УДК 004

О создании первых компьютеров в СССР

Крайнева И.А. (Институт систем информатики СО РАН),
Шилов В.В. (Научно-исследовательский университет «Высшая школа экономики»)

В этой публикации представлен отклик на статью С.П. Прохорова «Основополагающий вклад Академии наук СССР в становление компьютерных наук и компьютерных технологий» в журнале «Вестник Российской Академии наук», 2023, т. 93, № 10, с. 980–988.

Ключевые слова: *Академия наук, ЭВМ, М-1, МЭСМ, И.С. Брук, С.А. Лебедев, Б.И. Рамеев, С.Л. Соболев, А.А. Ляпунов, А.П. Ершов, конференция SoRuCom.*

1. Введение

В начале текущего 2024 г. мы подготовили рецензию на статью к.ф.-м.н. С.П. Прохорова (Институт истории естествознания и техники им. С.И. Вавилова РАН) «Основополагающий вклад Академии наук СССР в становление компьютерных наук и компьютерных технологий» и отправили ее в «Вестник РАН». Нам хотелось обратить внимание редакции на некоторые утверждения в работе С.П. Прохорова, которые, на наш взгляд, носят неоправданный, а порой и конъюнктурный характер, искажают уже установленные факты. Мы получили заключение редакции, в котором нам предлагалось написать новую статью, а не использовать уважаемый журнал «для выяснения отношений». Ознакомившись с ответом, мы решили, что писать еще одну статью нет необходимости, поскольку никаких открытий мы в ней не представим. Поэтому мы поместили здесь наше письмо, ответ редакции и некоторые комментарии к нему. Для удобства восприятия мы разделили наше письмо на подразделы.

2. Письмо в Вестник РАН

2.1. Преамбула

Интерес к созданию и развитию отечественной вычислительной техники в инженерно-научном сообществе России достаточно устойчив. То, что статья на эту тему опубликована в «Вестнике РАН», показательный момент, тем более, что ранее в данном журнале подобных, кажется, не было. Приветствуя и сам факт ее появления, и содержание в целом, мы, тем не

менее, хотели бы дополнить ее некоторыми свидетельствами, которые были упущены, и замечаниями в ответ на не совсем корректно представленные автором факты и события. Судя по всему, статья приурочена к отмечавшемуся в декабре 2023 г. 75-тилетию авторского свидетельства, полученного И.С. Бруком и Б.И. Рамеевым на изобретение «Автоматическая цифровая вычислительная машина». Жанр юбилейной статьи – жанр особый, в них не предполагается наличие каких-то новых фактов, открытий. Задача такой статьи – изложить уже известные факты, относящиеся к юбилею. Но автор утверждает, что некоторые архивные документы публикуются им впервые, что не соответствует действительности. Ссылок на новые архивные документы в статье нет; документы известны, в том числе и самому автору, который на них уже ссылался в своих ранних работах. Столь же спорен тезис о «малоизвестных» фактах, приводимых в статье.

2.2. Изучение истории информатики

Вначале мы хотели бы предложить короткий экскурс, чтобы показать, что Институт истории, естествознания и техники им. С.И. Вавилова РАН не есть единственный исследователь этой темы, как это декларировано в заключении статьи. Впервые к истории вычислительной техники (ВТ) обратился тогда еще д.ф.-м.н. (академик, 1984) А.П. Ершов, создатель новосибирской школы программирования. В октябре 1967 г. он записал интервью с академиком М.А. Лаврентьевым («Первые годы развития советской вычислительной техники») [25]. Затем в 1976 г. вышла работа А.П. Ершова (ВЦ СО АН СССР) и М.Р. Шура-Буры (ИПМ им. М.В. Келдыша РАН) [6], где тема создания первых ЭВМ в СССР стала контуром текста о становлении программирования в СССР. Но устойчивый тренд в исследовании по истории ВТ сформировался только в 1990-е годы. Вышли первая фундаментальная монография [13], сборник статей воспоминаний представителей новосибирской школы программирования [22] и другие книги и статьи. Был создан Виртуальный компьютерный музей [4], ставший агрегатором материалов по истории ВТ и смежных областей. В 2006 г. состоялась первая регулярная конференция «Развитие вычислительной техники в России, странах бывшего СССР и СЭВ» (SoRuCom). В ее организации неоднократно принимал участие и С.П. Прохоров, о чем он не упомянул. Конференция SoRuCom собрала под своим крылом инженеров, техников, математиков и физиков, которые принимали непосредственное участие в советских проектах по созданию ЭВМ, программного обеспечения, в формировании кадрового потенциала и прочее. К участию в ней присоединились и историки науки и техники, – к сожалению, их всё еще не

слишком много [23]. С тех пор были освещены многие вопросы истории отечественной ВТ [15], но кое-что остается еще малоизученным, хотя мы практически перешли на микроуровень исследований.

2.3. Хронологические рамки

С.П. Прохоров в своей работе представил довольно широкую картину формирования и решения проблем с вычислительной техникой в СССР в предвоенный и послевоенный периоды. Поэтому акцент на 1948–1951 гг., как хронологический фрейм статьи, если выйти за пределы ее «юбилейности», не совсем корректен. Начнем с того факта, что идея создания Института точной механики и вычислительной техники, который стал флагманом Академии наук в деле развития цифровой ВТ, зародилась еще в 1946 г. Соответствующий вопрос рассматривался на заседании Отделения технических наук АН СССР. Более того, в это время работал межинститутский семинар по вычислительной технике под руководством академика Н.Г. Бруевича, где и вызрела идея нового института [5]. Образован он был по Постановлению Совета Министров СССР № 2369 от 29.06.1948. Далее. Важное с точки зрения формирования проблематики более мощной вычислительной техники выступление академика М.А. Лаврентьева прозвучало в октябре 1947 г. на торжественном заседании АН СССР по случаю 30-летия Октябрьской революции, а опубликовано оно было в 1948 г. [9] Что касается верхней границы – 1951 г. – она отсекает другое существенное обстоятельство: создание операторного метода программирования А.А. Ляпуновым. Мы считаем это актуальным, поскольку цифровая вычислительная техника без этого просто мертва!

Как справедливо написал С.П. Прохоров, на первых ЭВМ, появившихся в 1951 г., математики программировали и даже решали важные задачи. Но назван в числе таких математиков лишь академик С.Л. Соболев. Хотя известно, например, что в Москве к программированию был привлечен математик Н.Н. Мейман (который с 1947 г. по совместительству, а в 1948–1954 гг. на постоянной основе заведовал вычислительным бюро Института физических проблем, входившим в теоретический отдел под руководством Л.Д. Ландау). Приведем воспоминания одного из авторов первой академической книги по программированию д.ф.-м.н. А.А. Абрамова. Он свидетельствовал, что в работе над БЭСМ в ИТМиВТ (с 1951 г.) принимали участие и сотрудники отдела приближенных вычислений, руководимого чл.-корр. АН СССР Л.А. Люстерником. «Отдел участвовал в разработке математической структуры БЭСМ. Кроме этой непосредственной работы Лазарь Аронович организовал семинар по программированию, где обсуждались общие вопросы математического обслуживания ЭВМ. А затем Лазарь Аронович подобрал коллектив авторов

для написания учебной книги по программированию и работе математиков на ЭВМ. Он сказал, что нужно сделать так, чтобы когда у нас появятся ЭВМ (одновременно с работой в ИТМиВТ промышленность начала работу над созданием ЭВМ «Стрела»), эта книга уже была». Авторами книги «Решение математических задач на автоматических цифровых машинах» стали Л.А. Люстерник, А.А. Абрамов, В.И. Шестаков и М.Р. Шура-Бура [7]. Она вышла в 1952 г. с грифом «секретно», но он вскоре был снят. В Киеве первая пробная задача была выбрана из области баллистики с весьма существенными упрощениями (не учитывалось сопротивление воздуха). Программу составили математики С.Г. Крейн и С.А. Авраменко. Контрольный расчет они выполнили непосредственно в двоичной системе, что обеспечило возможность проверки ЭВМ по циклам и по тактам, наблюдая по индикации пульта управления за правильностью выполнения программы (осень 1951 г.) [13. С. 38]. Еще ранее, в 1950 г. в Ленинграде, Л.В. Канторович, после войны возглавлявший отдел Института математики и механики ЛГУ, разработал и прочел для сотрудников ЛОМИ им. В.А. Стеклова и аспирантов ММФ ЛГУ курс программирования для абстрактной одноадресной машины [14]. Наконец, в 1952 г. А.И. Китов защитил первую кандидатскую диссертацию по программированию.

2.4. О корифеях и основателях

Не отрицая того факта, что С.Л. Соболев проявлял живой интерес к ЭВМ и программированию, что вполне естественно, учитывая его работу в Советском атомном проекте, мы считаем, что называть его «родоначальником отечественной школы программирования» оснований нет. У него нет работ по теории программирования. Соболев возглавил кафедру вычислительной математики механико-математического факультета Московского государственного университета, образованную в 1949 г., и лишь в 1952 г. пригласил в качестве профессора А.А. Ляпунова для чтения курса «Программирование». В архиве академика А.П. Ершова сохранился конспект лекций Ляпунова [12]. Известно, что в процессе чтения курса (1952–1953 гг.), Алексей Андреевич выезжал в Феофанию под Киевом, чтобы знакомиться с работой МЭСМ. Он «заметил, что структура программы включает в себя операторы из небольшого набора типовых операторов и может быть представлена формально в виде строки, соответствующей последовательности операторов программы. На языке операторных схем программа представляется как схема, соответствующая управляющему графу программы, и совокупность спецификаций каждого оператора» [10]. Эти революционные идеи легли в основу автоматизации программирования

и создания первых отечественных трансляторов (программирующих программ, 1954 г.). Впервые исследования Ляпунова были опубликованы в 1958 г., поскольку он работал в закрытом режиме советского атомного проекта [11]. Таким образом, одному С.Л. Соболеву никак нельзя приписывать какое-либо первенство в области программирования на ЭВМ, и уж тем более основание школы в этой области.

Что касается развертывания проблематики ВТ в ИТМиВТ и в Специальном конструкторском бюро (СТБ-245), то тезис С.П. Прохорова о том, что «заниматься созданием электронных машин ни одна из этих организаций не планировала», выглядит несколько поспешным. Мы согласны с тем, что Н.Г. Бруевич отдавал приоритет аналоговой ВТ. Но С.П. Прохоров упомянул, что к октябрю 1948 г. И.С. Бруком и Б.И. Рамеевым (Энергетический институт им. Г.М. Кржижановского) был составлен документ «Проектные соображения о создании лаборатории ИТМиВТ для разработки и строительства автоматической цифровой вычислительной машины». Известно, что был и другой документ – «Проектные соображения по организации особого конструкторского бюро для разработки и строительства автоматических цифровых вычислительных машин» [16]. Они различаются по объему и по содержанию. Например, если в первом документе речь идет о создании одной АЦВМ (20-тиламповая электронная схема средней сложности), то во втором – о двух (АЦВМ и БВМ – быстродействующая вычислительная машина – для интегрирования дифференциальных уравнений). Кроме того, при обсуждении положения дел в ИТМиВТ на заседании Отделения технических наук АН СССР в июле 1949 г., о котором пишет С.П. Прохоров, Н.Г. Бруевич сообщил, что создание ЭВМ они обсуждали с чл.-корр. И.С. Бруком осенью 1948 г. на специальном научно-техническом совещании в ИТМиВТ. После консультации с министерствами и Гостехникой был подготовлен проект специального решения Правительства по организации и обеспечению работ [1. Ф. 1559. Оп.1. Д. 4. Л. 60–61]. Возвращаясь к «Проектным соображениям» Брука и Рамеева, скажем, что они содержали подробные предложения по программе исследовательских, конструкторских и производственных работ, научно-производственным связям с другими НИИ и предприятиями, составу лаборатории, материальным и финансовым затратам и т.д. Что остановило продвижение этих проектов, требует дополнительного исследования. Возможно, не последнюю роль сыграла кампания по борьбе с космополитизмом, которая затронула и ИТМиВТ. В этом контексте показательно письмо Министра госбезопасности СССР В.С. Абакумова, на которое ссылаются Ю.А. Жданов в своем письме в Секретариат ЦК от 08.03.1950 [19. Ф.17. Оп. 118 Д. 758. Л. 22] и главный ученый секретарь президиума АН СССР А.В. Топчиев в письме секретарю ЦК ВКП (б) Г.М. Маленкову от 05.02.1950 [19.

Ф.17. Оп. 118 Д. 758. Л. 27]. Авторы указывают на справедливость критики В.С. Абакумовым работы ИТМиВТ и его и.о. директора. Речь шла «о засоренности кадров ИТМиВТ АН СССР» и прочих недостатках в его работе.

В записке на имя Г.М. Маленкова от 08.03.1950 Отделом пропаганды ЦК, сообщалось, что помимо этого «в институте практикуются частные подряды на вычислительные работы (лаборатория Гутенмахера), в результате чего в институт стекается в виде технических условий и других сведений информация из различных военных учреждений о советской бомбардировочной авиации, бомбардировочных прицелах и их точности, типах бомб, ракетах дальнего действия и пр.» [19. Ф.17. Оп. 118 Д. 758. Л. 23–24]. Хотя названные специалисты занимались проблематикой бомбометания еще в годы войны, имели публикации, а в условиях нарождающихся реалий холодной войны, очевидно, продолжали сотрудничество с военными на условиях подряда, обвинения в нарушении секретности грозили большими неприятностями. В результате академик Н.Г. Бруевич так и не был утвержден в качестве директора ИТМиВТ, и в 1950 г. его сменил на этом посту академик М.А. Лаврентьев. К слову, объясняя снятие Н.Г. Бруевича нарушением режима секретности, С.П. Прохоров в одной из своих публикаций связывает это с «Ленинградским делом» [17]. Опубликованные нами документы дают этому факту более реальное объяснение [8].

2.5. МЭСМ и М-1

С.П. Прохоров пишет: «М-1 – первая в мире ЭВМ, в которой все логические схемы были выполнены на полупроводниках [3], благодаря чему она получилась компактной и экономичной. Компьютер содержал в себе всего 730 электровакуумных ламп, его производительность при работе с “быстрой” памятью составила 20 тыс. операций в секунду для сложения и 500 операций в секунду для умножения». Все так, но не без лукавства. При работе с памятью на магнитном барабане производительность М-1 составляла 20 оп/сек. Можно процитировать самого И.С. Брука: «На это чудо техники, которое давало 15–20 не тысяч, не миллионов, а 15–20 операций в секунду <…> приезжали смотреть и президент Академии наук СССР А.Н. Несмеянов, и другие видные советские ученые и государственные деятели» [13. С. 183]. Однако эту цифру автор не называет, – похоже, чтобы постулировать преимущество М-1 перед известным американским компьютером UNIVAC I, который начали эксплуатировать в середине 1951 г., и который выполнял около 2000 сложений в секунду... (Заметим, что завершенная в 1954 г. ЭВМ М-2 имела производительность около 2000 оп/сек., в связи с чем возникает вопрос – как могло случиться, что производительность

более поздней машины снизилась на порядок?! А может быть, на самом деле она выросла на два порядка?). Думается, что вопрос о реальной производительности М-1 требует более пристального изучения (если оно, конечно, возможно на основе сохранившихся материалов), но в любом случае историк должен более строго относиться к приводимым им данным, не скрывая другие.

Далее С.П. Прохоров зачем-то поднимает давно решенный отечественными историками вопрос о первенстве: какая ЭВМ может считаться первой, «московская» М-1 или «киевская» МЭСМ. Он сообщает, что 15 декабря 1951 г. ЭВМ М-1 «была принята в эксплуатацию». Но на самом деле в этот день директором Энергетического института АН СССР академиком Г.М. Кржижановским был утвержден отчет «Автоматическая цифровая вычислительная машина М-1», хорошо известный историкам. Он действительно содержит подробное описание машины и ее узлов, но документом о приеме в эксплуатацию отнюдь не является! Более того, такой документ неизвестен. Согласно воспоминаниям разработчиков, эксплуатация машины началась в январе 1952 г., сам же И.С. Брук называл и другую дату – весна 1952 г. [2].

Уже в процитированном выше фрагменте статьи мы видим, как автор отклонился от фактов. Следующий абзац статьи, на наш взгляд, представляет собой образец сознательной фальсификации: «Есть косвенные данные, которые свидетельствуют о том, что до конца того же года была сдана в эксплуатацию вычислительная машина МЭСМ, которая конструировалась под руководством С.А. Лебедева» [21]. При этом хорошо известна выписка из протокола заседания Президиума АН СССР от 4 января 1952 года «О вводе в эксплуатацию малой счетной электронной машины. Докладчик проф. С.А. Лебедев». В ней говорится, что «Институт точной механики и вычислительной техники АН СССР совместно с Институтом электротехники АН СССР в IV квартале 1951 г. ввел в эксплуатацию малую счетную электронную машину, являющуюся первой в СССР быстродействующей электронной цифровой машиной, доведенной до состояния эксплуатации» [13. С. 40–41]. Эта публикация автору статьи известна, он на нее ссылается (хотя и с ошибкой), но почему-то называет ее «косвенными данными»... На наш взгляд, спор о первенстве МЭСМ и М-1, тем более, переводимый в политическую плоскость, контрпродуктивен, – обе эти замечательные разработки по праву можно назвать первыми отечественными ЭВМ. Повторим, что российское научное сообщество давно пришло к согласию в вопросе о первенстве ЭВМ. О том, что и МЭСМ С.А. Лебедева, и М-1 И.С. Брука были в числе первых ЭВМ в СССР, сказал в своем недавнем докладе на заседании Отделения нанотехнологий и

информационных технологий РАН директор ФИЦ «Информатика и управление» РАН академик И.А. Соколов.

Отходит от общепринятого научного тона и заключение статьи. Проанализируем по порядку: «К сожалению, надо констатировать, что в настоящее время мало внимания уделяется утверждению приоритета отечественной науки даже в ведущих научных направлениях. Это касается и вычислительной техники». При этом С.П. Прохорову наверняка известны недавние работы российских исследователей, в которых обосновывается приоритет тех или иных отечественных разработок в области ВТ как российского (до 1917 г.), так и советского периода (например, о логических машинах С.Н. Корсакова и А.Н. Щукарёва, арифмометре И. Штаффеля, проекте А.И. Китова и др.). Несомненно, работы С.А. Лебедева, Б.И. Рамеева, И.С. Брука и других советских ученых и конструкторов лежали в русле мировых трендов научно-технического развития. Однако не стоит забывать, что и М.А. Лаврентьев в 1947 г. в уже цитированной работе, и М.В. Келдыш в соавторстве с С.А. Лебедевым и Д.Ю. Пановым в отчете 1952 г. говорили о нашем существенном отставании в этой области [1. Ф. 1939. Оп. 2. Д. 2. Л. 60–61]. Этот и другие архивные документы 1950-х – 1960-х гг., в которых советские ученые сообщали руководству о реальном состоянии компьютерной отрасли, проанализированы в литературе [20]. Историку следовало бы вести речь не об искусственном «утверждении приоритетов», а о развертывании реальной картины.

Другой тезис С.П. Прохорова настолько же странен, насколько ложен. «Только благодаря усилиям Института истории естествознания и техники РАН мировым научным сообществом признан приоритет российских учёных в создании первой советской (и первой в континентальной Европе!) ЭВМ. А ведь этот факт десятилетиями оспаривался некоторыми зарубежными учёными», пишет он. О каком же «факте» идет речь? Дело в том, что определение «первая ЭВМ в континентальной Европе» С.П. Прохоров относит к М-1, которую он объявляет первой советской ЭВМ. Однако уже на протяжении 70 лет и в отечественных, и в зарубежных публикациях это определение используется только и исключительно применительно к МЭСМ! Автор не приводит ни одной ссылки на зарубежные публикации, отрицающие то, что «первая ЭВМ в континентальной Европе» была построена в СССР. В то же время можно привести множество ссылок на зарубежные публикации с описанием и высокой оценкой первых советских ЭВМ, в которых МЭСМ С.А. Лебедева, вслед за советскими авторами, называли именно так [26, 27].

3. Ответ редакции «Вестника РАН»

3.1. Не место для дискуссий

Ответ редакции – коллективное творчество. Основной текст его подготовлен рецензентами статьи С.П. Прохорова, передан нам Г.А. Заикиной, заместителем главного редактора журнала «Вестник РАН» и сопровожден послесловием редактора отдела журнала С.С. Поповым:

«Редакция журнала «Вестник РАН» ознакомила рецензента и редколлегию с Вашим письмом в ответ на публикацию статьи С.П. Прохорова. Заключение по рассмотрении этого вопроса сводится к следующему:

Ознакомившись с публикацией статьи С.П. Прохорова «Основополагающий вклад Академии наук СССР в становление компьютерных наук и компьютерных технологий», а также с Письмом в редакцию Вестника РАН И.А. Крайневой и В.В. Шилова и ответом С.П. Прохорова на это письмо считаем необходимым отметить следующее.

1. К сожалению, мы не имели возможности ознакомиться с отредактированной версией статьи С.П. Прохорова до ее публикации в журнале. Возможно это сняло бы некоторые замечания оппонентов.
2. Развернутое письмо И.А. Крайневой и В.В. Шилова в редакцию представляет интерес, поскольку история вообще и история техники, в частности, не является точной наукой, и трактовка ее во многом зависит от взглядов авторов. Квалификация авторов письма не вызывает вопросов. По нашему мнению, можно предложить авторам письма написать новую статью по обсуждаемой тематике и направить ее для публикации.
3. 12 декабря 2023 г. состоялось Общее собрание ОНИТ РАН, на котором академик И.А. Соколов выступил с докладом «Отделение нанотехнологий и информационных технологий РАН и развитие информатики и вычислительной техники в Советском Союзе и России», содержащем многие интересные исторические факты. По нашему мнению, было бы целесообразно направить академику И.А. Соколову или его коллегам, по его рекомендации, рассмотреть статью И.А. Крайневой и В.В. Шилова, если таковая будет представлена в редакцию.
4. Мы также не отказываемся принять участие в рассмотрении предлагаемой статьи.
5. Публиковать письмо И.А. Крайневой и В.В. Шилова, с учетом вышеизложенного, считаем нецелесообразным.

Таким образом, Вам предлагается написать собственную статью с изложением Вашего видения истории развития электронно-вычислительной техники в СССР и России. Публикация письма с критикой статьи С.П. Прохорова признана нецелесообразной.

Г.А. Заикина, заместитель главного редактора журнала «Вестник РАН»

P.S. Уважаемая Ирина Александровна, от себя могу добавить, что ваша гипотетическая статья (если Вы с Вашим соавтором решите её подготовить) должна носить самостоятельный характер. Выяснение отношений с вашим коллегой С.П. Прохоровым на страницах «Вестника РАН» контрпродуктивно.

С наилучшими пожеланиями С.С. Попов, редактор отдела «Вестник РАН»

3.2. Заключение

Ответ редакции наводит на серьезные мысли. Многие из нас, сталкиваясь с теми или иными бюрократическими структурами, получали в ответ на свои вполне конкретные обращения бессодержательные отписки. Таков, на наш взгляд, ответ из редакции. Из него следует, что рецензенты не придали значения тем утверждениям автора, на которые обратили внимание мы, что, видимо, говорит об уровне их подготовленности. Далее, содержащееся в п. 2 утверждение, что науки есть точные и неточные, как история, тоже, мягко говоря, удивило нас. Мы-то полагали, что есть наука и псевдонаука (не путать с не-наукой по Р. Фейнману) [18]. Но, в отличие от уважаемого физика, мы считаем историю наукой, которая имеет свои методы и методологию, стремится к точности, к реконструкции, которая, опираясь на факты, дает новое знание. Разумеется, можно вести речь о различных «трактовках» истории (или, в частности, тех или иных исторических событий). Но в нашем письме говорится не об иной, отличной от нашей, трактовке истории вычислительной техники С.П. Прохоровым, а о том, что он отметает и искажает (причем сознательно!) документально установленные факты.

Один из авторов этой статьи более 20 лет (иногда в очень жесткой форме) разбирает сочинения на тему истории ВТ. Выявлены ошибки, связанные чаще всего с вопиющей некомпетентностью их авторов. И это не «выяснение отношений», а естественное стремление очистить науку от сора. Эти статьи собраны в книге 2023 г. выпуска [24]. К сожалению, тема эта далеко не исчерпана, публикации, подобные прорецензированным в книге, продолжают появляться…

Список литературы

1. Архив Российской Академии наук (АРАН).
2. Брук И.С. Быстродействующая электронная вычислительная машина М-2 // Электричество. 1956. № 9. С. 14.
3. Были использованы трофейные купроксные выпрямители.
4. Виртуальный компьютерный музей Электронный ресурс URL: https://www.computer-museum.ru/ (дата обращения 05.01.2024).
5. Доклады на семинаре по вопросам математической техники (Известия Академии наук СССР ОТН, № 8 за 1946 г.; № 5 и № 11 за 1947 г.) // УМН. 1948. Т. 3. Вып. 2 (24).
6. Ершов А.П., Шура-Бура М.Р. Становление программирования в СССР. 2016. Изд-е 2-е, дополн. Электронный ресурс. URL: https://www.iis.nsk.su/files/articles/ershov_180.pdf (дата обращения 09.01.2024).
7. Книга и воспоминания А.А. Абрамова опубликованы на сайте ИСИ СО РАН «Редкие математические книги» http://books.mathtree.ru/book/lyusternik
8. Крайнева И.А. К истории ИТМИВТ АН СССР: Лаврентьев vs Бруевич (1948-1953) // Труды SoRuCom-2023/ Электронный ресурс. URL: https://www.iis.nsk.su/files/page/krayneva_i.a.pdf (дата обращения 05.01.2024).
9. Лаврентьев М.А. Пути развития советской математики // Изв. АН СССР. Сер. матем. 1948. Т. 12. Вып. 4. С. 411–416.
10. Любимский Э.З., Поттосин И.В., Шура-Бура М.Р. От программирующих программ к системам программирования (российский опыт) // Электронный ресурс URL: //www.iis.nsk.su/files/articles/mozaika.pdf (дата обращения 09.01.2024).
11. Ляпунов А. А. О логических схемах программ // Проблемы кибернетики. Вып. 1. М.: Физматгиз, 1958.
12. Ляпунов А.А. Принципы программирования. Лекции на ММФ МГУ 1952–1953 гг. // Электронный архив академика А.П. Ершова. Электронный ресурс URL: http://ershov.iis.nsk.su/ru/node/795235 (дата обращения 09.01.2024).
13. Малиновский Б.Н. История вычислительной техники в лицах. Киев: Фирма «КИТ», ПТОО «А.С.К.», 1995.
14. Мартыненко Б.К. Из истории отделения информатики математико-механического факультета Санкт-Петербургского университета // История информатики и кибернетики в Санкт-Петербурге (Ленинграде). СПб.: Наука, 2008. Вып. 1. С. 65.

15. На сайте SoRuCom можно ознакомиться с трудами шести прошедших конференций. Электронный ресурс. URL: https://www.sorucom.org/ (дата обращения 09.01.2024).
16. Политехнический музей. Ф. 221. Личный фонд Б.И. Рамеева. №№ по КП 27108/196 (26 л.) и 27108/197 (65 л.).
17. Прохоров С., Волков Д. Политика и первые отечественные компьютеры // Открытые системы. СУБД. 2022. № 2. С. 42–45.
18. Р. Фейнман, в своей книге «Наука, не-наука и все-все-все», АСТ, 2023 определенно говорит о неточности любой науки (Лекция 1 «Неточность науки», с. 23–45).
19. Российский государственный архив социально-политической истории (РГАСПИ).
20. Ревич Ю.В., Шилов В.В. Советская вычислительная техника в непубличных оценках современников // Труды SoRuCom-2017. С. 308–313.
21. Ссылка С.П. Прохорова ошибочная, должна быть указана с. 40.
22. Становление новосибирской школы программирования (мозаика воспоминаний). Новосибирск, 2001. Электронный ресурс URL: https://www.iis.nsk.su/files/articles/mozaika.pdf (дата обращения 05.01.2024).
23. Томилин А.Н., Крайнева И.А., др. Развитие вычислительной техники и ее программного обеспечения в России и странах бывшего СССР: страницы истории // История науки и техники. 2016. № 10.
24. Шилов В.В. Вырубить топором! М.: МАКС-Пресс, 2023. Электронный ресурс URL: https://elibrary.ru/item.asp?id=51746415 (дата обращения 05.02.2024).
25. Электронный архив академика А.П. Ершова. Электронный ресурс URL: http://ershov.iis.nsk.su/ru/node/782516 (дата обращения 09.01.2024).
26. Fitzpatrick A., Kazakova T., Berkovich S. MESM and the Beginning of the Computer Era in the Soviet Union // IEEE Annals of the History of Computing. 2006. Iss. 3. P. 4.
27. Ware W.H. (ed.). Soviet Computer Technology, 1959. Rand Corp. RM-2541, March 1, 1960. P. 44.

УДК 004, 009

SoRuCom-23 – VI Международная конференция по истории информатики

Крайнева И.А. (Институт систем информатики СО РАН),
Шилов В.В. (Национальный исследовательский университет «Высшая школа экономики»)

В статье представлен обзор содержания и проведенных исследований участниками VI международной конференции по истории информатики «Развитие вычислительной техники в России, странах бывшего СССР и СЭВ» SoRuCom-23. В обзоре представлены основные темы докладов: советская научно-техническая политика в области ВТ, создание соответствующих организаций, элементная база ВТ, создание программных систем и языков программирования, области применения ВТ (с т.ч. космос и оборона), юбилейные даты отрасли, информатика и образование, искусственный интеллект, реконструкция раритетов ВТ докомпьютерной эры. Конференция прошла 25–27 сентября 2023 г. в Нижегородском кампусе НИУ «Высшая школа экономики».

Ключевые слова: *история науки и техники, информатика, вычислительная техника, программирование, искусственный интеллект.*

1. Введение

VI Международная конференция по истории информатики «Развитие вычислительной техники в России, странах бывшего СССР и СЭВ» SoRuCom-23 была организована при поддержке Нижегородского кампуса НИУ «Высшая школа экономики». Она проходила в сложных условиях. Во-первых, сказалась международная изоляция России и нашего научного сообщества, мы не получили технической поддержки со стороны IEEE. Это означает, что у нас не будет рейтинговых публикаций на английском языке. Во-вторых, Российский научный фонд с момента своего образования не поддерживает проведение научных мероприятий. Наши Труды опубликованы онлайн [6].

В-третьих, мы потеряли ряд своих активных участников. В их числе наш бессменный председатель Программного комитета д.ф.-м.н. А.Н Томилин (1933–2021); историк, д.и.н. В.Н. Парамонов (1957–2022); председатель Совета Виртуального компьютерного музея д.ф-м.н. Я.А. Хетагуров (1926–2021), участники наших конференций, ветераны компьютерной

отрасли к.т.н. Ю.В. Рогачев (1925–2021) и к.т.н. В.Ф. Гусев (1940–2021), ветеран компьютерного машиностроения д.т.н. М.В. Тяпкин (1927–2021), ветеран компьютерной индустрии к.т.н. Т.М. Александриди (1924–2020), заведующий редакцией «Техника» Большой Российской энциклопедии к.т.н. С.Б. Оганджанян (1952–2020), специалист в области кибернетики и информатики д.т.н. М.Б. Игнатьев (1932–2019). Это горькие, невосполнимые потери.

Тем не менее, мы сохранили международный статус конференции, получили более 60 заявок на участие. 51 доклад был отобран для представления в смешанном формате. Выступили три приглашенных докладчика: к.ф.-м.н. И.Р. Агамирзян (директор Школы инноватики и предпринимательства НИУ ВШЭ) осветил тему «Развитие информационных технологий в России: 30 постсоветских лет»; член-корр. РАН И.Б. Петров (научный руководитель кафедры вычислительной физики МФТИ), который представил «Компьютерное моделирование динамических процессов в неоднородных сплошных средах: история, задачи, проблемы» и д.т.н. В.В. Кореньков (директор Лаборатории информационных технологий Объединенного института ядерных исследований) – «Исторические этапы, статус и перспективы развития компьютерной инфраструктуры ЛИТ ОИЯИ». В конференции приняли участие более 60 человек, расширилось представительство России (Москва, Санкт-Петербург, Новосибирск, Казань, Пермь, Екатеринбург, Красноярск, Саров, Самара, Лениногорск, Боровск, Тверь, Йошкар-Ола, Королев) и зарубежья (Ереван, Минск, Саннивейл и Санта-Клара, Калифорния, США). Доклады представили член-корреспондент РАН, 16 докторов наук и 28 кандидатов наук. Профильные специалисты по-прежнему в большинстве, а среди участников – сотрудники академических, отраслевых институтов, вузов и музеев, ветераны отрасли и независимые исследователи.

2. Программа SoRuCom-23

В этот раз на конференции не было тематических секций, поскольку нам хотелось, чтобы наши специалисты могли познакомиться с исследованиями коллег в более широком спектре. Помимо традиционных направлений, было заявлено ряд новых тем по истории счетных приборов и цифровой вычислительной техники, ее программного обеспечения и областей применения, подготовки кадров, были представлены вновь открытые исторические документы и забытые имена, отмечены юбилейные даты и многое другое.

2.1. Научно-техническая политика СССР в области вычислительной техники

Научно-техническая политика СССР в области вычислительной техники отражена в нескольких выступлениях. **В.В. Тихонов** (*Институт российской истории РАН, Архив РАН)* представил доклад «Развитие электронной вычислительной техники в СССР и ведущих капиталистических странах в 1960–70-е гг.: взгляд из ЦК КПСС». В докладе акцентировано внимание на догоняющем характере советской отрасли ВТ в указанный период. Автор приводит убедительные архивные свидетельства технологического отставания СССР и попыток его преодоления путем унификации архитектуры советских ЭВМ на базе американских серийных компьютеров IBM и DEC. Это решение было принято ГКНТ уже в 1966 г. при поддержке министра МРП В.Д. Калмыкова и президента АН СССР М.В. Келдыша. Автор утверждает, что именно глобальное противостояние СССР и стран Запада стало основным фактором развития электронно-вычислительной техники в Советском Союзе. На протяжении 1960–70-х гг. можно было наблюдать, что в СССР происходило постепенное наращивание ресурсов, направленных на развитие данной отрасли, но в аппарате ЦК понимали, что отставание в области электронно-вычислительной техники по сравнению с ведущими капиталистическими странами будет только нарастать – даже если удастся выполнить плановые показатели.

В этом контексте показателен доклад **М.Б. Кузьминского** (*ИОХ им. Н.Д. Зелинского РАН*) «О нескольких поколениях больших ЭВМ в СССР и РФ в конце прошлого века: с точки зрения пользователей». Он проанализировал большие ЭВМ, разрабатываемые и производимые с 1970-х по 1990-е годы в СССР и странах СЭВ, в основном в период активного применения БЭСМ-6 и ЕС ЭВМ. Из изложенного он сделал вывод, что в рассмотренный временной период в СССР было необходимо развивать оба направления – ЕС ЭВМ и условной «линии БЭСМ». Но оба оказались закрытыми, и стала применяться, в основном, зарубежная вычислительная техника. Судя по докладу **В.Н. Захарова** (*ФИЦ «Информатика и управление» РАН*) «Развитие отрасли массовой вычислительной техники в СССР в 1980–1990 гг. в аспекте деятельности МНТК “Персональные ЭВМ”» практика копирования прочно вошла в арсенал производства ЭВМ. Докладчик привёл конкретные данные, основанные на анализе ежегодных докладов МНТК о создании и производстве в стране ПЭВМ и систем на их основе. Анализ развития ПЭВМ в стране к 1988 г. показал, что каких-либо принципиальных улучшений ни в разработке ПЭВМ, ни в их производстве не

произошло. Одной из основных причин, изначально определяющих технический уровень и качество советских ПЭВМ, являлось отсутствие элементной базы, отвечающей современным требованиям. В итоге, заключил докладчик, «распад СССР привел к тому, что основные предприятия-производители средств вычислительной техники и ПЭВМ в том числе, оказались вне России. Да и внутри нее были нарушены традиционные научные и производственные связи». Нужно отметить, что политика копирования ЭВМ в данном случае не является чем-то отличным от ситуации в других отраслях техники народного хозяйства СССР [8].

Предприятия отрасли производства ВТ в СССР в основном были сосредоточены в РСФСР, Украине, Беларуси и Армении. Были они и в странах Балтии, о чем доклад **Э.М. Пройдакова** (*Виртуальный компьютерный музей*) «К истории вычислительной техники в странах Балтии». Даны краткие сведения об основных предприятиях и институтах стран Балтии, оказавших заметное влияние на развитие вычислительной техники в СССР. С НПО «ВЭФ» (Рижский ордена Ленина государственный электротехнический завод ВЭФ имени В.И. Ленина) связан один из крупнейших в СССР секретных проектов – создание Единой системы средств коммуникационной техники (ЕС СКТ), который выполнялся в рамках Совета экономической взаимопомощи (СЭВ) и о котором известно немного. Автор использовал личные впечатления и воспоминания, приобретенные во время многочисленных командировок в Латвию и Литву. Э.М. Пройдаков (правда, без аргументов) утверждал, что вычислительная техника в странах Балтии в советский период бурно развивалась и оказала значительное влияние на развитие ВТ в СССР, вопреки мнению, что их влияние на неё было минимальным.

2.2. Организации развития

Формирование профильных институций, которые были призваны обеспечить научно-техническое развитие страны в области вычислительной техники и программирования – важное направление исторических исследований. **И.А. Крайнева** (*ИСИ им. А.П. Ершова СО РАН*) в докладе «К истории ИТМиВТ АН СССР: Лаврентьев vs Бруевич (1948–1953)» остановилась на начальном периоде работы флагмана отечественного компьютеростроения – ИТМиВТ АН СССР. Она раскрыла причины смены его руководства в 1950 г., когда директором стал академик М.А. Лаврентьев. В Трудах конференции опубликованы два документа, которые доказывают влияние кампании по борьбе с космополитизмом (1947–1953) на формирование коллектива и на деятельность ИТМиВТ АН СССР. **Г.И. Минеев**

(*Общество актуальной истории Перми*) в докладе «НИИУМС: причины возникновения и становление основных направлений деятельности» рассказал о создании в начале 1960-х гг. Научно-исследовательского института управляющих машин и систем в городе Перми. Как следует из доклада, история данного НИИ требует дальнейшего исследования, основные блоки которого: развитие выбранных направлений работы и реализация конкретных проектов института (автоматизированные системы управления предприятием, интегрированные АСУ, АСУ технологическими процессами, создание информационно-поисковых систем). Отдельного внимания заслуживает вопрос о разработке сотрудниками института системы предварительного расчёта определения эффективности внедрения автоматизированных систем управления на предприятии.

Н.А. Куперштох (*Институт истории СО РАН)* в докладе «Институт информационных технологий и прикладной математики: организационные коллизии 1990-х гг.» подробно рассмотрела перипетии создания нового НИИ в Омске в составе Сибирского отделения АН СССР. Организация ИИТПМ осуществлялась в рамках стратегии академика Г.И. Марчука по созданию НИИ математического профиля, а также вычислительных центров в крупных промышленных городах Сибири. В 1990-е гг. новый институт, как и ряд других в СО РАН, был включен в состав более крупных и успешных институтов, в данном случае в состав Института математики СО РАН. Автор назвала несколько причин утраты самостоятельности ИИТПМ, в том числе отсутствие на периферии академических традиций и научных школ, рассогласованность в действиях по развитию научного потенциала в Омской области на областном, республиканском, союзном уровнях, а также и ряд других. Она справедливо полагает, что вопрос требует дополнительного изучения. На наш взгляд, крайне важное значение приобретает изучение проблемы актуальности и востребованности институций данного профиля в Омском регионе (помимо амбиций партийных чиновников), а также кадровой и тематической структур академических НИИ в целом.

2.3. Элементная база ЭВМ

Проблемы развития элементной базы ЭВМ в СССР всегда актуальны. Благодаря исследованию историка науки и техники **Р.Н. Парамоновой** (*Самарский национальный исследовательский университет имени академика С.П. Королёва*), которая представила доклад «Электронное машиностроение в СССР в 1965–1985 гг.: планы и результаты развития отрасли», мы в широком историческом контексте познакомились с подходами к ее созданию. Отправной момент в развитии отрасли она связала с историей Госприемки ЭВМ БЭСМ,

которая при дефиците комплектующей элементной базы (на ртутных линиях задержки в 1952 г.) не смогла дать запланированное быстродействие, и лишь в 1955 г., оснащенная потенциалоскопами, показала свое превосходство над серийной «Стрелой». Затем появились полупроводники и полупроводниковые ЭВМ, в т.ч. для военного применения. Важность элементной базы для ЭВМ была осознана на государственном уровне, что привело к образованию Госкомитета по электронной технике в 1961 г. К середине 1960-х гг., констатирует автор, электроника (микроэлектроника – *И.К, В.Ш.*) стала в СССР отраслью промышленности, которая включала в себя весь спектр работ, от научных изысканий и проектирования электронных компонентов и изделий из них до запуска опытного и серийного производства. Доклад Р.Н. Парамоновой основан на широком историографическом и архивном материале, содержит много новых фактических и статистических данных по исследованному ею предмету.

Б.М. Малашевич специалист по микроэлектронике, ныне известный исследователь истории этой области, автор статей и монографий, независимый исследователь, представил три доклада, уделив внимание некоторым спорным моментам в истории микроэлементной базы в СССР. В докладе «Две ошибки на заре микроэлектроники» автор проанализировал и оспорил тезис о 15-летнем отставании отечественной микроэлектроники. Этот тезис был выдвинут в аналитической записке 1965 г. «Сравнение достижений микроэлектроники в СССР и за рубежом», обнаруженной Малашевичем в архиве первого директора и основателя зеленоградского Центра микроэлектроники (ЦМ) Ф.В. Лукина (1908–1971). Автор назвал причины ошибки (или, в терминологии докладчика, «дезинформации») низким уровнем экспертизы – информации было крайне мало, готовых специалистов в стране не было. Логично предположить, что в понятие «микроэлектроника» составители записки вложили представление о полупроводниковых приборах как таковых. В исторической литературе можно встретить периодизацию процесса развития элементно-конструктивной базы электронных приборов: 1940-е гг. – электровакуумные приборы, 1950-е гг. – полупроводниковые приборы, 1960-е гг. – микроэлектроника (интегральные схемы) [4, с. 4]. Известно, что работы по ИС в СССР были начаты в 1958 г. (НИИ а/я 233, Е. Ляхович, Л. Реймеров), практически одновременно с США (Джек Килби, 1958).

В докладе «Начала микроэлектроники» Малашевич отметил, что серийное производство микроэлектроники СССР и США начали одновременно, в 1962 г. Он привел сравнительный анализ работ создателей интегральных схем в СССР и США. В заключение этого доклада Б.М. Малашевич констатировал, но не развернул другой аспект проблемы: отставание в объёмах производства. В докладе «Старт отечественно микроэлектроники» в

научный оборот введены черновики и конспекты директивных документов, составленных Ф.В. Лукиным, рассмотрен процесс создания этого инновационного центра.

2.4. Программные системы и языки программирования

Проблематика программирования представлена несколькими докладами. Два доклада сделали разработчики операционной системы Диспак: **Н.Е. Балакирев** (*НИУ МАИ*), **Ю.Г. Бартенев** (*Институт теоретической и математической физики РФЯЦ ВНИИЭФ*), **С.А. Зельдинова** (*независимый исследователь*) «О самой распространенной операционной системе ДИСПАК и других ОС на машинах БЭСМ-6» и **А.И. Немецков** (*АО «ПФ «НТЦ «Атлас»*), **Н.Е. Балакирев** (*НИУ МАИ*), **С.А. Зельдинова** «ОС ДИСПАК в разработке космического комплекса “МИР”». Следует сказать, что столь развернутого освещения создания ОС Диспак на нашей конференции еще не проводилось, хотя другие операционные системы были представлены [5]. Несомненно, этот факт заслуживает положительной оценки: то, что эта ОС из Советского атомного проекта вышла в свободное использование и развитие – редкое явление в истории нашей науки и техники. Первая публикация, посвященная истории создания ОС Диспак, появилась на сайте Виртуального компьютерного музея в 2016 г. [7].

Доклад «ОС ДИСПАК в разработке космического комплекса “МИР”» посвящен использованию операционной системы ДИСПАК для машин БЭСМ-6 при создании космической станции «Мир», одной из важнейших научно-производственных задач космических программ СССР. Один из авторов был непосредственным участником создания пилотируемой космической станции «Мир» (ПКС «Мир»). Коллективом разработчиков были созданы системы разработки, автономной и комплексной отладки штатного (бортового управляющего) программного обеспечения ПКС «Мир» на базе бортовой цифровой вычислительной машины (БЦВМ) «Салют-51», комплексной отработки функционирования аппаратно-программных средств в процессе полного цикла эксплуатации станции, системы принятия решений при возникновения нештатных ситуаций в процессе эксплуатации космического комплекса и другие подсистемы. Стоит отметить, что открытые публикации и отчеты по данному вопросу отсутствуют. С учетом срока давности авторы раскрыли некоторые детали этих разработок и отметили значимость и важность использования многоплановой функциональности ОС ДИСПАК в процессе разработки аппаратно-программных средств ПКС «Мир».

В развитие темы можно отметить, что на конференции прозвучали еще два доклада, посвященных космической программе СССР: **С.Я. Нагибин** *(НИУ МАИ),* **В.Г. Ровенко, В.В. Ясюкевич** *(НИЦ ЦНИИ ВКС, Королёв)* «Об истории автоматизации баллистико-навигационного обеспечения космических программ в СССР и Российской Федерации» и **С.Я. Нагибин, Н.А. Тихомиров, В.В. Ясюкевич** (*Межрегиональная общественная организация ветеранов космодрома Байконур*) «Вычислительная техника при испытаниях космических средств на космодроме Байконур». В первом докладе авторы – участники событий – повествуют о создании и развитии баллистико-навигационного обеспечения (БНО) в контексте космических программ и развития вычислительной техники. Они также проследили путь модернизации специального программного комплекса на машинах БЭСМ-6, АС-6, ПМ-6, дальнейшего развития программного обеспечения БНО на вычислительном комплексе ВС1-К2, а затем на «Эльбрусе 1-КБ». Во втором докладе освещен малоизвестный широкой общественности опыт применения вычислительной техники при испытаниях космических средств, при подготовке и пусках. Авторы представили примеры применения вычислительной техники при обработке траекторной и телеметрической информации в 3-м Научно-испытательном управлении (3 НИУ, с 1989 г. – в Центре испытаний и применения космических средств, 4 ЦИП КС), а также при испытаниях ракетно-космической техники на технических и стартовых комплексах 4-го Научно-испытательного управления космодрома Байконур (с 1989 г. 1275-й ЦИП КС).

Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования (включая экзотические). В Советском Союзе практика разработки алгоритмических языков также имела место, свою историю и применение, свои научные школы. **В.А. Китов** (*Российский экономический университет им. Г.В. Плеханова*) в своей работе «Страницы истории отечественных языков программирования» рассказал о трёх созданных в СССР алгоритмических языках. Это алгоритмический язык программирования АЛГЭМ, созданный А.И. Китовым в середине 1960-х годов и массово использовавшийся при программировании задач экономики и управления на ЭВМ серии «Минск». В те же годы в киевском Институте кибернетики АН УССР для ЭВМ инженерных расчетов серии «МИР» А.А. Летичевским, Ю.В. Благовещенским и А.А. Дородницыной был разработан язык АНАЛИТИК, близкий к алгоритмическим языкам высокого уровня. В середине 1970-х годов для ЭВМ Единой Серии А.И. Китовым был разработан алгоритмический язык программирования НОРМИН, использовавшийся при создании прикладных программ в области медицины и здравоохранения, а также в других областях.

Ю.С. Владимирова (*ВМК МГУ им. М. В. Ломоносова*) представила доклад «Диалоговая система структурированного программирования». ДССП была создана в начале 1980-х годов в МГУ под руководством главного конструктора троичных машин «Сетунь» и «Сетунь-70» Н.П. Брусенцова. ДССП разрабатывалась как средство программирования мини- и микрокомпьютеров, в основе были идеи, примененные в машинном языке «Сетуни-70». Существовало несколько версий ДССП, в том числе поддерживающие троичную арифметику и логику. Несмотря на все достоинства ДССП, после 1990-х годов ее развитие и использование практически прекратилось.

Г.А. Егоров (*ИНЭУМ им. И.С. Брука*) в своем докладе «Системное программное обеспечение СМ ЭВМ» отметил, что с середины 1970-х годов (идейно на 10 лет раньше) две международные системы, СМ ЭВМ и ЕС ЭВМ, в совокупности, дополняя друг друга, стали технической базой автоматизации управления и обработки информации во всех сферах народного хозяйства нашей страны. С 1974 по 1990 годы по разработкам ИНЭУМ было выпущено более 60 тысяч вычислительных и управляющих комплексов, а также измерительно-вычислительных комплексов (ИВК) и автоматизированных рабочих мест (АРМ) на базе СМ ЭВМ. Мы получили обширную справочную информацию, еще один пример разработок в стиле политики копирования, но не увидели, в чем состоял оригинальный вклад разработчиков в создание данных систем.

Л.А. Брухис (*Synopsys Inc, Саннивейл, Калифорния, США*) рассказал об «Опыте восстановления функциональности архивной системы МАРС для БЭСМ-6 и работы по её дизассемблированию». Не найдя объемных публикаций о разработке архивной системы Марс-6 для БЭСМ-6, представляющей собой интерпретируемый «микрокод» основных и вспомогательных операций обращения к базам данных, автор попытался восстановить функциональность этой системы в режиме эмуляции, базируясь на имеющейся документации и двоичных образах дисков.

2.5. Области применения ЭВМ

Обширны области применения ВТ в Советском Союзе и в РФ: от автоматизации различных процессов, включая автоматизацию проектирования ЭВМ, до решения сложных задач управления экономикой и обороной. **М.В. Тумбинская, В.М. Трегубов** (*Казанский национальный исследовательский технический университет им. А.Н. Туполева*), **О.В. Денисов, А.В. Чирикин** (*Лениногорский филиал Казанского национального исследовательского технического университета им. А.Н. Туполева*) в докладе

«Цифровизация нефтяной компании «Татнефть»: от автоматизации ручных расчетов до технологий искусственного интеллекта» привели хронологию развития средств вычислительной техники и автоматизированных систем управления, применения средств автоматизации и специального программного обеспечения в производстве нефтяной компании «Татнефть». Доклад «Два полюса технической диагностики», авторы **Б.М. Басок** (*независимый исследователь*), **С.Л. Френкель** (*ФИЦ «Информатика и управление» РАН*), посвящен научным биографиям двух выдающихся ученых, стоявших у истоков отечественной технической диагностики, П.П. Пархоменко и Д.М. Гробмана. Их имена заслуженно стоят в одном ряду с именами ведущихся специалистов в области разработки вычислительной техники и компьютерных технологий. Идеи и подходы П.П. Пархоменко и Д.М. Гробмана находят свое применение при оценке качества современных информационных систем.

В.А. Лущекин (МГУ *им. М.В. Ломоносова*) в докладе «Автоматизация проектирования – школа Н.Я. Матюхина (взгляд пользователя)» отметил социальные проблемы САПР: автоматизация проектирования порой не находила поддержки у конструкторов из-за боязни конкуренции и сокращения штатов, отсутствия знаний и кругозора. Видимо, больше «повезло» автоматизации проектирования систем программирования, реализующих языки описания и дискретного моделирования цифровой аппаратуры. Они стали важной частью САПР электронной аппаратуры и больших интегральных схем, обеспечивая процесс анализа проектных решений. Об этом доклад **А.К. Полякова** и **И.И. Ладыгина** (*НИУ МЭИ*) «История развития отечественных систем дискретного моделирования цифровой аппаратуры».

В ряде докладов нашел отражение широкий спектр военных применений ЭВМ. **Г.А. Арутюнян** (*ЕрНИИММ, Ереван*) представил доклады «Разработка Ереванским НИИ математических машин специализированного двухмашинного вычислительного комплекса СВК и операционной системы реального масштаба времени» 1970-х гг. и «Разработка Ереванским НИИ математических машин многопроцессорной вычислительной системы «Севан» и операционной системы реального масштаба времени» 1980-х. Это были системы для АСУ Вооруженных сил. По мнению автора, их архитектура и системные возможности превосходили отечественные и зарубежные системы аналогичного класса и должны занять достойное место в истории развития вычислительной техники и программного обеспечения Советского Союза.

А.Б. Барский (*НИИЦ ЦНИИ ВКС Минобороны России*) и **Ю.В. Ревич** (*независимый исследователь*) в докладе «Вычислительная техника и программирование в ЦНИИ 45

Министерства обороны (1960–1990 гг.)» доложили об истории применения высокопроизводительных вычислительных средств для систем воздушно-космической обороны (ВКО). В течение советских лет ЦНИИ 45 МО РФ был головным институтом по научным аспектам темы ВКО: в нем разрабатывались методические основы и программные средства применения ЭВМ. Многие самые яркие отечественные разработки в области супер-ЭВМ осуществлялись либо по прямому заказу ЦНИИ 45, либо с учетом требований, выдвинутых его специалистами. Значительный вклад специалисты ЦНИИ 45 внесли в теорию и практику программирования высокопроизводительных вычислительных комплексов и систем реального времени. Другая разработка освещена **Л.Е. Карповым** (*ИСП РАН им. В.П. Иванникова, МГУ*) в докладе «Базовые типы данных управляющих ЭВМ серии 5Э26 и современные языки программирования» второй половины 1960-х гг.» Разработка предназначалась для применения в составе систем противовоздушной обороны С-300 различных конфигураций. На этих машинах также работали несколько систем программирования, выполнялась трансляция программ с различных языков программирования, велась подготовка текстовой документации. **И.Ф. Богданова** (*Белорусская сельскохозяйственная библиотека им. И.С. Лупиновича НАН Беларуси*) и **Н.Ф. Богданова** (*независимый исследователь*) посвятили свой доклад «Из истории белорусских специализированных ЭВМ военного назначения» ряду стационарных и мобильных ЭВМ, разработанных российскими и белорусскими конструкторами и серийно выпускавшимися Минским производственным объединением вычислительной техники и Брестским электромеханическим заводом в 1964–1995 гг. (ЭВМ «Весна», «Снег», возимые ЭВМ РВ-2 и РВ-3, первая в СССР серийно выпускаемая защищенная ПЭВМ ЕС-1845 и др.).

В.И. Штейнберг и **В.А. Шпиев** (*АО «НИИ «Аргон»*) представили историю разработок бортовых цифровых вычислительных комплексов, проводимых в НИИ «Аргон» для воздушных командных пунктов стратегического управления вооруженными силами страны в докладе «Разработка средств бортовой вычислительной техники для воздушных пунктов стратегического управления». **С.А. Инютин** (*НИУ МАИ*) в докладе «Развитие вычислительных методов для многомерных математических объектов в АН КазССР» рассказал, что перенесение научных исследований в области модулярной арифметики (МА) из Зеленограда в АН Казахстана в конце 1970-х гг. дало мощный импульс развитию параллельных вычислительных методов с использованием целочисленной МА комплексных чисел и приложениям ее в системах оперативного управления движением летательных аппаратов. Проблема исследовалась группой специалистов под руководством академика АН КазССР В.М. Амербаева. Работы в Казахстане были тесно связаны с Зеленоградом: ЭВМ

К340А для ПРО Акушского-Юдицкого была основана на МА. Работы в Алма-Ате были направлены на дальнейшее расширение ее использования.

2.6. Юбилейные даты

Традиционно в нескольких докладах были отмечены юбилейные даты информатики. **В.Н. Захаров** (*ФИЦ «Информатика и управление» РАН*) напомнил слушателям о 75-летии получения первого в стране патента на изобретение автоматической цифровой вычислительной машины И.С. Бруком и Б.И. Рамеевым, о 40-летии образования Отделения информатики, вычислительной техники и автоматизации АН СССР (с декабря 2007 г. – Отделение нанотехнологий и информационных технологий, ОНИТ РАН), о 40-летии создания Института проблем информатики АН СССР, о 100-летии выдающихся ученых, сыгравших важную роль в отечественной информатике – М.А. Карцева (1923–1983) и В.М. Глушкова (1923–1982), 60-летии создания СУНЦ МГУ (школы им. А.Н. Колмогорова), 120-летии самого академика А.Н. Колмогорова (1903–1987). Развернутый доклад, посвященный 100-летию со дня рождения академика В.М. Глушкова сделали **О.В. Китова** и **В.А. Китов** (*РЭУ им. Г.В. Плеханова*). Авторы констатировали, что «В.М. Глушков как мыслитель отличался широтой и глубиной научного видения, своими работами он предвосхитил то, что только сейчас появляется в современном информационном обществе. Его имя заслуженно стоит в одном ряду с именами ведущихся мировых учёных в области разработки компьютерной техники и информационных технологий. Он воспитал целую плеяду учеников и последователей, которые с успехом продолжили его дело. Многие идеи В.М. Глушкова еще ждут своей реализации».

Нечто новое об академике В.М. Глушкове мы узнали из доклада **Н.Ю. Пивоварова** (*Институт всеобщей истории РАН, НИЯУ МИФИ*) «Электронный мозг партии: создание и первый этап функционирования информационно-вычислительного центра ЦК КПСС (1970–1972 гг.)». Автор с привлечением ранее неизвестных архивных данных исследовал историю проекта «Полюс» по информатизации ЦК КПСС, главного центра принятия решений в СССР. В.М. Глушков, выступив рецензентом проекта, предложил создать не просто некий банк данных о партийных организациях, а универсальную систему, способную анализировать информацию, показывая основные тенденции развития в экономике и политике, в том числе прогноз кризисных ситуаций в любом заданном районе земного шара. Главным препятствием в создании такой системы стал тезаурус, понятийный словарь, необходимый для описания и прогнозирования ситуаций в политике и экономике, в

том числе международного уровня. Сложность реализации привела к остановке очередного грандиозного проекта академика В.М. Глушкова. **Ю.Е. Поляк** (*ЦЭМИ РАН*) посвятил свой доклад «Идеи академика В.М. Глушкова и современный электронный документооборот» последней монографии ученого «Основы безбумажной информатики», которая увидела свет в 1982 г. через несколько месяцев после кончины автора (в 1987 г. вышло 2-е издание). В.М. Глушков считал, что «к началу следующего столетия в технически развитых странах основная масса информации будет храниться в безбумажном виде: в памяти ЭВМ». Ю.Е. Поляк, в целом высоко оценив прогностические идеи академика Глушкова, тем не менее отметил, что в обиходе остаются и телевизоры, и печатные издания. Точно так же электронный документооборот не отменяет бумажный, который порой становится даже еще более объемным, чем прежде: мы убедимся в этом, заглянув в плановый отдел или бухгалтерию любой организации.

Деятельность еще одного корифея информатики – академика А.П. Ершова (1931–1988) – была освещена в докладах **Л.В. Городней** (*ИСИ им. А.П. Ершова СО РАН*) «Ершовские научные конференции по программированию» и **Г.В. Курляндчик** в соавторстве с **Н.А. Черемных** (*независимые исследователи из США и Москвы*) «История создания и автоматизации Мемориальной библиотеки академика Андрея Петровича Ершова». Л.В. Городняя продолжила традицию освещения ершовских конференций, которая была заложена ранее на SoRuCom-20 [3]. В своем докладе она показала разнообразие тематики, участие в конференциях лучших представителей отечественной программистской науки и авторитетных иностранных учёных, оценила доброжелательную атмосферу, допускавшую импровизированные дискуссии вне регламента и многое другое, что определяло исключительно плодотворный климат научного общения. Г.В. Курляндчик и Н.А. Черемных не впервые обращаются к истории создания и работы Мемориальной библиотеки академика А.П. Ершова в ИСИ СО РАН, поскольку обе являлись активными участницами организации научно-справочного аппарата библиотеки, его информатизации, формирования фонда и его мемориализации. Авторы отметили, что библиотека продолжает развиваться, ныне реализованы современные приемы ее автоматизации. В 2018 году сотрудник ИСИ С. Трошков перенес виртуальный фонд на платформу Drupal, библиотека стала веб-приложением (http://lib.iis.nsk.su/).

2.7. Информатика и образование

Одно из тематических направлений нашей конференции – информатика образования, подготовка кадров в области информатики на всех уровнях образовательной лестницы. В докладе **А.Г. Гейна** и **Н.А. Юнерман (Гейн)** (*Уральский федеральный университет имени первого Президента России Б.Н. Ельцина*) «От первых компьютеров в школе к всепоглощающей цифровизации образования» рассмотрена эволюция подходов и смена парадигм в преподавании информатики в школе. Первым направлением в этой области, как известно, стала обоснованная академиком А.П. Ершовым необходимость развития у школьников алгоритмического стиля мышления, что было возможно в отсутствии персональных компьютеров или при обучении на базе компьютерных классов. Этот период сопровождался работой над созданием обучающих прикладных программ по разным школьным предметам. Переход в конце 1990-х – начале 2000-х годов к использованию ПК в школе привело к эволюции обучающих программ в двух направлениях: к созданию компьютерных курсов, полностью обеспечивающих преподавание того или иного учебного предмета, и созданию компьютерных обучающих сред. Появление интернета, по мнению докладчиков, разнообразило педагогические технологии, хотя и не облегчило жизнь учителю. Онлайн курсы; дистанционное обучение; использование внешних ресурсов – вот неполный перечень приемов информатизации образования. К сожалению, авторы оставили за пределами своего доклада объяснение, что представляет собой заявленный в заголовке тезис «всепоглощающая цифровизация образования». Является ли она данностью или перспективой?

Можно было ожидать, что ответ на часть вопросов даст доклад **И.А. Чудакина, Е.А. Халтурина** (*Сибирский федеральный университет*) и **С.А. Виденина** (*НИУ «ВШЭ»*) «Историческое развитие концепции применения видеоигр для геймификации образования». Авторы установили, что разработано множество подходов к геймификации, которые всё больше находят своё применение в приложениях различной сложности и различной направленности. При этом лишь небольшая часть из широкого разнообразия существующих приемов геймификации используется в обучающих приложениях. В работе представлена ретроспектива научных исследований видеоигр, как феномена. Заявленный анализ того, почему только некоторые подходы к геймификации нашли своё приложение в процессах обучения, обоснован лишь одним тезисом: «Нежелательным сценарием является подмена непрерывного учения с элементами игры непрерывной игрой с элементами учения». Все остальное – историография, которая, несомненно, будет полезна интересующимся данным вопросом.

Основательно подошли к своему исследованию **В.В. Буров** и **Е.Д. Патаракин** (*НИУ «ВШЭ»*), которые представили доклад «Путь Черепахи: эволюция Logo-подобных языков». В их понимании Logo – это название философии образования (конструкционализм) и, одновременно, постоянно развивающегося семейства языков программирования, которое помогает в ее реализации, решает задачи наглядного многоагентного моделирования для различных областей, в том числе научных исследований. В докладе рассмотрено развитие многочисленных потомков языка программирования высокого уровня Logo, изначально созданного в 1967 г. У. Фёрзегом, С. Пейпертом и С. Соломон для обучения школьников математике и алгоритмам. Анализ включает и отечественные разработки. Кроме разнообразия направлений, в которых эволюционировал Logo, внимание в докладе уделено возникновению и развитию российского сообщества, связанного с этим языком. Logo оказал влияние на российское образование, стал тем инструментом, который позволил обучаться программированию с раннего возраста. Говоря высоким стилем, в конце двадцатого века Черепашка Logo была интересным граничным объектом, который объединил людей из разных стран и разных профессиональных сообществ.

Информатизация образования, или процесс использования компьютеров в школе, появление «школьной информатики», явились результатом совместных и отдельных усилий множества специалистов. Одним из таких энтузиастов был профессор М.Б. Игнатьев (1932–2019), чью деятельность осветил в своем докладе «Петровские традиции в образовании: из истории ленинградской конференции “Школьная информатика”» **М.А. Вус** (*Санкт-Петербургское Общество научно-технических знаний*). Проводившаяся на протяжении без малого четырёх десятилетий под руководством профессора М.Б. Игнатьева ленинградская (впоследствии санкт-петербургская) конференция «Школьная информатика» сыграла большую роль в распространении знаний по информатике, стимулировала развитие новых технологий обучения.

Тему развития образования, формирования системы подготовки специалистов в области математического обеспечения ЭВМ в вузах продолжили **Б.К. Мартыненко** (*Санкт-Петербург, независимый исследователь*) в докладе «Кафедра математического обеспечения/информатики Ленинградского – Санкт-Петербургского университета в эпоху С.С. Лаврова. К 100-летию Святослава Сергеевича Лаврова» и **О.В. Марасанова** (*Пермский государственный национальный исследовательский университет*) в докладе «Подготовка кадров в области автоматизированных систем управления в Перми: история «отцов-основателей» (1950–1970-е гг.)». Б.К Мартыненко поделился воспоминаниями из истории создания кафедры МО в Ленинградском университете, формирования подходов к

преподаванию программирования на математико-механическом факультете ЛГУ, о профильных конференциях, рассказал несколько эпизодов из своих встреч и общения с ее руководителем С.С. Лавровым (1923–2004). О.В. Марасанова исследовала период 1950–70-х годов, которые, по ее утверждению, в истории Перми являлись временем бурного индустриального роста. В пермских вузах – университете и пединституте – были открыты кафедры прикладной математики, автоматики и телемеханики, экономической кибернетики. В первой части доклада она привела биографические сведения о создателях данных направлений: о Ю.В. Девингтале (1924–1996), М.С. Тер-Мхитарове (1924–2007) и И.А. Кручинине (1931–2005). Во второй части автор сравнила условия, при которых в пермских ВУЗах началась подготовка по трем образовательным программам. Сравнительный анализ биографий и опыта организации новых специальностей позволил понять, какие агенты действовали в инновационной сфере автоматизации во второй половине XX века в СССР и какие интересы и ценности они отстаивали.

Доклад **В.А. Биллига** (*Тверской государственный технический университет*) «Мой путь в программировании длиною в жизнь» – своего рода представление обратной связи между образованием и судьбой в профессии. История В.А. Биллига – один из примеров освоения тогда еще достаточно новой профессии. В 1960 г. автор окончил физико-математический факультет Днепропетровского государственного университета и по распределению был принят на должность ведущего инженера в ВЦ НИИ МО в городе Калинине (Тверь). Он самостоятельно освоил приемы программирования на первых ЭВМ – «Урал-1», М-20, М-220, БЭСМ, защитил кандидатскую диссертацию по физико-математическим наукам. В начале 1970-х перешел на преподавательскую работу, и обучение студентов стало основным делом его жизни. Воспоминания Биллига являются прекрасным источником информации для исследования социальной истории науки и техники, поскольку содержат массу фактов из его повседневной жизни – школьника, студента, специалиста, закрытого КБ, преподавателя университета [2].

2.8. Из истории искусственного интеллекта

Модная ныне проблематика искусственного интеллекта, к сожалению, редка на нашей конференции. Возможно, это связано с тем, что сообщество ИИ давно осознало себя отдельной (закрытой?) общностью, но не пришло еще к историческому осмыслению своей деятельности. Тем не менее мы заслушали доклад **В.П. Ильина** (*Институт вычислительной математики и математической геофизики СО РАН*) «Становление и развитие

искусственного интеллекта в СО РАН». Он остановился на той роли, которую играет искусственный интеллект для обеспечения эффективности современных компьютерных систем. Автор проследил историю зарождения и развития проблематики искусственного интеллекта в ВЦ СО АН СССР под руководством таких корифеев информатики и математики, как академики А.П. Ершов, Г.И. Марчук, С.К. Годунов и Н.Н. Яненко.

2.9. Счетные устройства докомпьютерной эры

Ряд докладов был посвящен раритетным счетным устройствам. Представители музеев рассказали о своих коллекциях. **М.Э. Смолевицкая** (*Политехнический музей*) в своем докладе «Коллекция логарифмических приборов в Политехническом музее» привела результаты научно-исторического исследования собрания Политехнического музея. Она дала краткую историческую характеристику логарифмических приборов до 1917 года, историю их производства в СССР. Представлены краткие исторические описания некоторых особо значимых экспонатов, таких как цилиндрические линейки А.Н. Щукарёва и М.Е. Подтягина, специальные логарифмические линейки В.Ф. Пояркова, П.В. Михеева и др. Основатель *Музея компьютеров в городе Боровске* **В.Ю. Архипов** в докладе «К истории программируемых калькуляторов СССР» представил коллекцию музея, а также затронул некоторые вопросы истории развития отечественных программируемых калькуляторов. Особенный интерес представил его рассказ о знаменитом «Клубе электронных игр», организованном в журнале «Техника – молодежи» писателем М. Пуховым. Этот клуб стал важнейшим центром обмена информацией между многочисленными энтузиастами – любителями калькуляторов.

Доклад **Г.А. Базанчук** и **С.В. Куракова** (*МГТУ им. Н.Э. Баумана*) «Применение аналоговых расчетных устройств при организации производства, НОТ и рационализации в первой половине XX века» был посвящен малоизвестной истории применения специальных логарифмических линеек в области организации производства в Российской империи, а затем СССР. Привлечены архивные источники и материалы коллекции математических инструментов музея МГТУ им. Н.Э. Баумана, **А.И. Басов** (*независимый исследователь*) в докладе «Герман Холлерит и Россия» дал обстоятельное, с привлечением широкого круга публикаций российской и американской прессы того времени, исследование использования в Российской империи системы обработки статистической информации, предложенной Холлеритом в конце XIX в. Это позволило сделать важные выводы об уровне экономического, политического и социального развития страны на рубеже XIX–XX веков.

Бруски Иофе, созданные в России около 1880 г., вот уже полтора столетия привлекали пристальное внимание исследователей разных стран. **Д.М. Златопольский** (*Музей истории вычислительной техники школы № 1530 «Школа Ломоносова»*) в докладе «Счётный прибор Иофе – как и почему он работал» впервые дал популярное математическое обоснование его устройства и работы, которое позволило ему осуществить реконструкцию прибора, дать разъяснения по его использованию. История российских механических счетных устройств до сих пор известна лишь в самых общих чертах. **Д.М. Златопольский** и **В.В. Шилов** (*НИУ ВШЭ*) в докладе «Самуил Авраамович Каценелленбоген и его счётные приборы» впервые описали конструкцию двух вычислительных приборов, созданных в России в 1875 г. и 1886 г. минским учителем Самуилом Авраамовичем Каценелленбогеном и предложили свою реконструкцию методов расчетов на них. Доклад содержит краткое изложение биографии этого забытого изобретателя [См. также 2].

3. Заключение

Программный комитет не принимал какого-либо решения после завершения SoRuCom-24, но в кулуарах обсуждалась возможность проведения следующей конференции в 2025-26 гг. в Перми, если Пермский национальный исследовательский университет, например, примет на себя обязанности по ее организации, как это было в Нижнем Новгороде. Отмеченные нами проблемы, связанные с нарастающей международной изоляцией со стороны Европы и США, которые могли бы изменить статус конференции, как показал опыт, решаемы. Финансовые проблемы представляются более значимыми, вряд ли стоит ожидать какой-либо специфической грантовой поддержки, если только в Перми не найдутся спонсоры. В противном случае финансовая нагрузка ляжет на участников, а проведение конференции в смешанном формате нами уже опробовано. На первый план выйдет техническая сторона дела. Что касается тематики, то как видим, «свежая кровь» у нас появилась, идеи не иссякают, и мы надеемся, что в будущем мы сформируем повестку не менее значимую, чем это было всегда.

Все участники конференции благодарят директора к.э.н. А.А. Бляхман, заведующего кафедрой д.ф.-м.н. В.А. Калягина, заместителя заведующего лабораторией к.ф.-м.н. Т.В. Медведева и студентов НИУ «Высшая школа экономики» в Нижнем Новгороде за помощь в подготовке конференции, ее четкую организацию, культурную программу и гостеприимство.

Список литературы

1. Биллиг В.А. Хроника языков программирования // Прошлое, настоящее, будущее». М. : «Интуит», 2024. 286 с.
2. Златопольский Д.М., Шилов В.В. Из истории банковского дела. «Счеты сложных процентов» // Банковское дело. 2024. № 2. С. 72–75.
3. Меньщиков В.Ф., Павловская И.Ю., Черемных Н.А. Вторая всесоюзная конференция по программированию // Труды SoRuCom-20, с. 226-233.
4. Симонов Н.С. Несостоявшаяся информационная революция: условия и тенденции развития в СССР электронной промышленности и средств массовой коммуникации. Ч. I. 1940–1960-е годы. М.: Русский фонд содействия образованию и науке. 2013. 276 с.
5. Томилин А.Н. Четыре поколения операционных систем Виктора Петровича Иванникова. Труды SoRuCom-20. С. 311–313.
6. Труды 6-ой Международной конференции «Развитие вычислительной техники в России, странах бывшего СССР и СЭВ (SORUCOM 2023)». Нижний Новгород, 25–27 сентября 2023 г. // URL: https://www.iis.nsk.su/news/preprints/sorucom-2023
7. Тюрин В.Ф., Зельдинова С.А., Крайнева И.А. Операционная система Диспак https://www.computer-museum.ru/articles/operatsionnye-sistemy/789/ (06.03.2016).
8. Sutton, Antony C. Western Technology and Soviet Economic Development 1945-1968. Hoover Inst. Press, Stanford, CA, 1973. 482 pp.

УДК 519.7

Сравнение эквивалентностей непрерывно-временных сетей Петри относительно стратегий сброса часов

Зубарев А.Ю. (Институт систем информатики СО РАН)

Непрерывно-временные сети Петри (НВСП) — расширение сетей Петри, где каждый переход имеет собственные часы и временной интервал. Данная модель рассматривается в контексте слабой временной стратегии, в которой ход времени не заставляет переходы срабатывать. Для НВСП исследуются эквивалентности в дихотомиях «интерливинг — истинный параллелизм» и «линейное — ветвящееся время». Анализируются взаимосвязи между эквивалентностями относительно промежуточной и устойчиво атомарной стратегий, определяющих порядок сброса часов переходов.

Ключевые слова: непрерывно-временные сети Петри, слабая временная стратегия, промежуточная и устойчиво атомарная стратегии сброса часов, поведенческие эквивалентности, семантика интерливинга, процессно-сетевая семантика, языковая и бисимуляционная эквивалентности, бисимуляционная эквивалентность с сохранением истории

1. Введение

Безопасность функционирования современных информационно-компьютерных систем, ошибки в которых могут привести к большим убыткам или создать угрозу для жизни людей, имеет критическое значение. Эквивалентности помогают сравнивать поведение данных систем с точки зрения различных аспектов их работы, что играет важную роль в спецификации и верификации. При определении эквивалентностей моделей параллельных и распределенных систем принято рассматривать две дихотомии: *«интерливинг — истинный параллелизм»* и *«линейное — ветвящееся время»*. Критерием первой дихотомии является степень учета частичного порядка между действиями системы, второй — степень учета точек недетерминированного выбора альтернативных действий моделируемой системы.

Сети Петри (СП) являются хорошо изученной моделью для анализа параллельных и распределенных систем. События и условия системы в данной модели наглядно представлены в виде *переходов* (барьеров) и *мест* (окружностей), образующих вершины двудоль-

ного направленного графа. Состояние СП определяется *разметкой* — множеством мест с *фишками*, которые обозначаются жирными точками в соответствующих окружностях. *Допустимый переход*, т.е. переход, имеющий фишки в каждом входном месте, может сработать и тем самым изменить разметку. Новая разметка получается в результате удаления фишек из входных мест перехода и добавления фишек в его выходные места.

Для учета непрерывно-временных (количественных) параметров системы были предложены различные временные расширения СП. В литературе представлены два типа подходов: дискретно-временные сети Петри (ДВСП) и непрерывно-временные сети Петри (НВСП). В ДВСП используются глобальные часы, и каждому переходу присваивается целочисленное значение, определяющее продолжительность его срабатывания. В то время как в НВСП каждый переход имеет собственные часы и временной интервал действительных чисел. Известно, что НВСП обладают той же вычислительной мощностью, что и машины Тьюринга, и включают в себя ДВСП, благодаря чему модели НВСП вызывают особый интерес у исследователей. Состояние НВСП описывается разметкой и вектором показаний часов допустимых переходов. Допустимый переход может сработать, если значение его часов принадлежит связанному с ним интервалу. Срабатывание перехода не занимает время, поэтому рассматривают два способа изменения состояний: *ход времени* (увеличение значений часов допустимых переходов) и *срабатывание перехода* (смена разметки и сброс часов некоторых переходов). В работе рассматривается слабая временная стратегия НВСП. Данная стратегия не ограничивает увеличение значений часов переходов. Таким образом, часы допустимого перехода могут пересечь верхнюю границу его временного интервала, что недопустимо в сильной стратегии. Было доказано, что НВСП со слабой временной стратегией ($\text{НВСП}_{\text{сл}}$) и НВСП с сильной временной стратегией не сравнимы по выразительной мощности [7].

После срабатывания перехода часы некоторых переходов сбрасываются, т.е. переустанавливаются в ноль. Порядок данного сброса определяется стратегией сброса часов. В литературе используются два подхода к сбросу часов: промежуточный и устойчиво атомарный. В промежуточной стратегии учитывается разметка, которая получается после удаления старых фишек и до добавления новых. Часы допустимого в новой разметке перехода сбрасываются, если он не был допустимым в промежуточной разметке. В устойчиво атомарной стратегии часы допустимого в новой разметке перехода сбрасываются только в том случае, если переход не был допустимым в старой разметке (до удаления фишек),

т.е. промежуточная разметка не учитывается. Известно, что для $\text{НВСП}_{\text{сл}}$ промежуточная стратегия не является более выразительной, чем устойчиво атомарная [8]. Кроме того, существуют системы, где при моделировании использовать устойчиво атомарную стратегию предпочтительнее. С другой стороны, в промежуточной стратегии проблемы достижимости и ограниченности являются разрешимыми, в отличие от устойчиво атомарной стратегии [6, 8].

Определение эквивалентностей и изучение их взаимосвязей для моделей параллельных и распределенных систем является важной задачей, которой посвящено большое число работ. В работе [3] было проведено систематическое исследование эквивалентностей для «невременных» СП; исследованы невременные и временные эквивалентности ДВСП. Для НВСП с сильной временной стратегией и промежуточной стратегией сброса часов в работе [4] была предложена «истинно параллельная» сематическая модель. На основе данной модели в работе [9] были разработаны эквивалентности и построена их иерархия.

Поскольку предложенный в [4] метод построения «истинно параллельной» семантики не может быть использован для $\text{НВСП}_{\text{сл}}$, то в [10] нами была предложена конструкция временных процессов, где время связано с состояниями системы. Используя полученную семантическую модель, в [2] для $\text{НВСП}_{\text{сл}}$ с устойчиво атомарной стратегией сброса часов были разработаны и изучены эквивалентности, построена их иерархия. Данная работа продолжает исследования [2, 10], посвященные семантическим моделям и эквивалентностям $\text{НВСП}_{\text{сл}}$, ее целью является исследование эквивалентностей относительно промежуточной и устойчиво атомарной стратегии сброса часов. Исходя из поставленной цели, в этой статье были решены следующие задачи: обобщение конструкций эквивалентностей работы [2] для промежуточной стратегии; изучение условий для совпадения данных эквивалентностей относительно разных стратегий сброса часов; исследование влияния выбора стратегии сброса часов на эквивалентность $\text{НВСП}_{\text{сл}}$.

Работа имеет следующую структуру. В разделе 2 мы познакомимся с $\text{НВСП}_{\text{сл}}$, промежуточной и устойчиво атомарной стратегиями сброса часов переходов. В разделе 3 изучим временные процессы, состоящие из ацикличной конструкции временных причинно-следственных сетей и гомоморфизма в $\text{НВСП}_{\text{сл}}$. На основе данной конструкции в разделе 4 мы рассмотрим языковые и бисимуляционные эквивалентности для дихотомии «интерливинг — истинный параллелизм». В разделе 5 мы исследуем взаимосвязи рассмотренных эквивалентностей относительно стратегий сброса часов. В заключительном разделе 6 бу-

дут рассмотрены полученные результаты и описаны планы дальнейшей работы.

2. Непрерывно-временные сети Петри

Рассмотрим определение модели сетей Петри с временными интервалами, представляющими временные задержки срабатываний переходов. Сначала напомним синтаксис и семантику «невременных» сетей Петри.

Определение 1. (Помеченная на множестве действий Act) сеть Петри *(СП) — это набор* $\mathcal{N} = (P,\ T,\ F,\ M_0,\ L)$, *где* P — *конечное* множество мест *и* T — *конечное* множество переходов *такие, что* $P \cap T = \emptyset$ *и* $P \cup T \neq \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ — отношение инцидентности; $\emptyset \neq M_0 \subseteq P$ — начальная разметка; $L : T \to Act$ — помечающая функция. *Для элемента* $x \in P \cup T$ *определим множество* ${}^\bullet x = \{y \mid (y, x) \in F\}$ входных *и множество* $x^\bullet = \{y \mid (x, y) \in F\}$ выходных элементов, *которые для подмножества элементов* $X \subseteq P \cup T$ *обобщаются соответственно до множеств* ${}^\bullet X = \bigcup_{x \in X} {}^\bullet x$ *и* $X^\bullet = \bigcup_{x \in X} x^\bullet$.

Разметка M *СП* $\mathcal{N}$ — *это любое подмножество множества* P. *Переход* $t \in T$ *является* допустимым в разметке M, *если* ${}^\bullet t \subseteq M$. *Обозначим через* $En(M)$ *множество всех переходов, допустимых в разметке* M. Срабатывание перехода, *допустимого в разметке* M, *приводит к новой разметке* $M' = (M \backslash {}^\bullet t) \cup t^\bullet$ *(обозначается* $M \xrightarrow{t} M'$*). Разметка* M *является* достижимой *в* $\mathcal{N}$, *если существует последовательность переходов* $t_1 \dots t_n$ *для* $n \geq 0$ *такая, что* $M_0 \xrightarrow{t_1} M_1 \dots M_{n-1} \xrightarrow{t_n} M_n = M$. *СП* $\mathcal{N}$ *называется* бесконтактной, *если для любой достижимой в* $\mathcal{N}$ *разметки* M *и любого перехода* t, *допустимого в* M, *верно, что* $(M \backslash {}^\bullet t) \cap t^\bullet = \emptyset$.

В работе рассматриваются *безопасные* сети Петри, т.е. при последовательном срабатывании переходов из начальной разметки сети каждое ее место будет иметь не более одной фишки. Это следствие определенного ранее свойства бесконтактности и того, что начальная маркировка — (обычное) множество.

Непрерывно-временная сеть Петри со слабой временной стратегией состоит из *базовой* сети Петри и *статической временной функции*, отображающей каждый переход во временной интервал. Подразумевается, что у каждого перехода есть собственные локальные часы, которые отсчитывают ход времени с момента, когда переход стал допустимым. Пусть $Interv = \{[a, b], [a, b) \mid a \in \mathbb{Q},\ b \in (\mathbb{Q} \cup \{\infty\}),\ a \leq b\}$ — множество интервалов с границами из множества рацилнальных чисел.

Определение 2. (Помеченная на Act) непрерывно-временная сеть Петри со слабой временной стратегией *(НВСП$_{сл}$) — это пара* $\mathcal{TN} = (\mathcal{N},\ D)$, *где* $\mathcal{N} = (P,\ T,\ F,\ M_0,\ L)$ — базовая *(помеченная на Act) сеть Петри и* $D : T \to Interv$ — статическая временная функция, *сопоставляющая каждому переходу из* T *временной интервал из* $Interv$.

Состояние НВСП$_{сл}$ $\mathcal{TN}$ — *это пара* $S = (M, I)$ *такая, что* M — *разметка и* $I :\ En(M) \to \mathbb{R}$ — динамическая временная функция. *Переход* $t \in En(M)$ может сработать в состоянии S, *если* $I(t) \in D(t)$. *Обозначим через* $Fi(S)$ *множество всех переходов из* $En(M)$, *которые могут сработать в состоянии* S. *Для НВСП$_{сл}$ возможны два способа изменения состояния* S: ход времени *и* действие *в результате* срабатывания перехода.

- Ход времени $\theta \in \mathbb{R}$ *приводит к новому состоянию* $S' = (M', I')$ *(обозначается* $S \xrightarrow{\theta} S'$*), где* $M' = M$ *и* $I'(t) = I(t) + \tau$ *для всех* $t \in En(M')$.
- Срабатывание перехода $t \in Fi(S)$ (действие $L(t)$) *приводит к новому состоянию* $S' = (M', I')$ *(обозначается* $S \xrightarrow{t}_{\dagger} S'$ *или* $S \xrightarrow{L(t)}_{\dagger} S'$*) такому, что:*

  – $M' = (M \backslash {}^{\bullet}t) \cup t^{\bullet}$;

  – $\forall t' \in En(M') : I'(t') = \begin{cases} 0, & \text{если } \uparrow enabled_{\dagger}(t', M, t), \\ I(t'), & \text{иначе;} \end{cases}$

  где $\dagger \in \{I, A\}$ *определяет стратегию сброса часов, а предикат* $\uparrow enabled_{\dagger}(t', M, t)$ *указывает на необходимость сброса часов для перехода* t'.

  I: *В случае промежуточной стратегии производится сброс часов перехода, который не являлся допустимым в промежуточной разметке* $M \backslash {}^{\bullet}t$ *и стал допустимым в разметке* M' *(после срабатывания перехода* t*):*

$$\uparrow enabled_I(t', M, t) = t' \notin En(M \backslash {}^{\bullet}t) \wedge t' \in En(M').$$

  A: *В случае устойчиво атомарной стратегии производится сброс часов перехода, который не являлся допустимым в разметке* M *(до срабатывания перехода* t*) и стал допустимым в разметке* M'*:*

$$\uparrow enabled_A(t', M, t) = t' \notin En(M) \wedge t' \in En(M').$$

Начальное состояние $\mathcal{TN}$ — *пара* $S_0 = (M_0, I_0)$, *где* M_0 — *начальная разметка и* $I_0(t) = 0$ *для всех* $t \in En(M_0)$.

Если $\sigma = \theta_0\ t_1\ \theta_1\ \ldots\ t_n\ \theta_n$ *для* $n \geq 0$ — *последовательность* временных значений $\theta_i \in \mathbb{R}$ $(0 \leq i \leq n)$ *и* срабатываний переходов $t_j \in T$ $(0 < j \leq n)$ *таких, что* $S^0 \xrightarrow{\theta_0} \widetilde{S}^0 \xrightarrow{t_1}_{\dagger}$

$S^1 \xrightarrow{\theta_1} \widetilde{S}^1 \dots \widetilde{S}^{n-1} \xrightarrow{t_n}_{\dagger} S^n \xrightarrow{\theta_n} \widetilde{S}^n$, *то* σ — пробег $\mathcal{TN}$ из состояния S^0 (в состояние $\widetilde{S}^n$) относительно стратегии сброса часов $\dagger$ *(обозначается* $S^0 \xrightarrow{\sigma}_{\dagger} \widetilde{S}^n$*).*

Пусть $\mathcal{TN}$ — НВСП$_{\text{сл}}$ и $\dagger \in \{I, A\}$. Обозначим через $\mathcal{RUN}_{\dagger}(\mathcal{TN}, S)$ множество всех пробегов НВСП$_{\text{сл}}$ $\mathcal{TN}$ из состояния S относительно стратегии сброса часов $\dagger$ и $\mathcal{RUN}_{\dagger}(\mathcal{TN}) = \mathcal{RUN}_{\dagger}(\mathcal{TN}, S_0)$. Состояние S будем называть *достижимым* в $\mathcal{TN}$ относительно стратегии сброса часов $\dagger$ (обозначается $S \in \mathcal{RS}_{\dagger}(\mathcal{TN})$), если существует пробег из $\mathcal{RUN}_{\dagger}(\mathcal{TN})$ в данное состояние. НВСП$_{\text{сл}}$ $\mathcal{TN}$ называется *бесконтактной*, если базовая сеть является бесконтактной. Кроме того, НВСП$_{\text{сл}}$ является *T-закрытой*, если ${}^\bullet t \neq \emptyset$ и $t^\bullet \neq \emptyset$ для всех переходов сети. В дальнейшем будем рассматривать только бесконтактные и T-закрытые НВСП$_{\text{сл}}$.

Результаты, представленные в работе [11], показывают, что для любого пробега из $\mathcal{RUN}_{\dagger}(\mathcal{TN})$ существует аналогичный пробег с той же последовательностью срабатываний переходов, в котором используются только целочисленные временные значения. Данный факт позволяет далее рассматривать ход времени в виде целых чисел.

Пример 1. *Рассмотрим НВСП$_{сл}$* $\mathcal{TN}_1$*, представленную на рисунке 1. На графике места сети изображены окружностями* ($\{p_1, p_2, p_3, p_4\}$)*, переходы изображены барьерами* ($\{t_1, t_2, t_3\}$)*, отношение инцидентности представлено направленными дугами, рядом с элементами указаны их имена. Каждому переходу ставится в соответствие временной интервал из* $Interv$ *и действие из* $Act = \{a, b\}$*. Начальной разметке* ($\{p_1, p_2\}$) *соответствует множество мест с фишками (жирными точками). Покажем, что последовательность* $\sigma = 0\ t_1\ 1\ t_2\ 2\ t_3\ 3$ *является пробегом НВСП$_{сл}$* $\mathcal{TN}_1$ *из начального состояния относительно промежуточной стратегии сброса часов* (I)*, т.е.* $\sigma \in \mathcal{RUN}_I(\mathcal{TN}_1)$*.*

- *Исходно сеть находится в состоянии* $S_0 = (M_0, I_0)$*, где* $M_0 = \{p_1, p_2\}$ *— начальная разметка,* $En(M_0) = \{t_1, t_2\}$ *— множество допустимых переходов в данной разметке и* $I_0(t) = 0$ *для каждого перехода* t *из* $En(M_0)$*. Ход времени* 0 *не изменит состояние, т.е.* $\widetilde{S}_0 = (\widetilde{M}_0, \widetilde{I}_0) = S_0$*,*
- *Поскольку переход* $t_1 \in En(\widetilde{M}_0)$ *и* $\widetilde{I}_0(t_1) = 0 \in D(t_1) = [0, 1]$*, то* t_1 *может сработать в состоянии* $\widetilde{S}_0$*. Срабатывание перехода* t_1 *из* $\widetilde{S}_0$ *приведет к новому состоянию* $S_1 = (M_1, I_1)$ *(*$\widetilde{S}_0 \xrightarrow{t_1}_I S_1$*), где* $M_1 = (\widetilde{M}_0 \backslash {}^\bullet t_1) \cup t_1^\bullet = \{p_3, p_2\}$*,* $En(M_1) = \{t_2\}$ *и* $I_1(t_2) = 0$*. Ход времени* 1 *из состояния* S_1 *приведет к состоянию* $\widetilde{S}_1 = (\widetilde{M}_1, \widetilde{I}_1) = (M_1, \widetilde{I}_1)$ *с* $\widetilde{I}_1(t_2) = 1$*.*
- *Переход* t_2 *является допустимым в разметке* $\widetilde{M}_1$ *и* $\widetilde{I}_1(t_2) = 1 \in D(t_2) = [0, 1]$*,*

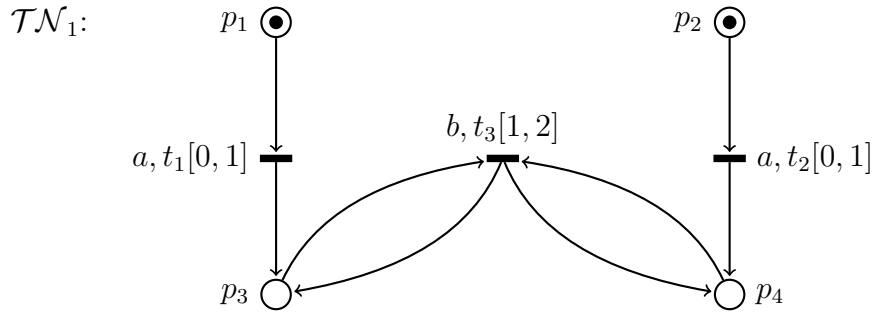

Рис. 1: Непрерывно-временная сеть Петри $\mathcal{TN}_1$

т.е. t_2 *может сработать в состоянии* $\widetilde{S}_1$. *Срабатывание* t_2 *из* $\widetilde{S}_1$ *приведет к новому состоянию* $S_2 = (M_2, I_2)$ *(*$\widetilde{S}_1 \xrightarrow{t_2}_I S_2$*), где* $M_2 = (\widetilde{M}_1 \backslash {}^\bullet t_2) \cup t_2{}^\bullet = \{p_3, p_4\}$, $En(M_2) = \{t_3\}$ *и* $I_2(t_3) = 0$, *поскольку переход* t_3 *не являлся допустимым в промежуточной разметке* $\widetilde{M}_1 \backslash {}^\bullet t_2$. *Ход времени* 2 *из состояния* S_2 *приведет к состоянию* $\widetilde{S}_2 = (\widetilde{M}_2, \widetilde{I}_2) = (M_2, \widetilde{I}_2)$ *с* $\widetilde{I}_2(t_3) = I_2(t_3) + 2 = 2$.

– *Допустимый переход* $t_3 \in En(\widetilde{M}_2) = \{t_3\}$ *готов сработать из* $\widetilde{S}_2$, *поскольку* $\widetilde{I}_2(t_3) = 2 \in D(t_3) = [1,2]$. *Следовательно,* $\widetilde{S}_2 \xrightarrow{t_3}_I S_3 = (M_3, I_3)$, *где* $M_3 = (\widetilde{M}_2 \backslash {}^\bullet t_3) \cup t_3{}^\bullet = \{p_3, p_4\}$, $En(M_3) = \{t_3\}$ *и* $I_3(t_3) = 0$, *так как переход* t_3 *не был допустимым в промежуточной разметке* $\widetilde{M}_2 \backslash {}^\bullet t_3$. *Заметим, что в случае устойчиво атомарной стратегии не было бы сброса локальных часов перехода* t_3. *Далее,* $S_3 \xrightarrow{3} \widetilde{S}_3$ *увеличит значение часов перехода* t_3, *т.е.* $\widetilde{S}_3 = (\widetilde{M}_3, \widetilde{I}_3) = (M_3, \widetilde{I}_3)$, *где* $\widetilde{I}_3(t_3) = 3$. *Заметим, что такой ход времени был бы невозможен в случае сильной временной стратегии, поскольку значение времени на часах перехода* t_3 *превысило верхнюю границу его временного интервала* $D(t_3) = [1,2]$.

Значит, $S_0 \xrightarrow{0} \widetilde{S}_0 \xrightarrow{t_1}_I S_1 \xrightarrow{1} \widetilde{S}_1 \xrightarrow{t_2}_I S_2 \xrightarrow{2} \widetilde{S}_2 \xrightarrow{t_3}_I S_3 \xrightarrow{3} \widetilde{S}_3 \in \mathcal{RUN}_I(\mathcal{TN}_1)$.

3. Временные процессы

Данный раздел представляет предложенную в [10] модель временных причинно-следственных процессов, используемую для описания поведения НВСП$_{\text{сл}}$. Сначала определим ацикличную причинно-следственную сеть (ПСС), состоящую из двух разных множеств — событий и условий, связанных отношением инцидентности. Каждое условие ПСС должно иметь не более одного входного и не более одного выходного условия. Кроме того, каждое событие ПСС должно иметь как входное, так и выходное условие.

Определение 3. (Помеченная на Act) причинно-следственная сеть *(ПСС) — это ациклическая сеть* $N = (B,\ E,\ G,\ l)$, *где* B — *конечное* множество условий *и* E — *конечное* множество событий *такие, что* $B \cap E = \emptyset$; $G \subseteq (B \times E) \cup (E \times B)$ — отношение инцидентности *такое, что* $\mid b^\bullet \mid \leq 1$, $\mid {}^\bullet b \mid \leq 1$ *для всех* $b \in B$ *и* $E = {}^\bullet B = B^\bullet$; $l : E \to Act$ — помечающая функция.

Для ПСС $N = (B, E, G, l)$ введем дополнительные понятия и обозначения:

- $x \prec x' \iff x\ G^+\ x'$ и $x \preceq x' \iff x\ G^*\ x'$, где $x, x' \in B \cup E$, — *отношение причинной зависимости на элементах* ПСС N.
- $x \smile x' \iff \neg(x \preceq x') \wedge \neg(x' \preceq x)$, где $x, x' \in B \cup E$, — *отношение параллелизма на элементах* ПСС N.
- $\smile$*-множество* ПСС N — непустое подмножество условий $B' \subseteq B$ такое, что $b \smile b'$ для всех $b \neq b' \in B'$.
- *Сечение* C ПСС N — максимальное по включению $\smile$-множество.
- $\mathcal{CUT}(N)$ — множество всех сечений в ПСС N.
- ${}^\bullet N = \{b \in B \mid {}^\bullet b = \emptyset\}$ — *начальное сечение* ПСС N.
- $N^\bullet = \{b \in B \mid b^\bullet = \emptyset\}$ — *конечное сечение* ПСС N.

Неформально говоря, сечение — это «разметка» ПСС. Событие $e \in E$ *допустимо в сечении* $C \in \mathcal{CUT}(N)$ (обозначается $e \in En(C)$), если ${}^\bullet e \subseteq C$. В этом случае будем писать $C \xrightarrow{e} C'$ или $C \longrightarrow C'$, где $C' = (C \backslash {}^\bullet e) \cup e^\bullet$. На основе отношения $(\longrightarrow)$ определяются отношения причинной зависимости и параллелизма на сечениях ПСС:

- $C \prec C' \iff C \to^+ C'$ и $C \preceq C' \iff C \to^* C'$, где $C, C' \in \mathcal{CUT}(N)$, — *отношение причинной зависимости на сечениях ПСС.*
- $C \smile C' \iff \neg(C \preceq C') \wedge \neg(C' \preceq C)$, где $C, C' \in \mathcal{CUT}(N)$, — *отношение параллелизма на сечениях* ПСС.

Для сечения $C \in \mathcal{CUT}(N)$ обозначим:

- $Cut(C) = \{C' \in \mathcal{CUT}(N) \mid C \smile C'\}$ — *множество сечений ПСС* N*, параллельных сечению* C.
- $\downarrow C = \{e \in E \mid e \preceq e', e' \in {}^\bullet C\}$ — *множество событий ПСС* N*, предшествующих сечению* C.

Описываемое сечением состояние достигается после того, как предшествующие сечению события произошли. Как следствие, $C \preceq C' \iff \downarrow C \subseteq \downarrow C'$. Отметим, что $\downarrow {}^\bullet N = \emptyset$ и $\downarrow N^\bullet = E$, где E — множество всех событий ПСС N.

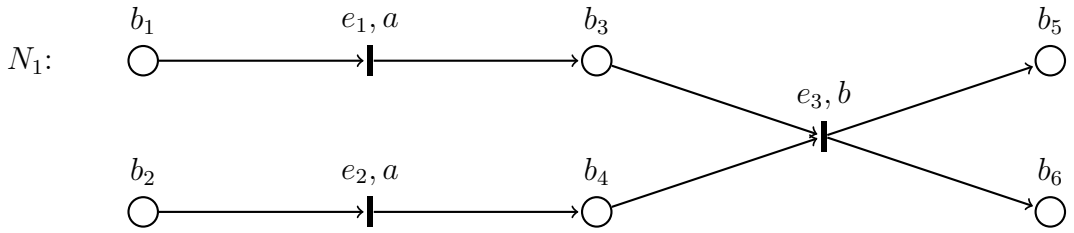

Рис. 2: Причинно-следственная сеть N_1

Пример 2. *Рассмотрим изображенную на рисунке 2 сеть* $N_1 = (B, E, G, l)$, *где* $B = \{b_1, \ldots, b_6\}$ — *множество условий,* $E = \{e_1, e_2, e_3\}$ — *множество событий,* $G = \{(b_1, e_1), (b_2, e_2), (e_1, b_3), (e_2, b_4), (b_3, e_3), (b_4, e_3), (e_3, b_5), (e_3, b_6)\}$ — *отношение инцидентности и* $l : E \to \{a, b\}$ — *помечающая функция такая, что* $l(e_1) = l(e_2) = a$ *и* $l(e_3) = b$. *Видно, что* $|{}^\bullet b| \leq 1$ *и* $|b^\bullet| \leq 1$ *для всех* $b \in B$. *Кроме того,* $E = {}^\bullet B = B^\bullet$. *Значит,* N_1 — *помеченная на* $Act = \{a, b\}$ *ПСС.*

Поскольку $b_1 \smile b_2$, *т.е.* $\neg(b_1 \preceq b_2) \wedge \neg(b_2 \preceq b_1)$, *то* $\{b_1, b_2\}$ — $\smile$*-множество. Видно, что не существует* $b \in B$ *такого, что* $b \smile b_1$ *и* $b \smile b_2$ *одновременно. Значит,* $\{b_1, b_2\}$ — *максимальное по включению* $\smile$*-множество, т.е. сечение. Аналогично можно показать, что* $\{b_1, b_4\}$, $\{b_2, b_3\}$, $\{b_3, b_4\}$, $\{b_5, b_6\}$ *также являются сечениями. Сечение* $\{b_1, b_2\}$ ($\{b_5, b_6\}$) *с условиями без входных (выходных) событий — начальное (конечное) сечение ПСС* N_1. *Из* ${}^\bullet e_1 \subseteq \{b_1, b_2\}$, $(\{b_1, b_2\} \backslash {}^\bullet e_1) \cup e_1{}^\bullet = \{b_2, b_3\}$ *следует, что* $\{b_1, b_2\} \xrightarrow{e_1} \{b_2, b_3\}$ *и* $\{b_1, b_2\} \preceq \{b_2, b_3\}$. *С другой стороны,* $\{b_1, b_4\} \smile \{b_2, b_3\}$, *так как* $(\downarrow\{b_1, b_4\} = \{e_2\}) \not\subseteq (\downarrow\{b_2, b_3\} = \{e_1\})$ *и* $\downarrow\{b_2, b_3\} \not\subseteq \downarrow\{b_1, b_4\}$, *т.е.* $\neg(\{b_1, b_4\} \preceq \{b_2, b_3\})$ *и* $\neg(\{b_2, b_3\} \preceq \{b_1, b_4\})$.

Для определения связи между ПСС и СП используется гомоморфизм — отображение, сохраняющее отношение инцидентности и помечающую функцию.

Определение 4. *Пусть* $N = (B,\ E,\ G,\ l)$ — *ПСС,* $\mathcal{N} = (P,\ T,\ F,\ M_0,\ L)$ — *СП и* $M \subseteq P$ — *разметка* $\mathcal{N}$. Гомоморфизмом из N в $\mathcal{N}$ относительно разметки M *называется отображение* $\varphi : (B \cup E) \to (P \cup T)$, *для которого выполняется:*

- $\varphi(B) \subseteq P$ *и* $\varphi(E) \subseteq T$;
- *сужение* φ *на подмножество* ${}^\bullet e$ — *биекция между* ${}^\bullet e$ *и* ${}^\bullet\varphi(e)$ *для всех* $e \in E$;
- *сужение* φ *на подмножество* $e^\bullet$ — *биекция между* $e^\bullet$ *и* $\varphi(e)^\bullet$ *для всех* $e \in E$;
- *сужение* φ *на подмножество* ${}^\bullet N$ — *биекция между* ${}^\bullet N$ *и* M;
- $l(e) = L(\varphi(e))$ *для всех* $e \in E$.

Пример 3. *Рассмотрим базовую СП* $\mathcal{N}_1 = (P, T, F, M_0, L)$ *НВСП*$_{сл}$ $\mathcal{TN}_1$*, изображенную на рисунке 1, ПСС* $N_1 = (B, E, G, l)$*, изображенную на рисунке 2, и функцию* $\varphi : (B \cup E) \to (P \cup T)$*, связывающую условия и события ПСС* N_1 *соответственно с местами и переходами СП* $\mathcal{N}_1$ *так, что* $\varphi(b_1) = p_1$*,* $\varphi(b_2) = p_2$*,* $\varphi(b_3) = \varphi(b_5) = p_3$*,* $\varphi(b_4) = \varphi(b_6) = p_4$ *и* $\varphi(e_i) = t_i$ *для* $1 \le i \le 3$*. Легко проверить, что сужение* φ *на подмножество* ${}^\bullet e$ *(*$e^\bullet$*) является биекцией между данным подмножеством и* ${}^\bullet\varphi(e)$ *(*$\varphi(e)^\bullet$*) для всех* $e \in E$*. Например, для события* e_3 *множество входных условий* $\{b_3, b_4\}$ *биективно отображается на множество входных мест* $\{p_3, p_4\}$ *перехода* $\varphi(e_3) = t_3$*, а множество выходных условий* $\{b_5, b_6\}$ *— на то же множество* $\{p_3, p_4\}$ *выходных мест* t_3*. Видно, что сужение* φ *на подмножество* ${}^\bullet N_1 = \{b_1, b_2\}$ *является биекцией между данным подмножеством и начальной разметкой* $M_0 = \{p_1, p_2\}$*. Кроме того,* $l(e) = L(\varphi(e))$ *для всех* $e \in E$*. Следовательно,* φ *— гомоморфизм из* N_1 *в* $\mathcal{N}_1$ *относительно начальной разметки* M_0*.*

Определим временное расширение ПСС, где сечениям ставится в соответствие временное значение (длительность соответствующего состояния) либо указывается, что сечение (состояние) недостижимо по времени.

Определение 5. (Помеченная на Act) временная ПСС *(ВПСС) — это пара* $TN = (N,\ \tau)$*, где* N *— (помеченная на* Act*) ПСС и* $\tau : \mathcal{CUT}(N) \to \mathbb{N} \cup \{\bot\}$ — временная функция *такая, что для каждого сечения* $C \in \mathcal{CUT}(N)$ *выполняется:*

$$\tau(C) = \bot \iff \exists C' \in Cut(C) \colon \tau(C') > 0.$$

Пусть $\mathcal{RC}(TN) = \{C \in \mathcal{CUT}(N) \mid \tau(C) \in \mathbb{N}\}$ — множество всех временных сечений.

Рассмотрим ВПСС $TN = (N, \tau)$, где $N = (B, E, G, l)$. Событие $e \in E$ *может произойти в* $C \in \mathcal{RC}(TN)$ (обозначается $e \in Fi(C)$), если $e \in En(C)$ и $C' = (C \setminus {}^\bullet e) \cup e^\bullet \in \mathcal{RC}(TN)$. В этом случае будем писать $C \overset{e}{\Longrightarrow} C'$ или $C \Longrightarrow C'$. Таким образом, отношение $(\Longrightarrow)$ запрещает получение недостижимого по времени сечения в результате выполнения события, так как это означает, что время его выполнения еще не наступило. Заметим, что ${}^\bullet N, N^\bullet \in \mathcal{RC}(TN)$, поскольку они не имеют параллельных сечений.

Помеченные на одном и том же множестве Act ВПСС $TN = (N,\ \tau)$ и $TN' = (N',\ \tau')$, где $N = (B,\ E,\ G,\ l)$ и $N' = (B',\ E',\ G',\ l')$, *изоморфны* (обозначается $TN \simeq TN'$ или $f : TN \simeq TN'$), если существует биективное отображение $f : B \cup E \to B' \cup E'$ (*изоморфизм*) такое, что:

- $f(B) = B'$ и $f(E) = E'$;
- $xGy \iff f(x)G'f(y)$ для всех $x, y \in B \cup E$;
- $\tau(C) = \tau'(f(C))$ для всех $C \in \mathcal{CUT}(N)$;
- $l(e) = l'(f(e))$.

Класс эквивалентности относительно изоморфизма, образованный ВПСС TN, будем обозначать как $[TN]_{\simeq}$.

Пусть $TN = (N, \tau)$ — ВПСС, $\mathcal{TN} = (\mathcal{N}, D)$ — НВСП$_{\text{сл}}$ и $S = (M, I)$ — состояние $\mathcal{TN}$. Обозначим через $\mathcal{HOM}_S(TN, \mathcal{TN})$ множество всех гомоморфизмов из N в $\mathcal{N}$ относительно разметки M.

Пример 4. *Рассмотрим ПСС N_1, изображенную на рисунке 2. Определим функцию $\tau : \mathcal{CUT}(N_1) = \{\{b_1, b_2\}, \{b_1, b_4\}, \{b_2, b_3\}, \{b_3, b_4\}, \{b_5, b_6\}\} \to \mathbb{N} \cup \{\perp\}$ следующим образом: $\tau(\{b_1, b_2\}) = 0, \tau(\{b_1, b_4\}) = \perp, \tau(\{b_2, b_3\}) = 1, \tau(\{b_3, b_4\}) = 2$ и $\tau(\{b_5, b_6\}) = 3$. Поскольку в данной сети только два параллельных сечения $\{b_1, b_4\} \smile \{b_2, b_3\}$, $\tau(\{b_1, b_4\}) = \perp$ и $\tau(\{b_2, b_3\}) = 1 > 0$, то функция τ удовлетворяет условию из определения 5. Следовательно, $TN_1 = (N_1, \tau)$ — ВПСС и $\mathcal{RC}(TN_1) = \{\{b_1, b_2\}, \{b_2, b_3\}, \{b_3, b_4\}, \{b_5, b_6\}\}$.*

Рассмотрим последовательность событий ВПСС (временной график), для которой существует цепочка временных сечений, полученная в результате последовательного выполнения данных событий.

Определение 6. *Пусть $TN = (N, \tau)$ — ВПСС и $C, C' \in \mathcal{RC}(TN)$. Последовательность событий $\omega = e_1 \dots e_n$ $(n \geq 0)$ называется* временным графиком из C в C' *ВПСС TN, если существует цепочка сечений из $\mathcal{RC}(TN)$ вида:*

$$C = C_0 \overset{e_1}{\Longrightarrow} C_1 \dots C_{n-1} \overset{e_n}{\Longrightarrow} C_n = C'.$$

Пусть $\mathcal{GRF}_{C \Rightarrow C'}(TN)$ — множество всех временных графиков ВПСС TN из C в C'. Обозначим $\mathcal{GRF}_{\Rightarrow C}(TN) = \mathcal{GRF}_{\bullet N \Rightarrow C}(TN)$ и $\mathcal{GRF}(TN) = \mathcal{GRF}_{\bullet N \Rightarrow N \bullet}(TN)$.

Пример 5. *Пусть $TN_1 = (N_1, \tau)$ — ВПСС из примера 4. Рассмотрим последовательность $\omega = e_1\ e_2\ e_3$. Очевидно, что $\{b_1, b_2\} \xrightarrow{e_1} \{b_3, b_2\} \xrightarrow{e_2} \{b_3, b_4\} \xrightarrow{e_3} \{b_5, b_6\}$. Из примера 4, $\{b_1, b_2\}, \{b_3, b_2\}, \{b_3, b_4\}, \{b_5, b_6\} \in \mathcal{RC}(TN_1)$. Значит, $\{b_1, b_2\} \overset{e_1}{\Longrightarrow} \{b_3, b_2\} \overset{e_2}{\Longrightarrow} \{b_3, b_4\} \overset{e_3}{\Longrightarrow} \{b_5, b_6\}$, где $\{b_1, b_2\}$ — начальное, а $\{b_5, b_6\}$ — конечное сечение. Следовательно, $\omega \in \mathcal{GRF}(TN)$.*

Таким образом, временные графики иллюстрируют «выполнения» ВПСС. Следующее утверждение доказывает существование временного графика между двумя произволь-

ными временными сечениями и, как следствие, между начальным и конечным сечением ВПСС.

Утверждение 1. *Пусть TN — ВПСС. Если $C, C' \in \mathcal{RC}(TN)$ и $C \preceq C'$, то существует $\omega' \in \mathcal{GRF}_{C \Rightarrow C'}(TN)$.*

Доказательство. Аналогично доказательству предложения 1 работы [1]. □

Определим функцию **Clock**, которая будет связывать временные конструкции ВПСС и НВСП$_{\text{сл}}$ относительно промежуточной и устойчиво атомарной стратегий сброса часов. Данная функция определяется в контексте некоторого состояния НВСП$_{\text{сл}}$ и гомоморфизма φ относительно разметки этого состояния. В качестве параметров функции выступают временное сечение C и переход t, допустимый в разметке $\varphi(C)$. Значение функции **Clock** определяется с помощью произвольного временного графика из начального сечения в сечение C и соответствует времени с того момента, как переход t стал допустимым и оставался таким до конца временного графика.

Пусть $\dagger \in \{I, A\}$, TN — ВПСС, $\mathcal{TN}$ — НВСП$_{\text{сл}}$, $S = (M, I)$ — состояние $\mathcal{TN}$ и $\varphi \in \mathcal{HOM}_S(TN, \mathcal{TN})$. Для сечения $C \in \mathcal{RC}(TN)$, перехода $t \in En(\varphi(C))$ и временного графика $\omega = e_1 \dots e_n \in \mathcal{GRF}_{\Rightarrow C}(TN)$ из начального сечения в сечение C с последовательностью $C_0 \overset{e_1}{\Longrightarrow} C_1 \dots C_{n-1} \overset{e_n}{\Longrightarrow} C_n = C$ определим функцию $\mathbf{Clock}^{\dagger}_{\varphi,S}(\omega, t)$ следующим образом:

$$\mathbf{Clock}^{I}_{\varphi,S}(\omega, t) = \begin{cases} \sum\limits_{max(k)<i\le n} \tau(C_i), \text{если } \exists k < n \colon t \notin En(\varphi(C_k \cap C_n)), \\ \\ \sum\limits_{0\le i\le n} \tau(C_i) + I(t), \text{иначе;} \end{cases}$$

$$\mathbf{Clock}^{A}_{\varphi,S}(\omega, t) = \begin{cases} \sum\limits_{max(k)<i\le n} \tau(C_i), \text{если } \exists k < n \colon t \notin En(\varphi(C_k)), \\ \\ \sum\limits_{0\le i\le n} \tau(C_i) + I(t), \text{иначе;} \end{cases}$$

где $\dagger \in \{I, A\}$ указывает на промежуточную или устойчиво атомарную стратегию сброса часов.

В работе [2] показано, что для временного сечения C значение функции **Clock** не зависит от выбора временного графика из начального сечения в сечение C. Кроме того, в качестве параметров функции **Clock** можно рассматривать только сечение $C \in \mathcal{RC}(TN)$

и переход $t \in En(\varphi(C))$.

Далее определим временной процесс как пару из ВПСС и гомоморфизма с ограничениями на корректность временных конструкций.

Определение 7. *Пусть* $\dagger \in \{I, A\}$, $\mathcal{TN} = (\mathcal{N}, D)$ — *НВСП*$_{сл}$, $S = (M, I) \in \mathcal{RS}_\dagger(\mathcal{TN})$, $TN = (N, \tau)$ — *ВПСС и* $\varphi \in \mathcal{HOM}_S(TN, \mathcal{TN})$. *Пара* $\pi = (TN, \varphi)$ *называется* временным процессом НВСП$_{\text{сл}}$ $\mathcal{TN}$ из состояния S относительно стратегии сброса часов $\dagger$, *если для каждого сечения* $C \in \mathcal{RC}(TN)$ *и события* $e \in Fi(C)$ *выполняется:*

$$\mathbf{Clock}^{\dagger}_{\varphi,S}(C, \varphi(e)) \in D(\varphi(e)).$$

Пусть $\mathcal{PRC}_\dagger(\mathcal{TN}, S)$ — множество всех временных процессов НВСП$_{\text{сл}}$ $\mathcal{TN}$ из состояния $S \in \mathcal{RS}_\dagger(\mathcal{TN})$ относительно стратегии сброса часов $\dagger$. Кроме того, обозначим $\mathcal{PRC}_\dagger(\mathcal{TN}) = \mathcal{PRC}_\dagger(\mathcal{TN}, S_0)$, где S_0 — начальное состояние $\mathcal{TN}$. Временной процесс $\pi_0 = (((B_0, \emptyset, \emptyset, \emptyset), \tau_0), \varphi_0)$ такой, что φ_0 — гомоморфизм относительно начальной разметки ($\varphi(B_0) = M_0$) и $\tau_0 \equiv 0$, называется *начальным.*

Два временных процесса $\pi = (TN,\ \varphi)$ и $\pi' = (TN',\ \varphi')$ из $\mathcal{PRC}_\dagger(\mathcal{TN}, S)$, где $\dagger \in \{I, A\}$, $S \in \mathcal{RS}_\dagger(\mathcal{TN})$ и $TN = ((B,\ E,\ G,\ l),\ \tau)$, *изоморфны* (обозначается $\pi \simeq \pi'$ или $f : \pi \simeq \pi'$), если существует *изоморфизм* $f : TN \simeq TN'$ такой, что $\varphi(x) = \varphi'(f(x))$ для любого $x \in B \cup E$. Класс эквивалентности относительно изоморфизма, образованный временным процессом π, будем обозначать как $[\pi]_\simeq$.

Пример 6. *Рассмотрим НВСП*$_{сл}$ $\mathcal{TN}_1 = (\mathcal{N}_1, D)$ *из примера 1, ВПСС* $TN_1 = (N_1, \tau)$ *из примера 4 и гомоморфизм* φ *из* N_1 *в* $\mathcal{N}_1$ *относительно начальной разметки из примера 3. Покажем, что пара* $\pi = (TN_1, \varphi)$ *является временным процессом* $\mathcal{TN}_1$ *из начального состояния* $S_0 = (M_0, (I_0 \equiv 0))$ *относительно промежуточной стратегии сброса часов. Для этого вычислим для каждого события* e *и сечения* $C \in \mathcal{RC}(\mathcal{TN}_1)$, *в котором оно может произойти, значение функции* $\mathbf{Clock}^{I}_{\varphi,S_0}(C, \varphi(e))$ *и убедимся, что оно принадлежит интервалу* $D(\varphi(e))$. *Напомним все временные сечения ВПСС* TN_1*:* $C_0 = \{b_1, b_2\}$, $C_1 = \{b_3, b_2\}$, $C_2 = \{b_3, b_4\}$, $C_3 = \{b_5, b_6\}$. *Легко проверить, что* $C_0 \xrightarrow{e_1} C_1 \xrightarrow{e_2} C_2 \xrightarrow{e_3} C_3$. *Тогда:*

- $\mathbf{Clock}^{I}_{\varphi,S_0}(C_0, \varphi(e_1)) = \tau(C_0) = 0 \in D(\varphi(e_1) = t_1) = [0, 1]$,
- $\mathbf{Clock}^{I}_{\varphi,S_0}(C_1, \varphi(e_2)) = \tau(C_0) + \tau(C_1) = 0 + 1 \in D(\varphi(e_2) = t_2) = [0, 1]$,
- $\mathbf{Clock}^{I}_{\varphi,S_0}(C_2, \varphi(e_3)) = \tau(C_2) = 2 \in D(\varphi(e_3) = t_3) = [1, 2]$.

Следовательно, $\pi = (TN_1, \varphi) \in \mathcal{PRC}_I(\mathcal{TN}_1)$.

Установим взаимосвязь между временными процессами и пробегами НВСП$_{\text{сл}}$. Введем функцию, отображающую временной график в последовательность хода времени и срабатываний переходов НВСП$_{\text{сл}}$.

Пусть $\mathcal{TN}$ — НВСП$_{\text{сл}}$, TN — ВПСС, S — состояние $\mathcal{TN}$ и $\varphi \in \mathcal{HOM}_S(TN, \mathcal{TN})$. Для временного графика $\omega = e_1 \dots e_n \in \mathcal{GRF}_{\Rightarrow C}(TN)$ из начального сечения в сечение $C \in \mathcal{RC}(TN)$ с последовательностью $C_0 \stackrel{e_1}{\Longrightarrow} C_1 \dots C_{n-1} \stackrel{e_n}{\Longrightarrow} C_n = C$ определим функцию Run_φ следующим образом: $Run_\varphi(TN, \omega) = \tau(C_0)\ \varphi(e_1)\ \tau(C_1)\ \dots\ \varphi(e_n)\ \tau(C_n)$.

Покажем, что если φ — гомоморфизм относительно начальной разметки и Run_φ отображает ω в пробег из начального состояния в состояние (M', I') относительно стратегии $\dagger \in \{I, A\}$, то $\varphi(C) = M'$ и значение $\mathbf{Clock}^\dagger_{\varphi, S_0}(C, t)$ равно $I'(t)$ для любого перехода $t \in En(M')$.

Утверждение 2. *Пусть* $\dagger \in \{I, A\}$, $\mathcal{TN}$ *—* НВСП$_{сл}$, TN *—* ВПСС, $\varphi \in \mathcal{HOM}_{S_0}(TN, \mathcal{TN})$ *и* $\omega = e_1 \dots e_n \in \mathcal{GRF}_{\Rightarrow C}(TN)$ *для* $C \in \mathcal{RC}(TN)$ *с последовательностью* $C_0 \stackrel{e_1}{\Longrightarrow} C_1 \dots C_{n-1} \stackrel{e_n}{\Longrightarrow} C_n = C$. *Если* $Run_\varphi(TN, \omega) \in \mathcal{RUN}_\dagger(\mathcal{TN})$, *т.е.* $(M_0, I_0) \stackrel{\tau(C_0)}{\longrightarrow} (M_0, \widetilde{I}_0) \stackrel{\varphi(e_1)}{\longrightarrow}_\dagger (M_1, I_1) \stackrel{\varphi(C_1)}{\longrightarrow} (M_1, \widetilde{I}_1) \dots (M_{n-1}, \widetilde{I}_{n-1}) \stackrel{\varphi(e_n)}{\longrightarrow}_\dagger (M_n, I_n) \stackrel{\varphi(C_n)}{\longrightarrow} (M_n, \widetilde{I}_n)$, *то для каждого* $0 \le i \le n$ *выполняется:*

(а) Сужение функции φ *на* C_i *— биекция между* C_i *и* M_i.

(б) $\mathbf{Clock}^\dagger_{\varphi, S_0}(C_i, t) = \widetilde{I}_i(t)$ *для всех* $t \in En(M_i)$.

Доказательство. Аналогично доказательству утверждения 2 работы [10]. □

Для временного процесса $\pi = (TN, \varphi)$ относительно стратегии $\dagger \in \{I, A\}$ обозначим через $\mathcal{RUN}_\dagger(\pi)$ множество всех последовательностей $Run_\varphi(TN, \omega)$, где $\omega \in \mathcal{GRF}(TN)$. Рассмотренные в работе [10] теоремы указывают на взаимно однозначное соответствие между пробегами НВСП$_{\text{сл}}$ и временными графиками временных процессов НВСП$_{\text{сл}}$, которое достигается при помощи функции Run.

Утверждение 3. *Пусть* $\dagger \in \{I, A\}$ *и* $\mathcal{TN}$ *—* НВСП$_{сл}$.

- *Если* $\pi \in \mathcal{PRC}_\dagger(\mathcal{TN})$, *то* $\mathcal{RUN}_\dagger(\pi) \subseteq \mathcal{RUN}_\dagger(\mathcal{TN})$.
- *Если* $\sigma \in \mathcal{RUN}_\dagger(\mathcal{TN})$, *то существует единственный с точностью до изоморфизма* $\pi \in \mathcal{PRC}_\dagger(\mathcal{TN})$ *такой, что* $\sigma \in \mathcal{RUN}_\dagger(\pi)$.

Доказательство. Аналогично доказательствам теорем 1-3 работы [10]. □

Далее рассмотрим последовательности изменений (расширений) временных процессов и их взаимосвязи с состояниями НВСП$_{\text{сл}}$. Сначала введем конструкцию «подсети» ПСС

— части ПСС, заключенной между двумя её сечениями.

Пусть $N = (B,\ E,\ G,\ l)$ — ПСС и $C, C' \in \mathcal{CUT}(N)$ такие, что $C \preceq C'$. Определим $N_{C \to C'} = (B',\ E',\ G',\ l')$, где:

- $B' = \bigcup_{C \preceq \widehat{C} \preceq C'} \widehat{C}$;
- $E' = {\downarrow}C' \backslash {\downarrow}C$;
- $G' = G \cap ((B' \times E') \cup (E' \times B'))$;
- $l' = l|_{E'}$.

Используя конструкцию причинно-следственных «подсетей», определим понятие расширения для ВПСС.

Пусть $\widetilde{TN} = (\widetilde{N},\ \widetilde{\tau})$ — ВПСС. Будем писать $TN \xrightarrow{\widehat{TN}} \widetilde{TN}$, если существует $\widetilde{C} \in \mathcal{RC}(\widetilde{TN})$ такое, что:

- $TN = (N,\ \tau)$, $\widehat{TN} = (\widehat{N},\ \widehat{\tau})$;
- $N = \widetilde{N}_{\bullet \widetilde{N} \to \widetilde{C}}$, $\widehat{N} = \widetilde{N}_{\widetilde{C} \to \widetilde{N} \bullet}$;
- $\tau : \mathcal{CUT}(N) \to \mathbb{N} \cup \{\bot\}$ и $\tau(C) = \widetilde{\tau}(C)$ для всех $C \in \mathcal{CUT}(N) \backslash \widetilde{C}$;
- $\widehat{\tau} : \mathcal{CUT}(\widehat{N}) \to \mathbb{N} \cup \{\bot\}$ и $\widehat{\tau}(\widehat{C}) = \widetilde{\tau}(\widehat{C})$ для всех $\widehat{C} \in \mathcal{CUT}(\widehat{N}) \backslash \widetilde{C}$;
- $\widetilde{\tau}(\widetilde{C}) = \tau(\widetilde{C}) + \widehat{\tau}(\widetilde{C})$.

Аналогично определим расширения для временных процессов НВСП$_{\text{сл}}$.

Рассмотрим произвольный временной процесс $\widetilde{\pi} = (\widetilde{TN}, \widetilde{\varphi}) \in \mathcal{PRC}(\mathcal{TN})$. Будем писать $\pi \xrightarrow{\widehat{\pi}} \widetilde{\pi}$, если выполняется:

- $\pi = (TN, \varphi)$ и $\widehat{\pi} = (\widehat{TN}, \widehat{\varphi})$;
- $TN \xrightarrow{\widehat{TN}} \widetilde{TN}$;
- $\varphi = \widetilde{\varphi}|_{B \cup E}$ и $\widehat{\varphi} = \widetilde{\varphi}|_{\widehat{B} \cup \widehat{E}}$.

В этом случае $\widetilde{\pi}$ — *расширение* π *на* $\widehat{\pi}$. Кроме того, π будем называть *префиксом*, а $\widehat{\pi}$ — *суффиксом* $\widetilde{\pi}$.

В работе [1] показано, что префикс и суффикс расширения временного процесса НВСП$_{\text{сл}}$ также являются временными процессами.

Пусть $\pi \xrightarrow{\widehat{\pi}} \widetilde{\pi}$ для $\widetilde{\pi} \in \mathcal{PRC}_{\dagger}(\mathcal{TN})$, НВСП$_{\text{сл}}$ $\mathcal{TN}$ и $\dagger \in \{I, A\}$. Рассмотрим частные случаи расширений временных процессов:

- $\widetilde{\pi}$ — *расширение* π *на время* θ (обозначается $\pi \xrightarrow{\theta} \widetilde{\pi}$), если $\widehat{E} = \emptyset$ и $\widehat{\tau}(\widehat{B}) = \theta$;
- $\widetilde{\pi}$ — *расширение* π *на действие* a (обозначается $\pi \xrightarrow{a} \widetilde{\pi}$), если $\widehat{E} = \{e\}, \widehat{l}(e) = a$ и $\widehat{\tau} \equiv 0$.

4. Эквивалентности

Рассмотренная в предыдущей главе модель временных процессов позволяет сравнивать поведение НВСП$_{\text{сл}}$ с учетом частичного порядка между их элементами. В этом разделе для НВСП$_{\text{сл}}$ мы рассмотрим предложенные в [2] языковые (поведение описывается набором «линейных» последовательностей выполнений системы) и бисимуляционные (учитывают точки недетерминированного выбора альтернативных действий) эквивалентности для семантики *интерливинга* и *процессно-сетевой семантики*. Далее будем использовать $\dagger \in \{I, A\}$ для обозначения промежуточной или устойчиво атомарной стратегии сброса часов НВСП$_{\text{сл}}$. Кроме того, через i будем обозначать интерливинговую, а через n — процессно-сетевую семантики.

Обобщим классическое определение языковой эквивалентности, основанное на пробегах сетей Петри. Для НВСП$_{\text{сл}}$ $\mathcal{TN}$ с помечающей функцией L и пробега $\sigma = \theta_0\ t_1\ \theta_1 \dots t_n\ \theta_n \in \mathcal{RUN}_\dagger(\mathcal{TN}, S)$ из произвольного состояния S определим последовательность из хода времени и действий $L(\mathcal{TN}, \sigma) = \theta_0\ L(t_1)\ \theta_1 \dots L(t_n)\ \theta_n$.

Определение 8. *Пусть $\mathcal{TN}$, $\mathcal{TN}'$ — НВСП$_{\text{сл}}$, помеченные на Act.*

- $Lang^\dagger(\mathcal{TN}) = \{L(\mathcal{TN}, \sigma) \mid \sigma \in \mathcal{RUN}_\dagger(\mathcal{TN})\}$.
- *$\mathcal{TN}$ и $\mathcal{TN}'$* интерливингово языково эквивалентны относительно стратегии $\dagger$ *(обозначается $\mathcal{TN} \cong_i^\dagger \mathcal{TN}'$), если $Lang^\dagger(\mathcal{TN}) = Lang^\dagger(\mathcal{TN}')$.*

Далее определим языковые эквивалентности в двух семантиках, основанные на временных процессах НВСП$_{\text{сл}}$. Для ВПСС $TN = ((B,\ E,\ G,\ l),\ \tau)$ и временного графика $\omega = e_1 \dots e_n \in \mathcal{GRF}(TN)$ с последовательностью сечений $C_0 \xrightarrow{e_1} C_1 \dots C_{n-1} \xrightarrow{e_n} C_n$ определим последовательность из хода времени и действий $l(TN, \omega) = \tau(C_0)\ l(e_1)\ \tau(C_1) \dots l(e_n)\ \tau(C_n)$.

Определение 9. *Пусть $\mathcal{TN}$, $\mathcal{TN}'$ — НВСП$_{\text{сл}}$, помеченные на Act, и $\star \in \{i, n\}$.*

- $Trace_i^\dagger(\mathcal{TN}) = \{l(TN, \omega) \mid (TN, \varphi) \in \mathcal{PRC}_\dagger(\mathcal{TN}), \omega \in \mathcal{GRF}(TN)\}$;
- $Trace_n^\dagger(\mathcal{TN}) = \{[TN]_\simeq \mid (TN, \varphi) \in \mathcal{PRC}_\dagger(\mathcal{TN})\}$.
- *$\mathcal{TN}$ и $\mathcal{TN}'$* $\star$-языково эквивалентны относительно стратегии $\dagger$ *(обозначается $\mathcal{TN} \equiv_\star^\dagger \mathcal{TN}'$), если $Trace_\star^\dagger(\mathcal{TN}) = Trace_\star^\dagger(\mathcal{TN}')$.*

Пример 7. *Рассмотрим НВСП$_{\text{сл}}$ $\mathcal{TN}_2$ и $\mathcal{TN}_3$ на рисунке 3. Видно, что $Lang^A(\mathcal{TN}_2) = Lang^A(\mathcal{TN}_3)$. Значит, согласно определению 8, $\mathcal{TN}_2$ и $\mathcal{TN}_3$ интерливингово языково эквивалентны относительно устойчиво атомарной стратегии сброса часов, т.е. $\mathcal{TN}_2 \cong_i^A \mathcal{TN}_3$. С другой стороны, $\mathcal{TN}_2 \not\cong_i^I \mathcal{TN}_3$, поскольку, например, $2b2a2 \in$*

$Lang^I(\mathcal{TN}_2)$, но $2b2a2 \notin Lang^I(\mathcal{TN}_3)$. Аналогично, $\mathcal{TN}_2 \equiv_i^A \mathcal{TN}_3$ и $\mathcal{TN}_2 \not\equiv_i^I \mathcal{TN}_3$.

Покажем, что НВСП$_{сл}$ $\mathcal{TN}_2$ и $\mathcal{TN}_3$ на рисунке 3 не являются n-языково эквивалентны ни относительно промежуточной, ни относительно устойчиво атомарной стратегии сброса часов, т.е. $\mathcal{TN}_2 \not\equiv_n^I \mathcal{TN}_3$ и $\mathcal{TN}_2 \not\equiv_n^A \mathcal{TN}_3$. Действительно, у $\mathcal{TN}_3$ найдется временной процесс из начального состояния, где события действий a и b будут параллельными, тогда как у $\mathcal{TN}_2$ подобного процесса не существует.

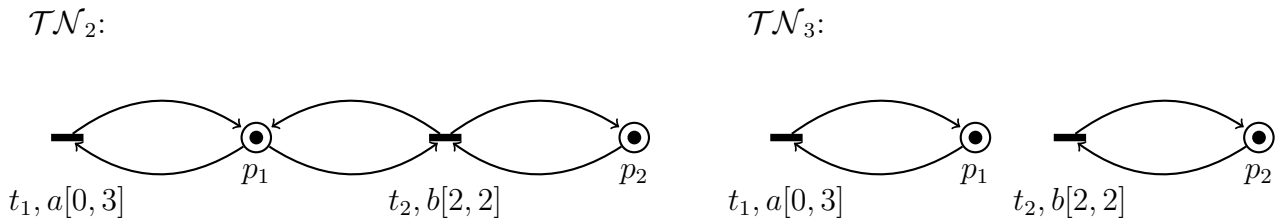

Рис. 3: Непрерывно-временные сети Петри $\mathcal{TN}_2$ и $\mathcal{TN}_3$

Бисимуляционные эквивалентности, в отличие от языковых, учитывают момент недетерминированного выбора между несколькими расширениями процесса (ветвления). Сначала рассмотрим обычные бисимуляционные эквивалентности. Начнем с определения интерливинговых бисимуляционных эквивалентностей, основанных на состояниях НВСП$_{сл}$.

Определение 10. *НВСП$_{сл}$ $\mathcal{TN}$ и $\mathcal{TN}'$, помеченные над Act, являются* интерливингово бисимуляционно эквивалентными относительно стратегии † *(обозначается $\mathcal{TN} \sim_i^\dagger \mathcal{TN}'$), если существует отношение (*бисимуляция*) $R \subseteq \mathcal{RS}_\dagger(\mathcal{TN}) \times \mathcal{RS}_\dagger(\mathcal{TN}')$ такое, что $(S_0, S_0') \in R$ и для всех $(S, S') \in R$ выполняется:*

(1) если $S \xrightarrow{x}_\dagger \widetilde{S}$, где $x \in Act \cup \mathbb{N}$, тогда существует пара $(\widetilde{S}, \widetilde{S}') \in R$ такая, что $S' \xrightarrow{x}_\dagger \widetilde{S}'$;

(2) как пункт (1), но роли $\mathcal{TN}$ и $\mathcal{TN}'$ меняются.

Теперь определим бисимуляционные эквивалентности, основанные на временных процессах НВСП$_{сл}$.

Определение 11. *Пусть $\star \in \{i, n\}$. НВСП$_{сл}$ $\mathcal{TN}$ и $\mathcal{TN}'$, помеченные над Act, являются* $\star$-бисимуляционно эквивалентными относительно стратегии † *(обозначается $\mathcal{TN} \leftrightarrows_\star^\dagger \mathcal{TN}'$), если существует отображение (бисимуляция) $R \subseteq \mathcal{PRC}_\dagger(\mathcal{TN}) \times \mathcal{PRC}_\dagger(\mathcal{TN}')$ такое, что $(\pi_0, \pi_0') \in R$ и для всех $(\pi, \pi') \in R$ выполняется:*

(1) если $\pi \xrightarrow{\widehat{\pi}} \widetilde{\pi}$ для $\widetilde{\pi} \in \mathcal{PRC}_\dagger(\mathcal{TN})$ и

- *$\widetilde{\pi}$ — расширение π на время $\theta \in \mathbb{N}$ (действие $a \in Act$), в случае $\star = i$,*

тогда существует пара $(\widetilde{\pi}, \widetilde{\pi}') \in R$ *такая, что* $\pi' \xrightarrow{\widehat{\pi}'} \widetilde{\pi}'$ *и*

- $\widetilde{\pi}'$ *— расширение* π' *на время* θ *(действие* a*), в случае* $\star = i$*;*
- $\widehat{TN} \simeq \widehat{TN}'$*, в случае* $\star = n$*;*

(2) как пункт (1), но роли $\mathcal{TN}$ *и* $\mathcal{TN}'$ *меняются.*

Пример 8. *Рассмотрим НВСП*$_{сл}$ $\mathcal{TN}_4$ *и* $\mathcal{TN}_5$ *на рисунке 4. Как можно видеть,* $\mathcal{TN}_4 \equiv_n^\dagger \mathcal{TN}_5$ $(\dagger \in \{I, A\})$*, согласно определению 9.*

Покажем, что $\mathcal{TN}_4 \not\equiv_i^\dagger \mathcal{TN}_5$*. Предположим, что это не так, т.е. существует некоторая бисимуляция* R*, соответствующая определению 11. Тогда* $(\pi_0, \pi_0') \in R$*, согласно данному определению.*

Рассмотрим временной процесс π_1' *НВСП*$_{сл}$ $\mathcal{TN}_5$*, который соответствует срабатыванию перехода* t_2 *и является расширением* π_0' *на событие* a*, т.е.* $\pi_0' \xrightarrow{a} \pi_1'$*. Согласно определению 11, для* π_0 *должно существовать аналогичный временной процесс* π_1 *— расширение* π_0 *на событие* a *и* $(\pi_1, \pi_1') \in R$*. Это означает, что* π_1 *соответствует срабатыванию перехода* t_1 *НВСП*$_{сл}$ $\mathcal{TN}_4$*.*

Процесс π_1 *может быть расширен до процесса* π_2 *в результате действия* a *(повторное срабатывание перехода* t_1 *НВСП*$_{сл}$ $\mathcal{TN}_4$*), т.е.* $\pi_1 \xrightarrow{a} \pi_2$*. Значит, по определению 11, должна существовать пара* $(\pi_2, \pi_2') \in R$ *такая, что* $\pi_1' \xrightarrow{a} \pi_2'$*. Однако, процесс* π_1' *не имеет расширений в результате действия* a*. Полученное противоречие доказывает, что* $\mathcal{TN}_4 \not\equiv_i^\dagger \mathcal{TN}_5$*.*

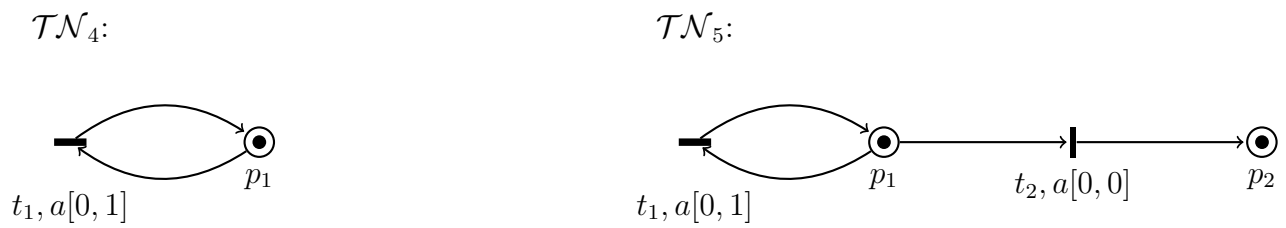

Рис. 4: Непрерывно-временные сети Петри $\mathcal{TN}_4$ и $\mathcal{TN}_5$

В работе [2] показано, что интерливинговая языковая эквивалентность и интерливинговая бисимуляционная эквивалентность, основанные на пробегах и состояниях НВСП$_{сл}$, совпадают соответственно с интерливинговой языковой и интерливинговой бисимуляционной эквивалентностями, основанными на временных процессах. Данный факт подтверждает корректность предложенных определений эквивалентностей, основанных на временных процессах.

Утверждение 4. *Для НВСП$_{сл}$ $\mathcal{TN}$, $\mathcal{TN}'$, помеченных на одном и том же множестве действий, выполняется:*

- $\mathcal{TN} \cong_i^\dagger \mathcal{TN}' \iff \mathcal{TN} \equiv_i^\dagger \mathcal{TN}'$;
- $\mathcal{TN} \sim_i^\dagger \mathcal{TN}' \iff \mathcal{TN} \leftrightarrows_i^\dagger \mathcal{TN}'$.

Доказательство. Аналогично доказательству теорем 1 и 2 работы [2]. □

Далее рассмотрим эквивалентности, которые учитывают предыдущее функционирование системы, ту часть процесса, которая привела из начального состояния в текущее. Для произвольных НВСП$_{сл}$ $\mathcal{TN}$ и $\mathcal{TN}'$ определим множество изоморфизмов ВПСС их временных процессов следующим образом: $\mathcal{F}_\dagger(\mathcal{TN}, \mathcal{TN}') = \{f : TN \simeq TN' \mid (TN, \varphi) \in \mathcal{PRC}_\dagger(\mathcal{TN}), (TN', \varphi') \in \mathcal{PRC}_\dagger(\mathcal{TN}')\}$. Через f_0 будем обозначать изоморфизм между ВПСС начальных временных процессов.

Определение 12. *НВСП$_{сл}$ $\mathcal{TN}$ и $\mathcal{TN}'$, помеченные над Act, являются n-бисимуляционно эквивалентными с сохранением истории относительно стратегии $\dagger$ (обозначается $\mathcal{TN} \leftrightarrows_n^{h\dagger} \mathcal{TN}'$), если существует отображение (бисимуляция) $R \subseteq \mathcal{PRC}_\dagger(\mathcal{TN}) \times \mathcal{PRC}_\dagger(\mathcal{TN}') \times \mathcal{F}_\dagger(\mathcal{TN}, \mathcal{TN}')$ такое, что $(\pi_0, \pi_0', f_0) \in R$ и для всех $(\pi, \pi', f) \in R$ выполняется:*

(1) $f : TN \simeq TN'$;

(2) если $\pi \to \widetilde{\pi}$ для $\widetilde{\pi} \in \mathcal{PRC}_\dagger(\mathcal{TN})$, тогда существует тройка $(\widetilde{\pi}, \widetilde{\pi}', \widetilde{f}) \in R$ такая, что $\pi' \to \widetilde{\pi}'$ и $f \subseteq \widetilde{f}$;

(3) как пункт (2), но роли $\mathcal{TN}$ и $\mathcal{TN}'$ меняются.

Пример 9. *Рассмотрим НВСП$_{сл}$ $\mathcal{TN}_6$ и $\mathcal{TN}_7$ на рисунке 5. Покажем, что $\mathcal{TN}_6 \not\leftrightarrows_n^{h\dagger} \mathcal{TN}_7$ для $\dagger \in \{I, A\}$. Предположим обратное, т.е. что существует бисимуляция R из определения 12. Тогда $(\pi_0, \pi_0', f_0) \in R$.*

Пусть π_1 — расширение π_0, соответствующее срабатыванию перехода t_4, т.е. $\pi_0 \to \pi_1$. Значит, по определению 12, существует тройка $(\pi_1, \pi_1', f_1) \in R$ такая, что $\pi_0' \to \pi_1'$ и $f_1 : TN_1 \simeq TN_1'$. В этом случае, π_1' соответствует срабатывание перехода t_1 НВСП$_{сл}$ $\mathcal{TN}_7$.

Далее, π_1' может быть расширено до временного процесса π_2' в результате срабатывания перехода t_2 с временной задержкой 2. Согласно определению 12, должна существовать тройка $(\pi_2, \pi_2', f_2) \in R$ такая, что $\pi_1 \to \pi_2$ и $f_2 : TN_2 \simeq TN_2'$. Однако, такого расширения временного процесса π_1 с изоморфизмом ВПСС у $\mathcal{TN}_6$ не существует, что

приводит к противоречию. Следовательно, $\mathcal{TN}_6 \not\approx_n^{h\dagger} \mathcal{TN}_7$.

С другой стороны, данные сети n*-бисимуляционно эквивалентны относительно промежуточной и устойчиво атомарной стратегий сброса часов.*

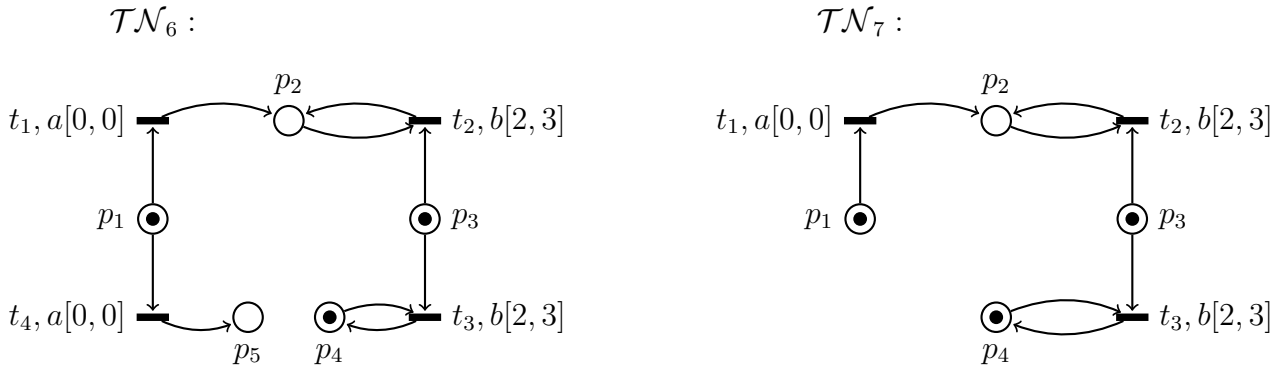

Рис. 5: Непрерывно-временные сети Петри $\mathcal{TN}_6$ и $\mathcal{TN}_7$

В работе [2] рассмотрен ряд других эквивалентностей относительно устойчиво атомарной стратегии сброса часов, изучены взаимосвязи между ними. Заметим, что представленные в [2] определения аналогично можно рассмотреть относительно промежуточной стратегии, а доказательство иерархии данных эквивалентностей будет в точности повторять рассуждения, используемые для устойчиво атомарной стратегии.

Утверждение 5. *Пусть* $\mathcal{TN}$, $\mathcal{TN}'$ — *НВСП*$_{сл}$, *помеченные на одном и том же множестве действий. Для эквивалентностей* R *и* $\widetilde{R}$ *из множества* $\{ \cong_i^\dagger , \equiv_i^\dagger , \equiv_n^\dagger , \sim_i^\dagger , \leftrightarrows_i^\dagger , \leftrightarrows_n^\dagger , \leftrightarrows_n^{h\dagger} \}$

$$\mathcal{TN}\ R\ \mathcal{TN}' \Rightarrow \mathcal{TN}\ \widetilde{R}\ \mathcal{TN}'$$

тогда и только тогда, когда в графе на Рисунке 6 существует путь от R *к* $\widetilde{R}$.

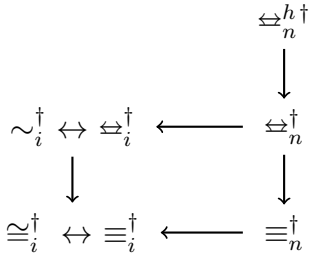

Рис. 6: Иерархия эквивалентностей

Доказательство. Является обобщением доказательства иерархии эквивалентностей работы [2] (Следствие 1). □

5. Сравнение эквивалентностей относительно стратегий сброса часов

В этом разделе будет исследовано, как изменение стратегии сброса часов НВСП$_{\text{сл}}$ влияет на эквивалентность рассматриваемых сетей. Сначала исследуем данный вопрос на подклассе НВСП$_{\text{сл}}$, исключающем места, которые являются одновременно входными и выходными для одного перехода.

Определение 13. *Пусть* $\mathcal{TN} = ((P,\ T,\ F,\ M_0,\ L),\ D)$ *— НВСП$_{сл}$.*

- *Пара* $(p,t) \in P \times T$ — петля*, если* $p \in {}^\bullet t \cap t^\bullet$.
- $\mathcal{TN}$ *называется* простой *НВСП$_{сл}$ (без петель), если* ${}^\bullet t \cap t^\bullet = \emptyset$ *для всех* $t \in T$.

Рассмотрим пример, сравнивающий поведение НВСП$_{\text{сл}}$ с петлей относительно разных стратегий сброса часов.

Пример 10. *Сеть* $\mathcal{TN}_8$ *на рисунке 1 имеет петлю, образованную местом* p_1 *и переходом* t_1*. При использовании промежуточной стратегии сброса часов, срабатывание перехода* t_1 *будет каждый раз сбрасывать часы данного перехода. Как следствие, последовательность* $\sigma = 1\ t_1\ 1\ t_1\ 1$ *будет являться пробегом из начального состояния относительно промежуточной стратегии сброса часов, т.е.* $\sigma \in \mathcal{RUN}_I(\mathcal{TN})$*. Напротив, при устойчиво атомарной стратегии часы перехода* t_1 *никогда не будут сбрасываться. Значит,* $\sigma \notin \mathcal{RUN}_A(\mathcal{TN})$*, поскольку второе срабатывание* t_1 *в данной последовательности невозможно (значение часов перехода* 2 *выйдет за пределы временного интервала* $[0,1]$ *перехода). Следовательно,* $\mathcal{RUN}_I(\mathcal{TN}) \neq \mathcal{RUN}_A(\mathcal{TN})$*, т.е. поведение НВСП$_{сл}$ для двух стратегий сброса часов будет отличаться.*

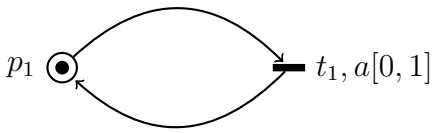

Рис. 7: Непрерывно-временная сеть Петри $\mathcal{TN}_8$

Докажем, что промежуточная и устойчиво атомарная стратегии сброса часов порождают одинаковое поведение для простых НВСП$_{\text{сл}}$.

Теорема 1. *Пусть* $\mathcal{TN}$ *— простая НВСП$_{сл}$. Тогда:*

(а) $\mathcal{RUN}_I(\mathcal{TN}) = \mathcal{RUN}_A(\mathcal{TN})$,

(б) $\mathcal{PRC}_I(\mathcal{TN}) = \mathcal{PRC}_A(\mathcal{TN})$.

Доказательство. Сначала покажем, что для НВСП$_{\text{сл}}$ $\mathcal{TN}$ срабатывание перехода относительно промежуточной и устойчиво атомарной стратегий сброса часов приводит к одинаковым состояниям.

Замечание 1. Для состояния $S = (M, I)$ и перехода $t \in Fi(S)$ НВСП$_{\text{сл}}$ $\mathcal{TN}$ выполняется $(M, I) \xrightarrow{t}_I (M', I') \iff (M, I) \xrightarrow{t}_A (M', I')$.

Доказательство замечания. Поскольку, $t \in Fi(S)$, то $(M, I) \xrightarrow{t}_I (M', I')$ и $(M, I) \xrightarrow{t}_A (M', I'')$, где $M' = (M \backslash {}^\bullet t) \cup t^\bullet$, $I'(t') = \begin{cases} 0, & \text{если } \uparrow enabled_I(t', M, t), \\ I(t'), & \text{иначе;} \end{cases}$ и $I''(t') = \begin{cases} 0, & \text{если } \uparrow enabled_A(t', M, t), \\ I(t'), & \text{иначе;} \end{cases}$ для всех $t' \in En(M')$, по определению 2. Покажем, что $\forall t' \in En(M') : I'(t) = I''(t)$, т.е. $\uparrow enabled_I(t', M, t) = \uparrow enabled_A(t', M, t)$. Рассмотрим произвольное $t' \in En(M)$. По определению 2, $\uparrow enabled_I(t', M, t) = t' \notin En(M \backslash {}^\bullet t) \wedge t' \in En(M')$. Если $t' \notin En(M \backslash {}^\bullet t)$, то $t' \notin En(M)$ или ${}^\bullet t' \cap {}^\bullet t \neq \emptyset$. Значит, $\uparrow enabled_I(t', M, t) = (t' \notin En(M) \vee {}^\bullet t' \cap {}^\bullet t \neq \emptyset) \wedge t' \in En(M') = (t' \notin En(M) \wedge t' \in En(M')) \vee ({}^\bullet t' \cap {}^\bullet t \neq \emptyset \wedge t' \in En(M'))$. Предположим, ${}^\bullet t' \cap {}^\bullet t \neq \emptyset \wedge t' \in En(M')$ — истина. Тогда ${}^\bullet t' \subseteq M' = (M \backslash {}^\bullet t) \cup t^\bullet$, так как $t' \in En(M')$. Кроме того, из ${}^\bullet t' \cap {}^\bullet t \neq \emptyset$ следует, что $((M \backslash {}^\bullet t) \cup t^\bullet) \cap {}^\bullet t = ((M \backslash {}^\bullet t) \cap {}^\bullet t) \cup (t^\bullet \cap {}^\bullet t) = t^\bullet \cap {}^\bullet t \neq \emptyset$. Получили противоречие с тем, что ${}^\bullet t \cap t^\bullet = \emptyset$, поскольку $\mathcal{TN}$ — простая. Следовательно, ${}^\bullet t' \cap {}^\bullet t \neq \emptyset \wedge t' \in En(M')$ — ложь. Получаем, $\uparrow enabled_I(t', M, t) = t' \notin En(M) \wedge t' \in En(M') = \uparrow enabled_A(t', M, t)$, по определению 2. □

Доказательство теоремы.

(а) Рассмотрим произвольный пробег $\sigma = \theta_0 t_1 \theta_1 \dots t_n \theta_n \in \mathcal{RUN}_I(\mathcal{TN})$ с последовательностью состояний $(M_0, I_0) \xrightarrow{\theta_0} (M_0, \widetilde{I}_0) \xrightarrow{t_1}_I (M_1, I_1) \xrightarrow{\theta_1} (M_1, \widetilde{I}_1) \dots (M_{n-1}, \widetilde{I}_{n-1}) \xrightarrow{t_n}_I (M_n, I_n) \xrightarrow{\theta_n} (M_n, \widetilde{I}_n)$. По определению 2 и замечанию 1, $(M_0, I_0) \xrightarrow{\theta_0} (M_0, \widetilde{I}_0) \xrightarrow{t_1}_A (M_1, I_1) \xrightarrow{\theta_1} (M_1, \widetilde{I}_1) \dots (M_{n-1}, \widetilde{I}_{n-1}) \xrightarrow{t_n}_A (M_n, I_n) \xrightarrow{\theta_n} (M_n, \widetilde{I}_n)$, т.е. $\sigma \in \mathcal{RUN}_A(\mathcal{TN})$. Значит, $\mathcal{RUN}_I(\mathcal{TN}) \subseteq \mathcal{RUN}_A(\mathcal{TN})$. Аналогично доказывается, что $\mathcal{RUN}_A(\mathcal{TN}) \subseteq \mathcal{RUN}_I(\mathcal{TN})$. Следовательно, $\mathcal{RUN}_I(\mathcal{TN}) = \mathcal{RUN}_A(\mathcal{TN})$.

(б) Рассмотрим произвольный временной процесс $\pi = (TN, \varphi) \in \mathcal{PRC}_I(\mathcal{TN}) = \mathcal{PRC}_I(\mathcal{TN}, S_0)$, где $TN = (N, \tau)$ и $N = (B, E, G, l)$. Тогда для каждого сечения $C \in \mathcal{RC}(TN)$ и события $e \in Fi(C)$ выполняется $\mathbf{Clock}^I_{\varphi, S_0}(C, \varphi(e)) \in D(\varphi(e))$, по определению 7. Покажем, что $\pi = (TN, \varphi) \in \mathcal{PRC}_A(\mathcal{TN})$, т.е. $\mathbf{Clock}^A_{\varphi, S_0}(C, \varphi(e)) \in D(\varphi(e))$ для всех $C \in \mathcal{RC}(TN)$ и $e \in Fi(C)$.

Рассмотрим произвольные $C \in \mathcal{RC}(TN)$ и $e \in Fi(C)$. Тогда $C \xRightarrow{e} C'$ для $C' =$

$(C\backslash {}^\bullet e) \cup e^\bullet \in \mathcal{RC}(TN)$. Так как ${}^\bullet N \preceq C$, $C' \preceq N^\bullet$ и ${}^\bullet N, C, C', N^\bullet \in \mathcal{RC}(TN)$, то, по утверждению 1, существует $\omega = e_1 \dots e_n \in \mathcal{GRF}(TN)$ такой, что ${}^\bullet N = C_0 \stackrel{e_1}{\Longrightarrow} C_1 \dots (C_{i-1} = C) \stackrel{(e_i = e)}{\Longrightarrow} C_i = C' \dots C_{n-1} \stackrel{e_n}{\Longrightarrow} C_n = N^\bullet$ для $n > 0$ и $0 < i \leq n$. По утверждению 3, последовательность $Run_\varphi(TN, \omega) = \tau(C_0)\varphi(e_1)\tau(C_1)\dots\varphi(e_n)\tau(C_n) \in \mathcal{RUN}_I(\pi) \subseteq \mathcal{RUN}_I(\mathcal{TN})$ — пробег, т.е. имеет место последовательность состояний $(M_0, I_0) \stackrel{\tau(C_0)}{\longrightarrow} (M_0, \widetilde{I}_0) \stackrel{\varphi(e_1)}{\longrightarrow}_I (M_1, I_1) \stackrel{\tau(C_1)}{\longrightarrow} (M_1, \widetilde{I}_1) \dots (M_{n-1}, \widetilde{I}_{n-1}) \stackrel{\varphi(e_n)}{\longrightarrow}_I (M_n, I_n) \stackrel{\tau(C_n)}{\longrightarrow} (M_n, \widetilde{I}_n)$. Согласно определению 2 и замечанию 1, $(M_0, I_0) \stackrel{\tau(C_0)}{\longrightarrow} (M_0, \widetilde{I}_0) \stackrel{\varphi(e_1)}{\longrightarrow}_A (M_1, I_1) \stackrel{\tau(C_1)}{\longrightarrow} (M_1, \widetilde{I}_1) \dots (M_{n-1}, \widetilde{I}_{n-1}) \stackrel{\varphi(e_i)}{\longrightarrow}_A (M_n, I_n) \stackrel{\tau(C_n)}{\longrightarrow} (M_n, \widetilde{I}_n)$. Кроме того, $\varphi(e_i) \in En(M_{i-1})$. Значит, $\mathbf{Clock}^A_{\varphi,S_0}(C_{i-1}, \varphi(e_i)) = \widetilde{I}_{i-1}(\varphi(e_i)) = \mathbf{Clock}^I_{\varphi,S_0}(C_{i-1}, \varphi(e_i))$, благодаря утверждению 2. Следовательно, $\mathbf{Clock}^A_{\varphi,S_0}(C, \varphi(e)) = \mathbf{Clock}^A_{\varphi,S_0}(C_{i-1}, \varphi(e_i)) = \mathbf{Clock}^I_{\varphi,S_0}(C_{i-1}, \varphi(e_i)) = \mathbf{Clock}^I_{\varphi,S_0}(C, \varphi(e)) \in D(\varphi(e))$. Получаем, $\mathcal{PRC}_I(\mathcal{TN}) \subseteq \mathcal{PRC}_A(\mathcal{TN})$. Аналогично доказывается, что $\mathcal{PRC}_A(\mathcal{TN}) \subseteq \mathcal{PRC}_I(\mathcal{TN})$, т.е. $\mathcal{PRC}_I(\mathcal{TN}) = \mathcal{PRC}_A(\mathcal{TN})$.

□

Из данной теоремы следует совпадение эквивалентностей относительно промежуточной и устойчиво атомарной стратегий сброса часов для класса простых НВСП$_{\text{сл}}$.

Теорема 2. *Пусть $\mathcal{TN}$, $\mathcal{TN}'$ — простые НВСП$_{сл}$, помеченные на одном и том же множестве действий. Тогда $\mathcal{TN}\ R^I\ \mathcal{TN}' \iff \mathcal{TN}\ R^A\ \mathcal{TN}'$, где $R \in \{\ \cong_i\ ,\ \equiv_i\ ,\ \equiv_n\ ,\ \sim_i\ ,\ \leftrightarrows_i\ ,\ \leftrightarrows_n\ ,\ \leftrightarrows^h_n\ \}$.*

Доказательство. Следует из теоремы 1 и определений 8-12. □

Покажем, что в общем случае из эквивалентности НВСП$_{\text{сл}}$ относительно одной стратегии сброса часов не будет следовать эквивалентность данных сетей относительно другой стратегии сброса часов.

Теорема 3. *Пусть $\mathcal{TN}$, $\mathcal{TN}'$ — НВСП$_{сл}$, помеченные на одном и том же множестве действий. Тогда:*

(а) $\mathcal{TN}\ R\ \mathcal{TN}' \not\Rightarrow \mathcal{TN}\ \widetilde{R}\ \mathcal{TN}'$,

(б) $\mathcal{TN}\ \widetilde{R}\ \mathcal{TN}' \not\Rightarrow \mathcal{TN}\ R\ \mathcal{TN}'$,

где $R \in \{\ \cong^I_i\ ,\ \equiv^I_i\ ,\ \equiv^I_n\ ,\ \sim^I_i\ ,\ \leftrightarrows^I_i\ ,\ \leftrightarrows^I_n\ ,\ \leftrightarrows^{h\,I}_n\ \}$ и $\widetilde{R} \in \{\ \cong^A_i\ ,\ \equiv^A_i\ ,\ \equiv^A_n\ ,\ \sim^A_i\ ,\ \leftrightarrows^A_i\ ,\ \leftrightarrows^A_n\ ,\ \leftrightarrows^{h\,A}_n\ \}$.

Доказательство. (а) Рассмотрим НВСП$_{\text{сл}}$ $\mathcal{TN}_9$ и $\mathcal{TN}_{10}$ на рисунке 8. Видно, что $\mathcal{TN}_9 \leftrightarrows^{h\,I}_n \mathcal{TN}_{10}$. Покажем, что $\mathcal{TN}_9 \not\cong^A_i \mathcal{TN}_{10}$. Поскольку срабатывание пере-

хода t_1 в НВСП$_{\text{сл}}$ $\mathcal{TN}_9$ при устойчиво атомарной стратегии сброса часов не сбрасывает часы этого перехода, то действие a может повторяться только в пределах временного интервала $[0,1]$. Как следствие, слова «1a1a1» языка $Lang^A(\mathcal{TN}_{10})$ не будет в языке $Lang^A(\mathcal{TN}_9)$, т.е. $Lang^A(\mathcal{TN}_9) \neq Lang^A(\mathcal{TN}_{10})$. Следовательно, $\mathcal{TN}_9 \not\cong_i^A \mathcal{TN}_{10}$, по определению 8. Значит, $\mathcal{TN} \leftrightarrows_n^{h\,I} \mathcal{TN}' \not\Rightarrow \mathcal{TN} \cong_i^A \mathcal{TN}'$. Так как $\mathcal{TN} \leftrightarrows_n^{h\,I} \mathcal{TN}' \Rightarrow \mathcal{TN}\ R\ \mathcal{TN}'$ и $\mathcal{TN}\ \widetilde{R}\ \mathcal{TN}' \Rightarrow \mathcal{TN} \cong_i^A \mathcal{TN}'$, согласно утверждению 5, то $\mathcal{TN}\ R\ \mathcal{TN}' \not\Rightarrow \mathcal{TN}\ \widetilde{R}\ \mathcal{TN}'$.

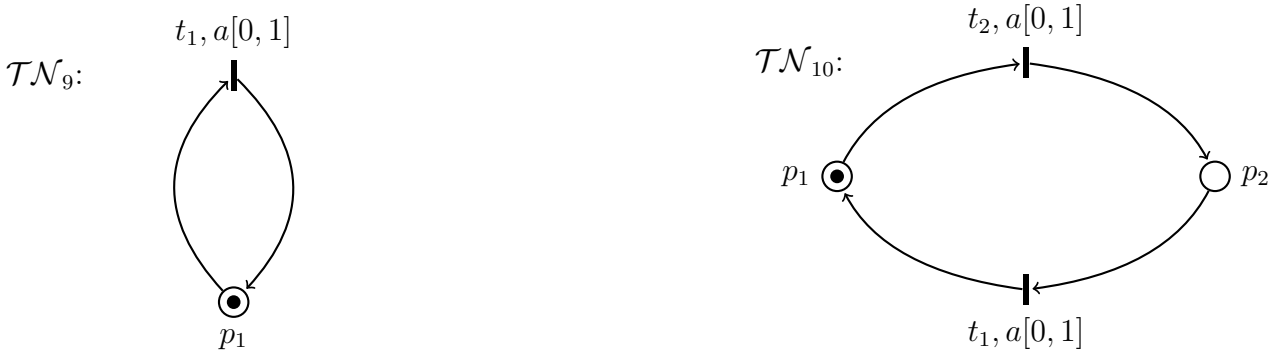

Рис. 8: Непрерывно-временные сети Петри $\mathcal{TN}_9$ и $\mathcal{TN}_{10}$

(б) Рассмотрим НВСП$_{\text{сл}}$ $\mathcal{TN}_{11}$ и $\mathcal{TN}_{12}$ на Рисунке 9. Видно, что в случае устойчиво атомарной стратегии сброса часов $\mathcal{TN}_{11} \leftrightarrows_n^{h\,A} \mathcal{TN}_{12}$.

Покажем, что $\mathcal{TN}_{11} \not\cong_i^I \mathcal{TN}_{12}$. Поскольку срабатывание перехода t_2 в НВСП$_{\text{сл}}$ $\mathcal{TN}_{11}$ при промежуточной стратегии сбрасывает часы перехода t_1, то действие a может следовать за действием b. Подобного поведения нет в НВСП$_{\text{сл}}$ $\mathcal{TN}_{12}$. Как следствие, слова «1b0a1» языка $Lang^I(\mathcal{TN}_{11})$ не будет в языке $Lang^I(\mathcal{TN}_{12})$, т.е. $Lang^I(\mathcal{TN}_{11}) \neq Lang^I(\mathcal{TN}_{12})$. Следовательно, $\mathcal{TN}_{11} \not\cong_i^I \mathcal{TN}_{12}$, по определению 8. Значит, $\mathcal{TN} \leftrightarrows_n^{h\,A} \mathcal{TN}' \not\Rightarrow \mathcal{TN} \cong_i^I \mathcal{TN}'$. Так как $\mathcal{TN} \leftrightarrows_n^{h\,A} \mathcal{TN}' \Rightarrow \mathcal{TN}\ \widetilde{R}\ \mathcal{TN}'$ и $\mathcal{TN}\ R\ \mathcal{TN}' \Rightarrow \mathcal{TN} \cong_i^I \mathcal{TN}'$, согласно утверждению 5, то $\mathcal{TN}\ \widetilde{R}\ \mathcal{TN}' \not\Rightarrow \mathcal{TN}\ R\ \mathcal{TN}'$.

□

6. Заключение

В данной работе было показано, что выбор между промежуточной и устойчиво атомарной стратегиями сброса часов оказывает большое влияние на эквивалентность НВСП$_{\text{сл}}$. При исследовании рассматривались эквивалентности в двух дихотомиях: *«интерливинг — истинный параллелизм»* и *«линейное — ветвящееся время»*. Для первой дихотомии ис-

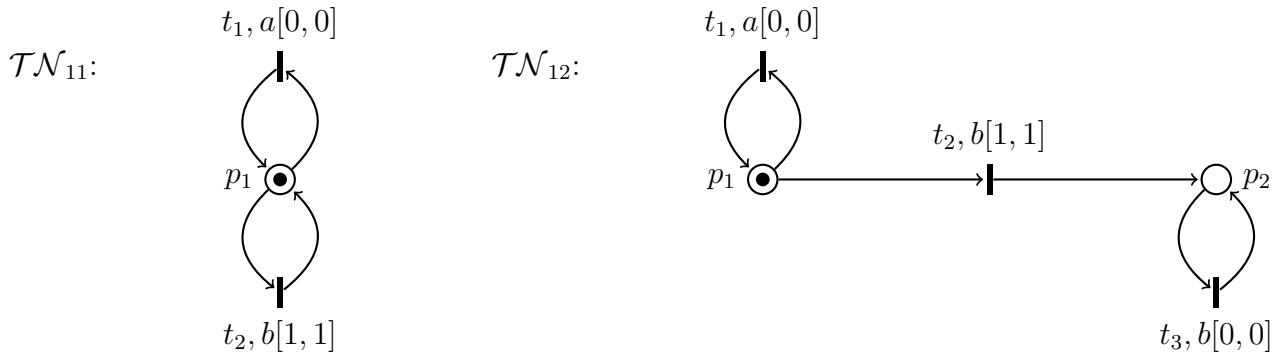

Рис. 9: Непрерывно-временные сети Петри $\mathcal{TN}_{11}$ и $\mathcal{TN}_{12}$

следовались интерливинговая и процессно-сетевая семантика, в то время как для второй дихотомии использовались языковая, обычная бисимуляционная и сохраняющая историю бисимуляционная семантики. Относительно каждой из стратегий сброса часов интерливинговая языковая эквивалентность является наиболее слабой, а процессно-сетевая сохраняющая историю бисимуляционная эквивалентность выступает как наиболее сильная. Основанные на данном факте примеры показали, что из произвольной эквивалентности двух $\text{НВСП}_{\text{сл}}$ относительно одной из стратегий сброса часов не будет следовать ни одна из рассмотренных эквивалентностей тех же $\text{НВСП}_{\text{сл}}$ относительно другой стратегии. Однако, был найден подкласс $\text{НВСП}_{\text{сл}}$, для которых поведение будет схожим независимо от выбранной стратегии. Исключение из структуры $\text{НВСП}_{\text{сл}}$ петель, т.е. переходов, множество входных и выходных мест которых пересекается, ведет к совпадению множеств пробегов из начального состояния относительно обеих стратегий сброса часов. Кроме того, было доказано, что в этом случае множество временных процессов при смене стратегий также остается одинаковым. Это приводит к совпадению эквивалентностей $\text{НВСП}_{\text{сл}}$ относительно промежуточной и устойчиво атомарной стратегий.

В качестве дальнейшей работы планируется разработка и исследование «истинно-параллельных» семантик и эквивалентностей в терминах реверсивных сетей Петри [5] и их временных расширений, позволяющих моделировать вычисления как в прямом, так и в обратном направлениях.

Список литературы

1. Зубарев, А. Ю. Сравнение языковых и бисимуляционных эквивалентностей непрерывно-временных сетей Петри со слабой временной стратегией / А. Ю. Зубарев // Проблемы информатики. — 2022. — № 4. — С. 5—27.

2. Зубарев, А. Ю. Иерархия эквивалентностей непрерывно-временных сетей Петри со слабой временной стратегией / А. Ю. Зубарев // Проблемы информатики. — 2024. — № 1. С. 5—40.
3. Тарасюк, И. В. Эквивалентности для поведенческого анализа параллельных и распределенных вычислительных систем / И. В. Тарасюк. — Новосибирск : Академическое издательство “Гео”, 2007.
4. Aura, T. A causal semantics for time Petri nets / T. Aura, J. Lilius // Theoretical Computer Science. — 2000. — Vol. 243, no. 1/2. — P. 409—447.
5. Barylska, K. Reversing Transitions in Bounded Petri Nets / K. Barylska, E. Erofeev, M. Koutny L. Mikulski, M. Piatkowski // Fundamenta Informaticae. — 2018. — Vol. 157 — P. 341—357.
6. Be’rard, B. Comparison of the expressiveness of timed automata and time Petri nets / B. Be’rard // Formal Modeling and Analysis of Timed Systems – FORMATS 2005 / ed. by P. Pettersson, W. Yi. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005. — P. 211—225.
7. Boyer, M. Comparison of the Expressiveness of Arc, Place and Transition Time Petri Nets / M. Boyer, O. Roux // Petri Nets and Other Models of Concurrency – ICATPN 2007 / ed. by J. Kleijn, A. Yakovlev. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2007. — P. 63–82
8. Reynier, P.-A. Weak time Petri nets strike back! / P.-A. Reynier, A. Sangnier // CONCUR 2009 – Concurrency Theory / ed. by M. Bravetti, G. Zavattaro. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. — P. 557–571.
9. Virbitskaite, I. True concurrent equivalences in time Petri nets / I. Virbitskaite, D. Bushin, E. Best // Fundamenta Informaticae. — 2016. — Vol. 149, no. 4. — P. 401—418.
10. Virbitskaite, I. B. ‘True Concurrency’ Semantics for Time Petri Nets with Weak Time and Persistent Atomic Policies / I. B. Virbitskaite, A. Y. Zubarev // Programming and Computer Software. — 2021. — Vol. 47, no. 5. — P. 389—401.
11. Zubarev, A. State space reduction for time Petri nets with weak semantics / A. Zubarev // Bulletin of the Novosibirsk Computing Center. Series: Computer Science. — 2019. — No. 43. — P. 39—52.

УДК 519.681.2, 519.681.3

Семантика систем переходов структур событий с отменяемыми событиями при сохранении причинно-следственной зависимости

Грибовская Н.С. (Институт систем информатики СО РАН)

Вирбицкайте И.Б. (Институт систем информатики СО РАН)

Реверсивные (обратимые) вычисления — это новая парадигма, которая расширяет традиционную организацию вычислений возможностью их выполнения в обратном направлении. Структуры событий, являющиеся одной из центральных моделей параллельных недетерминированных процессов, широко используются для установления взаимосвязей между различными моделями параллелизма. В литературе выделяются два метода разработки семантики систем переходов для моделей структур событий. В первом методе состояния рассматриваются как конфигурации (наборы событий, которые уже произошли в структуре), а переходы между состояниями строятся, начиная с начальной конфигурации и расширяя конфигурации за счёт произошедших событий. Во втором методе состояния понимаются как остаточные структуры (фрагменты модели, которые остались после удаления из неё уже произошедших и конфликтующих событий), а переходы создаются по мере построения остаточных структур.

Данная статья посвящена построению и исследованию взаимосвязей двух типов систем переходов, построенных из реверсивных первичных структур событий с отменяемыми событиями и с сохранением причинно-следственной зависимости между событиями. Устанавливается факт существования бисимуляционной эквивалентности между такими системами переходов в контексте истинно-параллельной (шаговой) семантики рассматриваемой модели структур событий.

Ключевые слова: структуры событий, отменяемые события, системы переходов, частичный порядок, истинный параллелизм, бисимуляция

1. Введение

Реверсивные (обратимые) вычисления, широко изучаемые в последние годы, представляют собой нетрадиционную форму вычислений, которые могут быть выполнены как в прямом, так и в обратном направлении. Любая последовательность действий, выполняемых системой, впоследствии может быть отменена по какой-либо причине (например, в случае ошибки), что позволяет восстановить предыдущие состояния системы, как если

бы отмененные действия вообще не выполнялись. Обратимые вычисления привлекают интерес благодаря своим применениям во многих областях, включая: анализ программ и отладку [25], абстракции программирования для надежных систем [12, 27], моделирование биохимических процессов [22], проектирование аппаратуры и квантовые вычисления [13] и т.д.

Несмотря на то, что реверсивные вычисления в параллельных/распределенных системах имеют много перспективных применений, такие вычисления сопряжены со многими концептуальными и техническими трудностями. Одна из наиболее существенных проблем касается методов, которые следует использовать для обеспечения корректных вычислений в обратном направлении. Недавно было выявлено несколько различных способов отмены вычислений. Наиболее заметными из них являются обратное отслеживание, обратимость с учетом причинно-следственной зависимости (ПСЗ) между действиями системы, а также обратимость без учёта такой зависимости. Эти способы отличаются порядком выполнения действий в обратном направлении. Под обратным отслеживанием обычно понимается способность отменять действия, строго соблюдая порядок, в котором они были выполнены. Обратимость с учетом ПСЗ в параллельных системах означает, что действия, которые являются причиной для других действий, могут быть отменены только после того, как эти другие действия будут отменены прежде, и что независимые друг от друга (параллельные) действия могут быть отменены в произвольном порядке. Обратимость без учета ПСЗ, наиболее характерная для биохимических систем, не сохраняет ПСЗ. Взаимодействие между обратимостью и параллелизмом широко изучалось в различных моделях: системах параллельной перезаписи [1], клеточных автоматах [20], алгебраических исчислениях параллельных процессов [12, 24], сетях Петри [6, 14, 29], структурах событий [28, 32, 34], мембранных системах [33] и т.д.

Структуры событий — одна из центральных моделей в теории параллелизма. Первоначально структуры событий были предложены Винскелем в его диссертации [35] и рассматривались как промежуточная абстракция между доменами Скотта (т.е. денотационной моделью) и сетями Петри (т.е. операционной моделью). По сути, структуры событий представляют собой наборы событий моделируемой системы, некоторые из которых конфликтуют друг с другом (т.е. выполнение одного события запрещает выполнение других событий), в то время как другие события находятся в отношении ПСЗ (т.е. событие не может быть выполнено, если ему не предшествовали другие события), и события, ко-

торые не связаны ни отношением ПСЗ, ни отношением конфликта, рассматриваются как независимые (параллельные). Первичные структуры событий — это модель параллельных недетерминированных процессов, в которой отношение ПСЗ представляется частичным порядком, а конфликт между событиями наследуется по ПСЗ.

Известно, что установление взаимосвязей между моделями систем переходов и структур событий способствует изучению и решению различных задач анализа и верификации поведения параллельных систем. Различают два метода обеспечения семантики систем переходов для структур событий. В первом методе (см. [2, 3, 15, 19, 21, 35, 36] среди прочего) состояния рассматриваются как наборы уже произошедших событий, называемые конфигурациями, а переходы между состояниями строятся, начиная с начальной конфигурации и расширяя конфигурации за счет произошедших событий. Во втором, более 'структурно-композиционном' методе (см., в частности, [5, 9, 11, 21, 23]), состояния понимаются как остаточные части структуры, получаемые посредством удаления уже произошедших и конфликтующих событий, при этом начальное состояние — это исходная структура события, переходы создаются по мере построения остаточных структур. В литературе системы переходов, основанные на конфигурациях, активно применяются для определения семантики и эквивалентностей параллельных моделей, а системы перехода, основанные на остаточных структурах, преимущественно используются для обеспечения операционной семантики алгебраических исчислений параллельных процессов и демонстрации согласованности операционной и денотационной семантик. Взаимосвязи между этими двумя типами систем переходов впервые изучались в статье [26] для наиболее простой модели — первичных структур событий, а затем в статьях [7] и [8] — для широкого спектра моделей структур событий с асимметричным и симметричным конфликтом соответственно.

Реверсивные структуры событий расширяют модели структур событий, чтобы представлять обратимые параллельные процессы, способные отменять выполненные действия, позволяя конфигурациям изменяться путем не только добавления произошедших событий, но их удаления. В работах [32, 34] Филлипс и др. определили учитывающие и неучитывающие ПСЗ модели первичных [32], асимметричных [32, 34] и обобщенных [34] структур событий и показали соответствие между их конфигурациями и конфигурациями традиционных (без отменяемых событий) моделей. В статье [17] Граверсен и др. представили категории различных классов реверсивных структур событий, включая упомянутые выше, и построили функторы между этими категориями. В работе [4] Обер и Кристес-

ку разработали в терминах структур конфигураций ‘истинно’ параллельную семантику реверсивного расширения CCS, RCCS (без автопараллелизма, автоконфликта и рекурсии). В статье [18] Граверсен и др. построили категорию реверсивных структур событий с расслоением и симметричным конфликтом и использовали подкатегорию, учитывающую отношение ПСЗ между событиями, для моделирования семантики другого реверсивного расширения CCS, CCSK. В работе [31] логика идентификатора событий (EIL) была введена для расширения логики Хеннесси-Милнера с помощью обратных модальностей. Оказалось, что в контексте стабильных структур конфигураций EIL-эквивалентность соответствует эквивалентности наследуемой бисимуляции с сохранением истории (hereditary history-preserving bisimulation). В статье [30] впервые были изучены взаимосвязи различных поведенческих эквивалентностей в реверсивных моделях.

Цель данной статьи — построение и исследование взаимосвязей двух типов систем переходов, основанных на конфигурациях и на остаточных структурах, для первичных структур событий, обогащенных отменяемыми событиями с сохранением отношения ПСЗ между событиями при выполнении модели в обратном направлении, что может помочь в разработке и сравнении операционной и денатоционной семантик алгебраических исчислений реверсивных параллельных процессов.

Статья построена следующим образом. В разделе 2 рассматриваются синтаксис реверсивных первичных структур событий и их истинно-параллельная (шаговая) семантика в терминах конфигураций. В разделе 3 определяется оператор удаления событий, который используется для построения остаточных структур, и демонстрируются его корректность и композиционный свойства. В разделе 4 разрабатываются два типа семантик систем переходов для реверсивных первичных структур событий с сохранением ПСЗ и устанавливаются взаимосвязи между этими семантиками. В разделе 5 приводятся заключительные замечания. В приложении представлены доказательства лемм и утверждений.

2. Реверсивные первичные структуры событий

В этом разделе сначала определяется модель первичных структур событий (ПСС) [35], а затем формулируется понятие реверсивных первичных структур событий (РПСС) [32], а также рассматриваются их шаговая, основанная на множестве параллельно происходящих событий, семантика и их свойства.

Для формального описания поведения параллельных/распределенных систем исполь-

зуются модели структур событий, в которых элементы поведения представлены событиями. Существуют различные способы определения отношений между событиями. В ПСС причинно-следственная зависимость между событиями задается частичным порядком, а несовместимость событий определяется отношением конфликта. Два события, которые не находятся ни в причинно-следственной зависимости, ни в конфликте, считаются независимыми (параллельными).

Определение 1. Первичная структура событий (ПСС) — *это кортеж* $\mathcal{E} = (E, <, \sharp, C_0)$, *где*

- E — *счетное множество* событий;
- $< \subseteq E \times E$ — *иррефлексивный частичный порядок (*причинно-следственная зависимость *(ПСЗ)), удовлетворяющий* принципу конечности причин*: для каждого* $e \in E$ *верно, что* $\lfloor e \rfloor_< = \{e' \in E \mid e' < e\}$ — *конечное множество. Пусть* $\leq = < \cup \{(e, e) \mid e \in E\}$.
- $\sharp \subseteq E \times E$ — *иррефлексивное симметричное отношение* конфликта*, удовлетворяющее* принципу наследования конфликта*: для всех* $e, e', e'' \in E$ *верно, что если* $e < e'$ *и* $e \sharp e''$*, то* $e' \sharp e''$;
- $C_0 = \emptyset$ — начальная конфигурация.

Итак, ПСС — это модель, основанная на событиях параллельных и недетерминированных процессов, в которой события рассматриваются как атомарные, неделимые и мгновенные действия, некоторые из которых могут происходить только после других (т.е. существует ПСЗ, представленная иррефлексивным частичным порядком $<$ между событиями) и некоторые из которых не могут происходить вместе (т.е. между событиями существует конфликт $\sharp$). Кроме того, необходимы принцип конечности причин и принцип наследования конфликта.

ПСС выполняется по мере того, как происходят события, начиная с начального состояния и переходя из одного состояния в другое. Состояние в ПСС называется конфигурацией и представляет собой множество уже произошедших событий. Подмножество $X \subseteq E$ событий является *лево-замкнутым относительно* $<$, если для всех $e \in X$ считается, что $\lfloor e \rfloor_< \subseteq X$; является *бесконфликтным*, если для всех $e, e' \in X$ верно, что $\neg(e \sharp e')$. Подмножество $C \subseteq E$ является *конфигурацией* в ПСС $\mathcal{E}$, если C конечно, лево-замкнуто относительно $<$ и бесконфликтно.

Реверсивные первичные структуры событий (РПСС) [32, 34] основаны на более сла-

бой форме ПСС, поскольку принцип наследования конфликта может не сохраняться при добавлении обратимости. Кроме того, в РПСС некоторые события рассматриваются как отменяемые, а также добавляются два отношения между событиями: обратная ПСЗ и отношение предотвращения отмены. Первое отношение — это ПСЗ в обратном направлении, т.е. чтобы отменить событие в текущей конфигурации, в ней должны присутствовать события, от которых это событие обратимо зависит. Второе отношение, напротив, идентифицирует те события, присутствие которых в текущей конфигурации предотвращает отмену события.

Определение 2. Реверсивная первичная структура событий (РПСС) — *это кортеж* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$, *где*

- E — *счетное множество* событий;
- $\sharp \subseteq E \times E$ — *иррефлексивное и симметричное отношение* конфликта;
- $<\subseteq E \times E$ — *иррефлексивный частичный порядок (*причинно-следственная зависимость*), удовлетворяющая условию: для каждого* $e \in E$ *верно, что* $\lfloor e \rfloor_<$ — *конечное и бесконфликтное множество, а также для каждых* $e, e' \in E$ *верно, что если* $e < e'$, *то* $\neg(e \sharp e')$;
- $F \subseteq E$ — *множество* отменяемых событий, *обозначаемое как* $\underline{F} = \{\underline{u} \mid u \in F\}$;
- $\prec\subseteq E \times \underline{F}$ — обратная причинно-следственная зависимость *такая, что для каждого* $u \in F$ *верно, что* $u \prec \underline{u}$ *и* $\llcorner u \lrcorner_\prec = \{e \mid e \prec \underline{u}\}$ — *конечное и бесконфликтное множество;*
- $\rhd \subseteq E \times \underline{F}$ — отношение предотвращения отмены *такое, что для каждого* $u \in F$ *верно: если* $e \prec \underline{u}$, *то* $\neg(e \rhd \underline{u})$;
- $\ll$ — *транзитивная* устойчивая причинно-следственная зависимость *такая, что* $e \ll e'$, *если и только если* $e < e'$ *и* $e \in F \Rightarrow e' \rhd \underline{e}$. *Отношение конфликта* $\sharp$ наследуется по устойчивой ПСЗ $\ll$: *если* $e \sharp e' \ll e''$, *то* $e \sharp e''$;
- $C_0 \subseteq E$ — начальная конфигурация.

Заметим, что в РПСС начальная конфигурация необязательно должна быть пустым множеством, она может содержать события, некоторые из которых впоследствии могут быть отменены. Несложно проверить, что любая ПСС также является РПСС, имеющая $F = \emptyset$ и $C_0 = \emptyset$. Тогда любое понятие, определенное для РПСС, применимо и к ПСС.

При графическом представлении РПСС используются следующие обозначения. Неотменяемые события рисуются квадратиками, а отменяемые — кружочками. Отношение

ПСЗ изображается сплошными стрелками (за исключением тех, которые выводятся по транзитивности), отношение обратной ПСЗ — пунктирными стрелками, а также явно показываются отношения конфликта и предотвращения отмены. События, относящиеся к начальной конфигурации, окрашены в темно-серый цвет. Очевидно, что если начальная конфигурация представляет собой пустое множество, то никакие события не окрашены в темно-серый цвет.

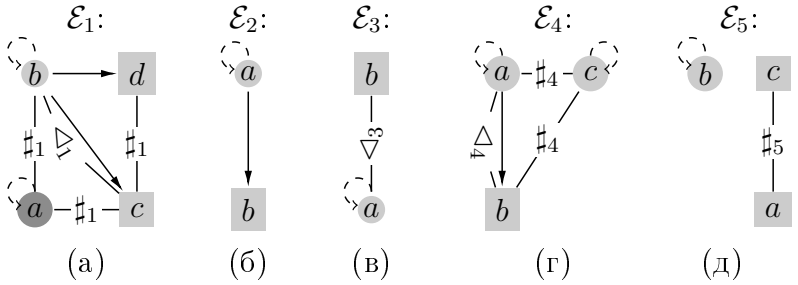

Рисунок 1: Примеры РПСС

Пример 1. *Рассмотрим показанную на рис. 1(а) структуру* $\mathcal{E}_1$ *с компонентами:* $E_1 = \{a, b, c, d\}$; $<_1 = \{(b, c), (b, d)\}$; $\sharp_1 = \{(a, b),\ (b, a),\ (a, c),\ (c, a),\ (c, d),\ (d, c)\}$; $F_1 = \{a, b\}$; $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$; $\rhd_1 = \{(c, \underline{b})\}$; $C_0^1 = \{a\}$. *Легко убедиться в том, что компоненты структуры* $\mathcal{E}_1$ *удовлетворяют соответствующим пунктам определения 2. В частности, видим, что для каждого события* $e \in E_1$ *(для каждого события* $u \in F_1$*) множество* $\lfloor e \rfloor_{<_1}$ *(*$\lfloor u \rfloor_{\prec_1}$*) конечно и бесконфликтно;* $<_1 = \{(b, c), (b, d)\}$ *и* $(b, c), (b, d) \notin \sharp_1$*, а также* $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$ *и* $(a, \underline{a}), (b, \underline{b}) \notin \rhd_1$*. Заметим, что начальная конфигурация* $C_0^1 = \{a\}$ *не является пустым множеством. Также, пары* (b, c) *и* (b, d) *принадлежат отношению* $<_1$*, тогда как отношение* $\rhd_1$ *содержит только пару* $(c, \underline{b})$*. Значит, только пара* (b, c) *находится в отношении* $\ll_1$ *устойчивой ПСЗ. Легко видеть, что конфликт* $\sharp_1$ *наследуется по* $\ll_1$*. Таким образом, структура* $\mathcal{E}_1$ *является РПСС.* ◇

РПСС выполняется по мере того, как происходят и/или отменяются события, начиная с начальной конфигурации и переходя от одной конфигурации к другой. Множества событий, которые происходят/отменяются при таком переходе, называются шагами в РПСС. Достижимые конфигурации — это подмножества событий, которые могут быть получены из начальной конфигурации путем выполнения шагов.

Определение 3. *Пусть* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$ *— РПСС и* $C \subseteq E$ *— конечное и бесконфликтное множество. Тогда*

- *для множеств* $A \subseteq E$ *и* $B \subseteq F$ *будем говорить, что шаг* $A \cup \underline{B}$ возможен из C, *если выполнены следующие условия:*

  а) $A \cap C = \emptyset$, $B \subseteq C$, $(C \cup A)$ *— конечное и бесконфликтное множество;*

  б) $\forall e \in A, \forall e' \in E : \; e' < e \Rightarrow e' \in (C \setminus B)$;

  в) $\forall e \in B, \forall e' \in E : \; e' \prec \underline{e} \Rightarrow e' \in (C \setminus (B \setminus \{e\}))$;

  г) $\forall e \in B, \forall e' \in E : \; e' \rhd \underline{e} \Rightarrow e' \notin (C \cup A)$.

  Если шаг $A \cup \underline{B}$ *возможен из* C, *то* $C \xrightarrow{A \cup \underline{B}} C' = (C \setminus B) \cup A$.

- C — (достижимая из начальной конфигурации C_0) конфигурация в $\mathcal{E}$, *если для всех* $i = 1, \ldots, n$ $(n \geq 0)$ *существуют множества* $A_i \subseteq E$ *и* $B_i \subseteq F$ *такие, что* $C_{i-1} \xrightarrow{A_i \cup \underline{B_i}} C_i$ *и* $C_n = C$. *Множество конфигураций в* $\mathcal{E}$ *обозначается как* $Conf(\mathcal{E})$.

Пример 2. *Сначала вспомним пример 1, где представлена РПСС* $\mathcal{E}_1$ *с компонентами:* $E_1 = \{a, b, c, d\}$; $<_1 = \{(b, c), (b, d)\}$; $\sharp_1 = \{(a, b), (b, a), (a, c), (c, a), (c, d), (d, c)\}$; $F_1 = \{a, b\}$; $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$; $\rhd_1 = \{(c, \underline{b})\}$; $C_0^1 = \{a\}$.

Рассмотрим возможные шаги в РПСС $\mathcal{E}_1$. *Поскольку пара* (a, b) *(*(a, c)*) принадлежит отношению* $\sharp_1$, *события* a *и* b *(*a *и* c*) не могут присутствовать вместе в какой-либо конфигурации. Кроме того, так как пары* (b, c) *и* (b, d) *находятся в ПСЗ* $<_1$, *то события* c *и* d *не могут произойти, пока событие* b *не произойдет. Поэтому в начальную конфигурацию* $C_0^1 = \{a\}$ *невозможно добавить какое-либо событие, несмотря на то, что события* a *и* d *независимы (параллельны). При этом, из конфигурации* $\{a\}$ *возможен шаг* $(\emptyset \cup \{\underline{a}\})$, *поскольку* $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$, *т.е. единственное событие, необходимое для отмены события* a, *— само это событие, и верно, что* $(\cdot, \underline{a}) \notin \rhd_1$, *где* $\cdot \in \{b, c, d\}$, *т.е. в РПСС* $\mathcal{E}_1$ *нет событий, которые могли бы препятствовать отмене события* a. *Далее из конфигурации* $\emptyset$ *возможен шаг* $(\{a\} \cup \emptyset)$, *вновь получая конфигурацию* $\{a\}$, *так как событие* a *не имеет предшественников по ПСЗ. Кроме того, поскольку событие* b *также не имеет предшественников по ПСЗ и пары* (b, c) *и* (b, d) *принадлежат отношению* $<_1$, *получаем следующие переходы:* $\emptyset \xrightarrow{(\{b\} \cup \emptyset)} \{b\} \xrightarrow{(\{c\} \cup \emptyset)} \{b, c\}$ *и* $\emptyset \xrightarrow{(\{b\} \cup \emptyset)} \{b\} \xrightarrow{(\{d\} \cup \emptyset)} \{b, d\}$. *Заметим, что конфигурация* $\{b, c\}$ *(*$\{b, d\}$*) не может быть расширена событием* d *(*c*), в силу того, что события* c *и* d *конфликтуют. Благодаря тому, что имеем* $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$, *т.е. единственное событие, необходимое для отмены события* b, *— само это событие, и* $\rhd_1 = \{(c, \underline{b})\}$, *т.е. событие* b *может быть отменено, только если событие* c *еще не произошло, следующие переходы возможны:* $\{b\} \xrightarrow{(\emptyset \cup \{\underline{b}\})} \emptyset$ *и* $\{b, d\} \xrightarrow{(\emptyset \cup \{\underline{b}\})} \{d\}$. *Так как события* a *и* d *не конфликтуют и событие* a *не имеет предшественников по ПСЗ, можем*

перейти из конфигурации $\{d\}$ *в конфигурацию* $\{a, d\}$ *посредством шага* $(\{a\} \cup \emptyset)$. *Поскольку единственное событие, необходимое для отмены события* a, — *само это событие и в РПСС* $\mathcal{E}_1$ *нет событий, которые могли бы препятствовать отмене события* a, *то можем отменить* a *в конфигурации* $\{a, d\}$, *получая вновь конфигурацию* $\{d\}$. *Таким образом, конфигурации РПСС* $\mathcal{E}_1$ — *это множества* $\emptyset$, $\{a\}$, $\{b\}$, $\{d\}$, $\{b, c\}$, $\{b, d\}$, $\{a, d\}$.

Построим конфигурации в изображенной на рис. 1(б) РПСС $\mathcal{E}_2$ *с компонентами:* $E_2 = \{a, b\}$; $<_2 = \{(a, b)\}$; $\sharp_2 = \emptyset$; $F_2 = \{a\}$; $\prec_2 = \{(a, \underline{a})\}$; $\rhd_2 = \emptyset$; $C_0^2 = \emptyset$. *Поскольку единственная пара* (a, b) *принадлежит ПСЗ* $<_1$, *то событие* a *не имеет предшественников по ПСЗ и является предшественником по ПСЗ для события* b, *т.е. событие* a *может произойти первым и только после этого может произойти событие* b. *Тогда получаем следующее:* $C_0^2 = \emptyset \overset{(\{a\} \cup \emptyset)}{\rightarrow} \{a\} \overset{(\{b\} \cup \emptyset)}{\rightarrow} \{a, b\}$. *Событие* a *может быть отменено в конфигурациях* $\{a\}$ *и* $\{a, b\}$, *так как имеем* $\prec_2 = \{(a, \underline{a})\}$ *и* $\rhd_2 = \emptyset$. *Следовательно, получаем такие конфигурации в* $\mathcal{E}_2$: $\emptyset$, $\{a\}$, $\{b\}$, $\{a, b\}$.

Рассмотрим на рис. 1(в) РПСС $\mathcal{E}_3$ *с компонентами:* $E_3 = \{a, b\}$; $<_3 = \emptyset$; $\sharp_3 = \emptyset$; $F_3 = \{a\}$; $\prec_3 = \{(a, \underline{a})\}$; $\rhd_3 = \{(b, \underline{a})\}$; $C_0^2 = \emptyset$. *Поскольку отношения* $<_3$ *и* $\sharp_3$ *пусты, события* a *и* b *параллельны и поэтому они могут приходить в любом порядке. Тогда получаем следующее:* $\emptyset \overset{(\{a\} \cup \emptyset)}{\rightarrow} \{a\} \overset{(\{b\} \cup \emptyset)}{\rightarrow} \{a, b\}$ *и* $\emptyset \overset{(\{b\} \cup \emptyset)}{\rightarrow} \{b\} \overset{(\{a\} \cup \emptyset)}{\rightarrow} \{a, b\}$. *Так как верно, что* $b \rhd_3 \underline{a}$, *событие* b *предотвращает отмену события* a, *т.е.* a *не может быть отменено, если* b *присутствует в конфигурации. Тогда шаг* $(\emptyset \cup \{\underline{a}\})$ *возможен из конфигурации* $\{a\}$, *приводя в конфигурацию* $\emptyset$, *и не возможен из конфигурации* $\{a, b\}$. *Конфигурации в* $\mathcal{E}_3$ — *это множества* $\emptyset$, $\{a\}$, $\{b\}$, $\{a, b\}$.

Проверим, как выполняется показанная на рис. 1(г) РПСС $\mathcal{E}_4$ *с компонентами:* $E_4 = \{a, b, c\}$; $<_4 = \{(a, b)\}$; $\sharp_4 = \{(a, c), (c, a), (b, c), (c, b)\}$; $F_4 = \{a, c\}$; $\prec_4 = \{(a, \underline{a}), (c, \underline{c})\}$; $\rhd_4 = \{(b, \underline{a})\}$; $C_0^4 = \emptyset$. *Так как верно* $<_4 = \{(a, b)\}$, *событие* a (c) *не имеет предшественником по ПСЗ, т.е.* a (c) *может произойти первым. Также, событие* a *является предшественником по ПСЗ для события* b, *т.е.* b *не может произойти до того, как* a *произойдет. События* a (b) *и* c *конфликтуют, т.е. события* a (b) *и* c *не могут находится вместе в какой-либо конфигурации. Нетрудно понять, что отношение* $\sharp_4$ *наследуется по отношению* $<_4$. *Единственное событие, необходимое для отмены события* a (c), — *само это событие, так как* $\prec_4 = \{(a, \underline{a}), (c, \underline{c})\}$. *Кроме того, если оба события* a *и* b *присутствуют в*

конфигурации, то событие a не может быть отменено, поскольку имеем $b \rhd_4 \underline{a}$. Поэтому конфигурациями в РПСС $\mathcal{E}_4$ являются множества $\emptyset$, $\{a\}$, $\{a,b\}$, $\{c\}$.

В конце построим конфигурации изображенной на рис. 1(д) РПСС $\mathcal{E}_5$ с компонентами: $E_5 = \{a,b,c\}$; $<_5 = \emptyset$; $\sharp_5 = \{(a,c),(c,a)\}$; $F_5 = \{b\}$; $\prec_5 = \{(b,\underline{b})\}$; $\rhd_5 = \emptyset$; $C_0^4 = \emptyset$. Видим, что a и c конфликтуют, т.е. они не могут вместе присутствовать в какой-либо конфигурации. Поскольку отношение $<_5$ пусто и отношение $\sharp_5$ содержит только пары (a,c) и (c,a), события a и b (b и c) параллельны и поэтому могут происходить в любом порядке. Следовательно, получаем следующее: $\emptyset \overset{(\{a\}\cup\emptyset)}{\to} \{a\} \overset{(\{b\}\cup\emptyset)}{\to} \{a,b\}$ и $\emptyset \overset{(\{b\}\cup\emptyset)}{\to} \{b\} \overset{(\{a\}\cup\emptyset)}{\to} \{a,b\}$ ($\emptyset \overset{(\{b\}\cup\emptyset)}{\to} \{b\} \overset{(\{c\}\cup\emptyset)}{\to} \{b,c\}$ и $\emptyset \overset{(\{c\}\cup\emptyset)}{\to} \{c\} \overset{(\{b\}\cup\emptyset)}{\to} \{b,c\}$). Единственное событие, необходимое для отмены события b, — само это событие, поскольку $\prec_5 = \{(b,\underline{b})\}$. Так как $\rhd_5 = \emptyset$, событие b может быть отменено в любой конфигурации, где оно присутствует. Конфигурации в $\mathcal{E}_5$ — это множества $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a,b\}$, $\{b,c\}$. ◇

РПСС способны моделировать такую особенность реверсивных вычислений, как согласованность отношения обратимости с отношением ПСЗ: событие может быть отменено при условии, что все его последователи по ПСЗ были отменены. Это понятие реверсивности естественно для надежных параллельных систем, поскольку при возникновении ошибки система пытается корректно вернуться к предыдущему состоянию.

Определение 4. *РПСС $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$ называется* сохраняющей причинно-следственную зависимость *(со свойством СПСЗ), если для событий $e \in E$ и $u \in F$ верно: (а) $e \prec \underline{u} \Leftrightarrow e = u$; (б) $e \rhd \underline{u} \Leftrightarrow u < e$.*

Неформально говоря, в РПСС со свойством СПСЗ отменяемые события могут быть удалены из конфигурации только, если они сами присутствуют в этой конфигурации, а также отменяемые предшественники по ПСЗ могут быть удалены только, если их последователи по ПСЗ не присутствуют в этой конфигурации.

Пример 3. *Вновь рассмотрим пример 2. Легко видеть, что РПСС $\mathcal{E}_1$, $\mathcal{E}_2$ и $\mathcal{E}_3$ не обладают свойством СПСЗ, поскольку их отношения ПСЗ и предотвращения отмены различаются, что противоречит пункту (б) в определении 4, хотя пункт (а) выполнен; тогда как РПСС $\mathcal{E}_4$ и $\mathcal{E}_5$ имеют свойство СПСЗ, так как их отношения ПСЗ, обратной ПСЗ и предотвращения отмены удовлетворяют обоим требованиям определения 4.* ◇

Следующая лемма говорит об особенности конфигураций в РПСС со свойсвом СПСЗ, которые остаются конечными, лево-замкнутыми относительно ПСЗ и бесконфликтными

при выполнении РПСС.

Лемма 1. *Пусть $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$ — РПСС со свойством СПСЗ и $C \in Conf(\mathcal{E})$. Тогда C конечно, лево-замкнуто относительно $<$ и бесконфликтно, если C_0 конечно, лево-замкнуто относительно $<$ и бесконфликтно.*

Доказательство: См. приложение. □

Приведенный ниже пример поясняет приведенную выше лемму.

Пример 4. *Вспомним необладающую свойством СПСЗ РПСС $\mathcal{E}_1$ ($E_1 = \{a, b, c, d\}$; $<_1 = \{(b,c),(b,d)\}$; $C_0^1 = \{a\}$) из примеров 1–3. Знаем, что $\{d\}$ и $\{a,d\}$ — конфигурации в $\mathcal{E}_1$. Видим, что эти конфигурации не являются лево-замкнутыми относительно $<_1$, хотя начальная конфигурация лево-замкнута относительно $<_1$. То же верно и для конфигурации $\{b\}$ в необладающей свойством СПСЗ РПСС $\mathcal{E}_2$ ($E_2 = \{a,b\}$; $<_2 = \{(a,b)\}$; $C_0^2 = \emptyset$) из примеров 2–3.*

Легко проверить, что в обладающих свойством СПСЗ РПСС $\mathcal{E}_4$ и $\mathcal{E}_5$ из примеров 2–3 все конфигурации, включая начальную, конечны, лево-замкнуты относительно $<$ и бесконфликтны. ◇

Класс РПСС, обладающих свойством СПСЗ и имеющих в качестве начальной конфигурации конечное, лево-замкнутое относительно $<$ и бесконфликтное множество событий, обозначим через $c\mathcal{RPES}$.

3. Остаточные структуры

Оператор удаления для РПСС, основанный на удалении из РПСС уже произошедших неотменяемых событий вместе с предшествующими им по ПСЗ и конфликтующими с ними событиями, необходим для построения остаточных структур.

Введем определение оператора удаления для РПСС, используя понятие конфигурации.

Определение 5. *Пусть $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0) \in c\mathcal{RPES}$ и $C \in Conf(\mathcal{E})$.* Остаточная структура $\mathcal{E}\backslash C$ *для РПСС $\mathcal{E}$ после конфигурации C при применении оператора $\backslash$ удаления определяется следующим образом:* $\mathcal{E} \setminus C = (E', <'=< \cap (E' \times E'), \sharp' = \sharp \cap (E' \times E'), F' = (F \cap E'), \prec'=\prec \cap (E' \times \underline{F'}), \rhd' = \rhd \cap (E' \times \underline{F'}), C_0' = C \cap E'),$ *где* $E' = E \setminus \left(\widetilde{C} \cup \sharp(\widetilde{C})\right)$ *и*

- $\widetilde{C} = \lfloor C \setminus F \rfloor_{\leq} = \{e' \in E \mid \exists e \in (C \setminus F) : \ e' \leq e\}$,
- $\sharp(\widetilde{C}) = \{e' \in E \mid \exists e \in \widetilde{C} \colon e' \sharp e\}$.

Интуитивная интерпретация приведенного выше определения оператора удаления заключается в следующем. Множество событий остаточной структуры $\mathcal{E}\setminus C$ формируется из множества событий исходной РПСС $\mathcal{E}$ посредством удаления событий, присутствующих в конфигурации C и являющихся неотменяемыми, вместе с предшествующими им по ПСЗ и конфликтующими с ними событиями, что приводит к редукции всех отношений и начальной конфигурации. Тогда как отменяемые события сохраняются в остатке, поскольку они могут быть отменены в последующих шагах.

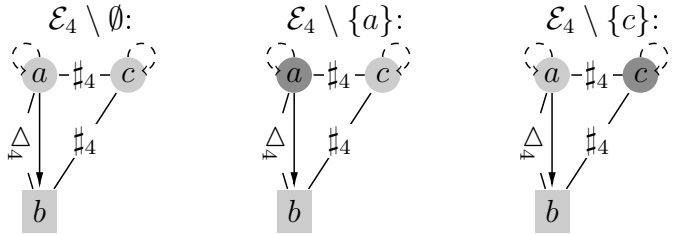

Рисунок 2: Остаточные структуры для $\mathcal{E}_4$

Пример 5. *Рассмотрим обладающую свойством СПСЗ РПСС $\mathcal{E}_4$ из примеров 2–4 с компонентами:* $E_4 = \{a, b, c\}$; $<_4 = \{(a, b)\}$; $\sharp_4 = \{(a, c), (c, a), (b, c), (c, b)\}$; $F_4 = \{a, c\}$; $\prec_4 = \{(a, \underline{a}), (c, \underline{c})\}$; $\rhd_4 = \{(b, \underline{a})\}$; $C_0^4 = \emptyset$. *Знаем, что конфигурации в $\mathcal{E}_4$ — это множества* $\emptyset$, $\{a\}$, $\{c\}$, $\{a, b\}$.

Построим остаточные структуры для РПСС $\mathcal{E}_4$ после её конфигураций:

- $\tilde{\mathcal{E}}_4 = \mathcal{E}_4 \setminus \emptyset = (E_4, <_4, \sharp_4, F_4, \prec_4, \rhd_4, C_0^4)$ *(см. рис. 2(а));*
- $\hat{\mathcal{E}}_4 = \mathcal{E}_4 \setminus \{a\} = (\hat{E} = E_4, \hat{<} = <_4, \hat{\sharp} = \sharp_4, \hat{F} = F_4, \hat{\prec} = \prec_4, \hat{\rhd} = \rhd_4, \hat{C}_0 = \{a\})$, *поскольку* $(\widetilde{\{a\}} \cup \sharp_4(\widetilde{\{a\}})) = \emptyset$ *благодаря тому, что* $a \in F_4$ *(см. рис. 2(б));*
- $\check{\mathcal{E}}_4 = \mathcal{E}_4 \setminus \{c\} = (\check{E} = E_4, \check{<} = <_4, \check{\sharp} = \sharp_4, \check{F} = F_4, \check{\prec} = \prec_4, \check{\rhd} = \rhd_4, \check{C}_0 = \{c\})$, *так как* $(\widetilde{\{c\}} \cup \sharp_4(\widetilde{\{c\}})) = \emptyset$ *в силу того, что* $c \in F_4$ *(см. рис. 2(в));*
- $\grave{\mathcal{E}}_4 = \mathcal{E}_4 \setminus \{a, b\} = (\grave{E} = \emptyset, \grave{<} = \emptyset, \grave{\sharp} = \emptyset, \grave{F} = \emptyset, \grave{\prec} = \emptyset, \grave{\rhd} = \emptyset, \grave{C}_0 = \emptyset)$, *поскольку* $(\widetilde{\{a, b\}} \cup \sharp_4(\widetilde{\{a, b\}})) = \{a, b, c\}$, *по причине того, что* $b \notin F_4$, $a <_4 b$ *и* $a \sharp_4 c$, $b \sharp_4 c$.

Далее рассмотрим обладающую свойством СПСЗ РПСС $\mathcal{E}_5$ из примеров 2–4 с компонентами: $E_5 = \{a, b, c\}$; $<_5 = \emptyset$; $\sharp_5 = \{(a, c), (c, a)\}$; $F_5 = \{b\}$; $\prec_5 = \{(b, \underline{b})\}$; $\rhd_5 = \emptyset$; $C_0^5 = \emptyset$. *Знаем, что конфигурации в $\mathcal{E}_5$ — это множества* $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{b, c\}$.

Построим остаточные структуры для РПСС $\mathcal{E}_5$ после её конфигураций:

- $\tilde{\mathcal{E}}_5 = \mathcal{E}_5 \setminus \emptyset = (E_5, <_5, \sharp_5, F_5, \prec_5, \rhd_5, C_0^5)$ *(см. рис. 3(а));*

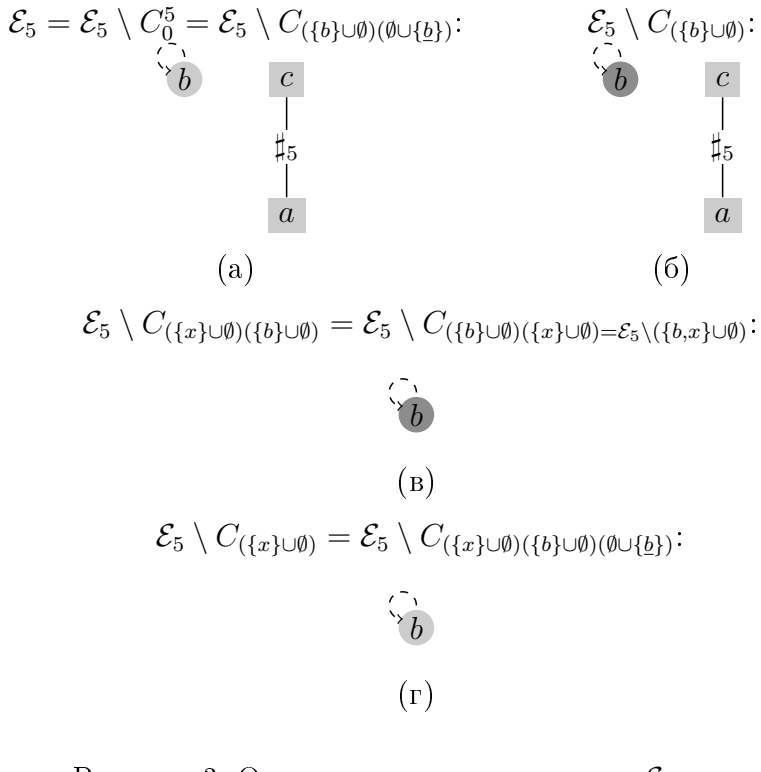

Рисунок 3: Остаточные структуры для $\mathcal{E}_5$

– $\breve{\mathcal{E}}_5 = \mathcal{E}_5 \setminus \{x\} = (\breve{E} = \{b\},\ \breve{<} = \emptyset,\ \breve{\sharp} = \emptyset,\ \breve{F} = \{b\},\ \breve{\prec} = \{(b, \underline{b})\},\ \breve{\rhd} = \emptyset,\ \breve{C}_0 = \emptyset)$, *так как* $\widetilde{\{x\}} = \{x\}$, *поскольку* $\{x\} = \{x\} \setminus (F_5 = \{b\})$, *и* $\sharp_5(\widetilde{\{x\}}) = \{x'\}$, *поскольку* $x \sharp_5 x'$, *где* $x \neq x' \in \{a, c\}$ *(см. рис. 3(б));*

– $\hat{\mathcal{E}}_5 = \mathcal{E}_5 \setminus \{b\} = (\hat{E} = E_5,\ \hat{<} = <_5,\ \hat{\sharp} = \sharp_5,\ \hat{F} = F_5,\ \hat{\prec} = \prec_5,\ \hat{\rhd} = \rhd_5,\ \hat{C}_0 = \{b\})$, *поскольку* $(\widetilde{\{b\}} \cup \sharp_5(\widetilde{\{b\}})) = \emptyset$ *благодаря тому, что* $b \in F_5$ *(см. рис. 3(в));*

– $\grave{\mathcal{E}}_5 = (\mathcal{E}_5 \setminus \{b, x\} = (\grave{E} = \{b\},\ \grave{<} = \emptyset,\ \grave{\sharp} = \emptyset,\ \grave{F} = \{b\},\ \grave{\prec} = \{(b, \underline{b})\},\ \grave{\rhd} = \emptyset,\ \grave{C}_0 = \{b\})$, *потому что* $\widetilde{\{b, x\}} = \{x\}$, *в силу* $\{x\} = \{b, x\} \setminus F_5 = \{b\}$, *и* $\sharp_5(\widetilde{\{b, x\}}) = \{x'\}$ *в силу* $x \sharp_5 x'$, *где* $x \neq x' \in \{a, c\}$ *(см. рис. 3(г)).* ◇

Ниже приведены характерные свойства оператора удаления.

Лемма 2. *Пусть* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0) \in c\mathcal{RPES}$, C *— конфигурация в* $\mathcal{E}$ *и* $\mathcal{E} \setminus C = (E',\ <',\ \sharp',\ F',\ \prec',\ \rhd',\ C_0')$. *Тогда верно:*

(а) $E' \subseteq E$, $F' \subseteq F$, $C_0' \subseteq C$, $\nabla' \subseteq \nabla$ $(\nabla \in \{<, \sharp, \prec, \rhd\})$;

(б) $\mathcal{E} \setminus C \in c\mathcal{RPES}$;

(в) $\widetilde{C} \subseteq C$.

Доказательство: См. приложение. □

Следующие два утверждения демонстрируют композиционные свойства оператора удаления для РПСС, принадлежащей классу $c\mathcal{RPES}$.

Утверждение 1. *Пусть* $\mathcal{E} \in c\mathcal{RPES}$, $\mathcal{E}' = \mathcal{E} \setminus C$ *и* $C'_0 \stackrel{(A \cup \underline{B})}{\rightarrow} C'$ *в* $\mathcal{E}'$. *Тогда верно:* $C \stackrel{(A \cup \underline{B})}{\rightarrow} C''$ *в* $\mathcal{E}$ *и* $\mathcal{E} \setminus C'' = \mathcal{E}' \setminus C'$.

Доказательство: См. приложение. □

Таким образом, оказалось, что в РПСС $\mathcal{E}$ из класса $c\mathcal{RPES}$ с конфигурацией C шаг, возможный из начальной конфигурации в остаточной структуре $\mathcal{E}' = \mathcal{E} \setminus C$ и приводящий в конфигурацию C', также возможен из конфигурации C в исходной РПСС $\mathcal{E}$, приводя в конфигурацию C'', и, кроме того, остаточные структуры $\mathcal{E} \setminus C''$ и $\mathcal{E}' \setminus C'$ совпадают.

Пример 6. *Сначала рассмотрим непринадлежащую классу* $c\mathcal{RPES}$ *РПСС* $\mathcal{E}_3$ *из примеров 2–3 с компонентами:* $E_3 = \{a, b\}$; $<_3 = \emptyset$; $\sharp_3 = \emptyset$; $F_3 = \{a\}$; $\prec_3 = \{(a, \underline{a})\}$; $\rhd_3 = \{(b, \underline{a})\}$; $C_0^3 = \emptyset$. *Как было показано в примере 2,* $\{a, b\}$ *— конфигурация в* $\mathcal{E}_3$. *Построим остаточную структуры для* $\mathcal{E}_3$ *после* $\{a, b\}$ *при применении оператора* $\setminus$ *удаления следующим образом:*

$\mathcal{E}_3 \setminus \{a, b\} = (E' = \{a\}, <' = \emptyset, \sharp' = \emptyset; F' = \{a\}; \prec' = \{(a, \underline{a})\}; \rhd' = \emptyset, C'_0 = \{a\})$, *так как* $\widetilde{\{a, b\}} = \{b\}$, *благодаря* $b \in \{a, b\} \setminus F_3$, *и* $\sharp_3(\widetilde{\{a, b\}}) = \emptyset$, *благодаря* $\sharp_3 = \emptyset$.

Тогда получаем, что выполняется $C'_0 \stackrel{(\emptyset \cup \{\underline{a}\})}{\rightarrow} C'$ *в* $\mathcal{E}_3 \setminus \{a, b\}$, *однако неверно* $\{a, b\} \stackrel{(\emptyset \cup \{\underline{a}\})}{\rightarrow} C'$ *в* $\mathcal{E}_3$.

Используя примеры 2 и 5, нетрудно убедиться в том, что утверждение 1 верно для РПСС $\mathcal{E}_4$ *и* $\mathcal{E}_5$ *из класса* $c\mathcal{RPES}$. ◇

Ниже устанавливается, что в РПСС $\mathcal{E}$ из класса $c\mathcal{RPES}$ шаг, возможный из конфигурации C' и приводящий в конфигурацию C'', также возможен из начальной конфигурации остаточной структуры $\mathcal{E}' = \mathcal{E} \setminus C'$, приводя в конфигурацию C, и, кроме того, остаточные структуры $\mathcal{E} \setminus C''$ и $\mathcal{E}' \setminus C$ совпадают.

Утверждение 2. *Пусть* $\mathcal{E} \in c\mathcal{RPES}$ *и* $C' \stackrel{(A \cup \underline{B})}{\rightarrow} C''$ *в* $\mathcal{E}$. *Тогда верно:* $C'_0 \stackrel{(A \cup \underline{B})}{\rightarrow} C$ *в* $\mathcal{E} \setminus C'$ *и* $\mathcal{E} \setminus C'' = (\mathcal{E} \setminus C') \setminus C$.

Доказательство: См. приложение. □

Пример 7. *Рассмотрим непринадлежащую классу* $c\mathcal{RPES}$ *РПСС* $\mathcal{E}_1$ *из примеров 1–4 с компонентами:* $E_1 = \{a, b, c, d\}$; $<_1 = \{(b, c), (b, d)\}$; $\sharp_1 = \{(a, b), (b, a), (a, c), (c, a), (c, d), (d, c)\}$; $F_1 = \{a, b\}$; $\prec_1 = \{(a, \underline{a}), (b, \underline{b})\}$; $\rhd_1 = \{(c, \underline{b})\}$; $C_0^1 = \{a\}$. *Знаем, что множества* $\{b, d\}$, $\{d\}$ *— конфигурации в РПСС* $\mathcal{E}_1$. *Проверим конфигурацию* $\{b, d\}$. *Используя определение 5, получаем остаточную структуру* $\mathcal{E}_1 \setminus \{b, d\} = (E'_1 = \emptyset, <'_1 = \emptyset, \sharp'_1 = \emptyset, F'_1 = \emptyset, \prec'_1 = \emptyset, \rhd'_1 = \emptyset, C_0'^1 = \emptyset)$, *поскольку* $\widetilde{\{b, d\}} = \{b, d\}$, *благодаря* $\{b, d\} = \lfloor \{b, d\} \setminus F_1 =$

$\{a,b\}\rfloor_{\leq_1=\{(b,c),(b,d)\}}$, *и* $\sharp_1(\widetilde{\{b,d\}}) = \{a,c\}$, *благодаря* $(b,a),(d,c) \in \sharp_1$. *В РПСС* $\mathcal{E}_1$ *из конфигурации* $\{b,d\}$ *возможен шаг* $(\emptyset, \cup\{\underline{b}\})$, *приводящий в конфигурацию* $\{d\}$, *однако такой шаг невозможен из конфигурации* $C_0'^1$ *в РПСС* $\mathcal{E}_1 \setminus \{b,d\}$.

Используя примеры 2 и 5, нетрудно проверить, что утверждение 2 справедливо для РПСС $\mathcal{E}_4$ *и* $\mathcal{E}_5$ *из класса* $c\mathcal{RPES}$. ◇

4. Семантика систем переходов для РПСС

В этом разделе сначала приводятся базовые определения, касающиеся систем переходов, а затем для РПСС $\mathcal{E}$ определяются отображения $TC(\mathcal{E})$ и $TE(\mathcal{E})$, которые строят два различных типа систем переходов.

На основе множества E событий в РПСС определим множество $\mathbb{L} := 2^E$ (множество подмножеств множества E), которое будем использовать как множество меток в системах переходов.

Система переходов $\mathcal{T} = (S, \rightarrow, i)$, помеченная на множестве $\mathbb{L}$ меток, состоит из множества S состояний, отношения перехода $\rightarrow \subseteq S \times \mathbb{L} \times S$ и начального состояния $i \in S$. Две системы переходов, помеченные на множестве $\mathbb{L}$, являются *изоморфными*, если существует биекция между их состояниями, сохраняющая отношение перехода и начальное состояние. Будем говорить, что отношение $R \subseteq S \times S'$ является *бисимуляцией* между системами переходов $\mathcal{T} = (S, \rightarrow, i)$ и $\mathcal{T}' = (S', \rightarrow', i')$, помеченными на $\mathbb{L}$, если $(i,i') \in R$ и для всех пар $(s,s') \in R$ и меток $\lambda \in \mathbb{L}$: если $(s,\lambda,s_1) \in \rightarrow$, то $(s',\lambda,s_1') \in \rightarrow'$ и $(s_1,s_1') \in R$ для некоторого состояния $s_1' \in S'$; а также если $(s',\lambda,s_1') \in \rightarrow'$, то $(s,\lambda,s_1) \in \rightarrow$ и $(s_1,s_1') \in R$ для некоторого состояния $s_1 \in S$. Две системы переходов, помеченные на множестве $\mathbb{L}$, являются *бисимуляционными*, если существует отношение бисимуляции между ними.

Определим понятие системы переходов, имеющей в качестве состояний конфигурации РПСС.

Определение 6. *Для РПСС* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$

$TC(\mathcal{E})$ — конфигурационная система переходов $(Conf(\mathcal{E}), \rightharpoonup, C_0)$, *помеченная на множестве* $\mathbb{L}$ *меток,*

где $C \overset{(A \cup \underline{B})}{\rightharpoonup} C'$ *в* $TC(\mathcal{E}) \iff C \overset{(A \cup \underline{B})}{\rightarrow} C'$ *в* $\mathcal{E}$.

Объясним приведенное выше определение на примере.

Пример 8. *Рассмотрим РПСС* $\mathcal{E}_4 \in c\mathcal{RPES}$ *из примеров 2–6. В примере 2, сказано, что множества* $\emptyset$, $\{a\}$, $\{a,b\}$, $\{c\}$ *— конфигурации в* $\mathcal{E}_4$, *и показаны возможные переходы между этими конфигурациями. Используя определение 6, получаем конфигурационную систему переходов для РПСС* $TC(\mathcal{E}_4)$, *которая показана на рис. 4.*

Перейдем к РПСС $\mathcal{E}_5 \in c\mathcal{RPES}$ *из примеров 2–6. В примере 2 видим, что* $\emptyset$, $\{a\}$, $\{b\}$, $\{c\}$, $\{a,b\}$, $\{b,c\}$ *— конфигурации в* $\mathcal{E}_5$, *там же показаны возможные переходы между этими конфигурациями. Используя определение 6, строим конфигурационную систему переходов для РПСС* $TC(\mathcal{E}_5)$, *которая изображена на рис. 5.* ◇

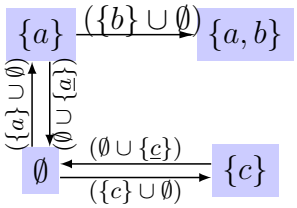

Рисунок 4: Конфигурационная система переходов $TC(\mathcal{E}_4)$

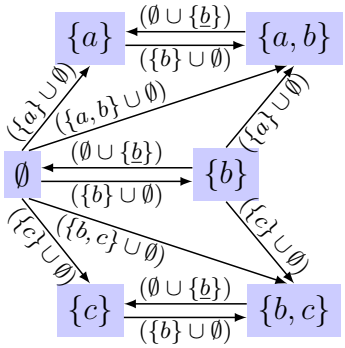

Рисунок 5: Конфигурационная система переходов $TC(\mathcal{E}_5)$

Теперь рассмотрим определение системы переходов, имеющей в качестве состояний остаточные структуры для РПСС после её конфигураций.

Определение 7. *Для РПСС* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0)$

$TE(\mathcal{E})$ *— остаточная система переходов* $(Reach(\mathcal{E}), \rightharpoonup, \mathcal{E} \setminus C_0)$, *помеченная на множестве* $\mathbb{L}$ *меток,*

где $\mathcal{F} \overset{(A \cup B)}{\rightharpoonup} \mathcal{F}'$ *в* $TE(\mathcal{E}) \iff C_0 \overset{(A \cup B)}{\rightarrow} C$ *в* $\mathcal{F}$ *и* $\mathcal{F}' = \mathcal{F} \setminus C$,

$Reach(\mathcal{E}) = \{\mathcal{F} \mid \exists \mathcal{E}_0, \ldots, \mathcal{E}_k \ (k \geq 0)$ *такие, что* $\mathcal{E}_0 = \mathcal{E} \setminus C_0$, $\mathcal{E}_k = \mathcal{F}$ *и* $\mathcal{E}_i \overset{(A \cup B)}{\rightharpoonup} \mathcal{E}_{i+1}$ $(0 \leq i < k)\}$.

Проиллюстрируем данное определение на примере.

Пример 9. *Рассмотрим РПСС* $\mathcal{E}_4 \in c\mathcal{RPES}$ *из примеров 2–6. Используя определения 5 и 7, получаем остаточную систему переходов* $TE(\mathcal{E}_4)$, *которая показана на рис. 6. Видим, что конфигурационная система переходов* $TC(\mathcal{E}_4)$ *(см. рис. 4) и остаточная система переходов* $TE(\mathcal{E}_4)$ *являются изоморфными.*

Рассмотрим РПСС $\mathcal{E}_5 \in c\mathcal{RPES}$ *из примеров 2–6. Используя определения 5 и 7, построим остаточную систему переходов* $TE(\mathcal{E}_5)$ *(см. рис. 7). Видим, что конфигурационная система переходов* $TC(\mathcal{E}_5)$ *(см. рис. 5) и остаточная система переходов* $TE(\mathcal{E}_5)$ *являются бисимуляционно эквивалентными, но не изоморфными.* ◇

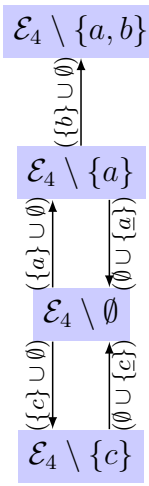

Рисунок 6: Остаточная система переходов $TE(\mathcal{E}_4)$

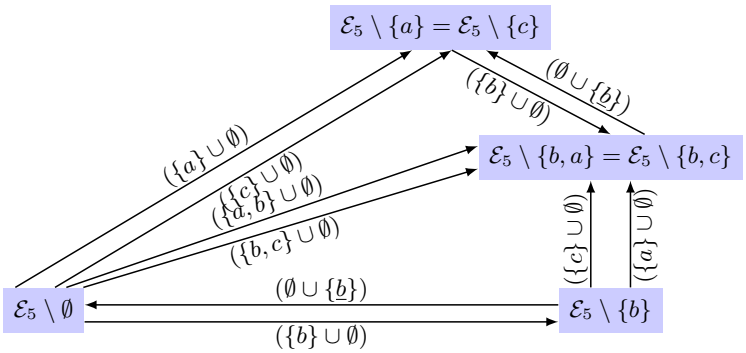

Рисунок 7: Остаточная система переходов $TE(\mathcal{E}_5)$

Установим взаимосвязи между состояниями и переходами конфигурационной и остаточной систем переходов для РПСС из класса $c\mathcal{RPES}$.

Утверждение 3. *Для РПСС* $\mathcal{E} = (E, <, \sharp, F, \prec, \rhd, C_0) \in c\mathcal{RPES}$ *верно:*

(a) для любой $C \in Conf(\mathcal{E})$ *верно* $\mathcal{E} \setminus C \in Reach(\mathcal{E})$;

(б) для любой $\mathcal{E}' \in Reach(\mathcal{E})$ *существует* $C \in Conf(\mathcal{E})$ *такая, что* $\mathcal{E}' = \mathcal{E} \setminus C$;

(в) для любых $C, C' \in Conf(\mathcal{E})$ *верно: если* $C \overset{(A \cup \underline{B})}{\rightarrow} C'$, *то* $\mathcal{E} \setminus C \overset{(A \cup \underline{B})}{\multimap} \mathcal{E} \setminus C'$;

(г) для любых $\mathcal{E}', \mathcal{E}'' \in Reach(\mathcal{E})$ *верно: если* $\mathcal{E}' \overset{(A \cup \underline{B})}{\multimap} \mathcal{E}''$, *то для любой* $C \in Conf(\mathcal{E})$ *такой, что* $\mathcal{E}' = \mathcal{E} \setminus C$, *существует* $C' \in Conf(\mathcal{E})$ *такая, что* $\mathcal{E}'' = \mathcal{E} \setminus C'$ *и верно* $C \overset{(A \cup \underline{B})}{\rightarrow} C'$.

Доказательство: См. приложение. □

Установим существование бисимуляции между конфигурационной и остаточной системами переходов для РПСС из класса $c\mathcal{RPES}$.

Теорема 1. *Для РПСС* $\mathcal{E} \in c\mathcal{RPES}$ *верно, что* $TC(\mathcal{E})$ *и* $TE(\mathcal{E})$ *являются бисимуляционно эквивалентными и неизоморфными в общем случае.*

Доказательство: Из примера 9 известно, что для РПСС $\mathcal{E}_5 \in c\mathcal{RPES}$ (см. примеры 2–8) конфигурационная система переходов $TC(\mathcal{E}_5)$ (см. рис. 5) и остаточная система переходов $TE(\mathcal{E}_5)$ (см. рис. 7) не изоморфны.

Проверим, что системы $TC(\mathcal{E})$ и $TE(\mathcal{E})$ бисимуляционно эквивалентны для каждой $\mathcal{E} \in c\mathcal{RPES}$. Определим отношение R следующим образом: $R = \{(C,\ \mathcal{E} \setminus C) \mid C \in Conf(\mathcal{E})\}$. Благодаря утверждению 3(а), верно $R \subseteq Conf(\mathcal{E}) \times Reach(\mathcal{E})$.

Проверим, что отношение R является бисимуляцией между системами $TC(\mathcal{E})$ и $TE(\mathcal{E})$. Очевидно, что $C_0 \in Conf(\mathcal{E})$ и, кроме того, $(C_0,\ \mathcal{E} \setminus C_0) \in R$.

Возьмём произвольную пару $(C,\ \mathcal{E} \setminus C)$, принадлежащую отношению R. Предположим, что $C \overset{(A \cup \underline{B})}{\rightarrow} C'$ в $TC(\mathcal{E})$ для некоторой $C' \in Conf(\mathcal{E})$. Благодаря утверждению 3(в) верно, что $\mathcal{E} \setminus C \overset{(A \cup \underline{B})}{\multimap} \mathcal{E} \setminus C'$. Кроме того, по определению отношения R, имеем $(C',\ \mathcal{E} \setminus C') \in R$.

Теперь предположим существование перехода $\mathcal{E} \setminus C \overset{(A \cup \underline{B})}{\multimap} \mathcal{E}'$ в $TE(\mathcal{E})$ для некоторой $\mathcal{E}' \in Reach(\mathcal{E})$. Согласно утверждению 3(г), для конфигурации $C \in Conf(\mathcal{E})$, существует конфигурация $C' \in Conf(\mathcal{E})$ такая, что $\mathcal{E}' = \mathcal{E} \setminus C'$ и $C \overset{(A \cup \underline{B})}{\rightarrow} C'$. Более того, по определению отношения R очевидно, что $(C',\ \mathcal{E} \setminus C') = (C',\ \mathcal{E}') \in R$. Следовательно, отношение R действительно является бисимуляцией. □

5. Заключение

В этой статье в контексте реверсивных первичных структур событий, сохраняющих причинно-следственные зависимости между событиями, были построены и исследованы семантики в терминах систем переходов, основанных на конфигурациях и остаточных структурах. С этой целью, во-первых, была определена истинно-параллельная (шаговая)

семантика рассматриваемой модели структур событий, которая основана на конфигурациях и которая строится из начальной конфигурации посредством добавления/отмены множеств произошедших параллельных событий, называемых шагом. Во-вторых, был предложен оператор удаления, который используется для построения остаточных структур, получаемых из заданной структуры событий посредством удаления из неё уже произошедших и конфликтующих событий, а также показана корректность и композиционные свойства данного оператора.

Есть надежда, что полученные здесь результаты могут быть полезны при разработке операционных семантик алгебраических исчислений реверсивных параллельных процессов, как результаты статей [10, 16, 21, 23] при построении и изучении традиционных (нереверсивных) алгебраических исчислений.

В дальнейшем планируется расширить список рассматриваемых моделей, включив реверсивные версии потоковых, расслоенных, обобщенных структур событий с симметричным и асимметричным конфликтом. Другой целью дальнейших исследований является изучение возможности получения изоморфизма, а не бисимуляции, между двумя типами семантик систем переходов за счёт обогащения модели реверсивных структур событий событиями, которые присутствуют в структуре, но которые не могут произойти из-за, например, отсутствия транзитивности/ацикличности в ПСЗ, наличия бесконечного количества предшественников по ПСЗ и т.д., как это было сделано для традиционных (нереверсивных) моделей в статье [8].

Список литературы

1. Aman B., Ciobanu G. Controlled reversibility in reaction systems. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) CMC 2017. Lecture Notes in Computer Science. 2017. Vol. 10725. P. 40–53.URL: https://doi.org/10.1007/978-3-319-73359-3_3
2. Arbach Y., Karcher D., Peters K., Nestmann U. Dynamic causality in event structures. Lecture Notes in Computer Science. 2015. Vol. 9039. P. 83–97.
URL: https://doi.org/10.1007/978-3-319-19195-9_6
3. Armas-Cervantes A., Baldan P., Garcia-Banuelos L. Reduction of event structures under history preserving bisimulation. J. Log. Algebr. Meth. Program. 2016. Vol. 85(6). P. 1110-1130.
URL: https://doi.org/10.1016/j.jlamp.2015.10.004
4. Aubert C., Cristescu I. Contextual equivalences in configuration structures and reversibility. J. Log. Algebr. Meth. Program. 2017. Vol. 86(1). P. 77–106.
URL: https://doi.org/10.1016/j.jlamp.2016.08.004

5. Baier C., Majster-Cederbaum M. The connection between event structure semantics and operational semantics for TCSP. Acta Informatica. 1994. Vol. 31.
URL: https://doi.org/10.1007/BF01178923
6. Barylska K., Gogolinska A., Mikulski L., Philippou A., Piatkowski M., Psara K. Formal translation from reversing Petri nets to coloured Petri nets. Lecture Notes in Computer Science. 2022. Vol. 13352. P. 172–186. URL: https://doi.org/10.1007/978-3-031-09005-9_12
7. Best E., Gribovskaya N., Virbitskaite I. Configuration- and residual-based transition systems for event structures with asymmetric conflict. Proc. SofSem 2017, Lecture Notes in Computer Science. 2017. Vol. 10139. P. 132–146. URL: https://doi.org/10.1007/978-3-319-51963-0_11
8. Best E., Gribovskaya N., Virbitskaite I. From event-oriented models to transition systems. In: Proc. Petri Nets 2018, Lecture Notes in Computer Science. 2018. Vol. 10877. P. 117–139.
URL: https://doi.org/10.1007/978-3-319-91268-4_7
9. Boudol G. Flow event structures and flow nets. Lecture Notes in Computer Science. 1990. Vol. 469. P. 62–95. URL: https://doi.org/10.1007/3-540-53479-2_4
10. Boudol G., Castellani I. Concurrency and atomicity. Theoretical Computer Science. 1988. Vol. 59. P. 25–84. URL: https://doi.org/10.1016/0304-3975(88)90096-5
11. Crafa S., Varacca D., Yoshid N. Event structure semantics of parallel extrusion in the pi-calculus. Lecture Notes in Computer Science. 2012. Vol. 7213. P. 225–239.
12. Danos V., Krivine J. Transactions in RCCS. In: M. Abadi and L. de Alfaro, editors, Proceedings of 16th International Conference CONCUR 2005, Lecture Notes in Computer Science. 2005. Vol. 3653. P. 398–412. URL: https://doi.org/10.1007/11539452_31
13. De Vos A., De Baerdemacker S., Van Rentergem Y. Synthesis of quantum circuits vs. Synthesis of Classical Reversible Circuits. Synthesis Lectures on Digital Circuits and Systems. Morgan & Claypool Publishers, 2018.
URL: https://doi.org/10.2200/S00856ED1V01Y201805DCS054
14. de Frutos Escrig D., Koutny M., Mikulski L. Reversing steps in Petri nets. In: Donatelli, S., Haar, S. (eds.) PETRI NETS 2019. Lecture Notes in Computer Science. 2019. Vol. 11522. P. 171–191. URL: https://doi.org/10.1007/978-3-030-21571-2 11
15. van Glabbeek R.J., Plotkin G.D. Configuration structures, event structures and Petri nets. Theoretical Computer Science. 2009. Vol. 410(41). P. 4111-4159.
URL: https://doi.org/10.1016/j.tcs.2009.06.014
16. van Glabbeek R.J., Vaandrager F.W. Bundle event structures and CCSP. Lecture Notes in Computer Science. 2003. Vol. 2761. P. 57–71. URL: https://doi.org/10.1007/978-3-540-45187-7_4
17. Graversen E., Phillips I.C.C., Yoshida N. Towards a categorical representation of reversible event structures. J. Log. Algebraic Methods Program. 2019. Vol. 104. P. 16–59.
URL: https://doi.org/10.1016/j.jlamp.2019.01.001
18. Graversen E., Phillips I.C.C., Yoshida N. Event structure semantics of (controlled) reversible CCS. J. Log. Algebraic Methods Program. 2021. Vol. 121: 100686.

URL: https://doi.org/10.1016/j.jlamp.2021.100686

19. Hoogers P.W., Kleijn H.C.M., Thiagarajan P.S. An event structure semantics for general Petri nets. Theoretical Computer Science. 1996. Vol. **153**. P. 129–170.
URL: https://doi.org/10.1016/0304-3975(95)00120-4
20. Kari J. Reversible cellular automata: from fundamental classical results to recent developments. New Generation Comput. 2018. Vol. 36(3). P. 145–172. URL: https://doi.org/10.1007/s00354-018-0034-6
21. Katoen J.-P. Quantitative and qualitative extensions of event structures. PhD Thesis. Twente University. 1996.
22. Kuhn S., Aman B., Ciobanu G.,Philippou A., Psara K., Ulidowski I. Reversibility in chemical reactions. In I. Ulidowski, I. Lanese, U. P. Schultz, and C. Ferreira, editors, Reversible Computation: Extending Horizons of Computing – Selected Results of the COST Action IC1405. Lecture Notes in Computer Science. 2020. Vol. 12070. P. 151–176.
URL: https://doi.org/10.1007/978-3-030-47361-7_7
23. Langerak R. Bundle event structures: a non-interleaving semantics for LOTOS. Formal Description Techniques V. IFIP Transactions. 1993. Vol. C-10. P. 331–346.
24. Lanese I., Mezzina C.A., Stefani J.-B. Reversibility in the higher-order π-calculus. Theoretical Computer Science. 2016. Vol. 625. P. 25–84. URL: https://doi.org/10.1016/j.tcs.2016.02.019
25. Lanese I., Palacios A., Vidal G. Causal-consistent replay debugging for message passing programs. In J. A. Pérez and N. Yoshida, editors, Formal Techniques for Distributed Objects, Components, and Systems – 39th IFIP WG 6.1 International Conference, FORTE 2019. Lecture Notes in Computer Science. 2019. Vol. 11535. P. 167–184. URL: https://doi.org/10.1007/978-3-030-21759-4_10
26. Majster-Cederbaum M., Roggenbach M. Transition systems from event structures revisited. Information Processing Letters. 1998. Vol. 67(3). P. 119-124. URL: https://doi.org/10.1016/S0020-0190(98)00105-7
27. Medic D., Mezzina C. A., Phillips I., Yoshida N. Towards a formal account for software transactional memory. In I. Lanese and M. Rawski, editors, Reversible Computation – 12th International Conference, RC 2020. Lecture Notes in Computer Science 2020. Vol. 12227. P. 255–263. URL: https://doi.org/10.1007/978-3-030-52482-1_16
28. Melgratti H.C., Mezzina C.A., Pinna G.M. A distributed operational view of reversible prime event structures. In: 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). 2021. P. 1–13. URL: https://doi: 10.1109/LICS52264.2021.9470623.
29. Philippou A., Psara K. Reversible computation in cyclic Petri nets. CoRR abs/2010.04000. 2020. URL: https://arxiv.org/abs/2010.04000
30. Phillips I.C.C., Ulidowski I. A hierarchy of reverse bisimulations on stable configuration structures. Math. Struct. Comput. Sci. 2012. Vol. 22. P. 333–372
URL: https://doi.org/10.1017/S0960129511000429
31. Phillips I.C.C., Ulidowski I. Event identifier logic. Math. Struct. Comput. Sci. 2014. Vol. 24.

P. 1–51 URL: https://doi.org/10.1017/S0960129513000510

32. Phillips I.C.C., Ulidowski I. Reversibility and asymmetric conflict in event structures. Logic and Algebraic Methods in Programming 2015. Vol. 84(6). P. 781–805
URL: https://doi.org/10.1016/j.jlamp.2015.07.004
33. Pinna G.M. Reversing steps in membrane systems computations. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) CMC 2017. LNCS 2018. Vol. 10725. P. 245–261.
URL: https://doi.org/10.1007/978-3-319-73359-3 16
34. Ulidowski I., Phillips I.C.C., Yuen S. Reversing event structures. New Generation Computing 2018. Vol. 36(3). P. 281–306.
URL: https://doi.org/10.1007/s00354-018-0040-8
35. Winskel G. Events in computation. PhD Thesis. University of Edinburgh. 1980.
36. Winskel G. Introduction to event structures. *Lecture Notes in Computer Science*. 1989. Vol. 354. P. 364–397.

Приложение

Доказательство леммы 1.

Поскольку $C \in Conf(\mathcal{E})$, то по определению 3, для каждого $1 \leq i \leq n$ существуют множества $A_i \subseteq E$ и $B_i \subseteq F$ такие, что $C_{i-1} \overset{(A_i \cup B_i)}{\rightarrow} C_i$ и $C = C_n$ $(n \geq 0)$. Покажем, что C_i конечно, лево-замкнуто относительно $<$ и бесконфликтно для каждого $0 \leq i \leq n$ методом математической индукции:

$i = 0$. C_0 конечно, лево-замкнуто относительно $<$ и бесконфликтно по условию леммы.

$i > 0$. Предположим, что C_i — конечно, лево-замкнуто относительно $<$ и бесконфликтно, и покажем, что множество C_{i+1} также конечно, лево-замкнуто относительно $<$ и бесконфликтно. Так как $C_i \overset{(A_{i+1} \cup B_{i+1})}{\rightarrow} C_{i+1}$, то $C_{i+1} = (C_i \setminus B_{i+1}) \cup A_{i+1}$ и шаг $(A_{i+1} \cup \underline{B}_{i+1})$ возможен из C_i. Тогда по пункту (а) определения 3 верно, что $C_i \cup A_{i+1}$ — конечное и бесконфликтное множество. Следовательно, множество C_{i+1} также конечно и бесконфликтно как подмножество конечного и бесконфликтного множества $C_i \cup A_{i+1}$. Проверим лево-замкнутость C_{i+1}. Выберем произвольным образом событие $e \in C_{i+1}$ такое, что $x < e$ для некоторого $x \in E$. Так как $C_{i+1} = (C_i \setminus B_{i+1}) \cup A_{i+1}$, то возможны два случая:

- $e \in C_i \setminus B_{i+1} \subseteq C_i$. Из лево-замкнутости множества C_i получаем, что $x \in C_i$. Предположим, что $x \in B_{i+1} \subseteq F$. Благодаря выполнимости свойства СПСЗ для $\mathcal{E}$, верно, что $e \rhd \underline{x}$. Поскольку шаг $(A_{i+1} \cup \underline{B}_{i+1})$ возможен из C_i, то по пункту (г) определения 3 известно, что $e \notin C_i \cup A_{i+1}$, что противоречит условию $e \in C_i$. Значит, верно $x \notin B_{i+1}$. Таким образом, $x \in C_i \setminus B_{i+1} \subseteq C_{i+1}$.
- $e \in A_{i+1}$. Поскольку шаг $(A_{i+1} \cup \underline{B}_{i+1})$ возможен из C_i, то по пункту (б) определения 3 истинно, что $x \in C_i \setminus B_{i+1} \subseteq C_{i+1}$.

Следовательно, множество C_{i+1} лево-замкнуто относительно $<$. □

Доказательство леммы 2.

(а) Следует из определения 5.

(б) Для начала проверим, что построенная структура $\mathcal{E} \setminus C = (E', <', \sharp', F', \prec', \rhd', C'_0)$, где $E' = E \setminus \widetilde{C} \cup \sharp(\widetilde{C})$ является РПСС. Для этого проверим все требования определения 2.

- Поскольку $\mathcal{E}$ — РПСС, то множество E счетно. По пункту (а) известно, что $E' \subseteq E$. Следовательно, множество E', являющееся подмножеством счетного множества E, счетно.

– По определению 5 верно $\sharp' = \sharp \cap\ (E' \times E') \subseteq (E' \times E')$. Благодаря тому, что $\mathcal{E}$ является РПСС, имеем, что отношение $\sharp \subseteq E \times E$ иррефлексивно и симметрично. По пункту (а) $\sharp' \subseteq \sharp$, следовательно отношение $\sharp'$ также иррефлексивно. Симметричность этого отношения следует из симметричности отношения $\sharp$ и равенства $\sharp' = \sharp \cap\ (E' \times E')$.

– В соответствии с определением 5 очевидно, что $<'=< \cap\ (E' \times E') \subseteq (E' \times E')$. Так как $\mathcal{E}$ — РПСС, то $<$ — иррефлексивный частичный порядок, множество $\lfloor e \rfloor_<$ конечно и бесконфликтно для каждого $e \in E$, и кроме того, для каждых $e, e' \in E$ верно, что если $e < e'$, то $\neg(e \sharp e')$. Поскольку $<'=< \cap\ (E' \times E')$, то в силу иррефлексивности и транзитивности отношения $<$ получаем, что отношение $<'$ также является иррефлексивным и транзитивным, то есть иррефлексивным частичным порядком. Далее по пункту (а) верно $<' \subseteq <$, и, следовательно, также истинно вложение $\lfloor e \rfloor_{<'} \subseteq \lfloor e \rfloor_<$ для любого события $e \in E'$. Значит множество $\lfloor e \rfloor_{<'}$ $(e \in E')$ конечно и бесконфликтно как подмножество конечного и бесконфликтного множества $\lfloor e \rfloor_<$. Проверим, что для каждых $e, e' \in E'$ верно, что если $e <' e'$, то $\neg(e \sharp' e')$. Предположим противное, то есть существуют $e, e' \in E'$ такие, что $e\ <'\ e'$ и $e \sharp' e'$. Благодаря пункту (а) получаем, что $e\ <\ e'$ и $e \sharp e'$, что противоречит тому, что $\mathcal{E}$ — РПСС. Значит для каждых $e, e' \in E'$ верно, что если $e\ <'\ e'$, то $\neg(e \sharp' e')$.

– По определению 5 имеем, что $F' = F \cap E'$, значит верно $F' \subseteq E'$.

– По определению 5 известно, что $\prec'=\prec \cap (E' \times F') \subseteq E' \times F'$. Более того, поскольку $\mathcal{E}$ — РПСС, то для каждого $u \in F$ верно, что $u \prec \underline{u}$ и $\llcorner u \lrcorner_\prec$ — конечное и бесконфликтное множество. По пункту (а) верно, что $F' \subseteq F$. Следовательно, для каждого $u \in F'$ истинно, что $u \prec \underline{u}$. Так как $\prec'=\prec \cap (E' \times F')$, то для каждого $u \in F' \subseteq E'$ справедливо $u \prec' \underline{u}$. Далее, по пункту (а) получаем, что $\prec' \subseteq \prec$, а значит истинно вложение $\llcorner u \lrcorner_{\prec'} \subseteq \llcorner u \lrcorner_\prec$. Таким образом, множество $\llcorner u \lrcorner_{\prec'}$ — конечно и бесконфликтно как подмножество конечного и бесконфликтного множества $\llcorner u \lrcorner_\prec$.

– По определению 5 имеем, что $\rhd' = \rhd \cap (E' \times F') \subseteq E' \times \underline{F'}$. Выберем произвольным образом $u \in F'$ и $e \in E'$ такие, что $e \prec' \underline{u}$ и покажем, что $\neg(e \rhd' \underline{u})$. По пункту (а) верно $e \prec \underline{u}$. Благодаря тому, что $\mathcal{E}$ является РПСС, истинно $\neg(e \rhd \underline{u})$. Вновь по пункту (а) имеем, что $\rhd' \subseteq \rhd$. Отсюда получаем, что $\neg(e \rhd' \underline{u})$.

– Пусть $\ll'$ определено по правилу: $e \ll' e'$, если и только если $e <' e'$, а также $e' \rhd' \underline{e}$, если $e \in F'$. Проверим, что отношение конфликта $\sharp'$ наследуется по $\ll'$. Пусть $e, e', e'' \in E'$ и $e \sharp' e' \ll' e''$. Поскольку $e' \ll e''$, имеем, что $e' <' e''$ и $e'' \rhd' \underline{e'}$, если $e' \in F'$. Заметим, что из пункта (а) следует, что $\sharp' \subseteq \sharp$, $<' \subseteq <$, $\rhd' \subseteq \rhd$ и $F' \subseteq F$. Отсюда получаем, что $e \sharp e'$, $e' < e''$ и $e'' \rhd \underline{e'}$, если $e' \in F'$. Поскольку $F' = F \cap E'$ по определению 5, то верно: $e' \ll e''$. Так как $\mathcal{E}$ — РПСС, то отношение конфликта $\sharp$ наследуется по $\ll$, и, следовательно, $e \sharp e''$. Благодаря определению 5, поскольку $e, e'' \in E'$, верно, что $e \sharp' e''$.

– По определению 5 справедливо $C'_0 = C \cap E'$, и, следовательно, $C'_0 \subseteq E'$.

Таким образом, мы убедились в том, что $\mathcal{E} \setminus C$ является РПСС. Теперь проверим, что $\mathcal{E} \setminus C \in c\mathcal{RPES}$. Для начала покажем, что $\mathcal{E} \setminus C$ обладает свойством СПСЗ. Возьмем произвольные события $e \in E'$ и $u \in F'$.

– Проверим истинность $e \prec' \underline{u} \Leftrightarrow e = u$.

Пусть $e \prec' \underline{u}$. Из пункта (а) следует $e \prec \underline{u}$. Так как $\mathcal{E}$ обладает свойством СПСЗ, то $e = u$.

Теперь предположим $e = u$. Поскольку $\mathcal{E} \setminus C$ является РПСС, то $u \prec' \underline{u}$ для всех $u \in F'$. Значит, справедливо $e \prec' \underline{u} \Leftrightarrow e = u$.

– Проверим истинность $e \rhd' \underline{u} \Leftrightarrow u <' e$.

Пусть $e \rhd' \underline{u}$ (или $u <' e$). Из пункта (а) следует $e \rhd \underline{u}$ (или $u < e$). Так как $\mathcal{E}$ обладает свойством СПСЗ, то $u < e$ (или $e \rhd \underline{u}$). Поскольку имеем $e \in E'$ и $u \in F' \subseteq E'$, то справедливо $u <' e$ ($e \rhd' \underline{u}$), согласно определению 5.

Таким образом, получаем, что $\mathcal{E} \setminus C$ обладает свойством СПСЗ.

Осталось проверить, что начальная конфигурация C'_0 является конечным, лево-замкнутым относительно $<'$ и бесконфликтным множеством. Так как $\mathcal{E} \in c\mathcal{RPES}$, то C_0 — конечное, лево-замкнутое относительно $<$ и бесконфликтное множество. Тогда по лемме 1 заключаем, что конфигурация C также является конечным, лево-замкнутым относительно $<$ и бесконфликтным множеством. Вложение $C'_0 \subseteq C$ истинно по лемме 2(а). Тогда множество C'_0 конечно и бесконфликтно как подмножество конечного и бесконфликтного множества C. Проверим, что C'_0 лево-замкнуто относительно $<'$. Возьмем произвольные события $x \in C'_0$ и $y \in E'$ такие, что $y <' x$. Благодаря лемме 2(а) верно $y < x$ и $x \in C$. Так как множество C лево-замкнуто относительно $<$, то $y \in C$. Согласно определению 5 верно $C'_0 = C \cap E'$. Отсюда, по-

скольку $y \in E'$, получаем, что $y \in C'_0$, что доказывает лево-замкнутость множества C'_0.

(в) Пусть $a \in \widetilde{C}$. По определению 5, существует $e \in (C \setminus F) \subseteq C$ такое, что $a \leq e$. Из леммы 1 знаем, что конфигурация C лево-замкнута относительно отношения $<$. Отсюда следует $a \in C$. Значит, верно $\widetilde{C} \subseteq C$. □

Доказательство утверждения 1.

Предположим $\mathcal{E} \in c\mathcal{RPES}$ и $C \in Conf(\mathcal{E})$. По определению 5, определим $\mathcal{E}' = \mathcal{E} \setminus C = (E', <'=< \cap (E' \times E'), \sharp' = \sharp \cap (E' \times E'), F' = (F \cap E'), \prec'=\prec \cap (E' \times \underline{F'}), \rhd' = \rhd \cap (E' \times \underline{F'}), C'_0 = C \cap E')$, где $E' = E \setminus \big(\widetilde{C} \cup \sharp(\widetilde{C})\big)$. И пусть $C' \in Conf(\mathcal{E}')$ такая, что $C'_0 \overset{(A \cup \underline{B})}{\rightarrow} C'$ в $\mathcal{E}'$. Это означает, что $C'_0 \subseteq E'$ — конечное и бескофликтное множество, $A \subseteq E'$, $B \subseteq F'$, шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E}'$ и $C' = (C'_0 \setminus B) \cup A$. Из леммы 2(а) известно, что $E' \subseteq E$ и $F' \subseteq F$, что позволяет сделать вывод о том, что $A \subseteq E$ и $B \subseteq F$. Более того, поскольку $\mathcal{E} \in c\mathcal{RPES}$, то C_0 — конечное, лево-замкнутое относительно отношения $<$ и бесконфликтное множество событий. Отсюда, благодаря лемме 1, получаем конечность, бесконфликтность и лево-замкнутость множества C.

Убедимся, что шаг $(A \cup \underline{B})$ возможен из C в $\mathcal{E}$. Проверим выполнение пунктов (а)–(г) определения 3.

(а) Так как шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E}'$, то $A \cap C'_0 = \emptyset$, $B \subseteq C'_0$ и $(C'_0 \cup A)$ — конечное и бесконфликтное множество. По определению 5 верно: $C'_0 = C' \cap E'$. Отсюда, поскольку $A \subseteq E'$ и $A \cap C'_0 = \emptyset$, то истинно $A \cap C = A \cap E' \cap C = \emptyset$. Согласно лемме 2(а), верно $C'_0 \subseteq C$. Значит, $B \subseteq C$. Заметим, что множество A, являющееся подмножеством конечного и бесконфликтного множества $(C'_0 \cup A)$, конечно и бесконфликтно. Тогда множество $C \cup A$ конечно как объединение двух конечных множеств. Осталось проверить бесконфликтность этого множества. Предположим обратное, то есть пусть существуют события $e, e' \in C \cup A$ такие, что $e \sharp e'$. Так как множества C и A являются бесконфликтными, то без ограничения общности можно считать, что $e \in A \subseteq E'$ и $e' \in C$. Рассмотрим два возможных случая:

1. $e' \in E'$. Тогда верно $e' \in C \cap E' = C'_0$. Следовательно, по определению 5, истинно $e \sharp' e'$, что противоречит бесконфликтности множества $C'_0 \cup A$, т.е. определению 3(а).
2. $e' \notin E'$. Поскольку $E' = E \setminus \widetilde{C} \cup \sharp(\widetilde{C})$, то $e' \in \widetilde{C} \cup \sharp(\widetilde{C})$.
   – $e' \in \widetilde{C}$. Поскольку $e \sharp e'$, то истинно $e \in \sharp(\widetilde{C})$, и, следовательно событие

e не принадлежит множеству E', что противоречит вложению $A \subseteq E'$, т.е. условию определения 3.

– $e' \in \sharp(\widetilde{C})$. По определению 5 существует событие $v \in \widetilde{C}$ такое, что $e' \sharp v$. По пункту (в) леммы 2 известно, что $\widetilde{C} \subseteq C$. Следовательно верно, что $v \in C$. Так как $e' \sharp v$, то пришли к противоречию с бесконфликтностью множества C, т. е. с леммой 1.

Таким образом, множество $C \cup A$ бесконфликтно.

(б) Пусть $e \in A$, $e' \in E$ и $e' < e$. Рассмотрим два допустимых случая:

1. $e' \in E'$. Согласно определению 5 получаем, что $e' <' e$. Так как шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E}'$, то событие e' принадлежит множеству $C'_0 \setminus B$ и $B \subseteq C'_0$. Согласно лемме 2 (а), верно $C'_0 \subseteq C$. Значит, $e' \in C \setminus B$.
2. $e' \notin E'$. Согласно определению 5 имеем, что $e' \in \widetilde{C} \cup \sharp(\widetilde{C})$. Покажем, что $e' \in \widetilde{C}$. Предположим противное, то есть $e' \in \sharp(\widetilde{C})$. По определению 5 существует событие $v \in \widetilde{C}$ такое, что $e' \sharp v$. Поскольку РПСС $\mathcal{E}$ обладает свойством СПСЗ и $e' < e$, то верно $e' \ll e$. Согласно определению 2, так как $v \sharp e' \ll e$, то $v \sharp e$, то есть $e \in \sharp(\widetilde{C})$, что противоречит принадлежности события e множеству E'. Значит, истинно $e' \in \widetilde{C}$. По пункту (в) леммы 2 известно, что $\widetilde{C} \subseteq C$, то есть $e' \in C$. Более того, поскольку $e' \notin E'$ и $B \subseteq E'$, то верно $e' \notin B$. Следовательно, получаем, что $e' \in C \setminus B$.

(в) Пусть $e \in B$, $e' \in E$ и $e' \prec \underline{e}$. Поскольку РПСС $\mathcal{E}$ обладает свойством СПСЗ, то $e' \prec \underline{e}$ тогда и только тогда, когда $e' = e$. Далее, так как $e \in B \subseteq C$, то $e' = e \in C \setminus (B \setminus \{e\})$.

(г) Пусть $e \in B$, $e' \in E$ и $e' \rhd \underline{e}$. Проверим два возможных варианта:

1. $e' \in E'$. Благодаря определению 5 верно $e' \rhd' \underline{e}$. Так как шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E}'$, то $e' \notin (C'_0 \cup A)$. Согласно определению 5, верно $C'_0 = C \cap E'$. Поскольку $e' \in E'$, то $e' \notin C \cup A$.
2. $e' \notin E'$. По определению 5 имеем $e' \in \widetilde{C} \cup \sharp(\widetilde{C})$. Покажем, что $e' \in \sharp(\widetilde{C})$. Предположим противное, то есть $e' \in \widetilde{C}$. По Определению 5 это означает существование события $v \in C \setminus F$ такого, что $e' \leq v$. Поскольку РПСС $\mathcal{E}$ обладает свойством СПСЗ и $e' \rhd \underline{e}$, то $e < e'$. Тогда, по транзитивности отношения $<$ получаем, что $e \leq v$, то есть $e \in \widetilde{C}$, что противоречит вложению $B \subseteq F' \subseteq E'$, т.е. определению 3. Таким образом, имеем $e' \in \sharp(\widetilde{C})$. Согласно определению 5, существует событие $c \in \widetilde{C}$ такое, что $e' \sharp c$. Далее, по леммы 2(а), известно $\widetilde{C} \subseteq C$. Поскольку

$C \cup A$ бесконфликтное множество, то $e' \notin C \cup A$.

Таким образом, шаг $(A \cup \underline{B})$ возможен из C в $\mathcal{E}$. Следовательно, получаем $C \stackrel{(A \cup \underline{B})}{\rightarrow} C''$ в $\mathcal{E}$ и $C'' = (C \setminus B) \cup A$.

Осталось показать, что $\mathcal{E} \setminus C'' = \mathcal{E}' \setminus C'$. Напомним, что по определению 5 верно:

- $\mathcal{E}' = (E', <'=< \cap (E' \times E'), \sharp' = \sharp \cap (E' \times E'), F' = (F \cap E'), \prec'=\prec \cap (E' \times \underline{F'}), \rhd' = \rhd \cap\ (E' \times \underline{F'}), C'_0 = C' \cap E')$, где $E' = E \setminus \big(\widetilde{C} \cup \sharp(\widetilde{C})\big)$;
- $\mathcal{E}' \setminus C' = (\overline{E}, \overline{<} =<' \cap (\overline{E} \times \overline{E}), \overline{\sharp} = \sharp' \cap (\overline{E} \times \overline{E}), \overline{F} = (F' \cap \overline{E}), \overline{\prec} =\prec' \cap (\overline{E} \times \underline{\overline{F}}), \overline{\rhd} = \rhd' \cap\ (\overline{E} \times \underline{\overline{F}}), \overline{C}_0 = C' \cap \overline{E})$, где $\overline{E} = E' \setminus \big(\widetilde{C'} \cup \sharp'(\widetilde{C'})\big)$;
- $\mathcal{E} \setminus C'' = (E'', <''=< \cap (E'' \times E''), \sharp'' = \sharp \cap (E'' \times E''), F'' = (F \cap E''), \prec''=\prec \cap (E'' \times \underline{F''}), \rhd'' = \rhd \cap\ (E'' \times \underline{F''}), C''_0 = C'' \cap E'')$, где $E'' = E \setminus \big(\widetilde{C''} \cup \sharp(\widetilde{C''})\big)$.

Сначала проверим, что верно $E'' = \overline{E}$. Для этого установим истинность следующих вложений.

1. $\widetilde{C''} \subseteq (\widetilde{C} \cup \widetilde{C'})$. Пусть $x \in \widetilde{C''}$. Тогда существует $y \in C'' \setminus F$ такое, что $x \leq y$. Так как $C'' = (C \setminus B) \cup A$, то два случая возможны.
   (a) $y \in (C \setminus B) \setminus F$. Так как $B \subseteq F$, верно $y \in C \setminus F$, т.е. $x \in \widetilde{C}$, по определению 5.
   (b) $y \in A \setminus F$. Тогда имеем $y \in E'$, поскольку верно $A \subseteq E'$. Допустимы два случая.
      i. $x \in E'$. По определению 5, верно $x \leq' y$. Более того, справедливо $A \setminus F \subseteq A \setminus F'$, потому что $F' \subseteq F$. Значит, в силу $C' = (C'_0 \setminus B) \cup A$, получаем $x \in \lfloor A \setminus F' \rfloor_{\leq'} \subseteq \lfloor C' \setminus F' \rfloor_{\leq'} = \widetilde{C'}$, по определению 5.
      ii. $x \notin E'$. Согласно определению 5, имеем $x \in \widetilde{C} \cup \sharp(\widetilde{C})$. Покажем, что $x \notin \sharp(\widetilde{C})$. Рассмотрим два возможных случая.
         - $x = y \in A \setminus F$. Поскольку $(C \cup A)$ — бесконфиликтное множество, по определению 3(а), и $\widetilde{C} \subseteq C$, по лемме 2(в), то верно $x \notin \sharp(\widetilde{C})$, по определению 5.
         - $x < y \in A \setminus F$. Тогда имеем $x \in C$, согласно определению 3(б). Так как C — бесконфиликтное множество, по определению 3, и $\widetilde{C} \subseteq C$, по лемме 2(в), то справедливо $x \notin \sharp(\widetilde{C})$, по определению 5.

      Следовательно, получаем $x \in \widetilde{C}$.
2. $\widetilde{C}, \widetilde{C'} \subseteq \widetilde{C''}$.

   Пусть $x \in \widetilde{C}$. Тогда существует $y \in C \setminus F$ такое, что $x \leq y$. Так как $C'' = (C \setminus B) \cup A$ и $B \subseteq F$, то имеем $y \in C'' \setminus F$, т.е. $x \in \widetilde{C''}$.

   Пусть $x \in \widetilde{C'}$. Тогда существует $y \in C' \setminus F' \subseteq E'$ такое, что $x \leq' y$. Кроме того,

согласно определению 5, имеем $F' = F \cap E'$ и $C'_0 = C \cap E'$. Так как $C'' = (C \setminus B) \cup A$ и $C' = ((C \cap E') \setminus B) \cup A$, то верно $y \in C'' \setminus F'$. В силу того, что справедливо $F' = F \cap E'$, $y \notin F'$ и $y \in E'$, заключаем, что $y \notin F$, т.е. $y \in C'' \setminus F$. По лемме 2(a), получаем $x \leq y$, поскольку верно $x \leq' y$. Следовательно, истинно $x \in \widetilde{C''}$.

3. $\sharp(\widetilde{C''}) \subseteq (\sharp(\widetilde{C}) \cup \sharp'(\widetilde{C'}))$. Пусть $x \in \sharp(\widetilde{C''})$. Тогда существует $y \in \widetilde{C''}$ такое, что $x \sharp y$. По определению 5, найдется $z \in C'' \setminus F$ такое, что $y \leq z$. Поскольку $\mathcal{E}$ принадлежит классу $c\mathcal{RPES}$, то получаем $x \sharp z$. Так как $C'' = (C \setminus B) \cup A$, то возможны два случая.
   (a) $z \in (C \setminus B) \setminus F$. Тогда верно $z \in C \setminus F$, благодаря $B \subseteq F$. Значит, справедливо $x \in \sharp(\widetilde{C})$.
   (b) $z \in A \setminus F$. Тогда имеем $z \in E'$, поскольку верно $A \subseteq E'$. Допустимы два случая.
      i. $x \in E'$. Так как $F' \subseteq F$, согласно лемме 2 (a), то $z \in A \setminus F'$. В силу того, что $C' = (C'_0 \setminus B) \cup A$, получаем $z \in C' \setminus F'$. Поскольку $C' \setminus F' \subseteq \widetilde{C'}$, по определению 5, имеем $z \in \widetilde{C'}$. Вновь по определению 5, верно $x \sharp' z$, так как $x \sharp z$. Тогда истинно $x \in \sharp'(\widetilde{C'})$.
      ii. $x \notin E'$. По определению 5, это означает $x \in \widetilde{C} \cup \sharp(\widetilde{C})$. Так как имеем $z \in E'$, то верно $z \notin \widetilde{C} \cup \sharp(\widetilde{C})$. Поскольку $z \notin \sharp(\widetilde{C})$, получаем $x \notin \widetilde{C}$. Тогда истинно $x \in \sharp(\widetilde{C})$.
4. $\sharp(\widetilde{C}), \sharp'(\widetilde{C'}) \subseteq \sharp(\widetilde{C''})$.
   Пусть $x \in \sharp(\widetilde{C})$. Тогда существует $y \in \widetilde{C}$ такое, что $x \sharp y$. По пункту 2, имеем $y \in \widetilde{C''}$. Значит, верно $x \in \sharp(\widetilde{C''})$.
   Пусть $x \in \sharp'(\widetilde{C'})$. Тогда существует $y \in \widetilde{C'}$ такое, что $x \sharp' y$. По лемме 2(a), получаем $x \sharp y$. По пункту 2, имеем $y \in \widetilde{C''}$. Значит, истинно $x \in \sharp(\widetilde{C''})$.

Следовательно, получили $\widetilde{C''} = (\widetilde{C} \cup \widetilde{C'})$ и $\sharp(\widetilde{C''}) = (\sharp(\widetilde{C}) \cup \sharp'(\widetilde{C'}))$. Тогда нетрудно убедиться в том, что верно $E'' = \overline{E}$. Проверим, что $F'' = \overline{F}$. По определению 5, получаем, что $F'' = F \cap E''$ и $\overline{F} = F' \cap \overline{E} = (F \cap E') \cap \overline{E}$. Согласно лемме 2(a), известно $\overline{E} \subseteq E'$. Отсюда заключаем, что верно $\overline{F} = F \cap \overline{E} = F \cap E'' = F''$, поскольку множества E'' и $\overline{E}$ совпадают. Аналогично доказывается, что $\nabla'' = \overline{\nabla}$ для каждого отношения $\nabla \in \{\prec, \sharp, <, \rhd\}$. Осталось показать совпадение начальных конфигураций. По определению 5, имеем $C''_0 = C'' \cap E''$ и $\overline{C}_0 = C' \cap \overline{E}$, где $C'' = (C \setminus B) \cup A$ и $C' = (C'_0 \setminus B) \cup A = ((C \cap E') \setminus B) \cup A$. Поскольку $A \subseteq E'$ и $B \subseteq F'$, то следующее равенство верно: $C' = ((C \setminus B) \cup A) \cap E' = C'' \cap E'$. В силу леммы 2(a), имеем $\overline{E} \subseteq E'$. Тогда справедливо $\overline{C}_0 = C' \cap \overline{E} = C'' \cap E' \cap \overline{E} = C'' \cap \overline{E} = C''_0$.

Таким образом, истинно $\mathcal{E} \setminus C'' = (\mathcal{E} \setminus C) \setminus C'$. □

Доказательство утверждения 2.

Предположим $\mathcal{E} \in c\mathcal{RPES}$ и $C', C'' \in Conf(\mathcal{E})$ такие, что $C' \overset{(A \cup \underline{B})}{\rightarrow} C''$ в $\mathcal{E}$. Поскольку $C' \overset{(A \cup \underline{B})}{\rightarrow} C''$ в $\mathcal{E}$, то имеем следующее: $C' \subseteq E$ — конечное и бескофликтное множество, $A \subseteq E$, $B \subseteq F$, шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$ и $C'' = (C' \setminus B) \cup A$. По определению 5, определим $\mathcal{E} \setminus C' = (E', <'=< \cap (E' \times E'), \sharp' = \sharp \cap (E' \times E'), F' = (F \cap E'), \prec'=\prec \cap (E' \times \underline{F'}), \rhd' = \rhd \cap\ (E' \times \underline{F'}), C'_0 = C' \cap E')$, где $E' = E \setminus \big(\widetilde{C'} \cup \sharp(\widetilde{C'})\big)$.

Предложение A. (а) $A \subseteq E'$; (б) $B \subseteq F'$.

Доказательство предложения A.

(а) Покажем, что верно $A \subseteq E'$. Предположим обратное, т.е. существует $a \in A$ такое, что $a \notin E'$. Поскольку $E' = E \setminus \big(\widetilde{C'} \cup \sharp(\widetilde{C'})\big)$ и $A \subseteq E$, то a принадлежит $\widetilde{C'} \cup \sharp(\widetilde{C'})$.

– $a \in \widetilde{C'}$. По лемме 2(в), имеем $a \in C'$, что противоречит $A \cap C' = \emptyset$, т.е. определению 3(а).

– $a \in \sharp(\widetilde{C'})$. Тогда $a \sharp e$ для некоторого события $e \in \widetilde{C'}$. В силу леммы 2(в), верно $e \in C'$. Поскольку имеем, что $a \in A$, $e \in C'$ и $a \sharp e$, то пришли к противоречию с бесконфликтностью множества $(C' \cup A)$, т.е. с определением 3(а).

Таким образом, верно $A \subseteq E'$.

(б) Покажем, что справедливо $B \subseteq F'$. Предположим обратное, т.е. существует $b \in B$ такое, что $b \notin F'$. Поскольку $E' = E \setminus \big(\widetilde{C'} \cup \sharp(\widetilde{C'})\big)$, $F' = F \cap E'$ и $B \subseteq F$, то b принадлежит $\widetilde{C'} \cup \sharp(\widetilde{C'})$.

– $b \in \widetilde{C'}$. По определению 5, существует $e \in (C' \setminus F)$ такое, что верно $b < e$, поскольку имеем $e \neq b \in B \subseteq F$. Благодаря тому, что РПСС $\mathcal{E}$ обладает свойством СПСЗ, получаем $e \rhd \underline{b}$. Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то пришли к противоречию с требованием $e \notin (C' \cup A)$, т.е. с определением 3(г).

– $b \in \sharp(\widetilde{C'})$. Тогда верно $b \sharp e$ для некоторого события $e \in \widetilde{C'}$. В силу леммы 2(в), верно $e \in C'$. Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то $b \in B \subseteq C'$, что противоречит бесконфликности множества C', т.е. условию определения 3.

Таким образом, справедливо $B \subseteq F'$. □

Убедимся, что шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E} \setminus C'$. Проверим выполнение пунктов (а)–(г) определения 3.

(а) Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то $A \cap C' = \emptyset$, $B \subseteq C'$ и $(C' \cup A)$ — конечное и бесконфликтное множество. Согласно лемме 2(а), верно $C'_0 \subseteq C'$. Значит, получаем, что $A \cap C'_0 = \emptyset$ и множество $(C'_0 \cup A)$, являющееся подмножеством конечного и бес-

конфликтного множества $(C' \cup A)$, конечно и бесконфликтно. Осталось проверить, что $B \subseteq C'_0$. Согласно предложению А(б), имеем $B \subseteq F'$. Так как по определению 5 верно: $F' = F \cap E'$ и $B \subseteq F$, то также верно, что $B \subseteq E'$. Вновь согласно определению 5, поскольку истинно $B \subseteq C'$, то истинно $B \subseteq C' \cap E' = C'_0$.

(б) Пусть $e \in A$, $e' \in E'$ и $e' <' e$. Благодаря лемме 2(а), получаем $e' < e$. Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то имеем $e' \in C' \setminus B$. Отсюда верно, что $e' \in C' \cap E'$ и $e' \notin B$. Следовательно, справедливо $e' \in (C'_0 = C' \cap E') \setminus B$, в силу определения 5.

(в) Пусть $e \in B$, $e' \in E'$ и $e' \prec' \underline{e}$. По лемме 2(а), получаем $e' \prec \underline{e}$. Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то $e' \in C' \setminus (B \setminus \{e\})$, т.е. $e' \in C'$ и $e' \notin B \setminus \{e\}$. Следовательно, истинно $e' \in ((C'_0 = C' \cap E') \setminus (B \setminus \{e\}))$, согласно определению 5.

(г) Пусть $e \in B$, $e' \in E'$ и $e' \rhd' \underline{e}$. Благодаря лемме 2(а), верно $e' \rhd \underline{e}$. Так как шаг $(A \cup \underline{B})$ возможен из C' в $\mathcal{E}$, то $e' \notin (C' \cup A)$. Из леммы 2(а) известно $C'_0 \subseteq C'$. Следовательно, верно $e' \notin (C'_0 \cup A)$.

Таким образом, шаг $(A \cup \underline{B})$ возможен из C'_0 в $\mathcal{E} \setminus C'$. Следовательно, получаем $C'_0 \xrightarrow{(A \cup \underline{B})} C$ в $\mathcal{E} \setminus C'$ и $C = (C'_0 \setminus B) \cup A$.

Поскольку $\mathcal{E} \in c\mathcal{RPES}$, то по утверждению 1 истинно $C' \xrightarrow{(A \cup \underline{B})} C'''$ в $\mathcal{E}$ и $\mathcal{E} \setminus C''' = (\mathcal{E} \setminus C') \setminus C$. По определению 3, так как шаг $(A \cup \underline{B})$ возможен из C', то $C''' = ((C' \setminus B) \cup A)$, а значит конфигурации C'' и C''' совпадают. Таким образом, $\mathcal{E} \setminus C'' = (\mathcal{E} \setminus C') \setminus C$. □

Доказательство утверждения 3.

(а) Возьмем произвольную конфигурацию $C \in Conf(\mathcal{E})$. По определению 3, для каждого $1 \leq i \leq n$ существуют множества $A_i \subseteq E$ и $B_i \subseteq F$ такие, что $C_{i-1} \xrightarrow{(A_i \cup B_i)} C_i$ и $C = C_n$ $(n \geq 0)$. По этому же определению очевидно, что для каждого $0 \leq i \leq n$ верно: $C_i \in Conf(\mathcal{E})$. Тогда, поскольку $\mathcal{E} \in c\mathcal{RPES}$, благодаря лемме 2(б) для каждого $0 \leq i \leq n$ истинно, что $\mathcal{F}_i = \mathcal{E} \setminus C_i \in c\mathcal{RPES}$. Более того, так как $\mathcal{E} \in c\mathcal{RPES}$ и $C_i \xrightarrow{(A_{i+1} \cup B_{i+1})} C_{i+1}$ для каждого $0 \leq i < n$, то по утверждению 2 для каждого $0 \leq i < n$ верно, что $C^i_0 \xrightarrow{(A_{i+1} \cup B_{i+1})} C^i_1$ в $\mathcal{F}_i$ и $\mathcal{F}_{i+1} = \mathcal{F}_i \setminus C^i_1$. По определению 7 это означает, что $\mathcal{F}_i \xrightarrow{(A_{i+1} \cup B_{i+1})} \mathcal{F}_{i+1}$ для каждого $0 \leq i < n$. Таким образом, так как $\mathcal{F}_0 = \mathcal{E} \setminus C_0$ и $\mathcal{F}_n = \mathcal{E} \setminus C$, вновь по определению 7 получаем, что $\mathcal{E} \setminus C \in Reach(\mathcal{E})$.

(б) Возьмем произвольную $\mathcal{E}' \in Reach(\mathcal{E})$. Следовательно, по определению 7, существуют $\mathcal{E}_0, \ldots, \mathcal{E}_n$ $(n \geq 0)$ такие, что $\mathcal{E}_0 = \mathcal{E} \setminus C_0$, $\mathcal{E}_n = \mathcal{E}'$ и для каждого $0 \leq i < n$ верно: $\mathcal{E}_i \xrightarrow{(A_{i+1} \cup B_{i+1})} \mathcal{E}_{i+1}$. Воспользуемся методом математической индукции и покажем, что для любого $0 \leq i \leq n$ существует $C_i \in Conf(\mathcal{E})$ такая, что $\mathcal{E}_i = \mathcal{E} \setminus C_i$.

$i = 0$. Очевидно, $C_0 \in Conf(\mathcal{E})$ и $\mathcal{E}_0 = \mathcal{E} \setminus C_0$.

$i > 0$. По предположению индукции имеем, что $\mathcal{E}_i = \mathcal{E} \setminus C_i$ для некоторой конфигурации $C_i \in Conf(\mathcal{E})$. Покажем, что существует $C_{i+1} \in Conf(\mathcal{E})$ такая, что $\mathcal{E}_{i+1} = \mathcal{E} \setminus C_{i+1}$. Поскольку $\mathcal{E}_i \overset{(A_{i+1} \cup B_{i+1})}{\rightharpoonup} \mathcal{E}_{i+1}$, то по определению отношения $\rightharpoonup$ получаем, что $\mathcal{E}_{i+1} = \mathcal{E}_i \setminus C_1^i$ для некоторой $C_1^i \in Conf(\mathcal{E}_i)$ такой, что $C_0^i \overset{(A_{i+1} \cup B_{i+1})}{\rightarrow} C_1^i$ в $\mathcal{E}_i$. Тогда, так как $\mathcal{E} \in c\mathcal{RPES}$, $\mathcal{E}_i = \mathcal{E} \setminus C_i$ и $C_0^i \overset{(A_{i+1} \cup B_{i+1})}{\rightarrow} C_1^i$ в $\mathcal{E}_i$, то по утверждению 1 получаем, что $C_i \overset{(A_{i+1} \cup B_{i+1})}{\rightarrow} C_{i+1}$ в $\mathcal{E}$ и $\mathcal{E} \setminus C_{i+1} = \mathcal{E}_i \setminus C_1^i = \mathcal{E}_{i+1}$.

Таким образом, показали существование $C_n \in Conf(\mathcal{E})$ такой, что $\mathcal{E}' = \mathcal{E}_n = \mathcal{E} \setminus C_n$.

(в) Пусть конфигурации $C, C' \in Conf(\mathcal{E})$ выбраны так, что $C \overset{(A \cup \underline{B})}{\rightharpoondown} C'$. По определению отношения $\rightharpoondown$ это означает, что $C \overset{(A \cup \underline{B})}{\rightarrow} C'$ в $\mathcal{E}$. Согласно пункту (а) имеем, что $\mathcal{E} \setminus C,\ \mathcal{E} \setminus C' \in Reach(\mathcal{E})$. Так как $\mathcal{E} \in c\mathcal{RPES}$ и $C \overset{(A \cup \underline{B})}{\rightarrow} C'$ в $\mathcal{E}$, то по утверждению 2 истинно, что $C_0' \overset{(A \cup \underline{B})}{\rightarrow} C_1'$ в $\mathcal{E} \setminus C$ и $\mathcal{E} \setminus C' = (\mathcal{E} \setminus C) \setminus C_1'$. Однако, в соответствии с определением отношения $\rightharpoonup$, это означает, что $\mathcal{E} \setminus C \overset{(A \cup \underline{B})}{\rightharpoonup} \mathcal{E} \setminus C'$.

(г) Выберем произвольным образом $\mathcal{E}', \mathcal{E}'' \in Reach(\mathcal{E})$ так, чтобы $\mathcal{E}' \overset{(A \cup \underline{B})}{\rightharpoonup} \mathcal{E}''$. И пусть $\mathcal{E}' = \mathcal{E} \setminus C'$ для некоторой $C' \in Conf(\mathcal{E})$. Заметим, что по пункту (б) как минимум одна такая конфигурация существует. Согласно определению отношения $\rightharpoonup$, верно: $\mathcal{E}'' = \mathcal{E}' \setminus C_1'$, где $C_0' \overset{(A \cup \underline{B})}{\rightarrow} C_1'$ в $\mathcal{E}'$. Тогда, поскольку $\mathcal{E} \in c\mathcal{RPES}$, по утверждению 1, получаем, что $C' \overset{(A \cup \underline{B})}{\rightarrow} C''$ в $\mathcal{E}$ и $\mathcal{E} \setminus C'' = \mathcal{E}' \setminus C_1' = \mathcal{E}''$. Благодаря определению отношения $\rightharpoondown$ имеем, что $C' \overset{(A \cup \underline{B})}{\rightharpoondown} C''$. □

UDK 004.43

Programming operational semantics of programming languages

Anureev I. S. (A.P. Ershov Institute of Informatics Systems SB RAS)

The paper presents a method for describing the operational semantics of programming languages. It is based on a domain-specific language designed to specify executable specifications for programming language constructs. This language is an extension of the Lisp language. The peculiarity of the method is to set the semantics for the program model, and not for the program itself. Since the specifications are executable, the method actually allows to 'program' the semantics of programming languages. The method can be used in teaching computer languages, as well as in creating new programming languages, since it allows you to describe the constructions of a new language immediately at the level of an abstract syntactic tree.

Keywords: programming language, operational semantics, domain-specific language, abstract syntax tree, program model, Lisp

1. Introduction

One of the key methods to ensure the reliability of software is the use of formal methods. However, such use requires a formal definition of the program. One of these natural definitions is the operational semantics of programming languages.

The existing definitions of operational semantics can be divided into three groups. The first group consists of semantics presented manually in a natural or mathematical language. There is a huge list of works of this kind for specific programming languages. Here ([12], [10], [3], [8], [7], [4], [5], [17], [2], [6], [16], [13]) are some of them. The second group consists of semantics represented by program executors (implementations of programming languages). The third group consists of semantics encoded in formal machine systems ([15], [1], [18], [9], [14]).

All of these ways of representing operational semantics have a number of common disadvantages. First, even a small modification of the semantics requires efforts to avoid making mistakes when editing the representation. With a significant change in semantics, it is necessary to rewrite a significant part of its representation. Secondly, such representations of semantics cannot provide an understandable and readable general structure at the same time as the completeness of the analysis of all cases. Thirdly, these representations are not univer-

sal, capable of accurately preserving the structure of programs of a formalized programming language, in particular, due to the fact that the structure of the representation is rigidly defined. Fourth, these representations are practically unsuitable for describing the semantics of programs composed of constructs of several programming languages (A way of combination can be found in [11]). Fifth, these representations are difficult to adapt to a subset of the same language, given the specifics of this subset, which makes it possible to simplify the general semantics.

We present metalanguage approach to the development of formal semantics of programming languages which is able to largely cope with these disadvantages.

2. Metalanguage approach

The proposed approach has the following features:

1. It applies to program models, not to programs themselves.
2. It is fine-grained, i. e. for each type of program constructs it has its own type of models and, thus, can be applied in isolation to individual program constructs, building their operational semantics.
3. It is flexible, since it allows to develop operational semantics only for a fragment of a target programming language represented by a variety of acceptable types of program constructs or to develop a family of different operational semantics for the same target programming language.
4. Models of program constructs are presented in a unified way in a metalanguage of description of models.
5. The metalanguage also allow to present models of execution contexts of these constructs.
6. The metalanguage has the means to transform these models, thus allowing develop operational semantics as a sequence of transformations of models of program constructs and their execution contexts and reducing process of development of formal semantics of programming languages to writing programs in a metalanguage that implements corresponding sequences of transformations.
7. The model transformation means make operational semantics of these constructs executable.
8. This approach can be applied to programs that use combinations of several programming languages.

2.1. Metalanguage Design

Within the framework of the proposed metalanguage approach, the metalanguage language PSML, a syntactic extension of Common Lisp, was designed.

PSML contains one extra type *mt*. It describes a finite set of model languages, as well as a set of operations for each language from this set. A model language specifies either models of programming language constructs, or models of execution contexts of these constructs.

This type is extensible, which means that its contents, which include many languages, many types of constructs for each of these languages, and many value constructs for these types, can be expanded by special PSML-functions. Let us list these functions.

Let letters s, o, os (possible with indices) denote Lisp symbols, objects and object sequences, respectively. The initial empty content of type *mt* is set by function (*mt-empty*). Function (*mt-lan* s) adds a new model language s to type *mt*. Function (*mt-clan* s) makes the model language s the current language. Model execution is always performed in the context of the current model language.

Let l, t, and c denote a model language, type of models, and value constructor of a type of models, respectively.

Function (*mt-type* (c t) os_1 $*$ os_2 $**$ os_3) adds a new type t of models and value constructor c for this type to the current model language. Sequences os_1, os_2, and os_3 are called positional, attributing, and tagging components and specify three kinds of arguments of c called positional, attributing and tagging arguments, respectively.

Component os_1 specifies positional arguments of c. Its elements have the form (s v), where v specifies type and optional property of a positional argument, and s specifies an argument name (called a positional attribute), allowing access to the argument not only by its position, but also by name of this attribute. Object v can have one of the following forms t', (t' l), (TC t'), (TC (t' l)), (*opt* t'), (*opt* (t' l)), (*opt* (TC t')), and (*opt* (TC (t' l))) where s specifies argument type, i. e. the type of models that it can take as a value, l indicates that type t' is a type of models of language l, $CT \in \{list, map\}$ *list* indicates that values of the argument are lists of models of type t, *map* indicates that values of the argument are are finite mappings from symbols to type t, and *option* indicates that this argument is optional and can be skipped. If $l = lisp$, t' is a built-in lisp type name (for example, *symbol*).

Component os_2 specifies named arguments of c that are used to annotate models of type c. Its elements have the form (s v), where s specifies an argument name (called an annotating

attribute), and v specifies type of the argument and has the same form as in component os_1 except for the optional property.

Component os_3 specifies tagging arguments of c that are used to tag models of type c. Its elements have the form s, where s is called a tag.

Function (*mt-utype* t (t_1 ... t_n)) adds a new type t of models which is a union of types t_1, ..., t_n. Let us note that this type does not have its own value constructor.

Function (*mt-ltype* t s) adds a new type t of models which values match the values of the Lisp type s.

PSML has functions of access to contents of mt and attribute update in it.

Function (*mt-aget* m a) returns the value of attribute a of model m. Function (*mt-aset* m a o) assigns value o to attribute a of model m. These functions have generalized forms (*mt-aget* m a_1 ... a_n) and (*mt-aset* m a_1 ... a_n o) that allow you to access the value and assign the value according to the hierarchy of attributes a_1, ..., a_n, respectively.

Functions (*mt-type* m) and (*mt-aname* p) return a type of model m and a name of positional attribute in position p.

Function (*mt-is-model* m) checks whether m is a model of the current model language. Function (*mt-clan-get* m) returns the current model language.

Function (*mt-lans*) returns a current list of model languages. Function (*mt-mtypes* l) returns a list of types of models of language l. Function (*mt-mtypes*) returns a list of types of models of the current language.

Function (*mt-function* f ((l t x) y) o) specifies actions given by body o for function f of two arguments, model x of type t of construct language l and execution context y. If l is omitted, the current language is assumed.

There are variants of the above functions wit prefix *mt-g-* instead of *mt-*. These variants have additionally a model language as the first argument.

2.2. Approach Steps

The application of the proposed approach consists of the following four steps:

1. *To program the content of type mt corresponding to the target programming language L (its program constructs and execution contexts) in PSML.*
2. *To develop a translator of L-constructs to their primary models.* A primary model of a program construct is a model that directly represents the construct without adding

additional information (for example, as result of semantic analysis). A primary model corresponds an AST of the construct obtained as a result of syntactic analysis, with one exception that this tree is additionally semantically marked with attributes and concepts (types of models) within the terminology of the subject area (a programming language). This is the only step that is performed by an external tool (for example, a parser generator), but because of one-to-one translation, it is simple.

3. *To program a translator of primary models of L-constructs to their secondary models in PSML.* A secondary model is a result of enriching the primary model of a program construct with additional information. This result is achieved by adding new annotating attributes to the construct and their meanings, as well as adding new tags. At the same time, the positional component of the construct does not change, just as the old annotating attributes and tags do not change. Enriching program construct models with additional information allows to simplify operational semantics development. Specialized external tools (for example, static analysis tools) can be used to execute this step.
4. *To specify function op-sem for each program construct model in PSML.* Let us note that although the full power of Common Lisp can be used in defining such a function, experiments show that it is possible to identify a limited subset of functions sufficient to describe the semantics of typical programming languages, thus defining DSL (Domain-Specific Language) for the operational semantics development.

2.3. Example

In this section the proposed approach is applied to a 'programming language' consisting exactly of the constructs contained in the program fragment below.

```
int a = 1; int b = 2; int c; int* p = &c;
asm(".intel_syntax noprefix\n\t" // GAS directive
  "mov eax, %1\n\t"
  "add eax, %2\n\t"
  "mov %0, eax\n\t"
  :"=r"(c)
  :"r"(a), "r"(b)
  : "eax"
);
```

This fragment denoted further by PF illustrates combination of two languages: C and Assembler.

In the first step, the content of type *mt* is set. Due to lack of space, this type will immediately contain the specification of the primary and secondary models, while attributes and tags related to the secondary model will be highlighted in bold.

```
(mt-empty) ; innitial intialization of type mt
(mt-lan C) ; adding model language C for C–constructs
; specifying model types for C language
(mt-clan C) ; making C current
(mt-type (seq declaration-sequence) (seq (list declaration))
(mt-utype expression (integer variable derefence addressof))
(mt-ltype integer int)
(mt-ltype variable symbol)
(mt-type (* derefence) (pointer (symbol name)))
(mt-type (& addressof) (variable (symbol name)))
(mt-utype declaration (variable-declaration asm-insert))
(mt-type (var variable-declaration) (type type) (name (symbol lisp))
     (value (opt expression)) * (always-points-to-var (symbol lisp))
     ** non-shared )
(mt-type (asm asm-insert-declaration) (code Assembler))
(mt-utype type (simple-type pointer-type))
(mt-ltype simple-type symbol)
(mt-type (pointer pointer-type) (type type))
; specifying model types for Inline Assembler language
(mt-clan Assembler) ; making Assembler current
(mt-type (insert insert) (GAS (string lisp)) (code (list instruction))
     (outputs (list binding)) (inputs (list binding))
          (destructed (list (symbol lisp))) )
(mt-type (binds binding) (register-type (symbol lisp)) (var (symbol lisp)))
(mt-utype instruction (mov add))
(mt-type (mov mov) (first operand) (second operand))
(mt-type (add add) (first operand) (second operand))
```

```
(mt-utype operand (simple-operand index-operand))
(mt-ltype simple-operand symbol)
(mt-type (& index-operand) (index (nat lisp)))
```

After completing the second and third steps, the following secondary model of fragment PF is obtained:

```
(seq (var int a 1 ** non-shared) (var int b 2 ** non-shared) (var int c 2)
    (var (pointer int) p (& c) * (always-points-to-var c))
    (asm (insert (GAS ".intel_syntax noprefix")
        (code (mov eax (% 1)) (add eax (% 2)) (mov (% 0) eax))
        (outputs ((binds (register-type r) (var c))))
        (inputs ((binds (register-type r) (var a))
            (binds (register-type r) (var b))))
        (destructed (eax)))))
```

Tag *non-shared* specifies variables values of which can be accessed in fragment PF only through their names. Annotating attribute *always-points-to-var* specifies pointers that always point to the same variable and returns this variable. From the arrangement of these entities in this model, it follows that a simple memory model without addresses can be used in the execution context for PF.

The language *C-context* specifies the execution context for C constructs of PF and is defined as follows:

```
(mt-lan C-context) ; adding C-context language
; specifying model types for C-context language
(mt-clan C-context) ; making C-context current
(mt-type (context context) * (var-values (map (object lisp)))
    (var-types (map type)) (pointers (map (symbol lisp))) )
```

Thus, this context stores the values of variables in annotating attribute *var-values* and the variables through which pointers get values in annotating attribute *pointers*. An example of model of language *C-context* is given below:

```
(context * (var-values (map (a 1) (b 2) (c 3)))
    (var-types (map (a int) (b int) (c int) (p (pointer int))))
    (pointers (map (p c))) )
```

The language *Assembler-context* specifies the execution context for Assembler constructs of PF and is defined as follows:

```
(mt-lan Assembler-context) ; adding Assembler-context language
; specifying model types for Assembler-context language
(mt-clan Assembler-context) ; making Assembler-context current
(mt-type (context context) * (reg-values (map (object lisp)))
    (index-values (list (object lisp)))
    (free-GP-registers (list (symbol lisp))) )
```

This context stores the values of registers in annotating attribute *reg-values* and unallocated general-purpose registers in annotating attribute *free-GP-registers*. An example of model of language *Assembler-context* is given below:

```
(context * (reg-values (map (eax 1)))
    (index-values (nil 1 2))
    (free-GP-registers (ecx edx ebx esp ...)) )
```

In the fourth step, function *op-sem* is specified for each program construct model. The PSML code for two cases of specification of this function for C model of type *variable* corresponding to variable access construct in C and similar Assembler model of type *index-operand* corresponding to C variable access construct in Assembler are given below.

```
(mt-clan C)
(mt-function op-sem ((variable x) y)
    (if (symbolp (mt-aget y var-types x))
        (mt-aget y var-values x)
        (mt-aget (mt-aget y pointers x) var-values))
(mt-clan Assembler)
(mt-function op-sem ((index-operand x) y)
    (mt-aget y index-values (mt-aget x index)) )
```

3. Conclusion

A metalanguage model approach has been proposed that allows programming the operational semantics of programming languages in terms of these languages themselves, preserving the

structure of the source program at the model level and representing operational semantics as a code of transformation of the model in a metalanguage.

In the near future, it is planned to finally fix the design of the metalanguage, clarify the property lists of global variable *mt*, modeling the corresponding type *mt* and its contents, and implement the designed functions in the form of Lisp macros.

References

1. Attali I., Caromel D., Russo M. A formal executable semantics for Java // Proceedings of Formal Underpinnings of Java, an OOPSLA / Citeseer. — 1998. — Vol. 98.
2. Börger E., Schulte W. A programmer friendly modular definition of the semantics of Java // Formal Syntax and Semantics of Java. — 1999. — P. 353–404.
3. Camilleri J. An operational semantics for occam // International Journal of Parallel Programming. — 1989. — Vol. 18. — P. 365–400.
4. A compositional operational semantics for Java mt / Ábrahám E., de Boer F. S., de Roever W.-P., and Steffen M. // Verification: Theory and Practice: Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday. — 2003. — P. 290–303.
5. Coscia E., Reggio G. An operational semantics for Java // Ter¬ chical report, DISI, Univ. of Genova, Italy. — 1998.
6. da Silva Feitosa S., Ribeiro R. G., Du Bois A. R. Formal Semantics for Java-like Languages and Research Opportunities // Revista de Informática Teórica e Aplicada. — 2018. — Vol. 25, no. 3. — P. 62–74.
7. Drossopoulou S., Eisenbach S. Towards an operational semantics and proof of type soundness for Java // Formal Syntax and Semantics of Java. — 1998. — Vol. 1523.
8. An event-based structural operational semantics of multi-threaded Java / Cenciarelli P., Knapp A., Reus B., and Wirsing M. // Formal syntax and semantics of Java. — 1999. — P. 157–200.
9. Hartel P. H. LETOS–a lightweight execution tool for operational semantics // Software: Practice and Experience. — 1999. — Vol. 29, no. 15. — P. 1379–1416.
10. Maffeis S., Mitchell J. C., Taly A. An operational semantics for JavaScript // Programming Languages and Systems: 6th Asian Symposium, APLAS 2008, Bangalore, India, December 9-11, 2008. Proceedings 6 / Springer. — 2008. — P. 307–325.
11. Matthews J., Findler R. B. Operational semantics for multi-language programs // ACM

SIGPLAN Notices. — 2007. — Vol. 42, no. 1. — P. 3–10.

12. Matthews J., Findler R. B. An operational semantics for scheme1 // Journal of Functional Programming. — 2008. — Vol. 18, no. 1. — P. 47–86.
13. Ramananandro T. Mechanized Formal Semantics and Verified Compilation for C++ Objects : Ph. D. thesis ; Université Paris-Diderot-Paris VII. — 2012.
14. Van Tassel J. P. An operational semantics for a subset of VHDL // Formal Semantics for VHDL. — Springer, 1995. — P. 71–106.
15. Verdejo A., Martí-Oliet N. Executable structural operational semantics in Maude // The Journal of Logic and Algebraic Programming. — 2006. — Vol. 67, no. 1-2. — P. 226–293.
16. Wallace C. The semantics of the C++ programming language. — 1993.
17. The semantics of the Java programming language: Preliminary version : Rep. / Citeseer ; executor: Wallace C. : 1997.
18. Wesonga S. O. Javalite-An Operational Semantics for Modeling Java Programs. — Brigham Young University, 2012.

УДК 004.9

Построение онлайн системы с Web интерфейсом для хранения, обработки и анализа генетических последовательностей вируса SARS-CoV-2.

Старцев П.А. (Институт систем информатики СО РАН)

Работа с геномами, поиск в известной математически формализованной последовательности как можно более стабильной или максимальной по длине подцепочке, поиск, как математическая задача для программного обеспечения по различным другим критериям являются одними из самых актуальных задач при работе с современным инструментарием исследователя в области вирусологии. Спектр прикладных задач сегодняшнего времени включает в себя создание новых праймеров для диагностики вируса по стабильным участкам генома и выявления последних мутаций вируса, классификации и кластеризации накопленного материала для более точных исследований дальнейшей мутации и тому подобное. Построение вычислительных комплексов на основе базы данных и расширение их функциональности при помощи как онлайн технологий, так и запуска серверных приложений, является сложной практической задачей в сфере программирования для более эффективной работы вирусологов и специалистов из смежных областей в сфере диагностики и лечения вирусных заболеваний.

Ключевые слова: *SARS-CoV-2, коронавирус, онлайн система, генетическая последовательность, геном, ИСИ СО РАН.*

1. Введение.

Несмотря на стабилизацию ситуации с коронавирусом в России, актуальность работ по этому типу вирусов остается достаточно высокой: наблюдается дальнейшее распространение вируса на фоне снижения смертности [4], выявляются мутации, проверяются новые механизмы работы с математическим инструментарием [5] и подходы для диагностики и лечения остающегося опасным вируса и сопутствующих заболеваний [3]. Поскольку SARS-CoV-2 обладает одним из самых больших геномов среди всех выявленных РНК-вирусов (см. [1], с.19), очень подробно и на больших объемах данных разделен на белки по типам (S, E, M, и т.п. [1]), то с каждым днем становится все более актуальной задача поиска по фрагменту генома, возможность найти часть последовательности нуклеотидов, начиная с какой-либо позиции или внутри одного белка. Кластеризация участков генома и работа с конкретными

мутациями внутри кластера также требуют более тонкой настройки поиска последовательностей нуклеотидов внутри БД (см. [4], с.4). Длина последовательностей одного генома SARS-CoV-2 в базе данных – около 30000 нуклеотидов. Повышенный интерес с точки зрения создания вакцин и диагностических систем представляет из себя S (Spike) белок [2, с. 13]. Перечисленные выше возможности поиска - это несколько примеров функциональности, не представленной непосредственно средствами GENBANK, которые необходимо разрабатывать под нужды специалистов самостоятельно. Более того, к собственной базе данных можно строить обращения помимо систем поиска и программ, в этом случае гибкость запросов поиска ограничена лишь средствами языка обращения к БД SQL.

Целью работы является построение программной системы на основе собственной базы данных для работы как с геномом SARS-CoV-2, так и с его отдельными участками, а также с метаинформацией (сведения о возрасте пациента, лаборатории, в которой получена последовательность, дате вакцинации, и т.п.). В текущей, описанной в настоящей работе, конфигурации, реализована работа как с онлайн интерфейсом (скриншоты, см. рис.1 и далее), так и с дополнениями, обладающей следующими качествами:

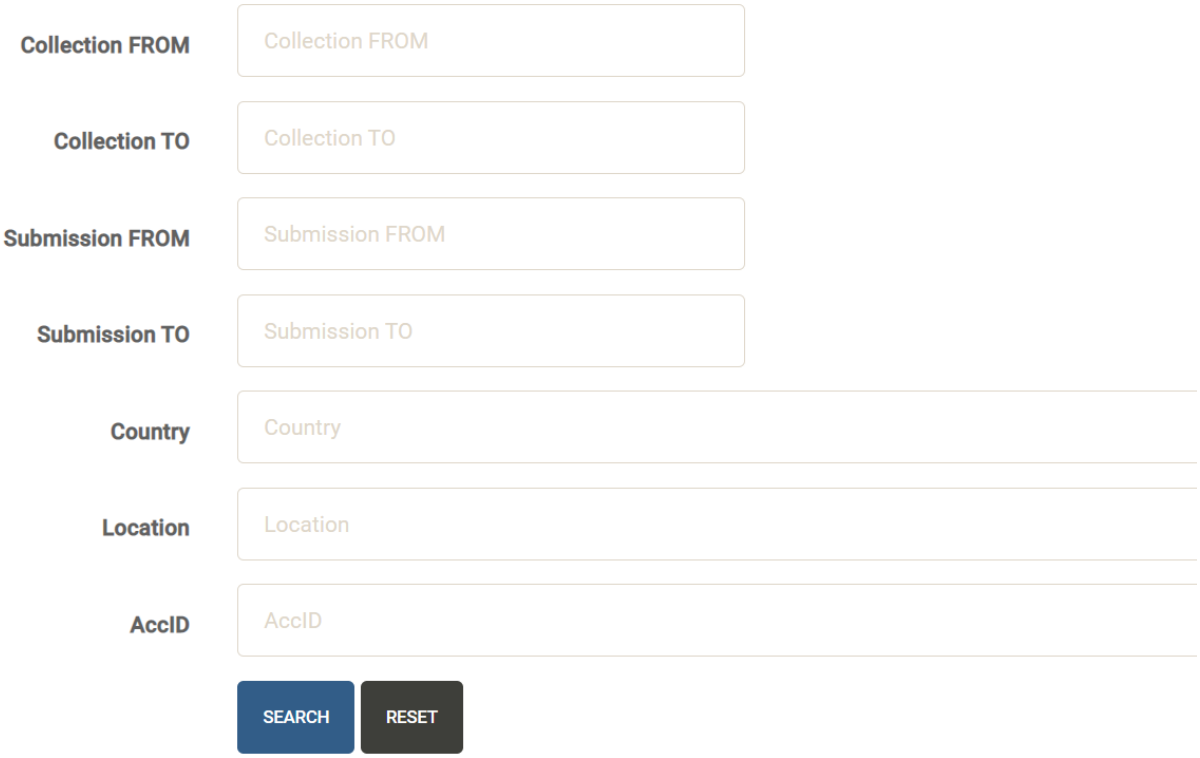

Рис.1 Внешний вид страницы поиска по параметрам системы. Язык интерфейса – английский.

- Возможность загрузки, первичной валидации/оценки и кластеризации/классификации генетических последовательностей из различных баз данных в виде больших файлов в формате fasta.
- Возможность запуска подпрограмм, необходимых для обработки как промежуточных результатов (например, результатов поиска), так и работы с единичными последовательностями.
- Развитие специфических задач программного обеспечения (далее ПО) системы для специалиста в области вирусологии.
- Накопление статистических данных и данных производительности ПО.

2. Построение и свойства системы.

2.1. Описание текущей конфигурации системы.

В настоящее время система включает в себя максимально возможную, сравнимую по объему базу данных с GENBANK по коронавирусу SARS-CoV-2 (оценки и сравнения приведены на середину 2023 года, см. п. 3.1). Больше о базе данных GENBANK (далее БДG) можно найти, например, здесь [8].

База GENBANK постоянно пополняется и является одним из доступных хранилищ данных, но в силу последних обстоятельств ограничительного свойства работы с международными хранилищами, постепенно возникает необходимость локализовать хранилище секвенированных последовательностей нуклеотидов с сохранением метаинформации. Также в БДG декларируются ограничения по валидации и правам собственности на последовательности геномов, что не позволяет научно использовать этот материал без локальной верификации, ссылки на ограничения проприетарного свойства и уточнения классификационных признаков[8]. Последовательности в БДG зачастую имеют процент нуклеотидов, которые явно не определены (помечены, как правило, символом N). Кроме того, извлечение данных в этом конкретном случае сопряжено с неудобством технического свойства при скачивании файлов большого объема. Поисковые механизмы сайта достаточно глубоко разработаны, но при некоторых критериях поиска, например по неточной дате секвенирования, есть достаточно серьезное несоответствие данных при составлении загрузочных файлов. Кроме того, согласно [8], в явном виде не разделяется информация о количестве поступающих новых цепочек нуклеотидов именно по SARS-CoV-2, а собирается более общая статистика, что не позволяет точно оценить динамику обновления данных именно в интересующем исследователей сегменте данных по гранту и

работе с SARS-CoV-2 в целом. Также есть список ошибок на официальном сайте в разделе validations [8].

Была создана локальная база данных (далее БД) Postgres SQL версии 14. Выбор версии и типа БД производился по следующим критериям:

- Потенциальный переход на отечественный продукт
- Минимизация проприетарных рисков
- Открытость кода, лицензирование, допускающее гибкую работу с ядром ПО и обслуживанием версий кода
- Возможность работы с большими объемами данных
- Бесплатная обновляемость и доступная система сопровождения ПО
- Безопасность, в том числе с точки зрения современных вызовов санкционного характера

В качестве серверного решения используется пакет открытого ПО Spring, язык программирования JAVA[6]. В качестве Web слоя используются библиотеки с открытым исходным кодом для современной линейки браузеров на базе Javascript [8,9]. Конфигурация системы управления БД, комплекс серверных технологий, конкретных слоев серверного ПО и подходов к разработке будут рассмотрены в отдельной статье.

2.2. Запуск подпрограмм: BLASTN.

В системе есть возможность из Web интерфейса запускать подпрограммы уровня операционной системы. Для этого используется механизм потоков JAVA [6]: Создается объект-процесс, указываются его параметры запуска, и, после выполнения, проверяется результат, возвращаемый операционной системой. Есть возможность логирования как работы самой программы, так и контекста запуска для отладки и контроля. (см. рис. 5 раздела 2.4)

Семейство программ BLASTN [5] представляет из себя набор библиотек и файлов запуска с консольным интерфейсом. На вход программе в системе передаются:

- Файл с последовательностями в формате fasta (одной или более) нуклеотидов для сравнения
- Файл, сформированный в формате fasta [7] с последовательностями нуклеотидов, отобранными по критериям поиска из системы. Результаты текущей конфигурации поиска можно сохранить в файл локальной системы в формате fasta и передать на обработку как параметр запуска
- Файл результата. В этот файл консольная программа выведет результат работы.

- Дополнительные параметры, которые можно настраивать в системе (формат вывода данных, столбцы таблиц информации, ограничения вывода и т.п.)

Возможен запуск других аналогичных программ с консольным выводом.

BLAST (Basic Local Alignment Search Tool) как семейство программ, создано для работы с нуклеотидными последовательностями, с базами данных секвенированных геномов и их участков [7]. В нашем случае подпрограмма BLASTN является одним из важнейших инструментов семейства для поиска относительно коротких участков последовательности или одного участка последовательности с небольшими отклонениями относительно изучаемой последовательности. Формат вывода результата может быть настроен при вызове и может содержать как сравниваемые участки последовательности, так и общую информацию в виде списка дополнительной метаинформации исследуемых сравнений. Исследователя может интересовать, например, страна происхождения генома, тип вируса и т.п.[5,7].

До создания системы приходилось подготавливать файлы, которые передаются на вход работе программы BLASTN вручную. Особое неудобство вызывает при таком методе работы формирование файла сравнения (аналога данных из БД), который может быть слишком большим по объему для работы с основными используемыми текстовыми редакторами. Вместо такого подхода можно использовать раздел поиска WEB интерфейса (см. рис. 1) и возможности системы сериализации непосредственно в файл текстового формата.

2.3. Web интерфейс

Web интерфейс (см. рис. 1-6) вместо локального приложения был выбран по следующим критериям:

- Возможность работы не только локально, но и удаленно.
- Поиск через Web интерфейс на сегодняшний день включает в себя поля дат (секвенирования и передачи в исходную базу данных), локации (как страны, так и более узкой области внутри страны), идентификатора записи и подстроки последовательности.
- Использование локальной программы сужает круг разработчиков самой специфической части интерфейса, возникают риски развитию системы – для дальнейшей разработки собственного специфического интерфейса для данного языка программирования и библиотек может не найтись разработчика в дальнейшем.

- Отдельная часть интерфейса локальной программы так или иначе дублирует соответствующие уже готовые части браузера, что повышает время разработки и вероятность ошибок в коде, который и не нужно было создавать.

Для создания Web интерфейса (см. рис 2), используются технологии Javascript библиотек с открытым кодом и библиотека шаблонов THYMELEAF[9].

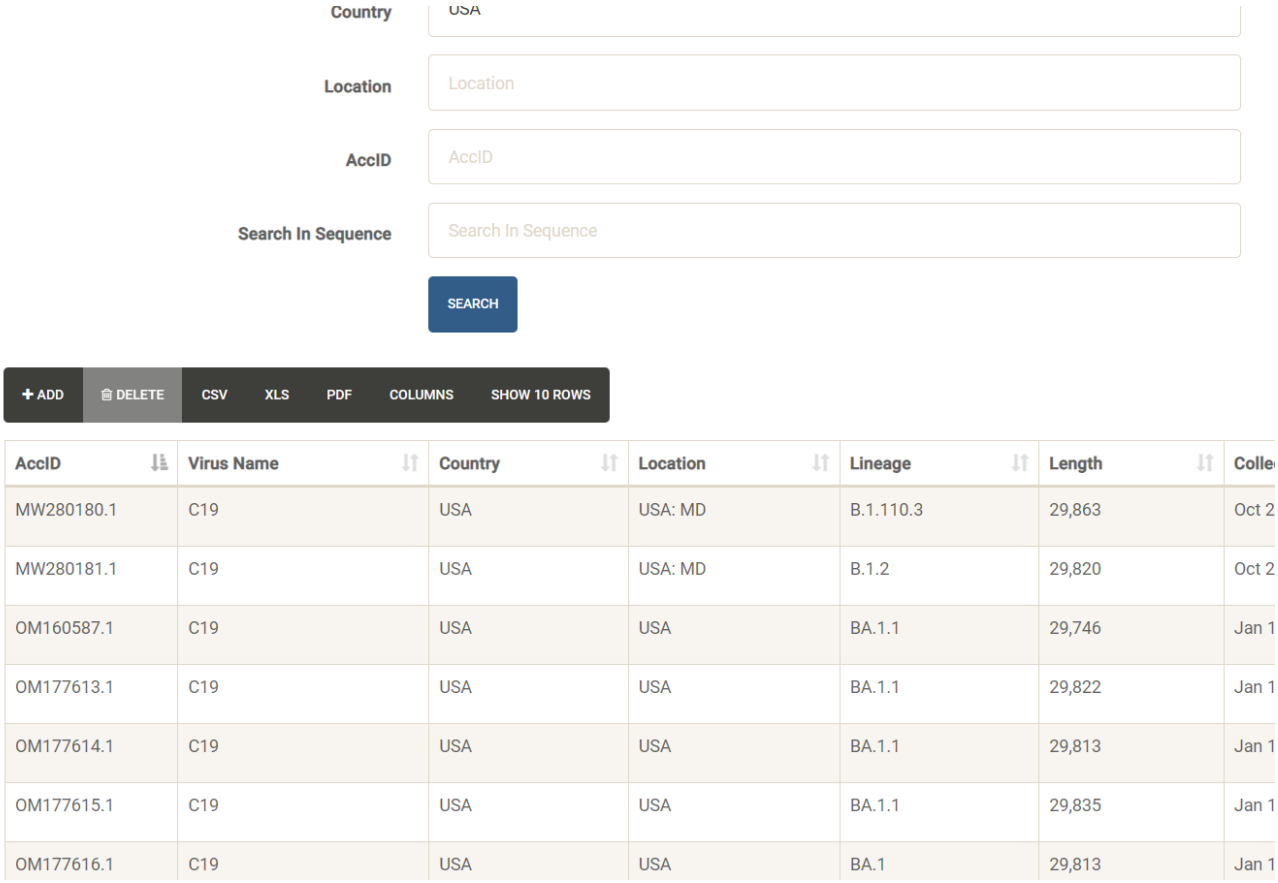

Рис.2 Результаты поиска по стране и более узкой локации внутри страны. Кнопки в столбце Tools позволяют перейти на страницу нуклеотидной последовательности (см. рис 3,4). Скриншот с браузера Опера.

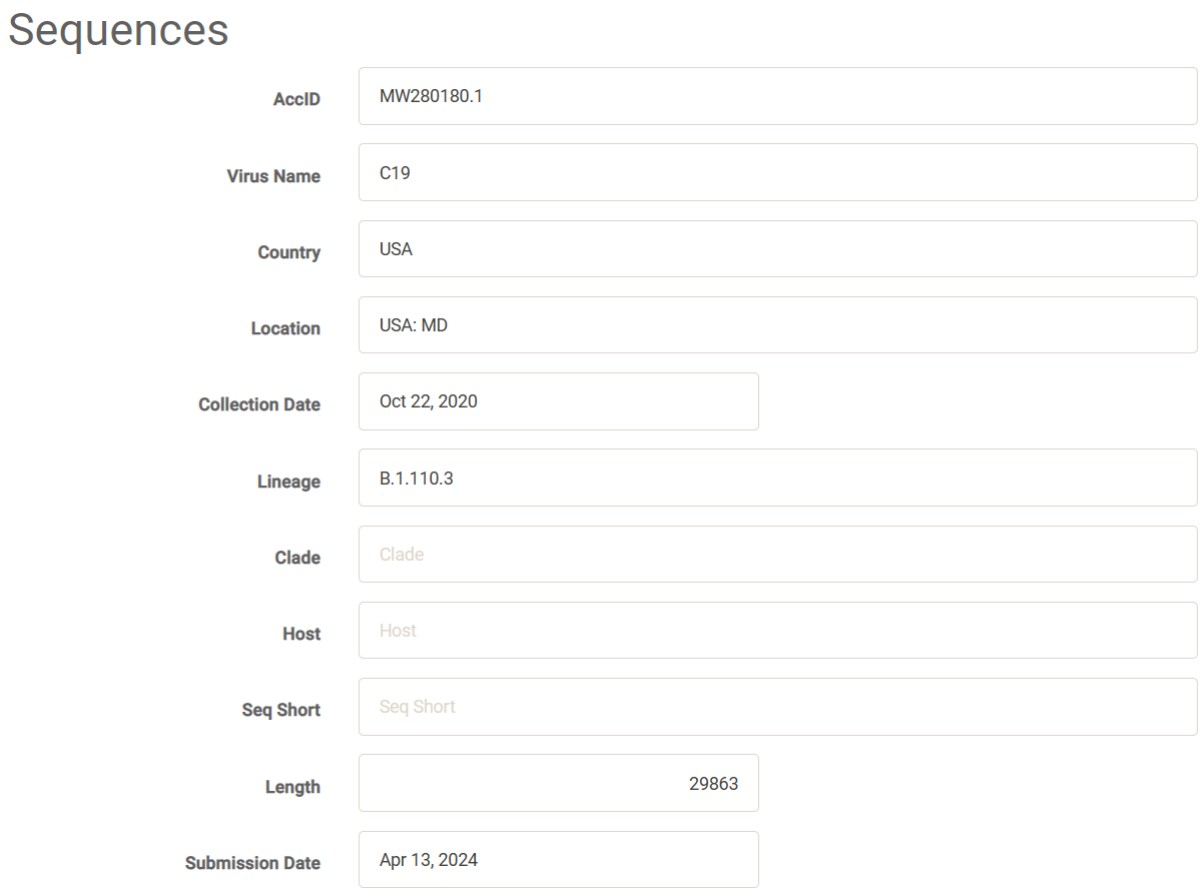

Рис.3 Страница с полной информацией о генетической последовательности из базы данных (Начало).

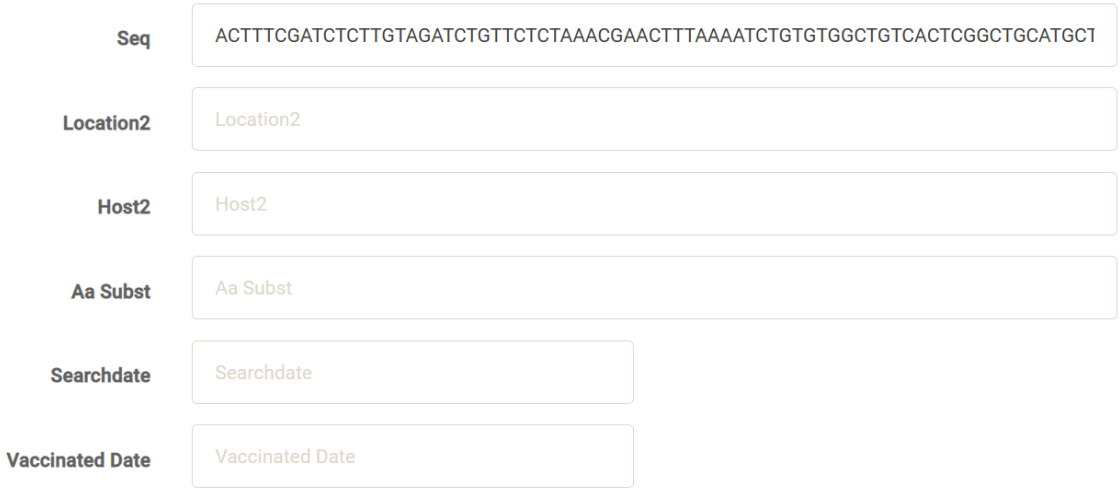

Рис.4 Страница с полной информацией о генетической последовательности из базы данных(Окончание). Из поля «Последовательность» (Seq) можно скопировать саму последовательность для дальнейшей работы.

2.4. Процесс и результаты запуска подпрограммы BLASTN

Для запуска программ в системе удобно пользоваться сериализацией результатов поиска в файлы, удобные для чтения и работы с текстовыми редакторами. Специально для этого существует возможность сериализации, в том числе, и в формат fasta[5].

После запуска через Web интерфейс (см. раздел 2.2) программа blastn начинает работу при помощи одного из исполняемых файлов (возможен запуск через файлы операционных систем Windows или Linux, настраивается при сборке системы).

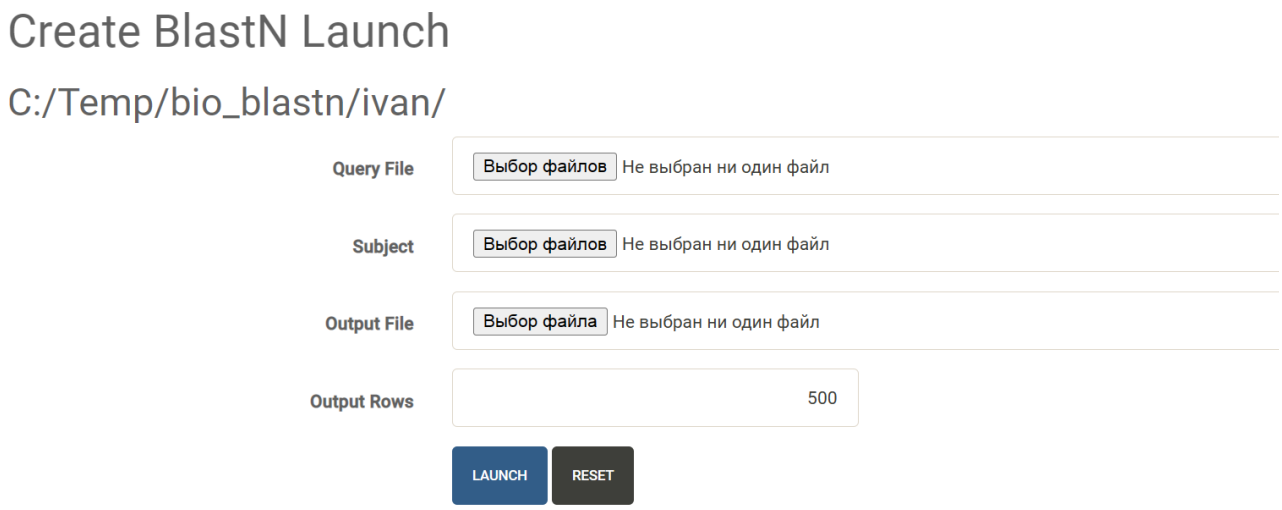

Рис.5 Страница запуска подпрограммы BLASTN. Есть возможность работать в собственной директории, указываемой в настройках.

Работа с blastn возможна и вне Web интерфейса.

После запуска программа переходит на страницу статистики (результатов) запусков (рис.6). Есть возможность поиска по дате, контекстный поиск, например, по году или части названия файла. Формат выходящих данных программы blastn является одной из настроек (0-11) и может содержать, например, таблицу данных без комментариев (значение 6, настроено в системе). Настройки столбцов могут содержать статистику совпадений/несовпадений участков генома, идентификаторы последовательностей, специальные функции и т.п., всего несколько десятков опциональных выходных данных, разделенных пробелом.

В качестве одного из направлений развития системы может быть рассмотрена база специального формата, непосредственно предназначенная для работы с blastn и создаваемая при помощи этой же командной строки. Это может быть и временный объект, наполняемый данными под один поиск, а также специальное хранилище для постоянных обращений под

конкретную задачу, наполняемый из Web интерфейса или по настраиваемой задаче (скрипту). Создается как файл в файловой системе[5].

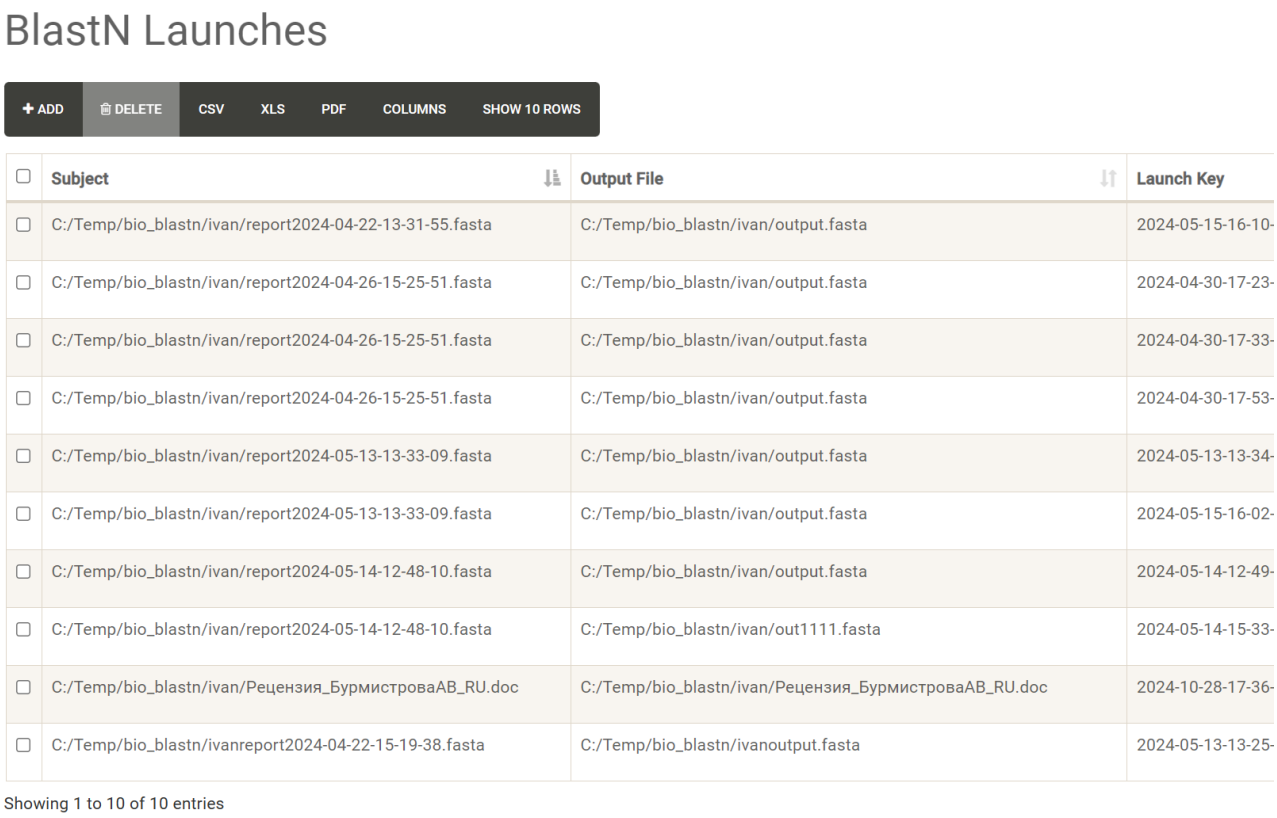

Рис.6 Страница статистики запусков подпрограммы BLASTN.

3. Актуальность, статистика, планы по развитию системы.

3.1. Актуальность проекта в цифрах и преимущества работы.

Создание базы данных и наполнение ее последовательностями генома и метаинформацией за несколько лет наблюдения позволило собирать статистику и обращаться к этим данным через удобные инструменты работы с языком запросов SQL вместо работы с большими файлами. Скорость поиска по базе данных (на точное соответствие по идентификатору, по датам периода, по частичному соответствию локации) не превышает нескольких секунд, что сравнимо с поиском по данным GENBANK на сайте, однако пока уступает по функциональности. При этом преимущество системы в том, что можно сразу использовать результаты выборки вместо скачивания файла с удаленного ресурса. Происходит сохранение набора последовательностей в fasta файл, готовый к работе

с другим инструментарием, причем приемлемого размера, например, легко редактируемый в привычных для работы редакторах.

Сохранены механизмы загрузки данных из GENBANK в систему для обновления данных со временем. Доступна статистика по загруженным данным, в том числе она может выдаваться по запросам к БД: как агрегатных функций, так и аналитических. Данные проверены на соответствие полученной информации по полям БД и прочим ошибкам (могут быть допущены некоторые неточности при выборе загрузки с сайта GENBANK) на тестах.

	records	country
1	3570727	USA
2	1857744	United Kingdom
3	800717	Germany
4	102676	Denmark
5	86051	Switzerland
6	46398	France
7	26885	Bahrain
8	23807	Slovakia
9	10469	Japan
10	9462	Iceland

Рис.7 Общее количество записей в Базе Данных по странам в порядке убывания, 10 первых позиций.

3.2. Текущее состояние и планы по развитию системы.

Как сказано выше, на данный момент в системе реализован функционал по поиску последовательностей в БД, сохранению результатов поиска в файл для дальнейшей работы в удобном текстовом формате, ряд настроек, запуск консольных приложений, логирование и обработка ошибок. Данные, подгруженные из GENBANK, проверяются на соответствие требованиям работы, доступны дальнейшие загрузки и ведется работа по улучшению тестирования этих процессов.

Планы на развитие включают в себя:

- Дополнение механизмов поиска
- Разделение функционала системы между пользователями, локальные настройки, например, последовательности, доступные только одному исследователю.

- Разделение функционала на открытый (публичный) и закрытый (доступный только локальным пользователям). Работы по безопасности контента от внешних угроз и источников.
- Развитие БД: Возможности загрузки данных из других источников. Загрузка метаинформации отдельно от последовательностей.

4. Благодарности.

Исследование выполнено за счет гранта Российского научного фонда (проект №23-64-00005).

Список литературы

1. Бруякин С.Д., Макаревич Д.А. Структурные белки коронавируса SARS-COV-2: роль, иммуногенность, суперантигенные свойства и возможности использования для терапевтических целей // Вестник ВолгГМУ. 2021. Вып. 2(78). С. 18-27.
2. Воробьев П. О., Тиллиб С. В. Однодоменное антитело для связывания консервативного эпитопа рецептор-связывающего домена белка Spike коронавируса SARS-COV-2 // Вестник РГМУ. 2023. №1. С.12-21.
3. Гарафутдинов Р.Р., Мавзютов А.Р., Никоноров Ю.М., Чубукова О.В., Матниязов Р.Т., Баймиев Ан.Х., Максимов И.В., Мифтахов И.Ю., Халикова Е.Ю., Кулуев Б.Р., Баймиев Ал.Х., Чемерис А.В. Бетакоронавирус SARS-CoV-2, его геном, разнообразие генотипов и молекулярно-биологические типы борьбы с ним. // Биомика. 2020. Т.12(2). С. 242-271.
4. Краснов Я.М., Попова А.Ю., Сафронов В.А., Федоров А.В., Баданин Д.В., Щербакова С.А., Кутырев В.В. Анализ геномного разнообразия SARS-Cov-2 и эпидемиологических признаков адаптации возбудителя COVID-19 к человеческой популяции // Проблемы особо опасных инфекций. 2020. №3. С.70.
5. Altschul S.F., Gish W., Miller W., Myers E.W., Lipman D.J. Basic Local Alignment Search Tool // Journal Of Molecular Biology. 1990. Vol. 215. N 3. P. 403-410.
6. Get Java for desktop applications. Официальная информация [Электронный ресурс]. URL: https://www.java.com/ru/ (дата обращения: 19.09.2024).
7. Hu G., Kurgan L., Sequence Similarity Searching // Current Protocols In Protein Science. 2019. Vol.95. N 1. P. e71.
8. What is GenBank? Официальная информация [Электронный ресурс]. URL: https://www.ncbi.nlm.nih.gov/genbank/ (дата обращения: 20.09.2024).
9. What Spring can do. Официальная информация [Электронный ресурс]. URL: https://spring.io/ (дата обращения: 20.09.2024).

УДК 004.056.57, 004.7, 004.8

Unveiling Malicious Patterns: Autoencoder-Based Malware Detection

Baghirov E. (Institute of Information Technology, Ministry of Science and Education of Republic of Azerbaijan),

Afzal M. (National University of Computer and Emerging Science)

Malware detection poses a significant challenge in cybersecurity, particularly with the increasing sophistication of attack methods. This study introduces an autoencoder-based approach to detect malware by learning the structure of benign data and identifying anomalies through reconstruction loss. By focusing on the detection of deviations in data patterns, this method offers an effective solution for identifying both known and unknown malware. Using the MALIMG dataset, the approach is evaluated with standard metrics such as accuracy, precision, recall, and F1-score, demonstrating strong performance and computational efficiency. This work highlights the potential of autoencoders as a robust anomaly-based detection tool.

Keywords: *malware detection, autoencoder, anomaly detection, reconstruction loss, cybersecurity, malware classification, machine learning, malware analysis*

1. Introduction

Malware, short for malicious software, refers to programs intentionally designed to cause harm to computer systems, steal sensitive information, or disrupt operations [1]. The rapid evolution of malware, with increasing sophistication and diversity, has made traditional detection techniques, such as signature-based approaches, less effective. Signature-based methods rely on known patterns to identify threats, leaving systems vulnerable to novel or polymorphic malware that can easily evade detection. This has necessitated the exploration of advanced detection techniques that can identify malware based on its behavior or anomalies in data patterns [2,3].

Recent advancements in machine learning and deep learning have provided promising alternatives to traditional methods for malware detection [4]. Unlike static signature-based techniques, these methods learn patterns from large datasets, enabling the detection of both known and unknown threats [5]. Among these, anomaly-based approaches have garnered attention for their ability to identify deviations from normal system behavior. Autoencoders, in particular, are well-suited for this task as

they are designed to learn compact representations of benign data and highlight anomalies that deviate from these learned patterns.

In this study, we explore the potential of autoencoders for malware detection. By training the autoencoder exclusively on benign samples, the model learns to reconstruct normal data patterns with minimal error. Malware, being anomalous, exhibits higher reconstruction loss, allowing it to be effectively identified. This reconstruction loss serves as the core metric for distinguishing between benign and malicious samples.

The proposed method is evaluated using the MALIMG dataset [6], a well-known benchmark dataset for malware detection. Our experimental results demonstrate the effectiveness of the autoencoder in detecting malware with high accuracy and efficiency. This work also addresses key challenges such as class imbalance and model evaluation using a variety of performance metrics, including accuracy, precision, recall, and F1-score.

The remainder of this paper is structured as follows:

- Section 2 reviews related work, highlighting existing approaches and gaps in malware detection research.
- Section 3 details the methodology, including dataset preparation, autoencoder architecture, and evaluation metrics.
- Section 4 presents the experimental results and analysis.
- Finally, Section 5 concludes the paper and outlines future research directions aimed at improving the precision and robustness of the proposed approach.

2. Related works

The method proposed by Xing et al. [4] employs autoencoders in conjunction with grayscale malware image representations to detect malicious software. The approach involves converting the bytecode of Android malware and benign applications into grayscale images, which serve as input for an autoencoder network. The model leverages the reconstruction error of these images to differentiate between malware and benign samples. However, the method has some limitations. The preprocessing stage of converting bytecode into grayscale images can introduce redundancy and inefficiency, potentially impacting the robustness of the model. Additionally, the reliance on grayscale image representation may restrict the applicability of the approach to certain types of malware, potentially missing important features that could be captured using other representations or techniques.

Panchagnula et al. [7] proposed a deep learning-based method for malware detection using autoencoders, where malware samples are transformed into grayscale images for feature extraction

and classification. The methodology involves two autoencoder models: AE-1, which assesses the feasibility of representing software features using grayscale images, and AE-2, which focuses on classifying malware from benign software. AE-1 uses unsupervised learning for feature extraction, while AE-2 integrates supervised learning with additional layers for classification. The model's performance was evaluated on multiple datasets containing various malware types, achieving high accuracy and F1-scores. The approach has limitations, including reliance on grayscale representations that may overlook complex malware behaviors, high computational costs for training autoencoders, and potential challenges in generalizing to unseen malware variants. These factors highlight the need for further enhancements, such as dynamic analysis or ensemble methods, to improve detection capabilities.

In a study by Halim et al. [8], the authors explored the use of recurrent neural networks (RNNs) for malware detection. By combining Long Short-Term Memory (LSTM) networks with Convolutional Neural Networks (CNNs), the model was designed to address both spatial and temporal challenges in classifying malware. This hybrid approach aimed to improve detection accuracy by capturing intricate patterns in malware behavior over time. However, despite its promising results, the model faced challenges related to high computational complexity and the need for large, labeled datasets for effective training.

In another work, MeMalDet by the authors [9] utilizes deep autoencoders for feature extraction combined with a stacked ensemble of supervised classifiers to detect malware through memory analysis, addressing limitations of traditional static and dynamic techniques. While the method achieves high accuracy (98.82%) and incorporates temporal evaluations for realistic testing, its reliance on computationally intensive memory analysis and a lack of adaptability to rapidly evolving malware behaviors remain significant limitations.

3. Methodology

This section outlines the approach employed for malware detection using autoencoders, detailing the dataset preparation, model architecture, training process, and evaluation criteria.

Dataset description. The dataset used in this study is the MALIMG dataset [6], a benchmark dataset commonly employed in malware detection research. It comprises grayscale images generated from the binary files of malware samples, representing 25 distinct malware families. These images are constructed by mapping the byte sequences of binary files into pixel values, creating visual representations that retain the structural information of the original binaries. This transformation allows machine learning models to leverage image-based analysis for malware detection.

The dataset is split into training, validation, and testing sets, with a stratified partitioning approach to preserve class distributions. During training, the autoencoder is exposed only to benign samples, enabling it to learn the structure of normal data. Malware samples are introduced during testing to evaluate the model's ability to detect anomalies based on reconstruction loss.

Table 1 provides an overview of the class distribution in the MALIMG dataset, listing all malware classes and their respective sizes.

TABLE 1. Class Distribution in the MALIMG Dataset.

Malware Class	Class Size	Malware Class	Class Size
Adialer.C	122	Lolyda.AA2	184
Agent.FYI	116	Lolyda.AA3	123
Allaple.A	2949	Lolyda.AT	159
Allaple.L	1591	Malex.gen!J	136
Alueron.gen!J	198	Obfuscator.AD	142
Autorun.K	1060	Rbot!gen	158
C2LOP.gen!g	200	Skintrim.N	80
C2LOP.P	146	Swizzor.gen!E	128
Dialplatform.B	177	Swizzor.gen!I	132
Dontovo.A	162	VB.AT	408
Fakerean	381	Wintrim.BX	97
Instantaccess	431	Yuner.A	800
Lolyda.AA1	213		

Autoencoder Architecture. An autoencoder is an unsupervised neural network designed to learn compressed representations of data by reconstructing the input as accurately as possible through an encoder-decoder architecture [7,9,10,11]. The encoder compresses input data into a latent representation, capturing its essential features, while the decoder reconstructs the original data from this compressed representation [12]. By minimizing reconstruction loss, such as Mean Squared Error, the autoencoder effectively models the structure of the input data. In malware detection, autoencoders are trained exclusively on benign samples to learn their normal patterns. During testing, malicious data, being anomalous, results in higher reconstruction loss, enabling the detection of novel or zero-day malware without relying on predefined signatures. This anomaly-based detection makes autoencoders a powerful and adaptive tool in cybersecurity. Figure 1 illustrates the architecture of an autoencoder-based anomaly detection system.

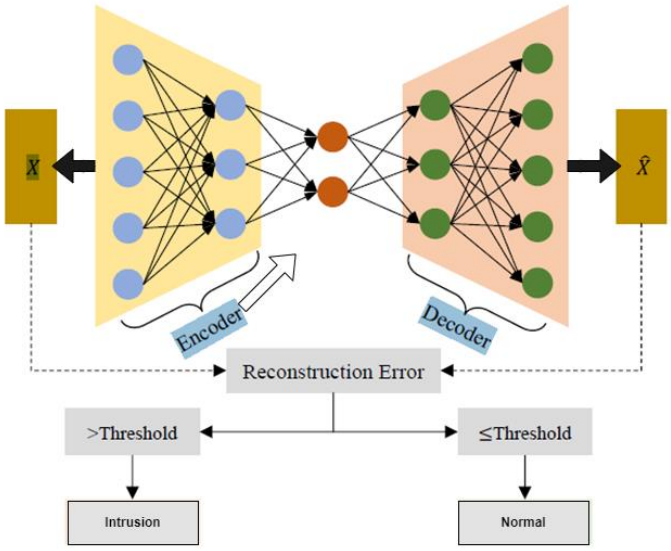

Figure 1. General architecture of autoencoder.

Evaluation metrics. To assess the performance of the proposed autoencoder-based malware detection approach, we employ a range of evaluation metrics commonly used in classification tasks. These metrics provide a comprehensive view of the model's effectiveness and its ability to generalize to unseen data. The metrics used in this study are as follows:

Accuracy: This metric measures the proportion of correctly classified samples out of the total number of samples. It is defined as:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FN + FP)}$$

Precision: Precision evaluates the model's ability to correctly classify positive (malicious) samples, minimizing the occurrence of false positives. It is defined as:

$$Precision = \frac{TP}{(TP + FP)}$$

Recall (Sensitivity): Recall measures the model's capability to identify all positive (malicious) samples. It is particularly important in malware detection, where missing a malicious sample can have severe consequences. It is defined as:

$$Recall = \frac{TP}{(TP + FN)}$$

F1-Score: This metric provides a balance between precision and recall, especially useful when the dataset is imbalanced. It is the harmonic mean of precision and recall, defined as:

$$F1 - score = \frac{2 * precision * recall}{(precision + recall)}$$

where TP, TN, FP, and FN denote True Positives, True Negatives, False Positives, and False Negatives, respectively.

Reconstruction Error: The reconstruction error is used to classify samples as benign or malicious. It is calculated as the difference between the input and the reconstructed output. A predefined threshold is used to separate normal samples from malware.

4. Experimental results and analysis

This section presents the experimental setup, results, and analysis to evaluate the performance of the proposed autoencoder-based malware detection method, highlighting its effectiveness and limitations.

Enviromental setup. For the experiments conducted in this research, the computational setup included a Linux-based system, which was chosen for its stability and compatibility with various machine learning frameworks. The system was equipped with 32GB of RAM, providing ample memory for handling large datasets like Malimg and training complex models such as GANs and autoencoders. The GPU used was an NVIDIA RTX 3090, which offers 24GB of dedicated VRAM and is optimized for high-performance deep learning tasks, significantly accelerating the training process for both generative models and autoencoders. The CPU was an Intel Core i9-13900K, a 24-core processor with 32 threads, ensuring fast and efficient handling of the data preprocessing tasks and other CPU-bound operations. This combination of hardware components allowed for the efficient execution of all computationally intensive tasks, minimizing the training time and enabling seamless experimentation. The setup was further supported by robust software tools, including popular deep learning library i.e PyTorch, which were leveraged for model development and training.

Experiments and results. During inference, a sample is classified as malware or benign based on its reconstruction loss. We experimented with several threshold values to find the optimal threshold that maximizes classification performance. After extensive testing, a threshold of 0.95 was found to produce the best results. If the reconstruction loss of an input image is below 0.95, it is classified as malware (label 0). If the reconstruction loss is above 0.95, it is classified as benign (label 1). We evaluated our model using the test set, calculating the confusion matrix and various performance metrics, including precision, recall, and F1-score. Figure 2 shows the training and validation loss curves over 250 epochs, indicating that the model converges effectively. Figure 3 presents the confusion matrix for the binary classification task, with 0 representing malware and 1 representing benign samples.

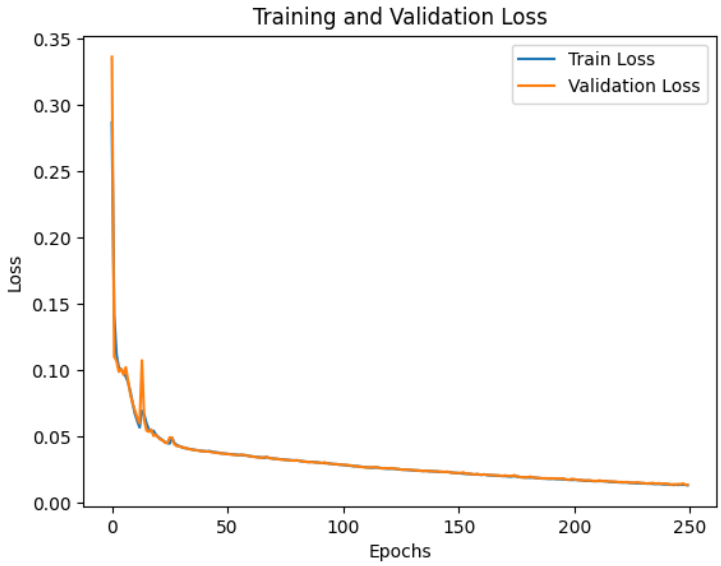

Figure 2. Training and Validation Loss Curve for Autoencoder Model.

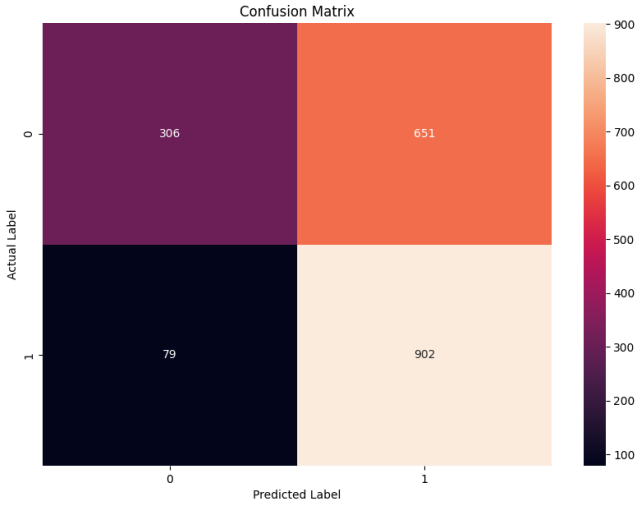

Figure 3. Confusion matrix for Autoencoder Model.

Based on the confusion matrix, the model achieved an accuracy of 62.3%, correctly classifying 1208 out of 1938 total samples. The precision, which measures the proportion of correctly identified malicious samples out of all predicted malicious samples, was 58.1%. The recall, indicating the model's ability to identify all actual malicious samples, was high at 91.9%. In malware detection tasks like this, recall is particularly crucial as failing to identify malicious samples (false negatives) can lead to severe security risks. The F1-score, which balances precision and recall, was 71.2%, reflecting the model's strong performance in detecting malicious samples despite a notable rate of false positives. These results suggest the model is highly sensitive to detecting malware but requires refinement to reduce false positives and improve overall precision.

Conclusion

In this study, we presented an autoencoder-based approach for malware detection, leveraging reconstruction loss to identify anomalies indicative of malicious activity. The results demonstrate the model's ability to achieve high recall (91.9%), indicating its effectiveness in detecting malware, including novel or unknown threats. Study confirms that autoencoders are a promising tool for anomaly-based malware detection, offering a scalable and signature-independent alternative to traditional methods.

To enhance the proposed autoencoder-based approach for malware detection, several directions for future research can be explored. Improving precision is a key priority, as reducing false positives will increase the model’s practicality in real-world applications. This can be achieved by incorporating additional benign data or combining autoencoders with supervised classifiers to create a hybrid detection framework. Expanding the model to include dynamic features, such as API calls, network activity, or runtime behavior, can provide a more comprehensive analysis of malware characteristics.

References

1. Baghirov E. A comprehensive investigation into robust malware detection with explainable AI // *Cyber Security and Applications*. 2025. Vol. 3. Article ID: 100072. ISSN 2772-9184. DOI: 10.1016/j.csa.2024.100072.
2. Baghirov E. Comprehensive framework for malware detection: Leveraging ensemble methods, feature selection, and hyperparameter optimization // *IEEE 17th International Conference on Application of Information and Communication Technologies (AICT)*. Baku, Azerbaijan, 2023. P. 1–5. DOI: 10.1109/AICT59525.2023.10313179.
3. Baghirov E.O. Analyzing the performance of behavioral-based malware detection approaches under real-world conditions // *Optical-Electronic Devices and Devices in Pattern Recognition and Image Processing Systems: Collection of Materials of the XVII International Scientific and Technical Conference*. Kursk, Russia, September 12–15, 2023. P. 15–17. Recognition - 2023 Kursk. South-West State University.
4. Xing, Xiaofei & Jin, Xiang & Elahi, Haroon & Jiang, Hai & Wang, Guojun. A Malware Detection Approach Using Autoencoder in Deep Learning. IEEE Access. 2022. Vol. 10. P. 25696-25706. DOI: 10.1109/ACCESS.2022.3155695.
5. Lee J., Lee J. A classification system for visualized malware based on multiple autoencoder models // *IEEE Access*. 2021. Vol. 9. P. 144786–144795. DOI: 10.1109/ACCESS.2021.3122083.

6. Lakshmanan Nataraj, S Karthikeyan, Gregoire Jacob, and BS Manjunath. Malware images: visualization and automatic classification. In Proceedings of the 8th international symposium on visualization for cyber security. ACM, 4, 2011.
7. Panchagnula V.M., N.V.L. Satya Keerthi Ch., Surekha S., Sujatha R., Veeraiah D., Ramesh E., Lakshmi B. A deep learning approach for detecting malware using autoencoder // *IAENG International Journal of Computer Science*. 2024. Vol. 51, No. 8. P. 1051–1059.
8. Halim M.A., Abdullah A., Zainol Ariffin K.A. Recurrent neural network for malware detection // *International Journal of Advances in Soft Computing and its Applications*. 2019. Vol. 11, No. 1. P. 46–63.
9. Maniriho P., Mahmood A.N., Chowdhury M.J.M. MeMalDet: A memory analysis-based malware detection framework using deep autoencoders and stacked ensemble under temporal evaluations // *Computers & Security*. 2024. Vol. 142. DOI: 10.1016/j.cose.2024.103864.
10. Jin X., Xing X., Elahi H., Wang G., Jiang H. A malware detection approach using malware images and autoencoders // *IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. Delhi, India, 2020. P. 1–6. DOI: 10.1109/MASS50613.2020.00009.
11. Panchagnula V.M., Ch. Satya Keerthi N.V.L., Surekha S., Sujatha R., Veeraiah D., Ramesh E., Lakshmi B. A deep learning approach for detecting malware using autoencoder // *IAENG International Journal of Computer Science*. 2024. Vol. 51, No. 8. P. 1051–1059.
12. Cassavia, N., Caviglione, L., Guarascio, M. *et al.* Learning autoencoder ensembles for detecting malware hidden communications in IoT ecosystems. *J Intell Inf Syst*, 2024. Vol. 62. p. 925–949. https://doi.org/10.1007/s10844-023-00819-8.