иллинойса США (University of Illinois School of Music at Urbana-Champaign, USA). Фактически SAL это язык для композиторов, пишущих электронную музыку. Он позволяет создавать сложные алгоритмические музыкальные произведения и описывать нестандартные музыкальные структуры через операционный код. Интегрирован в систему Common Music, а также в редактор Audacity® (составление синтаксических конструкций) через приглашение языка Nyquist.

* Официально «Simple Algorithmic Language» — (простой алгоритмический язык), есть и другое неофициальное название «Secretly Another Lisp» — (тайно ещё один LISP, секрет ещё одного LISP). Вообще название языка происходит от имени Sal (Salvatore) и является данью уважения к композитору Sal Martirano**, профессору музыкальной школы UIUC. Не путать с названием другого средства программирования SAL (Source-Annotation Language), являющегося системой аннотаций для кода на «C/C++», разработанной корпорацией Microsoft.

** Сальваторе Джованни Мартирано (Salvatore Giovanni Martirano) — американский композитор, автор современной классической музыки. В 1969 году Мартирано вместе с группой инженеров и музыкантов из Университета Иллинойса (University of Illinois) начал разработку электронного музыкального инструмента SAL-MAR Construction. Это гибридная система, в которой логические схемы TTL (Transistor-Transistor Logic) управляют аналоговыми модулями (генераторами, усилителями и фильтрами). Исполнитель взаимодействует с инструментом через горизонтальную панель из 291 светового сенсорного переключателя, что позволяет гибко управлять макро- и микропараметрами звучания.

[36] Прим. автора. Си (C) — многофункциональный, процедурный, статически-типизированный язык универсального назначения. Язык Си настолько хорош и универсален, что тут трудно что-либо выделить, отметим только ряд его ограничений (о возможностях всем специалистам известно). Отсутствие автоматического управления памятью, риск ошибок сегментации — необходимость вручную контролировать границы массивов. Последнее ограничение чаще всего возникает при попытке доступа к памяти, которая не выделена процессу или к которой у программы нет прав на доступ. Операционная система принудительно завершает программу, чтобы предотвратить повреждение данных или системы.

[37] Прим. автора. Java — это объектно-ориентированный, кроссплатформенный язык программирования общего назначения, разработанный компанией Sun Microsystems (позже приобретённой Oracle). Компилятор преобразует сценарии программы в ***байт***-код, затем виртуальная машина (Java Virtual Machine, JVM), являющаяся частью исполняющей системы (Java Runtime Environment, JRE), воспроизводит ***байт***-код, транслируя его в машинные инструкции для конкретной платформы. Язык имеет большое количество библиотек, что говорит о его развитости в области проектирования высокой сложности. Язык имеет чёткий синтаксис, что делает его удобным в обращении для людей, изучающих азы профессионального программирования.

[38] Прим. автора. Чаще всего ***неявно*** (программа — в программе).

[39] Прим. автора. Одним из принципиальных отличий связки NyquistIDE и Audacity® от других программных и программно-ап-

Инструментально-техническая и аппаратная компьютерная база, используемая автором исследования для воспроизведения и совершенствования эксперимента сигнально-комбинаторной обработки аудиоматериала (клеточные автоматы)

Для достижения целей эксперимента использовалась следующая инструментально-техническая база (рис. 2).

Рис. 2. Языки программирования, программы и компьютерные технологии, используемые автором для воспроизведения эксперимента и подготовки данной статьи

Для обработки аудиосигнала использовались языки программирования: LISP (XLISP, Nyquist), SAL. Инженерный процессинг осуществлялся в интерфейсе прикладного программирования NyquistIDE с выводом сигнала в Audacity®. Для моделирования правил клеточных автоматов использовался пакет Wolfram Mathematica® v.10.2, язык Wolfram (в качестве описания сценариев построения), среда трёхмерного проектирования Blender® (в качестве моделирования графических плашек), язык программирования Python™ (для компиляции графических композиций клеточных автоматов).

Аппаратная база для вычислений: ноутбук ASUS ROG G615JH-RV045, процессор (частота 2,4 ГГц): Intel Core i5-13450HX (десять ядер), SSD: 512, оперативная память (DDR5): 16 ГБ.

Разобравшись с причинами использования языка Nyquist для целей дизайна, синтеза и ресинтеза аудиоданных приступим к сути изложения эксперимента, поставленного Энн Льюис.

Экспериментальная программная модель Данненберга-Льюис. Компьютерный эксперимент: проверка воспроизводимости, уточнение параметров, элементы моделирования и практика использования клеточных автоматов в сфере аудиопроизводства

Эксперимент Энн Льюис и её научного руководителя Роджера Данненберга заключался в возможности использования клеточных автоматов для генерации полноценной алгоритмической аудиокомпозиции[40][8]. Клеточные автоматы (CA-генерацию)[41] с одинаковой эффективностью можно применять как к предзаписан-

паратных аналогов в аспекте инженерных практик редактирования звука (помимо открытой лицензии) является чёткая продуманность изложения базисных инструкций, сценариев и прочих элементов кода в отношении вычислительных возможностей данного программного комплекса, формирующих полифункциональную автономную архитектуру для решения конкретных утилитарно-технических задач. Иными словами, это тот самый случай, когда ядро программного комплекса выполнено на высоком уровне, а его вычислительные возможности настолько широки, что местами опережают аппаратные возможности ЭВМ, реализуя символьно-алгебраические операции в обход ограничений периферийной технической зоны. Эффект действительно очень мощный, поскольку, к примеру, мультичастотное исправление спектроакустических параметров сонограммы, которое обычно излишне нагружает, а в некоторых случаях физически перегружает центральный процессор, в данном программном комплексе позволяет получить оперативный и качественный результат при минимальной нагрузке на аппаратную оснастку компьютера.

[40] Прим. автора. Практика показала, что в зависимости от правил, клеточные автоматы могут генерировать не просто примитивные декларативно-стилевые композиции, но и формировать сложную алгоритмическую структуру звука за счёт мультиагентного пересчёта сигналов, характеризуемых экземплярами наборов состояний (atoms→cells).

[41] Прим. автора. CA-генерация — альтернативное название «клеточных автоматов», характеризующее структуру, принцип и работу различных топологий ячеек в соответствии с установленными правилами логики математических операций.

ному адиоматериалу[42], так и к *форм*-оболочкам[43], формирующим базовые элементы для полноценных звукоописательных структур[44] таблично-волнового синтеза. Клеточный автомат состоит из *массива* ячеек, где каждая ячейка инициализируется либо *чёрным*, либо *белым* цветом (вкл./выкл., 1/0) [9].

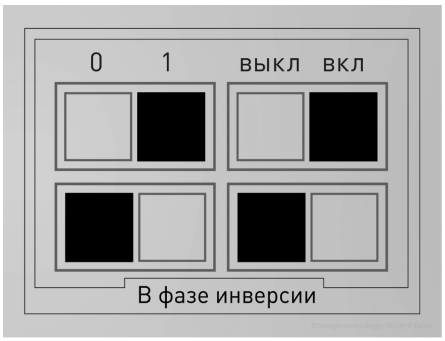

Рис. 3. Ячейки клеточных автоматов, характеризующие кластерно-бинарную систему хранения данных. Рисунок демонстрирует унарное обращение к бинарным сотам

[42] Прим. автора. Генеративные возможности клеточных автоматов позволяют воспроизводить новые оригинальные композиции, используя в качестве ресурсной базы любой органический аудиоматериал. Это может быть как простая аудиодорожка с предзаписанными живыми звуками любых музыкальных инструментов, включая вокальные партии, так и ***midi***-нотации с предзагруженной звуковой формой, предполагающей пустой заголовок.

[43] Прим. автора. Клеточные автоматы имеют возможность работать напрямую с ***форм***-оболочками. Для этого необходимо использование алгоритмов алгебраично-функционального* и таблично-волнового** синтеза. Алгоритмические принципы работы данных видов синтеза должны быть изложены в специальных программах-***инструкциях*** и локализованы в виде файлов с расширением того языка программирования, на котором предполагается производить операционные действия по манипуляции с аудиоданными. В нашем случае это язык программирования Nyquist и его прародитель LISP. В некоторых случаях это может быть SAL.

*Алгебраично-функциональный аудиосинтез — это вид синтеза, основанный на принципах физического моделирования, использующий математические функции в качестве опорных констант для получения оригинальных образцов различных волнообразных звуковых форм. Профиль волны из данного вида синтеза обычно получается при манипулятивном сочетании нескольких параметров, таких как синус, косинус, парабола, а также рекурсивных математических формул, нотированных под применяемый аудиоинженером язык программирования. Это один из немногих видов аудиосинтеза, позволяющий моделировать аналоговое звучание различных музыкальных инструментов и вокала в автономном режиме.

** Таблично-волновой аудиосинтез — это модель звукового синтеза, использующая отдельные блоки фрагментов аудиоматериала для их постепенной последовательной динамической смены между собой с целью получения различных тембральных оттенков звучания.

[44] Прим. автора. Под звукоописательными структурами понимаются ресурсы (звуковые массивы), представленные в виде семиотической структуры, топология символов которой управляется математическими функциями, в свою очередь, логика которых описана в символьно-знаковой системе конкретного языка программирования, например Nyquist, SAL или Python™.

На каждом шаге следующая *строка*/*генерация* (представленная визуально — рисунок №3, как показано ниже предыдущего массива — фаза инверсии) вычисляется из *первого* массива с использованием *правила обновления* для *каждого* элемента массива [10]. Правило обновления (которое определяет сам автомат) — это просто *функция* от *родительского элемента* массива и его *левого* и *правого* соседей [11].

Классически клеточный автомат строится по координатной сетке и имеет две фазы перехода [12]. Первая фаза (по умолчанию не активно) — ячейки пустые = «0». Вторая фаза (по умолчанию активно) — ячейки заполненные = «1». Параметры моделирования могут быть разными. Правило обновления — это чётко выстроенный алгоритм, который на каждом шаге эволюции задаёт *новое состояние* (положение) каждой из ячеек сетки, руководствуясь *текущим состоянием* самой ячейки и *текущим положением* соседних ячеек в *рамках заданной окрестности* [13]. Допустим, задаётся начальное состояние сетки t = 0 (означает, что активность распределения фрагментов сигнала 0). Для каждого шага $t \rightarrow t + 1$, далее «→»:

1) → для каждой ячейки анализируется её текущее состояние и состояния соседей;
2) → по правилу обновления вычисляется новое состояние ячейки;
3) → все ячейки переходят в новые состояния одновременно.

Процесс повторяется на заданное число шагов или до достижения устойчивого состояния. Приведём в пример элементарный одномерный клеточный автомат в двух состояниях (0/1) по правилу 30. Определим положение ячеек (таблица №1):

Таблица 1.

Динамика состояний клеток (ячеек) по правилу 30

Текущее состояние трёх соседних клеток	Новое состояние центральной клетки
111	0
110	0
101	0
100	1
011	1
010	1
001	1
000	0

Перечислим все возможные конфигурации первых трёх соседних клеток:

111, 110, 101, 100, 011, 010, 001, 000

Для каждой конфигурации укажем новое состояние центральной клетки (*правило* 30):

0, 0, 0, 1, 1, 1, 1, 0

Составим двоичное число из выходных значений (слева направо):

00011110

Переведём двоичное число в десятичную систему:

$$00011110_2 = 1\cdot2^4 + 1\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 = 16 + 8 + 4 + 2 = 30_{10}$$

Таблица пара «*вход → выход*»

Таблица 2.

Таблица состояний аудиофрагментов по семафорно-векторной карте перехода (унарный оператор «⟶» показывает направление перехода в соответствии с состоянием аудиофрагмента)[45]

Вход		Унарный оператор «⟶»	Выход	Состояние аудиофрагмента по семафорно-векторной карте перехода[45]
1.	111	→	0	3
2.	110	→	0	2
3.	101	→	0	2
4.	100	→	1	2
5.	011	→	1	3
6.	010	→	1	2
7.	001	→	1	2
8.	000	→	0	0

Представим элементы вычисления как начальный *блок*-массив:

A1 A2 A3 . . . (B1 B2 B3 . . .).

Пусть B2 — элемент, значение которого вычисляется. Таким образом, B2 зависит только от значений A1, A2 и A3. Примером правила обновления может быть:

если A1 = A3 и A2 = 1, то B2 = 1, *иначе* B2 = 0

Существуют два возможных значения для каждого из A1, A2 и A3, а это означает, что существуют $2^3 = 8$ возможных конфигураций. И есть $2^8 (2^8 = 2\times2\times2\times2\times2\times2\times2\times2) = 256$ возможных функций от A1, A2 и A3 до B2. Таким образом, существуют только 256 возможных правил обновления[46]. Обратите внимание, что количество возможных правил обновления не зависит от количества элементов в массиве. Правила могут быть пронумерованы от 0 до 256[47]. В этом виде синтеза используются два основных правила Вольфрама (Wolfram rules) *см*. рисунок № 4.

1) Хаотическое правило — (30)
2) Фрактальное правило — (90)

На рисунке ниже *правило* 30 (слева) используется для создания ряда строк, начиная с первой «единицы» в первой строке.

Рисунок демонстрирует классическое построение клеточных автоматов по правилам, *слева* — 30, *справа* — 90. Ветвление сигнала по правилу 30 будет выглядеть следующим образом:

B = 0 → F = 0 → A = 1 → H = 0 → Y = 0 → U = 1 → R = = 0 → Q = 1

Замыкание строки

Y = 0 → U = 1 → R = 0 → Q = 1 → B = 0 → F = 0 → A = = 1 → H = 0

Вычисление вероятности состояния центральной ячейки

$$new\_state = left \oplus (center \vee right)$$

где: ⊕ — операция XOR (*исключающее* «ИЛИ»), ∨ — операция OR («ИЛИ»).

1) Сначала вычисляется center∨right (логическое «ИЛИ» центра и права).
2) Затем результат XOR сопоставляется с left (левым соседом).
3) Полученное значение становится новым состоянием центральной ячейки.

[45] Прим. автора. Семафорно-векторная карта перехода (в клеточных автоматах) — это верхняя часть (плашка) графической идентификации, характеризующая начальные состояния побитовых массивов на входе в систему, ***фактически декларация начального набора значений***, например, начального состояния клетки или набора битов. С точки зрения физической обработки информации такая карта показывает нам состояние переходов (преобразований) различных мультимедийных данных в виде сигналов. В нашем случае карта показывает поэтапное преобразование фрагментированных массивов аудиоданных. Восьмиблочная плашка часто представляет собой 8-битное число, которое может задавать стартовое состояние системы, параметры или конфигурацию. Например:

– В двоичной системе — это 8 битов, где каждый блок — это ***бит***, ***включён*** или ***выключен***.

– В некоторых случаях она может обозначать контрольные или идентификационные данные.

[46] Прим. автора. Обратите внимание, что количество возможных правил обновления не зависит от количества элементов в массиве.

[47] Прим. автора. Таким образом, существуют только 256 возможных правил обновления.

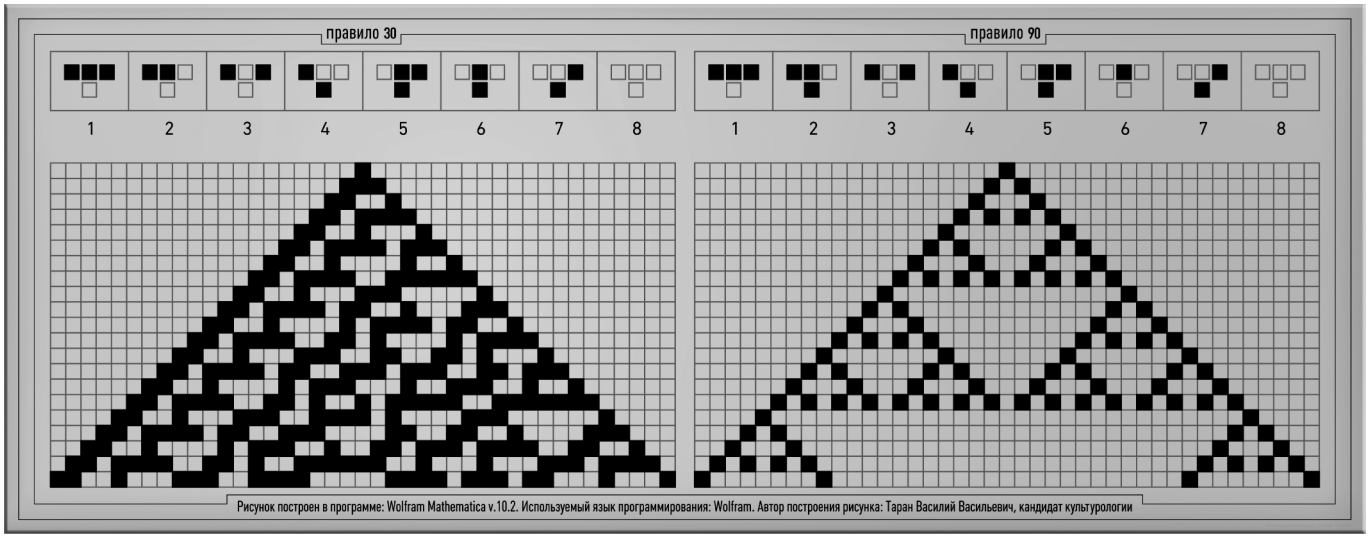

Рис. 4. Демонстрация построения клеточных автоматов: *слева — правило* 30, *справа — правило* 90

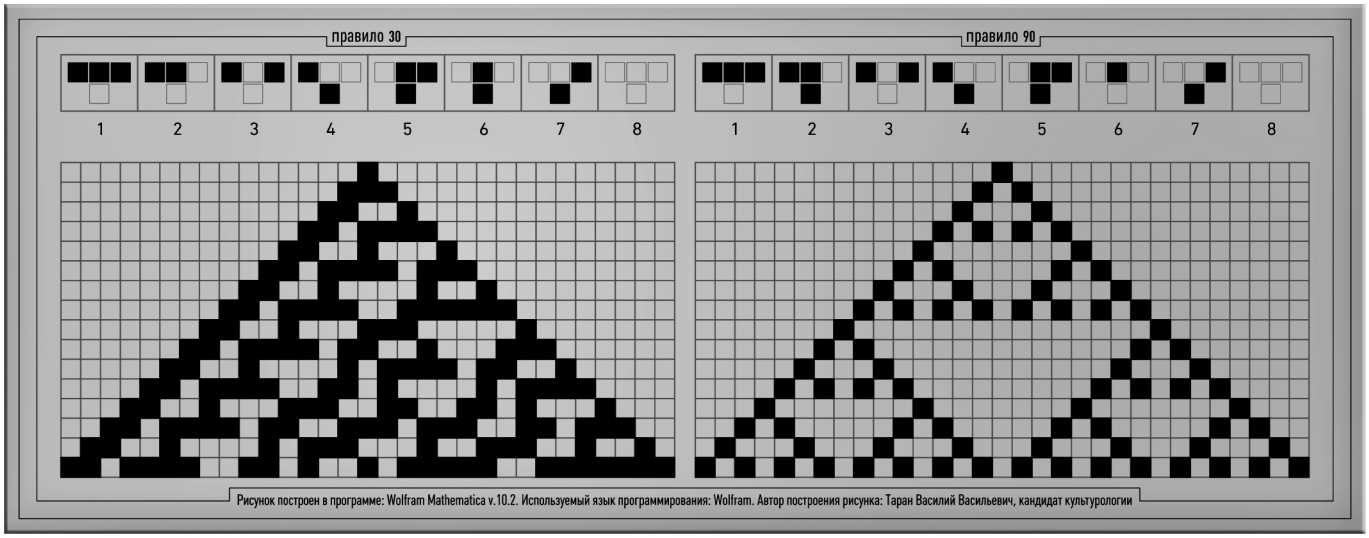

Рис. 5. Пример топологии построения клеточных автоматов по правилам 30 и 90 (шаг 1 к 15)

Ветвление сигнала — это особый элемент в структуре СА-*программирования*, позволяющий разделить один входной сигнал на несколько независимых выходных потоков, направляемых в разные части решётчатой структуры. Это ключевой элемент для построения сложных вычислительных схем (аналогов логических вентилей, маршрутизаторов, процессоров) в дискретных пространственно-распределённых системах. Замыкание строки в клеточных автоматах — это способ обработки граничных условий, при котором крайние ячейки строки «соединяются», образуя замкнутую структуру (аналог кольца или тора в одномерном случае). Это устраняет «крайние эффекты» и позволяет сигналу циркулировать бесконечно. Ниже приведён пример *топологии* правил по шагу 1 к 15.

В одномерных клеточных автоматах с диапазоном равным (=1) и только двумя состояниями имеются восемь соседствующих объектов, которые сопоставляются со значениями (1,0), что в общей сложности предполагает 256 возможных *правил* (чёрное = 1,белое = 0). Эволюция 256 элементарных правил закономерна коду Вольфрама[48], когда для любого клеточного автомата в момент t_0 чёрной является только средняя ячейка. Важно отметить, что даже если правило относится к классу 4, это не даёт основание, что оно будет вызывать сложное поведение, начиная с любой возможной конфигурации, здесь уместно вспомнить про *правило* 54[49]. *Принцип работы*:

[48] Прим. автора. Код Вольфрама — это инструкция, которая присваивает каждой клетке ***идентификационный*** номер, например белое поле «0», чёрное поле «1», при этом нижняя строка читается как двоичное число «01011010», для его преобразования в десятичную систему необходимо воспользоваться одним из порядковых названий правил, которое применяется для синтеза, всего их 256.

[49] Прим. автора. ***Правило*** 54 — это одно из правил классических одномерных клеточных автоматов, оно относится к семейству правил, которые были классифицированы в рамках системы Конвея* — таблицы правил для одномерных автоматов с ***двоичным*** состоянием. Эти правила характеризуют изменение состоя-

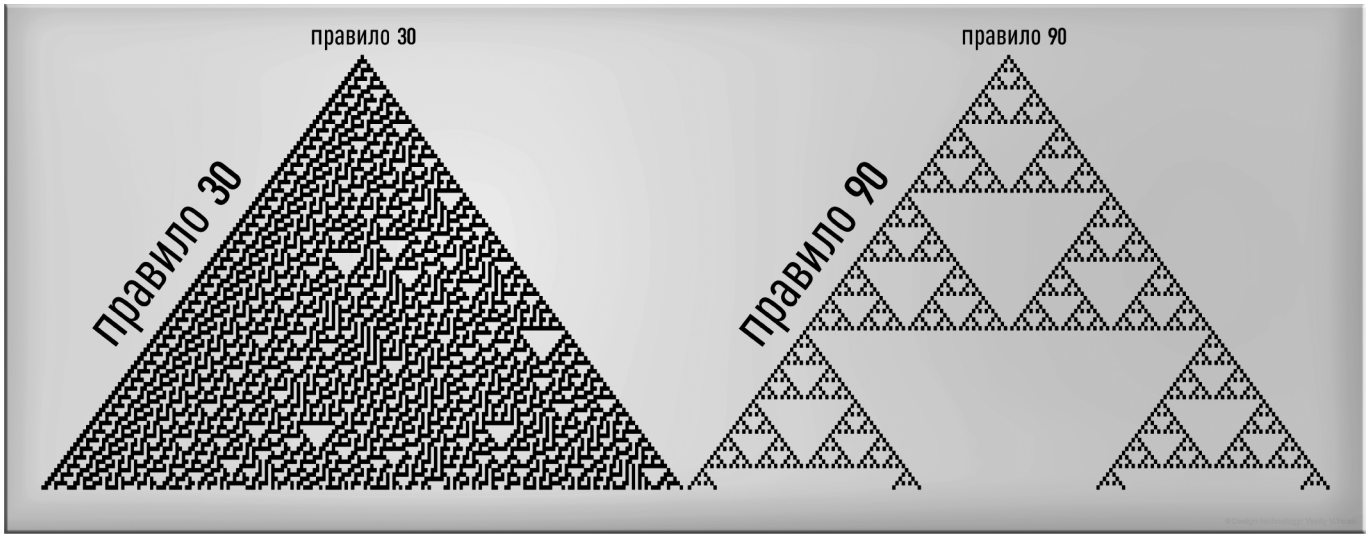

Рис. 6. Демонстрация построения клеточного автомата по правилам 30 и 90, (Шаг 100)

new_state=(left∧center∧¬right)∨(¬left∧center)∨(center ∧¬right)∨(¬left∧¬center∧right).

1) left — состояние левой соседней ячейки.
2) center — текущее состояние центральной ячейки.
3) right — состояние правой соседней ячейки.

— 1) ↓ → 2) ↓ → 3) ↓ —

1) ¬ — отрицание (NOT).
2) ∧ — логическое «И» (AND).
3) ∨ — логическое «ИЛИ» (OR).

Первая часть (left∧center∧¬right) даёт 1, если left = 1, center = 1, right = 0.

Вторая часть (¬left∧center) даёт 1, если left = 0, center = 1 (независимо от right).

Третья часть (center∧¬right) даёт 1, если center = 1, right = 0 (независимо от left).

Четвёртая часть (¬left∧¬center∧right) даёт 1, если left = 0, center = 0, right = 1.

Общее выражение истинно (*результат* = 1), если истинна, хотя бы одна из частей.

ния ячейки в зависимости от её текущего положения и локализации соседей.

* John Horton Conway — британо-американский математик, внёсший весомый вклад в различные области математики, включая теорию групп, теорию чисел, комбинаторную теорию игр и клеточные автоматы. Широкую известность получил как создатель Игры Жизни (англ. Game of Life) — ***клеточного автомата***, моделирующего развитие шаблонов на сетке по простым правилам, которая стала классическим примером в изучении сложности, эмерджентности и возникающего поведения. Он предложил систему классификации правил для одномерных клеточных автоматов, которая получила название системы Конвея, она помогает охарактеризовать поведение различных клеточных автоматов. Его исследования оказали значительное влияние на теорию сложности, моделирование самоорганизации и другие области математики и компьютерных наук (информатики).

Чуть выше (рисунок № 6) можно увидеть построение одномерного клеточного автомата по правилу 30 и 90 (Шаг 100).

Построение рисунков (клеточных автоматов) осуществлялось автором в программе Walfram Mathematica, на языке Walfram. Координаты построения были следующие[50]:

Таблица 3.

Координаты вариативного построения клеточных автоматов по указанным в таблице правилам

Правило 30	Правило 90
Алгебраическая форма конструкции	
$(p, q, r) \mapsto (p + q + r + q\ r) \bmod 2$	$(p, q, r) \mapsto (p + r) \bmod 2$
Форма булевых значений	
(p, q, r) ↦ p XOR (q OR r)	(p, q, r) ↦ p XOR r
Текстовая форма	
(p, q, r) ↦ p ⊻ (q ∨ r)	(p, q, r) ↦ p ⊻ r

Правило 30 представляет особый интерес, поскольку оно хаотично. Это правило любопытно тем, что оно взято за основу в качестве генератора случайных чисел, используемого для больших целых чисел в Wolfram Mathematica*. Ниже представлен рисунок № 7, на котором *проиллюстрирован характер действий*[51] и аргументов состояний с примером определения функции перехода по *правилу* 30.

[50] Прим. автора. Для точности воспроизведения эксперимента приводим три формы ключей построения: простая алгебраическая форма, форма записи булевых значений, текстовая форма.

[51] Прим. автора. Код на языке программирования Nyquist, предложенный Энн Льюис.

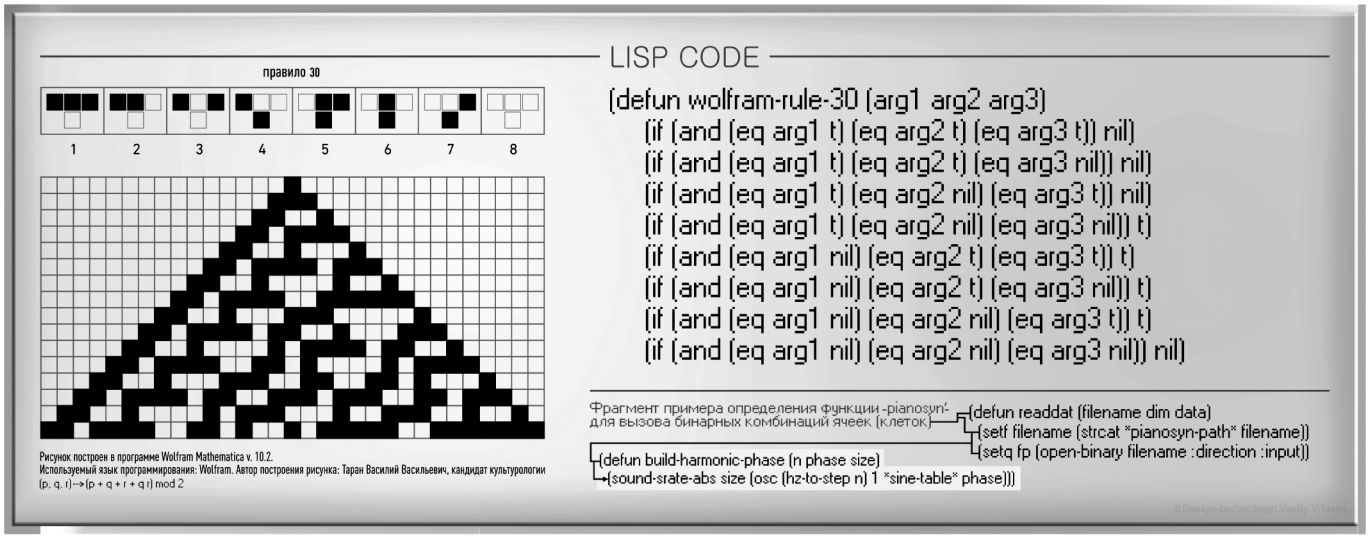

Рис. 7. Характер действий ячеек клеточного автомата по правилу 30 с фрагментом определения функции «pynosin»

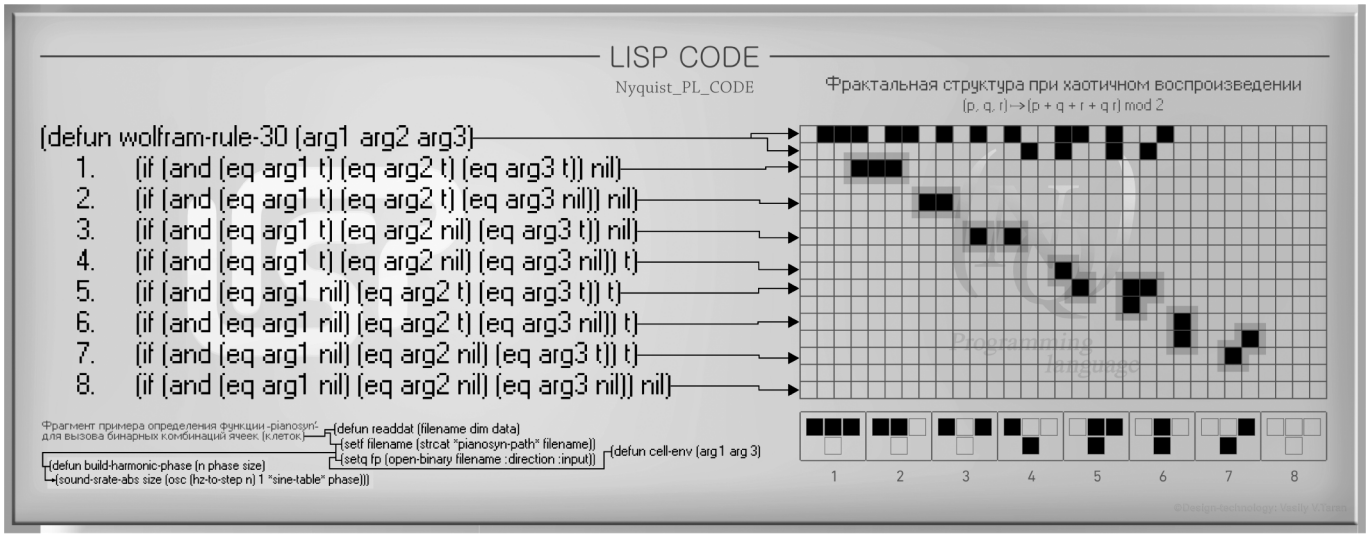

Рис. 8. Строковые параметры положения ячеек с тремя аргументами в условиях одномерности переходов из одного состояния в другое в границах *локального* цикла итерации

На рисунке № 7 показана универсальная форма программного кода, реализующая базовые переходы элементов на сетке по правилу 30. Такая форма успешно применима и в Nyquist-*программировании* и в традиционном LISP. Код реализует правило для клеточного автомата, известное как *правило* Вольфрама 30 (Rule 30). Он принимает три аргумента (arg1, arg2, arg3), каждый из которых представляет состояние соседних клеток (например, в предыдущем шаге), где t означает «истина» (или «заполнена», «активна»), а nil — «ложь» (или «пустая», «не активна»). Код проверяет все возможные комбинации состояний трёх соседних клеток и возвращает t или nil, основываясь на регламентах правила 30. Если все три клетки активны (t t t), возвращается nil. Если активны две клетки и третья — nil, возвращается nil. Если первая активна, а остальные — nil, возвращается t. Если первая — nil, вторая — активна, третья — активна, возвращается t. Если первая — nil, вторая — активна, третья — nil, возвращается t. Если первая — nil, вторая — nil, третья — активна, возвращается t. Если все три — nil, возвращается nil.

Таблица 4.

Сопоставление состояний ячеек в период перехода

	Предыдущие три соседние ячейки	Новое состояние ячеек
1.	t t t	nil
2.	t t nil	nil
3.	t nil t	nil
4.	t nil nil	t
5.	nil t t	t
6.	nil t nil	t
7.	nil nil t	t
8.	nil nil nil	nil

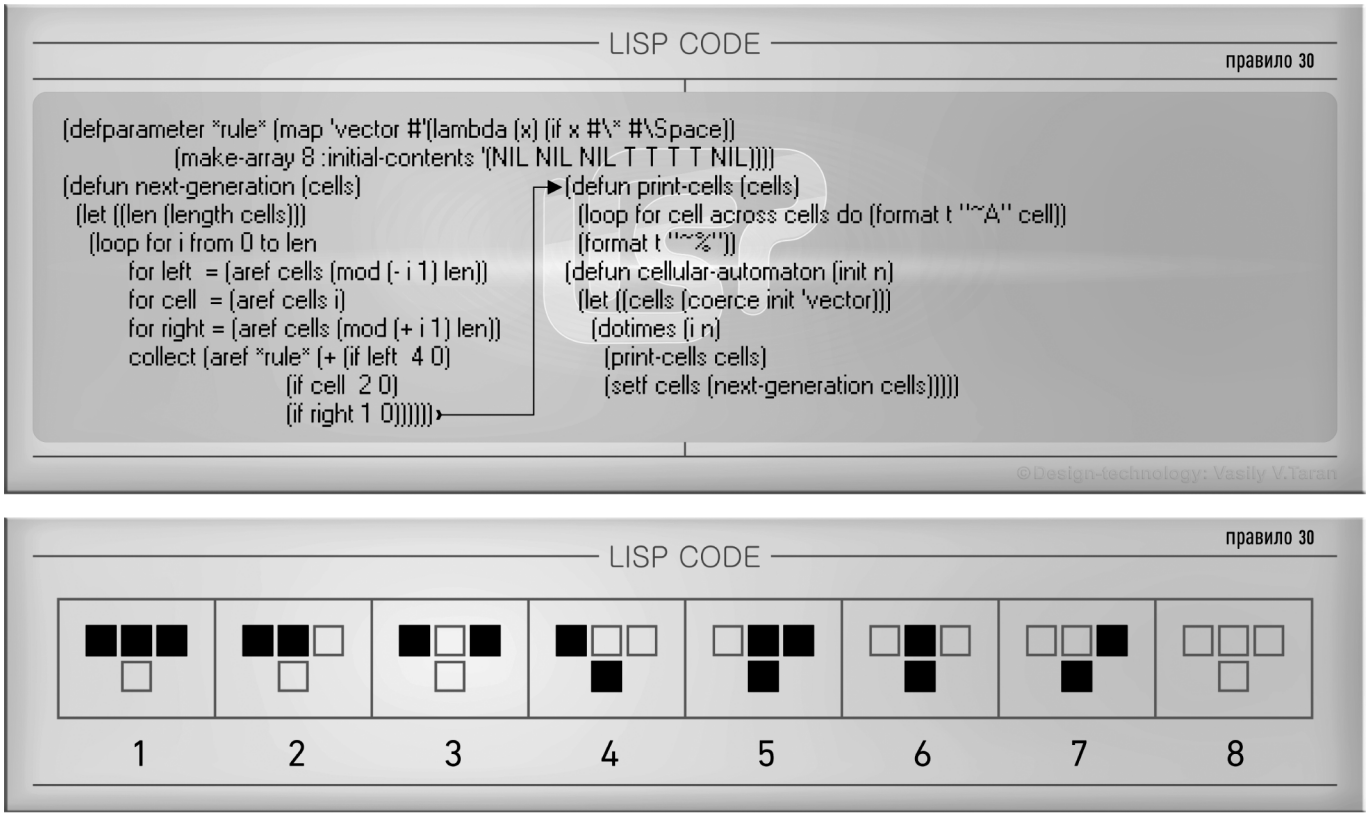

Рис. 9. Структура кода, демонстрирующая полноценную работу клеточного автомата

На предыдущей странице следует более детальный векторный разбор функционирования клеточного автомата (строковые параметры) по правилу 30 в условиях одномерности переходов из одного состояния в другое в рамках *локального* цикла итерации[52] (рис. 8).

Выше показана генеративная[53] структура кода, формирующая полноценный одномерный клеточный автомат по правилу 30 (рисунок № 9).

Это одномерный клеточный автомат[54], основанный на правилах, заданных в виде массива и функции для эволюции и отображения его состояния.

Переменная rule:

(defparameter *rule* (map 'vector #'(lambda (x) (if x #* #\Space))
(make-array 8 :initial-contents '(NIL NIL NIL T T T T NIL))))

Создает вектор *rule*, длиной 8, где каждая позиция соответствует одному из возможных трёхбитных сочетаний соседних клеток. Вектор содержит символы: #* (звёздочка) и #\Space (пробел). Начальное содержимое — (NIL NIL NIL T T T T NIL). Функция map применяет *лямбда-функцию* к каждому элементу массива, заменяя NIL на пробел, T на звездочку, таким образом, создавая визуальные символы для отображения.

Функция next-generation:

```
(defun next-generation (cells)
(let ((len (length cells)))
(loop for i from 0 to len[55]
for left = (aref cells (mod (— i 1) len))
for cell = (aref cells i)
for right = (aref cells (mod (+ i 1) len))
collect (aref *rule* (+ (if left 4 0)
(if cell 2 0)
(if right 1 0))))))
```

[52] Прим. автора. В информатике (теории компьютерных наук) под локальным циклом итерации понимают конструкцию цикла, которая позволяет выполнять блок кода многократно, пока выполняется определённое условие или процесс обучения нейросети, состоящий из множества повторений.

[53] Прим. автора. Под генеративной структурой кода понимается набор символьно-арифметических логических инструкций, формирующих тело клеточного автомата. Тело клеточного автомата, функционирующее по правилу 30, показанное на рисунке №8, исполнено на языке LISP, однако эта структура также хорошо интерпретируема одним из его расширенных диалектов Nyquist.

[54] Прим. автора. Этот код реализует одномерный автомат по правилам, заданным в *rule*, и отображает его эволюцию на экране. Если хотите запустить автомат, нужно передать начальную строку из символов #* и #\Space, например: (cellular-automaton «# * * * #» 10), где начальное состояние — строка из звездочек и пробелов, и автомат будет эволюционировать 10 шагов.

[55] Прим. автора. Обратите внимание, что цикл идёт от 0 до len, что создает len+1 элементов, возможно, это ошибка, обычно цикл идет до len-1. Но в данном случае, так как используются циклические соседние элементы, это не критично, однако, лучше было бы использовать 0 to (1— len).

Принимает текущий массив cells. Для каждого элемента по индексу i берет значение текущей клетки, её левого и правого соседа (по модулю для цикличности). Определяет индекс правила, суммируя значения: left (умноженное на 4), cell (на 2) и right (на 1). Находит новое состояние клетки из *rule* по полученному индексу.

Функция print-cells:
(defun print-cells (cells)
(loop for cell across cells do (format t «~A» cell))
(format t «~%»))

Выводит текущие состояния клеток в виде последовательности символов (звездочек и пробелов).

Функция cellular-automaton:
(defun cellular-automaton (init n)
(let ((cells (coerce init 'vector)))
(dotimes (i n)
(print-cells cells)
(setf cells (next-generation cells)))))

Инициализирует автомат начальным состоянием init (строкой или списком символов), преобразуя его в вектор. Выполняет n итераций: выводит состояние и вычисляет ближайший вектор дальнейшего перемещения.

Правило 90, иллюстрируемое рисунком №3 (справа), также представляет интерес, поскольку именно оно *генерирует* фрактальные структуры. Логика этого правила такова — *вычисление вероятности* состояния *центральной* ячейки:

$$new\_state = left \oplus right$$

где: left — состояние левой соседней ячейки, right — состояние правой соседней ячейки, ⊕ — операция XOR (результат равен 1, если входы различаются; 0 — если одинаковы).

1) Берём состояния левого и правого соседей.
2) Применяем XOR: если соседи разные (0 и 1 или 1 и 0), результат = 1; если одинаковые (0 и 0 или 1 и 1), результат = 0.
3) Полученное значение становится новым состоянием центральной ячейки.

Для генерации фрактала (треугольник Серпинского)[56] можно применить следующий код:

(defun sierpinski-coords (x1 y1 x2 y2 x3 y3 depth result)
;; *Возвращение списка координат треугольников*
(if (zerop depth)
(push (list x1 y1 x2 y2 x3 y3) result)
(let* ((mid1x (/ (+ x1 x2) 2.0))
(mid1y (/ (+ y1 y2) 2.0))
(mid2x (/ (+ x2 x3) 2.0))
(mid2y (/ (+ y2 y3) 2.0))
(mid3x (/ (+ x3 x1) 2.0))
(mid3y (/ (+ y3 y1) 2.0)))
(sierpinski-coords x1 y1 mid1x mid1y mid3x mid3y (1— depth) result)
(sierpinski-coords mid1x mid1y x2 y2 mid2x mid2y (1— depth) result)
(sierpinski-coords mid3x mid3y mid2x mid2y x3 y3 (1— depth) result)))
result)

;; *Пример вызова*
(let ((triangles (sierpinski-coords 100 100 300 100 200 300 4 nil)))
(format t «Сгенерировано ~d треугольников~%» (length triangles)))

После данной процедуры звук будет выглядеть как набор фракталов с топологией структуры по треугольнику Серпинского. Ниже на рисунке № 10 представлена топология распределения звуковых данных по треугольнику Серпинского (*фрактальная структура*).

Базовые операции в структуре программного кода (схематично)[57]:

Defparametr[58] — объявляет и инициализирует динамическую переменную на верхней строке (заголовка) программы. Map — функция применения заданной операции ко всем элементам коллекции (обычно списка), с получением новой коллекции результатов. Vector — создаёт одномерный массив. Lambda — это специаль-

[56] Прим. автора. Данный код на языке LISP реализует построение координат треугольников для фрактала Серпинского. Он принимает начальные координаты треугольника (x1, y1), (x2, y2), (x3, y3), глубину рекурсии (depth) и список результатов (result). На каждом шаге, если глубина равна нулю, он добавляет текущий треугольник в результат. Иначе он делит текущий треугольник на 3 меньших, вычисляя середины сторон, и рекурсивно вызывает себя для каждого из них, уменьшая глубину на 1. Чтобы построить треугольник Серпинского по этому коду, нужно вызвать функцию с начальными координатами большого треугольника и нужной глубиной рекурсии. Например: (serpinski-coords 0 0 100 0 50 86.6 4 '()). Где (0,0), (100,0), (50,86.6) — вершины равностороннего треугольника. Глубина 4 даст достаточно подробную фигуру.

[57] Прим. автора. Функции, операторы и другие ***составные части программного кода*** (вне кодовой нотации) по статье пишутся с ***заглавных букв***, в коде (язык LISP) используется строго ***строчный*** регистр (***за исключением правил оговоренных в прологе программы***). Непечатные ***кавычки*** используются только в параметрах программного кода. Расширения файлов и ***логические взаимосвязи*** указаны ***курсивом***.

[58] Прим. автора. Nyquist реализует лишь подмножество LISP с акцентом на аудиообработку. В Nyquist для объявления глобальных переменных используются: defvar — объявляет глобальную переменную и инициализирует её, если она ещё не определена. Например, для установления частоты дискретизации в аудиопотоке конструкция будет выглядеть следующим образом: (defvar *sample-rate* 44100 «Частота дискретизации по умолчанию»). Аналогично возможна операция присваивания через seft. Можно объявить глобальную переменную напрямую (но без защиты от повторного определения): (setf *gain* 0.8). В сценариях для Audacity® глобальные переменные сохраняются между вызовами.

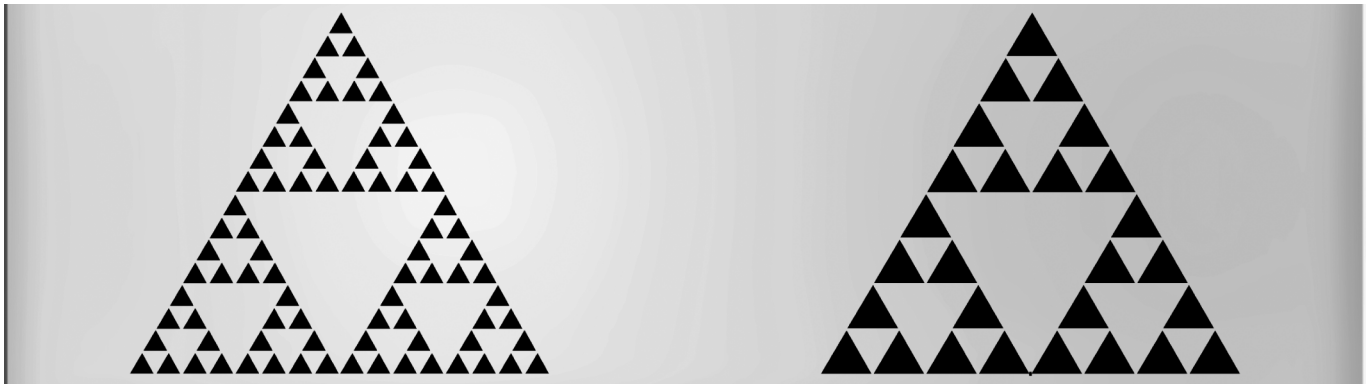

Рис. 10. Топология распределения звуковых данных по треугольнику Серпинского (фрактальная структура)

ный оператор для создания безымянных (анонимных) функций. Он основан на *лямбда*-исчислении Чёрча и позволяет определять функции прямо в месте их использования, без присвоения имени. Defun — определяет новую функцию. Let —используется для создания локальных переменных и присвоения им значений в рамках определенного блока кода. Loop — задаёт организацию циклов и повторяющихся операций. Это ключевой параметр (keyword argument) функции make-array, который позволяет задать начальное содержимое создаваемого массива. При вызове make-array параметр :initial-contents принимает в качестве значения список (или вложенные списки), которые определяют начальные элементы массива. Структура передаваемого списка должна соответствовать размерности создаваемого массива. В языке программирования Nyquist параметр :initial-contents используется в функции make-array для задания начальных значений элементов создаваемого массива. Далее построение массивов:

Двумерный массив (2×2):
```
(setf matrix (make-array '(2 2)
 :initial-contents '((1 2) (3 4))))
;; Результат: #2A((1 2) (3 4))
```

Одномерный массив:
```
setf arr (make-array 4 :initial-contents '(10 20 30 40)))
;; Результат: #(10 20 30 40)
```

Трёхмерный массив (2×2×2):
```
(setf cube (make-array '(2 2 2)
 :initial-contents '(((1 2) (3 4))
 ((5 6) (7 8)))))
;; Результат: #3A(((1 2) (3 4)) ((5 6) (7 8)))
```

Collect — на языке LISP (в контексте макроса iterate или аналогичных расширений) — это специальная конструкция для накопления элементов в коллекцию (обычно в список) во время итерации. Collect позволяет в цикле собирать результаты вычислений в список: на каждой итерации значение выражения добавляется в результирующий список, который возвращается после завершения цикла. Ниже приведена программная структура клеточного автомата по правилу 90 (рисунок №11).

В графическом виде эволюции развития клеточных автоматов по правилам 30 и 90 будут выглядеть следующим образом (рисунок № 12), синтезируемая волновая структура будет развиваться в зависимости от представленной на этом рисунке топологии, конечная волновая форма (*см*. рисунок №14).

Вместо B1 = 1, указывающего на то, что ячейка должна быть окрашена в черный цвет и B1 = 0, указывающего на то, что ячейка должна быть окрашена в белый цвет, в музыкальной модели это будет соответствовать *включению* и *выключению* определенных звуковых объектов. Например, мы имеем массив осцилляторов[59]:

Osc 60 — Osc 65 — Osc 67 — Osc 70 — Osc 75 — Osc 76 — Osc 79

Если «включены» только 1-й и 3-й элементы, это приведет к аккорду (*сумма* (Osc 60) (Osc 67)). Таким образом, каждый массив или уровень автоматов будет соответствовать аккорду, и последовательность аккордов будет меняться с течением времени по мере развития автоматов. Эта функция весьма универсальна, поэтому пользователь может указать базовый звуковой массив[60],

[59] Прим. автора. Вызываем одномерный массив состояний, каждое из которых может быть либо 0, либо 1. Музыкальная аналогия для этой идеи такова — вместо ряда состояний у нас есть массив звуков. Если состояние имеет значение 1, то этот звук «включён», если 0 — то он «выключен». Кроме того, предположим, что вы выбрали какой-то метод, с помощью которого можете комбинировать включённые звуки. Примером будет являться тот же массив осцилляторов: Osc 60 — Osc 65 — Osc 67 — Osc 70 — Osc 75 — Osc 76 — Osc 79. В данном случае, мы имеем элементарный набор звуков. В этой музыкальной структуре клеточных автоматов мы выбираем комбинирующую функцию, чтобы было SUM, ради простоты. Теперь, если включить состояния 2,3 и 6, мы получим звук Nyquist (sum (sc 65) (sc 67) (sc 76)), который является всего лишь простым аккордом. Эволюция этого автомата будет представлять собой последовательное развитие аккордов.

[60] Звуковые массивы — это просто массивы осцилляторов. В выборках используется фрактальное правило Вольфрама (90)

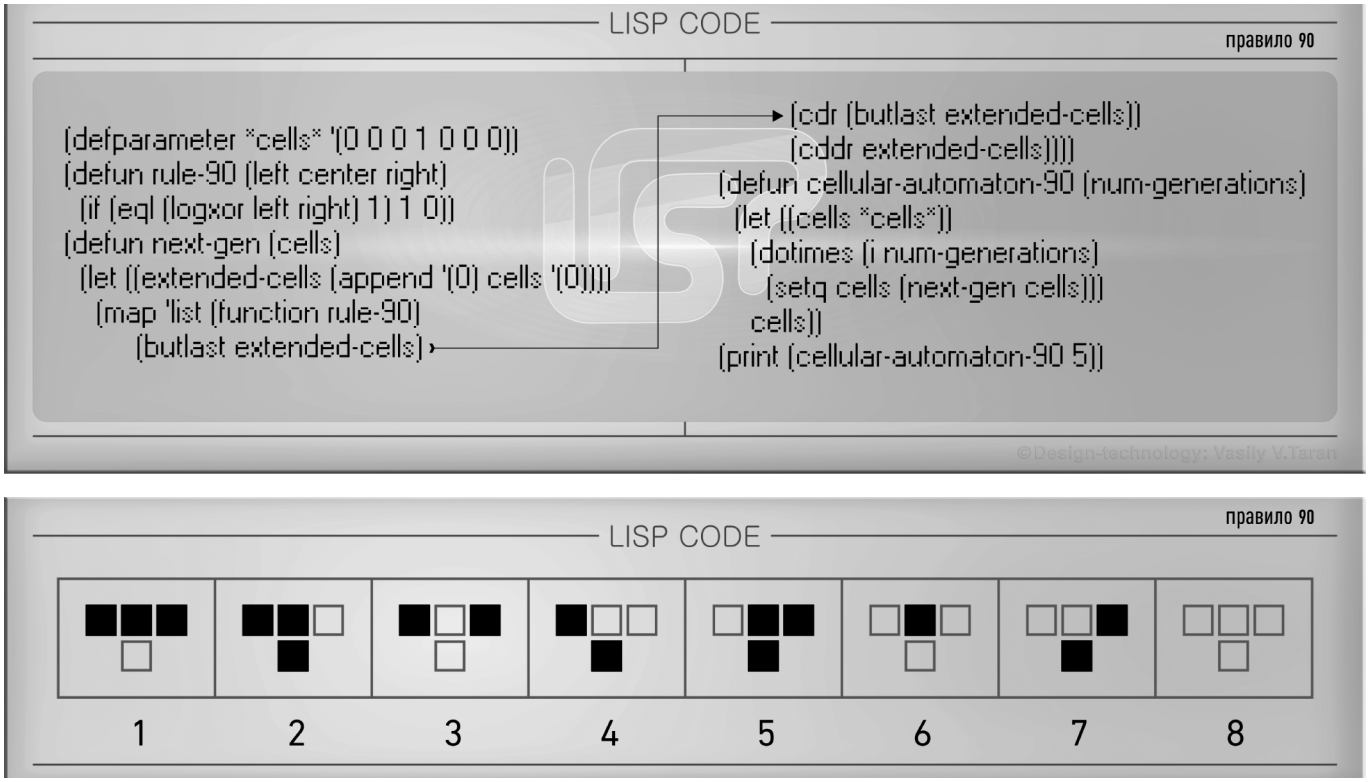

Рис. 11. Структура программного кода по клеточному автомату в соответствии с *правилом* 90

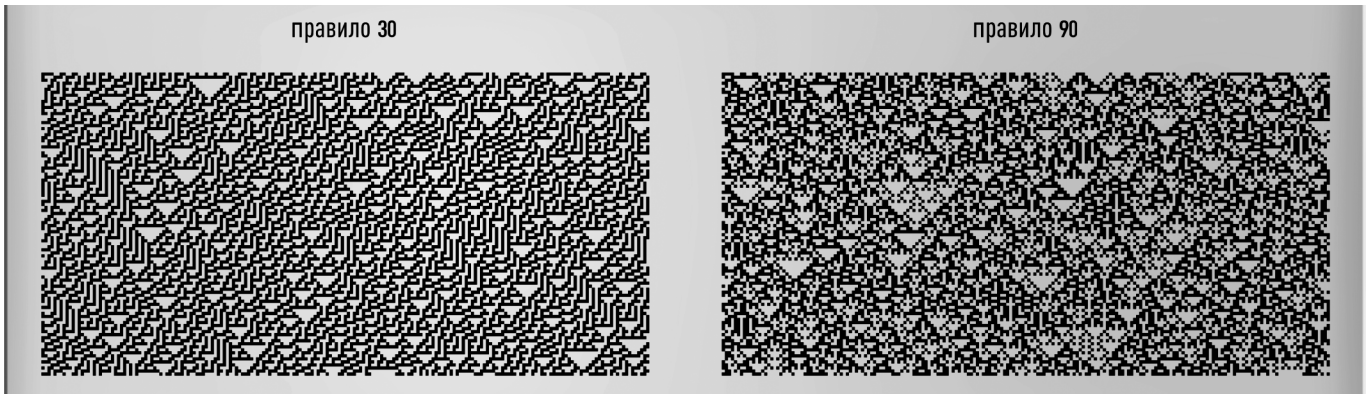

Рис. 12. Эволюция развития клеточного автомата слева по *правилу* 30, справа по правилу 90

продолжительность каждого шага, а также какую комбинирующую функцию необходимо использовать для объединения активированных звуков вместе. Эта схема позволяет пользователю применить любое выражение для создания звуков. На примере, указанном выше (рисунок № 8) реализована модель одномерного (элементарного) клеточного автомата[61] (работающего по правилу 30) который может быть привязан к любому массиву звуков, использующему любую *комбинирующую функцию* на активированных звуках. Для этого используется функция cell-aut (cell-aut.lsp), которая включает набор следующих параметров:

1) Массив звуковых объектов, задаваемых с помощью вычисляемых выражений.
2) Длительность каждого временного шага (также длительность вычисления звуковых объектов).
3) Правило обновления для использования при эволюции массива, заданное числом (0 — 255).
4) Число итераций/поколений автоматов должно позволять себе расти.

и хаотическое правило Вольфрама (30). Объединяющая функция — это просто SUM.

[61] Прим. автора. Элементарные клеточные автоматы — это простейший класс одномерных клеточных автоматов. Элементарные клеточные автоматы имеют два возможных значения для каждой ячейки (0 или 1) и правила, зависящие только от родительских и соседних родительских значений. Эволюция элементарного клеточного автомата может быть полностью описана двумерной таблицей, где запись (i, j) соответствует состоянию j в поколении i. Так как есть 2*2*2 = 23 = 8 возможных бинарных состояний для трёх ячеек, соседних с данной ячейкой, имеется в общей сложности 28 = 256 элементарных клеточных автоматов.

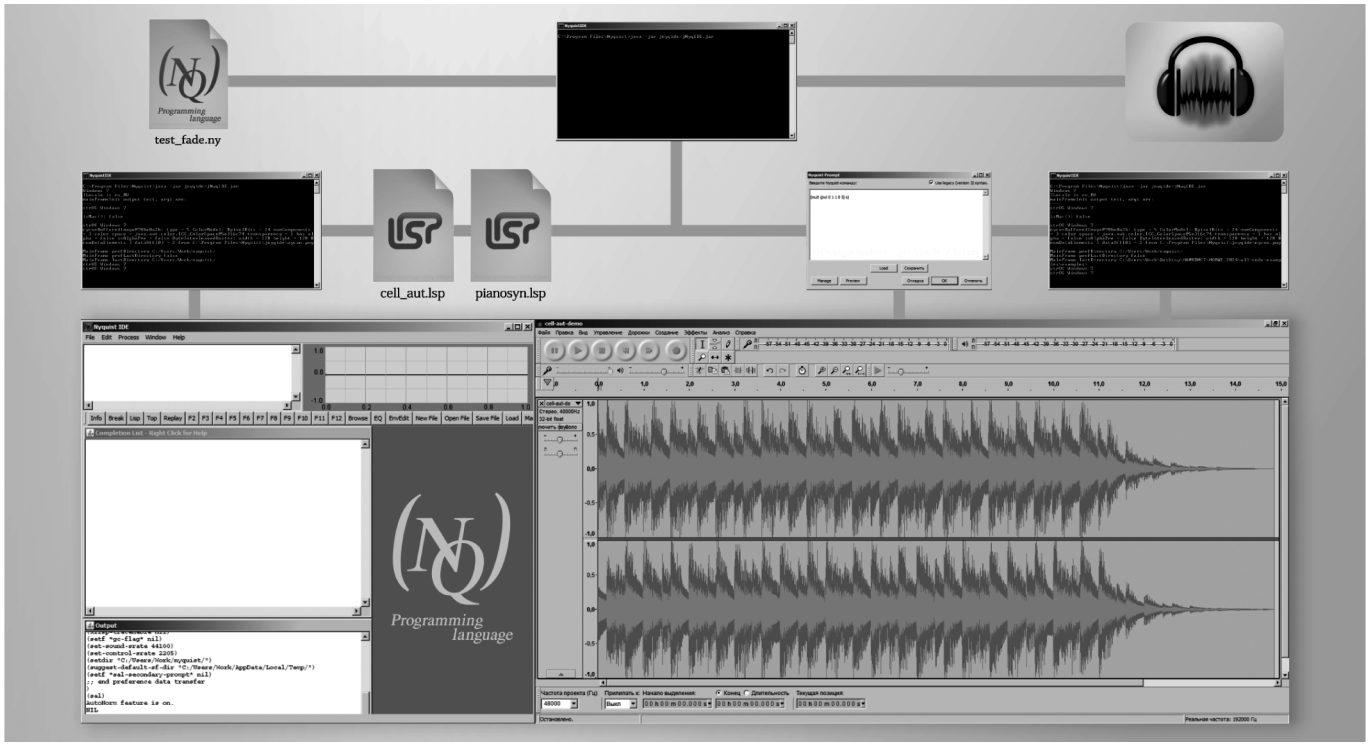

Рис. 13. Схема функционирования программного комплекса Audacity® с возможностью имитации клеточных автоматов для работы со звуком

Пример комбинирующей функции

(cell-aut <num iterations> <combining function> <sound list> <update rule>)

Расширения cell-aut.lsp и pianosyn.lsp подключаются к NyquistIDE и взаимодействуют с редактором Audacity®, это открывает большие возможности для интерфейсно-ориентированного редактирования синтезированного аудиоматериала и позволяет на уровне отдельных программных фрагментов кода манипулировать сигналом внутри аудиоредактора посредством приглашения NyquistPrompt.

Расширение cell-aut.lsp включает в себя функцию cell-aut-demo для проверки алгоритма элементарного клеточного автомата с возможностью захвата простого сигнала из pianosyn.lsp[62], она определяется следующим образом:

(defun cell-aut-demo ()

(play (scale 0.5 (cell-aut (cell-aut-major-scale) 0.2 30 80))))

[62] Прим. автора. Демонстрация работы алгоритма элементарного клеточного автомата в качестве представления синтеза в виде звуков фортепьяно. ***pianosyn***.lsp — подключаемая библиотека, входящая в пакет Nyquist, содержащая нотации звуков фортепьяно, а также некоторые алгоритмические функции, описанные на языке LISP и адаптированные под функциональный стиль Nyquist.

Выполнение данной функции обеспечит вывод аудиосигнала и даст возможность прослушать результат образца функции автоморфизма ячейки. Эксперимент показал, что выбранное правило CA — 30 успешно производит хаотичный синтез парных слоёв (eq arg1 t) (eq arg2 t), нотируемых расширением pianosyn.lsp, физическим результатом эксперимента служит короткий отрывок аудиокомпозиции с эмуляцией клавиш фортепьяно выведенных для дальнейшего волнового и спектрального редактирования в редактор Audacity®.

Поскольку автоматы различаются по своей эволюции или правилам обновления строк, вы можете указать автомат, который хотите использовать, просто указав правило обновления.

Правило обновления (<update rule>) — это просто функция, которая принимает 3 (t/nil) значения и возвращает t или nil. Три входных значения будут факторами зависимостей ячейки, а выходные —положением текущей ячейки. Пример подобной записи с аргументами представлен на рисунке № 7.

Разобравшись с итерацией, комбинирующей функцией и правилами обновления можно подытожить всё вышеизложенное следующим мультиблоковым алгоритмом, демонстрирующим общую работу клеточных автоматов в Nyquist-программе (рисунок №14). Фактически, это финальная универсальная схема базового алгоритма клеточных автоматов вне зависимости от вариатив-

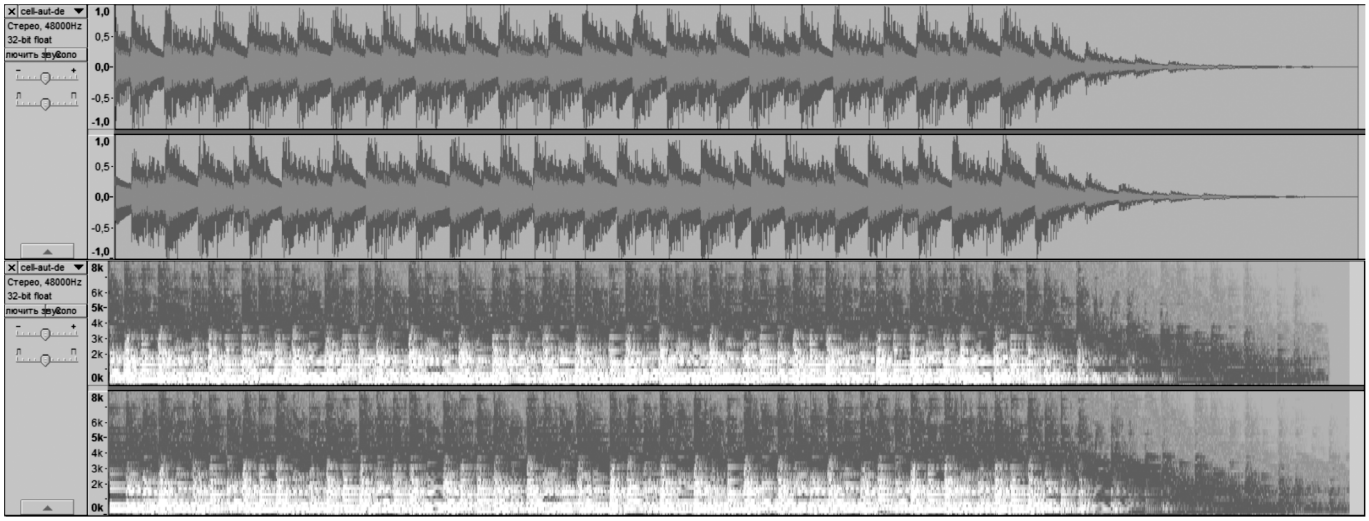

Рис. 14. Результат воспроизведения эксперимента Данненберга-Льюис*. *Звуковая форма в волновом и спектральном представлении, отображаемая в Audacity® и характеризующая звучание фортепианных клавиш через призму *правила* 30 клеточных автоматов

ности правил и используемых языков программирования, которая может использоваться в качестве полезной модели для реализации мультитональных вариаций аудиальных структур на основе инвариантного синтеза в соответствии с использованием выбранных оператором правил.

Блок-схема иллюстрирует функциональную работу (*инициализация*/*обработка*) элементарных клеточных автоматов применительно к аудиосигналу. Цикл *ввода*/*вывода* звука осуществляется через NyquistIDE с экспортом в Audacity® и возможностью дальнейшей коррекции звуковой формы непосредственно в редакторе через приглашение языка Nyquist. Тема *связки* библиотечных расширений «cell-aut.lsp» и «pianosyn.lsp» с целью воспроизведения алогритмически-комбинаторных аудиосигналов будет рассматриваться автором в дальнейших публикациях.

Заключение

Итак, мы рассмотрели с позиций научного анализа практическую возможность реализации алгоритма работы элементарных клеточных автоматов в соответствии с использованием *вариативных* правил перехода, характеризующих *состояние* и *поведение* клеток на сетчатом холсте для производства новых синтетических звуков и синтезированных аудиопартий однородного и неоднородного типа.

Эксперимент Данненберга-Льюис оказался успешно воспроизводим, а некоторые авторские уточнения и комментарии, соответствующие настоящему времени, позволяют расширить представления о методе синтеза, основанного на переменности правил клеточных автоматов.

Принцип работы клеточных автоматов можно успешно использовать в качестве метода генерации различной музыкальной фактуры, такой как алгоритмическая текстура, атональные мелодии, синтетические фонограммы, мультитональные симфонии.

В аудиоинформатике клеточные автоматы следует рассматривать как как вид универсального синтеза, позволяющий получить оригинальные результаты за относительно короткий промежуток времени. Аппаратная независимость клеточных автоматов позволяет использовать их в качестве мобильного синтез-генератора в любой вычислительной системе. Эффективность её работы будет определяться не физическими параметрами технической оснастки компьютера, а оригинальными решениями в области подходов к составлению формульных описаний, выраженных в виде точных конструкций с грамотно изложенным синтаксисом, представляющим программно-кодовые предписания для быстрой интерпретации процессором.

Квинтэссенция вышеизложенных положений настоящего заключения позволяет сделать вывод о том, что клеточные автоматы для *аудиоинформатики* и *аудиоинженерии* соответственно, представляют собой питательную почву для научных исследований и прикладных экспериментов, которые, несомненно, в обозримой перспективе будут способствовать многовекторному приращению научных знаний не только в области компьютерных наук (информатики), но и в других точных, а также естественнонаучных дисциплинах.

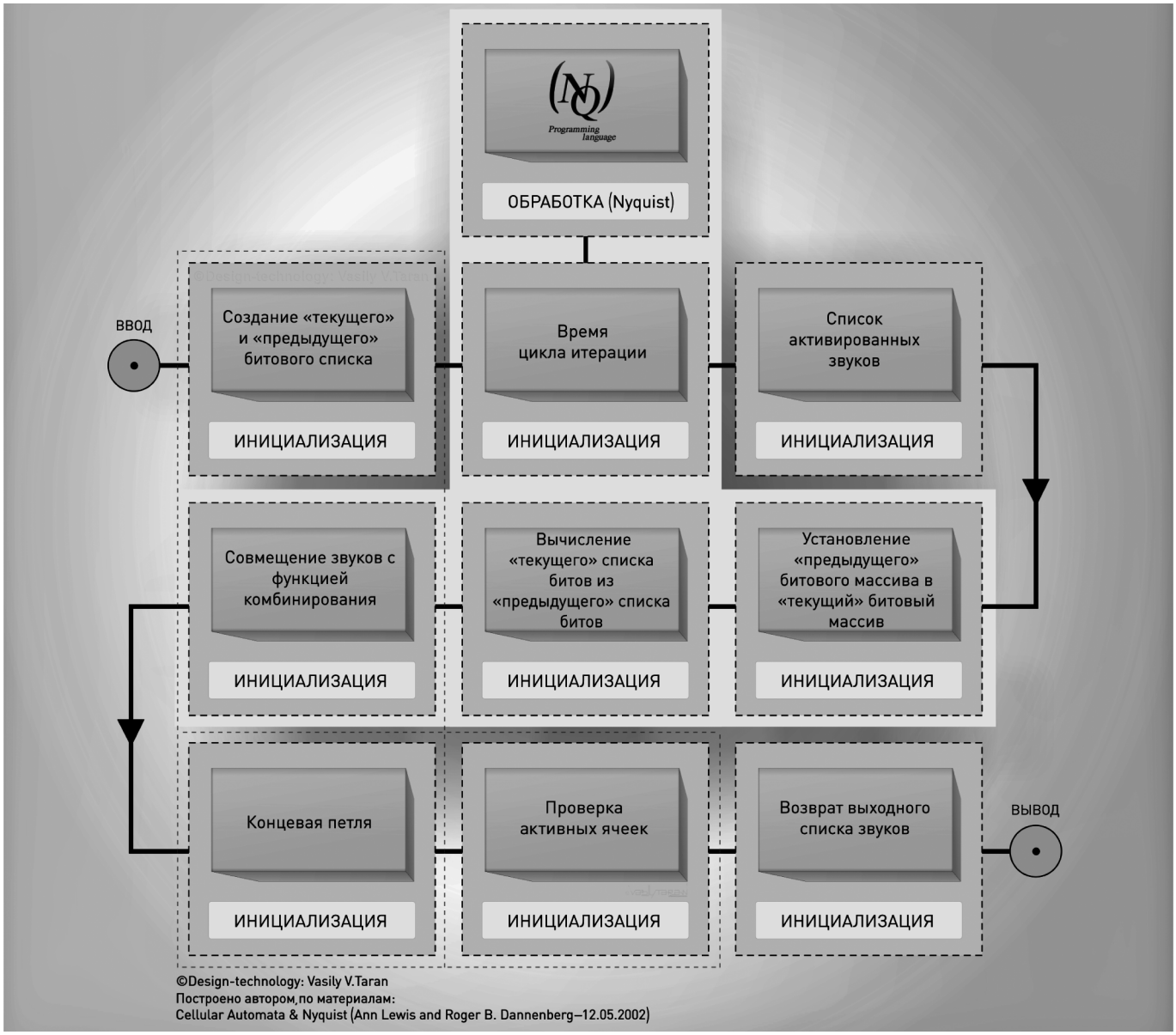

Рис. 15. Универсальная алгоритмическая блок-схема общей Nyquist-программы, иллюстрирующая общие принципы функционирования клеточных автоматов

ЛИТЕРАТУРА

1. Компьютерная программа Nyquist IDE v.3.15 / Файл директории (C:\Users\Name\nyquist) // OS: MS Windows, GNU Linux /// Полная реализация — Jesse Clark, David Hovard, David Movatt, David Deangelis, Roger B. Dannenberg. — 2002–2018. [электронный источник, автономная компьютерная программа].
2. Компьютерная программа Audacity® v.2.1.3 / Файл директории (C:\Program Files (x86)\Audacity) // OS: MS Windows, GNU Linux /// Полная реализация — Gale Andrews, Arturo «Buanzo», James Crook, Roger B. Dannenberg, Steve Daulton, Vaughan Johnson, Greg Kozikowski, Paul Licameli, Peter Sampson, Martyn Shaw, Bill Wharrie. — 1999–2017. [электронный источник, автономная компьютерная программа].
3. Официальный сайт редактора Audacity® — [электронный WEB-ресурс, дата обращения к источнику: 25.01.2026].
4. Таран В.В. Компьютерный аудиосинтез штатными средствами Audacity® с возможностью имитационного дизайн-моделирования на языке Nyquist/ В.В. Таран//Современная наука актуальные проблемы теории и практики. Серия: Естественные и технические науки — 2020. — №1. — С.115–129. [ISSN 2223-2966].
5. Таран В.В. Язык программирования Nyquist: настоящее время и перспективы его развития в области компьютерной аудиоинженерии и аудиоинформатики / В.В. Таран// Современная наука: актуальные проблемы теории и практики: Серия: Естественные и технические науки — 2020. — №4. — С.135–153. [ISSN 2223-2966]. (DOI 10.37882/2223–2966.2020.04.37).

[63] Прим. автора. Вместо языка программирования Nyquist (LISP) можно использовать любой другой язык программирования для реализации подобных экспериментов. Использование именно языка Nyquist обусловливается его практической ориентированностью на выполнение операций именно с аудиоданными. Об этом много сказано в начале статьи.

6. Touretzky, David S. Common LISP: a gentle introduction to symbolic computation /Carnegie Mellon University///Copyright (c) 1990 by Symbolic Technology, Ltd.//// Published by The Benjamin/Cummings Publishing Company, Inc. — 587 p. (ISBN0-8053-0492-4).
7. Dannenberg R.B. Nyquist Reference Manual Version 3.24 // Carnegie Mellon University — School of Computer Science / Pittsburgh, PA 15213, U.S.A. 05.03. 2025, 289 p.
8. Electronic repository of projects in the Nyquist programming language at Carnegie Mellon University. — Cellular Automata, 12.05.2002. (электронный WEB-ресурс: https://www.cs.cmu.edu/~music/nyquist/extensions/cellautomata/cellularautomata.html; дата обращения к ресурсу: 25.01.2026)
9. Wolfram, S. A New Kind of Science. — Champaign: Wolfram Media, 2002. —1197 p.
10. Von Neumann, J., Burks, A.W. Theory of Self-Reproducing Automata. — Urbana and London: University of Illinois Press, 1966. — 388 p.
11. Toffoli, T., Margolus, N. Cellular Automata Machines: A New Environment for Modeling. — Cambridge: MIT Press, 1987. — 259 p.
12. Kari, J. Theory of Cellular Automata: A Survey // Theoretical Computer Science. — 2005. — Vol. 334, № 1-3. — P. 3–33.
13. Gardner, M. The Fantastic Combinations of John Conway's New Solitaire Game «Life» / M. Gardner // Scientific American. — 1970. — Vol. 223, № 4. — P. 120–123.

Журнал «Современная наука: актуальные проблемы теории и практики»